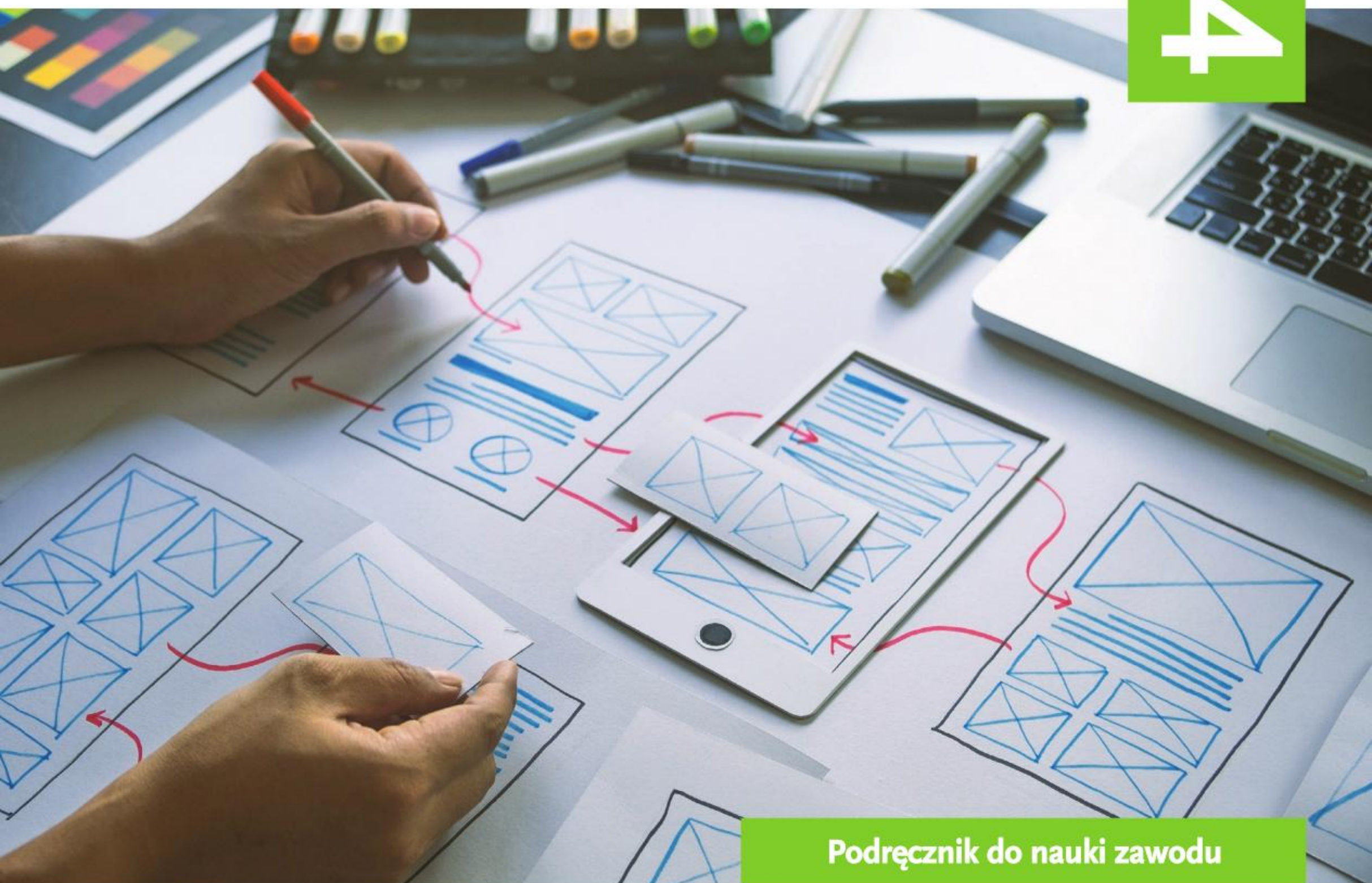


REFORMA
2019

Część 1

Projektowanie, programowanie i testowanie aplikacji

INF.04



Podręcznik do nauki zawodu

TECHNIK PROGRAMISTA

Autor: **Dariusz Gołębiowski**

Typ szkoły: **technikum**

Zawód: **technik programista**

Kwalifikacja: **INF.04. Projektowanie, programowanie i testowanie aplikacji**

© Copyright by Wydawnictwa Szkolne i Pedagogiczne
Warszawa 2021

Wydanie I (2021)

ISBN 978-83-02-19666-9 (część 1) (podręcznik w postaci elektronicznej – flipbook)

ISBN 978-83-02-19668-3 (całość) (podręcznik w postaci elektronicznej – flipbook)

ISBN 978-83-02-19659-1 (część 1) (podręcznik w postaci papierowej)

ISBN 978-83-02-19661-4 (całość) (podręcznik w postaci papierowej)

Opracowanie merytoryczne i redakcyjne: **Tomasz Kompanowski, Sebastian Jabłoński**
(redakcja merytoryczna)

Redakcja językowa: **Elżbieta Michalak**

Redakcja techniczna: **Elżbieta Walczak**

Projekt okładki: **Dominik Krajewski**

Ilustracje: **Dominik Krajewski**

Skład i łamanie: **Ledor**

Wydawnictwa Szkolne i Pedagogiczne Spółka Akcyjna

00-807 Warszawa, Aleje Jerozolimskie 96

KRS 0000595068

Infolinia: 801 220 555

www.wsip.pl

Druk i oprawa: Drukarnia Pozkał

Publikacja, którą nabyłaś / nabyłeś, jest dziełem twórcy i wydawcy. Prosimy, abyś przestrzegała / przestrzegał praw, jakie im przysługują. Jej zawartość możesz udostępnić nieodpłatnie osobom bliskim lub osobiście znanym. Ale nie publikuj jej w internecie. Jeśli cytujesz jej fragmenty, nie zmieniaj ich treści i koniecznie zaznacz, czyje to dzieło. A kopiując jej część, rób to jedynie na użytek osobisty.

prawolubni


Szanujmy cudzą własność i prawo.

Więcej na www.legalnakultura.pl

Polska Izba Książki

I. Aplikacje internetowe – wprowadzenie	9
1. Aplikacja internetowa – technologie, rodzaje	10
1.1. Aplikacja internetowa	10
1.2. Backend – frontend	12
1.3. Technologie internetowe	13
1.4. Backend i frontend – jeden język programowania ES6 / JavaScript	20
2. Programowanie – podstawy wiedzy	25
2.1. Podstawowe pojęcia programistyczne	25
2.2. Tworzenie zmiennych	30
2.3. Formatowanie kodu – kod programu a porządek	31
2.4. Opisy poszczególnych bloków kodu	35
3. Projekt: Serwis Ogłoszeniowy Aplikacja Webowa (SOAW) – opis	41
3.1. Serwis Ogłoszeniowy Aplikacja Webowa (SOAW)	41
3.2. Responsywność aplikacji	44
3.3. Podstawowa funkcjonalność tworzonego projektu	46
3.4. Model biznesowy SOAW	49
II. Proste typy danych	51
4. Proste typy danych	52
4.1. Typ danych	52
4.2. Proste typy danych	53
4.3. Przykładowe różnice w językach programowania	55
5. Typy numeryczne	57
5.1. Podstawowe wyjaśnienia dotyczące typu numerycznego	57
5.2. Przegląd typów numerycznych w różnych językach programowania	58
6. Typ logiczny	61
6.1. Definicja typu logicznego	61
6.2. Wykorzystanie w programowaniu	62
6.3. Flaga w programowaniu	63
7. Typy znakowe i łańcuchowe	66
7.1. Wyjaśnienie pojęć znaku i łańcucha	66
7.2. Cudzysłów a apostrof	67
7.3. Konkatenacja	69
7.4. Typy oparte na relacyjnej bazie danych SQL	71
7.5. Projekt SOAW – budowa bazy	72
8. Inne podstawowe typy danych w programowaniu	79
8.1. Typy danych – podsumowanie	79
8.2. Kolejne ważne typy danych	81
8.3. ASCII a UNICODE	82

III. Złożone typy danych 85

9.	Złożone typy danych – charakterystyka	86
9.1.	Najważniejsze złożone typy danych	86
10.	Tablice jednowymiarowe i dwuwymiarowe	89
10.1.	Tablice	89
10.2.	Tablice jednowymiarowe	90
10.3.	Tablice dwuwymiarowe	91
11.	Tablice dynamiczne i asocjacyjne	94
11.1.	Tablice dynamiczne	94
11.2.	Tablice asocjacyjne	95
12.	Typy rekordowe	97
12.1.	Struct	97
12.2.	Unia	99
13.	Wskaźniki	101
13.1.	Rola wskaźników	101
14.	Typy plikowe	104
14.1.	Zagadnienia	104
14.2.	Czytanie z pliku	104
14.3.	Zapis do pliku	109
15.	Kolekcje	112
15.1.	Zagadnienia	112
15.2.	Wektory	112
15.3.	Listy	114
15.4.	Stosy	115
15.5.	Kolejki	116
16.	Algorytmy – wprowadzenie	120
16.1.	Algorytm	120
16.2.	Schemat blokowy	121
16.3.	Lista kroków	123
16.4.	Drzewo decyzyjne	123
16.5.	Pseudokod	124
17.	Algorytmy iteracyjne, tekstowe, szyfrowania	125
17.1.	Algorytmy iteracyjne	125
17.2.	Algorytmy tekstowe	127
17.3.	Algorytmy szyfrowania	129
18.	Algorytmy tablicowe, rekurencyjne	132
18.1.	Algorytmy tablicowe	132
18.2.	Algorytmy rekurencyjne	135

IV. Narzędzia do planowania projektu, zarządzania nim i stosowane metodologie 141

19.	Narzędzia do planowania i zarządzania projektem i stosowane metodologie	142
19.1.	Zarządzanie projektem – wprowadzenie	142
19.2.	Zasady dobierania narzędzi do planowania projektu i zarządzania nim	144
19.3.	Cykl życia projektu: filozofia SCRUM – Agile	145

20.	Zastosowanie diagramów do zarządzania etapami projektu, zadaniami i czasem	152
20.1.	Wizualizacja	152
20.2.	Diagramy do zarządzania – przykłady	152
21.	Programy do wspierania zarządzania projektami i systemy kontroli wersji	155
21.1.	Funkcje programów do wspierania zarządzania projektami	155
21.2.	Przykładowe oprogramowanie wspierające zarządzanie projektami	156
21.3.	Systemy kontroli wersji na podstawie Git.	158

V. Projektowanie aplikacji 167

22.	SOAW: analiza wymagań klienta	168
22.1.	Profesjonalna obsługa klienta	168
22.2.	Analiza wymagań klienta	170
22.3.	Przykładowe narzędzia analityczne	171
22.4.	Specyfikacja projektu w programowaniu	174
22.5.	Paradygmaty programowania	175
23.	Projektowanie aplikacji klient–serwer – podstawowa specyfikacja projektu SOAW	180
23.1.	Filozofia Agile	180
23.2.	Podstawowa specyfikacja projektu SOAW	182
23.3.	Projektowanie interfejsu użytkownika i wyglądu aplikacji	184
23.4.	Dostosowanie interfejsu do różnych platform sprzętowych	184
24.	Aplikacja SOAW: interfejs, wygląd, funkcjonalność	185
24.1.	Interfejs aplikacji	185
24.2.	Wygląd aplikacji na przykładzie SOAW	192
24.3.	Funkcjonalność aplikacji na bazie SOAW	195

VI. Planowanie przedsięwzięć programistycznych 199

25.	SOAW – wybór technologii i środowiska programistycznego	200
25.1.	Wybór technologii	200
25.2.	Technologie PHP, JAVASCRIPT, HTML 5, CSS, SQL	201
25.3.	Technologia – sprzęt IT	202
25.4.	Technologia – narzędzia informatyczne i zasoby osobowe	204
26.	SOAW: przedstawienie alternatywnego rozwiązania	210
26.1.	Dodatkowe technologie mogące uzupełnić zestaw PHP, JavaScript, HTML 5, CSS, SQL	210
26.2.	Alternatywne rozwiązanie dla wyglądu SOAW – biblioteki (frameworki) CSS	214
26.3.	Alternatywne rozwiązanie dla budowy SOAW – node.js, React.js, Angular, Vue.js ...	215
27.	Plan prac programistycznych nad SOAW – etapy realizacji i harmonogram projektu	222
27.1.	Nowoczesne technologie – alternatywne planowanie projektu biznesowego	222
27.2.	Lean Canvas – szybkie i kompleksowe planowanie projektu technologicznego	222
27.3.	Kanban – przedstawienie zadań za pomocą tablicy	224
27.4.	Plan prac programistycznych: etapy realizacji i harmonogram projektu	226
28.	Zasoby ludzkie do realizacji SOAW	229
28.1.	Analiza projektu uwzględniająca potrzeby technologiczne	229
28.2.	Dobór kadry – „człowiek orkiestra” czy specjalista ukierunkowany na daną technologię	230
28.3.	Pozostała kadra projektowa	231

29.	Zarządzanie projektem	233
29.1.	Zarządzanie projektem w projektach technologicznych	233
29.2.	Zarządzanie personelem w projektach technologicznych	233
29.3.	Zarządzanie kontaktami z klientami	234

VII. Programowanie zaawansowanych aplikacji webowych 237

30.	Omówienie wyglądu aplikacji SOAW – HTML, CSS	238
30.1.	Wykorzystywane technologie – dodatkowe elementy	238
30.2.	Strona tytułowa – wygląd	242
30.3.	Widok mobilny – Viewport	243
30.4.	Programowanie strony tytułowej – HTML, CSS, JAVASCRIPT, PHP	245
30.5.	Header	246
30.6.	Content – zawartość główna	252
30.7.	Footer – stopka aplikacji	253
31.	SOAW – system logowania jako element serwisu ogłoszeniowego (logika i bezpieczeństwo aplikacji)	256
31.1.	SOAW – krótkie podsumowanie dotychczasowych prac	256
31.2.	Dostęp do portalu – logika logowania	257
31.3.	Bezpieczeństwo aplikacji – planowanie systemu zabezpieczeń	258
32.	SOAW: przygotowanie systemu logowania – HTML i PHP	263
32.1.	Katalog templates – początek zmian w HTML	263
32.2.	PHP – dołączenie zawartości jednego pliku do wnętrza innego pliku	265
32.3.	Logowanie php – plik login.php	266
32.4.	Plik index.php a login.php – przepływ danych	269
32.5.	Testowanie PHP	270
32.6.	HTML i PHP – adres e-mail, logika ‘and’	272
32.7.	Skrypty PHP wewnątrz HTML	274
33.	SOAW: programowanie systemu logowania – PHP	276
33.1.	Uruchomienie sesji PHP	276
33.2.	Flaga w programowaniu	277
33.3.	Logowanie PHP – sanityzacja loginu	278
33.4.	Walidacja loginu (adresu e-mail)	279
33.5.	Walidacja hasła	281
34.	SOAW – obiektowy dostęp do bazy danych z wykorzystaniem PDO	288
34.1.	Konfiguracja środowiska połączeniowego	288
34.2.	Podstawy PDO (PHP Data Objects)	294
34.3.	Połączenie z bazą danych z wykorzystaniem PDO – podstawy teoretyczne	297
34.4.	Testowanie połączenia z bazą danych z wykorzystaniem PDO	299
34.5.	Sprawdzenie istnienia odpowiednich danych – loginu i hasła w bazie z wykorzystaniem PDO	303
34.6.	Instrukcja try-catch PHP – obsługa błędów	305
34.7.	query() – metoda PDO	308
34.8.	Sprawdzenie poprawności hasła	312
34.9.	Metody PDO: prepare(), execute(), fetch() i symbole zastępcze =?	313

35.	Interakcja w aplikacji SOAW – „ciasteczkowy projekt”, kodowanie vanilla JavaScript	319
35.1.	Ciasteczka, RODO – podstawy	319
35.2.	Dodawanie skryptów JavaScript	321
35.3.	Zdarzenia na stronie internetowej	324
35.4.	JavaScript – kodowanie ciasteczka	328
35.5.	JavaScript – kodowanie logiki wyświetlania ciasteczek	336
36.	SOAW: interakcja w aplikacji – przykłady kodu, biblioteka jQuery	343
36.1.	Wprowadzenie do biblioteki jQuery	343
36.2.	Przykłady składni jQuery	347
36.3.	Projekt menu rozwijanego – kod HTML	348
36.4.	Projekt menu rozwijanego – kod CSS	350
36.5.	Projekt menu rozwijanego – kod jQuery	352
	Dodatek	355
	Projekt SOAW – kody PHP	356
	Netografia	369
	Wykaz podstawowych pojęć w językach: polskim, angielskim i niemieckim	371

I. Aplikacje internetowe – wprowadzenie

- Aplikacja internetowa – technologie, rodzaje
- Programowanie – podstawy wiedzy
- Projekt: Serwis Ogłoszeniowy Aplikacja Webowa (SOAW) – opis

1

Aplikacja internetowa – technologie, rodzaje

ZAGADNIENIA

- Aplikacja internetowa
- Backend – frontend
- Technologie internetowe
- Przykłady aplikacji internetowych

1.1. Aplikacja internetowa

Pod terminem „**aplikacja internetowa**” (dosyć często określanej również jako aplikacja webowa, ang. *web application*) kryje się szeroki zakres różnego rodzaju programów (aplikacji) komputerowych. Mają one cechy wspólne, takie jak:

- backend – elementy aplikacji, które „pracują” na serwerze;
- frontend – interfejs użytkownika widoczny w przeglądarce internetowej, będący interaktywnym klientem aplikacji internetowej;
- komunikacja przez sieć internetową (komputerową) backendu z hostem użytkownika komputera z wykorzystaniem przeglądarki internetowej użytkownika – frontend.

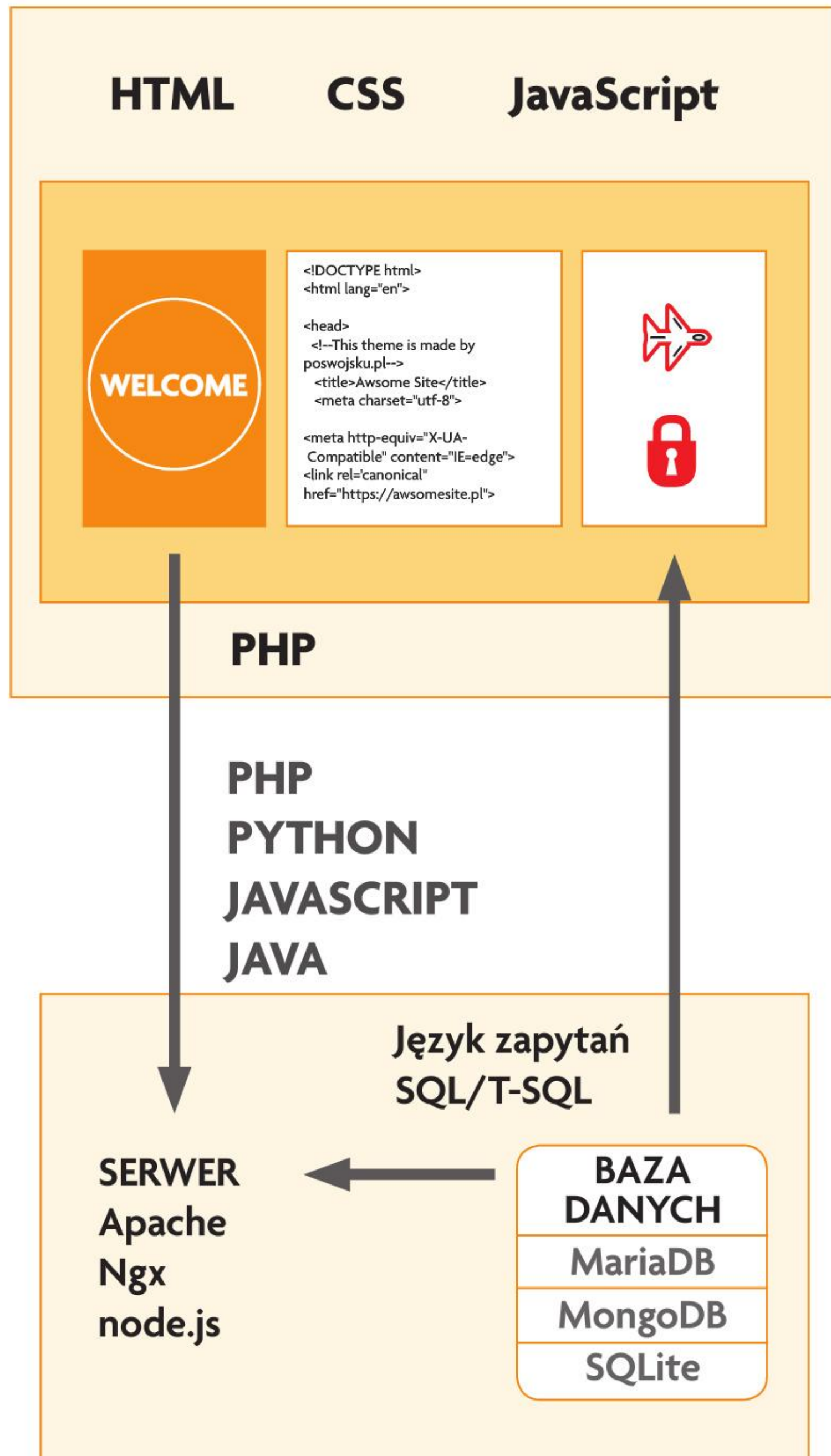
W pracy aplikacji webowej musi pośredniczyć serwer WWW. Do zaprojektowania i zaprogramowania aplikacji używa się różnych mechanizmów – tzw. środowisk programistycznych.

admin ▾					
LOGO HI		Logout			
ID	name	email	password	change	
1	admin	admin@pl	secret!	change	
2	Darek	darek@pl	secret!	change	

Rys. 1.1. Przykład aplikacji internetowej – system CRM

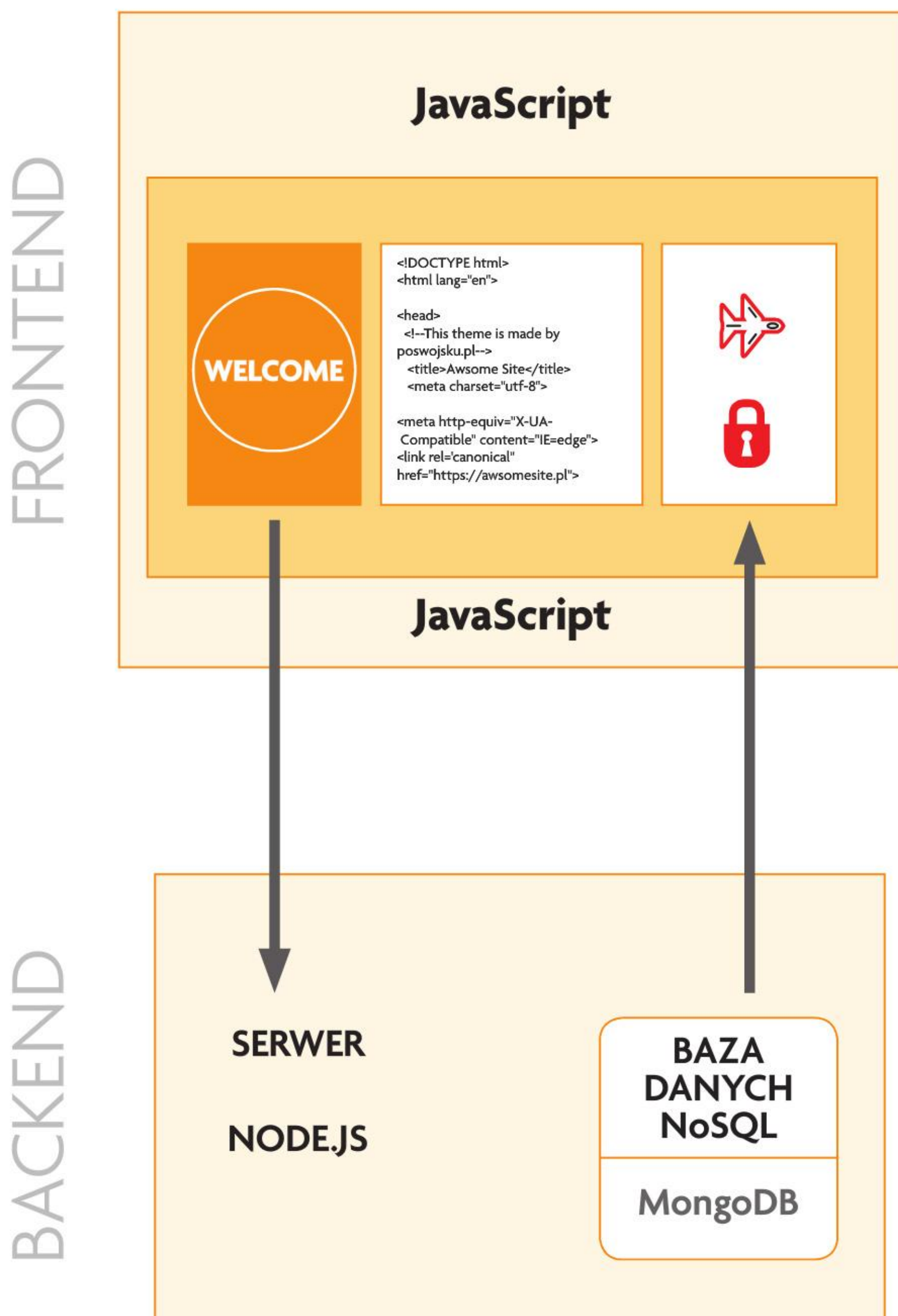
FRONTEND

BACKEND



Rys. 1.2. Grafika pogładowa – aplikacja internetowa

1.2. Backend – frontend



Rys. 1.3. Język programowania JavaScript zarówno na froncie, jak i na zapleczu technologii internetowych

Przykładowe technologie dla:

- **frontendu:** HTML, CSS, PHP, JavaScript, frameworki CSS, frameworki JavaScript, frameworki PHP;
- **backendu:** Apache, MariaDB (baza SQL), język SQL, PHP, phpMyAdmin, CGI, PYTHON, JSP, ASP.NET, Java, C#, .netCore, MongoDB, Postgres, MySQL, MS Access, node.js.

Pamiętaj, że wskazany wykaz języków programowania i technologii wspierających nie jest pełny. Jeśli zerkniesz do internetu, to z pewnością znajdziesz co najmniej dziesiątki języków programowania wykorzystywanych zarówno do backendu, jak i frontendu. Co ważne, najnowsze technologie są dużo bardziej skomplikowane niż przedstawiony podział. Jak widać, PHP wpisano zarówno w obszarze strony WWW (frontend), jak i zaplecza projektu (backend). Warto wiedzieć, że skrypty PHP, co prawda, wpisuje się w kody tworzące strony internetowe (frontend), ale ten język jest wykonywany na serwerze, a więc na backendzie. Oznacza to, że PHP jest łącznikiem między stroną WWW a zapleczem bazodanowym projektu.

Nie tylko PHP może działać po obu stronach technologii internetowych. Podobnie jest z wieloma innymi językami programowania. Prawdopodobnie najbardziej uniwersalna jest pod tym względem JavaScript. Ten język programowania ma tak wielkie możliwości, że teoretycznie całą aplikację webową można stworzyć (prawie) bez użycia innej technologii. Nawet serwer JavaScript ma własny język – node.js. – czyli JavaScript na froncie i na zapleczu.

1.3. Technologie internetowe

Świat internetu i technologii z nim związanych (nazywanych też internetowymi, webowymi lub WWW) od samego początku charakteryzował się szybkim i bardzo nieukierunkowanym rozwojem. Olbrzymie firmy, instytucje, a także różnego rodzaju organizacje tworzyły własne standardy technologii internetowych. Wiele z nich wzajemnie się wykluczało, a na rynku trwała wojna przeglądarek. Na szczęście możemy o tym (w zdecydowanej większości przypadków) mówić jako o czasie przeszłym. Organizacja The World Wide Web Consortium (W3C, <http://www.w3.org/>) zajmuje się tworzeniem standardów dla sieci Internet. Na jej stronach znajdziesz niezbędne informacje o wielu technologiach internetowych. Warto zacytować jedno zdanie zamieszczone na jej stronie internetowej: „The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards”¹.

¹ „Konsorcjum World Wide Web (W3C) to międzynarodowa społeczność, w której organizacje członkowskie, personel zatrudniony w pełnym wymiarze czasu pracy i opinia publiczna wspólnie opracowują standardy sieciowe”. Źródło: W3C, <http://www.w3.org/> [dostęp: 21.03.2021].



Rys. 1.4. Grafika poglądowa – skrócone (popularne w użyciu) nazwy jednych z najważniejszych technologii internetowych

Wykaz standardów sieciowych opracowywanych przez tę instytucję znajdziesz na stronie <https://www.w3.org/standards/>. Umieszczono na niej ciekawe zdanie: „Projektowanie i aplikacje internetowe obejmują standardy tworzenia i renderowania stron internetowych, w tym HTML, CSS, SVG, interfejsy API urządzeń i inne technologie dla aplikacji internetowych (WebApps). Ta sekcja zawiera również informacje o tym, jak udostępnić strony osobom niepełnosprawnym (WCAG), umiędzynarodowić je i sprawić, by działały na urządzeniach mobilnych”.

HTML

HyperText Markup Language to język programowania (choć niektórzy twierdzą, że język znaczników nie jest językiem programowania) oparty na znacznikach (tagach HTML), przeznaczony do tworzenia struktury stron internetowych. Dzięki niemu możemy panować nad układem i strukturą strony WWW.

CSS

Kaskadowe arkusze stylów (ang. *Cascading Style Sheets*) to język służący do opisu wyglądu strony internetowej. CSS umożliwia zmianę wyglądu elementów struktury HTML i ich zawartości (np. czcionek, kolorów, tła, odnośników, marginesów, pozycji poszczególnych elementów względem okna przeglądarki). Od wersji CSS3 możemy w tej technologii także wykonywać animacje na stronie WWW.



CIEKAWOSTKA

Na rys. 1.5 przedstawiono kod języka CSS, dzięki któremu określono wygląd strony WWW, wyświetlanej na urządzeniach mających ekran o rozdzielczości równej bądź poniżej 768 px. Zostają ustalone właściwości dla dwóch klas:

- `.introDG` – tekst: wielkość, pogrubienie, centrowanie i margines wewnętrzny;
- `.nawigacjaHamburger` – tekst: wielkość, brak pogrubienia, kolor czcionki; margines wewnętrzny; blok – wyrównanie, szerokość i sposób wyświetlania.


```
/* style dla urządzeń mobilnych */

@media screen and (max-width: 768px) {
    .introDG {
        font-weight: bold;
        font-size: 1.4em;
        padding: 6% 0 0 0;
        text-align: center;
    }

    .nawigacjaHamburger {
        font-size: 1em;
        font-weight: normal;
        color: #fff;
        padding: 0;
        text-align: left;
        width: 100%;
        display: inline;
    }
}
```

Rys. 1.5. CSS – przykład arkusza stylów – zapytanie o media

JavaScript

JavaScript (**JS**) to skryptowy język programowania, pierwotnie stworzony do stosowania głównie w technologiach internetowych. W ostatnich latach przeszedł rewolucję i stał się jedną z najbardziej popularnych technologii. Kiedyś skrypty JS służyły najczęściej zapewnieniu interaktywności – obsługiwały reagowanie na zdarzenia i sprawdzanie poprawności formularzy i/lub budowanie elementów nawigacyjnych. Obecnie technologię JavaScript można znaleźć niemal wszędzie, w tym po stronie serwera – node.js. O tym języku dowiesz się więcej w dalszej części podręcznika.

PHP

Jest to obiektowy, skryptowy język programowania, który zaprojektowano do generowania stron internetowych w czasie rzeczywistym. Skryptów PHP używa się do połączenia technologii widocznych dla internautów (stron WWW) z zapleczem, czyli m.in. z serwerem internetowym i bazą danych. Użycie PHP znacznie przyspiesza pracę nawet nad prostymi stronami internetowymi.

SQL – język programowania

Structured Query Language (SQL) to język używany do tworzenia baz danych i zarządzania ich zawartością (odczytywanie, tworzenie, modyfikacja, usuwanie). Stanowi standard w komunikacji z serwerami relacyjnych baz danych (np. MariaDB, MySQL, Postgres, SQLite).



CIEKAWOSTKA

Na rys. 1.6 przedstawiono kod języka SQL, dzięki któremu w bazie danych '**arts**' zostanie dodana tabela '**managers**' (o ile już nie istnieje). Stworzona tabela będzie zawierała kolumny o nagłówkach:

- id_managers,
- loginm,
- haslo.

```
-- -----  
-- Table `arts`.`managers` - Tabela 6. managers  
-- -----  
CREATE TABLE IF NOT EXISTS `arts`.`managers` (  
  `id_managers` INT NOT NULL AUTO_INCREMENT COMMENT 'Klucz  
główny - jest przydzielony automatycznie',  
  `loginm` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE  
  'utf8mb4_unicode_ci' NOT NULL COMMENT 'Nazwa potrzebna przy  
logowaniu',  
  `haslo` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE  
  'utf8mb4_unicode_ci' NOT NULL COMMENT 'Hasło zahasowane',  
  PRIMARY KEY (`id_managers`))  
ENGINE = InnoDB  
COMMENT = 'Posiada informacje o osobach zarejestrowanych w  
programie.';
```

Rys. 1.6. Przykład kodu SQL

SQL – relacyjne bazy danych

SQL to pojęcie odnoszące się także do relacyjnych baz danych.

Relacyjna baza danych jest opisanym i zorganizowanym zbiorem tabel, które są połączone relacjami, czyli związkami między sobą. Tabele, nazywane także **encjami**, składają się z:

- wierszy – określanych także jako rekordy lub krotki;
- pól (komórek) przechowujących jedną daną – z nich są zbudowane rekordy;
- kolumn tabeli, zawierających atrybuty danych.

Każdy rekord zawiera zwykle wartość dla każdego atrybutu, co ułatwia ustalenie relacji między poszczególnymi elementami rekordu.

Są to jedne z najbardziej rozpowszechnionych rodzajów baz danych. Najczęściej wykorzystywane to m.in.:

- SQLite – strona projektu: <https://sqlite.org/index.html>;
- MariaDB – strona projektu: <https://mariadb.org/> (ta baza danych znajduje się wewnątrz pakietu XAMPP, który będziemy wykorzystywali w tym podręczniku do budowy aplikacji);
- PostgreSQL – strona projektu: <https://www.postgresql.org/>;
- MySQL – strona projektu: <https://www.mysql.com/>;
- Oracle – strona projektu: <https://www.oracle.com/database/technologies/>;
- Access – strona projektu: <https://www.microsoft.com/pl-pl/microsoft-365/access>.

Warto wiedzieć, że jedynie pierwsze trzy rozwiązania są projektami bezpłatnymi i można je bez żadnych obaw wykorzystywać do budowy każdego rodzaju aplikacji.

UWAGA

Relacyjne bazy danych to tylko jeden z rodzajów baz danych. Równie popularne są nie-relacyjne bazy danych, określane jako **NoSQL**. Należy podkreślić to, że są one w stanie obsłużyć ogromne ilości bardzo szybko zmieniających się danych, które są nieustrukturyzowane. Wykonują to innymi sposobami niż relacyjna baza danych z wierszami i tabelami. Jednymi z podstawowych narzędzi umożliwiających współpracę z tymi rodzajami baz danych są JavaScript i JSON (strona projektu: <https://www.json.org/json-en.html>, format przechowywania i przesyłania danych). Warto zajrzeć na strony kilku projektów NoSQL:

- <https://www.mongodb.com/>,
- <https://www.arangodb.com/>,
- <https://www.couchbase.com/>,
- <https://couchdb.apache.org/>.

Opisane technologie nie oddają bogactwa i różnorodności w świecie technologii internetowych, ale wskazują, o czym głównie będzie mowa w tym podręczniku. Pamiętaj, że zawsze, gdy będzie ci brakowało wiedzy z zakresu technologii internetowych, możesz zajrzeć do materiałów źródłowych standaryzujących internet, czyli na stronę: <https://www.w3.org/>.

Backend – serwer – node.js

Serwerem służącym do postawienia aplikacji webowej może być także node.js, które jest środowiskiem uruchomieniowym wykonującym kod JavaScript poza przeglądarką internetową. Node.js daje bardzo duże możliwości programistom znającym JavaScript. Mogą w tym języku pisać narzędzia wiersza poleceń i wykonywania skryptów po stronie serwera node.js. W ten sposób, dzięki uruchamianiu skryptów po stronie serwera, powstaje dynamiczna zawartość strony internetowej. Dopiero po stworzeniu zawartości stronę WWW wysyła się do przeglądarki internetowej użytkownika. Node.js umożliwia realizację postulatu programistów posługujących się wspomnianym językiem: „JavaScript wszędzie”.



Rys. 1.7. Skrócona nazwa języka JavaScript

Takie podejście ma co najmniej dwie ważne zalety:

1. Jeden język programowania do tworzenia aplikacji internetowych (frontend i backend) zamiast różnych języków dla skryptów po stronie serwera i po stronie klienta. Oznacza to, że do kompleksowego stworzenia aplikacji może wystarczyć tylko jeden programista znający jeden język programowania – JavaScript.
2. Aplikacje oparte na języku JavaScript (oraz jego frameworkach typu: React, Angular, Vue) są bardzo szybkie, skalowalne i stabilne. Zapewne właśnie dlatego technologia javascriptowa opanowała świat wielkich portali społecznościowych.

Warto dodać, że zdecydowana większość rozwiązań opartych na języku programowania JavaScript jest bezpłatna i rozwijana w ramach *open source* (wolne oprogramowanie). Technologia node.js również jest wolna od kosztów związanych z jej używaniem. Szczegóły dotyczące node.js można znaleźć na oficjalnej stronie projektu: <https://nodejs.org/>.

 CIEKAWOSTKA

```
{
  "name": "nazwa",
  "version": "0.1.4",
  "description": "opis projektu",
  "author": "autor projektu",
  "license": "rodzaj licencji",
  "scripts": {
    "start": "webpack-dev-server --progress",
    "build": "NODE_ENV=production
node_modules/.bin/webpack -p --progress
--colors"
  },
  "devDependencies": {
    "babel-core": "~6.7.*",
    "babel-loader": "~6.2.*",
    "babel-preset-es2015": "~6.6.*",
    "babel-preset-react": "~6.5.*",
    "webpack": "~1.12.*",
    "webpack-dev-server": "~1.14.*"
  },
  "dependencies": {
    "marked": "^0.3.6",
    "react": "^15.0.0",
    "react-dom": "^15.0.0"
  }
}
```

Rys. 1.8. Widok pliku kodu przykładowego pliku package.json – środowisko node.js

W projektach programistycznych wykorzystujących język JavaScript w wersji ES6 (i wyższej) zwykle w katalogu głównym projektu znajduje się plik package.json (na rys. 1.8 znajdziesz przykład tego rodzaju pliku). Jego przeznaczeniem jest udzielanie informacji pozwalającej na identyfikację projektu, a także na obsługę jego zależności (**devDependencies** i **dependencies**). Może również zawierać inne metadane, takie jak:

- nazwa projektu,
- wersja projektu w określonej dystrybucji,
- opis projektu,
- nazwisko / pseudonim autora projektu,
- informacje o licencji,
- dane konfiguracyjne.

1.4. Backend i frontend – jeden język programowania

ES6 / JavaScript

JavaScript ewoluuje i jego wersje serwerowe określa się jako **ECMAScript**. W 1996 r. organizacja ECMA rozpoczęła pracę nad nową, ujednoliconą specyfikacją języka JavaScript, która byłaby pozbawiona odniesień dotyczących interpretującego kodu środowiska. Język opisany w standardzie ECMA-262 nazwano ECMAScript. Od tego momentu nazwa JavaScript oznacza jedynie jeden z nadzbiorów języka ECMAScript. Najbardziej popularne określenie w świecie programistów JavaScript to **ES6**. Jest to oficjalna nazwa pierwszej rewolucyjnej wersji JavaScript – ECMAScript2015.

UWAGA

Pod tym adresem znajdziesz specyfikację języka ECMAScript:

<http://www.ecma-international.org/publications/standards/Ecma-262.htm>.

Przykłady aplikacji internetowych

Przykłady aplikacji internetowych – technologie:

- portale społecznościowe (np. Facebook, LinkedIn);
- sklepy internetowe (Amazon);
- serwisy aukcyjne (Allegro);
- systemy CMS (np. Wordpress, Drupal, Joomla!);
- sklepy internetowe (np. Woocommerce, PrestaShop, Magento);
- portale elearningowe (np. Moodle);
- portale użytkowe – rządowe: <https://www.gov.pl/>, <https://pz.gov.pl>, <https://prod.ceidg.gov.pl/>, <https://ekrs.ms.gov.pl/>, <https://mapy.geoportal.gov.pl>, <https://www.bip.gov.pl/>, <https://rcl.gov.pl/>, <https://login.e-dowod.gov.pl> itp.;
- portale użytkowe – unijne: <https://europa.eu>, <https://ec.europa.eu>, <https://www.europarl.europa.eu>, <https://www.consilium.europa.eu>, <https://www.eesc.europa.eu>.

CIEKAWOSTKA

Zgodnie z informacjami zawartymi na oficjalnej stronie ECMA:

„Ecma International jest stowarzyszeniem branżowym zajmującym się standaryzacją systemów informacyjnych i komunikacyjnych” (źródło: <https://www.ecma-international.org/>). Organizacja ma pod swoją opieką rozwój wielu technologii, w tym tych wykorzystywanych w świecie aplikacji webowych, m.in.:

- ECMA-404 – składnia wymiany danych JSON (więcej informacji na stronie: <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>);
- ECMA-262 – EcmaScript® 2020, specyfikacja języka (więcej informacji na stronie: <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>);
- ECMA-408 – specyfikacja języka programowania Dart (więcej informacji na stronie: <https://www.ecma-international.org/publications-and-standards/standards/ecma-408/>).



CIEKAWOSTKA

Na rys. 1.9 znajdziesz przykład programowania obiektowego, wykonanego w języku programowania JavaScript w wersji ES6.

```

es6Script.js

class TodoClass {
  constructor() {
    this.zadania = JSON.parse(localStorage.getItem('ZADANIA'));
    if(!this.zadania){
      this.zadania = [
        { task: 'Nauka JavaScript', isComplete: false },
        { task: 'Nauka PHP', isComplete: true },
        { task: 'Zrób projekt SOAW!', isComplete: false },
      ];
    }
    this.loadTasks();
    this.addEventListener();
  }

  addEventListener(){
    document.getElementById('addTask').addEventListener('keypress', event => {
      if(event.keyCode === 13 ){
        this.addTask(event.target.value);
        event.target.value = "";
      }
    });
  }
}

```

Rys. 1.9. Widok pliku z przykładowym kodem – standard ES6

TASKS

+ Add New

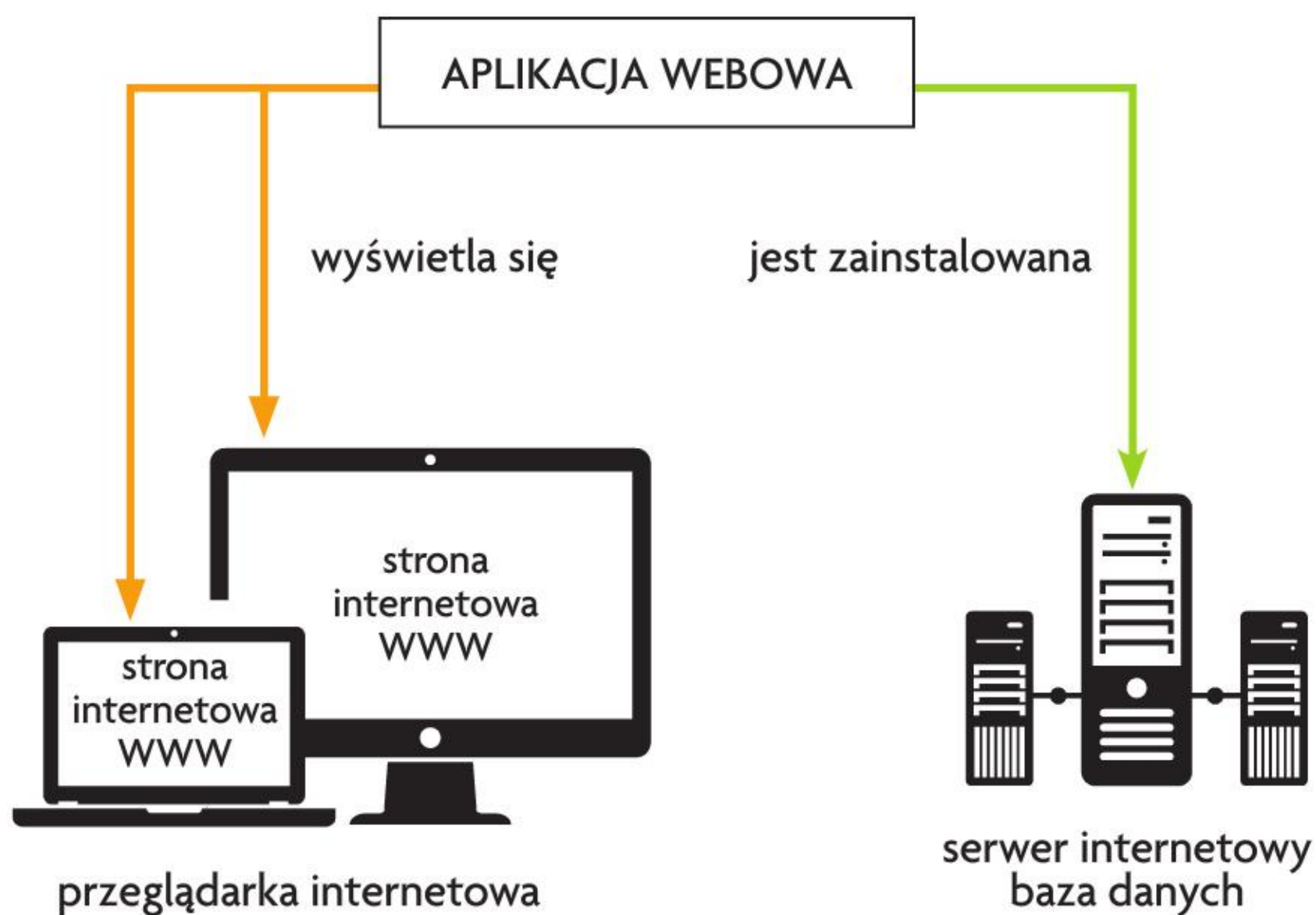
Search

Recipient's text

Search

User Email	Task Name	Task Description	first name, surname, position, email	more edit delete
dfgdhgh	dghdgh	dghdghdg	dghdgh	details edit delete
admin@pl	Buy CRMdg :)	Why should you buy Premium CRMdg? It's simple: huge development opportunities	???	details edit delete
darek@pl	Start TODAY! :) - cooperation with GDDM	Start cooperation with GDDM in the field of programming web applications.	???	details edit delete
admin@pl	Give: Like!	Give LIKE! or many LIKES :) on social portals	done	details edit delete

Rys. 1.10. Przykładowy system CRM – widok karty Tasks



Rys. 1.11. Aplikacje webowe

Co ważne, zatarła się różnica między stroną internetową (portalem tematycznym) a aplikacją. Przykładowo: polskie strony rządowe bądź unijne są zarówno zwykłymi stronami WWW, jak i wirtualnym miejscem, w którym można załatwić wiele spraw niezbędnych w codziennym życiu.



Rys. 1.12. Aplikacje mobilne



SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest aplikacja internetowa?
2. Podaj przykłady aplikacji internetowych.
3. Wyjaśnij pojęcia backendu i frontendu.
4. Jakie technologie mogą wchodzić w zakres obszaru backend?
5. Jakie technologie mogą wchodzić w zakres obszaru frontend?
6. Czym różni się język znaczników HTML od języka stylu CSS?
7. Czym różni się język JavaScript od PHP?
8. Jaka jest różnica między aplikacją webową a aplikacją mobilną?



SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. W tym rozdziale otrzymałeś ogólne informacje na temat relacyjnych i nierelacyjnych baz danych. Poszukaj w internecie i wypisz co najmniej trzy podstawowe różnice między wymienionymi rodzajami przechowywania danych.

2

Programowanie – podstawy wiedzy

ZAGADNIENIA

- Podstawowe pojęcia programistyczne
- Programowanie
- Algorytm
- Zmienna
- Tablica
- Funkcja
- Formatowanie kodu – kod programu a porządek

2.1. Podstawowe pojęcia programistyczne

Teoria programowania to bardzo obszerna dziedzina wiedzy zapisana w opasłych tomach materiałów edukacyjnych. Potrzebujemy jednak tylko podstawy teorii. Na początek warto poznać kilka pojęć.



Rys. 2.1. HTML 5

Programowanie

Jest to bardzo ciekawe zajęcie, w którym osoba programująca, czyli programista, jest skazana na ciągłe rozwiązywanie zagadek i różnorodnych problemów. Wszystko powinna wykonywać w kreatywny sposób. Pisanie kodu to często żmudna i czasochłonna praca, wymagająca dużego zaangażowania i ciągłej nauki. Technologia się zmienia, a jednym z najszybciej rozwijających się obszarów są technologie webowe.

Programowanie polega na tworzeniu kodu programistycznego, który umożliwia „rozmowę” z komputerem i systemami informatycznymi. Wykorzystuje się w nim zarówno

algorytmy, jak i zmienne. Programowanie w świecie informatyki opiera się na językach programowania, czyli na określonych (narzuconych) specyfikacjach – sposobach pisania kodu. W tym podręczniku poznamy podstawy kilku potężnych i powszechnie używanych języków programowania: PHP, HTML, CSS, SQL, JavaScript.



Rys. 2.2. CSS 3

Aby uzyskać kompleksowy obraz świata programowania, spojrzymy także na języki tradycyjne (złośliwi mawiają – stare) programowania – np. C++.

Algorytm

Jest to ciąg jednoznacznie zdefiniowanych czynności (działań, instrukcji) niezbędnych do wykonania określonego rodzaju zadania i/lub zadań. Można stwierdzić, że polega on na zmianie stanu, czyli przeprowadzeniu czegoś określonego z pewnego stanu początkowego do (zwykle wcześniej zaplanowanego) pożądanego stanu końcowego.

Algorytm można przedstawić na schemacie blokowym. W tym konkretnym przypadku (rys. 2.3) zwierzę przejdzie ze stanu, w którym jest głodne, do stanu, w którym np. wypije mleko.



Rys. 2.3. Przykład prostego algorytmu

W codziennym życiu niemal bez przerwy wykonujesz określone algorytmy. Budzisz się rano, a potem wykonujesz algorytm mycia zębów. Na śniadanie robisz kanapkę także według określonego algorytmu. Wsiadasz do samochodu, autobusu, tramwaju i jedziesz według określonego algorytmu. Masz zajęcia w szkole – według określonego algorytmu. I tak dalej, aż do nocy. Cały twój dzień można określić mianem algorytmu, składającego się z wielu drobniejszych (jak już zapewne zauważyłeś) algorytmów.

Jednym z najbardziej popularnych sformułowań w programowaniu jest powiedzenie (niektórzy programiści twierdzą, że jest to nawet aksjomat): „dziel i rządź”. Przekładając na nasz język: zadanie, które otrzymasz, podziel na małe, jednorodne elementy. Oprogramuj każdy z osobna, a potem połącz. Tego rodzaju czynności są powszechnie wykorzystywane w praktyce programowania. Zapewne wiele razy spotkasz się z opisanym podejściem programistycznym.

CIEKAWOSTKA

Na rys. 2.4 znajdziesz przykład wykorzystania zasady „dziel i rządź” w programowaniu obiektowym (język programowania JavaScript w wersji ES6+). Całość kodu podzielono na elementy odpowiedzialne za wykonanie określonych czynności. Każdą samodzielną funkcjonalność umieszczono w plikach: „Komponent1.js”, „Komponent2.js”, „Komponent3.js”, „Komponent4.js”. Następnie całość jest łączona w pliku głównym App.js. Takie podejście do budowy aplikacji jest charakterystyczne dla rozwiązań opartych na serwerze node.js.



```
JS App-oryginal.js
# App.css
JS App.js
JS App.test.js
# index.css
JS index.js
JS Komponent1.js
JS Komponent2.js
JS Komponent3.js
JS Komponent4.js
logo.svg
# styles.css
```

Rys. 2.4. Przykład zasady „dziel i rządź” – podział aplikacji na komponenty

Najważniejsze dla nas jest to: **program komputerowy (czyli także planowana w tym podręczniku aplikacja) składa się z algorytmów i zmiennych.**

Zmienna

Jest to pojemnik, który pomaga gromadzić różnorodne dane i układać je w odpowiedni sposób. **Zmienne** są różnego typu – charakterystyczne dla danego języka programowania. Zwykle typy danych dzieli się na proste (np. jeden znak bądź zwykły string, czyli ciąg znaków) i złożone (np. obiekt). Żeby było ciekawiej: string bądź number także może być obiektem (a nawet zapewne jest). Zostawmy jednak takie dylematy na zaawansowany poziom programowania w JavaScript. Póki co jesteśmy zdecydowanie początkującymi programistami.

CIEKAWOSTKA

Na rys. 2.5 znajdziesz przykład skryptu wykonanego w języku programowania JavaScript.

```
<script>
  var tablica = [1, 2, 3, 4, 5];

  var tablica2 = [...tablica];
  tablica.push("rr");
  console.log(...tablica);
  console.log(...tablica2);

  let obiekt = {
    a: 111,
    b: 2,
    c: 3
  } {
    qq,
    ww,
    ee
  } = {...obekt
};
</script>
```

Rys. 2.5. Przykładowe zmienne JavaScript

Przedstawiono na nim tworzenie zmiennych z wykorzystaniem słów kluczowych:

- **var** – najstarsza forma określania, że dana nazwa jest zmienną – wykorzystywana głównie w środowisku webowym;
- **let** – jest to element, który wszedł do użycia stosunkowo niedawno, najczęściej kojarzony z wersją ES6 – używany przede wszystkim w środowisku node.js i aplikacjach opracowywanych w technologii ES6+.

Ważne, abyś zrozumiał, że typ zmiennej musi być właściwy dla rodzaju przechowywanych danych. Z jednej strony musi umieć „przetrzymać” dane w sobie, z drugiej zaś nie może być ani zbyt duży, ani zbyt mały.

Wyobraź sobie plastikowy kubek – to pojemnik na twoje dane, czyli zmienna. A teraz konkretny przykład rodzaju danych – zastanów się nad przykładami przedstawionymi na następnej stronie.

CIEKAWOSTKA

Na rys. 2.6 znajdziesz przykład skryptu wykonanego w języku programowania C#. Przedstawiono na nim tworzenie zmiennych o typach:

- **char** – znakowy,
- **string** – ciąg znaków,
- **int** – liczby całkowite,
- **bool** logiczny,
- **decimal** – liczby dziesiętne.

```
// Przykładowe typy zmiennych języka  
programowania C#
```

```
char zmieChar = 'c';
```

```
string ciagTekstowy = "Witaj w C#";
```

```
int liczba = 69;
```

```
bool logiczny = true;
```

```
decimal dziesiętny = 0.23M;
```

Rys. 2.6. Przykładowe zmienne języka programowania C#

Co spowoduje zimna woda wlana do wspomnianego pojemnika? Jeżeli nie będzie jej więcej, niż może pomieścić kubek, to będzie można ją w tym pojemniku przechować. Jeżeli wody będzie więcej, tzn. gdy jej objętość będzie większa niż objętość kubka, wyleje się z niego.

Rozżarzona stal – spróbuj wlać ją do plastikowego kubka. Co się wydarzy? Kubek się roztopi, a stal wypłynie. Prosty wniosek: kubek w tym przypadku jest nieodpowiednim typem pojemnika do przechowywania danych.

Wyobraź sobie wielki kontener przeładunkowy. Jaką może mieć objętość? Zapewne zmieściłoby się do niego kilka samochodów osobowych. A teraz pomyśl, czy wykorzystywanie go do przechowywania małego pudełka zapalek ma sens. Jest to po prostu nieekonomiczne. Podobnymi kryteriami powinien posługiwać się programista, tworząc kody niezbędne do poprawnego i wydajnego funkcjonowania aplikacji webowej (i każdej innej).

2.2. Tworzenie zmiennych

W **różnych językach programowania** zmienne są definiowane w różny sposób. W PHP przed nazwą zmiennej stoi znak dolara, np.: \$toJestZmienna, a w JavaScript wpisuje się przed nazwą słowo kluczowe (var, let, const).

Tablica

Jest to zbiór uporządkowanych danych, w którym poszczególne elementy są dostępne za pomocą kluczy. Indeks może przyjmować wartości:

- numeryczne – mówimy wtedy o tablicy indeksowanej numerycznie;
- dowolne – mówimy wtedy o tablicy asocjacyjnej.

Tablice mogą mieć:

- rozmiar ustalony z góry – są to wówczas tablice statyczne, charakterystyczne dla języków programowania typowanych statycznie (np. C, C++);
- rozmiar, który może się zmieniać w trakcie wykonywania programu – są to tablice dynamiczne, charakterystyczne dla języków typowanych dynamicznie, np. JavaScript (przykład znajdziesz na rys. 2.7) i PHP.

```
<script>

    var tablica1 = [1, "kot", 3, null, 5];

    var tablica2 = [tablica1, 9, "zielony"];

</script>
```

Rys. 2.7. Przykładowe tablice – język JavaScript

Zarówno JavaScript, jak i PHP mają tablice dynamiczne. Kolejnym podobnym elementem jest możliwość wpisania różnych typów danych do jednej tablicy. W niektórych językach, np. w C++, dane w tablicy muszą być jednorodne pod względem typu, tzn. mogą zawierać np. tylko liczby bądź ciągi znakowe.

Funkcja

Funkcja w programowaniu działa podobnie jak w matematyce. Za pomocą sposobu charakterystycznego dla danego języka programowania tworzymy zapis matematyczny, który po wywołaniu coś wykona. Dokładnie to, co zaprogramujesz. O ile wykonasz to poprawnie.

```
<script>

    function powiedzHello()
    {
        console.log('Hello');
    }

</script>
```

Rys. 2.8. Przykładowa funkcja – język JavaScript

Funkcja ma określoną strukturę (rodzaj zapisu). Ta zapisana powyżej spowoduje, że po jej wywołaniu w konsoli przeglądarki pojawi się napis „Hello”.

W dalszej części podręcznika bez dodatkowych objaśnień będziemy się posługiwać takimi terminami, jak: zmienna, funkcja, algorytm i programowanie.

2.3. Formatowanie kodu – kod programu a porządek

Jedną z zalet dobrego, profesjonalnego programisty jest idealny (lub prawie idealny) porządek w kodzie, a drugą – opisy poszczególnych bloków kodu.

Zacznijmy od porządku w kodzie. Jednym z jego najważniejszych elementów są tzw. wcięcia. Spójrz na poniższy fragment kodu. Zwróć uwagę na jego kolejne linijki. Używanie „wcięć” powoduje, że kod jest czytelny. Wiemy, w którym miejscu zaczyna się dany blok kodu (czyli dana myśl programisty), a w którym kończy. Istnieje wiele wariantów tworzenia kodu. Jedni stosują jako wcięcia jedną spację, inni – tabulację, a jeszcze inni dwie spacje. Podobnie wygląda sytuacja z nawiasami. Czasami nawiasy klamrowe zaczynają się – jak u nas – na końcu pierwszego wiersza. Wielu programistów zaczyna

jednak nawias klamrowy zawsze w nowej linii, a nawias klamrowy kończący wstawia także w nowej linii, i to z takim samym „poziomem wcięcia” jak nawias początkowy.

Gdy spojrzysz na rys. 2.10, to zobaczysz, że w jego dolnej i górnej części znajduje się ten sam kod. Co ważne, w pierwszej części jest on stworzony poprawnie, dzięki temu jest czytelny i łatwy do analizy. Dolna część zdjęcia zawiera natomiast kod napisany niemalże „ciurkiem”, co oczywiście powoduje brak czytelności. Kod pisany w taki sposób jest bardzo trudno analizować i dalej rozwijać.

```
<!DOCTYPE html>
<html lang="pl">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Witaj w naszej wspaniałej firmie</title>
  <link rel="stylesheet" href="css/style.css">
</head>
```

Rys. 2.9. Kod języka programowania HTML – przykład poprawnego sformatowania kodu nagłówka strony internetowej

! UWAGA

Pamiętaj, że HTML jest językiem programowania składającym się z tzw. znaczników. W strukturze tej technologii można wyodrębnić dwa podstawowe rodzaje wspomnianych elementów:

- mające znaczniki otwierający i zamykający, czyli strukturę typu:

```
<znacznik> ... </znacznik>
```

Przykładowo:

```
<h1> ... </h2>
```

```
<div> ... </div>
```

- samo zamykające się, nazywane także pustymi, czyli strukturę typu:

```
<znacznik > (dopuszcza się także formę <znacznik />)
```

Przykładowo:

```
<img >
```

```
<input >
```



```

try {
    $connection = new PDO($dsn, $db_user,
        $db_password, $options);

    $sql = "SELECT *
        FROM tasks
        WHERE taskname = :taskname";

    $taskname = $_POST['taskname'];
    $statement = $connection->prepare($sql);
    $statement->bindParam(':taskname', $taskname,
        PDO::PARAM_STR);
    $statement->execute();

    $result = $statement->fetchAll();
} catch(PDOException $error) {
    echo $sql . "<br>" . $error->getMessage();
}

```

```

try {$connection = new PDO($dsn, $db_user,
$db_password, $options);
$sql = "SELECT * FROM tasks WHERE taskname =
:taskname";
$taskname = $_POST['taskname'];
$statement = $connection->prepare($sql);
$statement->bindParam(':taskname', $taskname,
PDO::PARAM_STR);
$statement->execute();
$result = $statement->fetchAll();
} catch(PDOException $error) {echo $sql . "<br>".
$error->getMessage();
}

```

Rys. 2.10. Formatowanie kodu PHP – dzięki odpowiednim „wcięciom” z lewej strony poprawia się czytelność poszczególnych bloków kodu


```
<?php

declare(strict_types=1);

namespace Vendor\Package;

use Vendor\Package\{ClassA as A, ClassB, ClassC as C};
use Vendor\Package\SomeNamespace\ClassD as D;

use function Vendor\Package\{functionA, functionB, functionC};

use const Vendor\Package\{ConstantA, ConstantB, ConstantC};

class Foo extends Bar implements FooInterface
{
    public function sampleFunction(int $a, int $b = null): array
    {
        if ($a === $b) {
            bar();
        } elseif ($a > $b) {
            $foo->bar($arg1);
        } else {
            BazClass::bar($arg2, $arg3);
        }
    }

    final public static function bar()
    {
        // method body
    }
}
```

Rys. 2.11. PSR-12 standard tworzenia kodu PHP (źródło: <https://www.php-fig.org/psr/psr-12/>)

Bardzo dużo zależy od przyzwyczajenia programisty. Jeżeli pracujemy nad projektem grupowym, w który angażuje się wielu programistów, to ich kod powinien być pisany według takich samych zasad. Wówczas stosuje się określone standardy – w przypadku PHP zwykle są to standardy PSR. Szczegóły dotyczące tego zagadnienia możesz znaleźć na stronie dbającej o unowocześnianie i standaryzację zasad pisania kodu w PHP – <https://www.php-fig.org/>.

Obecnie najnowszym standardem zasad tworzenia kodu jest PSR-12 (źródło: <https://www.php-fig.org/psr/psr-12/>).

Od początku pracy z programowaniem warto pisać kod według światowych standardów. Jeśli programista „nasiąknie złymi nawykami”, to później będzie mu bardzo trudno dopasować się do właściwych zasad.

Co prawda, w przypadku PHP, HTML, CSS czy JavaScript skrypty, nawet jeżeli będą źle sformatowane, to i tak będą (zwykle) bez problemu działać. Nie wszystkie języki programowania są jednak tak przyjazne. Przykładem może być Python, w którym wcięcia służą do tworzenia bloków kodu. W ten sposób wymusza on na programistach stosowanie określonych konwencji związanych z formatowaniem kodu.

2.4. Opisy poszczególnych bloków kodu

Kodu, który teraz piszesz, za kilka miesięcy możesz już nie do końca rozumieć albo mieć duże problemy z przypomnieniem sobie, co chciałeś zrobić, nie mówiąc już o innym programiście, który ma własny styl programowania i być może zupełnie inne podejście do rozwiązywania danego problemu. Zapewne zdajesz sobie sprawę z tego, że być może twój projekt w przyszłości będzie rozwijał ktoś inny, a ty otrzymasz do dalszego rozwoju aplikację nieznanego tobie twórcy.

„Aby praca programisty miała sens, powinien opisywać swój kod”.

Istnieją co najmniej dwa sposoby tworzenia opisu do kodu:

- dokumentacja techniczna,
- komentarze w kodzie – przykład znajdziesz na rys. 2.12 (język programowania JavaScript).


```

var currentScene = 1;
// Obiekt: główny kontener - tworzymy planszę gry
// Podstawowe zmienne - parametry środowiska
var mainKontener = document.getElementById('ostrichmain');
var szerokoscOkna = window.innerWidth;
var wysokoscOkna = window.innerHeight;
mainKontener.style.width = szerokoscOkna * 0.92 + 'px';
mainKontener.style.height = wysokoscOkna * 0.98 + 'px';

// Test w konsoli
console.log('to jest szerokość okna' + szerokoscOkna);
console.log('to jest wysokość okna' + wysokoscOkna);

// Parametry głównego bohatera - createOstrich1
var createOstrich1 = document.createElement('img');
createOstrich1.setAttribute('id', 'ostrich1Id');
var adresOstrich1 = animalsTablica[0].obrazekAnimalPoza1;
createOstrich1.setAttribute('src', adresOstrich1);

// Obsługa dźwięków Ostrich
var soundOstrich

```

Rys. 2.12. Komentarze w kodzie – język programowania JavaScript

Pierwszym z punktów nie będziemy się szeroko zajmowali. Tworzenie dokumentacji, zwłaszcza takiej, która będzie zrozumiała dla innych osób z branży, to prawdziwa sztuka.

Co do komentarzy – znajdziesz ich mnóstwo na stronach tego podręcznika. **Komentarz** jest elementem kodu, który nie jest pokazywany przez silniki wyświetlające aplikację, np. przez przeglądarkę internetową.

Gdy w HTML (wewnątrz sekcji body) napiszesz:

```

<body>
  <h1>Witaj w świecie programowania</h1>

  <!-- <h2>Ten tekst jest zakomentowany, więc się
       nie wyświetli</h2> -->
</body>

```

Rys. 2.13. Komentarze w kodzie – język programowania HTML

to na stronie internetowej wyświetli się jedynie napis znajdujący się w znaczniku **<h1>**, natomiast tekst zawarty w **<h2>** się nie wyświetlił. Na stronie WWW będzie widoczne jedynie puste białe miejsce.



Rys. 2.14. Widok na stronie internetowej – efekt zastosowania komentarza dla znacznika **<h2>** (język HTML)

„Witaj w świecie programowania”.

Jeżeli użyjesz komentarza:

```
<!-- <h1>Witaj w świecie programowania</h1> -->
```

wówczas na stronie WWW nic się nie wyświetli, gdyż zakomentowanie oznacza, że twórca nie chce pokazywać tego elementu kodu.

Istnieje kilka znaków i/lub ciągów znaków (jak ten użyty powyżej w HTML **<!--** tutaj zakomentowana informacja **-->**), które są wykorzystywane do tzw. komentowania kodu, czyli ukrywania przed faktycznym wykonaniem. Nie oznacza to, że narzędzie wyświetlające, np. przeglądarka, zupełnie nie widzi komentarzy. Ona ich nie pokazuje, natomiast są one na stronie i można je podejrzeć, np. używając w przeglądarce opcji „Pokaż źródło strony”. Wówczas w wyświetlonym kodzie strony internetowej można zobaczyć także komentarze.

Warto, abyś zapamiętał, że dla nas nie istnieje termin „zbyt dużo komentarza”, szczególnie na początku nauki. Pamiętaj, że gdy piszesz komentarz, to nie tylko po to, żeby w przyszłości wiedzieć, jaki był zamiar autora, ale także po to, by dzięki opisowi samemu lepiej zrozumieć napisany kod.

Na koniec tej części podręcznika zapoznaj się z przykładami dotyczącymi sposobów komentowania i opisywania kodu programistycznego dla kilku wybranych języków. Wybrano te, które są związane z programowaniem aplikacji internetowych.

Przykładowe rodzaje komentowania w różnych językach programowania

HTML

```
<!--  
To jest komentarz wieloliniowy  
To jest komentarz wieloliniowy  
-->
```

CSS

```
/*  
To jest komentarz wieloliniowy  
To jest komentarz wieloliniowy  
*/
```

PHP, JavaScript (przykład znajdziesz na rys. 2.15)

```
/*  
To jest komentarz wieloliniowy  
To jest komentarz wieloliniowy  
*/  
// To jest komentarz jednoliniowy
```

SQL

```
/*  
To jest komentarz wieloliniowy  
To jest komentarz wieloliniowy  
*/  
-- To jest komentarz jednoliniowy
```

C#

Komentarz dotyczący tylko jednej linii wygląda następująco:

```
// To jest komentarz jednoliniowy  
/*  
To jest komentarz wieloliniowy  
To jest komentarz wieloliniowy  
*/  
/// To jest komentarz jednoliniowy do dokumentacji  
/**  
To jest komentarz wieloliniowy do dokumentacji  
To jest komentarz wieloliniowy do dokumentacji  
*/
```



```

<?php
/*
Jeżeli użytkownik jest zalogowany, to wykona się szereg poleceń będących
w nawiasie szelkowym po if
*/
if (isset($_SESSION['loggedIntoApp'])) {
    /*
    echo "<br>". 'to jest $_POST["csrf"]:' . $_POST['csrf']. "<br>";
    Zabezpieczenie przed atakiem CSRF
    if (!hash_equals($_SESSION['csrf'], $_POST['csrf'])) die();
    */
    /*
    POŁĄCZENIE Z BAZĄ DANYCH Z WYKORZYSTANIEM PDO
    */
    try {
        $connectionDelete = new PDO($dsn, $db_user, $db_password,
        $options);
    }
}

```

Rys. 2.15. Komentarze wieloliniowe w kodzie – język programowania PHP

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest programowanie?
2. Kim jest programista? Jaka jest, twoim zdaniem, różnica między programistą a informatykiem?
3. Co to jest algorytm?
4. Na czym polega formatowanie kodu programistycznego?
5. Opisz, czym są standardy kodowania.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wejdź na stronę <https://www.php-fig.org/psr/psr-12/> i poczytaj o standardach kodowania w PHP. Popraw kod zawarty na rys. 2.16 tak, aby był zgodny z PSR-12.

```
<?php if (isset($_POST['submitToInPage']))
) {if (!hash_equals($_SESSION['csrf'],
$_POST['csrf'])) die();
    try {$connection = new PDO($dsn,
        $db_user, $db_password, $options);
    $new_task = array("taskname" => $_POST
        ['taskname'], "taskdescr" => $_POST
        ['taskdescr'], "task_comm" => $_POST
        ['task_comm'], "date" => $_POST['date']);
    $sql = sprintf("INSERT INTO %s (%s)
        values (%s)", "tasks", implode(", ",
        array_keys($new_task)), ":" . implode(",
        :", array_keys($new_task)));
    $statement = $connection->prepare($sql);
    $statement->execute($new_task);
    echo "<meta http-equiv=\"refresh\"
        content=\"0;URL=delete.php\">";
    } catch(PDOException $error) {echo $sql
        . "<br>" . $error->getMessage();}}?>
```

Rys. 2.16. Język programowania PHP – niepoprawnie sformatowany kod

3

Projekt: Serwis Ogłoszeniowy Aplikacja Webowa (SOAW) – opis

ZAGADNIENIA

- Opis projektu SOAW
- Responsywność aplikacji
- Podstawowa funkcjonalność tworzonego projektu
- Model biznesowy SOAW

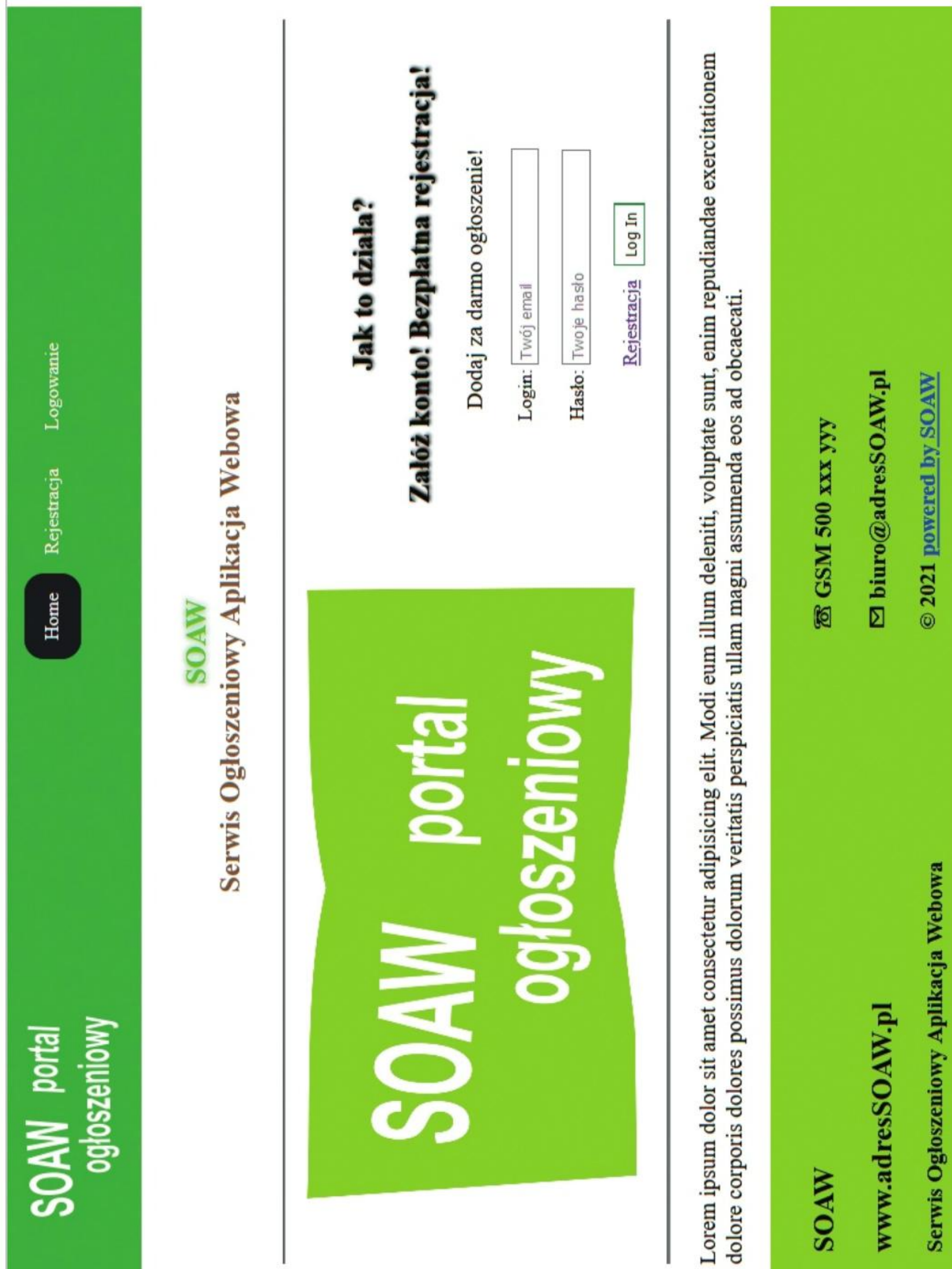
3.1. Serwis Ogłoszeniowy Aplikacja Webowa (SOAW)

W podręczniku znajdziesz mnóstwo przydatnych informacji z zakresu programowania, i to nie tylko obejmujących tradycyjne programowanie webowe (PHP, SQL, HTML, CSS, JavaScript), lecz także inne języki do tworzenia aplikacji. Będzie w nim powracał temat budowy serwisu ogłoszeniowego opartego na tradycyjnych technologiach webowych. Stworzymy podstawy teoretyczne aplikacji SOAW – Serwisu Ogłoszeniowego Aplikacja Webowa. Zaprojektujemy od podstaw jej elementy. Wreszcie dojdziemy do etapu, w którym podstawowe funkcjonalności zaprogramujemy, wykorzystując wspomniane technologie internetowe.



Rys. 3.1. SOAW portal ogłoszeniowy

Podręcznik jest oparty na zbudowaniu elementów projektu (prostego) serwisu ogłoszeniowego. Po zakończeniu jego lektury i dzięki wcześniej nabytej wiedzy z zakresu technologii webowych będziesz w stanie samodzielnie dokończyć, a nawet ulepszyć opisywany projekt SOAW. Widok ukończonej aplikacji znajdziesz na rys. 3.2. Jej wygląd jest prosty, ma wyraziste kolory i jasny przekaz słowny odnoszący się do celu przydatności portalu.



Rys. 3.2. Projekt graficzny projektu SOAW – widok desktop (dla zwykłego monitora)

Spójrz teraz na wygląd wersji mobilnej.

SOAW portal
ogłoszeniowy

Home

Rejestracja

Logowanie

SOAW portal
ogłoszeniowy

Jak to działa?

Zalóż konto! Bezpłatna rejestracja!

Dodaj za darmo ogłoszenie!

Login:

Hasło:

[Rejestracja](#)

Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi eum illum
 deleniti, voluptate sunt, enim repudiandae exercitationem dolore corporis
 dolores possimus dolorum veritatis perspiciatis ullam magni assumenda
 eos ad obcaecati.

SOAW

www.adresSOAW.pl

Serwis Ogłoszeniowy Aplikacja Webowa

☎ GSM 500 xxx yyy

✉ biuro@adresSOAW.pl

© 2021 [powered by SOAW](#)

Rys. 3.3. Projekt graficzny projektu SOAW – widok mobile (dla ekranów urządzeń mobilnych)

3.2. Responsywność aplikacji

Warto wspomnieć o pewnym elemencie CSS dającym tzw. responsywność aplikacji. Została ona osiągnięta z wykorzystaniem zapytań (@media screen) o media, w wyniku podziału pliku stylów CSS na dwie części:

- duże ekrany – szerokość powyżej 768 px,
- małe ekrany (np. smartfony) – szerokość poniżej 768 px.

Przykład zapytania o media znajdziesz na rys. 3.4:

- pierwsza część kodu definiuje właściwości dla ekranów o rozdzielczości powyżej 67 em;
- druga część kodu definiuje właściwości dla urządzeń – ekranów – o rozdzielczości co najmniej 73 em.

```
@media screen and ( min-width: 67em) {  
    /* 1072px */  
    /* NAPISY PRZED LOGO */  
    .zadzwonTel {  
        display: inline-block;  
        padding: 0 5% 5% 11%;  
        position: relative;  
        top: -33px;  
    }  
    .napiszTel {  
        display: inline-block;  
        padding: 0 5% 5% 0;  
        position: relative;  
        top: -33px;  
    }  
}  
  
@media screen and ( min-width: 73em) {  
    /* 1264px */  
    /* LOGO OBRAZEK */  
    .site-header .custom-logo-link img {  
        position: relative;  
        left: 95%;  
        padding-top: 2%;  
        width: 100%;  
    }  
}
```

Rys. 3.4. Kod CSS – zapytanie o media

Zakładamy, że zdobyłeś już wiedzę (silne fundamenty) z zakresu kwalifikacji INF.03. Tworzenie i administrowanie stronami i aplikacjami internetowymi oraz bazami danych. Nie zostaną w takim razie opisane:

1. technologie: HTML, CSS, MariaDB (SQL) i języka SQL;
2. podstawy technologii: PHP, JavaScript (ES6/jQuery) – omijamy podstawy, skupiając się bezpośrednio na technikach programistycznych związanych z projektem jako całością.

Aby pokazać różnorodność świata programistycznego, dodatkowo zostaną wytłumaczone zagadnienia charakterystyczne dla języków typowanych statycznie (czyli np. C++, C#).

```
#include <iostream>
#include <locale.h>

//using namespace std;

int main()
{
    int a = 5, b = 3;
    double c = 5, d = 4;

    std::cout << "a:" << a << ", b:" << b <<
    ", c:" << c << ", d:" << d << std::endl;
    std::cout << "Operacje na liczbach w C+
    +: " << a+b << std::endl;
    return 0;
}
```

Rys. 3.5. Przykładowy kod C++ – operacje matematyczne

Operacje na liczbach w C++: 8

```
Process returned 0 (0x0)    execution time  
: 3.933 s  
Press any key to continue.  
_
```

Rys. 3.5. Cd. Przykładowy kod C++ – operacje matematyczne



CIEKAWOSTKA

Na rys. 3.5 znajdziesz przykład aplikacji konsolowej wykonanej w języku programowania C++. Górna część zawiera kod, a na dole znajduje się widok wykonanego programu w konsoli.

Wykorzystamy w tym celu kilka rozdziałów tego podręcznika, gdyż warto znać nie tylko jedno podstawowe środowisko dla danego programisty (u nas – webowe), ale trzeba zerknąć szerzej. Warto spojrzeć na inne obszary IT, poznać możliwości, jakie oferują nowoczesne technologie programistyczne. Takie podejście jest prorozwojowe i uczyni z ciebie lepszego programistę.

3.3. Podstawowa funkcjonalność tworzonego projektu

Zacznijmy od opisu podstawowej funkcjonalności tworzonego projektu. Podstawowe funkcje portalu SOAW:

- założenie konta administrującego serwisem:
 - jego właścicielem będzie administrator;
- system logowania do aplikacji:
 - użytkownik z poziomu strony internetowej będzie mógł zalogować się do stworzonego portalu – zaprogramujemy tę funkcjonalność od początku do końca;
- rejestracja (a dokładniej autorejestracja) kont użytkowników – klientów wraz z nadaniem odpowiednich uprawnień:
 - z technologicznego punktu widzenia, to zadanie trzeba wykonać za pomocą CRUD (ang. *Create Read Update Delete*, czyli utwórz, odczytaj, aktualizuj, usuń) – tej funkcjonalności w tym podręczniku niestety nie zaprogramujemy, gdyż jest to już wiedza bardzo zaawansowana);
- dodawanie ogłoszeń w określonych kategoriach (w tym podręczniku tego jednak także nie zaprogramujemy).

Jak to działa?

Załóż konto! Bezplatna rejestracja!

Dodaj za darmo ogłoszenie!

Login:

Hasło:

[Rejestracja](#)

Rys. 3.6. Widok planowanego projektu SOAW – logowanie do systemu

SOAW portal ogłoszeniowy

[Home](#)
[Rejestracja](#)
[Logowanie](#)

Zarejestruj się

Email Address

Wprowadź poprawny adres email

Hasło

Wprowadź hasło: min 8 znaków, maks 12 znaków

Powtórz Hasło

Wprowadź hasło- musi być takie samo jak powyżej

Imię

Wpisz Imię: min 2 litery, maks 18 liter

Nazwisko

Wpisz Nazwisko: min 2 litery, maks 18 liter

Telefon

Wprowadź poprawny numer telefonu: 9 znaków

☐

Akceptuję regulamin [Zobacz Regulamin](#)

Only for human :)

☐ Nie jestem robotem


reCAPTCHA
Prywatność - Warunki

Wyślij

[Powrót na stronę główną](#)

SOAW

www.adresSOAW.pl

Serwis Ogłoszeniowy Aplikacja Webowa

☎ GSM 500 xxx yyy

✉ biuro@adresSOAW.pl

© 2021 [powered by SOAW](#)

Rys. 3.7. Projekt graficzny strony do rejestracji użytkowników planowanego projektu SOAW

Jak widzisz, funkcjonalność jest bardzo prosta. Większość z powyższych funkcjonalności planowanej aplikacji zaprojektujemy jako opisowe elementy dokumentacji i napiszemy dla części z nich odpowiednie kody, wykorzystując głównie: PHP, SQL, HTML oraz CSS.

3.4. Model biznesowy SOAW

Na działającym portalu SOAW osoby fizyczne i firmy będą mogły założyć bezpłatne konto, by uzyskać dostęp do umieszczania różnych ogłoszeń i ofert w co najmniej kilku kategoriach. Z punktu widzenia naszego modelu biznesowego konto dla użytkowników może być:

- na zawsze bezpłatne – wówczas od strony biznesowej trzeba będzie pozyskać reklamodawców, od których wpływy zapewnią co najmniej pokrycie kosztów utrzymania przedsięwzięcia biznesowego; niektóre funkcje (czynności) będą jednak płatne, np.:
 - wyróżnianie ogłoszeń (podkreślenie, pogrubienie, przesunięcie na początek listy itp.),
 - umieszczenie ogłoszeń na określony (dłuższy niż w standardzie) czas,
 - wysyłanie e-maili z ofertą wewnątrz stworzonej aplikacji;
- na zawsze bezpłatne, ale jedynie w wersji podstawowej; za dostęp do dodatkowej funkcjonalności użytkownik będzie musiał płacić np. abonament kwartalny – przykładowymi dodatkowymi funkcjonalnościami mogą być:
 - wyróżnianie ogłoszeń (podkreślenie, pogrubienie, przesunięcie na początek listy itp.),
 - umożliwienie kontaktu bezpośredniego wewnątrz portalu (czat, e-mail, rozmowa głosowa itp.);
- na początku bezpłatne (np. przez pierwsze 90 dni), później płatne w formie kwartalnego abonamentu.

Szczegóły dotyczące aplikacji będziemy omawiali podczas tworzenia podstaw specyfikacji technicznej i założenia biznesowego co do utrzymania i dalszego rozwoju SOAW.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Opisz podstawową funkcjonalność projektu SOAW.
2. Wskaż, w jaki sposób projekt SOAW może zarabiać na swoje utrzymanie.
3. Co oznacza responsywność aplikacji?
4. Opisz różne metody zarabiania na projektach internetowych.

II. Proste typy danych

- Proste typy danych
- Typy numeryczne
- Typ logiczny
- Typy znakowe i łańcuchowe
- Inne podstawowe typy danych w programowaniu

4

Proste typy danych

ZAGADNIENIA

- Typ danych
- Proste typy danych
- Przykładowe różnice w językach programowania

4.1. Typ danych

Typ danych określa rodzaj pojemnika, do którego możemy włożyć określone zasoby i je w nim przetrzymywać. Przykładowo: aby przechowywać mleko, nie można skorzystać z papierowego pudełka, gdyż mleko rozmiękczy papier i się wyleje. Oznacza to, że typ danych – papierowe pudełko nie nadaje się do zasobu – mleka (oraz innych płynów).



CIEKAWOSTKA

Na rys. 4.1 przedstawiono kod języka programowania C#, zawierający: zmienną typu znakowego, dwie zmienne liczbowe typu double (zmiennoprzecinkowe) i dwie zmienne typu string.

```
char znak = 'c';  
double num1 = 0;  
double num2 = 870;  
string answer1 = "Twój wybór jest super!";  
string answer2 = "Programowanie, to fantastyczna przygoda!";
```

Rys. 4.1. Przykład zmiennych – język C#

W językach programowania (a jest ich kilkaset) kryteriów, według których możemy omawiać i segregować dane, jest dużo. Część z nich ma jednak znaczenie głównie akademickie, czyli mniej więcej teoretyczne. W praktyce informatycy bardzo często wykorzystują podział na:

- typy proste,
- typy złożone.

Ten podział zastosujemy podczas omawiania podstawowych typów danych używanych w różnych językach programowania.

Aby dobrze zrozumieć typy danych (ich budowę i poprawne zastosowanie), trzeba sięgnąć do nauki podstaw programowania. Dotyczy to wiedzy z zakresu rozróżniania dwóch ważnych pojęć związanych bezpośrednio z programowaniem:

- **typowanie statyczne** – nadawanie typów zmiennym w czasie kompilacji programu, czyli mówiąc obrazowo: pojemnik określa typ danych i to dane trzeba pod względem typu dopasować do pojemnika;
- **typowanie dynamiczne** – przypisywanie typów do wartości, które są przechowywane w zmiennych podczas działania programu, co oznacza, że typ danej przypisanej do danego pojemnika narzuca temu drugiemu, jakiego rodzaju typem danych będzie.

Omawiając temat tego rozdziału, trzeba także zwrócić uwagę na taki dobór zastosowanego typu danych, aby:

- można wprowadzić właściwe dane;
- umieć wyprowadzić poprawne wyniki;
- dane zajmowały jak najmniej miejsca w pamięci komputera.



Rys. 4.2. Niewłaściwie dobrany pojemnik na dane określonej wielkości

4.2. Proste typy danych

Proste typy danych, często określane jako prymitywne, przechowują tylko „surowe” dane, czyli takie, jak:

- liczby stałe (np. 10, 200),
- znaki,
- litery (zarówno z tradycyjnego alfabetu łacińskiego, jak i inne),
- wyrazy – ciągi znaków.

W różnych językach programowania podejście do tego, co jest prostym typem danych, jest nieco odmienne. Przykładowo: w języku JavaScript zdefiniowano siedem typów wbudowanych, z czego sześć uważa się za prymitywne:

- null,
 - undefined,
 - boolean,
 - number,
 - string,
 - symbol (dodany do języka ECMAScript 6!),
- a jeden za typ złożony – Object (w tym: Array, function, Map i Set).

CIEKAWOSTKA

Na rys. 4.3 przedstawiono kod języka programowania JavaScript. Pokazano na nim tworzenie zmiennych za pomocą trzech różnych słów kluczowych: **var**, **let**, **const**. Do zmiennych nie przypisano jednak tylko znaków, ciągów znaków i liczb. W językach programowania mamy bardzo duże bogactwo elementów, które możemy przechowywać w pojemnikach, jakimi są zmienne. Bardzo często przypisuje się do nich różne elementy, np. takie jak na omawianej grafice:

- **obiekty** – tutaj do zmiennych **odbicieRakiеткиP** i **odbicieRakiеткиC** przypisano obiekty potrafiące obsługiwać dźwięk (więcej na ten temat znajdziesz m.in. na stronie: <https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement/Audio>);
- **właściwości** – tutaj do zmiennych **canvasWidth** i **canvasHeight** przypisano wymiary elementu canvas (więcej na ten temat znajdziesz m.in. na stronie: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API);
- **obliczenia matematyczne** – tutaj do zmiennych **ballX** i **ballY** przypisano działania matematyczne na innych zmiennych (wcześniej zadeklarowanych): dzielenie i odejmowanie.

```
var odbicieRakiеткиP = new Audio("sound/odbicieP.mp3");
var odbicieRakiеткиC = new Audio("sound/odbicieC.mp3");

const canvasWidth = canvas.width;
const canvasHeight = canvas.height;

const ballSize = 12; //wielkość piłki
let ballX = canvasWidth/2 - ballSize/2;
let ballY = canvasHeight /2 - ballSize/2;

const lineWidth = 4;
const lineHeight = 9;
```

Rys. 4.3. Przykładowe zmienne w języku programowania JavaScript

Z jednej strony możemy usłyszeć, że w JavaScript wszystko jest obiektem (czyli typem złożonym), a z drugiej strony istnieją typy prymitywne. Aby tę kwestię bardziej zagmatwać, warto wskazać, że string pisany małą literą jest typem prostym, ale String pisany wielką literą to już obiekt, czyli typ złożony. Jak za chwilę zobaczysz, z ciągiem znakowym możemy pracować tak jak z tablicą indeksowaną numerycznie. Pokażemy to już na bazie języka PHP, który jest bardzo podobny w wielu obszarach do JavaScript. Nie ma się co dziwić, oba należą do grupy języków typowanych dynamicznie. Aby to wszystko pojąć, trzeba pamiętać, że w języku JavaScript zmienne są pozbawione typów, bo to język typowany dynamicznie. To wartości mają typy, a nie zmienne.

! UWAGA

W języku JavaScript zmienne są pozbawione typów, gdyż to wartości mają typy, a nie zmienne. W dowolnym momencie zmienne mogą przechowywać różną wartość.

4.3. Przykładowe różnice w językach programowania

Pamiętaj, że JavaScript jest językiem typowanym dynamicznie. Istnieje wiele innych podobnie zachowujących się języków programowania, takich jak Python czy PHP. Zwróć uwagę na szczegóły, bo „podobnie” nie oznacza „tak samo”. Całkiem duża jest także grupa języków programowania typowanych statycznie. Najbardziej znane z nich to: C, C++ i Java. Są one równie popularne jak wspomniane języki typowane dynamicznie.



Rys. 4.3. Język programowania Python

Spójrzmy na język typowany statycznie, jakim jest C++. Jednym z podstawowych podziałów typów danych w tym języku jest podział na:

- **typy proste** (wszystkich typów prostych jest kilkanaście):
 - porządkowe (m.in. znakowy, całkowity, logiczny, wyliczeniowy),
 - rzeczywiste;
- **typy pochodne:**
 - tablice,
 - struktury,
 - unie;
- **typy wskaźnikowe;**
- **typy obiektowe.**

Można przyjąć na tym poziomie edukacji, że typami prostymi są znaki i liczby. Zresztą, bywa z tym różnie w zależności od języka programowania, którego się używa.

Tabela 4.1. Przykładowe rodzaje języków programowania

Języki typowane	
statycznie	dynamicznie
C	PHP
C++	JavaScript
C#	Python
Java	Lua
Kotlin	Ruby

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest typ danych?
2. Wymień co najmniej dwa proste typy danych.
3. Wskaż różnicę między typowaniem statycznym a typowaniem dynamicznym w programowaniu.
4. Podaj nazwę przynajmniej jednego języka typowanego statycznie i jednego typowanego dynamicznie.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Twoim zadaniem jest realizacja autorskiego projektu programistycznego. Wskaż zasady, które powinny być zastosowane podczas wyboru określonego typu danych bądź określonych typów danych.
2. Masz pięć zmiennych o wartościach:

- 'b',
- „pięć”,
- 6,
- true ,
- 2.35.

Napisz program w co najmniej dwóch językach:

- typowanym statycznie (np. C++, C#),
- typowanym dynamicznie (np. JavaScript, PHP).

W każdym ze stworzonych programów:

- przypisz pięć podanych wartości do zmiennych,
- pokaż stworzone zmienne w konsoli (np. CMD, BASH, konsoli przeglądarkowej) bądź na stronie WWW.

5

Typy numeryczne

ZAGADNIENIA

- Podstawowe wyjaśnienia typu numerycznego
- Przegląd typów numerycznych w różnych językach programowania

5.1. Podstawowe wyjaśnienia dotyczące typu numerycznego

Codziennie spotykasz się z liczeniem i liczbami. Uczysz się matematyki, dodajesz, odejmujesz, mnożysz, dzielisz itd. Oznacza to, że prowadzisz operacje na typach numerycznych. Teraz wyobraź sobie, czym są:

- mała liczba i duża liczba,
- liczba całkowita i liczba po przecinku,
- liczba dodatnia i liczba ujemna.



Rys. 5.1. Dane liczbowe

Spójrz na poniższe przykłady liczb, które należą do typu numerycznego:

- 3 – liczba całkowita, to jest zdecydowanie najprostszy (choć wcale nie najczęściej spotykany) przypadek w informatyce, a szczególnie w programowaniu;
- 2.5 – liczba zmiennoprzecinkowa, z precyzją jednego miejsca po kropce. W programowaniu, w przeciwieństwie do codziennej matematyki, używamy kropki jako elementu oddzielającego liczbę całkowitą od jej wartości ułamkowej;
- 1 – mała liczba;
- 9999999999 – duża liczba;
- -765 – liczba ujemna.

Jak widzisz, pojęcie typu numerycznego jest niby proste, ale niestety typy bardzo się różnią w zależności od języków programowania. Sprawę komplikuje jeszcze to, że nie zawsze to, co wydaje się nam liczbą, jest nią w rzeczywistości. Przykładowo: liczba '2', zapisana w apostrofie, dla komputera nie jest z reguły typu numerycznego, ale może być typem numerycznym.

5.2. Przegląd typów numerycznych w różnych językach programowania

Poniżej znajdziesz informacje na temat dwóch języków typowanych dynamicznie i jednego typowanego statycznie. Pamiętaj, że jest to jedynie część wiedzy, a przykładowe typy numeryczne pokazują jedynie podstawy w tym zakresie. Po więcej zawsze możesz zajrzeć np. do internetu.

JavaScript

Zacznijmy od języka JavaScript – tutaj mamy po prostu liczbę, a zwykle rozróżniamy tylko:

- integer – liczba całkowita (`parseInt()`),
- float – liczba zmiennoprzecinkowa (`parseFloat()`).

PHP

Gdy sięgniemy do PHP, poziom skomplikowania typów zmiennych jest bardzo podobny do JavaScript (zapewne jest nieco więcej możliwości), ale typów podstawowych jest tyle samo:

- typ **integer** – całkowitoliczbowy (oznaczany jako **int**), reprezentuje zarówno ujemne, jak i dodatnie liczby całkowite, które mogą być zapisane w formatach:
 - dziesiętnym,
 - ósemkowym (oktalnym),
 - szesnastkowym (heksadecymalnym)
 - dwójkowym (binarnym).

Formatem domyślnym jest zapis dziesiętny. Dla przykładu – do zapisu liczby ósemkowej – należy poprzedzić ją znakiem 0 (zero), a – szesnastkowej – należy poprzedzić ją znakami 0x lub 0X.

- typ **float**, często nazywany również double – reprezentuje liczby zmiennopozycyjne (ang. *floating point*). Zakres tego typu, podobnie jak typu integer, zależy od platformy sprzętowo-systemowej (zwykle maksymalna wartość jest zbliżona do 1.8e308). Co ciekawe, dostępna jest notacja wykładnicza w postaci liczba_potęga_liczba, w której pierwsza liczba jest liczbą w notacji z kropką dziesiętną, a druga to żądana potęga liczby 10. Zamiast słowa „potęga” wpisujemy literę „e”, np.:
 - 4e3 to 4000,
 - 0.17e4 to 1700,
 - 1.9e1 to 19.



Rys. 5.2. Nazwa skrócona języka programowania PHP Hypertext Preprocessor

C++

Jednym z najpopularniejszych języków typowanych statycznie jest C++. Typy zmiennych numerycznych stosowane w tym języku to już dużo bardziej zaawansowany poziom wiedzy. Jest ich całkiem dużo. Na naszym etapie nauki wystarczy, jeśli weźmy pod uwagę i wymienimy najważniejsze, to znaczy jedno z najczęściej używanych w rzeczywistym świecie programowania.



Rys. 5.3. C++ – statyczny język programowania

Przykładowe typy liczbowe z języka C++:

- int – liczby całkowite,
- short int – liczby całkowite,
- long int – liczby całkowite,
- double – liczby rzeczywiste,
- float – liczby rzeczywiste.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Opisz własnymi słowami typ numeryczny.
2. Podaj kilka przykładowych typów numerycznych języka C++.
3. Jakim typem jest liczba 287.5?
4. Wyjaśnij różnice w JavaScript między typami integer i float.
5. Co to jest „notacja wykładnicza” w języku PHP?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Całość zadania ma zostać wykonana w PHP (bez wykorzystania CSS czy HTML).
Masz pięć zmiennych o wartościach: 439.396, 33, 45.5, 921, (−321). Napisz program w PHP, w którym:
 - podane liczby zostaną przypisane do zmiennych;
 - zostanie zakodowane co najmniej dziewięć obliczeń matematycznych, wykorzystujących działania: +, −, /, *, %;
 - w każdym z podanych obliczeń zostaną wykorzystane co najmniej trzy zmienne z tych stworzonych.Napisz kod PHP, który wyświetli na stronie internetowej (w wybranej przeglądarce) wszystkie opracowane:
 - zmienne,
 - obliczenia,
 - wyniki obliczeń.
2. Wykonaj ponownie zadanie 1., ale tym razem użyj jedynie technologii HTML, CSS, JavaScript.

6

Typ logiczny

ZAGADNIENIA

- Definicja typu logicznego
- Wykorzystanie w programowaniu
- Flaga w programowaniu

6.1. Definicja typu logicznego

Typ logiczny, nazywany także od nazwiska angielskiego pioniera logiki – matematyka George’a Boole’a – typem boolowskim (ang. *boolean*), to uporządkowany zbiór wartości logicznych, który składa się wyłącznie z dwóch elementów:

- **prawda** – w programowaniu występuje jako:
 - true,
 - 1,
 - +;
- **fałsz** – w programowaniu występuje jako:
 - false,
 - 0,
 - -.

**Tło, będące pod literami,
które właśnie czytasz, ma
kolor niebieski.**

prawda?

fałsz?

true?

false?

Rys. 6.1. Typ logiczny – boolean – przykład zastosowania w podejmowaniu decyzji

Zwróć uwagę na zastosowane zapisy: true, 1, +. Każdy z nich oznacza prawdę. Z kolei: false, 0, - oznaczają zawsze fałsz. Obie grupy są niezmiernie ważne w programowaniu. Co istotne, **każdy zapis w kodzie programistycznym jest prawdą albo fałszem**. Innej możliwości nie ma. Dzięki temu prawie każdy kawałek kodu może się stać elementem decyzji true lub false.

6.2. Wykorzystanie w programowaniu

Wartości logiczne są tak powszechnie wykorzystywane w programowaniu, że programiści zapewne nie byliby sobie w stanie wyobrazić ich braku. Najczęściej – oczywiście – jako elementy logiki programu w tzw. instrukcjach warunkowych. Zanim do tego przejdziemy, warto przypomnieć jedną z podstawowych zasad, na której opiera się informatyka i budowa komputera (oczywiście w znacznym uproszczeniu):

„Prąd płynie – prąd nie płynie”.

CIEKAWOSTKA

JavaScript to język programowania, za którego pomocą możesz w szybki sposób – używając kilkunastu linii kodu i odrobiny HTML-a – stworzyć interaktywne (kolorowe, animowane) elementy na stronę internetową. Przykład kodu, który powoduje przemieszczanie się elementów HTML na stronie WWW, przedstawiono na rys. 6.3. Widok możliwości JavaScript w kwestii wyglądu elementów HTML możesz zobaczyć na rys. 6.2.



Rys. 6.2. Widok projektu edukacyjnego wykonanego w JavaScript, pokazującego bramki logiczne, czyli zastosowanie logiki w informatyce

Oczywiście chodzi o system zero – jeden, czyli fundamenty budowy i funkcjonowania wszystkich urządzeń, które potocznie nazywamy komputerami (także smartfon). Okazuje się, że całe twoje urządzenie przenośne składa się z zer i jedynek!

System zero – jeden jest idealny do podejmowania decyzji, skoro mamy tylko dwa wyjścia: „Będziesz programistą albo nim nie będziesz, innej drogi nie ma”.

```
// LOGIKA - BRAMKI LOGICZNE

function sprawdzAnimacjeBramkiOr() {
    if ((kulkaAnimacjaBramkiOr.classList.contains
        ("kulkaAnimacjaBramkiOr")) && (pierwszy.innerHTML == "pierwszy
        - wciśnięty" || drugi.innerHTML == "drugi - wciśnięty")) {
        console.log("1");
        // kulkaAnimacjaBramkiOr.setAttribute("class",
        "kulkaAnimacjaBramkiOr");
    } else if (!(kulkaAnimacjaBramkiOr.classList.contains
        ("kulkaAnimacjaBramkiOr")) && (pierwszy.innerHTML == "pierwszy
        - wciśnięty" || drugi.innerHTML == "drugi - wciśnięty")) {
        kulkaAnimacjaBramkiOr.setAttribute("class",
        "kulkaAnimacjaBramkiOr");
    } else {
        kulkaAnimacjaBramkiOr.removeAttribute("class",
        "kulkaAnimacjaBramkiOr");
    }
}
```

Rys. 6.3. Widok kodu wykonanego w JavaScript, pokazującego bramki logiczne, czyli zastosowanie logiki w informatyce

Pora przejść na nasze terytorium – „kodzenia”, bo tak potocznie wielu programistów mówi o kodowaniu. Znajomość żargonu programistów bywa bardzo pomocna. Na początku poznamy flagę programistyczną i jej wykorzystanie w informatyce.

6.3. Flaga w programowaniu

Flaga w programowaniu zwykle oznacza zmienną, której przypisujemy wartość, np. true (prawda) lub – w zależności od potrzeb programu i programisty – false (fałsz). Dalej kod wykonuje kolejne kroki, obwarowane np. instrukcjami **if**, a my (programiści) w kodzie ustawiamy flagę:

- na true, gdy program może iść dalej, lub
- na false, gdy musi nastąpić zatrzymanie programu, bo np. użytkownik wpisał niepoprawny login (np. zawierający niedozwolone znaki czy też po prostu niezgodny z zapisami w bazie danych).


```
// Czy walidacja zakończyła się sukcesem? Jeżeli tak,  
to flage została ustawiona na: $flagaTest = true;  
  
if ($flagaTest == true) {  
    // POŁĄCZENIE Z BAZĄ DANYCH Z WYKORZYSTANIEM PDO  
    try {  
        $connection = new PDO($dsn, $db_user,  
            $db_password, $options);  
        if (!$connection) {  
            throw new Exception($connection->error);  
        } else {  
            // Kod sprawdzający użytkownika o danej nazwie  
            oraz zgodność podanego hasła.  
        }  
    } catch (Exception $e) {  
        echo 'Błąd serwera! .. (tutaj także może być  
            szczegółowa informacja o błędach)';  
    }  
}
```

Rys. 6.4. Widok kodu wykonanego w języku programowania PHP (zastosowanie omawianej flagi w programowaniu)

W praktyce pisania kodu ustawiamy flagę, czyli po prostu zmienną (rys. 6.4). Nadajemy jej nazwę. Może ona się równać:

- false, gdy nasz kod nie przejdzie pozytywnie jakiegoś testu;
- true, gdy nasz kod przejdzie pozytywnie jakiś test.

W podobny sposób będziemy postępowali, tworząc kod dla aplikacji SOAW. Zaplanujemy kolejne etapy, a w kodzie PHP na każdym z nich ustawimy flagę. Będzie nam ona wyznaczała kierunek, a raczej kontynuację ruchu. Gdy użytkownik wpisze poprawny login, pójdzie dalej, będzie mógł wpisać hasło. Gdy także hasło będzie zgodne, to kolejny test zostanie zdany, a użytkownik uzyska dostęp do środka aplikacji.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest boolean w programowaniu?
2. Wy tłumacz wartości zmiennej logicznej: true, false.
3. Wyjaśnij w kilku zdaniach, czym jest system zerojedynkowy.
4. Jakie jest zadanie tzw. flagi w programowaniu?



SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1.

- Stwórz podstawową strukturę HTML strony internetowej. Wewnątrz znaczników: `<body>` `</body>` umieść znacznik na kody JavaScript `<script>` `</script>`.
- Za pomocą wyłącznie JavaScript stwórz blok HTML `<div></div>`. Stworzonemu elementowi nadaj kolor żółty.
- Wewnątrz wcześniej stworzonego elementu dodaj kolejne dwa bloki `<div></div>`. Nadaj im kolory: zielony i czerwony. Za pomocą kodu JavaScript spraw, aby wyświetlały się jako koła, a nie prostokąty.

Pamiętaj: do kodowania możesz używać wyłącznie JavaScript.

2. W tym zadaniu będą potrzebne kody z poprzedniego zadania. Używając języka programowania JavaScript, animuj przynajmniej jeden z wewnętrznych elementów `div`, stworzonych w zadaniu 1.

7

Typy znakowe i łańcuchowe

ZAGADNIENIA

- Znak, stałe znakowe i łańcuch
- Cudzysłów kontra apostrof w programowaniu
- Konkatenacja
- Typy oparte na relacyjnej bazie danych SQL
- Projekt SOAW – budowa bazy

7.1. Wyjaśnienie pojęć znaku i łańcucha

Zacznijmy od języka C++, gdyż istnieje w nim typ **char**, który przechowuje pojedynczy znak. Dzięki pokazaniu omawianego pojęcia w tym języku łatwiej będzie zrozumieć kolejny element programowania – funkcjonowanie znaków w PHP, a dokładniej apostrofy i cudzysłowy.



Rys. 7.1. C – statyczny język programowania

C++

Stałe znakowe to stałe, które reprezentują np. znaki alfanumeryczne. Zapisuje się je, ujmując dany znak w dwa apostrofy:

- `'b'` – oznacza literę b;
- `'5'` – oznacza cyfrę 7 (cyfrę, nie liczbę).

Tak zapisane znaki są typu `char`. Można je więc przypisać do obiektów (zmiennych) typu `char`.

PRZYKŁAD

```
char znak;
znak = 'b';
```

```
// poprawne przypisanie
```

```
char zmieChar = 'x';
```

```
// błędne
//przypisanie
```

za dużo znaków w stałej znakowej C/C++(26)

Peek Problem (Alt+F8) No quick fixes available

```
char zmieChar2 = 'Witaj';
```

Rys. 7.2. Widok kodu wykonanego w C++ – poprawne i błędne użycie char

Zmienne znakowe w C++ są deklarowane z wykorzystaniem typu char. Do każdego z nich można przypisać tylko jeden znak. Przykład poprawnego i błędnego przypisania możesz zobaczyć na rys. 7.2. Zauważ, że w momencie popełnienia błędu dobre środowisko IDE (w moim wypadku Visual Studio Code) wskazuje błąd za pomocą pofalowanego czerwonego znaku.

7.2. Cudzysłów a apostrof

Gdy chcemy przypisać więcej znaków do zmiennej, to musimy użyć zmiennej typu string, a jej zawartość wpisać w cudzysłów.

! UWAGA

Do **char** przypisujemy tylko jeden znak w **apostrofie**, a do **stringa** – nieograniczoną liczbę znaków w **cudzysłowie**.

PHP

Zapewne widzisz różnicę w wyniku dla znaków zapisywanych w cudzysłowie i w apostrofie. W winnych językach programowania wygląda to bardzo różnie. Przykładowo: w JavaScript prawie (a przynajmniej w tradycyjnej JS na stronach WWW) nie ma różnicy między zapisem prawej strony przypisania do zmiennej przez apostrof a cudzysłów. Za to już w PHP różnica jest duża, ale głównie na poziomie pokazywania zmiennych, gdyż apostrof pokazuje zawsze swoją zawartość w postaci zwykłego stringu (ciągu znaków), gdy tymczasem cudzysłów dokonuje interpretacji zawartości. Oznacza to, że jeżeli wstawimy wewnątrz struktury „ ” (cudzysłów), np. zmienną, to zostanie ona zinterpretowana, więc zostanie wyświetlona wartość przypisana do danej zmiennej.


```
<?php  
  
$apostrof = 'cudzyslow';  
  
echo "Na stronie pokaże się: " . '<br>';  
echo "$apostrof" . '<br>' . '$apostrof';
```

Rys. 7.3. Widok kodu wykonanego w PHP – apostrof a cudzysłów

Zerknij na kod napisany w języku programowania PHP przedstawiony na rys. 7.3. Zmienna **\$apostrof** w konstrukcji echo jest wpisana za pierwszym razem w cudzysłów, a za drugim – w apostrof. Spójrz na rys. 7.4, jak oba zapisy zostały wyświetlone na stronie przeglądarki internetowej. Zauważ, że cudzysłów dokonał interpretacji zmiennej, gdy tymczasem apostrof po prostu przepisał ciąg tekstowy.

Na stronie pokaże się:
cudzyslow
\$apostrof

Rys. 7.4. Widok w przeglądarce internetowej kodu wykonanego w PHP – apostrof a cudzysłów

Łańcuchy znaków

Jest to element bardziej rozbudowany niż sam znak, gdyż są to właściwie tablice znaków, które są zakończone znakiem `\0` (podczas wyświetlania tej końcówki nie widać). Spójrz dla przykładu na PHP. Jeżeli przypiszemy ciąg znaków do zmiennej, to potem możemy odczytywać kolejne znaki (tutaj: litery) w taki sposób, jak z tablicy indeksowanej numerycznie. Możesz to zobaczyć na rys. 7.5.

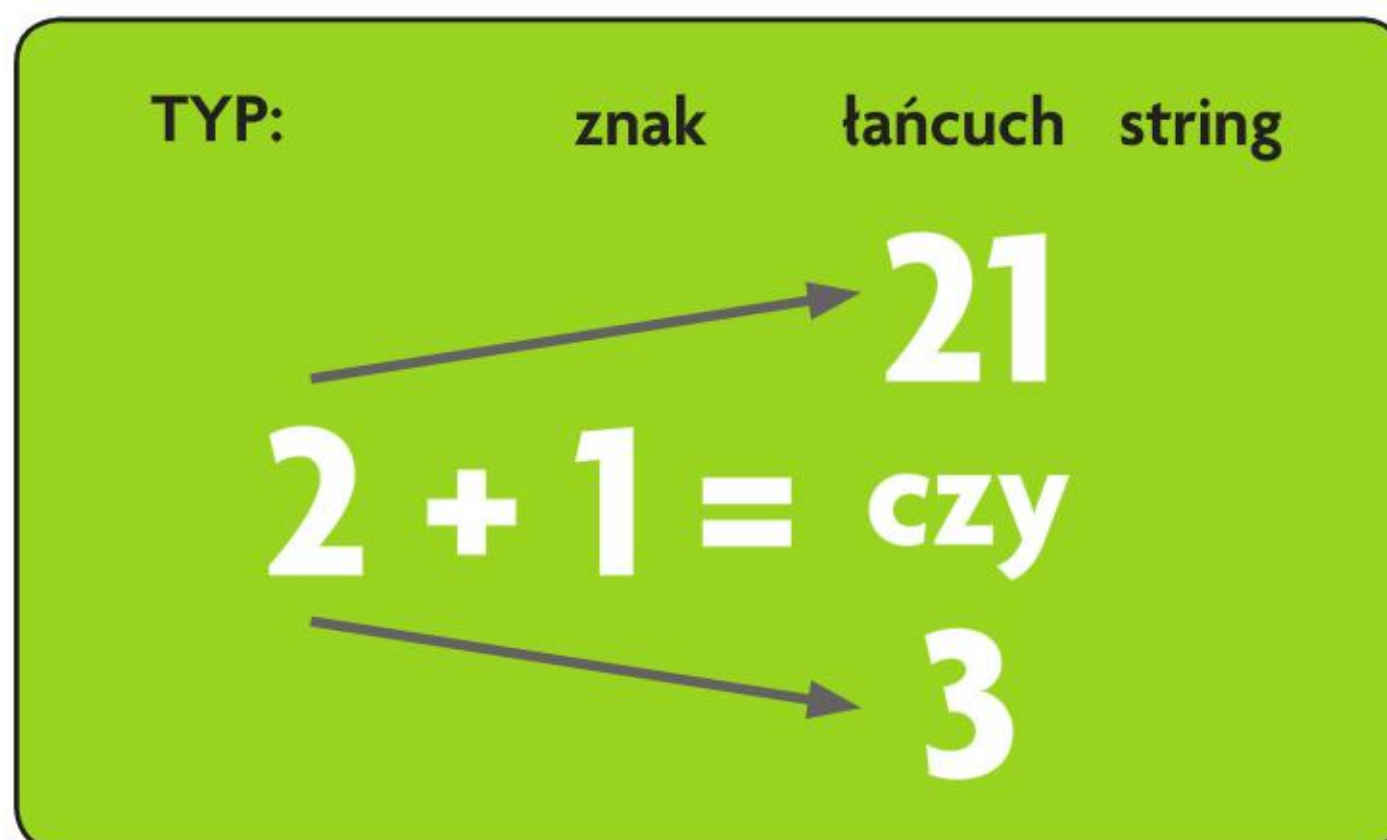
Dzięki temu, znając zasady funkcjonowania tablic w programowaniu, możemy w dowolny sposób pracować z dowolnym fragmentem tekstu i wprowadzać w nim takie zmiany, jakie są nam potrzebne. O tablicach będzie już wkrótce, i to nie tylko indeksowanych numerycznie.


```
// Zmienna znak składa się z trzech znaków  
a, b i c. Z jednej strony jest to string,  
więc typ prosty. Z drugiej jest tablicą  
znaków.  
$znak = 'abc';  
  
echo $znak[1];  
  
// Na stronie otrzymamy wynik  
// b
```

Rys. 7.5. PHP – ciąg znaków a tablica

7.3. Konkatenacja

Konkatenacja w językach programowania oznacza łączenie dwóch wyrażeń, np. tekstowych (stringów), w jedno wyrażenie, w którym oba składniki zostaną ustawione jeden za drugim.



Rys. 7.6. Konkatenacja a matematyka

Oto operatory konkatenacji ciągów w kilku popularnych językach programowania:

- + (plus)
JavaScript, C++, C#, Java, Ruby, Python;
- . (kropka)
Perl, PHP.

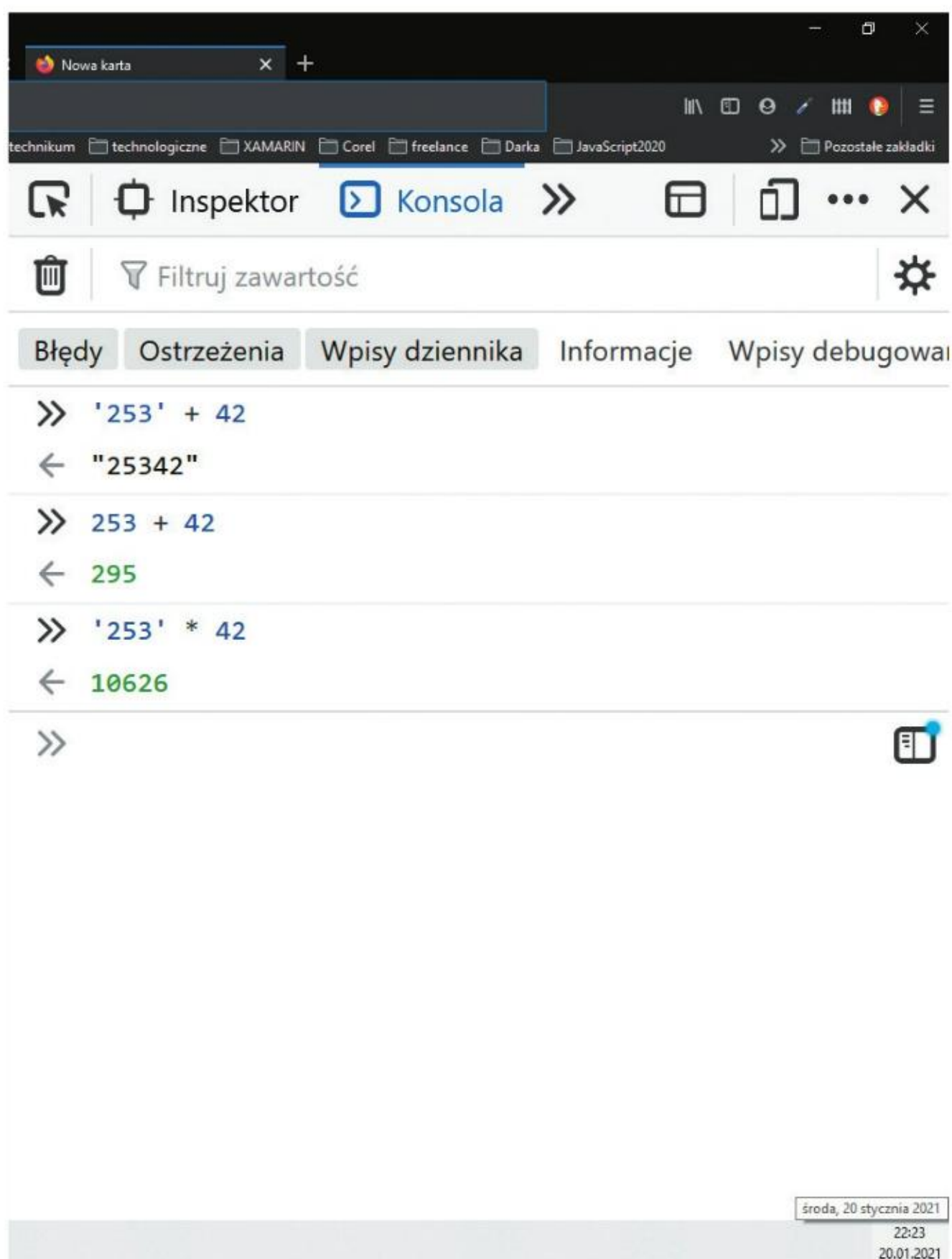
Podczas procesu konkatencji silnik danego języka programowania sprawdza tzw. pierwszeństwo typów. Przykładowo: w JavaScript (podobnie jak w większości innych języków) pierwszeństwo ma zamiana na typ łańcuchowy. Oczywiście dotyczy to głównie języków należących do grupy typowanych dynamicznie. Spójrz na rys. 7.7. Zawiera on (widok w konsoli przeglądarki) kilka działań matematycznych. O ile w przypadku:

$253 + 42 = 295$ – sytuacja jest prosta i oczywista, nastąpiło działanie matematyczne, dodawanie, czego efektem jest liczba 295,

o tyle już w działaniu

$'253' + 42 = „25342”$ – mamy do czynienia ze wspomnianym pierwszeństwem typów i nastąpiła tzw. konkatencja, czyli połączenie dwóch ciągów znakowych.

Z kolei mnożenie spowodowało uzyskanie pierwszeństwa dla typu numer, dlatego otrzymaliśmy wynik zwykłego działania matematycznego – mnożenia. Powyższe działania znajdziesz na rys. 7.7.



Rys. 7.7. Działania matematyczne w konsoli przeglądarki – język programowania JavaScript

7.4. Typy oparte na relacyjnej bazie danych SQL

Relacyjne bazy danych mają własne rodzaje typów, niektóre są wspólne dla różnych systemów baz SQL, a inne specyficzne wyłącznie dla danej bazy.

The screenshot shows the phpMyAdmin interface for the 'user' table. The table structure is as follows:

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo	Działanie
1	id_user	int(11)			Nie	Brak	Klucz główny nadawany automatycznie	AUTO_INCREMENT	Zmień Usuń Więcej
2	email	varchar(30)	utf8mb4_unicode_ci		Nie	Brak	Adres e-mail - nazwa użytkownika potrzeba przy logowaniu.		Zmień Usuń Więcej
3	haslo	varchar(256)	utf8mb4_unicode_ci		Nie	Brak	Niezaszyfrowane hasło		Zmień Usuń Więcej
4	imie	varchar(20)	utf8mb4_unicode_ci		Nie	Brak	Imię użytkownika		Zmień Usuń Więcej
5	nazwisko	varchar(30)	utf8mb4_unicode_ci		Nie	Brak	Nazwisko użytkownika		Zmień Usuń Więcej
6	telefon	varchar(12)	utf8mb4_unicode_ci		Tak	NULL	Telefon użytkownika 9-io cyfrowy		Zmień Usuń Więcej

The interface also shows the 'Indeksy' (Indexes) tab with the following information:

Działanie	Nazwa klucza	Typ	Jednoznaczny	Spakowany	Kolumna	Moc	Metoda porównywania napisów	Null	Komentarz
Edytuj Usuń	PRIMARY	BTREE	Tak	Nie	id_user	0	A	Nie	
Edytuj Usuń	email	BTREE	Tak	Nie	email	0	A	Nie	

Rys. 7.8. phpMyAdmin – widok struktury danych, wskazuje m.in. typ danych projektowych

Jak już wspomniano, zakładamy, że masz już podstawy wiedzy z zakresu relacyjnej bazy danych (a ściślej – jej bezpłatnej implementacji zawartej w pakiecie XAMPP, czyli MariaDB). Dlatego jedynie wymienimy podstawowe typy używane w bazach danych.

Typy tekstowe

CHAR(size), VARCHAR(size), TINYTEXT, TEXT, BLOB, MEDIUMTEXT, MEDIUMBLOB, LONGTEXT, LONGBLOB, ENUM(a,b,c,etc.), SET

Typy liczbowe

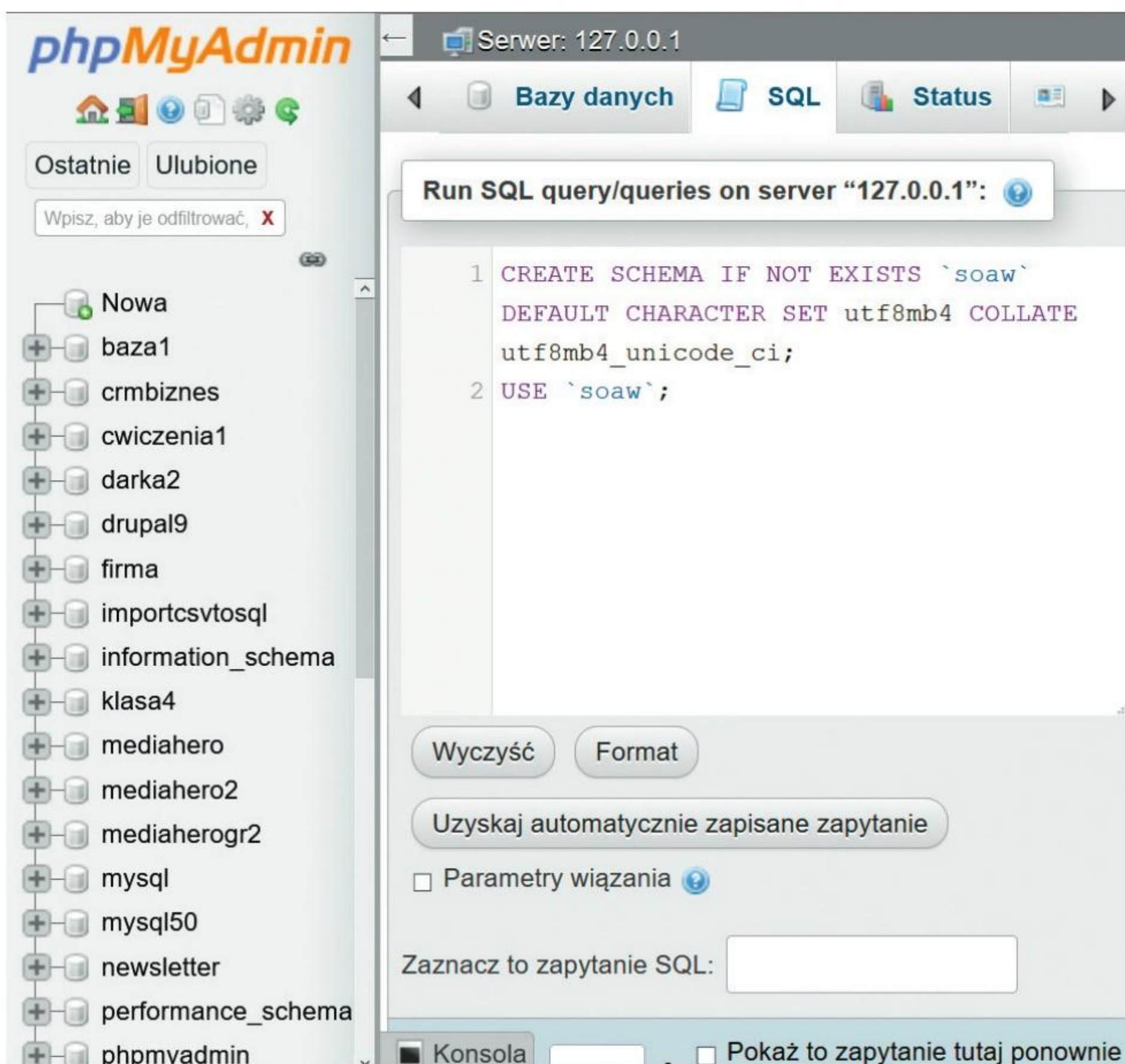
TINYINT(size), SMALLINT(size), MEDIUMINT(size), INT(size), BIGINT(size),
 FLOAT(size,d), DOUBLE(size,d), DECIMAL(size,d)

Typy daty

DATE(), DATETIME(), TIMESTAMP(), TIME(), YEAR()

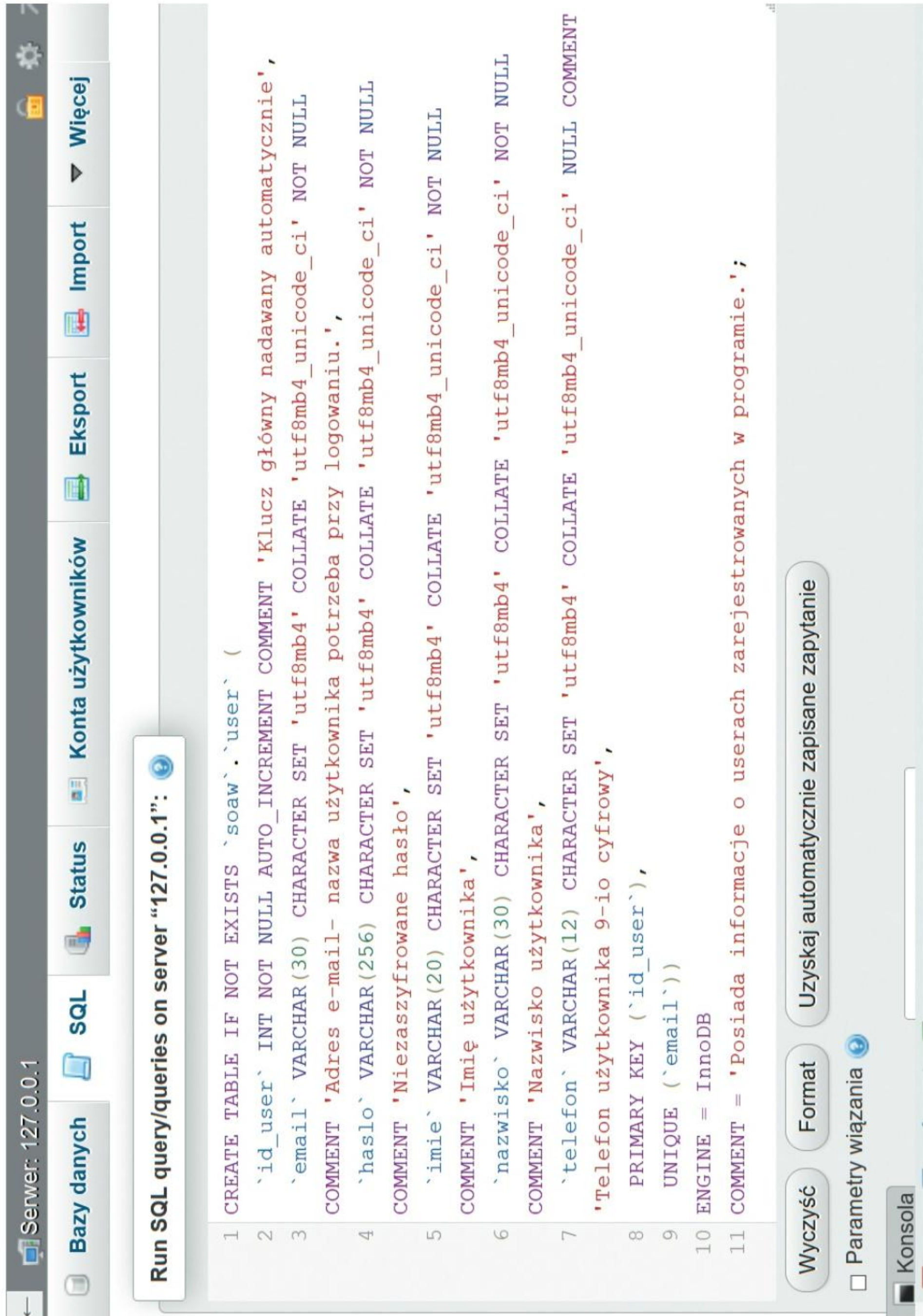
7.5. Projekt SOAW – budowa bazy

Skoro już jesteśmy przy tematyce związanej z bazami danych, utwórzmy z pomocą języka SQL bazę dla tworzonego projektu i pierwszą tabelę.



Rys. 7.9. phpMyAdmin – widok panelu SQL – tworzenie nowej bazy danych

Napisany kod zadba od razu o właściwe formatowanie i wyświetlanie polskich znaków (**utf8mb4 COLLATE utf8mb4_unicode_ci**). Gdy już mamy bazę danych, możemy zrobić kolejny krok – przygotować, napisać i uruchomić kod tworzący w bazie danych pierwszą tabelkę projektową, która będzie zawierała użytkowników.



Rys. 7.10. Stworzenie tabeli 'user' w bazie danych 'soaw'

Tabela **'user'** będzie przechowywała dane o użytkownikach aplikacji, czyli osobach, które będą chciały umieścić ogłoszenie. Aby umieścić zaproszenie, internauta będzie musiał założyć konto na portalu aplikacji SOAW.

Stworzona tabela **user** będzie zawierała kolumny: **'id_user'**, **'email'**, **'haslo'**, **'imie'**, **'nazwisko'**, **'telefon'**.

Struktura tabeli została tak zaplanowana, aby przechowywać dane użytkowników – niezbędne do świadczenia im usługi dostępu do portalu i umieszczania na nim (w bazie) danych. Tworząc strukturę profili użytkowników, musimy pamiętać o RODO (tj. Rozporządzeniu Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE – więcej znajdziesz na stronie: <https://www.uodo.gov.pl/pl/404/224>). Opierając się na zapisach wspomnianego dokumentu, powinniśmy przechowywać taki zakres danych osób fizycznych, który jest niezbędny do ochrony firmy pobierającej dane. Zakres danych naszej tabeli jest uzasadniony z punktu widzenia działań biznesowych i prawnych, niezbędnych do kierowania firmą prowadzącą portal SOAW.

UWAGA

Zawsze kiedy planujemy pobieranie i przechowywanie danych osobowych, trzeba pamiętać o zapisach prawnych dotyczących ochrony praw i wolności osób, których dane są przetwarzane. Aby lepiej zrozumieć temat, warto zaznajomić się z dwoma zapisami z Rozporządzenia Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r.

(78) Ochrona praw i wolności osób fizycznych w związku z przetwarzaniem danych osobowych wymaga wdrożenia odpowiednich środków technicznych i organizacyjnych, by zapewnić spełnienie wymogów niniejszego rozporządzenia. Aby móc wykazać przestrzeganie niniejszego rozporządzenia, administrator powinien przyjąć wewnętrzne polityki i wdrożyć środki, które są zgodne w szczególności z zasadą uwzględniania ochrony danych w fazie projektowania oraz z zasadą domyślnej ochrony danych. Takie środki mogą polegać m.in. na minimalizacji przetwarzania danych osobowych, jak najszybszej pseudonimizacji danych osobowych, przejrzystości co do funkcji i przetwarzania danych osobowych, umożliwieniu osobie, której dane dotyczą, monitorowania przetwarzania danych, umożliwieniu administratorowi tworzenia i doskonalenia zabezpieczeń. Jeżeli opracowywane, projektowane, wybierane i użytkowane są aplikacje, usługi i produkty, które opierają się na przetwarzaniu danych osobowych albo przetwarzają dane osobowe w celu realizacji swojego zadania, należy zachęcać wytwórców tych produktów, usług i aplikacji, by podczas opracowywania i projektowania takich produktów, usług i aplikacji wzięli pod uwagę prawo do ochrony danych osobowych i z należyтым uwzględnieniem stanu wiedzy technicznej zapewnili administratorom i podmiotom przetwarzającym możliwość wywiązania się ze spoczywających na nich obowiązków ochrony danych. Zasadę uwzględniania ochrony danych w fazie projektowania i zasadę domyślnej ochrony danych należy też brać pod uwagę w przetargach publicznych.

(79) Ochrona praw i wolności osób, których dane dotyczą, oraz obowiązki i odpowiedzialność prawna, administratorów i podmiotów przetwarzających – także w odniesieniu do monitorowania ze strony organów nadzorczych i do środków przez nie stosowanych – wymagają dokonania w ramach niniejszego rozporządzenia jasnego podziału obowiązków, także w sytuacji, gdy administrator określa cele i sposoby przetwarzania wspólnie z innymi administratorami lub gdy operacji przetwarzania dokonuje się w imieniu administratora.

Serwer: 127.0.0.1 » Baza danych: soaw » Tabela: user "Posiada informacje o userach zarejestrowanych w programie."

Przeglądaj Struktura SQL Szukaj Wstaw Eksport Import Uprawnienia Operacje Więcej

Struktura tabeli Widok relacyjny

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo	Działanie
☐ 1	id_user	int(11)			Nie	Brak	Klucz główny nadawany automatycznie	AUTO_INCREMENT	Zmień Usuń Więcej
☐ 2	email	varchar(30)	utf8mb4_unicode_ci		Nie	Brak	Adres e-mail- nazwa użytkownika potrzeba przy logowaniu.		Zmień Usuń Więcej
☐ 3	haslo	varchar(256)	utf8mb4_unicode_ci		Nie	Brak	Niezaszyfrowane hasło		Zmień Usuń Więcej
☐ 4	imie	varchar(20)	utf8mb4_unicode_ci		Nie	Brak	Imię użytkownika		Zmień Usuń Więcej
☐ 5	nazwisko	varchar(30)	utf8mb4_unicode_ci		Nie	Brak	Nazwisko użytkownika		Zmień Usuń Więcej
☐ 6	telefon	varchar(12)	utf8mb4_unicode_ci		Tak	NULL	Telefon użytkownika 9-io cyfrowy		Zmień Usuń Więcej

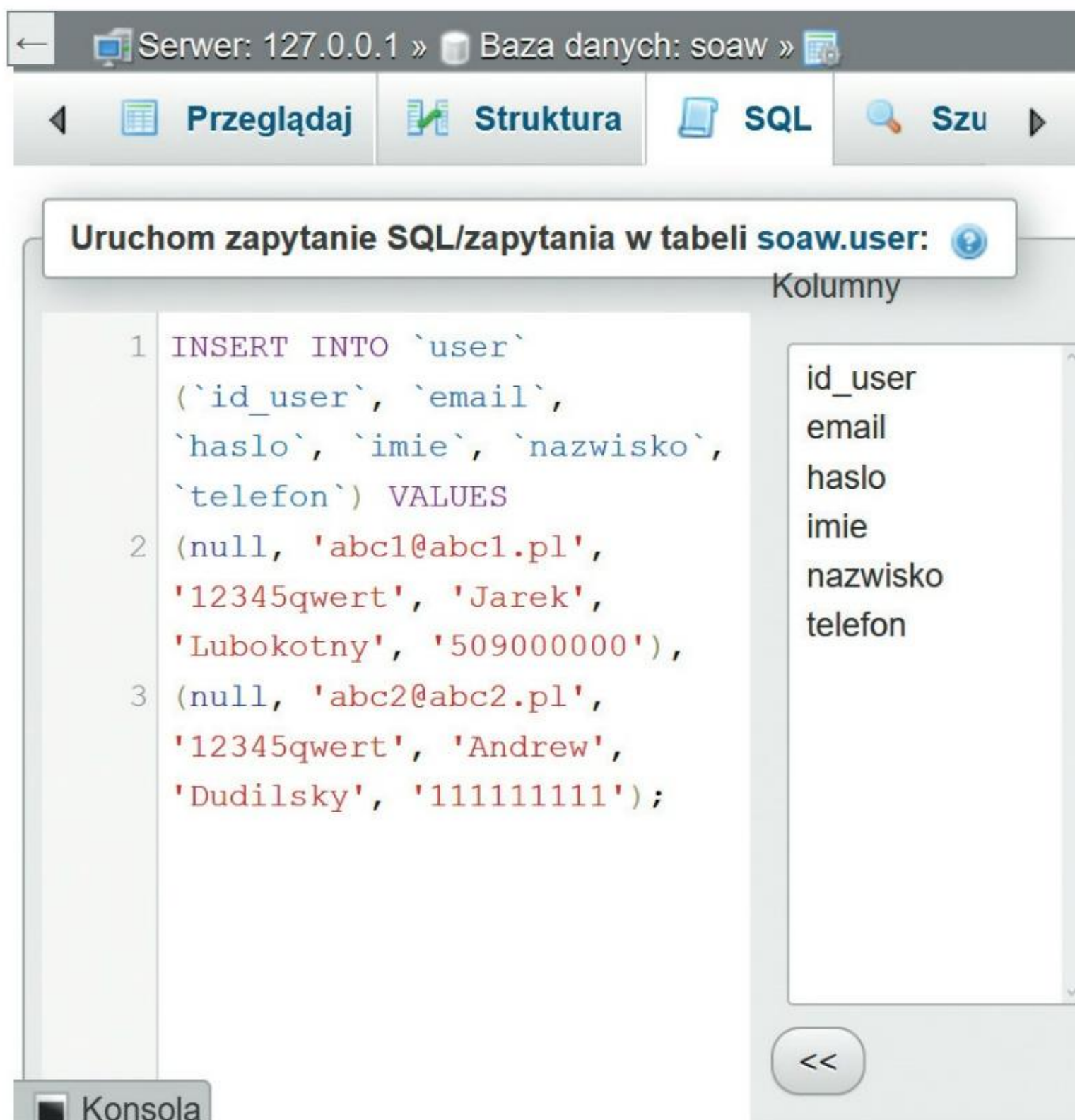
☐ Zaznacz wszystko Z zaznaczonymi: Przeglądaj Zmień Usuń Podstawowy Jednoznaczny Indeks

☐ Pełny tekst Dodaj do środkowych kolumn Usuń z centralnych kolumn

Drukuj Zaproponowanie struktury tabeli Śledź tabelę Przenieś kolumny Normalizuj

Rys. 7.11. Struktura pustej tabeli 'user'

Aby w jednym z kolejnych rozdziałów można było stworzyć system logowania łączący się z bazą danych **'soaw'**, musimy mieć jakieś dane, dzięki którym będziemy mogli przeprowadzić logowanie. Dodajmy zatem dwa rekordy z danymi, wykorzystując menedżera baz phpMyAdmin i język służący do komunikacji z relacyjnymi bazami danych, czyli SQL.



Rys. 7.12. phpMyAdmin – widok panelu SQL – dodanie kodu wstawiającego dwa rekordy do tabeli **'user'**

Dodaliśmy bardzo proste dane. W szczególności dotyczy to adresów e-mail, będących loginami, i haseł. Pamiętaj, że to tylko dane testowe. Nie są jeszcze w żaden sposób zabezpieczone. Bardzo istotne jest to, aby powyższe operacje na bazie MariaDB zostały przez ciebie skutecznie przeprowadzone, gdyż dodane rekordy będą służyły do testowania połączenia w PHP, które będziemy robili w ostatnim rozdziale tego podręcznika. Efektem końcowym tej części podręcznika muszą być działające w XAMPP baza **'soaw'** i umieszczona w niej tabela **'user'**, zawierająca co najmniej uzupełniony jeden rekord.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co w programowaniu oznaczają nazwy: „znak”, „stałe znakowe”, „łańcuch”?
2. Wyjaśnij różnice między znakami cudzysłowu i apostrofu w programowaniu.
3. Opisz pojęcie konkatencji.
4. Wymień kilka typów danych opartych na relacyjnej bazie SQL.
5. Napisz polecenie SQL tworzące bazę danych o nazwie twojego imienia.
6. W pakiecie XAMPP zawarta jest baza danych. Która to baza: MySQL, MariaDB czy Postgres?
7. Otwórz pasek dewelopera w twojej ulubionej przeglądarce internetowej (zwykle klawisz F12 przy aktywnej karcie przeglądarki). Wykonaj działania matematyczne na liczbach, wpisując część liczb w apostrof, inne w cudzysłów, a jeszcze inne bez dodatków. Zapisz otrzymane wyniki i je porównaj.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Aby wykonać zadanie, w swoim komputerze musisz mieć działający pakiet XAMPP. Jest to zadanie przypominające wiedzę z zakresu SQL. Dzięki jego wykonaniu szybciej i łatwiej przyswoisz wiedzę zawartą w dalszej części podręcznika.
 - Utwórz bazę danych – nazwa powinna składać się z trzech pierwszych liter twojego nazwiska, bez polskich znaków (np. gol).
 - W utworzonej bazie stwórz tabelę o nazwie według wzoru: twoje imię (np. darek), z kolumnami (pamiętaj o autoinkrementacji):
 - id, rodzaj danych – liczbowe;
 - jezyk, typ danych – krótki tekst składający się z 30 znaków;
 - typ_jezyka, typ danych – krótki tekst składający się maksymalnie z 30 znaków;
 - ranking, typ danych – liczbowe.
 - Do powyższej tabeli wstaw następujące dane – w podanej kolejności:
 - id, jezyk, typ_jezyka, ranking (wzór dla poniższych danych);
 - PHP, typowany dynamicznie, 5;
 - Python, typowany dynamicznie, 5;
 - JavaScript, typowany dynamicznie, 6;
 - C++, typowany statycznie, 4;
 - C, typowany statycznie, 3;
 - Lua, typowany dynamicznie, 4;
 - Ruby, typowany dynamicznie, 4;
 - Java, typowany statycznie, 3;
 - C#, typowany statycznie, 2;
 - Kotlin, typowany statycznie, 4.

2. Na stworzonej bazie w zadaniu 1. i umieszczonej w niej tabeli wykonaj ćwiczenia:
- Napisz zapytanie sortujące malejąco – według kolumny `ranking`.
 - Znajdź wszystkich z kolumny `jezyk`, którzy mają identyfikator id nieparzysty.
 - Znajdź wszystkich z kolumny `jezyk`, którzy zawierają w nazwie literę s.
 - Znajdź najwyższą ocenę z kolumny `ranking`.
 - Napisz zapytanie, które zwróci `jezyk` i `typ\_jezyka` w połączeniu z oceną wyższą niż średnia z sumy wszystkich pozycji `ranking`.

8

Inne podstawowe typy danych w programowaniu

ZAGADNIENIA

- Typy danych – podsumowanie
- Kolejne ważne typy danych
- ASCII a UNICODE

8.1. Typy danych – podsumowanie

Typów danych w programowaniu jest bardzo dużo. Niestety, w poszczególnych językach programowania różnią się one od siebie, czasem dość znacznie. Jest to jeden z powodów, dla których osoby programujące w kilku językach powinny systematycznie utrzymywać wiedzę z tego zakresu. (Chyba że informatyk używa tylko jednego języka, ale wówczas jest dość ograniczony i trudno takiej osobie znaleźć dobrą i rozwojową pracę).

```
#include <iostream>
#include <locale.h>

//using namespace std;

int main()
{
    setlocale(LC_ALL, "");

    std::cout << "Witaj w świecie" <<
        std::endl << "polskich ogonków"
        << std::endl;
    std::cout << "Polskie fonty: ń,
        ć, ź, itp. - w C++. " << std::
        endl;
    return 0;
}
```

```
Witaj w świecie
polskich ogonków
Polskie fonty ń, ć, ś, itp. - w C++.
```

```
Process returned 0 (0x0)    execution time
: 3.443 s
Press any key to continue.
```

Rys. 8.1. Widok kodu napisanego w języku C++ – `setlocale()` i jego wynik w konsoli



CIEKAWOSTKA

Na rys. 8.1 przedstawiono, jak za pomocą jednej małej linijki kodu

```
Setlocale(LC_ALL, "");
```

można rozwiązać problem „polskich ogonków”, czyli wyświetlania fontów typu: ą, ę, ć, ń, ó.

Skoro jest takie bogactwo, to spróbujmy podsumować wiedzę i nieco uogólnić pozyskane informacje o typach danych w programowaniu.



Rys. 8.2. Popularne języki programowania

Oto przykładowe – najważniejsze – typy występujące w wielu językach programowania:

- **typ całkowity** (m.in. w językach: C, C++, JavaScript, PHP, Python, Java) – reprezentuje liczbę całkowitą z określonego zakresu zależnego od danego języka, a czasami nawet konkretnej implementacji;
- **typ zmiennoprzecinkowy** (m.in. w językach: C, C++, JavaScript, PHP, Python, Java) – reprezentuje przybliżoną wartość liczby rzeczywistej;
- **typ znakowy** (m.in. w językach: C, C++, Java, Pascal) – reprezentuje pojedynczy znak (ASCII lub w nowszych implementacjach Unicode);
- **typ tekstowy** (m.in. w językach: C, C++, JavaScript, PHP, Python, Java) – reprezentuje cały tekst;
- **typ logiczny** (m.in. w językach: C, C++, JavaScript, PHP, Python, Java) – może przyjmować wartości logiczne 1 (true) lub 0 (false).

CIEKAWOSTKA

Literał to ustalona wartość (liczbowa, tekstowa itp.), wpisana przez programistę bezpośrednio w określonym miejscu kodu programu. Większość języków programowania ma formy zapisu dla podstawowych wartości, takich jak:

- łańcuchy znaków,
- znak,
- liczby całkowite,
- liczby zmiennoprzecinkowe,
- wartości typu logicznego.

Literały są zwykle używane do inicjalizacji zmiennych – tak jak na rys. 8.3.

Przeciwieństwem literałów są stałe i zmienne – określane jako identyfikatory.

```
/** LITERAŁY - czyli co rozumie nasz komputer ? */  
1, -2, 3, 4, -5 ;      // liczby dodatnie i ujemne  
1.211; // liczba dziesiętna, kropka jest separatorem  
0x4F; // liczba szesnastkowa  
'A'; // znak, dokładnie JEDEN znak  
"Fajny przykład tekstu"; // ciąg znaków  
  
return 0; // oznacza, że: .....
```

Rys. 8.3. Widok kodu napisanego w języku C++ – literały

8.2. Kolejne ważne typy danych

Wymienione typy danych to oczywiście nie wszystkie, które są dostępne w językach programowania. Na kolejnych stronach poznasz jeszcze kilka nowych. Tutaj jedynie wymienimy najważniejsze z jeszcze nieomawianych typów danych:

- typ wskaźnikowy,
- typ referencyjny,
- typ wyliczeniowy,
- typ tablicowy (tablica) – ciąg zmiennych,
- typ strukturalny,
- klasa (typ obiektowy),
- typ pusty.

8.3. ASCII a UNICODE

ASCII (ang. *American Standard Code for Information Interchange*) to siedmiobitowy system kodowania znaków, który jest używany w komputerach, sieciach komputerowych, a także w innych urządzeniach mających mikroprocesor. W ramach tego standardu liczbom z zakresu 0–127 przyporządkowuje on litery alfabetu łacińskiego języka angielskiego, cyfry, znaki przestankowe oraz inne symbole i polecenia sterujące. Na przykład znak:

- „c” jest kodowany jako liczba 99;
- „C” jest kodowany jako liczba 67;
- ! jest kodowany jako 33;
- spacja jest kodowana jako 32.



Rys. 8.4. ASCII – system kodowania znaków

Współczesne systemy kodowania znaków są zwykle rozszerzeniem standardu ASCII.

Unicode

Jest to komputerowy zestaw znaków, który w zamierzeniu twórców powinien obejmować wszystkie pisma używane na świecie. Jest on powszechnie wykorzystywanym standardem kodowania. Doczekał się wielu standardów i metod kodowania (jedne z najpopularniejszych są oznaczane jako UTF-32/UCS-4, UTF-16, UTF-8).



Rys. 8.5. UNICODE – system kodowania znaków

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest typ całkowity? Czym się różni od typu zmiennoprzecinkowego?
2. Wymień kilka ważnych, ale jeszcze w tej książce szczegółowo nieomawianych, typów danych.
3. Czy typ logiczny istnieje w językach programowania: C, C++, JavaScript, PHP, Python, Java? Odpowiedź uzasadnij.
4. Wyjaśnij, czym jest literał w programowaniu.
5. Czym się różni ASCII od UNICODE?
6. Znajdź w internecie tabelę znaków ASCII i sprawdź, jakie liczby odpowiadają znakom: a, G, k, P, #, \$, L, n.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Kod zawarty na rys. 8.6 napisano w języku programowania C++. Niestety, nie działa. Brakuje w nim importu (zainkludowania) kilku istotnych bibliotek. Sprawdź, których bibliotek brakuje. Dodaj je do kodu i napisz program, który losowo wybiera 10 cyfr.

```
int main()
{
    srand( time( NULL ) );

    for( int i = 0; i < 10; i++ )
        printf("Wylosowano %d\n", rand());

    getch();
}
```

Rys. 8.6. Program wskazujący losowe cyfry – brak w nim kilku istotnych linii kodu

III. Złożone typy danych

- Złożone typy danych – charakterystyka
- Tablice jednowymiarowe i dwuwymiarowe
- Tablice dynamiczne i asocjacyjne
- Typy rekordowe
- Wskaźniki
- Typy plikowe
- Kolekcje

9

Złożone typy danych – charakterystyka

ZAGADNIENIA

- Charakterystyka najważniejszych złożonych typów danych

9.1. Najważniejsze złożone typy danych

Wymienione w poprzednich rozdziałach typy danych to oczywiście nie wszystkie, jakie są dostępne w językach programowania. Na kolejnych stronach poznasz jeszcze kilka innych. Tutaj jedynie wymienimy najważniejsze z jeszcze nieomawianych typów danych:

- **typ wskaźnikowy** – typ zmiennej odpowiedzialnej za przechowywanie adresu do innej zmiennej (innego miejsca w pamięci) w obrębie naszej aplikacji, oznacza wskaźnik na zmienną danego typu;

CIEKAWOSTKA

W wielu językach programowania można tworzyć tablice wielowymiarowe. Jednym z nich jest C++. Na rys. 9.1 znajdziesz przykłady tablic zarówno jedno-, jak i dwuwymiarowych.

```
// Zmienne - typy złożone,  
// przykład dla języka C++  
  
int a[_size] = { 0,1,2,3,4 };  
int b[5] = { 8 };  
int c[] = { 0,1,2,3,4 };  
  
int d[3][2] = { {1,3},{4,5},{0,-1} };  
int e[][2] = { {1,3},{4,5},{0,-1} };
```

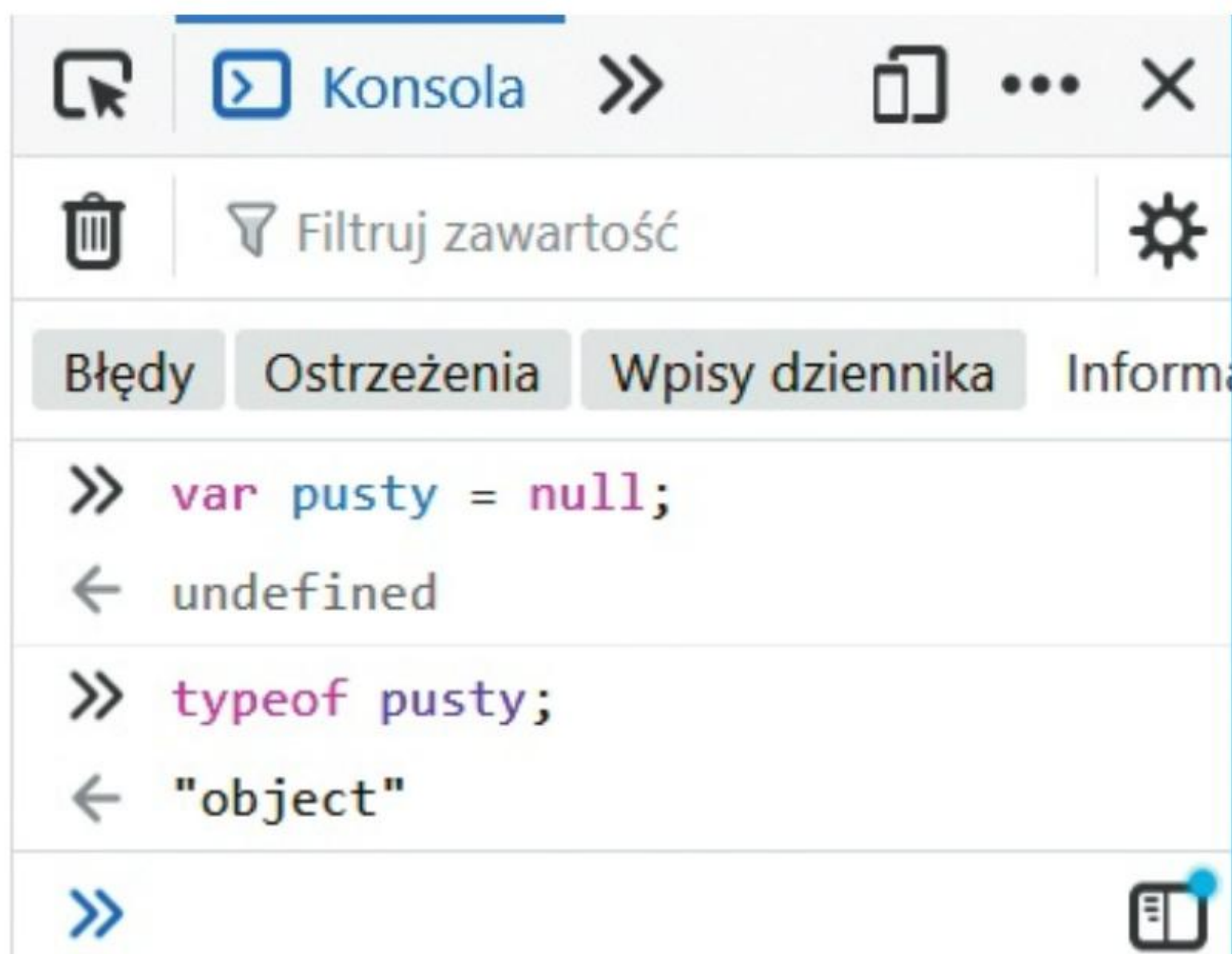
Rys. 9.1. Przykład zmiennych – język C++

- **typ referencyjny** – jest to odmiana wskaźnika, referencja jest różnorodnie interpretowana w różnych językach, ale zwykle oznacza wartość, która zawiera informacje o położeniu innej wartości w pamięci lub nośniku danych; w odróżnieniu od wskaźników zarządzanie referencjami jest realizowane wyłącznie przez kompilator lub interpreter, a programista nie ma żadnych informacji o konkretnym sposobie implementacji referencji;
- **klasa (typ obiektowy)** – określa dwa podstawowe elementy: pola (właściwości) i metody (zachowania, działania), na jej bazie są tworzone instancje obiektów, będące samodzielnymi bytami, zwykle rozumiana jako odmiana wspomnianej już struktury, z tym że oprócz zespołu danych są tu dodane działania (procedury, metody), operujące na wskazanych danych (obiekt jest nazywany zmienną typu klasa lub po prostu instancją klasy – jego utworzenie przyczyniło się do powstania i bardzo szybkiego rozwoju tzw. programowania obiektowego);

```
const car = {  
  brand: "electricCar",  
  color: "red",  
  maxSpeed: 290,  
  owner: "Janina Nowak",  
  ownerAge: 30  
}
```

Rys. 9.2. Przykład obiektu – język JavaScript/ES6

- **typ wyliczeniowy** – rodzaj typu danych zawierający listę wartości reprezentowanych za pomocą literałów wyliczeniowych, jakie może przyjmować zmienna tego typu; może przyjmować jedną z zadanych symbolicznych wartości, np. biały, czarny, żółty;
- **typ tablicowy (tablica)** – ciąg zmiennych, w zależności od języka programowania tego samego lub różnego typu, indeksowanych liczbą naturalną z określonego przedziału lub innym unikalnym kluczem mogącym być zarówno liczbą, jak i ciągiem znaków, np. w PHP możemy tworzyć tablice indeksowane kolejnymi liczbami całkowitymi, licząc od zera (tablice indeksowane), i tablice asocjacyjne, które są indeksowane kluczem, czyli niemal dowolnym ciągiem znaków;
- **typ strukturalny** – jest rozumiany jako zespół połączonych w całość zmiennych określanych jako pola struktury, charakterystyczny dla języka C;
- **typ pusty** – występuje np. w oznaczaniu funkcji niezwracających żadnych wartości.



Rys. 9.3. Przykład obiektu null (pusty) – język JavaScript – widok w konsoli przeglądarki

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to są złożone typy danych?
2. Wymień co najmniej pięć najważniejszych złożonych typów danych dostępnych w różnych językach programowania.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz prosty program konsolowy w C++ (projekt konsolowy), który pobierze od użytkownika 10 liczb, zapisze je do tablicy i wyświetli na ekranie ich sumę.

10

Tablice jednowymiarowe i dwuwymiarowe

ZAGADNIENIA

- Tablice
- Tablice jednowymiarowe
- Tablice dwuwymiarowe

10.1. Tablice

W informatyce tablica jest określana jako kontener uporządkowanych danych. Komórki w tablicy są dostępne za pomocą kluczy, które mogą być:

- **wartościami numerycznymi** – mówimy wówczas o tablicy indeksowanej numerycznie;

Tablica indeksowana numerycznie

PHP

```
$nazwaTablicy = [ 3, 5, 0, 9 ];
```

wartości
indeksy

0, 1, 2, 3

Rys. 10.1. Tablica indeksowana numerycznie – przykład z języka PHP

- **dowolnymi znakami** – wówczas mamy do czynienia z tablicą asocjacyjną.

`$nazwaTablicy`

Tablica asocjacyjna to nazwa abstrakcyjnego typu danych, który przechowuje pary: unikatowy klucz (czasami nazywany indeksem nienumerycznym) i wartość. Programista uzyskuje dostęp do wartości przez podanie klucza. To narzędzie programistyczne bardzo często jest także określane jako **tablica skojarzeniowa** lub **mapa**.

Tablica asocjacyjna (skojarzeniowa)

PHP

`$nazwaTablicy = [ „aba”, 5 , 0 , null ] ;`

wartości

klucze
`ojej ziele 2 inna`

pary:

klucze	wartości
ojej	„aba”
ziele	5
2	0
inna	null

Rys. 10.2. Tablica asocjacyjna – przykład z języka PHP

Tablice mogą mieć:

- **rozmiar zmieniany dynamicznie** – może się on zmieniać w trakcie wykonywania programu, wówczas mówimy o tablicach dynamicznych – charakterystyczne dla języków typowanych dynamicznie, np. JavaScript czy PHP;
- **rozmiar ustalony z góry** – takie tablice nazywamy statycznymi, są one charakterystyczne dla języków programowania typowanych statycznie, jak C i C++.

10.2. Tablice jednowymiarowe

Występują we wszystkich istotnych językach programowania. Można powiedzieć, że jest to tablica płaska, tzn. istnieje w niej tylko jeden poziom danych. Na każdej pozycji mamy dane, do których możemy mieć bezpośredni dostęp, bez potrzeby dalszego zagłębiania się.


```
// Tablica wielowymiarowa,  
// jednorodna pod względem  
// typów danych - JavaScript  
  
let tablica = [  
  [6, 3, 5],  
  [2, 4, 5],  
  [3, 65, 98, 1],  
  [3, 7, 82]  
];
```

Rys. 10.3. Tablica indeksowana numerycznie, jednowymiarowa – jednorodna pod względem typów danych – przykład z języka JavaScript

```
// Tablice jednowymiarowe,  
// niejednorodne pod względem  
// typów danych - JavaScript  
  
var tab1 = [6, 'to', '3', 5];  
let tab2 = [10, 'PHP', 'JS', 4];
```

Rys. 10.4. Tablica indeksowana numerycznie, jednowymiarowa – niejednorodna pod względem typów danych – przykład z języka JavaScript

10.3. Tablice dwuwymiarowe

Tak naprawdę ten dział mógłby się nazywać „Tablice wielowymiarowe”, ponieważ w wielu językach programowania możemy używać tablic wielowymiarowych, choć nie jest to, niestety, zbyt łatwe do opanowania.

Tablica dwuwymiarowa to tablica w tablicy (a nawet tablice w tablicy). Kody zawarte na rys. 10.5 i 10.6 wyjaśniają to zagadnienie.


```
// Tablica jednowymiarowa,  
// jednorodna pod względem  
// typów danych - JavaScript  
  
let tablica = [1, 2, 3, 4, 5];  
  
console.log(...tablica);
```

Rys. 10.5. Tablica indeksowana numerycznie, wielowymiarowa – jednorodna pod względem typów danych – przykład z języka JavaScript

```
// Tablica wielowymiarowa,  
// niejednorodna pod względem  
// typów danych - JavaScript  
  
let tablica = [  
  [6, 'to', '3', 5],  
  [2, 'C++', 5],  
  ['3', 'hej', 98, 1],  
  [3, 7, 'JavaScript']  
];
```

Rys. 10.6. Tablica indeksowana numerycznie, wielowymiarowa – niejednorodna pod względem typów danych – przykład z języka JavaScript

 SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest tablica w programowaniu?
2. Podaj przykład tablicy indeksowanej numerycznie jednowymiarowej – jednorodnej.
3. Podaj przykład tablicy indeksowanej numerycznie jednowymiarowej – niejednorodnej.
4. Podaj przykład tablicy indeksowanej numerycznie dwuwymiarowej – jednorodnej.
5. Podaj przykład tablicy indeksowanej numerycznie dwuwymiarowej – niejednorodnej.

 SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz prosty program w C++ (projekt konsolowy), który pobierze od użytkownika pięć punktów w układzie kartezjańskim (x, y) , zapisze je do tablicy dwuwymiarowej i wyświetli na ekranie każdą parę w formacie „ (x, y) ”. Każdy punkt wyświetl w osobnej linii.

11

Tablice dynamiczne i asocjacyjne

ZAGADNIENIA

- Tablice dynamiczne
- Tablice asocjacyjne

11.1. Tablice dynamiczne

Tablice, które mają rozmiar ustalony z góry, nazywamy **statycznymi**. Jak już wspomniano, są one charakterystyczne dla języków programowania typowanych statycznie, m.in. C, Java, C# i C++. Nie oznacza to, że w tych językach nie ma dynamizmu w tablicach, czyli tablic dynamicznych. Ich idea jest jednak zupełnie inna niż w językach typowanych dynamicznie.

Przykładowo: w C++ tablice dynamiczne są tworzone w czasie uruchomienia programu. Rozmiar takiej tablicy jest wyliczany, a gdy przestanie być potrzebna do dalszego działania napisanego programu, to można ją usunąć z pamięci. Takie własności tablicy dynamicznej sprawiają, że program wykorzystuje zasoby pamięciowe komputera w sposób bardziej zoptymalizowany.

Na rys. 11.1 znajdziesz opis, jak krok po kroku przygotować tablicę dynamiczną. Zapewne znasz już podstawy języka C++, ale nawet wtedy możesz mieć problem i nie wszystko być może zrozumiesz z poniższej instrukcji. Wróć do tego miejsca, gdy przerobisz kolejne elementy dotyczące programowania. Szczególnie dotyczy to tzw. wskaźników. Wówczas będzie na pewno dużo łatwiej zrozumieć zagadnienie tablicy dynamicznej języka programowania C++.



CIEKAWOSTKA

Istnieją setki (a może są ich już nawet tysiące) języków programowania. Można w nich znaleźć wiele różnorodnych konstrukcji i narzędzi programistycznych. W wielu z nich poszczególne powyżej wymienione elementy działają dość podobnie. Nie zawsze stosowane w nich nazewnictwo jest jednak takie samo; bywa, że nawet nie jest spójne. Przykładem mogą być tablice, które w języku programowania Python mają nazwy:

- lista (odpowiednik tablicy indeksowanej numerycznie): `list = [1, 2, 3]`
- krotka: `krotDg = (1,2,3)`
- słownik (odpowiednik tablicy asocjacyjnej): `sloownik = {klucz:wartość, klucz:wartość }`


```
#include <iostream>
using namespace std;

int main()
{
    int i, n;
    int* tablica;          // wskaźnik na int

    cout << "Podaj liczbę naturalną: \n";
    cin >> n;              // odczytujemy rozmiar tablicy

    // rezerwujemy obszar pamięci - to jest TABLICA DYNAMICZNA
    tablica = new int[n];

    // tablicę wypełniamy kolejnymi liczbami od 3
    for (i = 0; i < n; i++) {
        tablica[i] = 2 + i;
    }

    // wyświetlamy zawartość komórek tablicy dynamicznej
    for (i = 0; i < n; i++) {
        cout << tablica[i] << endl;
    }

    // zwalniamy przydzielony obszar
    delete[] tablica;

    return 0;
}
```

Rys. 11.1. Tablica dynamiczna – opis stworzenia w języku C++

11.2. Tablice asocjacyjne

O ile tablice indeksowane numerycznie są dość podobne w poszczególnych językach programowania, o tyle nie można tego już powiedzieć o tablicy asocjacyjnej (istnieje sporo jej odmian). Najważniejsze na początkowym etapie nauki jest zapoznanie się z jej konstrukcją programistyczną – zasadami jej budowy. Przedstawiono je dość dokładnie na rys. 11.2.

Podstawowa różnica między tą tablicą a tablicą indeksowaną numerycznie dotyczy indeksów, nazywanych także kluczami. W tablicy asocjacyjnej klucze mogą być niemal dowolnymi znakami. Pobieranie poszczególnych elementów tablic działa podobnie, tyle że w tym przypadku pobieramy je za pomocą klucza nienumerycznego.

Tablica asocjacyjna (skojarzeniowa)

JavaScript

```
let nazwaTablicy = [ „aba”, 5 , 0 , null ] ;
```

wartości
indeksy

ojej ziele 2 inna

pary:

ojej	„aba”
ziele	5
2	0
inna	null

Użycie tablicy:

```
nazwaTablicy [ ojej ]
```

```
nazwaTablicy [ 2 ]
```

autor: Dariusz Gołębiowski

Rys. 11.2. Tablica asocjacyjna, określana także jako skojarzeniowa

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij pojęcie tablicy asocjacyjnej.
2. Czym się różni tablica dynamiczna w C++ od tablic dynamicznych w językach programowania typowanych dynamicznie (np. JavaScript czy PHP)?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz prosty program w języku programowania C++ (projekt konsolowy), który pobierze od użytkownika liczbę naturalną N i z jej użyciem utworzy tablicę liczb typu double oraz pobierze od użytkownika N liczb. Liczby wyświetl na ekranie.

12

Typy rekordowe

ZAGADNIENIA

- Struct
- Unia

12.1. Struct

Typy rekordowe to struktury, które zawierają kilka typów prostych (bądź prostych i złożonych) w swoim wnętrzu. Można je przyrównać do pudełka, w którym przechowujemy różnego typu rzeczy i od czasu do czasu chcemy je z niego wyciągnąć. Nie mamy więc do czynienia z jedną zmienną, lecz typem, w którym możemy przechowywać różne zmienne.

Struct to typowa struktura, w której możemy umieścić wiele danych i mieć do nich dostęp dopóty, dopóki pozwala nam na to operator zasięgu. Domyślnym operatorem zasięgu dla struktur w C++ jest `public`, ale możemy ustawić dostęp na **protected** lub też **private** (w C++ 11 i wyższych, we wcześniejszych wersjach języka różnie to bywało). Aby dostać się do `private`, potrzebujemy struktury / klasy zaprzyjaźnionej, a aby dostać się do `protected`, możemy te wartości pobrać ze struktury / klasy dziedziczącej.

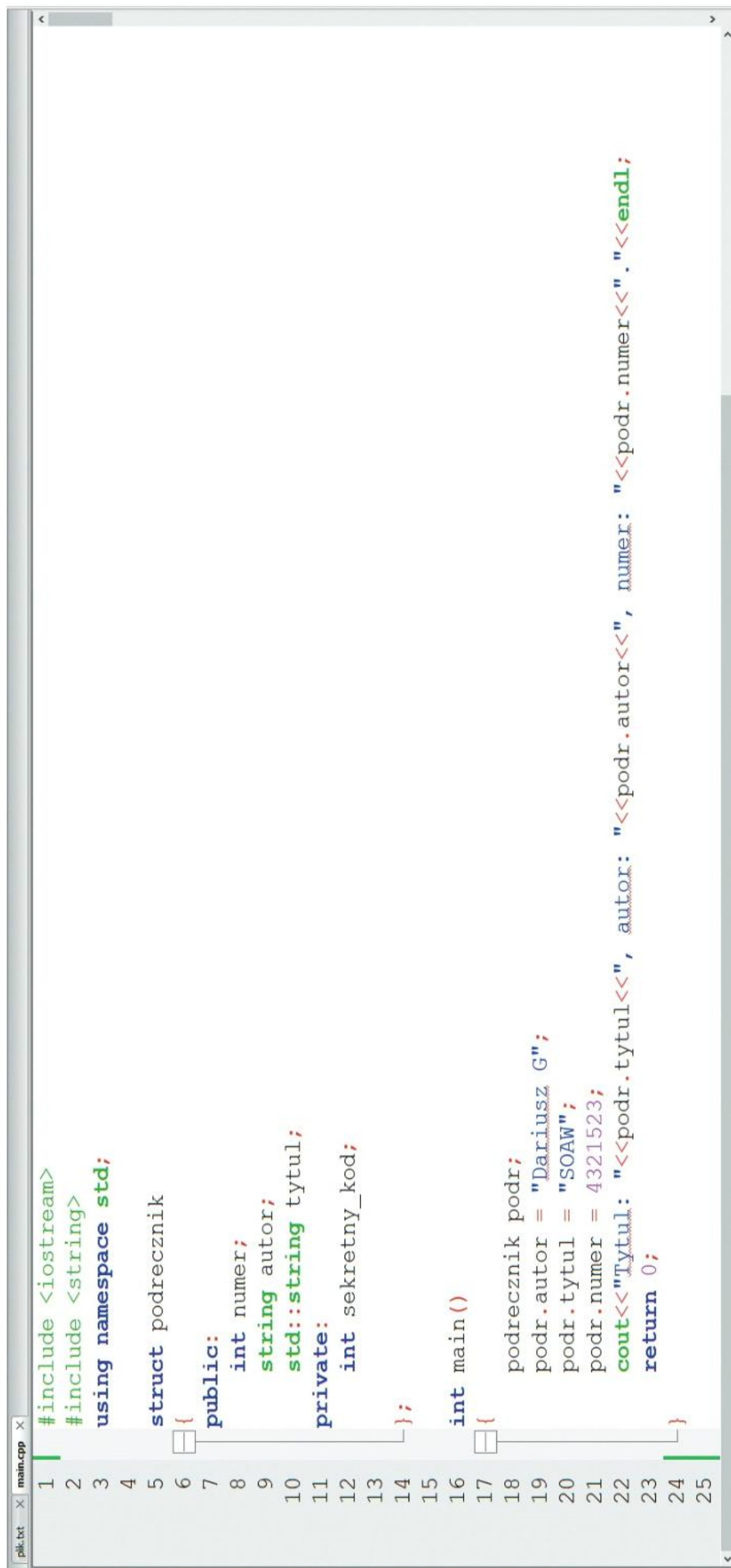
Mamy teraz dostęp do wszystkich pól i możemy je dowolnie zmieniać. Jest to wygodne rozwiązanie do przechowywania wielu typów danych, jeżeli nie zależy nam na ich bezpieczeństwie. W innym przypadku powinniśmy użyć klasy, aby zapewnić odpowiednie dostępy do wewnętrznych typów.

 CIEKAWOSTKA

Język C++ został stworzony w latach osiemdziesiątych XX wieku. Jest to jeden z najpotężniejszych i najbardziej rozpowszechnionych języków programowania. Znajduje zastosowanie – prawdopodobnie – w każdej dziedzinie przemysłu, a także w wielu obszarach codziennego życia. W opinii wielu fachowców z obszaru IT jest to najlepszy język programowania. Oczywiście pojawiają się też zdania odmienne, m.in. ze względu na stopień skomplikowania i trudność w przyswojeniu oraz wykorzystaniu przez przeciętnego programistę. Jak podano na stronie internetowej twórcy tego języka:

„C++ jest językiem programowania ogólnego przeznaczenia, z nastawieniem na programowanie systemów. [C++]:

- jest lepszym C;
- obsługuje abstrakcję danych;
- obsługuje programowanie obiektowe;
- obsługuje programowanie ogólne” (źródło: <https://www.stroustrup.com/>).



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct podrecznik
6 {
7     public:
8         int numer;
9         string autor;
10        std::string tytul;
11    private:
12        int sekretny_kod;
13
14 };
15
16 int main()
17 {
18     podrecznik podr;
19     podr.autor = "Dariusz G";
20     podr.tytul = "SOAW";
21     podr.numer = 4321523;
22     cout<<"Tytul: "<<podr.tytul<<"", autor: "<<podr.autor<<"", numer: "<<podr.numer<<" "<<endl;
23     return 0;
24 }
25
```

Rys. 12.1. Przykładowa struktura w C++

Powyższy kod zwróci nam:

```
D:\SOAW\SOAW\bin\Release\SOAW.exe
Tytuł: SOAW, autor: Dariusz G, numer: 4321523.
Process returned 0 (0x0) execution time : 0.085 s
Press any key to continue.
```

Rys. 12.2. Wynik działania programu

12.2. Unia

Jest to struktura pozwalająca zaoszczędzić pamięć przez zawieranie różnych typów prostych, choć działająca w jednym obszarze pamięci. Dzięki temu mamy dostęp tylko do ostatnio przypisanego typu z unii. Podczas próby pozyskania innego typu program zwróci nam „śmieci z pamięci”. W **unii** nie możemy umieszczać obiektów typu string.

Wielkość unii jest taka jak największego typu w jej wnętrzu, więc jeżeli mamy w niej np. typ short i typ int, to unia będzie zajmowała w pamięci wielkość inta. Oczywiście wielkość int jest zależna od procesora i kompilatora C++. Unia została stworzona głównie z myślą o zajmowaniu jak najmniejszej ilości pamięci.

```
#include <iostream>
#include <string>
using namespace std;

union komorka_pamieci
{
    short num1;
    int num2;
};

int main()
{
    komorka_pamieci kp;
    kp.num1 = 123;
    kp.num2 = 5368;
    cout<<"Short: "<<kp.num1<<", int: "<<kp.num2<<". "<<endl;
    return 0;
}
```

Rys. 12.3. Przykład unii w C++

Po użyciu powyższego kodu w konsoli zobaczymy:

A screenshot of a Windows command prompt window. The title bar at the top shows the file path 'D:\SOAW\SOAW\bin\Release\SOAW.exe'. The main area of the window displays the following text in a monospaced font: 'Short: 1233, int: 1233.', followed by 'Process returned 0 (0x0) execution time : 0.085 s', and finally 'Press any key to continue.'.

Rys. 12.4. Wynik działania programu

Jak widzisz, straciliśmy zawartość pola num1. W tym konkretnym przypadku jest wyświetlona zawartość pola typu int, lecz przy innym kompilatorze możemy dostać „śmieci” z pamięci komputera, więc podczas pracy na uniach zawsze musimy pamiętać, co robimy i w jakiej kolejności, aby nie stracić części informacji przed jej użyciem.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest struktura?
2. Czym różni się unia od struktury?
3. Ile zmiennych można przechowywać jednocześnie w unii?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz w języku programowania C++ (projekt konsolowy) własną strukturę, zawierającą dwie zmienne typu zmiennoprzecinkowego oznaczające liczbę zespoloną (część rzeczywistą i część urojoną).
2. Zaprogramuj własny dynamiczny typ zmiennej w języku programowania C++ (projekt konsolowy) przez utworzenie unii o zawartości każdego typu prostego, który poznałeś.

13

Wskaźniki

ZAGADNIENIA

■ Wskaźniki

13.1. Rola wskaźników

Wskaźniki to jedno z najmniej zrozumiałych pojęć z zakresu programowania w języku C++ wśród młodych adeptów tej technologii. Zapewne wiąże się to z poziomem skomplikowania, ale chyba także z próbą nauczania się wskaźników „na pamięć”. Tymczasem, aby je właściwie zrozumieć, trzeba zapoznać się z teorią, a następnie po prostu poćwiczyć ich praktyczne zastosowanie, co jest zalecane każdemu kandydatowi na programistę.

Wskaźnik jest zmienną, w której przechowujemy adres w pamięci. W tym adresie zaczyna się początek naszego typu (zmiennej bądź obiektu).

W C++ tworzymy wskaźnik za pomocą operatora gwiazdki – „*”. Mamy dwie możliwości tworzenia wskaźników:

- pierwszą w postaci przypisania wartości przy deklaracji wskaźnika, wtedy używamy konstrukcji: **typ * nazwa = &zmienna**;
- drugi – w postaci deklaracji pustego wskaźnika, a potem przypisania do niego naszej zmiennej w odrębnej instrukcji.

Wskaźnik zawiera adres zmiennej, co oznacza, że jeżeli spróbujemy go wypisać, to zwróci nam adres w pamięci. Jeżeli chcemy uzyskać wartość, która skrywa się pod tym adresem, trzeba użyć operatora „wyłuskania”. W C++ operatorem „wyłuskania”, czyli pobrania wartości tej komórki pamięci, jest znak **&**.

Wskaźniki w statycznym języku C++ to w programowaniu jeden z najtrudniejszych do zrozumienia elementów. Trudno jednoznacznie stwierdzić, co jest tego przyczyną, ale prawdopodobnie chodzi o pewien poziom abstrakcji, z którym niektóre osoby po prostu sobie nie radzą.

Poniższe ćwiczenie pomoże opanować wspomnianą tematykę. Najlepiej wykonać je wspólnie z nauczycielem i innymi uczniami.

Zwizualizuj opis teoretyczny wskaźników. Posługuj się jak największą liczbą kolorów. Grafiki najlepiej przygotuj:

- na wielkim arkuszu, np. kredkami, flamastrami, lub
- w programie graficznym, np. Inkscape.


```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    //pierwszy sposób
    int num = 123;
    int* numWsk = &num;

    //drugi sposób
    double num2 = 546.2;
    double* num2Wsk;
    *num2Wsk = num2;

    cout<<"numWsk: "<<numWsk<<" "<<*numWsk<<" "<<endl;
    cout<<"num2Wsk: "<<num2Wsk<<" "<<*num2Wsk<<" "<<endl;
    return 0;
}
```

Rys. 13.1. Przykład tworzenia wskaźników w C++

Powyższy kod zwróci nam:

A screenshot of a Windows command prompt window. The title bar shows the file path 'D:\SOAW\SOAW\bin\Release\SOAW.exe'. The main area of the window displays the output of a program. The first two lines are 'numWsk: 0x6cfeec, *numWsk: 123.' and 'num2Wsk: 0, *num2Wsk: 546.2.'. The third line is 'Process returned 0 (0x0) execution time : 0.090 s'. The fourth line is 'Press any key to continue.'.

Rys. 13.2. Wynik działania programu

Dla wskaźnika „numWsk” dostaliśmy najpierw adres do komórki pamięci, a potem także jej wartość. To samo otrzymaliśmy dla wskaźnika „num2Wsk” (akurat kompilator umieścił zmienną w komórce o adresie „0”, ale jest to jak najbardziej adres komórki pamięci w naszym komputerze).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest wskaźnik?
2. Za pomocą jakiego znaku tworzymy wskaźnik w C++?
3. Do czego służy operator wyłuskania?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz w języku programowania C++ program (projekt konsolowy), który pobierze od użytkownika kolejno:
 - imię (string),
 - wiek w latach (int),
 - wysokość (float).Następnie napisz kod, który wyświetli wymienione trzy dane i ich adresy do pamięci – w trzech kolejnych liniach konsoli.

14

Typy plikowe

ZAGADNIENIA

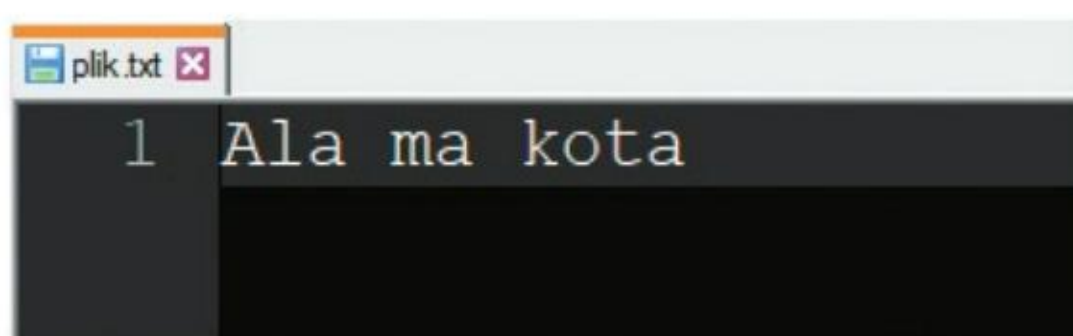
- Czytanie z pliku
- Zapis do pliku

14.1. Zagadnienia

W C++ mamy kilka bibliotek poświęconych czytaniu zawartości plików tekstowych, zapisywaniu ich oraz jednocześnie zapisywaniu i odczytywaniu. W tej lekcji omówimy użycie biblioteki **fstream**. Zawiera ona klasy **ofstream** (do zapisu do pliku) i **ifstream** (do odczytu zawartości z pliku).

14.2. Czytanie z pliku

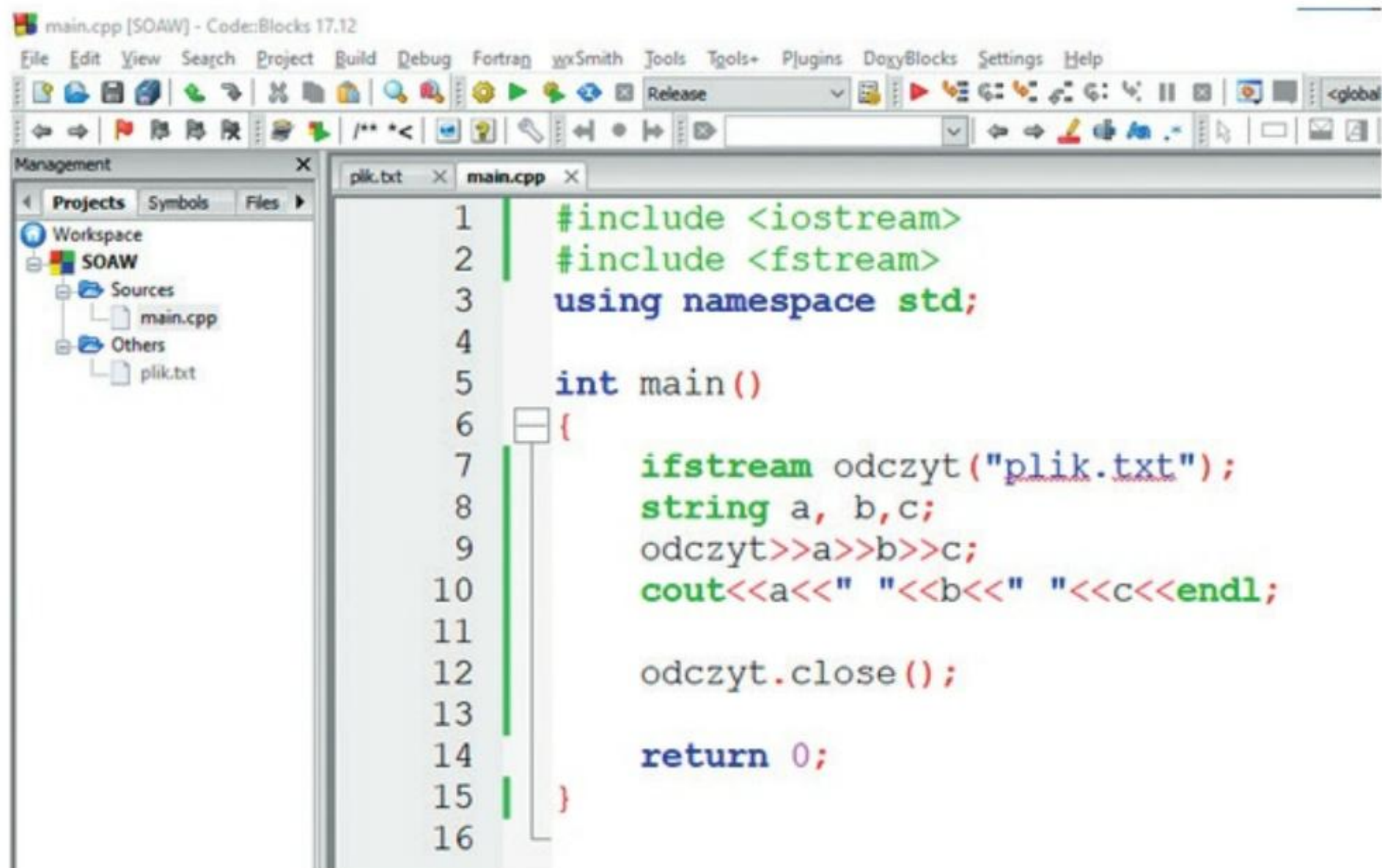
Aby odczytać dane z pliku w C++, musimy zaimportować bibliotekę **fstream** i utworzyć obiekt typu **ifstream** w postaci „**ifstream odczyt(„plik.txt”);**”. Następnie za pomocą operatora strumienia **>>** możemy pobrać po wyrazie z pliku. Operator strumienia przekazuje każdy znak, dopóki nie napotka tzw. białego znaku. Nasz plik wygląda następująco:



Rys. 14.1. Zawartość pliku plik.txt

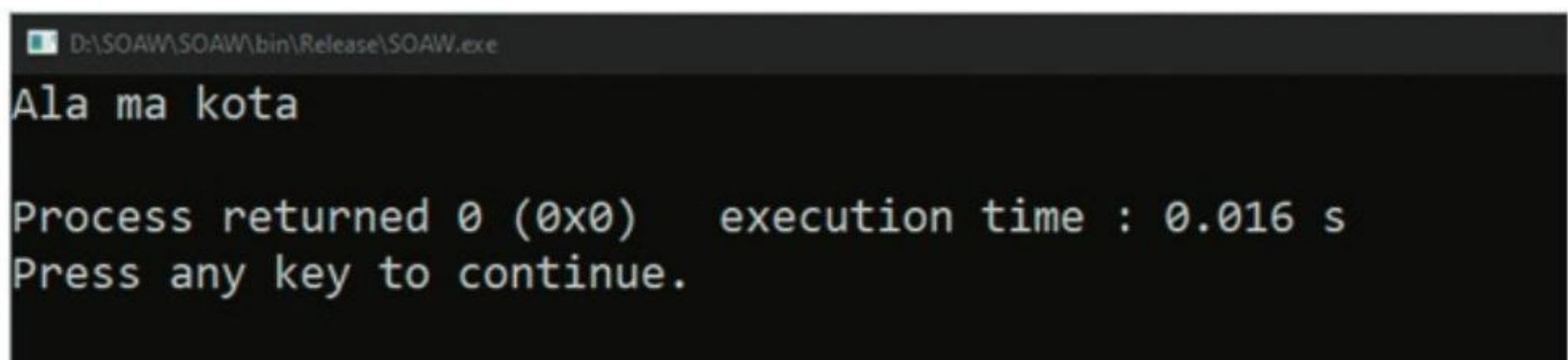
 CIEKAWOSTKA

Język programowania C++ jest następcą powstałego wcześniej języka C. Nauka w szkołach skupia się głównie na edukacji w zakresie tego pierwszego. Nie należy jednak zapominać, że język C w dalszym ciągu jest dość szeroko wykorzystywany. Szczegóły dotyczące standardu tego języka można znaleźć m.in. na stronie prowadzonej przez fundację Projektu GNU i The Free Software Foundation (FSF): <https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html>.



Rys. 14.2. Kod i układ folderów w projekcie code::blocks

Powyższy kod zwróci nam trzy stringi oddzielone spacjami. Musimy jednak pamiętać, że sczytywanie z pliku tą metodą usuwa wszystkie „białe znaki”, czyli spacje, tabulatory i znaki nowej linii. Najważniejsze w obsłudze pliku jest jego zamknięcie po zakończeniu obróbki (patrz linia nr 12 na rys. 14.2).



Rys. 14.3. Wynik działania programu

Powyższy przykład sczytuje tylko trzy pierwsze wyrazy, teraz nauczymy się, jak pobrać całą zawartość pliku. Są trzy metody pobierania danych z pliku:

- Znak po znaku

Pobieramy znak, tzn. zmienną typu char (pamiętajmy, że zmienna typu char jest zmienną liczbową wyświetlaną jako znak drukowalny według strony ASCII systemu, w którym aktualnie pracujemy).


```

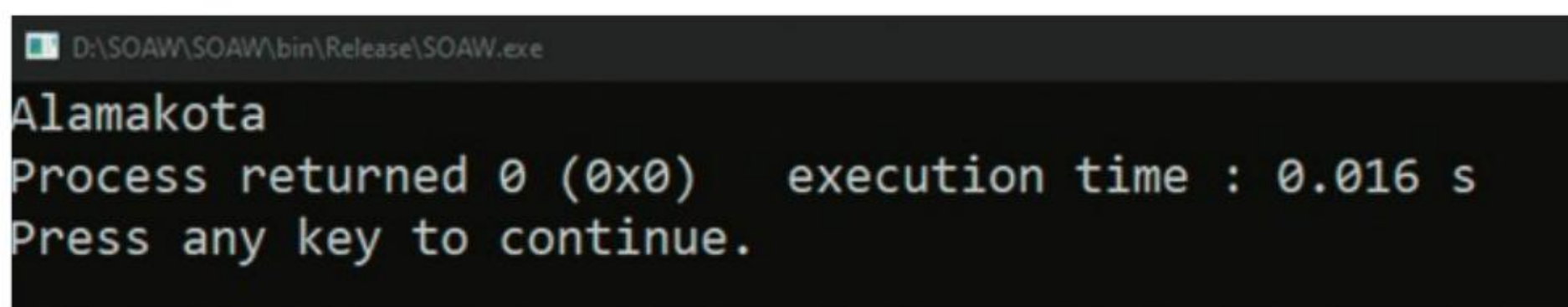
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream odczyt("plik.txt");
    char znak;
    if(odczyt.is_open()){
        while(odczyt >> znak){
            cout<<znak;
        }
    }else{
        cout<<"Podany plik nie istnieje"<<endl;
    }
    odczyt.close();

    return 0;
}

```

Rys. 14.4. Szczytywanie pliku znak po znaku



Rys. 14.5. Wynik działania programu

Jak widzisz, ten sposób również pomija białe znaki.

- Wyraz po wyrazie

Pokazaliśmy w pierwszym przykładzie sposób szczytywania na obiektach typu **string**. Teraz pokażemy, jak możemy czytać cały plik tą metodą na tablicach zmiennych typu **char**. Tablice mają pewną długość, której nie możemy przekroczyć. Musimy zadeklarować wielkość, której jesteśmy pewni, że długość wyrazu nie przekroczy. W języku polskim jednym z najdłuższych słów jest „Konstantynopolitańczykowieczka”, składa się ono z 32 znaków. Dodatkowo może wystąpić kropka lub przecinek, więc w praktyce 33 znaki. Ostatnim znakiem w tablicy musi być zawsze znak pusty „\0” – dlatego zadeklarujemy tablicę o długości 35 znaków.


```

#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream odczyt("plik.txt");
    char wyraz[35];
    if(odczyt.is_open()){
        while(odczyt >> wyraz){
            cout<<wyraz<<endl;
        }
    }else{
        cout<<"Podany plik nie istnieje"<<endl;
    }
    odczyt.close();

    return 0;
}

```

Rys. 14.6. Szczytywanie pliku wyraz po wyrazie

```

D:\SOAW\SOAW\bin\Release\SOAW.exe
Ala
ma
kota

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.

```

Rys. 14.7. Wynik działania programu

Ten sposób również usuwa białe znaki – trzeba o tym pamiętać.

- Linia po linii

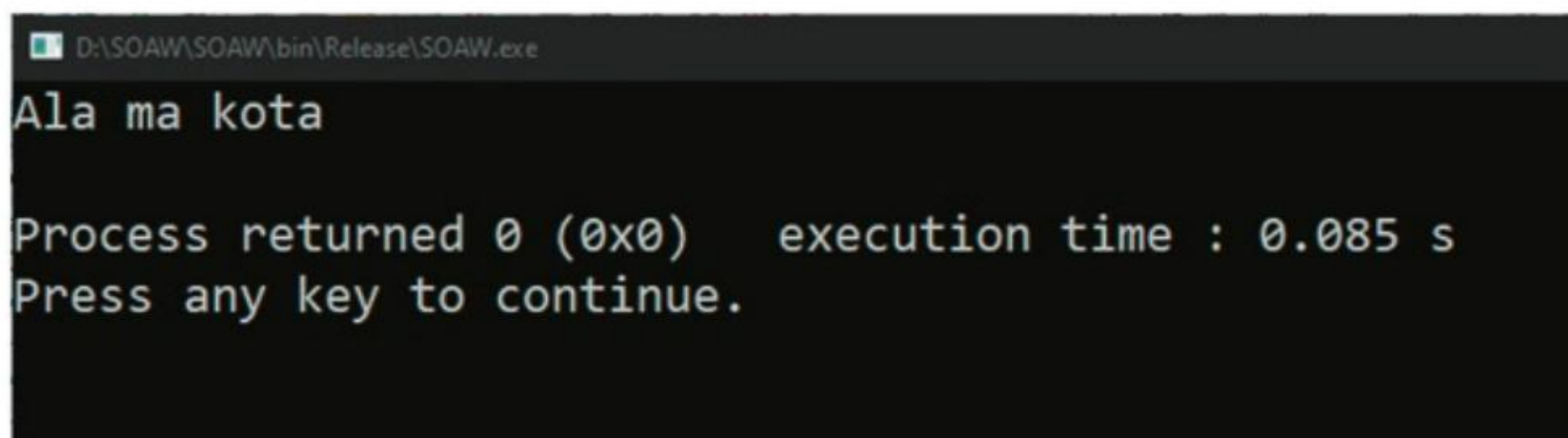
Aby pobierać całą linię, musimy zmienić operator „>>” na metodę `getline(char[], int)`. Dla naszego krótkiego pliku utworzymy tablicę wielkości 100 znaków. Operując tym sposobem na większych plikach, warto zadeklarować odpowiednio większą tablicę.


```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream odczyt("plik.txt");
    char linia[100];
    if(odczyt.is_open()){
        while(odczyt.getline(linia, 100)){
            cout<<linia<<endl;
        }
    }else{
        cout<<"Podany plik nie istnieje"<<endl;
    }
    odczyt.close();

    return 0;
}
```

Rys. 14.8. Szczytywanie pliku linia po linii



Rys. 14.9. Wynik działania programu

Powyższy program zachowuje wszystkie znaki oprócz znaku nowej linii „entera” (\n).

Podsumowując, mamy trzy sposoby szczytywania pliku do różnych celów. Można je łączyć, aby uzyskać upragniony efekt, np. kontrolę liczby wierszy i kolumn w złożonym pliku zawierającym kilka typów danych.

14.3. Zapis do pliku

Aby zapisać dane do pliku w języku C++, musimy:

- zaimportować bibliotekę **fstream**;
- utworzyć obiekt typu **ofstream** w postaci „ofstream zapisz(„plik.txt”);”;
- za pomocą operatora strumienia „<<” zapisać dane do pliku.

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream zapisz("tekst.txt");

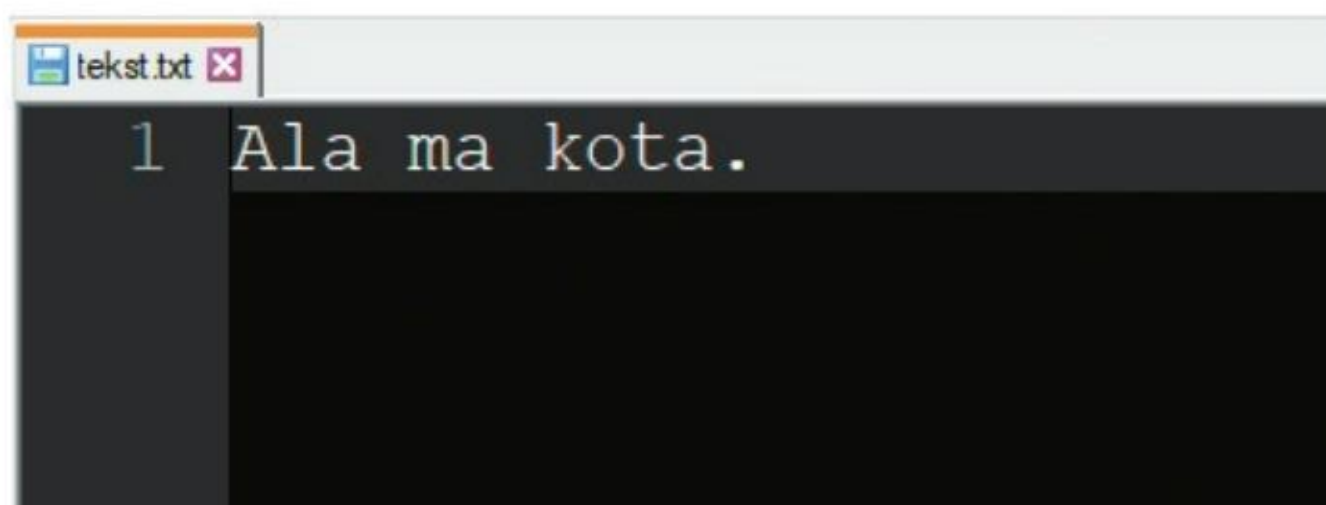
    string tekst = "Ala ma kota.";

    zapisz<<tekst ;

    zapisz.close();

    return 0;
}
```

Rys. 14.10. Zapis do pliku



Rys. 14.11. Utworzony plik

Uruchamiając ten program wiele razy i otwierając plik tekst.txt, zauważymy, że program najpierw wymazuje cały plik, a potem zapisuje nowy tekst. Aby zatrzymać wcześniejszą zawartość pliku i dodać na końcu pliku nową treść, musimy skorzystać z wbudowanej flagi w obiekt **ofstream**: „**ios_base::app**”:


```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream zapisz("tekst.txt", ios_base::app);

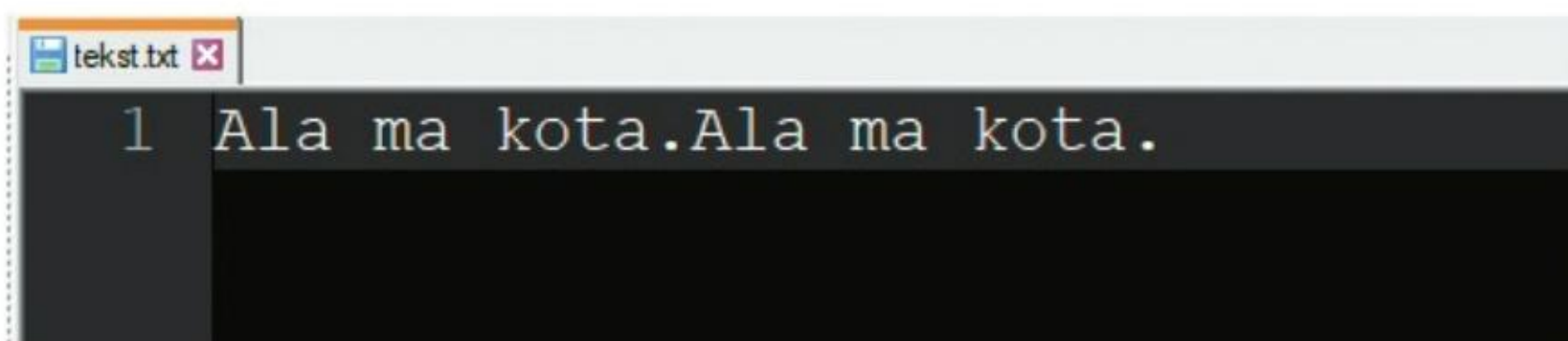
    string tekst = "Ala ma kota.";

    zapisz<<tekst ;

    zapisz.close();

    return 0;
}
```

Rys. 14.12. Zapis do pliku z zostawieniem oryginalnego tekstu



Rys. 14.13. Zawartość pliku po przejściu programu

Powyższy kod dodał nasz tekst na koniec istniejącego pliku. Jeżeli chcemy, by nasz plik pojawił się w nowej linii, wystarczy dodać znak nowej linii na początku naszego stringa „\nAla ma kota.”.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Ile jest sposobów na czytanie zawartości pliku? Opisz każdy z nich.
2. W jaki sposób zapisujemy tekst do pliku? Podaj przykład.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz program w języku programowania C++ (projekt konsolowy), który:
 - a) pobierze od użytkownika z wejścia standardowego liczbę naturalną N;
 - b) umożliwi wprowadzenie użytkownikowi N liczb;
 - c) pobrane elementy w punkcie B program zapisze do tablicy dynamicznej;
 - d) zawartość tablicy zapisze do pliku „Zadanie1.txt” w taki sposób, aby:
 - w pierwszej linii pliku znajdowała się liczba N,
 - w następnej – poszczególne elementy tablicy rozdzielone przecinkiem i spacją.
2. Za pomocą języka programowania C++ (projekt konsolowy):
 - a) wczytaj liczby podane przez użytkownika z zadania 1. (czyli zawartość drugiej linii), z utworzonego pliku „Zadanie1.txt”;
 - b) policz sumę wczytanych liczb i ich średnią, a następnie zapisz wyniki do pliku „Zadanie2.txt”:
 - w pierwszej linii ma się znaleźć suma,
 - w drugiej linii umieść średnią.
3. Za pomocą języka programowania C++ (projekt konsolowy) wczytaj do programu:
 - a) liczbę N i tablicę liczb z pliku „Zadanie1.txt”;
 - b) sumę i średnią z pliku „Zadanie2.txt”.Zapisz wszystkie wczytane liczby do nowego pliku „Zadanie3.txt” w kolejności:
 - pierwsza linia – liczebność tablicy,
 - druga linia – średnia,
 - trzecia linia – suma,
 - czwarta linia – liczby z tablicy oddzielone średnikami.

15

Kolekcje

ZAGADNIENIA

- Wektory
- Listy
- Stosy
- Kolejki

15.1. Zagadnienia

Operując na zwykłych tablicach, jesteśmy ograniczeni w niektórych czynnościach. Z tego powodu twórcy wielu języków programowania udostępnili programistom tzw. kolekcje. Ich metody, w zależności od używanego języka, mogą się trochę różnić nazwą, ale ich działanie zawsze jest takie samo.

15.2. Wektory

Wektor (ang. *vector*) to tablica dynamiczna o nieokreślonej długości dla obiektów / zmiennych jednego typu. Nie musimy się przejmować jej usuwaniem, jak w przypadku tablic dynamicznych, gdyż autorzy tej biblioteki zrobili to za nas. Mamy dostęp do każdego elementu dzięki metodzie `at(indeks)`. Biblioteka pozwala nam dodawać obiekty w każde miejsce w wektorze. Najwydajniejszym sposobem jest dodawanie elementów zawsze na koniec. Aby dodać element, użyjemy metody `push_back(obiekt)`.

Wektory mają jedną wadę – są bardzo „pamięciożerne”. Jeśli chcemy mieć wydajniejszy kod, to powinniśmy użyć listy zamiast wektora.

 CIEKAWOSTKA

Język programowania C++, podobnie jak inne tego typu projekty, ma własną organizację, która dba o jego standaryzację, promocję i rozwój:

„Goals and Purpose

Standard C++ Foundation (Standard CPP Foundation in some databases that don't support + in names) is a Washington 501(c)(6) not-for-profit organization whose purpose is to support the C++ software developer community and promote the understanding and use of modern Standard C++ on all compilers and platforms” (źródło: <https://isocpp.org/>). Więcej informacji o tym języku i planach związanych z jego rozwojem można znaleźć na oficjalnej stronie Standard C++ Foundation: <https://isocpp.org/>.


```

#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<string> wektor;
    wektor.push_back("0");
    wektor.push_back("1");
    wektor.push_back("2");
    wektor.push_back("3");
    cout<<"Wielkosc tablicy: "<<wektor.size()<<endl;
    for (int i =0;i<int(wektor.size());i++){
        cout<<wektor.at(i)<<" ";
    }
    cout<<endl;
    cout<<wektor.front()<<endl;
    cout<<wektor.back()<<endl;

    return 0;
}

```

Rys. 15.1. Przykładowe użycie kolekcji typu vector

W C++ klasa typu vector ma podstawowe metody, takie jak:

- **push_back** – dodaj element na koniec wektora;
- **size** – zwróć wielkość wektora;
- **front** – zwróć pierwszy element w wektorze;
- **back** – zwróć ostatni element w wektorze;
- **at([indeks])** – zwróć element o podanym indeksie.

```

D:\SOAW\SOAW\bin\Release\SOAW.exe
Wielkosc tablicy: 4
0 1 2 3
0
3

Process returned 0 (0x0)   execution time : 0.022 s
Press any key to continue.

```

Rys. 15.2. Wynik działania programu

15.3. Listy

Lista (ang. *list*) to obiekt typu tablicowego o nieokreślonej długości dla obiektów / zmiennych jednego typu. Główną różnicą w porównaniu z wektorem jest brak jawnego dostępu przez metodę do wewnętrznych elementów. Mamy tylko dostęp do pierwszego i ostatniego elementu. Aby uzyskać wartości z wnętrza obiektu, musimy użyć iteratora, tak jak na rys. 15.3.

```
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;

int main()
{
    int tab[5] = {0,1,2,3,4};

    std::list<int> lista;
    for(int i =0;i<5;i++){
        lista.push_back(tab[i]);
    }
    lista.push_back(5);
    lista.push_front(-1);

    cout<<"lista = {";
    for (list<int>::iterator it=lista.begin(); it != lista.end(); it++){
        cout<<*it<<" ";
    }
    cout<<"\b\b}";

    return 0;
}
```

Rys. 15.3. Przykładowe użycie kolekcji typu list

Rys. 15.4. Wynik działania programu

W powyższym kodzie używanie listy jest mniej intuicyjne niż kolekcji typu wektor i wymaga używania iteratora. Istnieją cztery główne metody:

- **push_front()** – dodaje obiekt na początek listy;
- **push_back()** – dodaje obiekt na koniec listy;
- **begin()** – zwraca wskaźnik na początek listy (pierwszy obiekt w liście);
- **end()** – zwraca wskaźnik na ostatni obiekt znajdujący się w liście.

Mimo że lista w C++ wydaje się mniej przyjazna od wektora, to jest skonstruowana na bazie modelu „podwójnie linkowanej listy”. Dzięki temu każdy element zna tylko swojego sąsiada, co jest lepszym rozwiązaniem dla pamięci komputera, lecz musimy wówczas iterować przez część listy, aby dostać się do n-tego elementu.

15.4. Stosy

Stos (ang. *stack*) to kolekcja, w której mamy dostęp tylko do ostatnio dodanego elementu, tzw. **wierzchołka**. Stos jest oparty na LIFO (ang. *Last In First Out*). Możemy to zobrazować jako stos ciężkich książek, z którego możemy zdejmować tylko jeden element w jednym ruchu.

Każdy stos musi implementować pięć podstawowych operacji:

- **empty** – zwraca wartość true, jeżeli stos jest pusty;
- **size** – rozmiar stosu (liczba obiektów / zmiennych na stosie);
- **top** – zwraca wartość z wierzchołka;
- **pop** – usuwa ostatni element ze stosu;
- **push** – dodaje nowy element na stos.

CIEKAWOSTKA

Na stronie internetowej Standard C++ Foundation warto zwrócić uwagę m.in. na zalecane standardy w zakresie kompilacji. Dotyczy to szczególnie dobrych, darmowych kompilatorów C++, dostępnych dla wszystkich głównych platform systemu operacyjnego (źródło: <https://isocpp.org/get-started>), m.in.:

- kompilator Gnu – <http://gcc.gnu.org/>,
- Clang – http://clang.llvm.org/get_started.html,
- społeczność Visual C++ 2017 – <https://www.visualstudio.com/vs/cplusplus/>,
- Embarcadero – <https://www.embarcadero.com/free-tools/ccompiler>.


```

#include <iostream>
#include <stack>
using namespace std;

int main()
{
    int tab[5] = {0,1,2,3,4};

    std::stack<int> stos;
    for(int i =0;i<5;i++){
        stos.push(tab[i]);
    }
    cout<<"Rozmiar stosu: "<<stos.size()<<endl;
    while(!stos.empty()){
        cout<<stos.top()<<" ";
        stos.pop();
    }
    cout<<endl<<"Rozmiar stosu: "<<stos.size()<<endl;
    return 0;
}

```

Rys. 15.5. Przykładowe użycie kolekcji typu stos

```

D:\SOAW\SOAW\bin\Release\SOAW.exe
Rozmiar stosu: 5
4 3 2 1 0
Rozmiar stosu: 0

Process returned 0 (0x0)   execution time : 0.022 s
Press any key to continue.

```

Rys. 15.6. Wynik działania programu

15.5. Kolejki

Alternatywą dla stosu jest **kolejka** (ang. *queue*), oparta na FIFO (ang. *First In First Out*). Zachowanie kolejki w komputerze jest takie samo jak kolejki do kasy w sklepie. Najpierw może być obsłużony obiekt, który znalazł się na jej początku, a na końcu obiekty, które znalazły się na jej końcu. Co ciekawe, w *queue* w C++ mamy również dostęp do modyfikacji i wyświetlania ostatniego elementu (nie możemy go usunąć, ale możemy np. poprawić ostatnio dodaną zmienną, jeśli popełniliśmy jakiś błąd).

Każda kolejka musi implementować sześć operacji:

- **empty()** – zwraca true, jeżeli kolejka jest pusta;
- **size()** – zwraca wielkość kolejki;
- **front()** – zwraca zawartość pierwszego miejsca w kolejce;
- **back()** – zwraca zawartość ostatniego elementu w kolejce;
- **push()** – dodaje element na koniec kolejki;
- **pop()** – usuwa pierwszy element w kolejce.

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
    int tab[5] = {0, 1, 2, 3, 4};

    std::queue<int> kolejka;
    for(int i = 0; i < 5; i++) {
        kolejka.push(tab[i]);
    }
    cout<<"Ostatni element kolejki: "<<kolejka.back()<<endl;
    cout<<"Rozmiar kolejki: "<<kolejka.size()<<endl;
    while(!kolejka.empty()) {
        cout<<kolejka.front()<<" ";
        kolejka.pop();
    }
    cout<<endl<<"Rozmiar kolejki: "<<kolejka.size()<<endl;
    return 0;
}
```

Rys. 15.7. Przykładowe użycie kolekcji typu kolejka


```
D:\SOAW\SOAW\bin\Release\SOAW.exe
Ostatni element kolejki: 4
Rozmiar kolejki: 5
0 1 2 3 4
Rozmiar kolejki: 0

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Rys. 15.8. Wynik działania programu

Najlepiej używać kolekcji typu wektor dla operacji, przy których potrzebujemy od razu n-tego elementu zbioru. Jeżeli mamy ograniczoną wielkość pamięci, warto rozważyć użycie listy.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Do których elementów wektora mamy stały dostęp?
2. Jaką przewagę ma lista nad wektorem?
3. Jakie są podstawowe operacje na stosie?
4. Wymień podstawowe operacje na kolejce.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Skorzystaj z biblioteki '**vector**' w języku programowania C++ (projekt konsolowy) i:
 - a) utwórz wektor;
 - b) wrzuć do przygotowanego powyżej wektora liczby z zakresu od 10 do 2;
 - c) wyświetl w konsoli zawartość wektora z punktu B.
2. Skorzystaj z biblioteki '**list**' w języku programowania C++ (projekt konsolowy) i:
 - a) utwórz listę obiektów typu string;
 - b) zapełnij przygotowaną w punkcie A listę opisem czterech podstawowych metod, które ma każda lista;
 - c) za pomocą iteratora wyświetl zawartość listy w kolejnych czterech liniach konsoli.

3. Skorzystaj z biblioteki '**stack**' w języku programowania C++ (projekt konsolowy) i:
 - a) utwórz stos;
 - b) zapełnij przygotowany w punkcie A stos pięcioma stringami, opisującymi pięć głównych metod zawartych w każdym stosie;
 - c) w pętli for – za pomocą metod top() i pop() – wyświetl w konsoli zawartość stosu z punktu B;
 - d) w pętli while – za pomocą metod top() i pop() – wyświetl w konsoli zawartość stosu z punktu B.
4. Skorzystaj z biblioteki '**queue**' i napisz w języku programowania C++ (projekt konsolowy) program, którego zadaniem będzie:
 - a) pobranie kolejnych sześciu liczb całkowitych od użytkownika;
 - b) wyświetlenie w konsoli danych pobranych w punkcie A.

16

Algorytmy – wprowadzenie

ZAGADNIENIA

- Algorytm
- Schemat blokowy
- Lista kroków
- Drzewo decyzyjne
- Pseudokod

16.1. Algorytm

Algorytm to jednoznacznie określony ciąg czynności potrzebnych do wykonania pewnego zadania. Termin „algorytm” pochodzi od łacińskiego słowa *algorithmus* (nazwa od przydomka perskiego matematyka Muhammada ibn Musy al-Chuwarizmiego – Algorithmi).

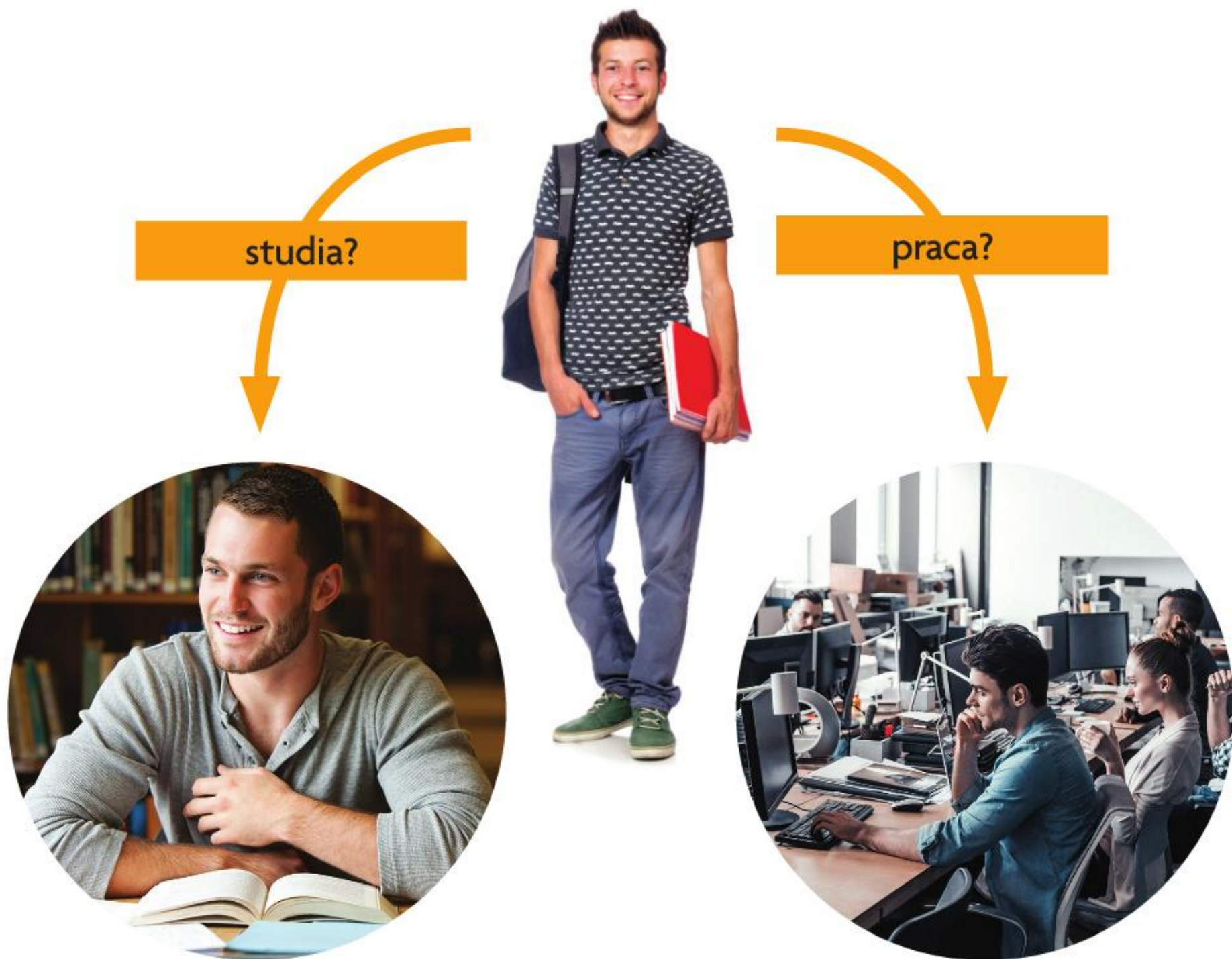
Cechy algorytmu

- skończoność – algorytm musi dojść do końca w określonym czasie, tzn. „złożoności czasowej algorytmu”;
- określoność – dla takich samych danych wejściowych algorytm powinien zawsze zwracać takie same dane wyjściowe;
- efektywność – algorytm powinien zwrócić wynik w najmniejszej możliwej liczbie kroków;
- poprawność – wynik zwrócony przez algorytm powinien być za każdym razem poprawny.

**CIEKAWOSTKA**

Zagadnienie algorytmów jest poruszane w dużej liczbie publikacji. W internecie można znaleźć wiele wartościowych pozycji. Na bezpłatnych stronach portalu Wikibooks (<https://en.wikibooks.org/wiki/Algorithms>) można zapoznać się z kompleksowym omówieniem ciekawych zagadnień.

O książce *Algorithms* napisano: „[...] obejmuje techniki projektowania i analizy algorytmów. Omówione techniki algorytmów obejmują: dziel i rządz, wycofywanie się, programowanie dynamiczne, zachłanne algorytmy i wspinanie się po wzgórzach”.



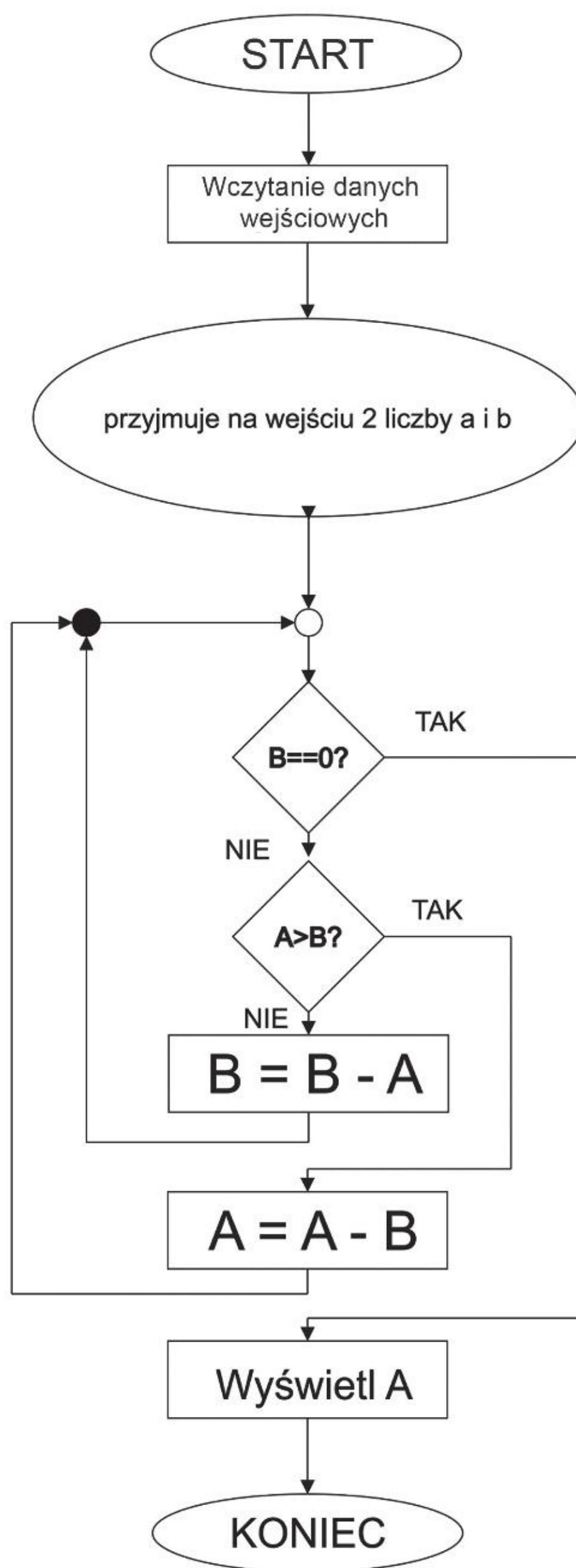
Rys. 16.1. Algorytm wyboru. Co po szkole ponadpodstawowej: studia czy praca?

16.2. Schemat blokowy

Schemat blokowy jest najprostszym sposobem zobrazowania algorytmu. Głównymi elementami schematów blokowych są:

- owal – punkt zaczęcia / skończenia / przeniesienia algorytmu;
- strzałka – pokazuje kierunek kolejności wykonywania;
- romb (predykat, instrukcja warunkowa) – w jego wnętrzu znajdują się wyłącznie instrukcje warunkowe;
- prostokąt (operator) – zawiera wszystkie możliwe obliczenia / opisu kroków niebędące wyborami.

Aby w najprostszy sposób zobrazować wygląd schematu blokowego algorytmu, posłużę się najstarszym algorytmem – algorytmem Euklidesa. Jego zadaniem jest znalezienie największego wspólnego dzielnika dwóch liczb całkowitych. Algorytm otrzymuje dwie liczby: a i b . Najpierw sprawdza, czy liczba b jest równa zero. Jeżeli jest, to wyświetli A i zakończy działanie. W przeciwnym wypadku sprawdzi, czy $A > B$. Jeżeli tak, to $A = A - B$, i wróci przed pierwszy predykat. Jeżeli zaś $B > A$, to $B = B - A$ i powróci przed pierwszy „romb” (instrukcję warunkową).



Rys. 16.2. Algorytm Euklidesa – schemat blokowy

16.3. Lista kroków

Lista kroków to lista numerowanych operacji krok po kroku, pokazujących w jednoznaczny sposób działanie algorytmu.

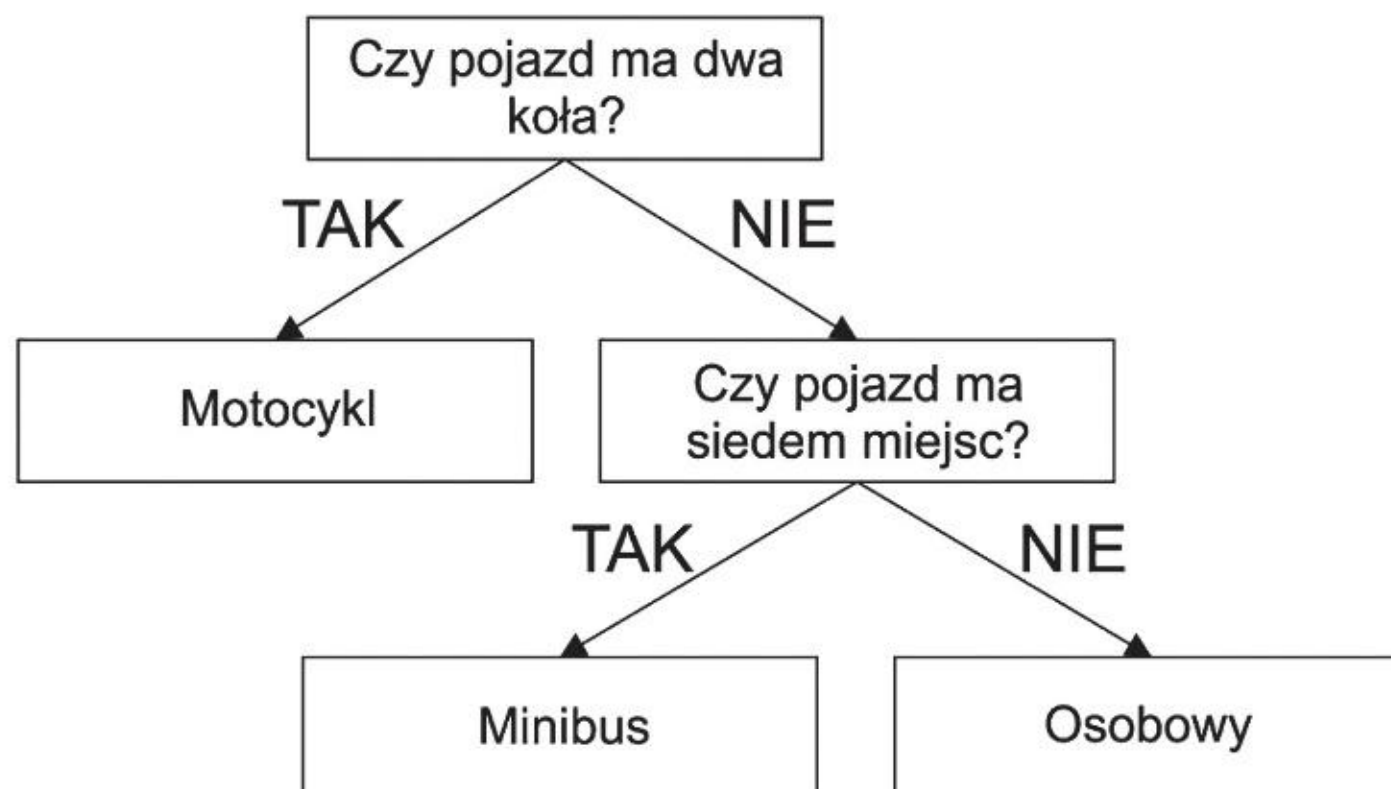
Lista kroków dla **algorytmu Euklidesa** wygląda następująco:

1. Podaj liczby A i B.
2. Jeżeli $B = 0$, idź do kroku 8.
3. Jeżeli $A > B$, idź do kroku 6.
4. $B = B - A$.
5. Idź do kroku numer 2.
6. $A = A - B$.
7. Idź do kroku numer 2.
8. Wyświetl A.
9. Koniec.

Algorytm wykonujemy krok po kroku, chyba że instrukcja nakaże przejść do innego kroku niż ten następny numerycznie.

16.4. Drzewo decyzyjne

Drzewo decyzyjne jest przedstawieniem prostych algorytmów bez pętli. Składa się z samych części warunkowych, przez co nie możemy za jego pomocą przedstawić algorytmu Euklidesa. Nadaje się ono do prostych algorytmów, np. klasyfikujących. Na rys. 16.3 przedstawiono proste drzewo decyzyjne, rozpoznające przykładowe trzy typy aut.



Rys. 16.3. Drzewo decyzyjne – prosty algorytm rozpoznawania

Oczywiście takie drzewo może pokazywać system decyzyjny dla drogi ewakuacji w szkole lub system działania prostego termostatu opartego na sztucznej inteligencji.

Drzewa decyzyjne są ograniczone i za ich pomocą nie jesteśmy w stanie przedstawić skomplikowanych algorytmów.

16.5. Pseudokod

Jest on najprostszym opisem działania algorytmu dla programisty. Niestety, nie istnieje jego konkretnie zdefiniowana postać. W związku z tym **pseudokod** w różnego rodzaju poradnikach, ale dotyczący tego samego algorytmu, może wyglądać trochę inaczej. Głównym założeniem pseudokodu jest pokazanie kolejnych kroków bez definiowania operacji konkretyzujących sposób alokacji pamięci czy działania funkcji natywnych (np. `swap()`). Pseudokod powinien być także niezależny od założeń języka programowania.

PRZYKŁAD

Przykładowy pseudokod dla algorytmu Euklidesa:

Definiuj algorytm_Euklidesa:

Wejście:

A – liczba całkowita

B – liczba całkowita

Wyjście: NWD – liczba całkowita

Dopóki $b \neq 0$:

Jeżeli $A > B$:

$A = A - B$

W przeciwnym wypadku:

$B = B - A$

Koniec, dopóki

Zwróć A.

Pseudokod w powyższym przykładzie przypomina trochę swoją strukturą języki skryptowe pokroju bash czy CLI. Jasno określa, w którym miejscu się zaczyna i w którym kończy każdy blok pętli. Może również (ale nie musi) pokazywać, w którym miejscu kończy się blok instrukcji warunkowej „jeżeli”.



SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień pięć typów przedstawienia algorytmów.
2. Jakie algorytmy można przedstawić za pomocą drzewa decyzyjnego?
3. Czy pseudokod ma jakąś standaryzację? Odpowiedź uzasadnij.
4. Wymień i opisz cztery cechy algorytmu.



SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Za pomocą sposobów przedstawienia algorytmu Euklidesa napisz w języku C++ algorytm Euklidesa liczący największy wspólny dzielnik dwóch liczb.
Podpowiedź: funkcja na wejściu powinna przyjmować dwie liczby całkowite i zwracać liczbę całkowitą.

17

Algorytmy iteracyjne, tekstowe, szyfrowania

ZAGADNIENIA

- Algorytmy iteracyjne
- Algorytmy tekstowe
- Algorytmy szyfrowania

17.1. Algorytmy iteracyjne

Jest to podstawowy rodzaj algorytmów, które wykonują pewne zadanie w określonej liczbie iteracji, czyli w określonej liczbie powtórzeń wykonują zestaw pewnych instrukcji. Jest to kilka poleceń zapisanych w jednej z pętli (np. **for**, **do...while**, **while**) i wykonanych skończoną liczbę razy. Przykładem takiego algorytmu jest m.in. algorytm liczenia potęg.

Na rys. 17.1 przedstawiono przykładowy kod algorytmu wyświetlającego trójkąt o wielkości zdefiniowanej przez użytkownika.

```
#include <iostream>
#include <string>
using namespace std;

void make_triangle(int n) {
    for(int i = 0; i<n; i++) {
        for(int j=0; j<=i; j++) {
            std::cout<<"*";
        }
        std::cout<<std::endl;
    }
}

int main()
{
    make_triangle(5);
    return 0;
}
```

Rys. 17.1. Przykład kodu: algorytm rysujący trójkąty



```
D:\SOAW\SOAW\bin\Release\SOAW.exe
*
**
***
****
*****

Process returned 0 (0x0)   execution time : 0.085 s
Press any key to continue.
```

Rys. 17.2. Wynik działania programu

Kolejnym przykładem jest prosty algorytm wyznaczający n -tą potęgę, który na wejściu przyjmuje x – podstawę i n – liczbę, do której potęgi chcemy podnieść x (rys. 17.3).

```
#include <iostream>
#include <string>
using namespace std;

int potega(int x, int n) {
    if (n==0) {
        return 1;
    }
    int temp = x;
    for (int i =1; i<n;i++) {
        temp*=x;
    }
    return temp;
}

int main()
{
    cout<<potega(2,5)<<"\n";
    cout<<potega(2,0)<<"\n";
    return 0;
}
```

Rys. 17.3. Algorytm liczący n -te potęgi liczb


```

D:\SOAW\SOAW\bin\Release\SOAW.exe
32
1
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.

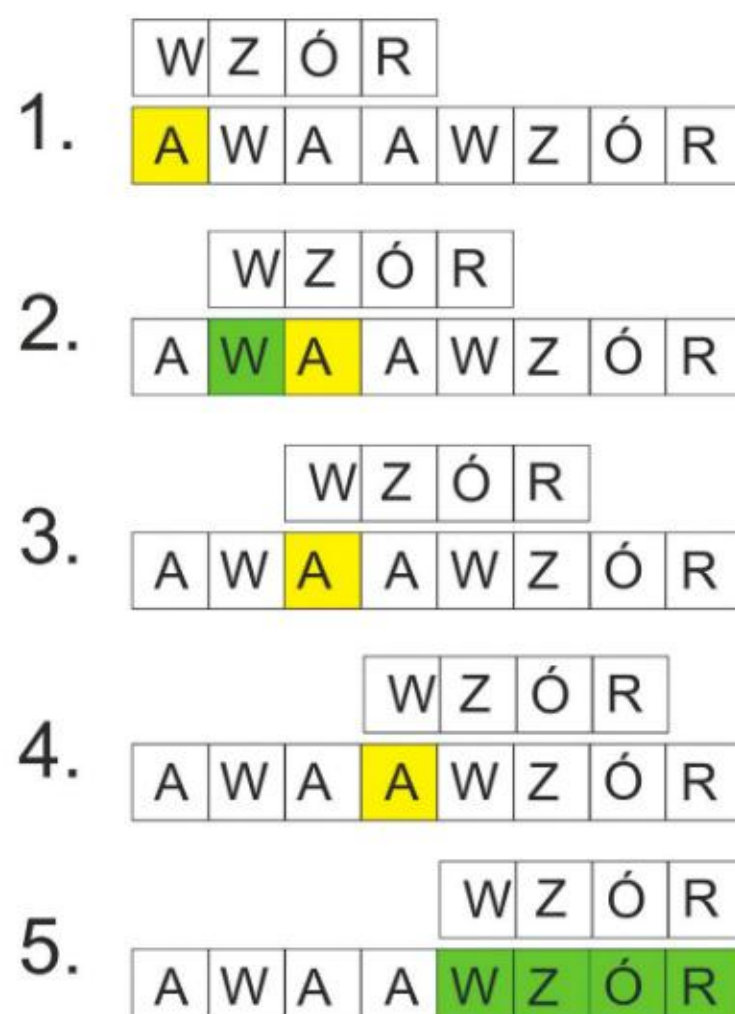
```

Rys. 17.4. Wynik działania programu

17.2. Algorytmy tekstowe

Służą one do znajdowania wzorca w tekście. Nie możemy przechodzić po całym tekście i sprawdzać każdego pojedynczego znaku, dlatego programiści wymyślili sposoby, jak znaleźć tekst w krótszym czasie. Najprostszym zoptymalizowaniem będzie przesuwanie pierwszej litery wzorca i szukanie, czy na n -tym znaku mamy poszukiwaną literę. Jeżeli ona wystąpi, to sprawdzamy kolejne. Jeżeli nie wystąpi, przesuwamy o znak dalej i zaczynamy od początku.

Na rys. 17.5 przedstawiono schemat przeszukiwania tekstu. Zaczynając od punktu nr jeden, przyrównujemy pierwszą literę wzorca do pierwszego znaku tekstu. Znaki nie są równe, więc przesuwamy wzorzec o jedno miejsce w prawo. W punkcie drugim znaki są równe, więc porównujemy drugi znak wzorca. Niestety, znaki są różne. Przesuwamy nasz wzorzec dalej w lewo i porównujemy pierwszy znak – ponownie „a” != „w”. To samo mamy w czwartym przypadku. W piątym przypadku nasz pierwszy znak pokrywa się ze wzorcem, więc sprawdzamy kolejne znaki. W ten sposób znaleźliśmy nasz wzorzec.



Rys. 17.5. Schemat przeszukiwania tekstu


```

#include <iostream>
#include <string>
using namespace std;

void szukaj(char tekst[], char wzor[]) {
    for (int i = 0; i < strlen(tekst); i++) {
        if (tekst[i] == wzor[0]) {
            for (int j = 1; j < strlen(wzor); j++) {
                if (tekst[j + i] != wzor[j]) {
                    break;
                }
                if (j == strlen(wzor) - 1) {
                    cout << "Znalazlem na: " << i << endl;
                    return;
                }
            }
        }
    }
    cout << "nie znalazlem" << endl;
}

int main()
{
    char tekst[] = "awawzor";
    char wzor[] = "wzor";
    szukaj(tekst, wzor);
    return 0;
}

```

Rys. 17.6. Implementacja prostego algorytmu do przeszukiwania tekstu

```

D:\SOAW\SOAW\bin\Release\SOAW.exe
Znalazlem na: 3
Process returned 0 (0x0)   execution time : 0.086 s
Press any key to continue.

```

Rys. 17.7. Wynik działania programu

Jest to prosty algorytm, który choć może niewiele, to jednak przyspiesza przeszukiwanie tekstu. Oczywiście istnieją dużo szybsze algorytmy, np. typu **Karp-Rabin**.

17.3. Algorytmy szyfrowania

Są to algorytmy, których zadaniem jest zamiana danych na inny format, z którego nikt poza nami nie będzie mógł wydobyć informacji, np. zamiana znaków w tekście tak, aby nie można było odczytać zawartości. Każdy algorytm szyfrujący ma taką własność, że możemy go deszyfrować, znając proces szyfrowania.

Najczęściej używane algorytmy szyfrujące to np.:

- **szyfr Cezara** – nie jest superbezpieczny, ale jest dobrym przykładem, jak działają bardziej skomplikowane algorytmy;
- **RSA** – jeden z najcięższych do rozszyfrowania algorytmów, próba złamania go zwykłym komputerem wysokiej klasy zajmie mniej więcej rok;
- **DSA** – służy do szyfrowania podpisów cyfrowych.

Szyfr Cezara to prosty i stary kod zamieniający znaki. Przesuwa alfabet o X znaków do przodu lub do tyłu. Zaimplementujemy teraz w C++ jego prostą wersję.

Kod jest bardzo prostą implementacją, na wejściu przyjmuje tylko małe znaki z zakresu a–z bez polskich znaków diakrytycznych (ą, ż, ć itp.). Ponieważ „char” jest typem liczbowym, to możemy go bez problemów zamieniać w liczby całkowite i na odwrót. Korzystając z tej właściwości, możemy zajrzeć do tabeli ASCII i dowiedzieć się, że znak „a” ma kod 97, a znak „z” ma kod 122. Dlatego w naszym alfabecie jest 122 minus 97 znaków, czyli 25 liter. Oznacza to, że nasz szyfr może „szyfrować” wiadomość na 25 sposobów. Oczywiście możemy powiększyć zakres szyfrowania, lecz musimy pamiętać, że w tabeli ASCII występują również znaki sterujące i znaki białe, które mogą sprawić problemy. Z tego powodu na rys. 17.8 ograniczyliśmy się do 25 znaków.

CIEKAWOSTKA

Propozycja ćwiczenia – algorytm na kanapkę

Dobierzcie się w pary. Jedna z osób przygotowuje przepis na kanapkę. Dane wyjściowe – na blacie leżą:

- A. dwie bułki – spodem do góry,
- B. trzy plastry pomidora – jeden na drugim,
- C. plaster żółtego sera,
- D. nóż.

Napisz taki algorytm, aby wybrana osoba przygotowała kanapkę w kolejności: bułka, a wewnątrz niej plaster pomidora na plastrze sera.

Pamiętaj! Można wykonać wyłącznie czynność zawartą w algorytmie.


```
#include <iostream>
#include <string>
using namespace std;

char * cezar(char tab[], int num) {
    int alfabet = 122 - 97;
    int i = 0;
    while (tab[i] != '\0') {
        if (tab[i] == ' '){
            i++;
            continue;
        }
        int temp = int(tab[i]) + num;
        if (temp > 122) {
            temp -= alfabet;
        } if (temp < 97) {
            temp += alfabet;
        }

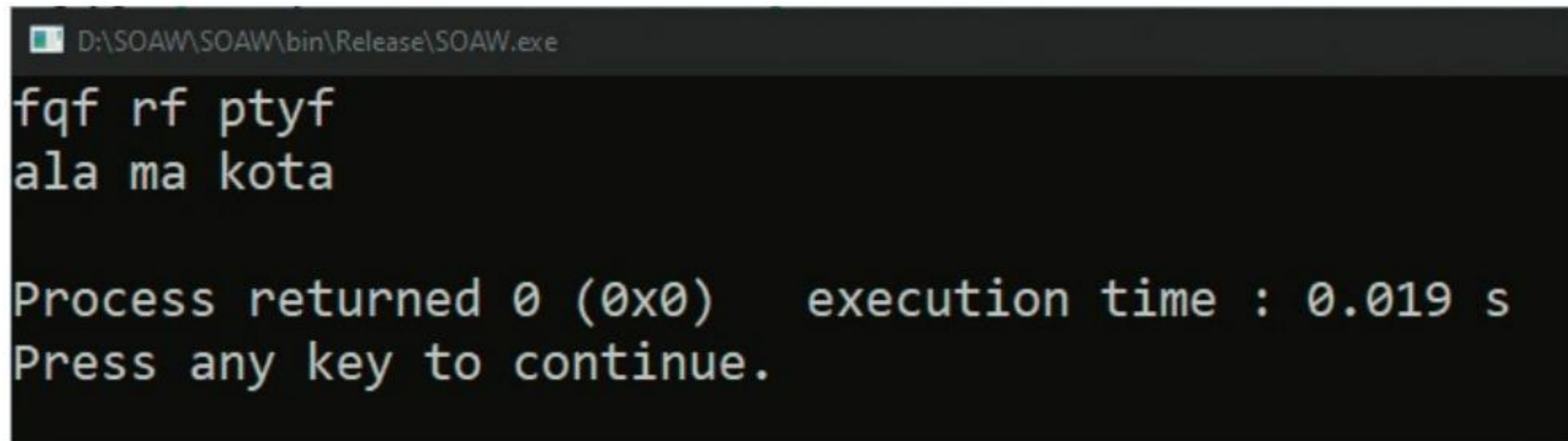
        tab[i] = char(temp);

        i++;
    }

    return tab;
}

int main()
{
    char tekst[] = "ala ma kota";
    cout<<cezar(tekst, 5)<<endl;
    cout<<cezar(tekst, -5)<<endl;
    return 0;
}
```

Rys. 17.8. Implementacja prostej wersji szyfru Cezara



```
D:\SOAW\SOAW\bin\Release\SOAW.exe
fqf rf ptyf
ala ma kota

Process returned 0 (0x0)    execution time : 0.019 s
Press any key to continue.
```

Rys. 17.9. Wynik działania programu

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień algorytmy szyfrujące.
2. Czy algorytm szyfrowania RSA jest bezpieczny? Odpowiedź uzasadnij.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Za pomocą języka C++ (projekt konsolowy) napisz program wyświetlający odwrotny trójkąt względem trójkąta z przykładu z rys. 17.2. Wynik programu powinien wyglądać następująco:

**
*
2. Rozbuduj algorytm „szyfr Cezara” o cyfry i duże litery. Kody znakowe ASCII dla cyfr: od 48 – „0” do 57 – „9”, a dla dużych liter: od 65 – „A” do 90 – „Z”.

18

Algorytmy tablicowe, rekurencyjne

ZAGADNIENIA

- Algorytmy tablicowe
- Algorytmy rekurencyjne

18.1. Algorytmy tablicowe

Istnieje wiele algorytmów do sortowania tablic, z których najpopularniejsze to:

- algorytm bąbelkowy – o złożoności czasowej $O(n^2)$;
- sortowanie przez wybór – o złożoności czasowej $O(n^2)$;
- algorytm quicksort – o złożoności czasowej $O(n \cdot \log n)$ i pesymistycznej złożoności (gdy tablica będzie posortowana odwrotnie do aktualnego sortowania) $O(n^2)$ – jest to rzadki przypadek, ale warto mieć go na uwadze.

Algorytm bąbelkowy służy do sortowania tablic. Niczym „bąbelki” pod wodą wyciąga najmniejsze wartości do „góry”, a największe / najcięższe wartości „opadają w dół”.

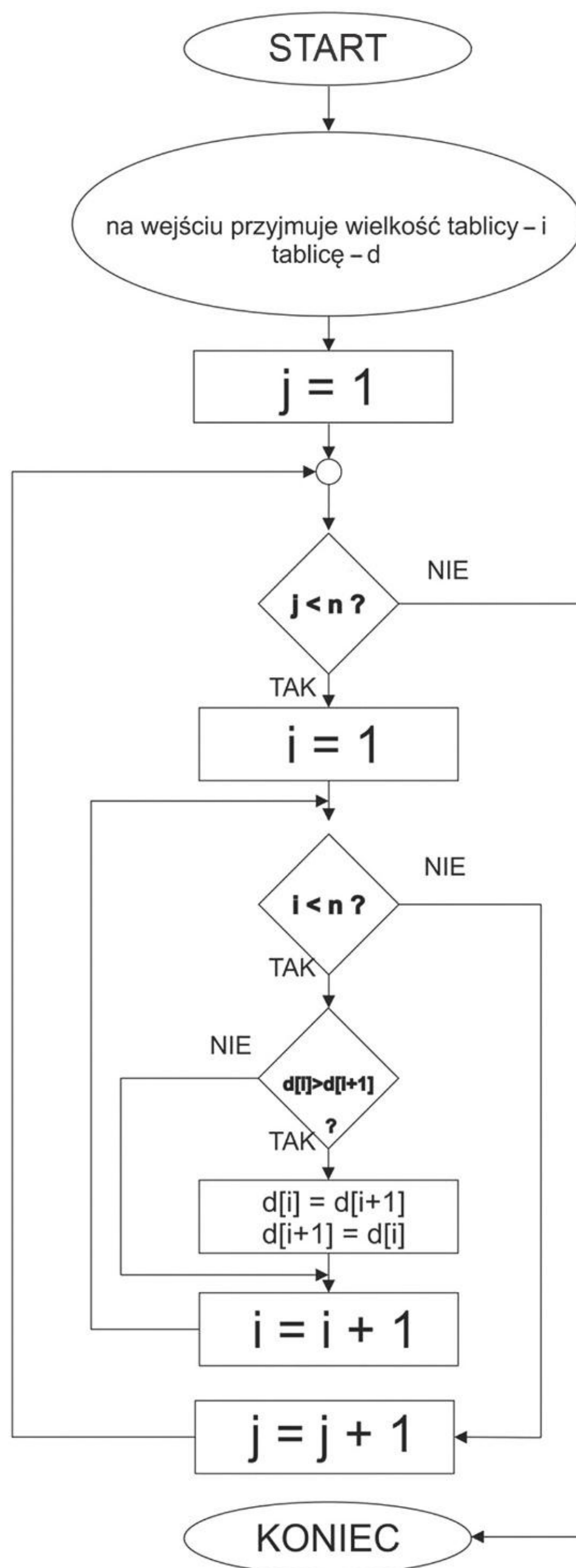
CIEKAWOSTKA

Przypomnienie!

Tablice asocjacyjne, czyli takie, których indeksy są rozpoznawane po niemal dowolnych nazwach, występują w wielu językach programowania. Niestety, JavaScript nie wspiera wprost takiego narzędzia. Obiekt Array w tym języku może być indeksowany tylko za pomocą liczb naturalnych (oczywiście zaczynając od zera). Co jednak ważne, istnieje konstrukcja językowa, która wygląda właściwie jak zwykła tablica asocjacyjna. Jest nią obiekt. Spójrz na poniższy przykład:

```
var jezykProgramowania =  
{  
  type: "dynamiczny",  
  nazwa: "JavaScript",  
  poziomTrudnosci: "sredni",  
  popularnosc: "bardzo popularny",  
  posiadaTabliceAsocjacyjne: "Nie? 😊"  
};
```

Pamiętaj, że to, co widzisz powyżej, jest jedynie przybliżonym mechanizmem dotyczącym tablic asocjacyjnych. Obiekty w JavaScript działają na nieco innych zasadach niż tablice. W programowaniu obiektowym mają zdecydowanie większe znaczenie.



Rys. 18.1. Schemat blokowy algorytmu sortowania bąbelkowego


```
#include <iostream>
#include <string>
using namespace std;

void sortowanie_bombelkowe(int tab[], int len) {
    for (int i = 0; i < len; i++) {
        for (int j = 1; j < len; j++) {
            if (tab[j - 1] > tab[j]) {
                int temp = tab[j - 1];
                tab[j - 1] = tab[j];
                tab[j] = temp;
            }
        }
    }
}

void show(int tab[], int n) {
    for (int i = 0; i < n; i++) {
        cout << tab[i] << " ";
    } cout << endl;
}

int main()
{
    int tab[] = { 5, 2, 8, 1, 9, 3, 4, 7 };
    show(tab, 8);
    sortowanie_bombelkowe(tab, 8);
    show(tab, 8);

    return 0;
}
```

Rys. 18.2. Implementacja sortowania bąbelkowego od najmniejszego do największego


```
D:\SOAW\SOAW\bin\Release\SOAW.exe
5 2 8 1 9 3 4 7
1 2 3 4 5 7 8 9

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

Rys. 18.3. Wynik działania algorytmu bąbelkowego

18.2. Algorytmy rekurencyjne

Jest to grupa algorytmów, które do rozwiązania problemu wywołują same siebie. Zaoszczędza to kod, lecz obciąża trochę bardziej pamięć komputera. Nazywa się je **rekurencyjnymi** bądź **rekursywnymi**.

Przykładowe algorytmy, które mogą zostać zaimplementowane w postaci rekursywnej:

- wyznaczanie n-tej liczby ciągu Fibonacciego;
- liczenie silni;
- liczenie n-tej potęgi;
- sortowanie quicksort.

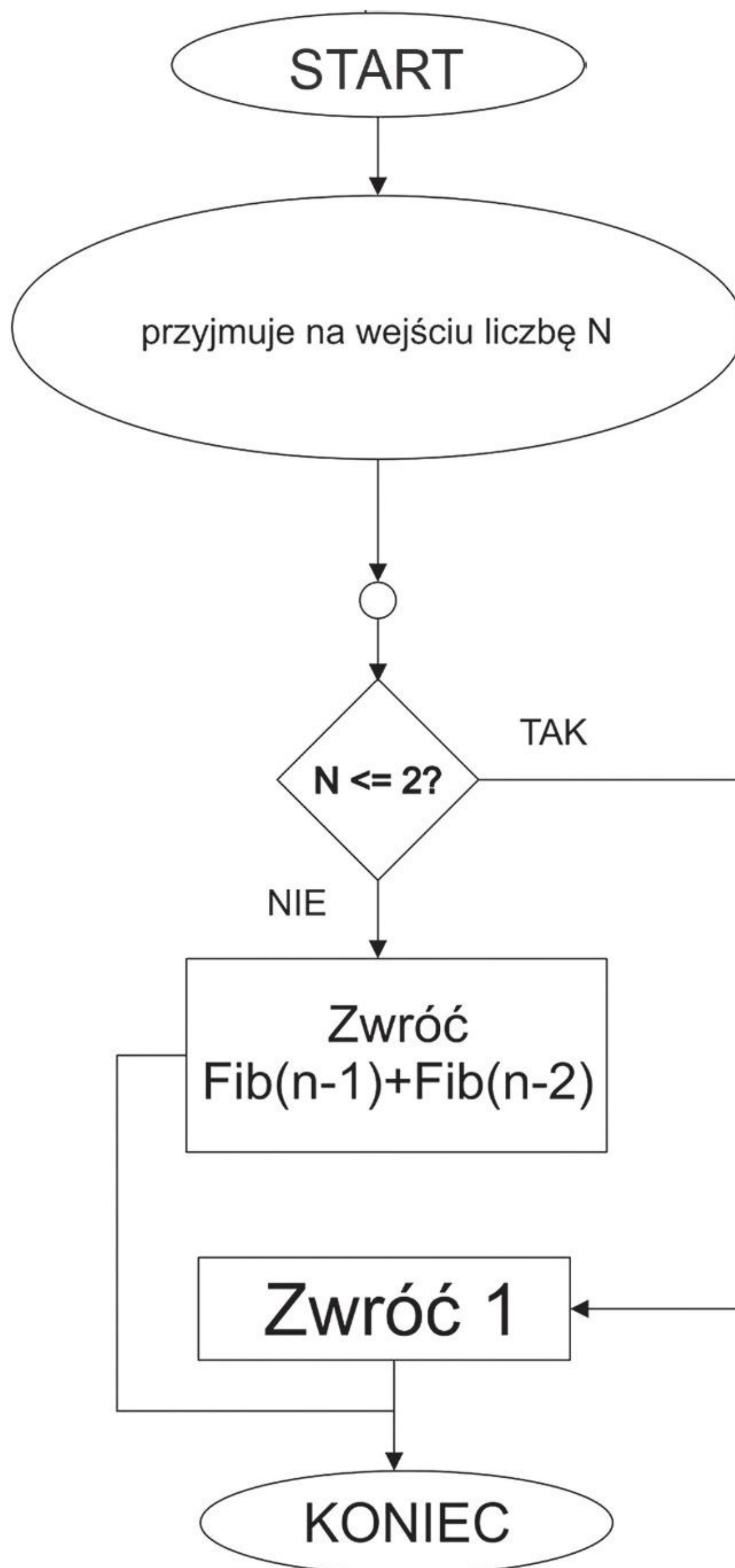
Za przykład posłuży nam algorytm wyznaczający n-ty wyraz ciągu Fibonacciego (rys. 18.5). Algorytm będzie schodził w głąb własnych wywołań, aż liczba „n” nie będzie równa jeden. Następnie będzie zwracać wartości sum $n-1 + n-2$, aż nie wróci do pierwotnego wywołania. Jak widać, trzeba przeprowadzić dość dużo operacji, by wywołać prostą czynność, dlatego łatwiej i szybciej wywołać ciąg Fibonacciego w zwykłej pętli z sumami. W praktyce lepszym zastosowaniem dla rekurencji będzie algorytm quicksort. Sortuje on tablice „po części”. **Algorytm quicksort** tnie tablicę na pół i sortuje w obrębie tych części dane lub dzieli je na kolejne połowy, aby szybciej posortować tablicę. Algorytm używa schematu „dziel i zwyciężaj”.



CIEKAWOSTKA

Warto odwiedzić stronę internetową polskiego projektu edukacyjnego http://wazniak.mimuw.edu.pl/index.php?title=Algorytmy_i_struktury_danych, na której można znaleźć wiele cennych informacji, m.in. dotyczących algorytmów:

- „złożoność obliczeniowa algorytmu (pesymistyczna, oczekiwana),
- metoda dziel i zwyciężaj,
- metoda zachłanna,
- programowanie dynamiczne,
- transformacyjna konstrukcja algorytmu,
- elementarne struktury danych: stosy, kolejki, listy,
- sortowanie przez porównania (InsertionSort, QuickSort, MergeSort),
- proste kolejki priorytetowe: kopce binarne”.



Rys. 18.4. Schemat blokowy algorytmu Fibonacciego


```
#include <iostream>
#include <string>
using namespace std;

int fibonaccici(int n) {
    if (n <= 1)
        return 1;
    return fibonaccici(n - 1) + fibonaccici(n - 2);
}

int main()
{
    cout << fibonaccici(0) << endl;
    cout << fibonaccici(1) << endl;
    cout << fibonaccici(2) << endl;
    cout << fibonaccici(3) << endl;
    cout << fibonaccici(4) << endl;
    cout << fibonaccici(5) << endl;

    return 0;
}
```

Rys. 18.5. Implementacja rekurencyjnej wersji wyznaczającej n-ty wyraz ciągu Fibonacciego

Rys. 18.6. Wynik działania programu

```

#include <iostream>
#include <string>
using namespace std;

void swaper(int*num1, int*num2) {
    int temp = *num1;
    *num1 = *num2;
    *num2 = temp;
}

int sortuj_czesc(int tab[], int lewa, int prawa) {
    int os = tab[prawa];
    int index = lewa-1;
    for (int i = lewa; i <= prawa-1; i++) {
        if (tab[i] <= os) {
            index++;
            swaper(&tab[index], &tab[i]);
        }
    }
    swaper(&tab[index + 1], &tab[prawa]);
    return index + 1;
}

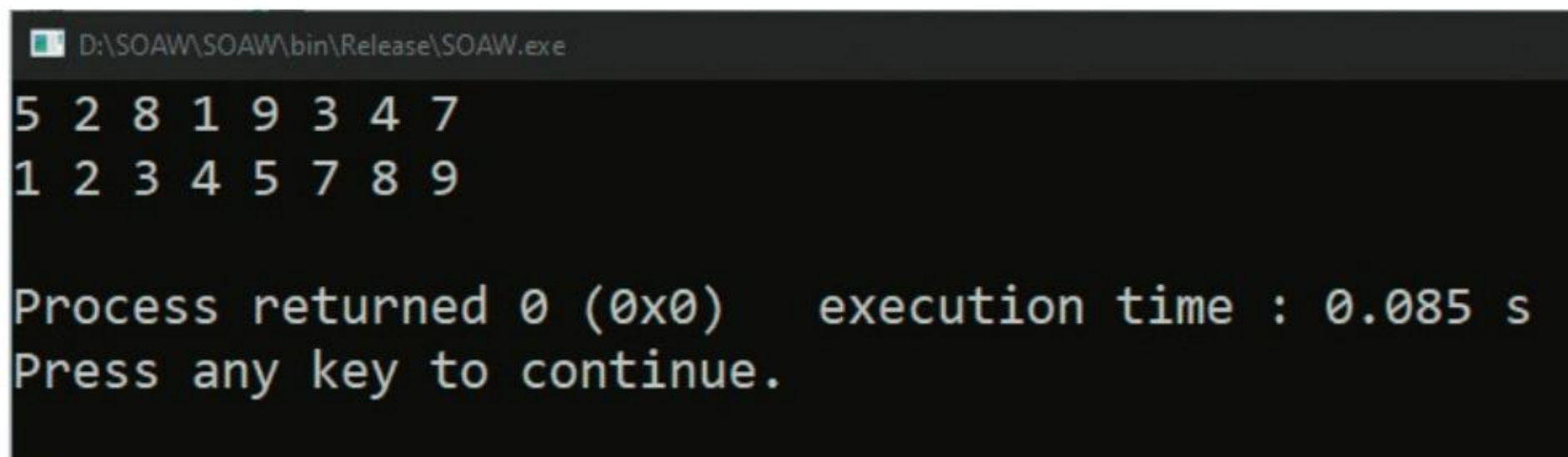
void quick_sort(int tab[], int lewa, int prawa) {
    if (lewa < prawa) {
        int srodek = sortuj_czesc(tab, lewa, prawa);
        quick_sort(tab, lewa, srodek - 1);
        quick_sort(tab, srodek + 1, prawa);
    }
}

void show(int tab[], int n) {
    for (int i = 0; i < n; i++) {
        cout << tab[i] << " ";
    }
    cout << endl;
}

int main()
{
    int tab[] = { 5, 2, 8, 1, 9, 3, 4, 7 };
    int n = sizeof(tab) / sizeof(tab[0]);
    show(tab, n);
    quick_sort(tab, 0, n-1);
    show(tab, n);
    return 0;
}

```

Rys. 18.7. Implementacja algorytmu quicksort



```
D:\SOAW\SOAW\bin\Release\SOAW.exe
5 2 8 1 9 3 4 7
1 2 3 4 5 7 8 9

Process returned 0 (0x0)   execution time : 0.085 s
Press any key to continue.
```

Rys. 18.8. Wynik działania programu

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień algorytmy sortujące tablice.
2. Podaj złożoność czasową algorytmu quicksort i opisz, dlaczego występuje i jaka jest jej złożoność pesymistyczna.
3. Opisz działanie algorytmu quicksort.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Za pomocą programowania w języku C++ (projekt konsolowy) napisz program, który z użyciem rekurencji będzie wyliczał potęgi n -tego stopnia dla wskazanej liczby.
2. Za pomocą języka programowania C++ (projekt konsolowy) zmień algorytm bąbelkowy, opisany w tym rozdziale, w taki sposób, aby zamiast tablic typu **int** sortował tablice typu **float**.
3. Za pomocą języka programowania C++ (projekt konsolowy) zmień algorytm quicksort, opisany w tym rozdziale, w taki sposób, aby zamiast tablic typu **int** sortował tablice typu **double**.

IV. Narzędzia do planowania projektu, zarządzania nim i stosowane metodologie

- Narzędzia do planowania projektu, zarządzania nim i stosowane metodologie
- Zastosowanie diagramów do zarządzania etapami projektu, zadaniami i czasem

19

Narzędzia do planowania i zarządzania projektem i stosowane metodologie

ZAGADNIENIA

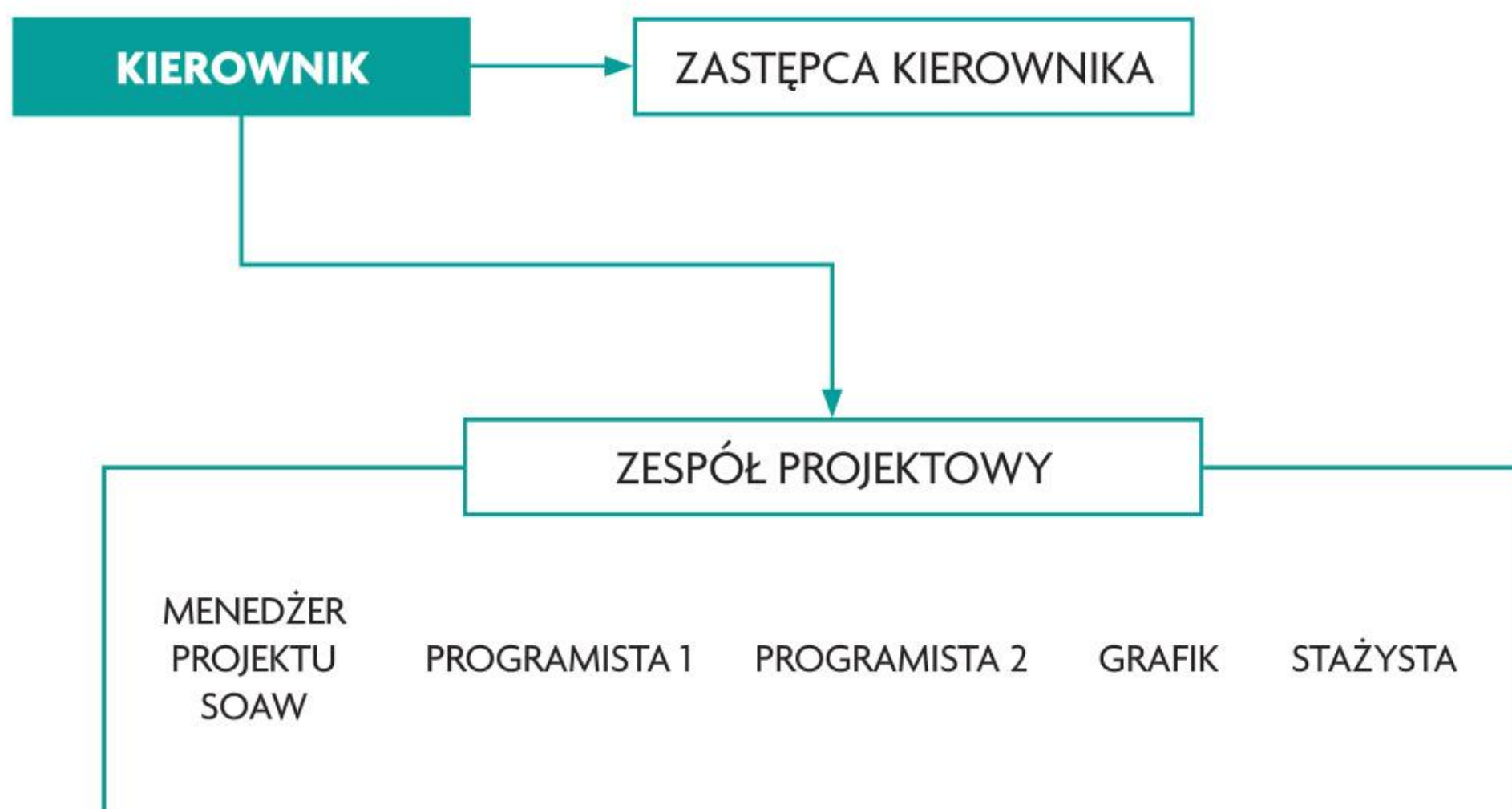
- Zarządzanie projektem – wprowadzenie
- Zasady dobierania narzędzi do planowania projektu i zarządzania nim
- Cykl życia projektu: filozofia SCRUM – Agile

19.1. Zarządzanie projektem – wprowadzenie

Zarządzanie projektem polega na planowaniu wielu działań zmierzających do osiągnięcia celów biznesowych przez realizację projektu i kierowaniu nimi. Za cały proces zarządzania powinna być odpowiedzialna jedna osoba, tak aby ewentualnymi niepowodzeniami nie obarczać zbyt dużej liczby osób. Co równie ważne, potencjalne awanse za profesjonalnie i efektywnie wykonany projekt łatwiej jest przyznać pracownikowi, który odniósł sukces, a nie całej grupie. Należy jednak pamiętać, że w sukcesie uczestniczy zwykle nie tylko szef, ale i poszczególni członkowie zespołu realizującego projekt.



Rys. 19.1. Tradycyjna struktura hierarchiczna (pionowa) zarządzania projektami



Rys. 19.2. Spłaszczona struktura zarządzania projektami

O ile w przypadku projektów tradycyjnych bardzo często stosuje się pionową ścieżkę zależności (podległości) i wynagradzania, o tyle w nowoczesnych technologiach mamy do czynienia z tzw. spłaszczeniem:

- **pionowa ścieżka zależności** – kierownik nadzoruje pracę osoby odpowiedzialnej za cały zespół – menedżera projektu (zarządza pracą zespołu), a bardzo często między nimi jest jeszcze zastępca kierownika;
- **spłaszczenie zależności** – każdy z członków zespołu projektowego jest sobie równy, przy czym menedżer projektu jest bardziej doradcą i organizatorem pracy członków zespołu niż ich szefem.

Zarządzanie projektem obejmuje:

- planowanie – działań zarówno bieżących, jak i w dłuższym terminie,
- ustalanie harmonogramu – dbałość o realizację projektu zgodnie z założonym planem,
- kontrolę osób – nadzór nad właściwym i efektywnym funkcjonowaniem zespołu projektowego,
- kontrolę zadań potrzebnych do osiągnięcia w określonym czasie i budżecie celów projektu.

CIEKAWOSTKA

Istnieje wiele teorii dotyczących efektywności zarządzania zasobami ludzkimi. Jednym z najważniejszych elementów, prowadzących do skutecznego osiągnięcia celów, jest skupienie się w zarządzaniu na ludziach, w szczególności na ich zaletach i wadach w zakresie: wiedzy, umiejętności, zdrowia, postaw społecznych – wartości, którymi się kierują, czynników ich motywujących.

19.2. Zasady dobierania narzędzi do planowania projektu i zarządzania nim

W biznesie jednym z najważniejszych elementów do odniesienia sukcesu jest zadowolony klient, który ma zazwyczaj określone potrzeby i oczekiwania. Jeżeli dostarczymy na rynek produkt, który zdoła zaspokoić potrzeby i oczekiwania klienta, wówczas będziemy mogli zarobić pieniądze i odnieść rynkowy sukces. W naszym przypadku zakładamy, że znamy już potrzeby i oczekiwania klienta, i zaczynamy zgodnie z nimi budować nasz projekt.



Rys. 19.3. Potrzeby i oczekiwania klienta a działalność producenta/dostawcy

Kryteria doboru narzędzi do planowania i zarządzania projektem:

- rodzaj klienta, w tym:
 - indywidualny czy instytucjonalny,
 - „mały” – „duży”, np. pod względem wielkości rocznych przychodów ze sprzedaży,
 - bogaty czy biedny – w dużej mierze od tego elementu mogą zależeć oczekiwania klienta i możliwości płatnicze;
- rodzaj realizowanego projektu, w tym:
 - tradycyjny czy technologiczny,
 - krótko- czy długookresowy,
 - skomplikowany pod względem realizacji czy raczej dość prosty,
 - wielkość projektu – budżet – wielkość budżetu projektu;

- zasoby firmy realizującej projekt, w tym:
 - personalne,
 - technologiczne,
 - sprzętowe,
 - organizacyjne,
 - finansowe;
- otoczenie rynkowe, w tym:
 - wnioski płynące z analizy konkurencji,
 - rozwój technologii,
 - sytuacja społeczno-ekonomiczna (szczególnie istotna w przypadku projektów długookresowych z wielkim budżetem i o wysokim stopniu skomplikowania).

19.3. Cykl życia projektu: filozofia SCRUM – Agile

Istnieje wiele różnych podejść do cyklu życia projektu. W dużej mierze jest to związane m.in. z:

- różnorodnością produktów,
- rodzajami planowanych celów projektu,
- metodologią zarządzania,
- geograficznym umiejscowieniem projektu.

W naszym przypadku mamy do czynienia z produktem technologicznym, który może się bardzo szybko „zestarzeć”, dlatego zastosujemy podejście oparte na **filozofii Agile**. Kładzie ona nacisk na działający produkt i współpracę między ludźmi, minimalizowana jest zaś sfera dokumentacyjna. Filozofia Agile wychodzi z założenia, że w szczególności w zakresie nowoczesnych technologii nie ma czasu na głębokie przemyślenia i dokumentacyjne planowanie, gdyż rynek zbyt szybko się zmienia.

Podejście Agile powinno doprowadzić do zminimalizowania procesu koncepcyjnego i (w miarę możliwości) produkcyjnego.

Fazy cyklu życia projektu – filozofia Agile, podejście SCRUM

I. Tworzenie graficzno-opisowej koncepcji

- idea projektu – identyfikacja oczekiwań właściciela pomysłu biznesowego,
- określenie „historyjkowe” pożądanego stanu przyszłego projektu (planowanej aplikacji),
- analiza możliwości realizacji projektu wraz z określeniem najbardziej optymalnego zestawu technologii.

II. Zdefiniowanie projektu

- struktura zarządzania projektem – dobranie zespołu: kierownik projektu, projektant / projektanci, deweloper / deweloperzy (programiści), grafik i ewentualnie pozostali członkowie zespołu realizującego projekt (w bardzo rozbudowanym przedsięwzięciu),
- sporządzenie wstępnego harmonogramu,
- oszacowanie kosztów,
- analiza możliwości realizacyjnych projektu (wykonalności).

III. Realizacja projektu – pierwsza iteracja

- realizacja pierwszej wersji projektu,
- spotkanie – wprowadzenie korekt, ustalenie zakresu kolejnego etapu prac,
- rozwiązywanie bieżących problemów,
- testowanie projektu.

IV. Realizacja projektu – kolejne iteracje

- realizacja prac nad projektem, zaplanowanych podczas poprzedniej iteracji,
- spotkanie – wprowadzenie korekt, ustalenie zakresu kolejnego etapu prac,
- rozwiązywanie bieżących problemów,
- testowanie projektu.

V. Etapy n-te – powtarzanie kolejnych iteracji aż do uzyskania aplikacji spełniającej oczekiwania i potrzeby klienta (właściciela pomysłu biznesowego)**VI. Etap ostatni – zakończenie projektu**

- oddanie działającej aplikacji klientowi (właścicielowi) i przekazanie odpowiedzialności za jej użytkowanie,
- uzyskanie pisemnej akceptacji potwierdzającej skuteczną realizację projektu,
- wewnętrzna ocena pracy zespołu projektowego,
- opracowanie raportu dotyczącego parametrów biznesowych zrealizowanego projektu: czasu, kosztów i jakości,
- zarchiwizowanie dokumentacji projektowej.

Zwróć uwagę na to, że gdyby był realizowany tradycyjny projekt biznesowy, a nie z obszaru nowoczesnych technologii, cykl życia byłby nieco inny. Przede wszystkim inaczej sformułowano by cel ostateczny. Tutaj mamy założenie **realizacji projektu aż do uzyskania aplikacji spełniającej oczekiwania i potrzeby klienta – właściciela pomysłu biznesowego**.

W przypadku produktu z obszaru tradycyjnego (starego) biznesu zapis mógłby brzmieć – **realizacja aż do uzyskania efektu końcowego zgodnego z początkowymi założeniami projektowymi**.

Różnica jest bardzo widoczna. W tradycyjnym podejściu z góry przewidujemy cel i efekt końcowy. W przypadku nowych technologii to, co wyjdzie na końcu, może nawet dość znacznie różnić się od początkowych założeń. Jest to sytuacja bardzo specyficzna podczas tworzenia produktów z zakresu nowych technologii (usług) internetowych. Jak już wspomniano, nie mamy czasu na zbyt głębokie przemyślenia i planowanie, gdyż mogłoby to oznaczać porażkę biznesową. Przykładowo: podobny produkt wprowadzi konkurencja, a być może okaże się, że planując i projektując, straciliśmy tyle czasu, iż zaplanowana aplikacja już nie spełnia wymaganych standardów technologicznych.



Rys. 19.4. Kolejne iteracje SCRUM – przykład

III ITERACJA SCRUM

<body>

<head>

<script src="JavaScript"> </script>

</head>

Home Logowanie O nas

SOAW portal ogłoszeniowy

Dodaj ogłoszenie ...

Wybierz właściwą kategorię ...

Ciesz się ze sprzedaży, albo z zakupu :).

Życzymy sukcesu - zespół SOAW!

SOAW, projekt, dane firmy

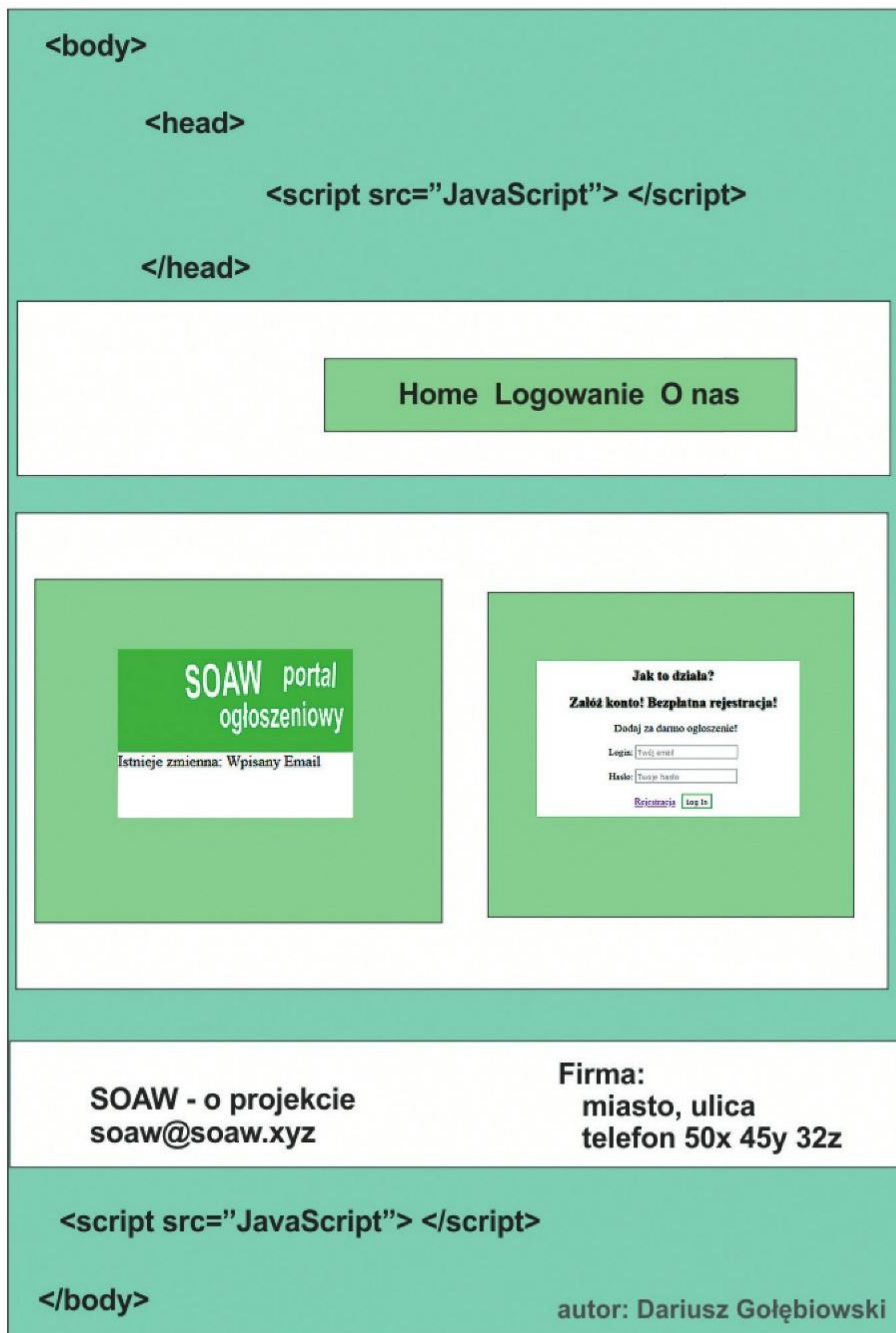
<script src="JavaScript"> </script>

</body>

autor: Dariusz Gołębiowski

Rys. 19.4. Cd. Kolejne iteracje SCRUM – przykład

IV ITERACJA SCRUM



Rys. 19.4. Cd. Kolejne iteracje SCRUM – przykład

Opisane podejście określa się jako **SCRUM**. Mamy tu na myśli SCRUM:

- iteracyjne – czyli w kolejnych okresach,
- przyrostowe – czyli są realizowane kolejne części projektu.

Rozwój produktu i/lub projektu składa się ze **sprintów** – są to opisane wcześniej iteracje, które trwają określony czas, często jeden miesiąc lub jeden tydzień, a czasem – w krótkich projektach – nawet jeden dzień. Na zakończenie każdego sprintu zespół powinien dostarczyć kolejną działającą wersję produktu. Pojęcie Scrum i metodologia z nim związana są bardzo często stosowane w trakcie tworzenia i rozwijania oprogramowania.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj pojęcie „zarządzanie projektem”.
2. Wymień przynajmniej kilka zasad dobierania narzędzi do planowania i zarządzania projektem.
3. Co oznacza sformułowanie „cykl życia produktu”?
4. Opisz krótko pojęcia SCRUM i Agile.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. CEL

Stworzenie nowej strony dla twojej szkoły metodologią Agile – SCRUM.

Żeby osiągnąć wskazany cel, wykonaj:

ZAŁOŻENIA

Podziel uczniów w klasie na pięć grup:

- A. Klient
- B. Programista backend
- C. Programista frontend
- D. Grafik
- E. Menedżer

PROJEKT

Uczniowie z grupy Klient przygotowują w sposób opisowy (np. w LibreOffice Writer) założenia nowej strony internetowej dla szkoły, m.in.:

- wygląd,
- podział treści na podstrony,
- podstawową funkcjonalność.

SCRUM – I iteracja

- Po przygotowaniu wymienionych zagadnień należy przekazać je do grupy Menedżer – wraz z instrukcją i oczekiwanym terminem przygotowania.

- Uczniowie z grupy Menedżer zbierają do współpracy pozostałe grupy uczniów, by przygotować dla nich historyjkę graficzną, na której zaprezentują, co należy przygotować. W trakcie spotkania trzeba określić podział zadań na grupy: Programista backend, Programista frontend, Grafik.
- Przez dwa tygodnie każda grupa pracuje oddzielnie nad projektem – zgodnie ze wskazówkami wynikającymi z pierwszego spotkania projektowego.

SCRUM – n-te iteracje

- Grupa Menedżer ma za zadanie tak poprowadzić projekt, czyli kolejne iteracje SCRUM, aby na koniec pojawił się gotowy produkt – nowa strona szkoły.

RADA PRAKTYCZNA

W projekt można zaangażować dyrekcję szkoły i poprosić, aby przynajmniej jeden jej przedstawiciel brał aktywny udział w pracach projektowych SCRUM – w ramach grupy Klient. Tak poprowadzony projekt zapewne zwiększy swoją użyteczność.

20

Zastosowanie diagramów do zarządzania etapami projektu, zadaniami i czasem

ZAGADNIENIA

- Wizualizacja
- Diagramy do zarządzania – przykłady

20.1. Wizualizacja

Człowiek może postrzegać otaczający go świat na kilka różnych sposobów, wykorzystując do tego swoje zmysły:

- wzrok,
- węch,
- dotyk,
- kinestetykę (odczuwanie emocji).

Każdy z nas w pewnym stopniu postrzega otoczenie za pomocą obrazów. Wielu ludziom łatwiej jest zrozumieć zagadnienia przedstawione w sposób graficzny niż opisowy czy liczbowy. Najlepszym przykładem są urządzenia przenośne, dla których elementy graficzne są podstawą funkcjonowania i oddziaływania na użytkowników. Zapewne z tego powodu lubimy różne elementy naszego życia zamieniać na obrazy. Mówimy wówczas o wizualizacji. Jest ona, jak już wspomniano, powszechna w każdej dziedzinie życia, także w zarządzaniu. W kolejnej części podręcznika zobaczysz przykład projektowej historyjki obrazkowej. Taki sposób tłumaczenia tego, co jest do zrobienia, ma wiele zalet. Podstawą jest zrozumienie danego problemu przez większą liczbę osób, niż można by to było osiągnąć za pomocą tekstu.

20.2. Diagramy do zarządzania – przykłady

Istnieje wiele narzędzi, w tym tzw. bibliotek programistycznych, które dostarczają stosowne, już gotowe rozwiązania. Wizualizacja w zarządzaniu opiera się na wielu różnych teoriach. Jedną z nich jest diagram Gantta. Jego głównym celem może być prezentacja harmonogramu zadań w projekcie. Istnieje wiele odmian diagramu Gantta, najbardziej popularne to:

- zaznaczenie realizacji projektu na linii czasu z określeniem, co i kiedy będzie wykonane;
- podział zadań na kilka kategorii, m.in. krytyczne, standardowe, podsumowujące.

Diagram Gantta możemy wykonać w programach komputerowych specjalnie do tego przeznaczonych, zwykle związanych z systemami ERP, CRM i aplikacjami do zarządzania projektami. Niestety, są one zwykle dość drogie i raczej nie nadają się do zastosowania w małych projektach. Warto rozważyć rozwiązanie alternatywne, wykonane własnoręcznie, w postaci pliku zrobionego w bezpłatnym programie LibreOffice Calc (program można pobrać pod adresem: <https://www.libreoffice.org/>). Przedstawiony przykład to jedna z najprostszych form planowania i rozliczania realizacji projektu w postaci graficznej.

Na rys. 20.1 i 20.2 przedstawiono:

1. diagram planu projektu – ujęcie w roboczogodzinach,
2. diagram powstały w trakcie realizacji – ujęcie w roboczogodzinach.

Jak zapewne zauważyłeś, plan jest nieco krótszy niż realizacja. Tak często bywa. Co najważniejsze, założono trzy godziny zapasowe i całość zmieściła się w tym czasie rezerwowym. Warto wiedzieć, że bardzo często zamiast dwóch oddzielnych obszarów (diagramów) realizacja jest wykonywana innym kolorem na planie, wówczas bardziej dobitnie widać ewentualne przesunięcia (opóźnienia) w realizacji projektu. Często zadania są dzielone na krytyczne i pozostałe, a każdy rodzaj oznaczany innym kolorem. Dzięki temu zarządzający mogą skupić się na tym, co najważniejsze. Taki podział ma zastosowanie głównie w bardzo dużych projektach.

TERMINARZ PROJEKTU SOAW – PLAN

	godz.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1. Projekt graficzny																		
2. Struktura HTML																		
3. Wygląd projektu CSS																		
4. Stworzenie bazy danych																		
5. Kodowanie strony głównej PHP																		
6. Kodowanie logowania PHP																		
7. Testowanie logowania																		
8. Dopracowanie grafiki																		
9. Wprowadzenie niezbędnych poprawek w kodzie																		

LEGENDA

czynność niezbędna



czynność zapasowa



Rys. 20.1. Wizualizacja terminarza planu realizacji projektu (prosty diagram Gantta)

TERMINARZ PROJEKTU SOAW – REALIZACJA

	godz.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1. Projekt graficzny																		
2. Struktura HTML																		
3. Wygląd projektu CSS																		
4. Stworzenie bazy danych																		
5. Kodowanie strony głównej PHP																		
6. Kodowanie logowania PHP																		
7. Testowanie logowania																		
8. Dopracowanie grafiki																		
9. Wprowadzenie niezbędnych poprawek w kodzie																		

LEGENDA

czynność niezbędna



czynność zapasowa



Rys. 20.2. Wizualizacja terminarza faktycznej realizacji projektu (prosty diagram Gantta)

 SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie znaczenie dla człowieka ma wizualizacja materiałów edukacyjnych?
2. Opisz, na czym polega zarządzanie planowaniem projektu z wykorzystaniem narzędzi graficznych (np. diagramu Gantta).

 SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Jako ucznia technikum czeka cię zdanie egzaminu państwowego z kwalifikacji INF.04. Projektowanie, programowanie i testowanie aplikacji. Aby zdać egzamin:
 - dowiedz się od nauczyciela:
 - kiedy będziesz zdawał egzamin;
 - jaki jest zakres tematyczny, z którego musisz się przygotować;
 - wspólnie z innymi uczniami z klasy i opiekunem szkolnym dbającym o przygotowanie do egzaminu opracujcie diagram Gantta, który wyznaczy plan takiego zorganizowania nauki, abyście byli gotowi (zapoznali się z całym materiałem edukacyjnym) do zdania egzaminu z kwalifikacji.

21

Programy do wspierania zarządzania projektami i systemy kontroli wersji

ZAGADNIENIA

- Funkcje programów do wspierania zarządzania projektami
- Przykładowe oprogramowanie wspierające zarządzanie projektami
- Systemy kontroli wersji na podstawie GIT

21.1. Funkcje programów do wspierania zarządzania projektami

Na rynku jest mnóstwo płatnych i bezpłatnych programów do wspierania zarządzania projektami. Bez problemu znajdziesz każdy z wymienionych rodzajów (tzn. programy są co najwyżej prawie bezpłatne, bo gdy coś jest zupełnie bezpłatne, to w internecie często znaczy, że płacisz własnymi danymi lub danymi klientów).



Rys. 21.1. Wizualizacja tablicy do realizacji projektu typu kanban – widok początkowy

Program do wspierania zarządzania projektami powinien przede wszystkim umożliwiać użytkownikom:

- organizowanie projektów, treści i pomysłów w uporządkowany schemat (system),
- wspomaganie rozwijania własnej logiki procesu zarządzania,
- monitorowanie i zgłaszanie projektów w czasie rzeczywistym,
- planowanie czasu pracy zespołu na bieżące i przyszłe projekty,
- współpracowanie z innymi członkami zespołu w czasie rzeczywistym,

- organizowanie projektów na wykresie tabeli, kalendarza, tablicy kanban czy diagramu Gantta,
- wykonanie podsumowania zrealizowanego projektu,
- filtrowanie, sortowanie i zmienianie kolejności zapisanych rekordów projektowych,
- tworzenie własnych, niestandardowych widoków i pól,
- łatwą integrację poczty e-mail, aplikacji i mediów społecznościowych,
- obsługę zarządzania formularzami,
- zarządzanie przepływem informacji między członkami zespołu.



Rys. 21.2. Wizualizacja tablicy do realizacji projektu typu kanban – widok rozpoczęcia projektu

21.2. Przykładowe oprogramowanie wspierające zarządzanie projektami

Oprogramowanie do zarządzania projektami może być dostępne jako:

- aplikacja webowa już gotowa do użytkowania,
- aplikacja do zainstalowania na własnych serwerze i/lub komputerze.

Oba rodzaje aplikacji mają zalety i wady.

Aplikacja webowa już gotowa do użytkowania

Dużą zaletą w tego typu rozwiązaniach jest to, że mamy gotowy do użytkowania program, który inna firma dla nas utrzymuje na własnych serwerach, dba o jego rozwój i **main-tenance**. Główna wada jest taka, że musimy ograniczyć nasze oczekiwania funkcjonalne co do programu do takich, jakie w ramach danego rozwiązania są dostępne. Tego typu rozwiązania są niby bezpłatne, ale w życiu niewiele można otrzymać za darmo. W tym przypadku trzeba albo dopłacać do bardziej zaawansowanych funkcji programu, albo płacić za program dostępem innych do danych, które przechowujemy.



Rys. 21.3. Wizualizacja tablicy do realizacji projektu typu kanban – widok w trakcie realizacji projektu

Poniżej znajdziesz linki do stron internetowych wybranych projektów webowych, których celem jest wspieranie zarządzania projektami:

- <https://clickup.com/>,
- <https://hive.com/>,
- <https://airtable.com/>,
- <https://blossom.co/>,
- <https://www.assembla.com/>,
- <https://www.pivotaltracker.com/>,
- <https://www.targetprocess.com/>,
- <https://www.atlassian.com/software/jira>,
- <https://trello.com/pl/>,
- <https://www.inflectra.com/>.

Aplikacja do zainstalowania na własnym serwerze i/lub komputerze

W tego typu rozwiązaniach trzeba samemu zainstalować, utrzymywać i rozwijać instancję aplikacji (rozwijane są także w ramach *open source*). Dzięki instalowaniu, utrzymywaniu i rozwijaniu tego typu oprogramowania programiści podnoszą swoje kwalifikacje. Poza tym nie przekazują nigdzie swoich danych ani danych klientów, dla których pracują.

To, co może być problemem dla zwykłych użytkowników, dla programistów powinno być standardowym podejściem do własnego rozwoju. Z punktu widzenia programisty korzystanie z tego typu rozwiązań ma jeszcze inne zalety. Największą jest to, że po zainstalowaniu we własnym środowisku można oprogramowanie niemal dowolnie dopasować do własnych potrzeb.

Poniżej znajdziesz linki do stron internetowych wybranych bezpłatnych aplikacji wspierających zarządzanie projektami (rozwijane w ramach *open source*):

- <https://www.redmine.org/>,
- <https://www.bugzilla.org/>,
- <https://trac.edgewall.org/>,
- <https://www.mantisbt.org/>,
- <https://www.dolibarr.org/>.

Szczególnie potężny pod względem funkcjonalności jest **Dolibarr** – projekt łączący system ERP z CRM. Jest bardzo wszechstronny i – co najważniejsze – działa na środowisku, w którym będziemy już wkrótce wspólnie tworzyć podstawy aplikacji SOAW, czyli PHP plus serwer webowy plus relacyjna baza danych. Ma także mnóstwo narzędzi do zarządzania projektami, m.in. organizację i zarządzanie projektami, wspieranie różnego rodzaju wykresami i diagramami (w tym Gantt). Jego dodatkową zaletą jest świetna i ogólnie dostępna dokumentacja, rozwijana przez społeczność zgromadzoną wokół tej aplikacji (<https://wiki.dolibarr.org/index.php?title=Home>) i różnego rodzaju samouczki.

Wymieniono tylko kilka przykładowych i chyba najbardziej znanych projektów wśród programistów. Aby sprawdzić bogactwo oprogramowania dostępnego na zasadach *open source*, spójrz na poniższe linki, w których znajdziesz bezpłatne zasoby tego typu oprogramowania:

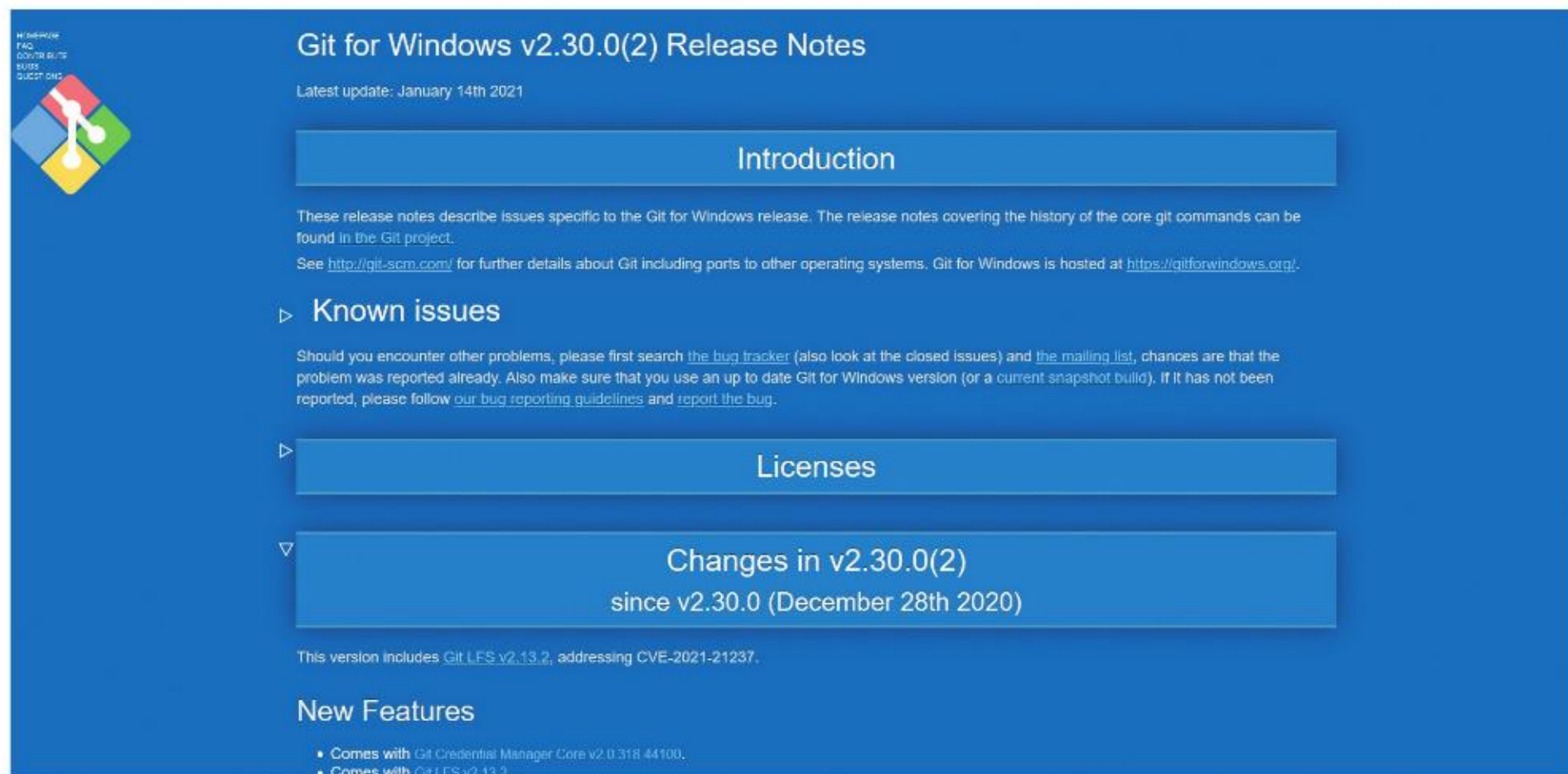
- projekty na GITHUB – <https://github.com/search?q=project+management> – ponad 52 tys. repozytoriów zawierających zagadnienia z zakresu Project management;
- <https://sourceforge.net/directory/business-enterprise/office/os:windows/?q=project+management> – ponad 1,5 tys. programów wspierających zarządzanie, w tym także zarządzanie projektami.

21.3. Systemy kontroli wersji na podstawie Git

Systemy kontroli wersji to programy, które wspomagają pracę programistów i zespołów deweloperskich (grup programistów). Umożliwiają one:

- śledzenie zmian w kodzie źródłowym (programistycznym),
- odtwarzanie poprzednich wersji plików,
- łączenie zmian w plikach wykonanych przez wiele osób w różnym czasie.

Na rynku jest dostępnych kilka ciekawych projektów dostarczających tego typu rozwiązania: od programów zainstalowanych i działających na komputerach lokalnych aż po potężne systemy zainstalowane w sieci. Przykładem tego drugiego rozwiązania jest Git, który obecnie jest prawdopodobnie najpopularniejszym rozwiązaniem wśród większości programistów i zespołów roboczych.



Rys. 21.4. Aplikacja Git for Windows – widok licencji

Git to tzw. rozproszony system kontroli wersji, który został opracowany przez twórcę systemu operacyjnego Linux – programistę Linusa Torvaldsa. Początkowo Git miał jedynie wspierać rozwój jądra systemu Linux. Obecnie jego zastosowanie jest zdecydowanie większe. Jego wielką zaletą jest to, że należy do grupy wolnego oprogramowania – jest opublikowany na licencji GNU GPL w wersji 2.

Git możesz pobrać z oficjalnej strony projektu: <http://git-scm.com/>. Pod tym adresem znajdziesz także wiele materiałów edukacyjnych, dzięki którym bez trudności poznasz ten fantastyczny system.

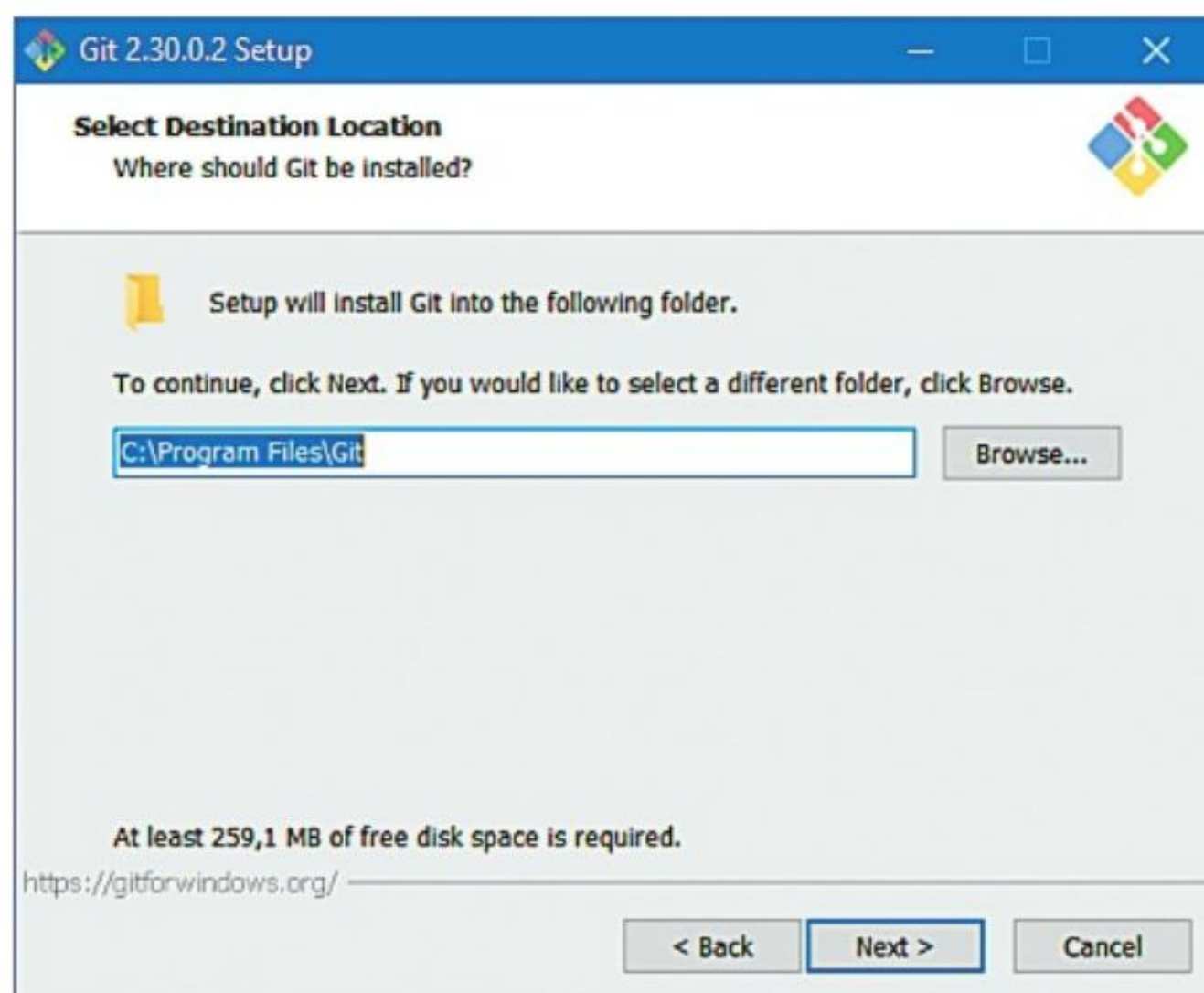
Git jako system kontroli wersji potrafi śledzić wszystkie zmiany na plikach (lub pliku). Co najważniejsze – umożliwia przywrócenie dowolnej wcześniejszej wersji. Teoretycznie rozwiązaniem, które przyniosłoby podobne rezultaty, jest każdorazowe dbanie podczas dokonywania zmian w pliku o to, żeby dany plik skopiować i najlepiej przenieść do innego folderu, by przez przypadek go nie zmodyfikować. Podczas sporadycznej pracy „na plikach” takie rozwiązanie także ma sens. Natomiast jeżeli pracuje się po kilka bądź kilkanaście godzin dziennie przy plikach (programiści, graficy, projektanci itp.), to ręczne ich zapisywanie jest co najmniej uciążliwe. Git może rozwiązać problemy w tym zakresie. Po to został stworzony.

Projekt Git ma wbudowane mechanizmy spójności danych. Pliki mogą znajdować się w trzech stanach:

- zatwierdzony – dane zostały bezpiecznie zachowane w twojej lokalnej bazie danych;
- zmodyfikowany – zmiany nie zostały wprowadzone do bazy danych, ale plik został zmieniony;
- śledzony – w następnej operacji commit zmodyfikowany plik będzie zatwierdzany.

Instalacja Git

Ze strony projektu <https://git-scm.com/> trzeba pobrać plik instalacyjny na swój komputer. Musi on być zgodny z wersją twojego systemu operacyjnego.



Rys. 21.5. Instalacja aplikacji Git w Windows

Po pobraniu trzeba rozpocząć instalację z pomocą prawego klawisza myszki – uruchom jako administrator.



Rys. 21.6. Instalacja aplikacji Git w Windows – rodzaj licencji GNU Public License

Po instalacji Git należy ustawić nazwę użytkownika i adres e-mail. Każdy utworzony commit używa tych informacji i jest z nimi związany, więc trzeba je jednoznacznie określić:

```
$ git config --global user.name „Jasiek Kowalski”  
$ git config --global user.email jaskiekkowal@pols.pl
```

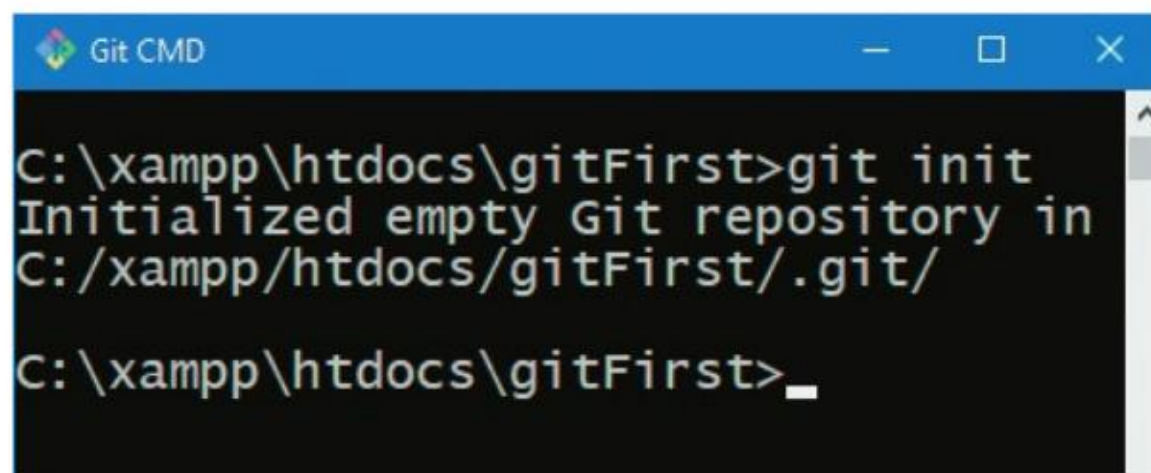
Aby sprawdzić bieżące ustawienia, trzeba napisać polecenie:

```
$ git config --list
```

Inicjalizacja Gita w istniejącym katalogu dla pustego projektu

Rozpoczęcie śledzenia zmian w plikach istniejącego projektu będzie możliwe po wykonaniu polecenia:

```
$ git init
```

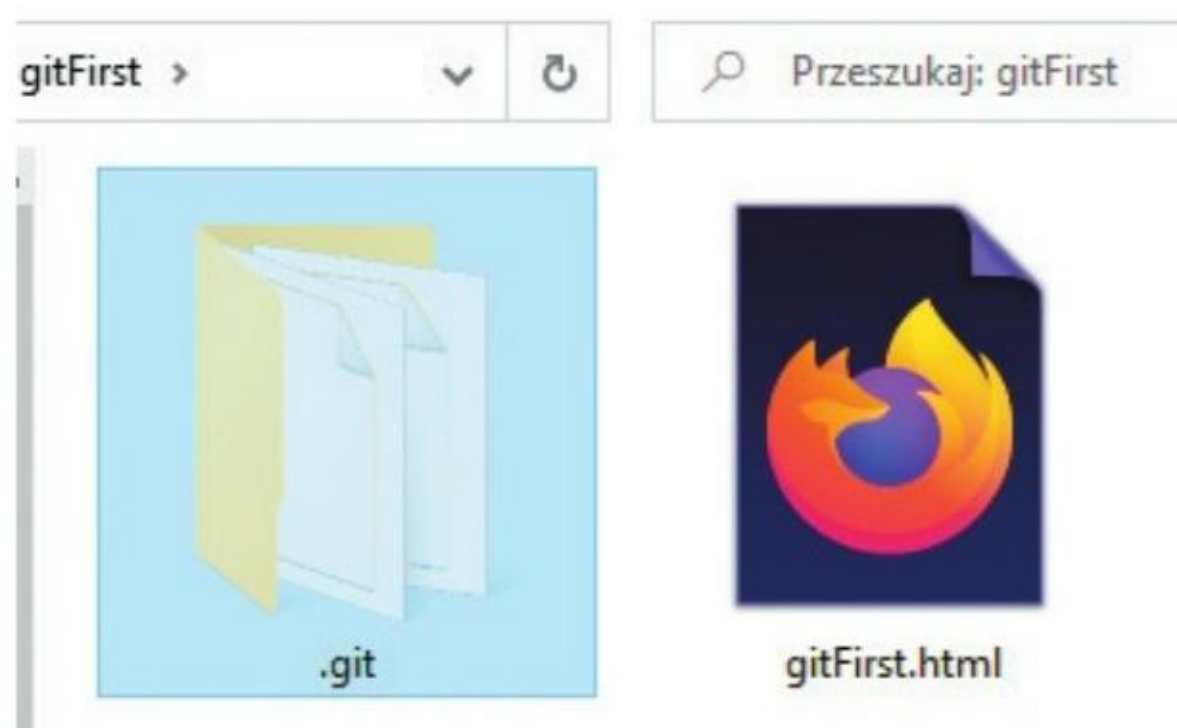


Rys. 21.7. Inicjalizacja pierwszego repozytorium Git

! UWAGA

Najpierw trzeba przejść do katalogu projektu, który ma być śledzony.

Powyższe polecenie stworzy nowy podkatalog o nazwie .git. Co ważne, po powyższym poleceniu jeszcze nie będziesz śledzony.



Rys. 21.8. Utworzenie katalogu Git w projekcie

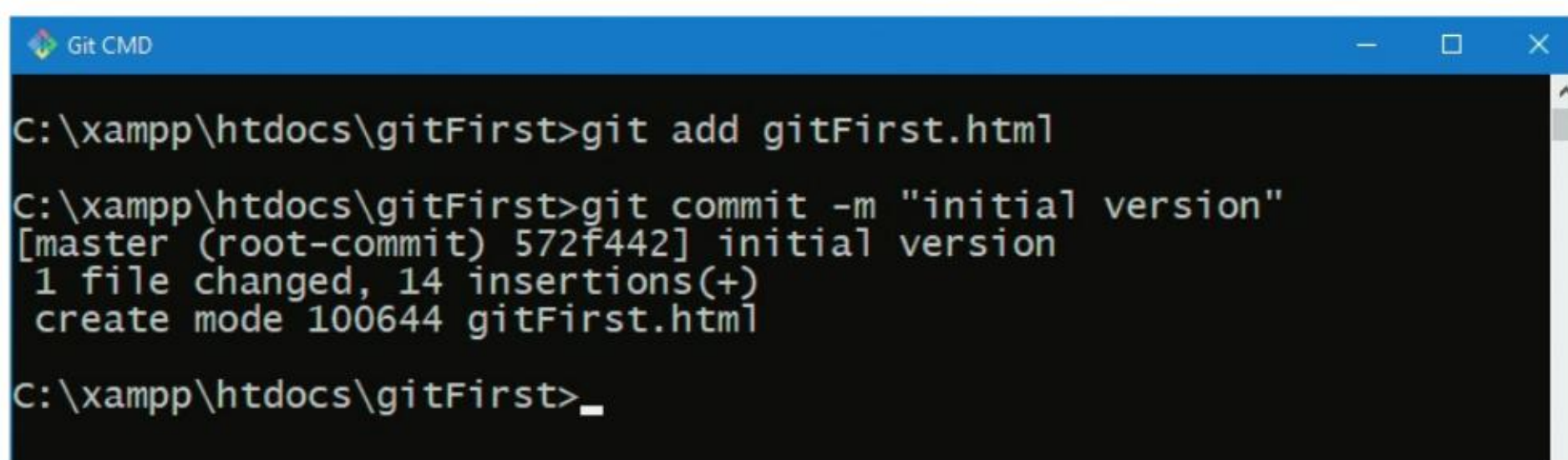
Inicjalizacja Gita dla istniejących już plików

Wystartowanie z kontrolą wersji istniejących plików rozpoczynamy od utworzenia tzw. rewizji początkowej. Dokonujemy tego za pomocą kilku poleceń `add` (dodaj), do których wybieramy pojedyncze pliki do śledzenia. Zatwierdzanie wykonujemy poleceniem `commit`:

```
$ git init
$ git add nazwaPliku
$ git commit -m „initial project version”
```

Na potrzeby tej części podręcznika założono katalog `gitFirst`, w którym utworzono prosty plik `gitFirst.html`. Na nim i jego zmianach pokażemy kilka komend Git.

```
$ git init
$ git add gitFirst.html
$ git commit -m „initial project version”
```

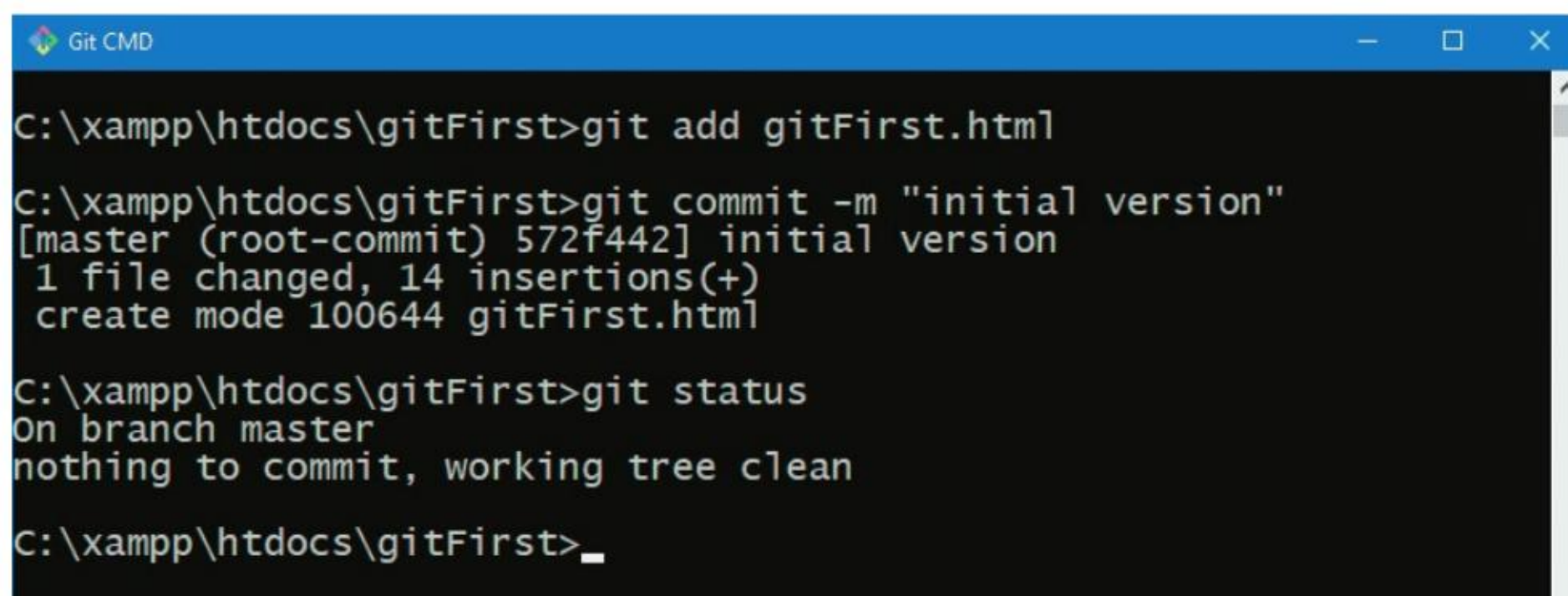
A screenshot of a Windows command prompt window titled "Git CMD". The window shows the following commands and output:

```
C:\xampp\htdocs\gitFirst>git add gitFirst.html
C:\xampp\htdocs\gitFirst>git commit -m "initial version"
[master (root-commit) 572f442] initial version
1 file changed, 14 insertions(+)
create mode 100644 gitFirst.html
C:\xampp\htdocs\gitFirst>_
```

Rys. 21.9. Pierwszy commit Git w projekcie – początkowy

Teraz masz już repozytorium Gita i ostatnią wersję lub kopię roboczą wybranego projektu. Możesz sprawdzić stan za pomocą polecenia:

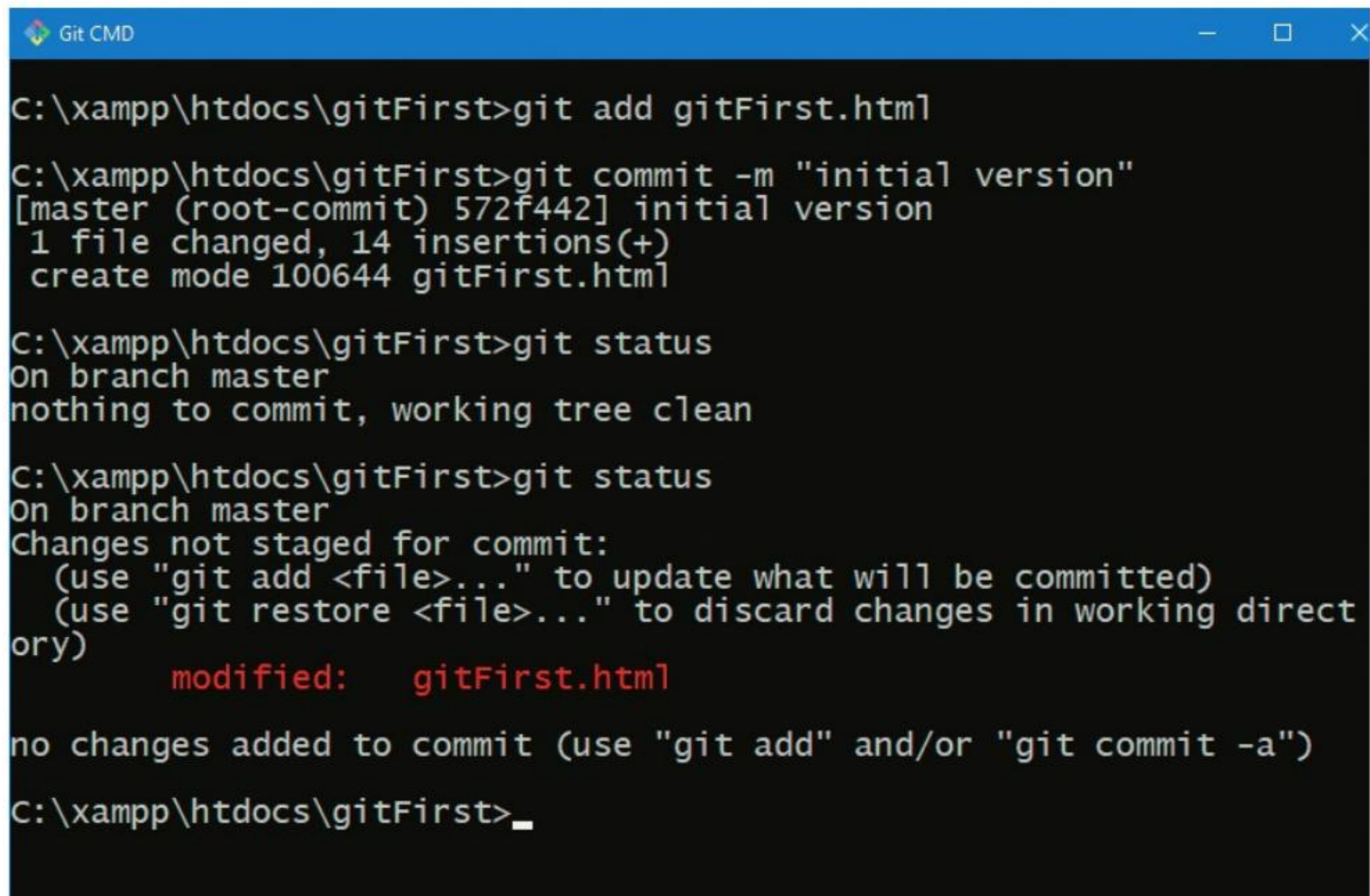
```
$ git status
```

A screenshot of a Windows command prompt window titled "Git CMD". The window shows the following commands and output:

```
C:\xampp\htdocs\gitFirst>git add gitFirst.html
C:\xampp\htdocs\gitFirst>git commit -m "initial version"
[master (root-commit) 572f442] initial version
1 file changed, 14 insertions(+)
create mode 100644 gitFirst.html
C:\xampp\htdocs\gitFirst>git status
On branch master
nothing to commit, working tree clean
C:\xampp\htdocs\gitFirst>_
```

Rys. 21.10. Sprawdzenie statusu projektu w Git

Wprowadzimy teraz małe zmiany w pliku `gitFirst.html`, a potem znowu uruchomimy `$ git status`.



```
Git CMD
C:\xampp\htdocs\gitFirst>git add gitFirst.html
C:\xampp\htdocs\gitFirst>git commit -m "initial version"
[master (root-commit) 572f442] initial version
1 file changed, 14 insertions(+)
create mode 100644 gitFirst.html
C:\xampp\htdocs\gitFirst>git status
On branch master
nothing to commit, working tree clean
C:\xampp\htdocs\gitFirst>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   gitFirst.html
no changes added to commit (use "git add" and/or "git commit -a")
C:\xampp\htdocs\gitFirst>_
```

Rys. 21.11. Sprawdzenie statusu w Git – po zmianie w pliku projektowym

Jak widzisz, Git śledzi plik i zauważył jego zmianę. Równocześnie zasugerował:

- `$ git add` – dodanie czegoś

lub

- `$ git commit -a` – zatwierdzenie.

Jeżeli pojawią się problemy związane z domyślnym edytorem twojego systemu, spróbuj dokonać stosownych ustawień. Przykładowy kod:

```
$ git config --global core.editor „code” --wait
```

Zmiany z komentarzem możesz zatwierdzić za pomocą polecenia:

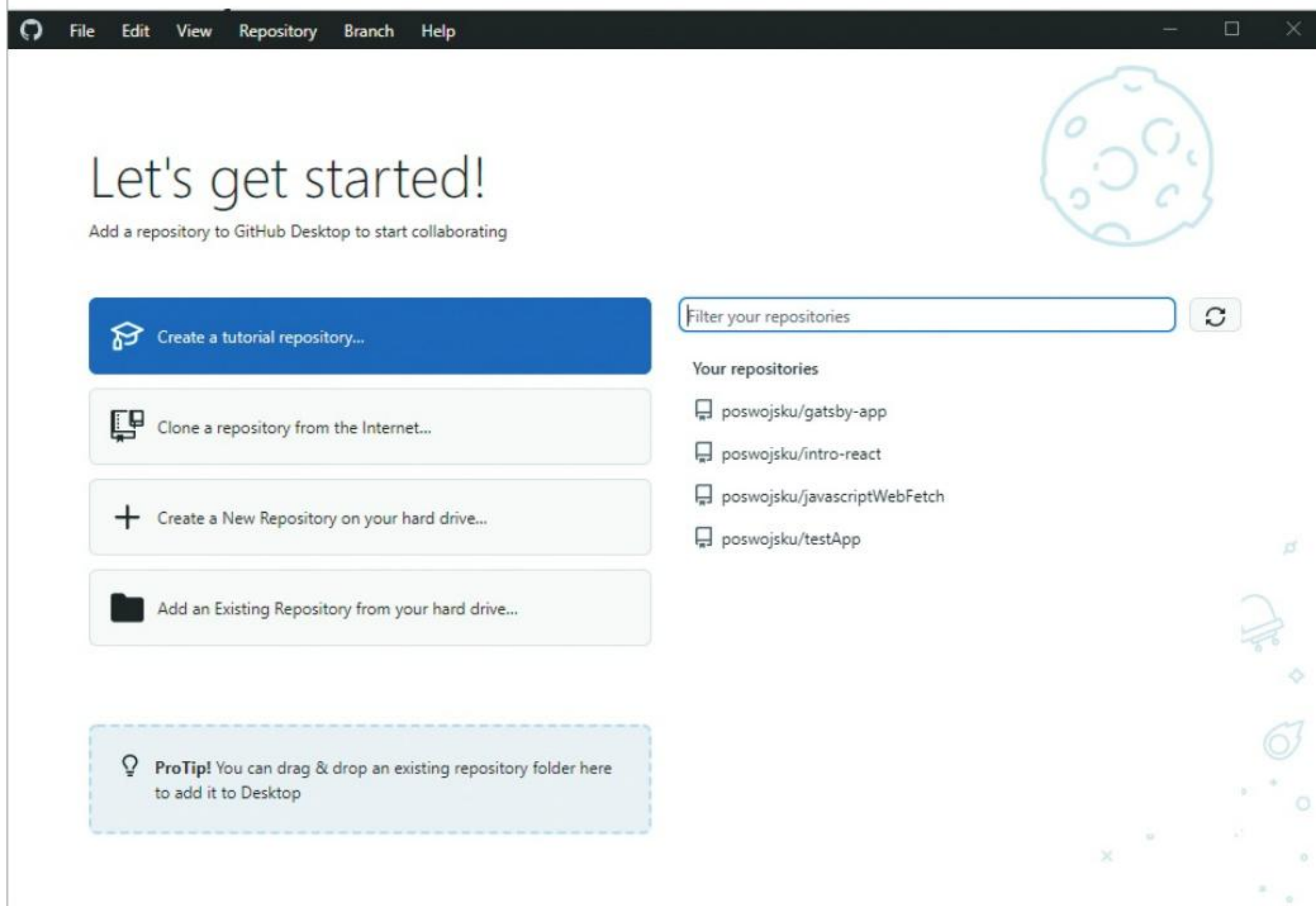
```
git commit -am „message”
```

Jeżeli używasz konsoli GIT/bash, przydatnym poleceniem może być `Cls` – czyści konsolę (zawartość) Gita.

Git – wersja desktop

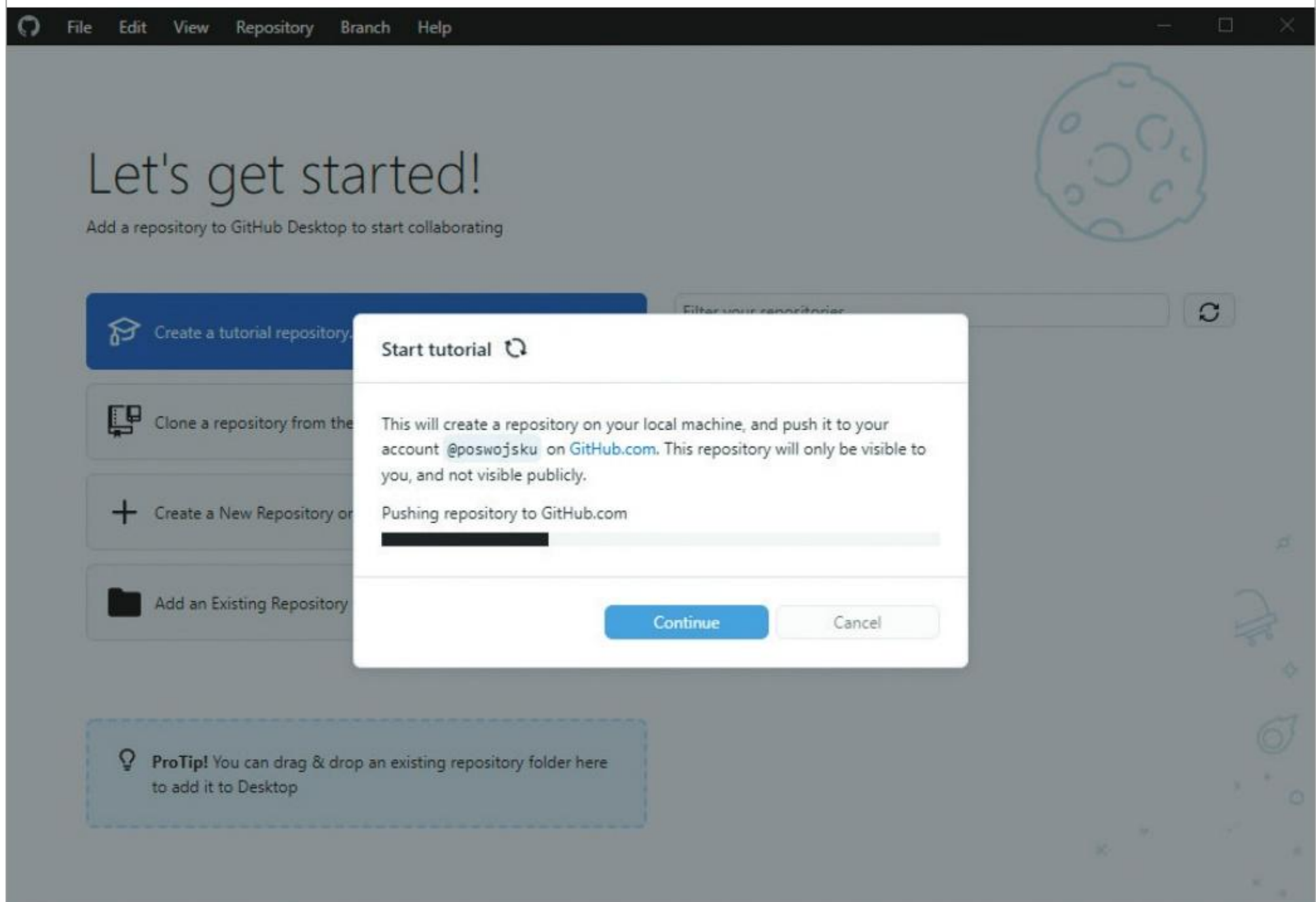
Dla osób, które nie mają dużego doświadczenia w pracy z konsolą i ustawieniami systemu operacyjnego Gita, basha itp., istnieje alternatywny sposób obsługi Gita – okienkowy.

Bezpłatną wersję desktopową do pobrania znajdziesz pod adresem <https://desktop.github.com/>. W trakcie jej instalowania możesz od razu założyć konto użytkownika GitHub – największego portalu przeznaczonego dla programistów. Przy okazji znajdziesz tam miliony różnego rodzaju rozwiązań w tysiącach technologii.



Rys. 21.12. Ekran startowy aplikacji GitHub Desktop

Po instalacji uzyskasz możliwość zainstalowania samouczka do GitHub Desktop, dzięki któremu poznasz podstawy jego obsługi. Warto skorzystać z tej zupełnie bezpłatnej opcji.



Rys. 21.13. Ekran aplikacji GitHub Desktop – tworzenie tutoriala

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym zajmują się systemy kontroli wersji w informatyce?
2. Podaj przynajmniej 10 funkcji programów do wspierania zarządzania projektami.
3. Wskaż kilka rozwiązań IT wspierających zarządzanie projektami.
4. Czy Gita trzeba zainstalować na własnym komputerze, żeby śledził zmiany w plikach? Odpowiedź uzasadnij.
5. Podaj dowolne polecenie Gita i opisz, do czego ono służy.
6. Czym się różni Git od GitHub Desktop?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. W swoim komputerze:
 - zainstaluj program Git;
 - stwórz katalog gitTest;
 - utwórz w wymienionym katalogu dwa pliki:
 - index.html – zaprogramuj w nim prostą stronę internetową, np. wyświetlającą przywitanie „Witaj na mojej stronie”; podlinkuj do niej plik style.css,
 - style.css – przygotuj estetyczny wygląd dla stworzonej strony index.html;
 - za pomocą poleceń git zgłoś do repozytorium przygotowany projekt, czyli pliki index.html i style.css;
 - sprawdź, czy twój projekt znajduje się w repozytorium.

V. Projektowanie aplikacji

- Analiza wymagań klienta
- Projektowanie aplikacji klient–serwer
- Aplikacja SOAW: interfejs, wygląd, funkcjonalność

22

SOAW: analiza wymagań klienta

ZAGADNIENIA

- Profesjonalna obsługa klienta
- Analiza wymagań klienta
- Przykładowe narzędzia analityczne
- Specyfikacja projektu w programowaniu
- Paradygmaty programowania

22.1. Profesjonalna obsługa klienta

Rozmawiając o tworzeniu aplikacji, konieczne trzeba poruszyć temat klienta zamawiającego aplikację (gdy tworzona jest ona na zamówienie). Aby odnieść sukces w kontaktach biznesowych, trzeba skupić się na profesjonalnej obsłudze klienta, czyli na jego potrzebach, oczekiwaniach i korzyściach, jakie powinien wynieść ze współpracy z naszą firmą. Podkreślenie wymienionych wartości w kontaktach z klientem ma na celu uświadomienie sobie, że sprzedawanie to nie to samo, co oferowanie klientowi różnych, niekoniecznie potrzebnych mu przedmiotów.



Rys. 22.1. Klienci – najważniejszy element przychodowy każdej firmy

Klient powinien być tak obsłużony, aby w przyszłości sam zgłosił się do danego dostawcy (handlowca, menedżera firmy), i to z przyjemnością. Właściwe podejście do klienta sprawia, że firma ponosi dużo mniejsze koszty podczas sprzedawania kolejnych produktów / usług temu samemu klientowi niż w trakcie pozyskiwania nowych partnerów biznesowych.



Rys. 22.2. Zależność sprzedaży od zadowolenia klienta

LOGO

Hi,

admin

Home

Users

Customers

Schedules

Tasks

Management

UPDATE TASK

User Email

admin@pl

Task Name

Buy CRMdg :)

Description

Why should you buy Premium CRMdg? It's simple: huge development opportunities

Rys. 22.3. Panel aplikacji CRM obsługujący klienta

Warto wspomnieć, że tylko w jednoosobowych firmach (czasem kilkuosobowych) programista piszący kody aplikacji ma bezpośredni kontakt z klientem. W zdecydowanej większości przypadków istnieją oddzielne stanowiska w firmie, które są łącznikiem między klientem zamawiającym określoną aplikację a programistą. Zwykle jest to menedżer (nazewnictwo bywa różne, zależy zwykle od standardów stosowanych w danej firmie), np.: menedżer projektu, menedżer produktu, menedżer klienta.

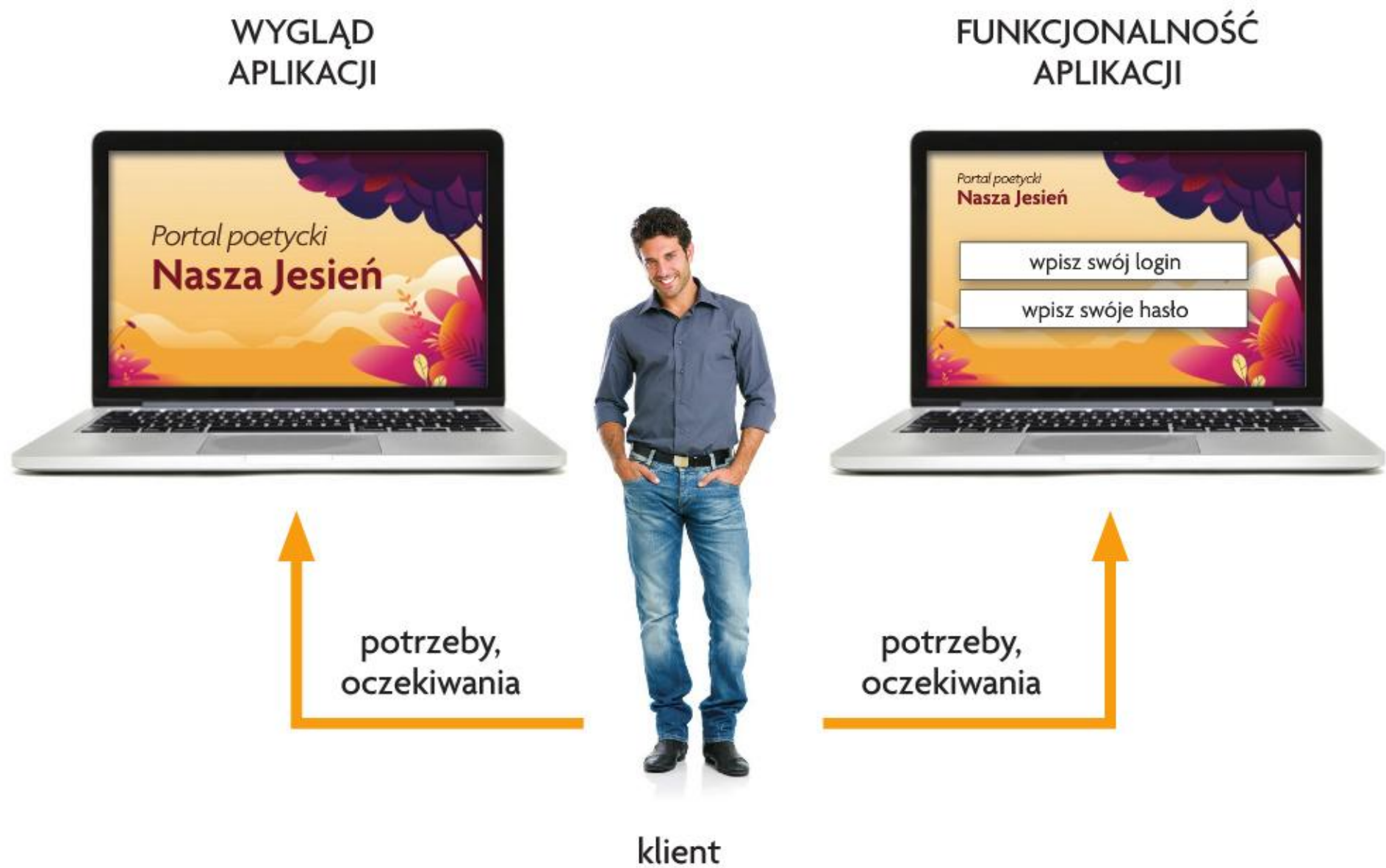


Rys. 22.4. Hierarchia ważności w obsłudze klienta

Zadaniem menedżera jest uzyskanie kompletu informacji o oczekiwaniach klienta co do zamawianego produktu (usługi) IT. Taka osoba powinna mieć co najmniej solidne podstawy z zakresu technologii, w których będzie realizowany projekt. Wówczas może się zająć samodzielnie przygotowaniem specyfikacji technicznej. Gdy menedżer nie ma stosownej wiedzy, musi wesprzeć się doświadczeniem jednego z programistów. Zwykle współpracuje on z szefem pracowników programujących kody bądź z szefem określonej grupy programistów (tej przydzielonej wstępnie do projektu). Takie zarządzanie projektem jest bardzo rozpowszechnione w codziennej praktyce tworzenia aplikacji. W kolejnych częściach podręcznika jeszcze wrócimy do tego tematu, choć zajmiemy się głównie filozofią Agile.

22.2. Analiza wymagań klienta

Ogólna wizja projektu to minimum, które musi przedstawić klient. Oczywiście najlepiej by było, gdyby przyszedł do programisty z już gotową specyfikacją techniczną, ale to się zdarza dość rzadko. Zwykle jedynie informuje o projekcie w postaci opisowej. Zadaniem odbiorcy danych jest ich opracowanie i przełożenie na określoną funkcjonalność i właściwy wygląd.



Rys. 22.5. Projekt aplikacji a klient

Funkcjonalność

Jest to opis zadań, które będzie realizowała działająca aplikacja. Trzeba zebrać szczegółowe informacje co do zadań, które powinien realizować gotowy program komputerowy, np.:

- umożliwienie autorejestracji użytkownikom strony WWW;
- możliwość zalogowania się zarejestrowanych użytkowników;
- działający system głosowania, który po naciśnięciu określonego przycisku spowoduje zliczanie głosów za i przeciw.

Wygląd

W żargonie grafików wygląd aplikacji jest zwykle nazywany „designem projektu”. Zwykle, tworząc określoną aplikację, mamy narzucone elementy graficzne, stanowiące system identyfikacji wizualnej firmy i/lub danego projektu. W jego skład mogą wchodzić:

- logo firmy i/lub projektu;
- kolory firmy i/lub projektu;
- hasła reklamowe i promocyjne identyfikujące firmę i/lub projekt.

22.3. Przykładowe narzędzia analityczne

Istnieje wiele różnego rodzaju narzędzi, które mogą umożliwić pobieranie informacji od klienta i zaplanowanie projektu. Można je podzielić na:

- gotowe narzędzia przeznaczone do tych celów;
- własne narzędzia przygotowane na bazie bezpłatnego oprogramowania.

LOGO admin ▾ Home Users Customers Schedules Tasks Management

TASKS + Add New

Search Recipient's text Search

User Email	Task Name	Task Description	first name, surname, position, email	more edit delete
dfghdgh	dghdgh	dghdghdgh	dghdgh	details edit delete
admin@pl	Buy CRMdg :)	Why should you buy Premium CRMdg? It's simple: huge development opportunities	???	details edit delete
darek@pl	Start TODAY! :) - cooperation with GDDM	Start cooperation with GDDM in the field of programming web applications.	???	details edit delete
admin@pl	Give: Like!	Give LIKE! or many LIKES :) on social portals	done	details edit delete

Rys. 22.6. CRMdg (materiał autora)

Gotowe narzędzia przeznaczone do tego typu zadań

W tej grupie można wymienić przede wszystkim dwa rodzaje systemów informatycznych:

- narzędzia do zarządzania projektami (zadaniami), czyli umożliwiające zebranie informacji o projekcie i potem skuteczne „dogłębienie” postępujących prac programistycznych. Część z nich jest bezpłatna, za inne trzeba wносить pewne opłaty. Jednymi z popularniejszych są: Trello, Asana, ClickUp, Kanban Tool;
- CRM (ang. *Customer Relationship Management*), czyli oprogramowanie ułatwiające organizację i prowadzenie kontaktów z klientami. Na rynku istnieje bardzo duży wybór tego typu rozwiązań. Część z nich jest bezpłatna, np.: Dolibarr ERP – CRM, Bitrix24, OroCRM, X2CRM.

LOGO admin ▾ Home Users Customers Schedules Tasks Management

USER UPDATE

Name

Darek

Email

darek@pl

Password

Make change

Rys. 22.7. CRMdg – dedykowane narzędzie sprzedażowe (materiał autora)

Własne narzędzia przygotowane na bazie bezpłatnego oprogramowania

W przypadku niewielkiej firmy programistycznej niekoniecznie trzeba od razu korzystać z rozwiązań dedykowanych. Można się po prostu zastanowić, jakiego rodzaju informacje będziemy zbierali, i na bazie takiej wiedzy sporządzić arkusz kalkulacyjny. Pomóc w tym może bezpłatne oprogramowanie biurowe LibreOffice Calc (źródło: <https://pl.libreoffice.org/>). Jak zrobić własny arkusz oczekiwań klienta? Przykładowe, bardzo proste rozwiązanie znajdziesz na rys. 22.8. Możesz je rozwinąć i dostosować do swoich potrzeb, wykorzystując własną wiedzę i kreatywność.

Pracownik nadzorujący kontakty z klientem	Imię i nazwisko, nr GSM pracownika
Szczegółowe informacje	
NAZWA KLIENTA	
Dane Klienta	firma...
Osoba do kontaktu	pani menedżer...
Uwagi podstawowe	...
Opis dodatkowy	...
WYGLĄD APLIKACJI	
Plik graficzny	tak, został przesłany przez klienta
Identyfikacja wizualna	
Kolor wiodący	opis
Kolor drugi	opis
Kolor trzeci	opis
Logo	dołączona specyfikacja oraz pliki .cdr, .eps
FUNKCJONALNOŚĆ APLIKACJI	
Baza danych	tak
Logowanie	tak
System rejestracji na stronie WWW	tak
Menu mobilne	tak
Menu desktop	tak
Umieszczanie ogłoszeń na stronie	tak, po zalogowaniu
Kolejne elementy listy kontrolnej	...
	...
	...
	...

Rys. 22.8. Projektowanie aplikacji – arkusz pomocniczy

Zauważ, że na tym etapie nie wspomniano o technologii wykonania projektu. Najpierw zbierzemy informacje od klienta zainteresowanego budową aplikacji, o jego potrzebach, przemyśleniach i oczekiwaniach. Przygotujemy też specyfikację techniczną (uproszczoną lub pełną), a dopiero potem dobierzemy odpowiednie środowisko informatyczne, które jest:

- nam – jako wykonawcy – znane;
- optymalne z punktu widzenia zaspokojenia potrzeb i oczekiwań klienta co do planowanej funkcjonalności i wyglądu aplikacji.

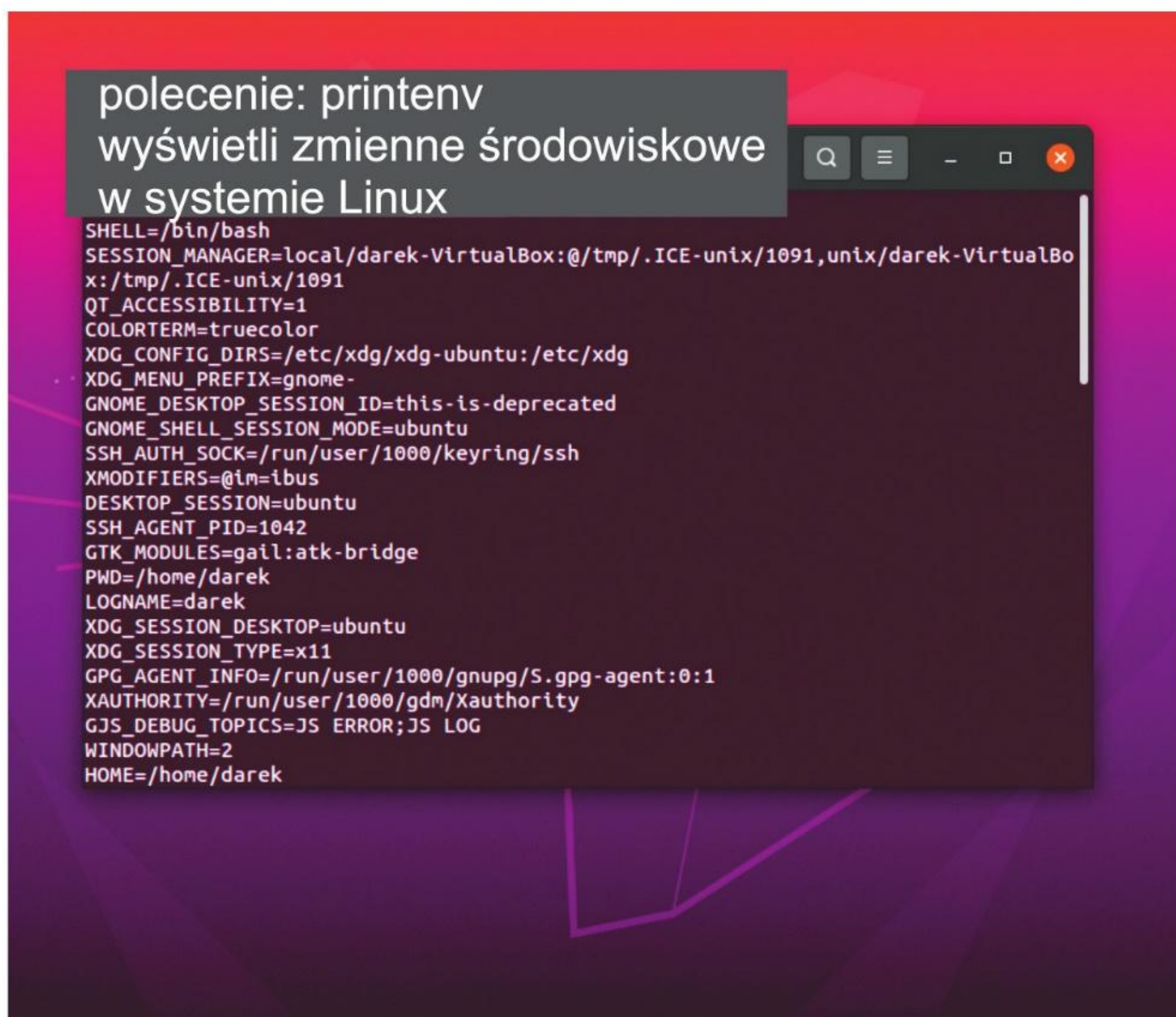
22.4. Specyfikacja projektu w programowaniu

Specyfikacja projektu to dokument, który wskazuje wymagania techniczne dla produktu i/lub usługi. W programowaniu może zawierać opis produktu informatycznego (usługi) końcowego z podziałem na:

- część graficzną, czyli wygląd aplikacji;
- funkcjonalność programu komputerowego i jego poszczególnych elementów;
- bazy technologicznej, na której powinien się opierać projekt.

O zawartości pierwszych dwóch elementów już wspomniano. Baza technologiczna powinna m.in. wskazywać:

- technologie, które będą użyte do zaprogramowania; w przypadku aplikacji webowej powinny to być m.in. informacje o:
 - rodzaju serwera i bazy danych (wraz z innymi zmiennymi środowiskowymi),
 - językach programowania dla frontendu,
 - językach programowania dla backendu;
- metodologię pisania kodu, np.:
 - programowanie strukturalne,
 - programowanie obiektowe;
- zasoby pracownicze, m.in.:
 - menedżer / menedżerowie,
 - programiści,
 - ewentualne wsparcie dla programistów (np. usługi zewnętrzne, graficzne, hostingowe);
- procedury związane z koordynowaniem tworzonego kodu projektu i zarządzania zadaniami dla członków zespołu projektowego.



Rys. 22.9. Linux – wyświetlenie zmiennych środowiskowych za pomocą polecenia **printenv**

CIEKAWOSTKA

W systemie operacyjnym Linux, gdy w terminalu użyjesz polecenia **printenv** (bez dodatkowych parametrów), to zostaną wyświetlone informacje o środowisku, którego używasz (wykaz zmiennych środowiskowych używanych w twoim systemie operacyjnym).

22.5. Paradygmaty programowania

Projekt, który wspólnie stworzymy, będzie wykorzystywał głównie instrukcje strukturalne, ale nie tylko. Łącząc się z bazą danych, skorzystamy z PDO, będącego elementem programowania obiektowego.

Programowanie strukturalne

Jest to jeden z paradygmatów programowania. Celem dla tego sposobu pisania kodu jest takie konstruowanie programu, które zmierza do osiągnięcia struktury kodu będącej odzwierciedleniem struktury rozwiązywanego problemu. Niezmiernie istotne

w programowaniu strukturalnym jest użycie kolejnych poziomów abstrakcji w ramach opracowywania kroków (kolejnych etapów) programu. Tworzenie programu rozpoczyna się od ogólnego opisu problemu, a następnie programista przechodzi do konkretnych (zagłębianych) szczegółów. W programowaniu strukturalnym w większości przypadków korzysta się tylko z kilku najważniejszych struktur sterujących:

- **iteracja** – program będzie powtarzał instrukcję w pętli do momentu, aż zostanie spełniony warunek iteracyjny;
- **wyboru ścieżki** (łącznie z cofnięciem o jeden krok) – realizacja poszczególnych instrukcji programu w zależności od jego stanu;

```
@echo off
title Prosty Kalkulator
color 05

:top
cls
echo Witaj w Prostym Kalkulatorze
set /p firstNumber=Wprowadź pierwszą liczbę:
if %firstNumber% GTR 0 echo liczba

set /p secondNumber=Wprowadź drugą liczbę:
set /p mathOper=Napisz jeden z operatorów ( -,
+, * or /):
if %mathOper%==* echo To jest mnożenie
set /a
answer=%firstNumber%%mathOper%%secondNumber%
echo Twoja odpowiedź: %answer%
echo Bardzo dobrze, brawo! ;)
echo Naciśnij dowolny klawisz, aby kontynuować!
pause>nul
goto top
```

Rys. 22.10. Przykład programowania strukturalnego – język BATCH, umożliwia tworzenie plików wsadowych w systemie operacyjnym Windows

- **sekwencja** – poszczególne instrukcje będą wykonywane „krok po kroku” według zapisanej kolejności;
- **instrukcje programistyczne** – procedury, które powodują operacje przypisania, wprowadzania danych, wyboru według założonego warunku.

Stosowanie programowania strukturalnego wymusza opisanie całej struktury programu, najważniejszych struktur danych i utworzenie niezbędnej dokumentacji, która pozwoli na zdiagnozowanie problemów, z którymi programista będzie musiał sobie poradzić w trakcie realizacji projektu.

```
var obiektLiteral =  
{  
  rodzaj: "div",  
  id: 1  
}  
  
function ObiektProto(rodzaj, id, image)  
{  
  this.rodzaj = rodzaj;  
  this.id = id;  
  this.image = image;  
}  
  
ObiektProto.prototype.pokazWKonsoli = function()  
{  
  console.log("rodzaj: " + this.rodzaj);  
  console.log("id: " + this.id);  
  console.log("image: " + this.image);  
}
```

Rys. 22.11. Przykład programowania obiektowego – język JavaScript



CIEKAWOSTKA

Obiekty w języku programowania JavaScript wywodzą się od prototypu obiektów klasy Object – jest to pierwotny, nadrzędny obiekt. Można go sobie wyobrazić jako rodzica wszystkich obiektów. Wszystkie inne obiekty dziedziczą metody i pola po Object.prototype. Większość z nich może zostać przeciążonych (poza Object.create(null)). Pamiętaj, że zmiany prototypu Object są propagowane do innych obiektów, chyba że dane metody i pola zostały przeciążone w prototypie klasy bazowej. Więcej na ten temat możesz się dowiedzieć m.in. na stronie:

https://developer.mozilla.org/pl/docs/Web/JavaScript/Reference/Global_Objects/Object.

Programowanie obiektowe

Programowania obiektowe zostanie wytłumaczone na podstawie rys. 22.12, na którym przedstawiono najważniejsze idee z tego rodzaju pisania kodu.

Każdy obiekt ma określone:

- metody (funkcje),
- właściwości.

Wyobraź sobie obiekt ‘człowiek’. Jego metody to np.: je, chodzi, skacze. Z kolei właściwościami człowieka są: kolor włosów, rodzaj uszu, nos, wąsy itd.

Więcej informacji otrzymasz już wkrótce, tworząc w PHP połączenie z bazą danych z wykorzystaniem PDO. Będzie ono wykonane w oparciu o programowanie obiektowe.



Rys. 22.12. Programowanie obiektowe – podstawowe założenia

SPRAWDŹ SWOJĄ WIEDZĘ

1. Opisz, jak powinna wyglądać profesjonalna obsługa klienta.
2. Na czym polega analiza wymagań klienta?
3. Wymień dostępne narzędzia analityczne związane z projektami i klientami. Podaj co najmniej trzy.
4. W tabeli wskazanej jako arkusz pomocniczy do planowania projektu dopisz brakujące elementy listy kontrolnej.
5. Scharakteryzuj specyfikację projektu.
6. Wskaż różnice między programowaniem strukturalnym i programowaniem obiektowym.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Opierając się na paradygmacie programowania strukturalnego, napisz w języku JavaScript program, który wyświetli na stronie internetowej (w dowolnej przeglądarce) kalkulator. Powinien on obsługiwać co najmniej cztery podstawowe działania matematyczne:
 - dodawanie,
 - odejmowanie,
 - mnożenie,
 - dzielenie.Strukturę strony internetowej możesz stworzyć w języku programowania HTML, a jej wygląd zaprojektować w języku stylów – CSS.
2. Stwórz w języku JavaScript podobny program do opisanego w zadaniu 1., ale tym razem użyj zasad programowania obiektowego.
3. Przeanalizuj kody z zadań 1. i 2. Stwórz dwukolumnową tabelę (w edytorze tekstu lub arkuszu kalkulacyjnym). Wypisz w niej:
 - w lewej kolumnie – elementy charakterystyczne dla tworzenia kalkulatora na zasadach strukturalnych,
 - w prawej kolumnie – elementy charakterystyczne dla tworzenia kalkulatora na zasadach obiektowych.

23

Projektowanie aplikacji klient–serwer – podstawowa specyfikacja projektu SOAW

ZAGADNIENIA

- Filozofia Agile
- Podstawowa specyfikacja projektu SOAW
- Projektowanie interfejsu użytkownika i wyglądu aplikacji
- Dostosowanie interfejsu do różnych platform sprzętowych

23.1. Filozofia Agile

Na początku tego rozdziału nasuwają się na myśl pytania: Czym jest specyfikacja techniczna? Do czego służy? Jak ją przygotować? Kto i dlaczego powinien ją opracować? Jak obszerna powinna być profesjonalna dokumentacja?

W tradycyjnym świecie projektowania, tzn. w zamierzonych czasach, gdy internet i programowanie nie istniały (lub prawie nie istniały), zdecydowana większość projektantów i menedżerów uważała, że: „Profesjonalna, naprawdę dobra specyfikacja musi mieć co najmniej kilkadziesiąt (a jeszcze lepiej kilkaset) stron”. Świat nowoczesnych technologii nie byłby w stanie się tak szybko rozwijać, gdyby zastosował się do powyższego założenia. Potrzebne było coś nowego, łamiącego tradycyjne standardy. Właśnie wtedy narodziła się **filozofia Agile**, która przede wszystkim jest stosowana w zarządzaniu projektami.

Czasy szybkiego rozwoju technologii komputerowych charakteryzują się m.in. tym, że co kilka tygodni lub co najwyżej miesięcy pojawiają się nowe rozwiązania informatyczne, a czasem wręcz zupełnie nowe języki programowania. Niektóre technologie, jak chociażby frameworki JavaScript, rozwijają się niesamowicie szybko. Wystarczy popatrzeć na ich liczbę lub choćby na liczbę wersji jednego z najlepszych i największych przedstawicieli tej grupy – frameworka Angular (źródło: <https://angular.io/>). Pojawił się on (wersja ostateczna) w 2016 roku jako Angular 2, ale w 2020 roku premierę miały już trzy kolejne jego wersje: Angular 9, Angular 10 i w listopadzie – Angular 11.

Powyższy przykład pokazuje szybkość zmian w świecie technologicznym. To dlatego poleganie na „starodawnym” podejściu do specyfikacji (opasłe tomy z dokumentacją) jest nieracjonalne. Może się okazać, że produkt, który otrzymamy, **będzie przestarzały w momencie wdrożenia (a może nawet w trakcie kończenia rozbudowanej specyfikacji)**. W związku z tym świat potrzebował zupełnie nowego podejścia do projektowania nowoczesnych technologii, określanego jako filozofia Agile (znana także jako programowanie zwinne).



Rys. 23.1. Agile – programowanie zwinne

O Agile można pisać i mówić bardzo obszernie, ale to przeczyłoby jej samej. Fundamenty opisywanego podejścia zwinnego najlepiej poznać „u źródła”, czyli na oficjalnej stronie *Manifestu Agile*: <https://agilemanifesto.org/>.

Manifest programowania zwinnego

Odkrywamy nowe metody programowania dzięki praktyce w programowaniu i wspieraniu w nim innych.

W wyniku naszej pracy zaczęliśmy bardziej cenić:

- **Ludzi i interakcje** od procesów i narzędzi.
- **Działające oprogramowanie** od szczegółowej dokumentacji.
- **Współpracę z klientem** od negocjacji umów.
- **Reagowanie na zmiany** od realizacji założonego planu.

Oznacza to, że elementy wypisane po prawej stronie są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej¹.

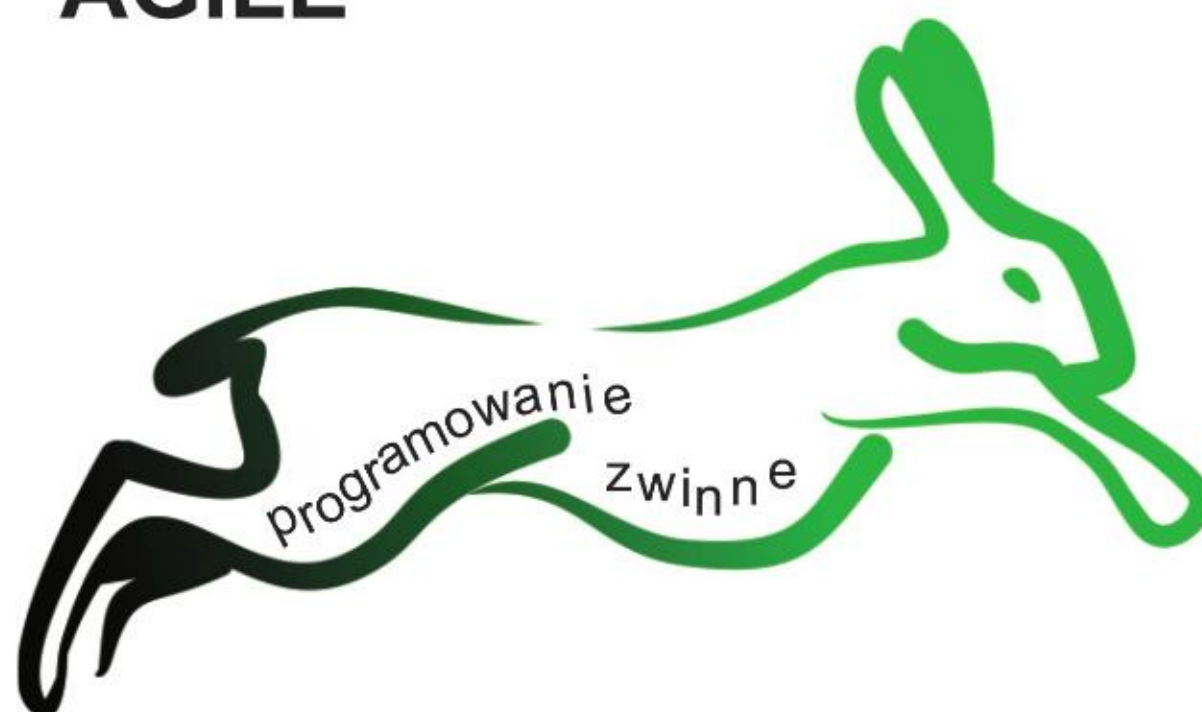
Gdy już przeczytasz powyższy manifest, zwróć szczególną uwagę na wnioski płynące z ostatniego zdania: „[...] elementy wypisane po prawej stronie są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej”. Rozwijając myśl twórców wymienionej filozofii, warto podkreślić, że najważniejsze są:

- ludzie i interakcje,
- działające oprogramowanie,
- współpraca z klientem,
- reagowanie na zmiany.

¹ Źródło: <https://agilemanifesto.org/iso/pl/manifesto.html> [dostęp: 11.02.2020].

FILOZOFIA

AGILE



Rys. 23.2. Agile – programowanie zwinne

Osoby (klienci), które będą brały udział w opracowywaniu koncepcji projektu, i cały zespół deweloperski muszą pojmować w ten sam sposób założenia technologiczne i funkcjonalne aplikacji. Trzeba zadbać, aby:

- wymagania, założenia i cele projektu były zrozumiałe w tym samym stopniu dla humanistów i programistów, dla klienta i dla podwykonawcy;
- możliwe było dokonanie wyceny wraz z opracowaniem planu projektu – harmonogramu, listy i kolejności prac do wykonania;
- specyfikacja projektu była maksymalnie krótka, ale dobrze doprecyzowana, powinna mieć swój odpowiednik w warstwie obrazkowej.

Agile (a dokładniej jej projektowe ujęcie – SCRUM) opiera się na kolejnych iteracjach, podczas których kawałek po kawałku jest wytwarzana i testowana niewielka część oprogramowania. Umożliwia to szybką reakcję na zmiany i ewentualne niepowodzenia, a także modyfikację założeń, możliwą do przeprowadzenia nawet na późniejszych etapach budowy i rozwoju projektu.

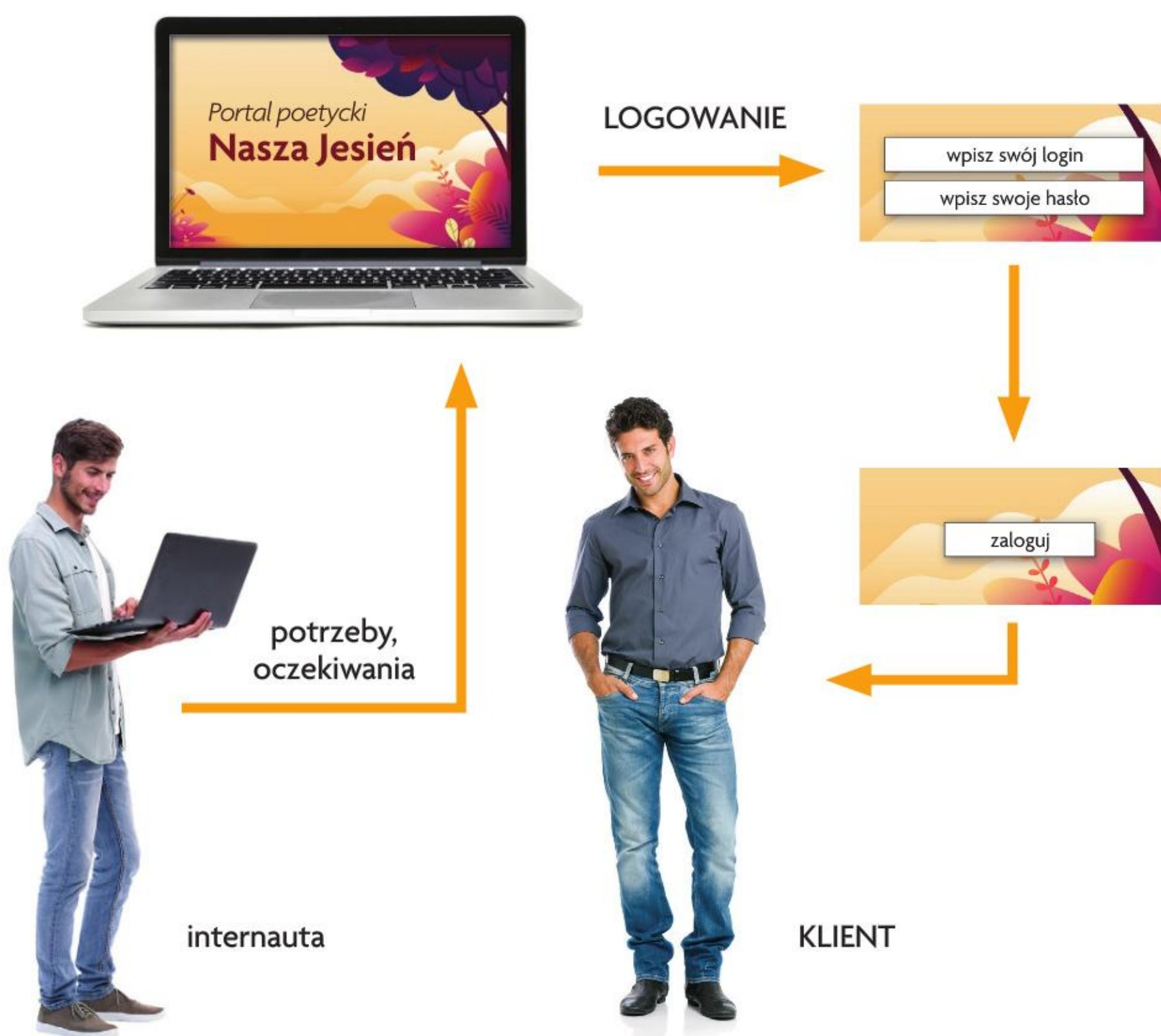
23.2. Podstawowa specyfikacja projektu SOAW

Zalety zwinnego podejścia to: „prosto, zrozumiale i szybko”. Nie ma bardziej zrozumiałego sposobu określenia założeń funkcjonalnych aplikacji webowej lub mobilnej niż **historie użytkownika** (ang. *user stories*), które opiszą z punktu widzenia użytkownika, jaka funkcja aplikacji jest potrzebna i dlaczego. Przykładowo: jako użytkownik wchodzę na stronę projektu, klikam przycisk **Rejestracja**, przechodzę do określonej strony, na której wypełniam formularz i... już jestem zarejestrowany. Mogę korzystać z aplikacji.

Gdy już jestem użytkownikiem, przechodzę na stronę, wpisuję login, hasło i klikam przycisk zaloguj.

To samo można pokazać za pomocą historyjki opisowej – także z punktu widzenia administratora systemu – jako administrator. Mogę zalogować się na konto, aby edytować każdy dowolny wpis i usunąć go, gdy nie spełnia on standardów portalu.

Trzeba także pamiętać o założeniach technologicznych – kilka zdań o tym, czym ma być tworzony program. Aplikacją mobilną na platformy Android, iOS? A może działającą offline apką PWA czy też może aplikacją webową – bazodanową, równie dobrze działającą we wszystkich popularnych przeglądarkach internetowych? Tego typu zagadnienia, poza opisem słownym, należy pokazać w postaci historyjek obrazkowych. Przykład znajdziesz na rys. 23.3.



Rys. 23.3. Logowanie – historyjka projektowa opracowana w stylu założeń filozofii Agile

Agile – podsumowanie

Model Agile zakłada „partnerstwo w drodze do celu”, czyli współpracę między firmą projektową a klientem. Jest to przedsięwzięcie, które wymaga zaangażowania w proces powstawania aplikacji zarówno klienta, jak i wykonawcy. Zgodnie z założeniami filozofii Agile na takim podejściu zyskują obie strony:

- klient, który otrzyma produkt idealnie dopasowany do swoich potrzeb (ważne jest także to, że biorąc udział w tworzeniu projektu, przy okazji może poznać nowoczesne technologie projektowania i programowania);
- zespół deweloperski, który dzięki odpowiedniemu zrozumieniu oczekiwań klienta, stworzy produkt skrojony na miarę jego faktycznych potrzeb, co zapewne przełoży się na sukces organizacyjny i finansowy firmy programistycznej).

23.3. Projektowanie interfejsu użytkownika i wyglądu aplikacji

Historyjki obrazkowe (obok krótkiego opisu) to najlepsza metoda na zobrazowanie projektu interfejsu. Rysuj, rysuj, rysuj i opowiadaj tak, żeby wszyscy wszystko dokładnie zrozumieli. Nie dodaliśmy jeszcze tylko jednego bardzo ważnego elementu: jakość graficzna historyjek agilowych. Powinna być taka, jaką użytkownik potrafi zrobić. Ma być zrozumiała dla wszystkich, a nie być arcydziełem, które wyślemy na konkurs graficzny. Nie powinieneś przejmować się ewentualnym brakiem talentu graficznego.

23.4. Dostosowanie interfejsu do różnych platform sprzętowych

Gdy już wiemy, co jest do stworzenia, tj. przygotowania grafiki i zaprogramowania funkcjonalności, musimy się zastanowić nad technologią, która jest nam dobrze znana. Dzięki niej będziemy mogli sprostać potrzebom i oczekiwaniom klienta.

W kodach HTML i CSS znajdziesz odpowiedzi na ewentualne pytania dotyczące wyglądu poszczególnych bloków i tzw. responsywności. W kodach CSS zwróć szczególną uwagę na zapytania o media i pojęcie grid. W dużej mierze odpowiednio wykorzystując tylko te dwa elementy, możesz uzyskać poprawny wygląd aplikacji na różnych ekranach (o różnej rozdzielczości).



SPRAWDŹ SWOJĄ WIEDZĘ

1. Wytlumacz główne założenia filozofii Agile.
2. Wymień elementy interfejsu użytkownika.
3. Co to jest responsywność strony WWW?
4. Czym charakteryzuje się programowanie zwinne?
5. Stwórz na kartce papieru historyjkę obrazkową: „Ja – programista języka (tutaj wpisz język programowania, który na razie najbardziej lubisz)”.

24

Aplikacja SOAW: interfejs, wygląd, funkcjonalność

ZAGADNIENIA

- Interfejs aplikacji
- Wygląd aplikacji na bazie SOAW
- Funkcjonalność aplikacji na bazie SOAW

24.1. Interfejs aplikacji

Aplikacja bazodanowa składa się zwykle z trzech podstawowych obszarów:

- **panel administratora** – dostępny wyłącznie dla administratora;
- **panel użytkownika** – dostępny dla użytkownika zarejestrowanego i administratora aplikacji;
- **interfejs internetowy** – dostępny dla każdego internauty, użytkownika zarejestrowanego i administratora aplikacji.

```
// Zaprogramowanie zdarzeń dla interfejsu gry
edukacyjnej
document.getElementById("wylosowanyTekst").innerHTML
= allElementy[wylosowanaLiczba];
document.getElementById('batonMmixB').addEventListener
('click', przyciskKliknij1);
document.getElementById('batonMmixC').addEventListener
('click', przyciskKliknij2);
document.getElementById('batonMmixD').addEventListener
('click', przyciskKliknij3);
document.getElementById('batonMmixE').addEventListener
('click', przyciskKliknij4);
console.log('liczba elementów w tablicy allElementy.
length: ' + allElementy.length);
document.getElementById("doZgadniecia").innerHTML =
pierwszaGrupaAll.length + drugaGrupaAll.length +
trzeciaGrupaAll.length + czwartaGrupaAll.length;
```

Rys. 24.1. Przykładowy kod JavaScript – programowanie zdarzeń dla interfejsu graficznego gry edukacyjnej

Wskazane elementy projektu webowego zazwyczaj są od siebie rozdzielone **middleware**, czyli warstwą pośrednią kodu programistycznego, który najczęściej umożliwia administratorowi dostęp do wszystkich poziomów aplikacji. Oczywiście dzieje się tak z zachowaniem zasad obowiązującego prawa, czyli np. nie ma możliwości, aby administrator miał dostęp do haseł użytkowników bądź informacji prywatnych, gdyby takowe były przechowywane w systemie na kontach.

Menu aplikacji

Tworzy się je po to, aby w prosty sposób użytkownik mógł się poruszać po dostępnych zasobach aplikacji. **Menu** (jak i cała reszta) powinno być tak wykonane, aby równie efektywnie mogła się przemieszczać między stronami osoba, która korzysta z dużego monitora (tzw. klient desktop), jak i ta, która serfuje z wykorzystaniem urządzenia mobilnego, czyli ma (zwykle) ekran o małej szerokości. Dodatkowym problemem do rozwiązania jest sposób używania smartfonów: w pionie (ang. *vertical*) i w poziomie (ang. *landscape*). Programista powinien przemyśleć i zaprogramować właściwy wygląd dla każdego z wymienionych sposobów wykorzystania urządzeń mobilnych.



Rys. 24.2. Przykładowe menu poziome strony internetowej – technologia SPA (ang. *Single-Page Application*)

Kreatywność projektantów stron i aplikacji jest bardzo duża. W internecie można znaleźć wiele różnych rozwiązań graficzno-funkcjonalnych związanych ze sposobem prezentacji i wykorzystania menu. Poniżej znajdziesz opis kilku popularnych rodzajów wyglądu i funkcjonowania menu. Osiągnięcie wielu zachwycających efektów umożliwia nam JavaScript.

Zwykle język JavaScript służy do programowania tego typu technologii, ale nie tylko. W zdecydowanej większości przypadków JS obsługuje tzw. zdarzenie, np. kliknięcie w element menu. Co ważne, za zmiany, które potem następują w wyglądzie strony i menu, bardzo często odpowiadają wspólnie JavaScript i język stylów internetowych CSS. Jeżeli nie „czujesz się zbyt pewnie” w zakresie solidnych podstaw programowania w wymienionym CSS i/lub HTML, to warto sięgnąć do źródła, czyli na portale zawierające wiedzę webową, z której zasobów czerpią programiści z całego świata:

- <https://developer.mozilla.org/pl/> (strona MDN Web Docs) – „zasoby dla deweloperów przez deweloperów” – takie jest hasło portalu; znajdziesz tam mnóstwo bardzo wartościowych informacji na temat technologii webowych;
- <https://www.w3.org/> (World Wide Web Consortium, W3C) – organizacja, która zajmuje się ustanawianiem standardów internetowych: tworzenia i przesyłu stron WWW.

Hamburger menu

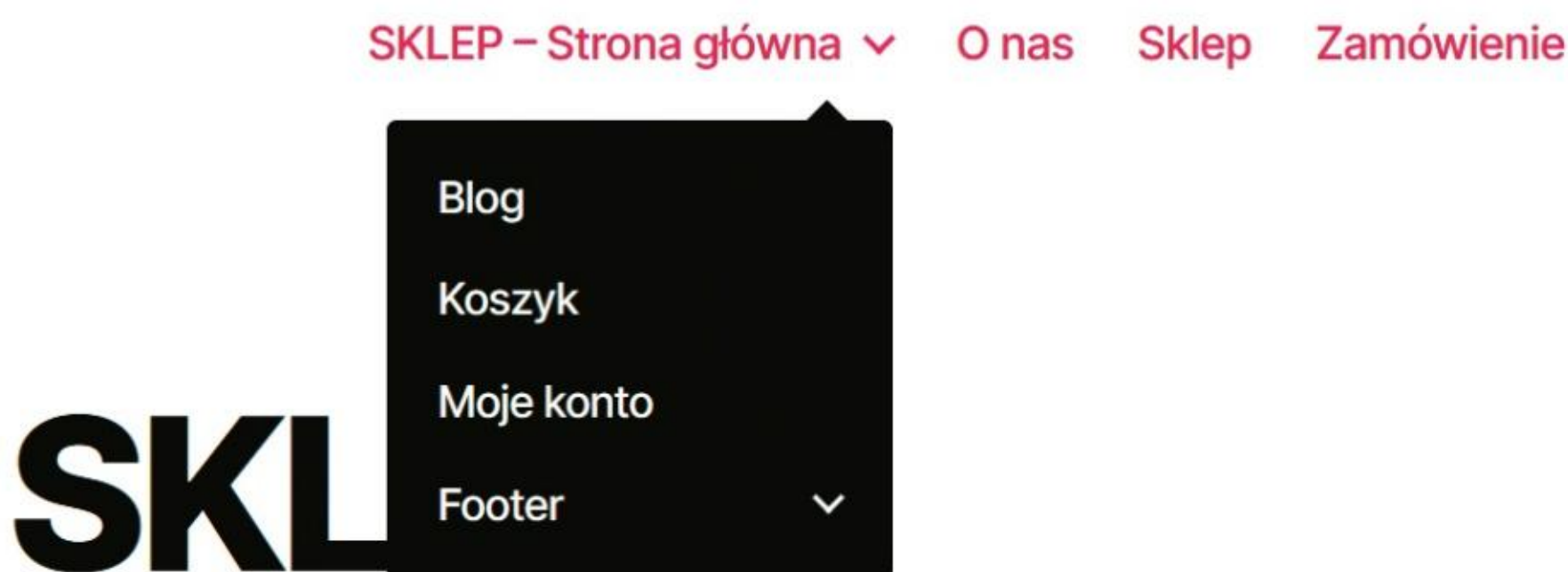
W przypadku urządzeń mobilnych najbardziej popularny jest **hamburger menu**, czyli menu w postaci niewielkiego prostokąta, który jest interaktywny, tzn. po kliknięciu w niego lewym przyciskiem myszy pojawia się menu. Zwykle na widoku strony przeznaczonym dla urządzeń desktop (zwykle i duże ekrany komputerów) na stronie jest ładne, duże menu, ale w wersji dla urządzeń przenośnych duże menu znika, a serwowane jest właśnie „hamburger menu”. Jedyną wadą tego rozwiązania jest potrzeba stworzenia i utrzymywania dwóch oddzielnych kodów: dla menu tradycyjnego (zwykle ekrany) i mobilnego (ekrany o małej szerokości). Ten niewielki wysiłek programisty daje wielkie zyski klientowi w postaci wygody użytkowania tak zaprogramowanego interfejsu.



Rys. 24.3. Przykładowe „hamburger menu” – przycisk uruchamiający np. menu mobilne aplikacji

Menu rozwijalne poziome

Menu rozwijalne jest szczególnie popularne na portalach blogowych, firmowych i informacyjnych, czyli tam, gdzie istnieje potrzeba prezentacji dużej ilości treści i obrazków, co przekłada się zwykle na dużą liczbę stron. Bardzo często wykorzystywanym standardem jest **menu poziome**, które jest interaktywne. Po kliknięciu w niego lewym przyciskiem myszy rozwijają się kolejne elementy menu.

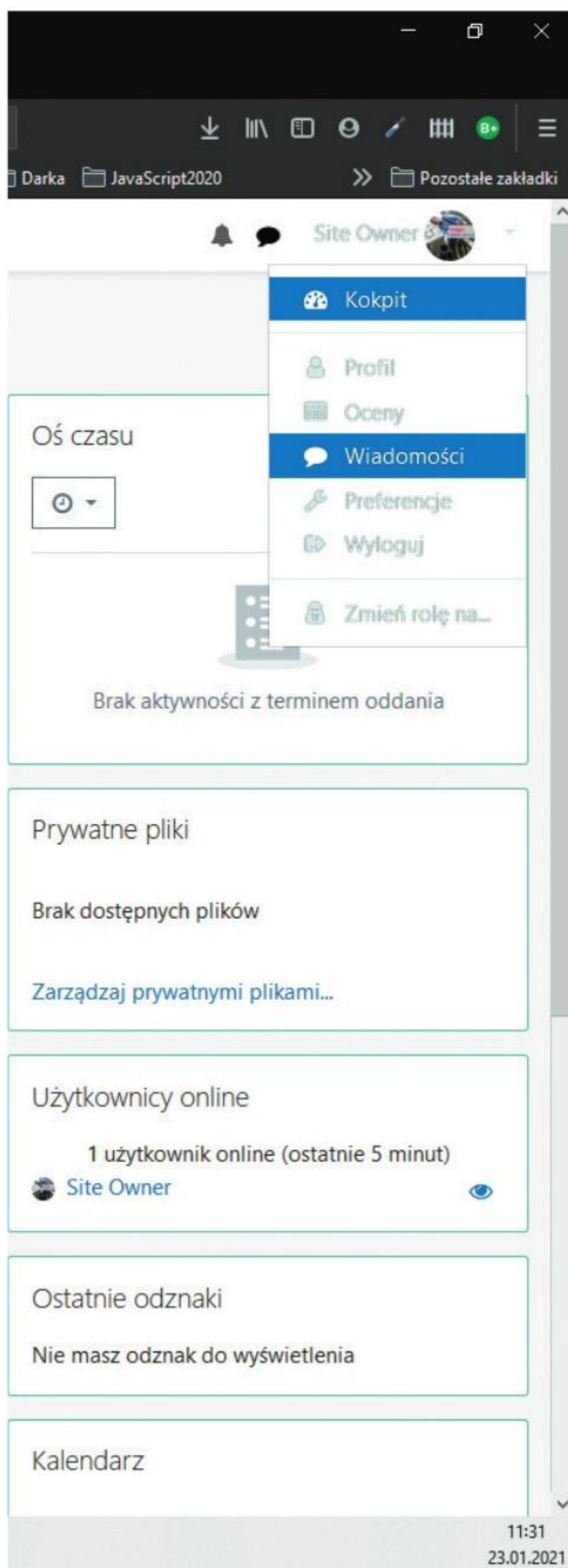


Rys. 24.4. Menu poziome rozwijalne – aplikacja CMS

Menu rozwijalne pionowe

Szczególnie często występuje ono w obszarach administracyjnych portali. Na rys. 24.5 przedstawiono przykład takiego menu, zaczerpnięty z portalu e-learning, zbudowanego na LMS MOODLE.

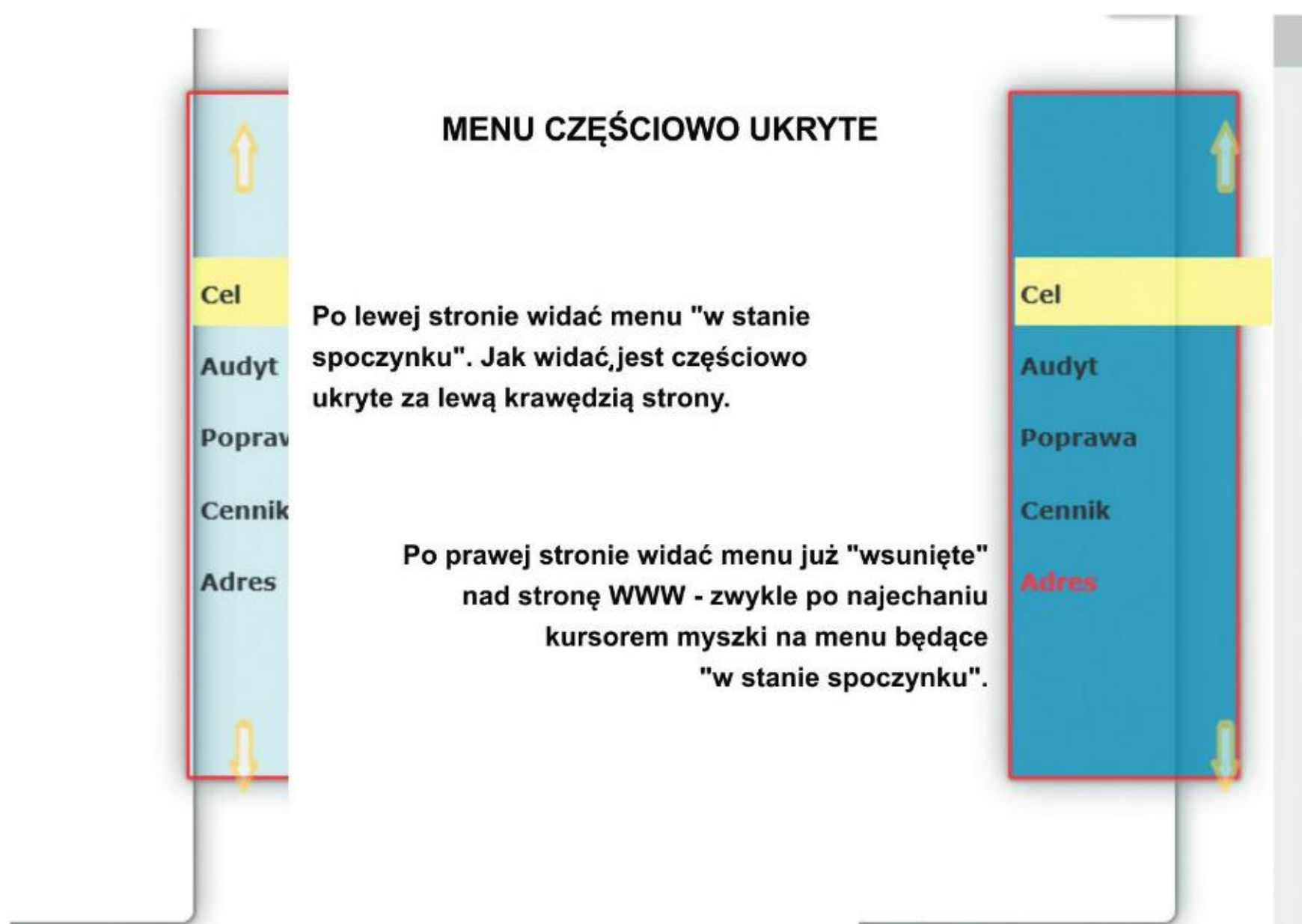
Menu rozwijalne pionowe jest popularne na portalach blogowych, firmowych i informacyjnych, czyli tam, gdzie istnieje potrzeba prezentacji dużej ilości treści i obrazków, co przekłada się zwykle na dużą liczbę stron. Bardzo częstym standardem jest menu interaktywne, tzn. po kliknięciu w niego lewym przyciskiem myszy rozwijają się kolejne elementy menu.



Rys. 24.5. Panel administracyjny aplikacji – menu rozwijane pionowe

Menu częściowo ukryte

Na rys. 24.6 przedstawiono menu częściowo ukryte poza prawą krawędzią strony. Widok z lewej strony pokazuje menu „w stanie spoczynku”, widok po prawej pokazuje menu po wsunięciu się nad stronę – w wyniku wcześniejszego najechania (pozycja myszki nad danym elementem) na nie myszką. Animację wykonano w języku JavaScript.



Rys. 24.6. Menu częściowo ukryte poza prawą krawędzią strony

Ten rodzaj menu pozwala na uzyskanie oszczędności na stronie, gdyż menu zajmuje niewiele miejsca (większość lub duża część jest poza stroną), a równocześnie nie nakłada na programistę podwójnej pracy, jak w przypadku „hamburger menu”. Dodatkowym elementem przemawiającym za takim menu może być efektowna animacja pojawiającego się menu, która może być **wodotryskiem**, wyróżniającym stronę od innych konkurencyjnych rozwiązań i świadczącym o nowoczesności firmy mającej takie rozwiązania na firmowej WWW.

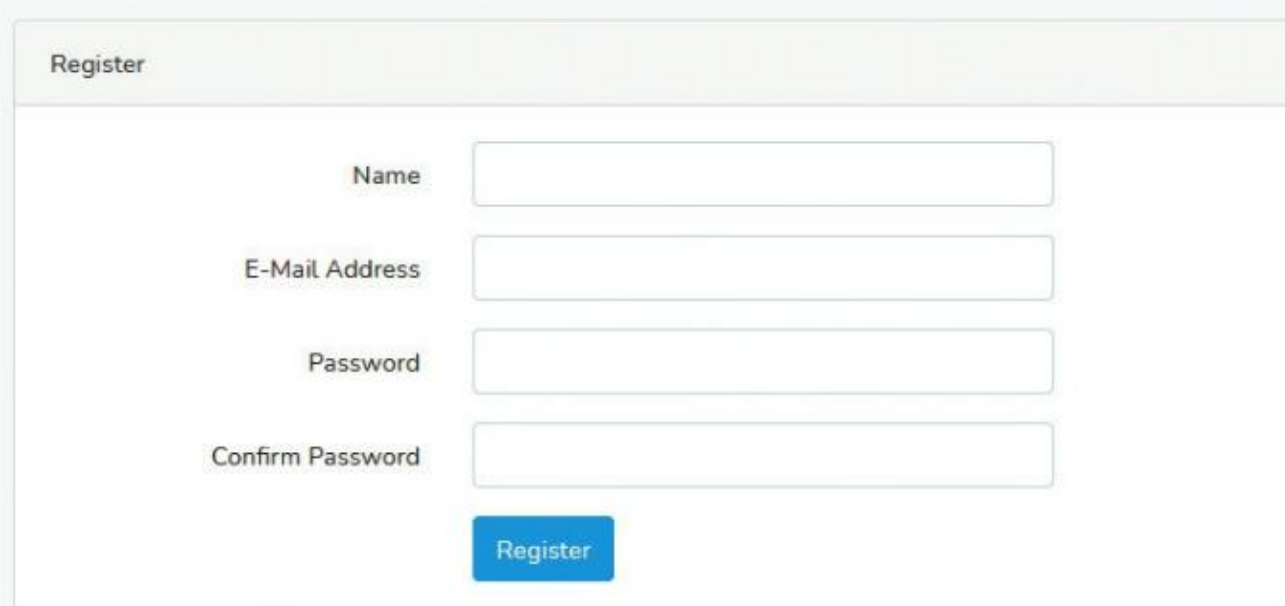
Formularze

Są to elementy strony internetowej (interfejs – technologia HTML), które umożliwiają przesłanie informacji do bazy danych bądź też protokołem obsługującym e-mail – do innego użytkownika. Przykładami wykorzystania tego elementu projektowania webowego mogą być funkcjonalności aplikacji:

- zalogowanie się do aplikacji przez zarejestrowanego użytkownika;
- założenie konta użytkownika na portalu internetowym przez dowolnego internautę.

CRMdg2

Login Register



Register

Name

E-Mail Address

Password

Confirm Password

Rys. 24.7. Formularz logowania do systemu CRM

Paski narzędziowe

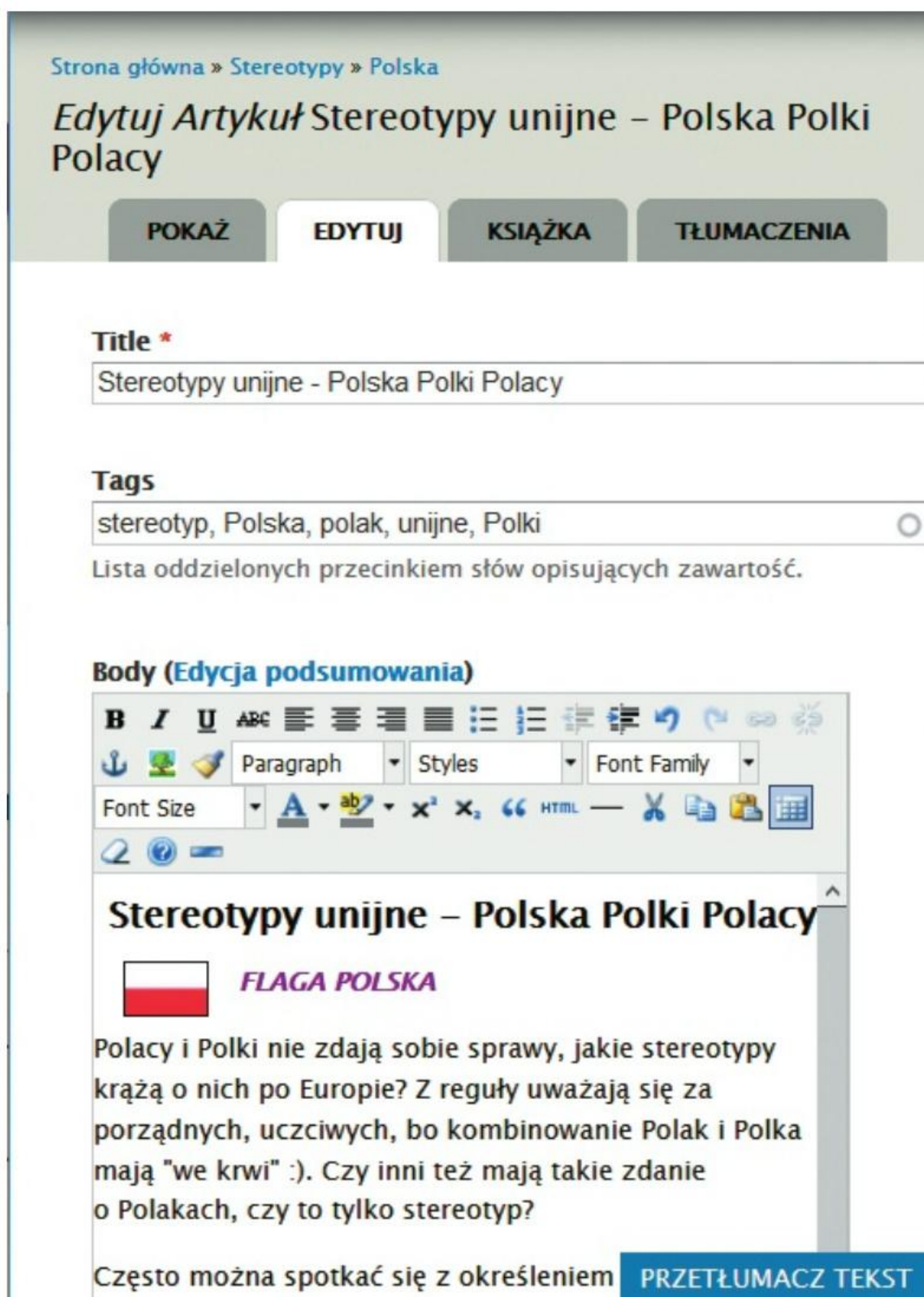
Przykładem paska z narzędziami może być technologia edytora tekstu dostępna do zaimplementowania wewnątrz tworzonej aplikacji. Na rys. 24.8 znajdziesz przykładowe rozwiązanie zaimplementowane wewnątrz jednego z popularnych systemów webowych.

Jedną z przykładowych technologii w tym zakresie jest CKEditor (szczegóły znajdziesz na stronie projektu: <https://ckeditor.com/>), dostarczający narzędzi do edycji i formatowania tekstu wewnątrz aplikacji webowych. Ten produkt IT jest edytorem WYSIWYG online, który umożliwia tworzenie tekstów na stronach WWW lub w aplikacjach online. Wspomniany edytor opiera się na JavaScript. CKEditor jest dostępny na licencjach *open source* i/lub komercyjnych. Tego typu rozwiązań jest wiele na rynku. Drugim pod względem popularności jest TinyMCE (<https://www.tiny.cloud/docs/quick-start/>). Warto wymienić również nieco mniej znane, co wcale nie oznacza, że jakościowo gorsze:

- <http://www.wymeditor.org/>,
- <https://clarle.github.io/yui3/yui/docs/editor/>,
- <https://cksource.com/>.

Tego typu rozwiązania nie ma sensu pisać od nowa, wystarczy zaimplementować do własnego projektu już gotowe kody (oczywiście pamiętając o prawach autorskich). Do tego służy **API** (ang. *Application Programming Interface*), czyli interfejs programowania aplikacji. Jest to zbiór reguł dokładnie opisujący, w jaki sposób programy komunikują się ze sobą. Jednym z najczęściej używanych języków programowania do zaprogramowania i wykorzystywania API jest JavaScript.

WYSIWYG to skrótowiec powstały z połączenia pierwszych liter wyrazów ze zdania „What You See Is What You Get”. Jest on używany w informatyce do określenia metod pozwalających uzyskać wynik w publikacji prawie taki sam (lub nawet taki sam) jak obraz na ekranie. W naszym kraju można spotkać spolszczoną wersję tego hasła: „Chwyć, Przeciągnij, Upuść, Zobacz” – CPUZ.



Rys. 24.8. Przykład edytora tekstu zaimplementowanego wewnątrz aplikacji internetowej

Kontrolki, pola dialogowe, tekstowe itp.

Technologia HTML 5 ma całkiem pokaźny zasób różnego rodzaju już gotowych elementów, które mogą być wykorzystane przez twórcę aplikacji do komunikacji i interakcji z użytkownikiem. Pełen wykaz elementów HTML 5 znajdziesz np. na stronie MDN, która jest „skarbnicą wiedzy” dla web dewelopera: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element>. Warto tam sięgnąć, tym bardziej że pod koniec wspomnianej strony umieszczono tzw. elementy przestarzałe, które – co prawda – częściowo jeszcze działają (lub już nie działają) na stronach WWW, ale zdecydowanie nie powinny być już używane, gdyż nie są zgodne ze specyfikacją HTML 5.

LOGO Hi, admin ▾

UPDATE TASK

User Email

Task Name

Description

Status

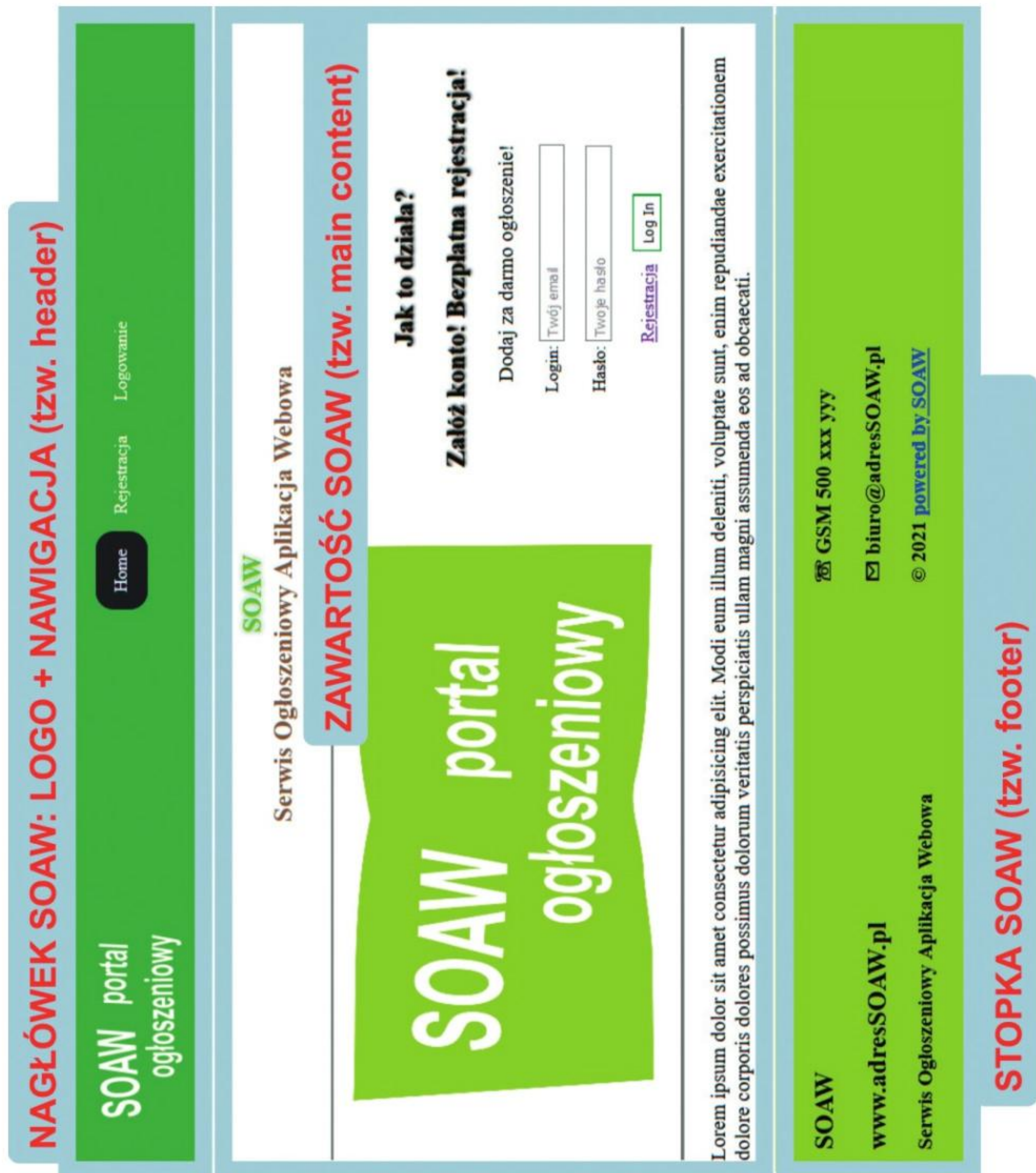
Make Change

Rys. 24.9. Przykład podstawowego edytora tekstu zaimplementowanego wewnątrz aplikacji internetowej

24.2. Wygląd aplikacji na przykładzie SOAW

Budowana aplikacja składa się z następujących części:

1. Nagłówek aplikacji zawierający logo i nawigację.
2. Zawartość główna aplikacji – powinna zawierać treści, które chcemy przekazać. W naszym wypadku umieściliśmy także formularz logowania dla użytkownika zarejestrowanego.
3. Stopka – zwykle ta część zawiera informacje o firmie zarządzającej daną aplikacją wraz z zamiarami na ten podmiot. Często w tym miejscu pojawia się także informacja o prawach autorskich do aplikacji. Mogą one należeć do podmiotu użytkującego aplikację albo do firmy, która ją wyprodukowała. Zależy to oczywiście od zapisów w umowie między stronami biznesowymi.



Rys. 24.10. Planowany wygląd graficzny aplikacji SOAW

Przedstawiony podział strony WWW jest jednym z najprostszych spośród spotykanych w rzeczywistości. Do budowy prostej aplikacji w zupełności wystarczy. Taka budowa modułowa jest spowodowana powtarzalnością pewnych elementów strony internetowej. Załącz, że informacje, które znajdują się w nagłówku, i te, którą zostaną umieszczone w stopce, będą się powtarzały na prawie każdej podstronie. W takim razie wystarczy stworzyć pliki .php po jednym dla nagłówka i dla stopki, a następnie odpowiednim poleceniem języka PHP `include_once` zaimportować je do danej strony. Przykłady przedstawiono na rys. 24.11.


```
<!DOCTYPE html>
<html lang="pl">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Serwis Ogłoszeniowy Aplikacja Webowa</title>
  <link rel="stylesheet" href="css/style.css">
</head>

<body>

  <!-- Import - dołączenie funkcjonalności - pliku zawierającego
  HEADER -->
  <?php include_once 'templates/headerUser.php'; ?>
```

Rys. 24.11. Kod PHP – dołączenie nagłówka aplikacji SOAW

```
  <!-- CONTENT - zawartość strony END -->

  <!-- FOOTER - stopka strony -->
  <?php include_once 'templates/footerForUser.php';
  ?>

</body>

</html>
```

Rys. 24.12. Kod PHP – dołączenie stopki aplikacji SOAW

Efektem powyższych działań będzie stworzenie szablonu, w którym góra strony (w postaci nagłówka) i dół strony (w postaci stopki) będą niezmiennie.

Dzięki takiemu podejściu (modularnemu) do budowy aplikacji możemy dołączać nie tylko nagłówki czy stopkę, ale także inne wybrane elementy. Zaoszczędza to czas przede wszystkim przeznaczany na dalszy rozwój aplikacji.

Pozostaje nam jeszcze element środkowy, czyli zawartość główna aplikacji, która będzie wyposażona w różne funkcje na poszczególnych podstronach budowanej aplikacji.

24.3. Funkcjonalność aplikacji na bazie SOAW

Podstawowa funkcjonalność budowanej aplikacji to **system logowania**. W kolejnym rozdziale tego podręcznika zaprogramujemy krok po kroku ten element aplikacji w HTML, PHP i SQL.



SOAW portal ogłoszeniowy

Jak to działa?

Zalóż konto! Bezplatna rejestracja!

Dodaj za darmo ogłoszenie!

Login:

Hasło:

[Rejestracja](#)

Rys. 24.13. Kod PHP – logowanie do aplikacji SOAW

Warto dodać, że tworząc system logowania, zadbamy także o podstawowe elementy bezpieczeństwa aplikacji internetowej, m.in.:

- funkcja **filter_var()** – filtruje dostarczoną jej zmienną określonym filtrem, może obsłużyć:
 - sanityzację – **filter_var(\$email, FILTER_SANITIZE_EMAIL)**,
 - walidację – **filter_var(\$email, FILTER_VALIDATE_EMAIL)**;
- **password_hash()** – używa jednokierunkowej, silnej funkcji mieszającej do wygenerowania skrótu, a następnie zapisania go w łańcuchu znaków;
- **PDO()** – poleceniem **new PDO(...)** stworzymy nowy obiekt PDO, czyli wykorzystując zasady programowania obiektowego, ustanowimy nowe, bezpieczne połączenie z bazą danych.

SOAW portal
ogłoszeniowy

[Home](#)
[Rejestracja](#)
[Logowanie](#)

Zarejestruj się

Email Address

Wprowadź poprawny adres email

Hasło

Wprowadź hasło: min 8 znaków, maks 12 znaków

Powtórz Hasło

Wprowadź hasło- musi być takie samo jak powyżej

Imię

Wpisz Imię: min 2 litery, maks 18 liter

Nazwisko

Wpisz Nazwisko: min 2 litery, maks 18 liter

Telefon

Wprowadź poprawny numer telefonu: 9 znaków

☐

Akceptuję regulamin [Zobacz Regulamin](#)
Only for human :)

☐ Nie jestem robotem


reCAPTCHA
Prywatność - Warunki

Wyślij

[Powrót na stronę główną](#)

SOAW
www.adresSOAW.pl
Serwis Ogłoszeniowy Aplikacja Webowa

 GSM 500 xxx yyy
 biuro@adresSOAW.pl
© 2021 [powered by SOAW](#)

Rys. 24.13. Widok systemu rejestracji do portalu SOAW – proponowany projekt graficzny

System rejestracji to kolejny element, który powinien zostać stworzony w ramach projektu SOAW. Przykładowy wygląd tego elementu aplikacji przedstawiono na rys. 24.13. Kod PHP i SQL będziesz mógł stworzyć osobiście (po przerobieniu tego podręcznika), wykorzystując także CRUD (ang. *Create Read Update Delete*).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień elementy interfejsu aplikacji, które można wykonać za pomocą elementów HTML 5.
2. Opisz wygląd aplikacji SOAW.
3. Wskaż trzy najważniejsze funkcje aplikacji SOAW.
4. Napisz kod PHP, który spowoduje dołączenie zawartości pliku .php wewnątrz innego pliku .php.
5. Poszukaj w internecie, co to jest CRUD, i opisz każdy z czterech elementów tego pojęcia.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zaprogramuj:
 - w języku PHP stronę doDolaczenia.php, która wyświetli napis: „ZOSTAŁAM DOŁĄCZONA DO STRONY HTML”;
 - w języku HTML stronę index.html, która wewnątrz elementu `<body>` `</body>` dołączy zawartość strony doDolaczenia.php.

Koniecznie sprawdź w przeglądarce efekt własnej pracy programistycznej. Zwróć uwagę na zagadnienie: Czy żeby powyższe zadanie działało poprawnie, to tego typu projekt musi być uruchomiony na serwerze internetowym (np. wewnątrz pakietu XAMPP)?

VI. Planowanie przedsięwzięć programistycznych

- Wybór technologii i środowiska programistycznego
- Przedstawienie alternatywnego rozwiązania
- Plan prac programistycznych
- Zasoby ludzkie
- Zarządzanie projektem

25

SOAW – wybór technologii i środowiska programistycznego

ZAGADNIENIA

- Wybór technologii
- Technologie PHP, JAVASCRIPT, HTML 5, CSS, SQL
- Technologia – sprzęt IT
- Technologia – narzędzia informatyczne
- Zasoby osobowe: technologia, deweloperzy i wybrane środowisko programistyczne

25.1. Wybór technologii

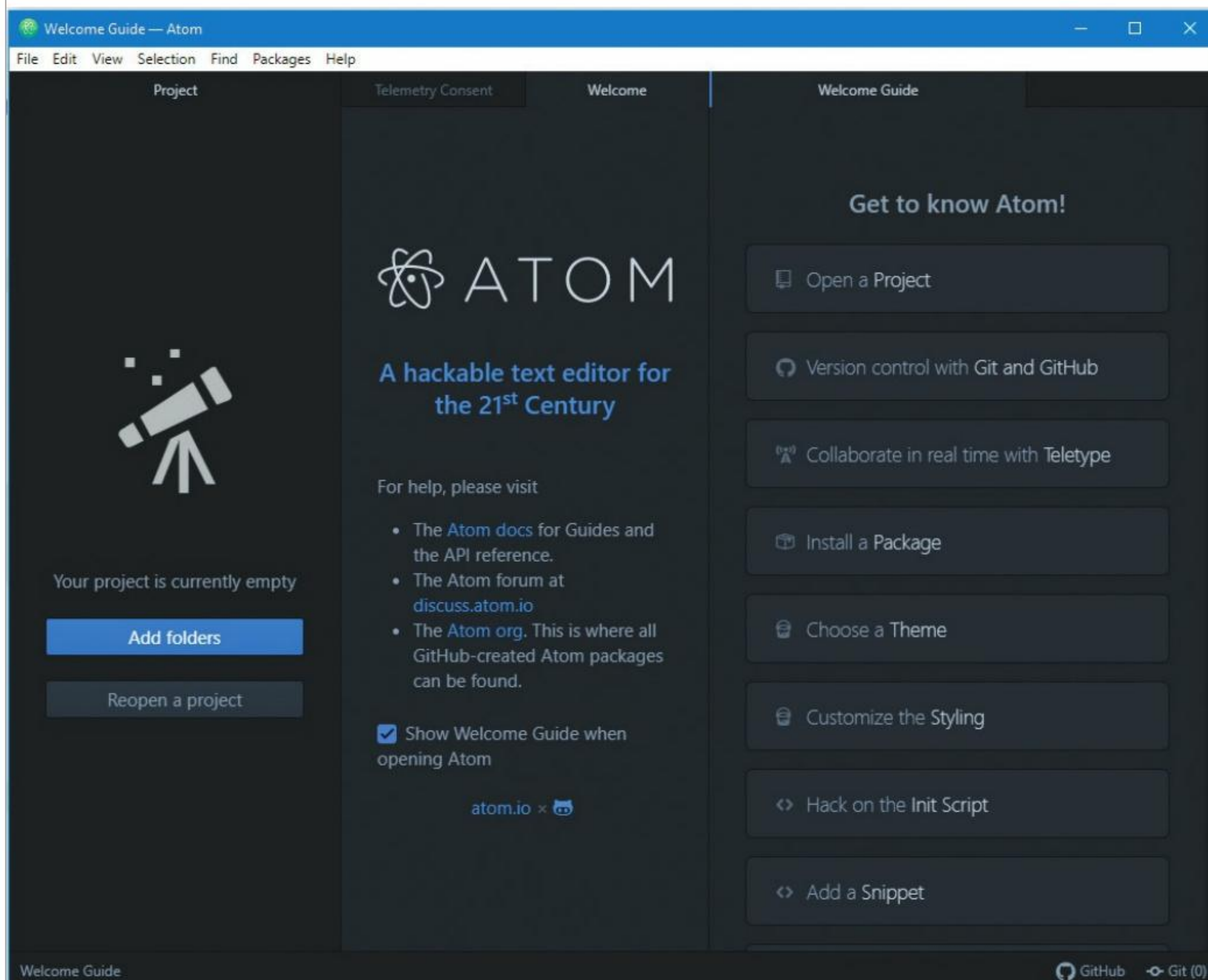
Realizacja każdego projektu jest zawsze związana z właściwym doбором różnorodnych składników, w tym:

- narzędzi pracy,
- pracowników.

Teoretycznie byłoby idealnie, gdyby najpierw przeprowadzić analizę możliwości i potrzeb, a na jej podstawie dobrać optymalne technologie informatyczne i dopiero na samym końcu zebrać zespół o określonych kwalifikacjach (wiedza i doświadczenie programistyczne z zakresu określonych obszarów IT).

O ile w przypadku wielkich firm programistycznych, zatrudniających setki lub tysiące programistów, można przypuszczać, że powyższe założenie jest spełnione, o tyle w przypadku małych firm działa to nieco inaczej. Właściciele niewielkich firm mają zwykle już zebrany zespół osób o określonych kompetencjach i na ten obszar nakierowują uwagę klienta. Ma to oczywiście zalety i wady. Niewątpliwie współpraca z małą firmą to większy komfort, ponieważ taka firma zawsze musi walczyć o każdego klienta. Proces decyzyjny jest w nich „bliżej” procesu tworzenia aplikacji, gdyż struktura zarządzania jest „płaska” (właściciel i tuż pod nim programiści).

W dużych firmach zaletą jest większy wybór możliwości, choćby technologicznych. Ze względu na rozbudowaną strukturę (dyrektor, kierownicy działów, kierownicy projektów i dopiero pod nimi programiści) zawsze trudniej dotrzeć do osób decyzyjnych, a sztywne procedury wielu klientom często utrudniają dość znacznie współpracę. Co więcej, ceny ich usług są dużo wyższe niż w przypadku mniejszych firm.

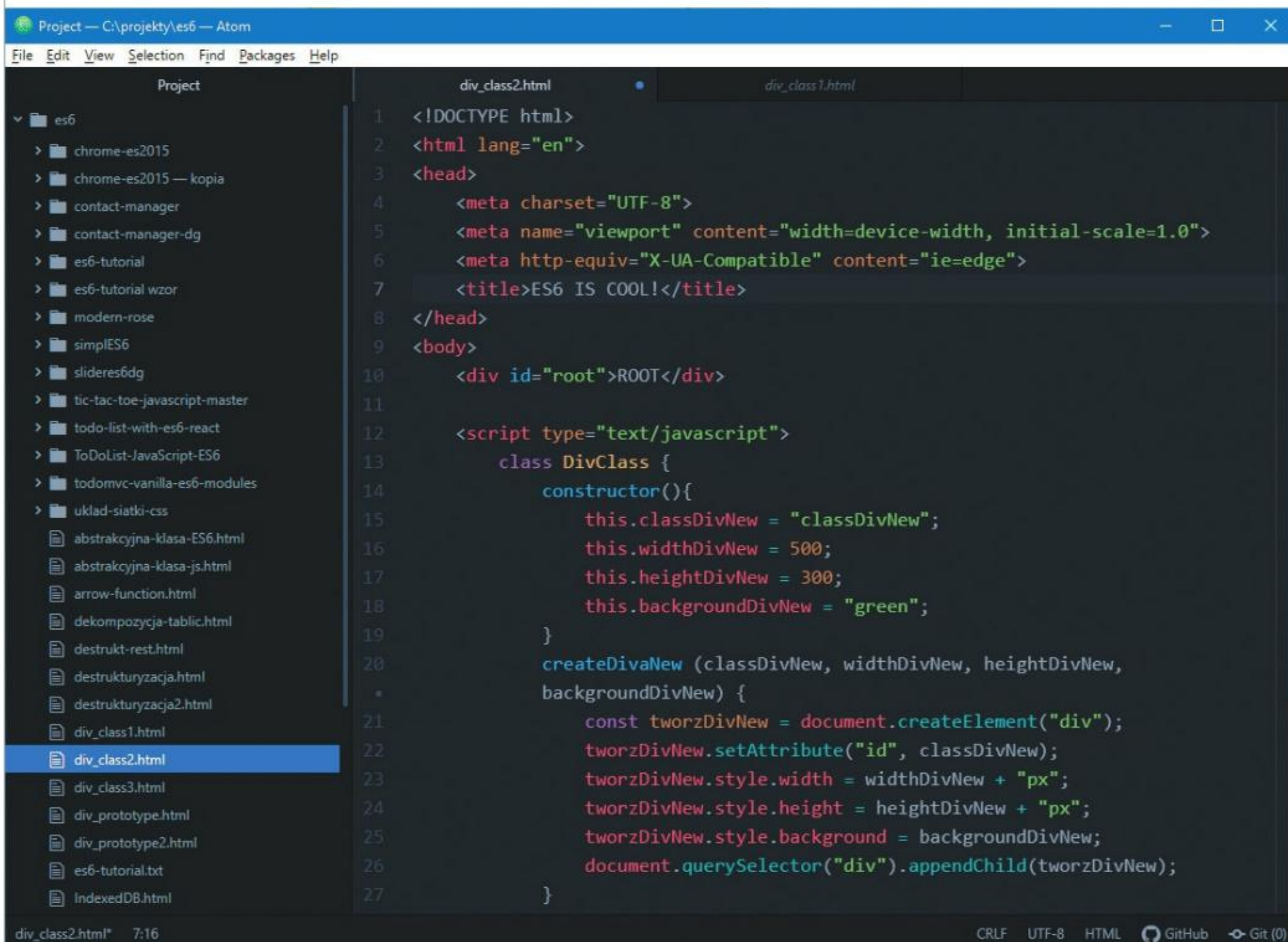


Rys. 25.1. Atom.js – widok tuż po instalacji

25.2. Technologie PHP, JAVASCRIPT, HTML 5, CSS, SQL

W przypadku realizacji SOAW wybraliśmy połączenie technologii: PHP, JavaScript, HTML 5, CSS, SQL, głównie ze względu na przewidywaną specyfikę projektu – niedużego, który (zapewne, choć nie ma oczywiście takiej pewności) nie będzie musiał w niedalekiej przyszłości obsługiwać bazy danych większej niż kilka milionów zapytań miesięcznie.

W kolejnej części tego podręcznika wskażemy kilka rozwiązań dla wybranych technologii i możliwości dodania np. frameworków CSS czy bibliotek JS, by usprawnić i przyspieszyć pracę zespołu deweloperów.



Rys. 25.2. Atom.js – widok roboczy pliku HTML

25.3. Technologia – sprzęt IT

W ramach realizowanego projektu ważne jest, aby wszyscy pracowali na takich samych narzędziach albo co najmniej bardzo podobnych, szczególnie od strony systemu operacyjnego i oprogramowania narzędziowego.

Zacznijmy od komputerów. Sprzęt powinien być adekwatny do wykonywanej pracy. Trzeba sprawdzić minimalne parametry programów (nazwy poznamy już za chwilę), na których będą pracowali programiści, a potem pomnożyć co najmniej razy 1,5, a najlepiej razy 2 lub nawet więcej. Co ważne, w każdym z komputerów powinien być dysk co najmniej SSD (HDD „strasznie spowalnia” podczas programowania). Powinniśmy otrzymać wydajny sprzęt, dzięki któremu będzie można szybko i sprawnie:

- postawić serwer wraz z niezbędnym oprogramowaniem,
- pisać kody programistyczne,
- efektywnie testować wyniki pracy.

Środowiska, w których pracujemy, można sprawdzać (testować) na różne sposoby. Przykładowo: po uruchomieniu serwera zawartego w XAMPP możesz sprawdzić posiadaną konfigurację pod adresem: <http://localhost/dashboard/phpinfo.php>. Podobne sposoby są dostępne w różnych programach (także w systemach operacyjnych).

Configuration**apache2handler**

Apache Version	Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.8
Apache API Version	20120211
Server Administrator	postmaster@localhost
Hostname:Port	localhost:80
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	No
Server Root	C:/xampp/apache
Loaded Modules	core mod_win32 mpm_winnt http_core mod_so mod_access_compat mod_actions mod_alias mod_allowmethods mod_asis mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_groupfile mod_authz_host mod_authz_user mod_autoindex mod_cgi mod_dav_lock mod_dir mod_env mod_headers mod_include mod_info mod_isapi mod_log_config mod_cache_disk mod_mime mod_negotiation mod_proxy mod_proxy_ajp mod_rewrite mod_setenvif mod_socache_shmcb mod_ssl mod_status mod_php7

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0

Apache Environment

Variable	Value
MIBDIRS	/xampp/php/extras/mibs
MYSQL_HOME	\\xampp\\mysql\\bin
OPENSSL_CONF	/xampp/apache/bin/openssl.cnf
PHP_PEAR_SYSCONF_DIR	\\xampp\\php
PHPRC	\\xampp\\php
TMP	\\xampp\\tmp
HTTP_HOST	localhost
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	pl,en-US;q=0.7,en;q=0.3
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_DNT	1
HTTP_CONNECTION	keep-alive
HTTP_REFERER	http://localhost/dashboard/
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_SEC_GPC	1

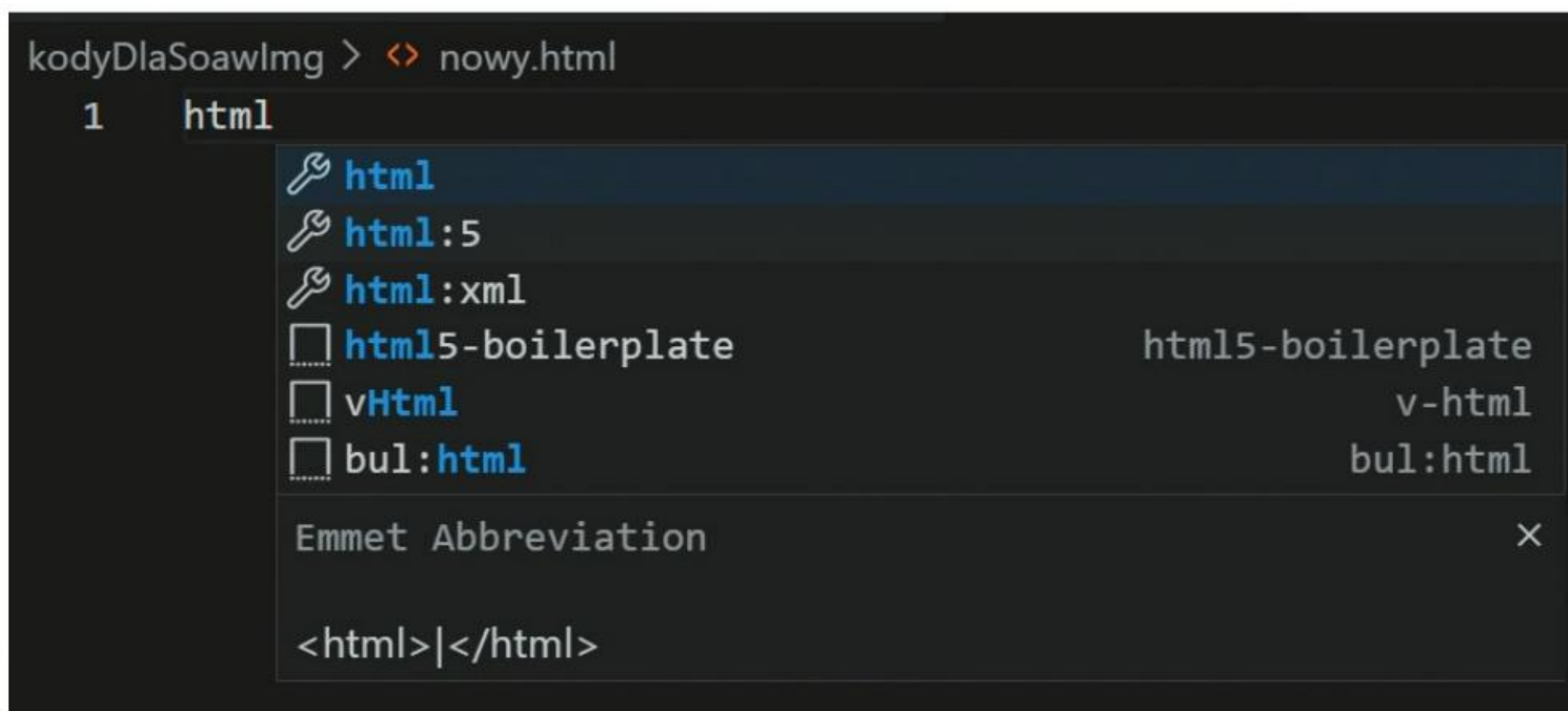
Rys. 25.3. Konfiguracja środowiska XAMPP – Apache

Źródło: <http://localhost/dashboard/phpinfo.php> [dostęp: 11.02.2020].

Czas w branży IT jest bardzo kosztowny, ponieważ programiści otrzymują wynagrodzenie zwykle za godzinę pracy. Oznacza to, że kwoty wydane na szybki i wydajny sprzęt IT dość prędko się zwróca. Spowalniający pracę i/lub podatny na awarię komputer będzie opóźniał realizację projektu i wywoływał niepotrzebną frustrację zarówno u deweloperów, którzy lubią wydajnie pracować, u szefa realizowanego projektu, jak i u klienta.

25.4. Technologia – narzędzia informatyczne i zasoby osobowe

Wydajne, znane wszystkim standardy w zakresie tworzenia oprogramowania wymagają zastosowania narzędzi informatycznych, które efektywnie wspomogą realizację projektu. Oczywiście – teoretycznie – możemy zmusić programistów do pisania kodów nawet w zwykłym notatniku. Pamiętajmy, że za czas ich pracy i tak później trzeba będzie zapłacić, więc okaże się to zupełnie nieopłacalne.

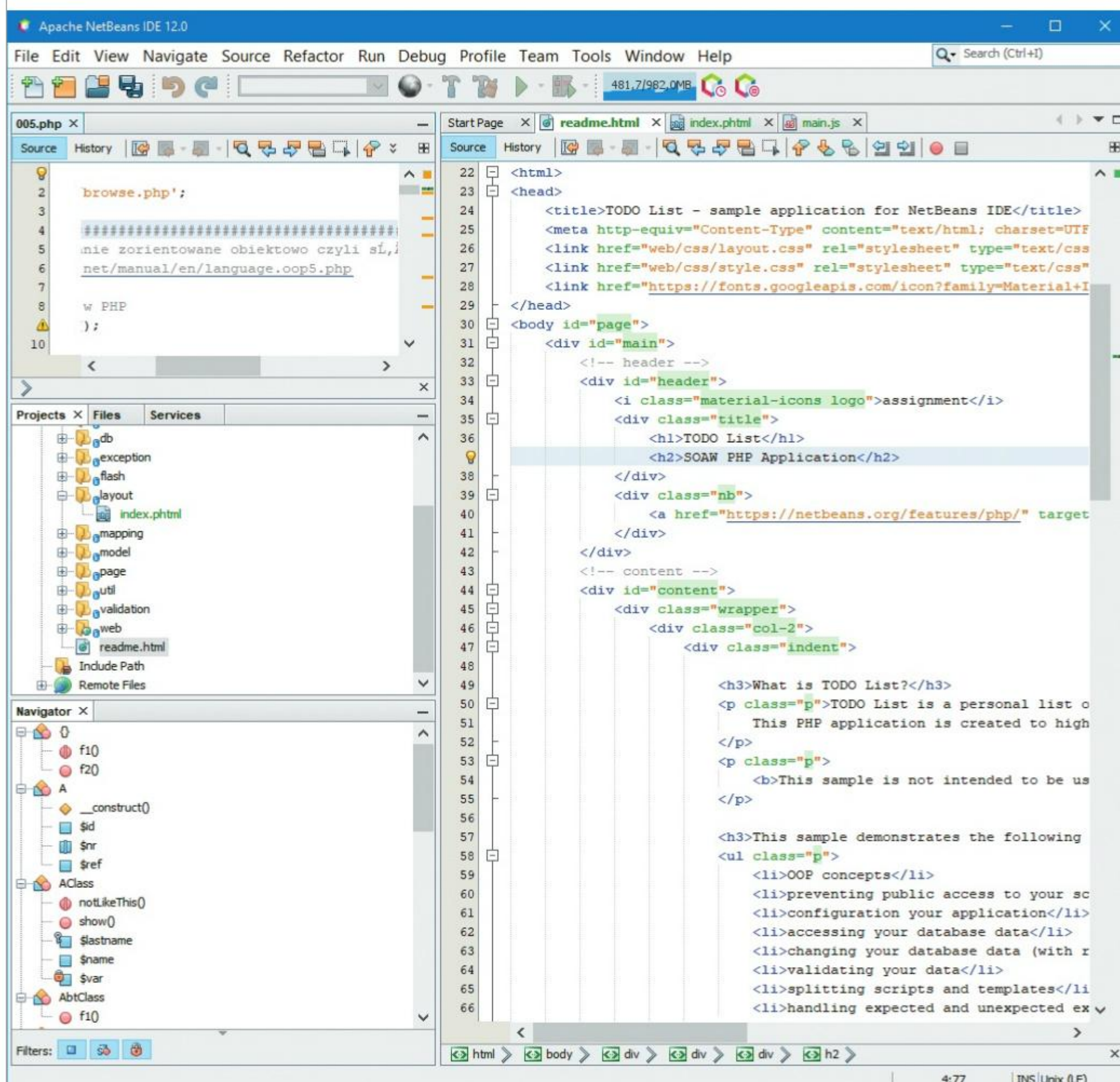


Rys. 25.4. Visual Studio Code – podpowiedź w trakcie pisania kodu (wystarczy wybrać jedną z linii podpowiedzi, a kod całej struktury HTML zostanie utworzony automatycznie przez VSC)

Dokładnie tak samo jest z wykorzystaniem odpowiednich narzędzi programistycznych. W przypadku programowania „narzędziem” przyspieszającym pracę jest właściwe oprogramowanie, tzw. IDE (ang. *Integrated Development Environment*), czyli zintegrowane środowisko programistyczne. Jest to program lub pakiet programów (środowisko) służących do tworzenia, modyfikowania, testowania i utrzymywania oprogramowania.

Tak zwanych IDE jest na rynku wiele, zarówno tych płatnych, jak i całkowicie bezpłatnych. Najlepsze z nich i – co równie ważne – bezpłatne to:

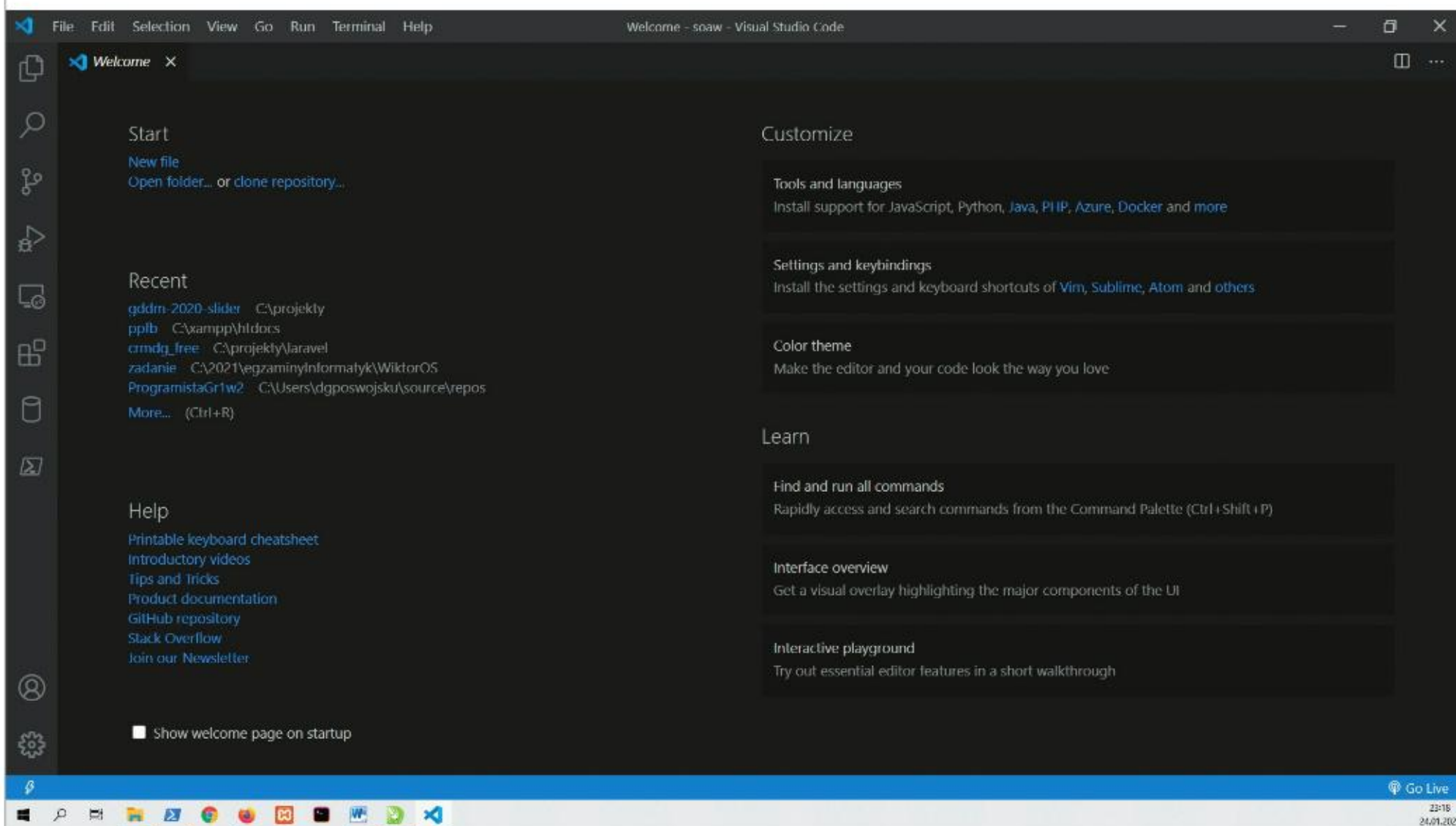
- Visual Studio Code – szczegóły projektu: <https://code.visualstudio.com/>,
- Atom – szczegóły projektu: <https://atom.io/>,
- Visual Studio Community 2019 – szczegóły projektu: <https://visualstudio.microsoft.com/pl/>,
- Brackets – szczegóły projektu: <http://brackets.io/>,
- NetBeans – szczegóły projektu: <https://netbeans.org/>,
- Eclipse – szczegóły projektu: <https://www.eclipse.org/>.



Rys. 25.5. NetBeans – widok roboczy projektu

Pamiętaj, że narzędzia, z których skorzystaliśmy, nie są obowiązkowe. Oczywiście nie musisz korzystać z podpowiedzi i użyć zupełnie innego rozwiązania. Najważniejsze, żebyś się swobodnie czuł podczas programowania w danym IDE. Zapewne zauważyłeś, że zasugerowano programy działające w systemie operacyjnym Windows, gdyż najczęściej z takiego systemu korzystamy (pamiętajmy, że Linux jest równie cenny). Jeżeli korzystasz z Linuxa bądź innych systemów operacyjnych, masz dostęp do programów charakterystycznych dla danego systemu. Istnieją także aplikacje ponadsystemowe – działające np. na Linuxa i Windows (dla przykładu Visual Studio Code). Powoduje to, że podczas zmiany systemu na inny można w dalszym ciągu korzystać z tego samego zestawu programów, dzięki czemu nie trzeba przyzwyczajać się do nowych programów bądź ustawień. Między innymi z powodu wspomnianej ponadsystemowości wybraliśmy zestaw, który znajdziesz na następnej stronie.

Zwykle wykorzystujemy wskazany zestaw narzędzi programistycznych, działających zarówno na systemach Linux, jak i Windows.

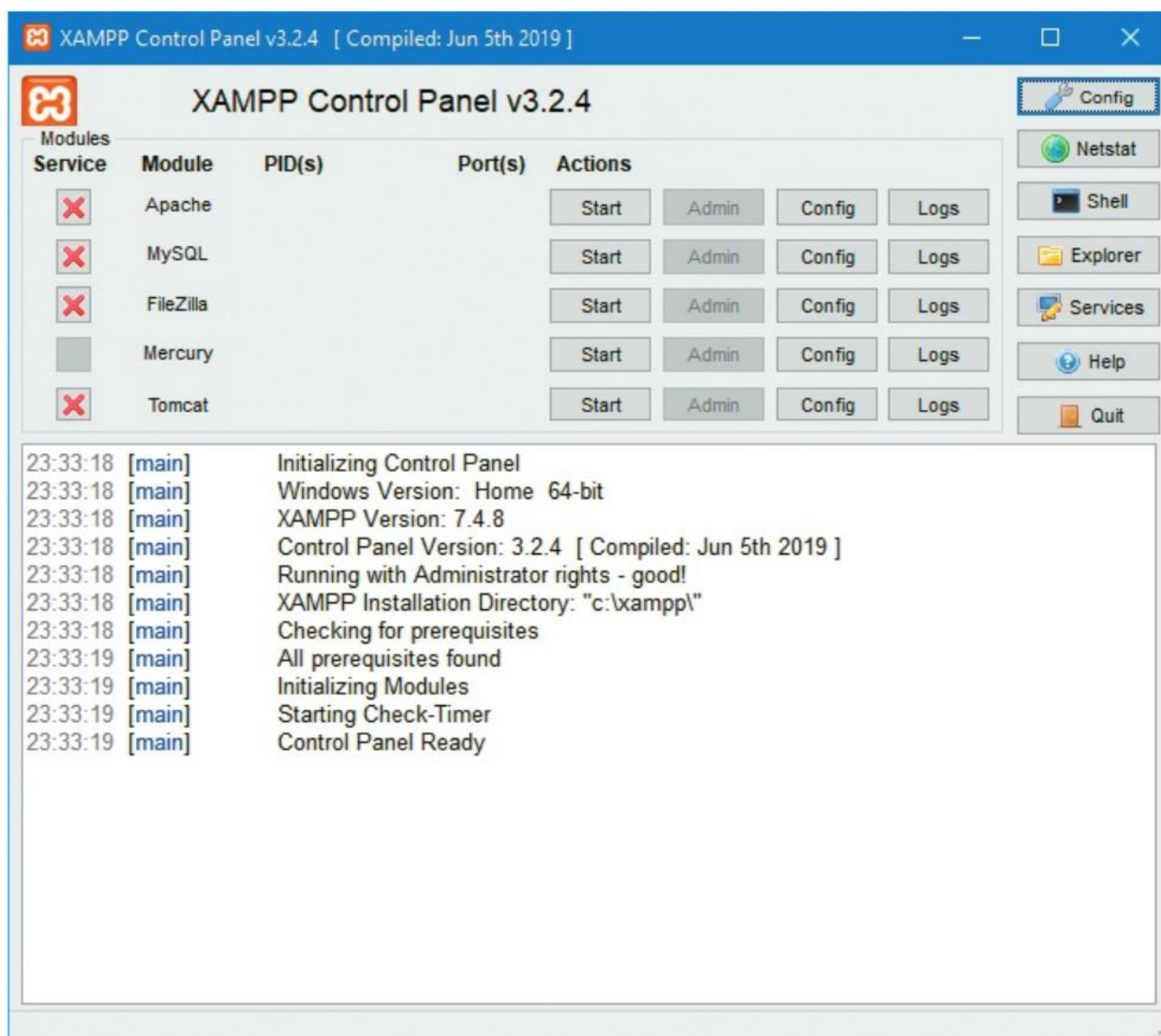


Rys. 25.6. Visual Studio Code – widok Welcome

Do programu, w którym programiści będą pisali kod, trzeba dobrać zestaw pozostałych narzędzi. Ponieważ nasz projekt będzie bardzo prostą aplikacją, zestaw dodatkowego oprogramowania możemy ograniczyć do niezbędnego minimum.

Podsumujmy założenia narzędziowe i osobowe (pracownicy). W naszym przypadku zakładamy, że program będzie pisany przez maksymalnie dwóch programistów:

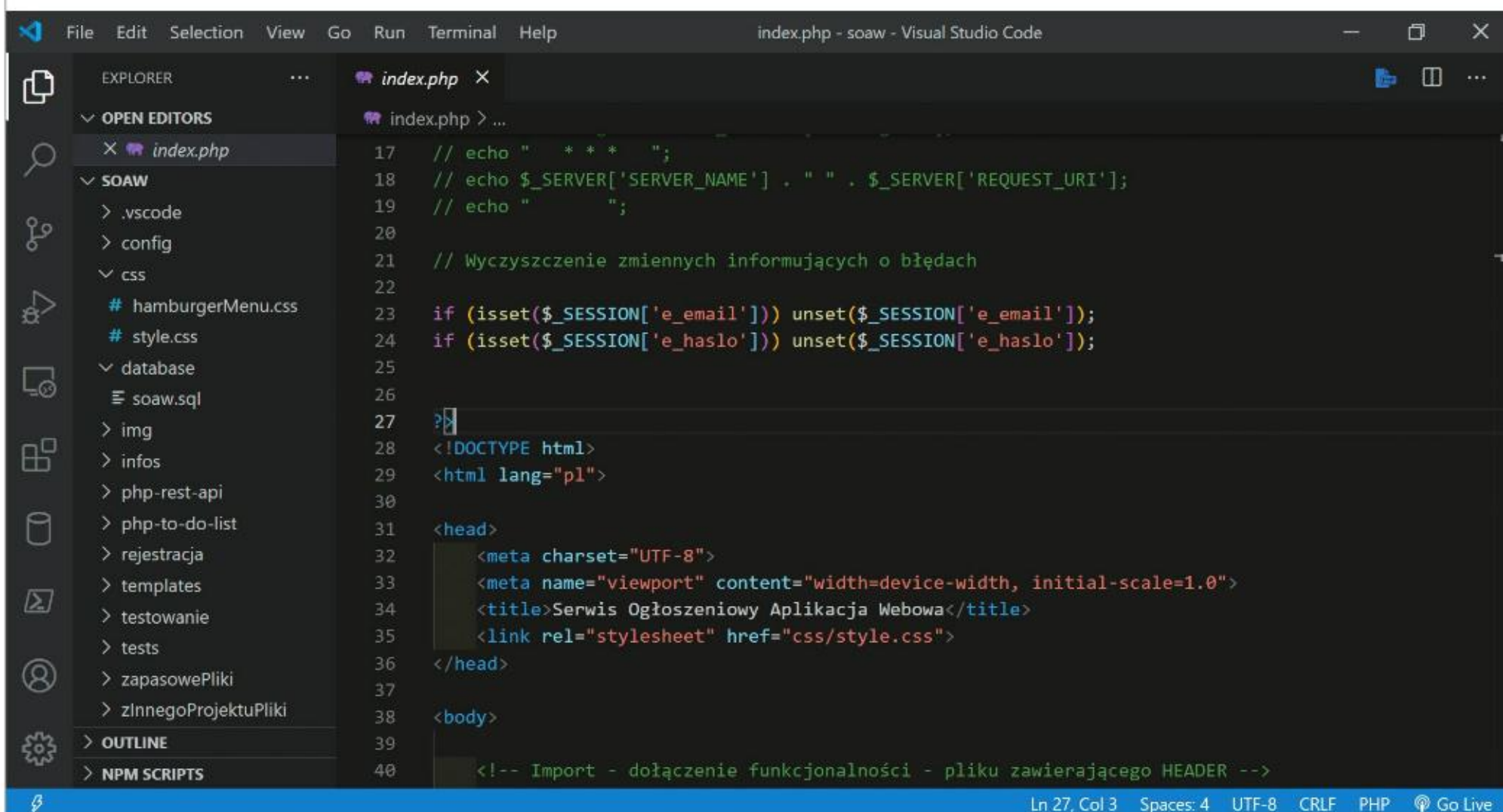
- jeden zajmie się backendem, a więc musi znać technologie:
 - podstawy wiedzy o serwerach – u nas Apache,
 - relacyjna baza danych SQL – u nas MariaDB,
 - język zapytań do bazy danych SQL,
 - PHP – język skryptowy łączący backend z frontendem;
- drugi zajmie się frontendem, musi znać co najmniej:
 - HTML 5 – celem zbudowania struktury aplikacji,
 - CSS – język stylów, dzięki któremu będzie stworzony określony wygląd aplikacji,
 - JavaScript i/lub jQuery – potrzeba znajomości zasad wykonywania interakcji na stronie WWW jest niezbędna.



Rys. 25.7. Pakiet XAMPP Control Panel – widok startowy przed uruchomieniem aplikacji

Dobierzmy narzędzia pracy – oprogramowanie dla osoby odpowiedzialnej za:

- backend:
 - XAMPP – zawierający zestaw oprogramowania backendowego, m.in.: serwer Apache, bazę danych MariaDB, język skryptowy PHP i phpMyAdmin, czyli narzędzie ułatwiające tworzenie, usuwanie i aktualizację poszczególnych baz danych i zarządzanie nimi,
 - Visual Studio Code – program, w którym będą tworzone kody PHP i języka SQL,
 - przeglądarki internetowe – do bieżącego testowania tworzonej aplikacji, najlepiej co najmniej dwie najbardziej popularne, tzn. FireFox i Mozilla, oczywiście może być więcej, aby obsłużyć większą liczbę potencjalnych klientów (np. jeszcze Opera, Safari, Edge),



Rys. 25.8. Visual Studio Code – widok roboczego kodu PHP w projekcie SOAW

- frontend:
 - Visual Studio Code – program, w którym będą tworzone kody HTML 5, CSS i JavaScript,
 - przeglądarki internetowe – do bieżącego testowania tworzonej aplikacji, najlepiej co najmniej dwie najbardziej popularne, tzn. FireFox i Mozilla, oczywiście może być więcej, aby obsłużyć większą liczbę potencjalnych klientów (np. jeszcze: Opera, Safari, Edge).



Rys. 25.9. Przeglądarka internetowa FireFox fundacji Mozilla

 SPRAWDŹ SWOJĄ WIEDZĘ

1. Wskaż podstawowe różnice w „cyklu życia projektu” między tradycyjnymi produktami a produktami nowoczesnych technologii (webowych).
2. Wyjaśnij, dlaczego wybraliśmy technologie: PHP, JavaScript, HTML 5, CSS, SQL do realizacji SOAW.
3. Co to jest Visual Studio Code? Wymień rozwiązania alternatywne.
4. Do czego służy XAMPP?
5. Do czego służy FireFox? Wymień rozwiązania alternatywne.
6. Opisz, w jaki sposób na pracę programisty wpływa bardzo wydajny (szybki) zestaw komputerowy.

26

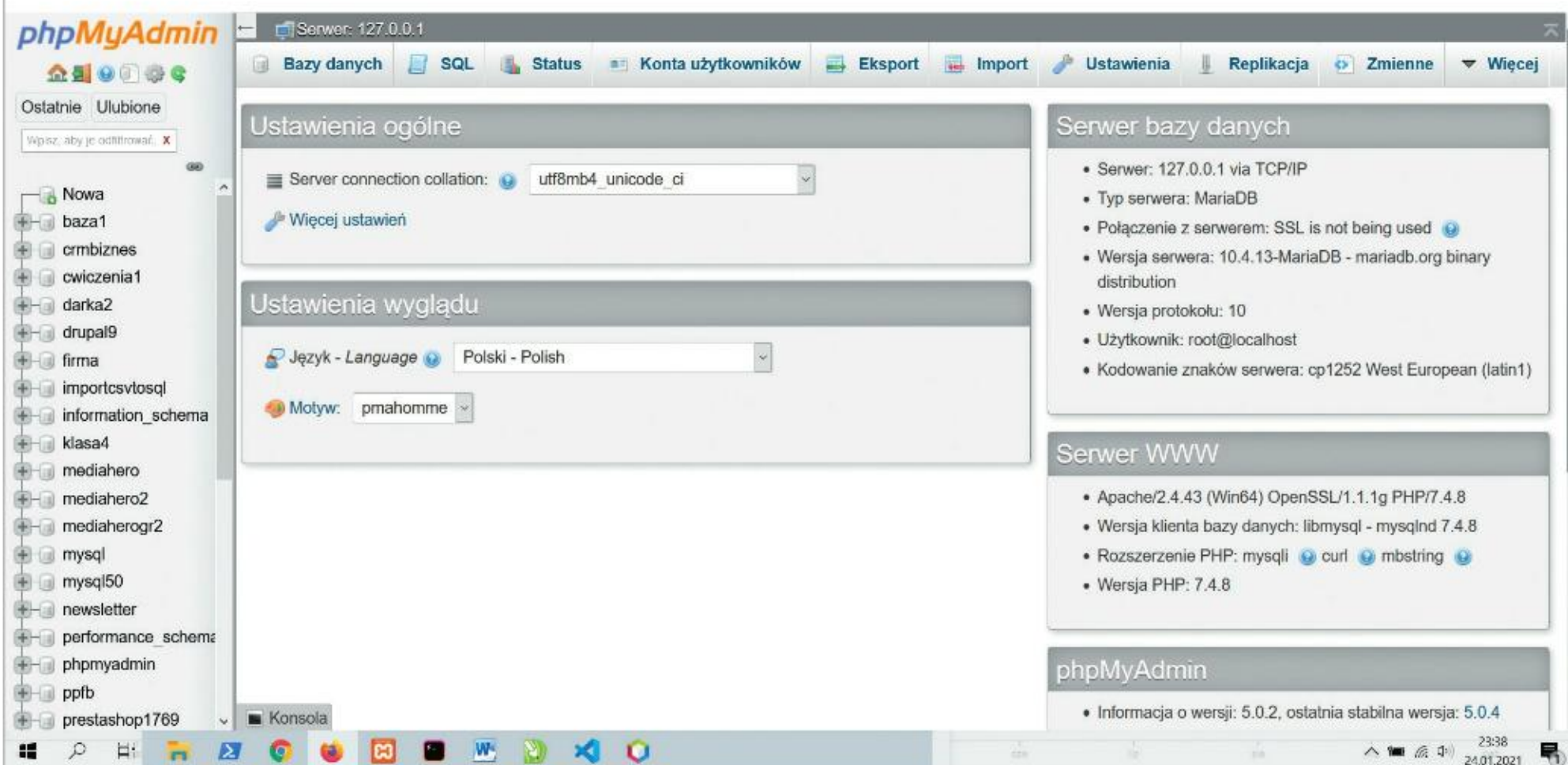
SOAW: przedstawienie alternatywnego rozwiązania

ZAGADNIENIA

- Dodatkowe technologie mogące uzupełnić zestaw PHP, JavaScript, HTML 5, CSS, SQL
- Alternatywne rozwiązanie dla wyglądu SOAW – biblioteki (frameworki) CSS
- Alternatywne rozwiązanie dla budowy SOAW – node.js, React, Angular

26.1. Dodatkowe technologie mogące uzupełnić zestaw PHP, JavaScript, HTML 5, CSS, SQL

Podczas tworzenia aplikacji, czyli kodowania w językach programowania bez użycia frameworków, mówimy, że programujemy np. w vanilla JavaScript, vanilla PHP czy vanilla Python. W języku polskim odpowiednikiem terminu vanilla jest ‘czysty’, np. czysta JavaScript, czyste PHP itd. Chodzi tutaj o korzystanie ze specyfikacji danego języka bez jakichkolwiek domieszek. Alternatywą dla programowania vanilla jest używanie frameworków.



Rys. 26.1. phpMyAdmin – narzędzie ułatwiające tworzenie, usuwanie, aktualizację bazy danych i zarządzanie poszczególnymi bazami danych

Framework (programistyczny) to bardzo pożyteczny wynalazek, dzięki któremu projekty są tworzone zdecydowanie szybciej niż w „czystych” językach programowania.

Framework albo platforma programistyczna to szkielet służący do budowy aplikacji, który:

- definiuje strukturę aplikacji;
- definiuje ogólny mechanizm działania projektu;
- dostarcza zestaw komponentów i/lub bibliotek ogólnego przeznaczenia, służących do wykonywania określonych zadań.

Używając tego rozwiązania, trzeba pisać kody zgodnie z narzuconymi standardami (często także zgodnie z określonym szkieletem aplikacji), charakterystycznymi dla danego frameworka. Warto poznać dobry framework PHP, a jest ich całkiem sporo. Rozpoczynanie nauki programowania od frameworka to jednak zły pomysł. Uczysz się wówczas tylko standardu pewnej grupy osób, dość często znacznie odbiegającego od „czystego” języka programowania. Otrzymujesz z góry ustalone zasady, obiekty, funkcje itp., dzięki czemu frameworki umożliwiają szybsze tworzenie projektów. Z drugiej strony jesteś równocześnie ograniczony zasadami tworzenia danego frameworka, przez co kreatywność i możliwości kreatywnego zachowania programisty są zwykle dużo mniej elastyczne. Następuje także pewne uwstecznienie w wyniku używania frameworka – programiści w dużej mierze zapominają „czystą” strukturę danego języka programowania.

Aby lepiej zrozumieć pojęcie framework, spójrz na rys. 26.2.

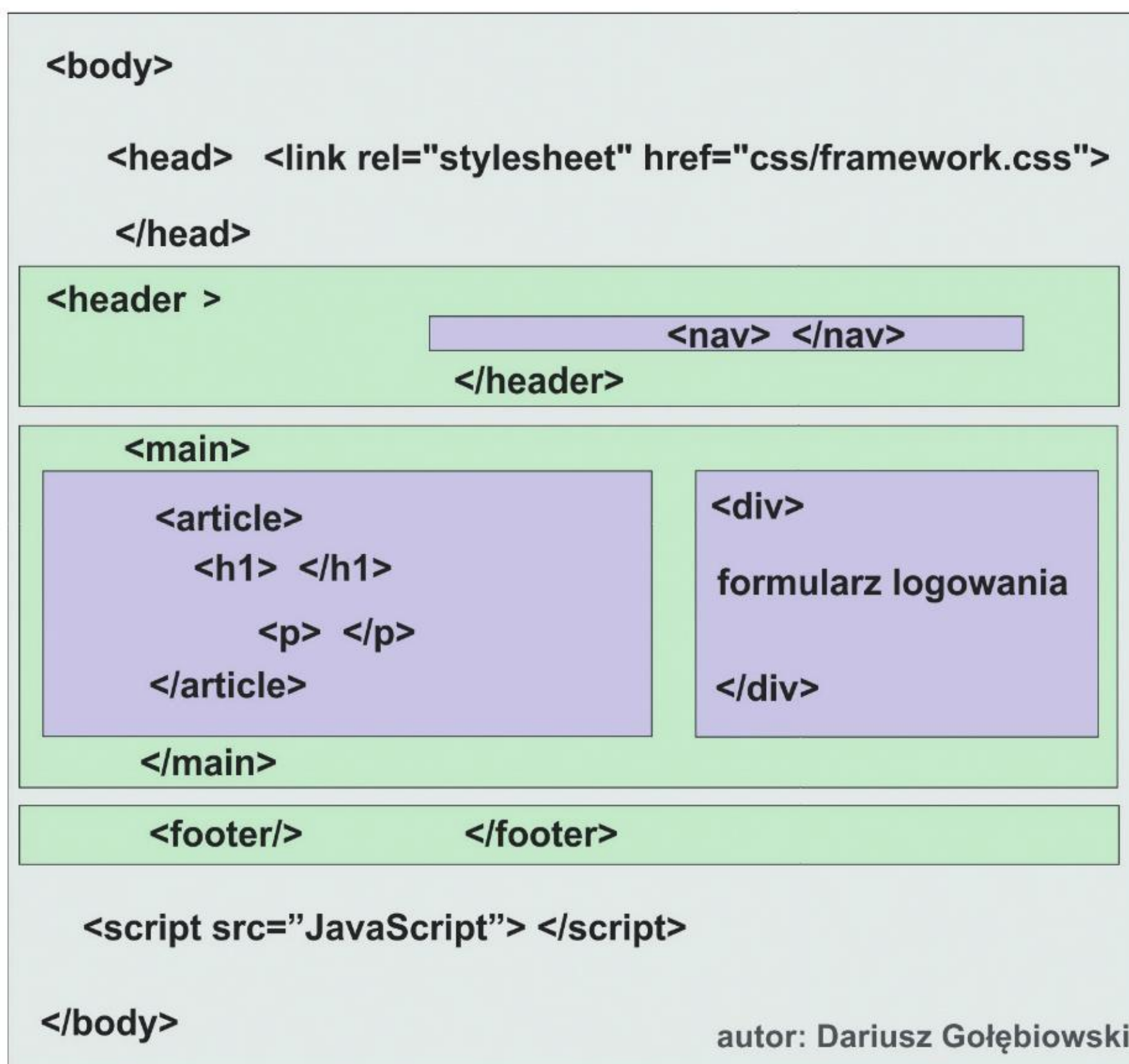
W czym może pomóc framework? Skupmy się na elementach ogólnych i najważniejszych. Przede wszystkim framework dobrze dobrany do specyfiki danego projektu może przyspieszyć pracę programisty (a szczególnie dużego zespołu deweloperskiego), m.in. przez:

- automatyczne stworzenie szkieletu aplikacji, zawierającego elementy typu system logowania (np. Symfony, Laravel) czy struktury dostępu do bazy danych;
- uzyskanie podstawowego wyglądu strony, a nawet elementów interakcyjnych (Bulma, Bootstrap – przyciski, formularze, slidery itp.);
- dostęp do gotowych elementów aplikacji – np. niektóre frameworki JavaScript (Angular);
- mniejszą ilość kodu niezbędną do zaprogramowania danej funkcjonalności – oszczędność czasu pracy programisty (React, Svelte, jQuery itp.).

Wiemy już, czym jest programowanie vanilla, znamy pojęcie frameworka i korzyści z jego zastosowania. Możemy się w takim razie zastanowić, przy założeniu, że jednak chcielibyśmy w planowanym projekcie SOAW skorzystać z frameworka, który będzie dla nas szczególnie przydatny.

FRAMEWORK

szkielet podstawowego wyglądu
oraz funkcjonalności



Rys. 26.2. Framework – ustawienia początkowe

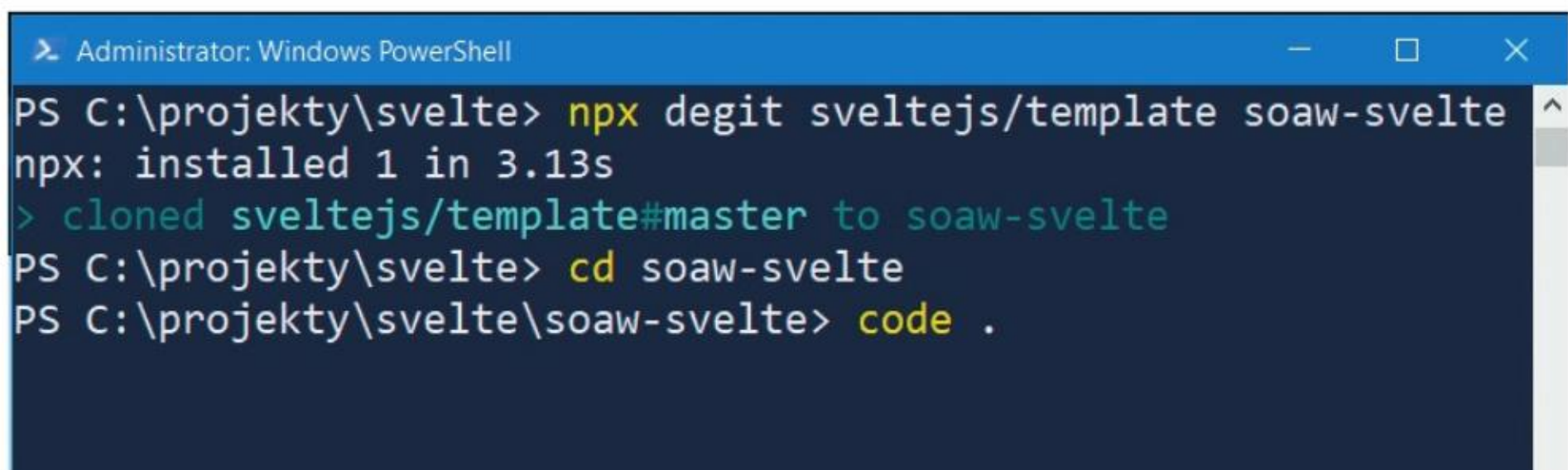


Rys. 26.3. Popularne frameworki PHP

Zacznijmy od wymienienia kilku ciekawych i bardzo dobrych (nowoczesnych) rozwiązań opartych na PHP. Poniżej odnośniki do ich stron internetowych i zawarte tam informacje:

- Laravel – <https://laravel.com/>
„PHP framework dla rzemieślników (hmm, a może artystów?) internetowych. Laravel to struktura aplikacji internetowych z wyrazistą, elegancką składnią. Położyliśmy już podwaliny i uwolniliśmy cię od kłopotów z tworzeniem drobiazgów”.
- Symfony – <https://symfony.com/>
„Symfony to zestaw komponentów PHP wielokrotnego użytku i ramy PHP dla projektów internetowych. Standardowy fundament, na którym są zbudowane najlepsze aplikacje PHP”.
- Yii – <https://www.yiiframework.com/>
„Yii to szybki, bezpieczny i wydajny szkielet PHP. Elastyczne, ale pragmatyczne. Działa od razu po wyjęciu z pudełka. Ma określone ustawienia domyślne”.

Pierwsze dwa wymienione frameworki PHP to zdecydowanie najbardziej popularne rozwiązania tego typu. Jeżeli sprawdzisz w internecie, z pewnością znajdziesz co najmniej kilkanaście solidnych rozwiązań, jakościowo zapewne zbliżonych do tych wymienionych, a może nawet lepszych, tylko jeszcze po prostu niedocenionych przez społeczność programistów.



```
Administrator: Windows PowerShell
PS C:\projekty\svelte> npx degit sveltejs/template soaw-svelte
npx: installed 1 in 3.13s
> cloned sveltejs/template#master to soaw-svelte
PS C:\projekty\svelte> cd soaw-svelte
PS C:\projekty\svelte\soaw-svelte> code .
```

Rys. 26.4. Instalacja frameworka JavaScript Svelte w systemie Windows

CIEKAWOSTKA

Svelte (instalację programu przedstawiono na rys. 26.4 – strona domowa projektu: <https://svelte.dev/>) to dość nowe rozwiązanie w świecie technologii internetowych. Konkuruje z potentatami rynku, jak np. React.js czy Vue.js. Dzięki innej filozofii budowy i działania w szybkim czasie stało się ciekawą alternatywą do budowy interfejsów użytkownika w projektach internetowych. Omawiane rozwiązanie nie oferuje typowej struktury frameworka, tylko ma własny kompilator do przetwarzania kodu aplikacji po stronie użytkownika podczas jej budowania. Dzięki niewielkim wymiarom i dobrej optymalizacji oferuje dużą szybkość działania projektów, co zapewne jest

jednym z filarów odnoszonego sukcesu w zakresie popularności wśród deweloperów języka JavaScript. Drugim jest mniejszy poziom skomplikowania kodów niezbędnych do uzyskania pożądanego efektu podczas tworzenia aplikacji. Można znaleźć wiele podobnych opinii zawodowych programistów. Pamiętaj jednak, że dużo zależy także od indywidualnych predyspozycji danego dewelopera.

26.2. Alternatywne rozwiązanie dla wyglądu SOAW – biblioteki (frameworki) CSS

W przypadku frameworków CSS mamy bogactwo urodzaju: od rozwiązań prostych i bardzo efektywnych typu Bulma aż do zaawansowanych, dających potężne możliwości budowania wspaniałych wyglądów aplikacji – typu Bootstrap. Poniżej odnośniki do ich stron internetowych i zawarte tam informacje:

- Bulma – <https://bulma.io/>
„Bulma to darmowa, otwarta struktura CSS oparta na Flexbox i używana przez ponad 200 000 programistów”.
- Bootstrap – <https://getbootstrap.com/>
„Twórz szybkie, responsywne witryny dzięki Bootstrap. Szybko projektuj i dostosowuj responsywne strony mobilne za pomocą Bootstrap, najpopularniejszego na świecie zestawu narzędzi typu frontend, zawierającego zmienne i miksery Sass, responsywny system grid, rozbudowane, wstępnie utworzone komponenty i potężne wtyczki JavaScript”.
- Semantic UI – <https://semantic-ui.com/>
„Szybko projektuj piękne strony internetowe. Semantic to platforma programistyczna, która pomaga tworzyć piękne, responsywne układy przy użyciu przyjaznego dla człowieka kodu HTML”.



Rys. 26.5. Popularne frameworki CSS

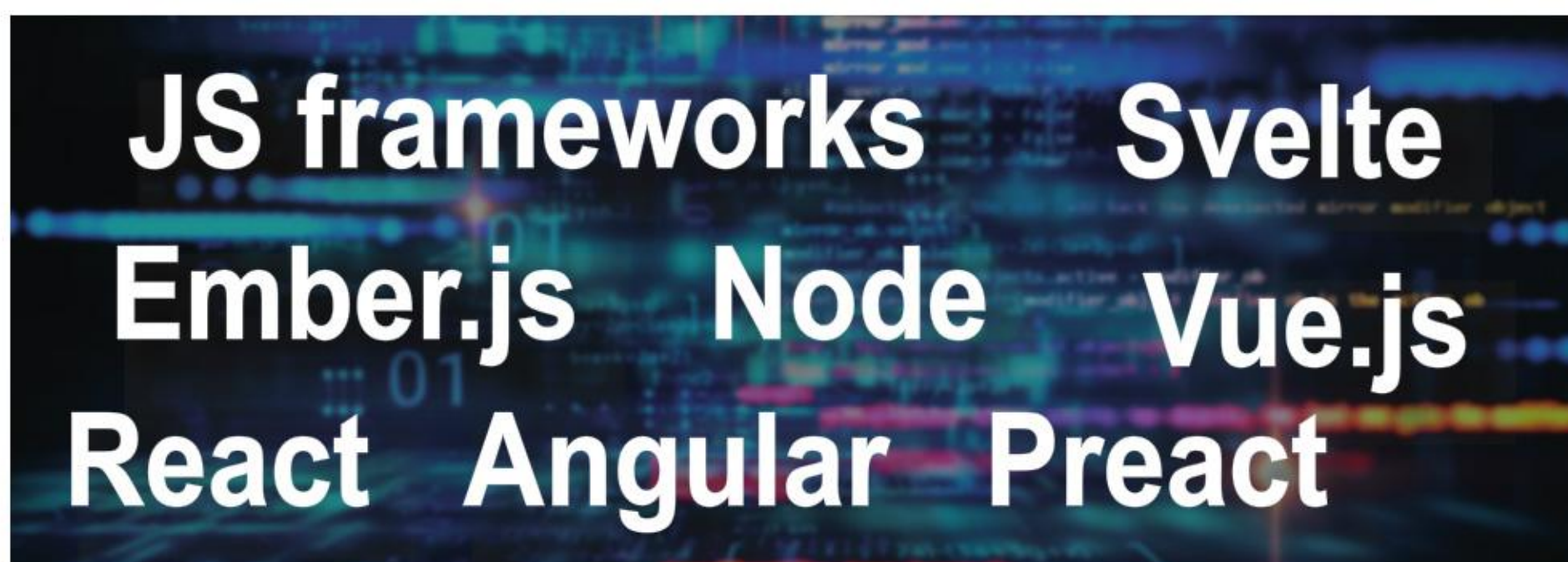

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport"
    content="width=device-width, initial-scale=1">
  <title>Strona wykonana z CSS Bulma!</title>
  <link rel="stylesheet" href="https://cdn.jsdelivrivr.
    net/npm/bulma@0.9.1/css/bulma.min.css">
</head>
<body>
  <section class="section">
    <div class="container">
      <h1>Witaj na stronie z CSS Bulma!</h1>
    </div>
  </section>
</body>
</html>
```

Rys. 26.6. Dołączenie frameworka CSS Bulma do strony HTML i kod CSS – przykładowe kontenery Bulma

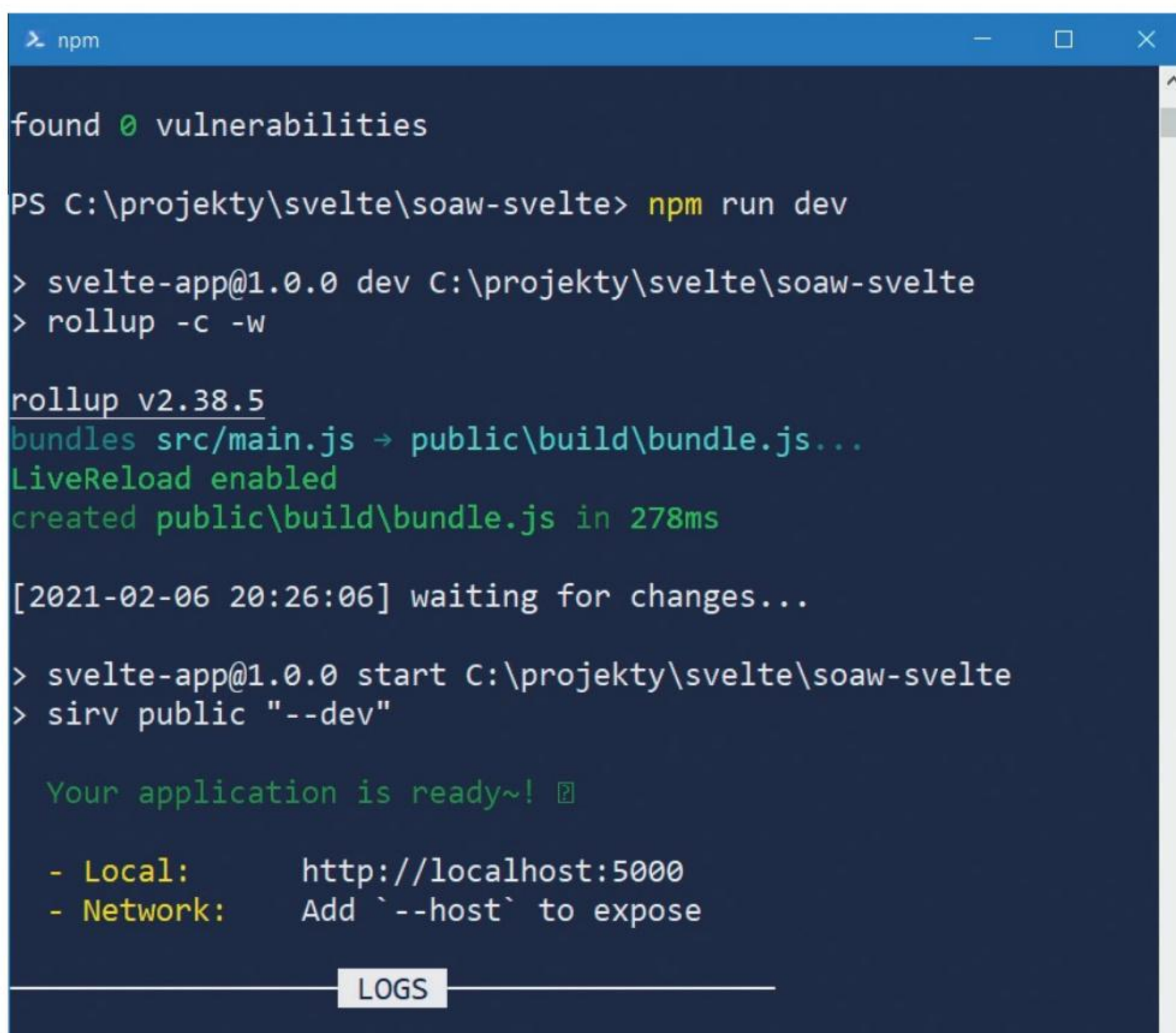
Szczególnie potężnym narzędziem i równocześnie jednym z najczęściej wybieranych przez programistów jest Bootstrap. Niestety, ma on co najmniej dwie wady – jest bardzo „ciężki” i skomplikowany (zwłaszcza dla początkujących twórców stron i aplikacji internetowych). Z tego powodu dla małych projektów praktycznie się nie nadaje. Lepszy jest „czysty CSS”, ewentualnie „lekka” Bulma. W przypadku dużego projektu, do którego zaangażowano wielu programistów, zastosowanie Bootstrap jest zapewne pomysłem godnym rozważenia.

26.3. Alternatywne rozwiązanie dla budowy SOAW – node.js, React.js, Angular, Vue.js

W poprzednich rozdziałach pokazaliśmy, jak można przyspieszyć realizację projektu, korzystając z półgotowych rozwiązań, jakimi są frameworki PHP i CSS. Podobnie jest w przypadku JavaScript. Co więcej, możemy z nią osiągnąć nawet lepsze wyniki.



Rys. 26.7. Popularne frameworki JavaScript



```
> npm  
found 0 vulnerabilities  
PS C:\projekty\svelte\soaw-svelte> npm run dev  
> svelte-app@1.0.0 dev C:\projekty\svelte\soaw-svelte  
> rollup -c -w  
  
rollup v2.38.5  
bundles src/main.js → public\build\bundle.js...  
LiveReload enabled  
created public\build\bundle.js in 278ms  
  
[2021-02-06 20:26:06] waiting for changes...  
  
> svelte-app@1.0.0 start C:\projekty\svelte\soaw-svelte  
> sirv public "--dev"  
  
Your application is ready~! 🎉  
  
- Local:      http://localhost:5000  
- Network:    Add `--host` to expose  
  
LOGS
```

Rys. 26.8. Framework Svelte (konsola NPM) – kod uruchomieniowy serwera deweloperskiego

JavaScript dostarcza programistom potężne możliwości. Możemy za jego pomocą:

- przyspieszyć pisanie aplikacji, używając frameworków typu Angular, React.js, Svelte czy Vue.js (podobny efekt dałoby zastosowanie jQuery, która to zwykle jest określana jako biblioteka JavaScript, a nie framework);

- zupełnie zmienić technologię i zastosować język JavaScript po stronie:
 - backend – w postaci serwera node.js i programowania w ECMAScript (czyli jest to tzw. nowoczesny JavaScript),
 - frontend – wykonać cały ten element w jednym z frameworków JavaScript, np. React.js, Vue.js, Angular.js (jest ich dużo więcej).

```

{} package.json ×
{} package.json > ...
4   "scripts": {
5       "build": "rollup -c",
6       "dev": "rollup -c -w",
7       "start": "sirv public"
8   },
9   "devDependencies": {
10      "@rollup/plugin-commonjs": "^17.0.0",
11      "@rollup/plugin-node-resolve": "^11.0.0",
12      "rollup": "^2.3.4",
13      "rollup-plugin-css-only": "^3.1.0",
14      "rollup-plugin-livereload": "^2.0.0",
15      "rollup-plugin-svelte": "^7.0.0",
16      "rollup-plugin-terser": "^7.0.0",
17      "svelte": "^3.0.0"
18  },
19  "dependencies": {
20      "sirv-cli": "^1.0.0"
21  }
22  }

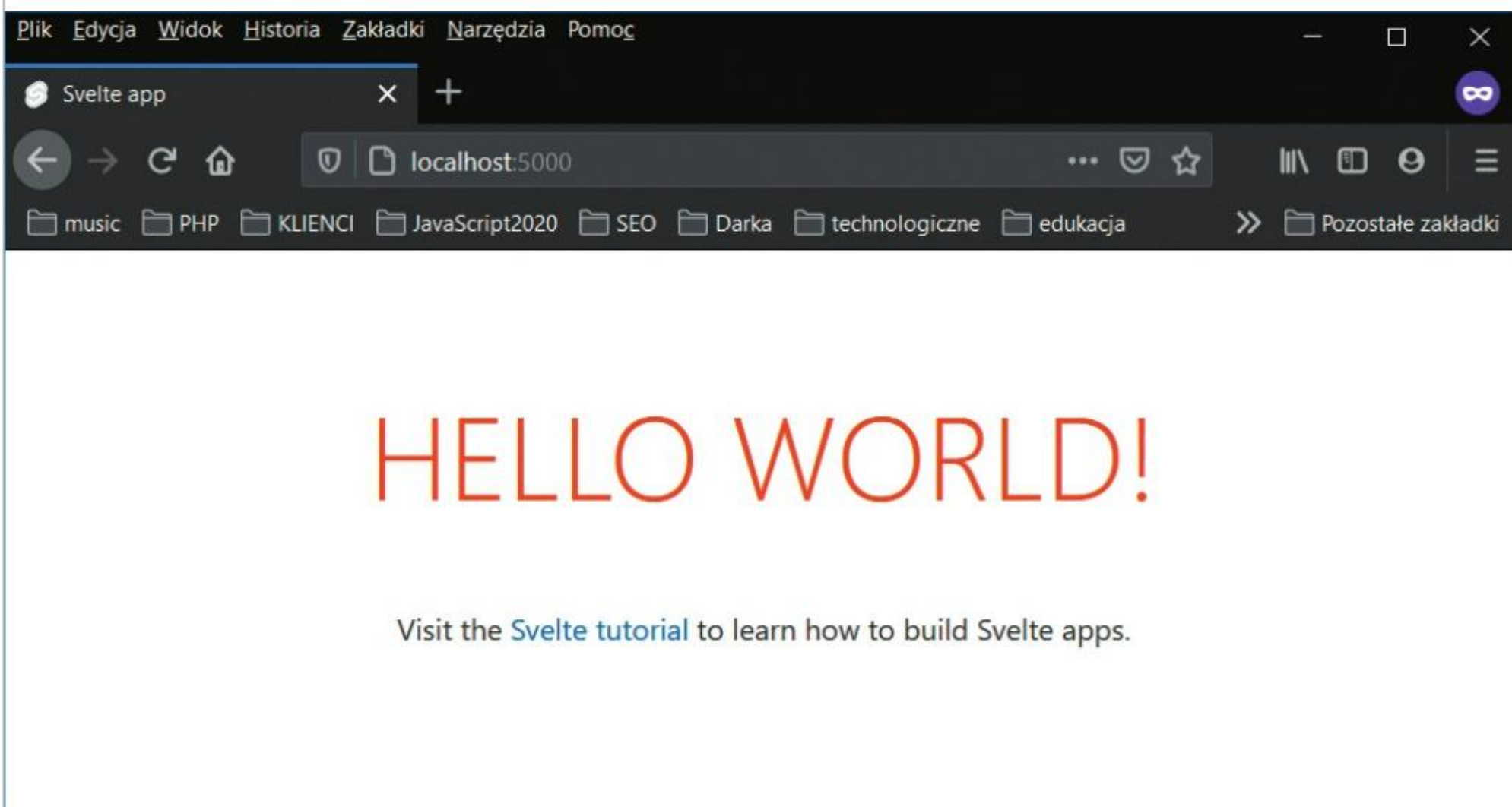
```

Rys. 26.9. Framework Svelte – widok kodów pliku konfiguracyjnego package.json

Podobnie jak w przypadku CSS i PHP, przybliżmy nieco możliwości zastosowania technologii JavaScript (opis na bazie informacji z oficjalnych stron frameworków):

- Svelte – <https://svelte.dev/>
„Aplikacje internetowe ulepszone cybernetycznie”.
- Angular – <https://angular.io/>
„Nowoczesna platforma dla programistów internetowych”.

- React – <https://pl.reactjs.org/>
„Javascriptowa biblioteka służąca do tworzenia interfejsów użytkownika”.
- Vue – <https://vuejs.org/>
„Progresywna struktura JavaScript”.
- jQuery – <https://jquery.com/>
„jQuery to szybka, mała i bogata w funkcje biblioteka JavaScript. To sprawia, że takie rzeczy jak przechodzenie i manipulowanie dokumentami HTML 5, obsługa zdarzeń, animacja i Ajax są znacznie prostsze dzięki łatwemu w użyciu API, które działa w wielu przeglądarkach. Dzięki połączeniu wszechstronności i rozszerzalności jQuery zmienił sposób, w jaki miliony ludzi piszą kod JavaScript”.



Rys. 26.10. Framework Svelte – widok w przeglądarce po instalacji i uruchomieniu serwera deweloperskiego

Zawsze gdy dobieramy framework, trzeba wziąć pod uwagę wiele czynników, m.in.:

- wielkość projektu,
- wiedzę, doświadczenie programistów w danej technologii i ich dostępność,
- poziom skomplikowania projektu,
- rodzaj projektu pod względem wyglądu i funkcjonalności.

CIEKAWOSTKA

Język TypeScript, który jest nadzbiorem języka JavaScript, oferuje opcjonalne statyczne typowanie, a także programowanie zorientowane obiektowo. Co ważne, każdy program napisany w języku JavaScript powinien być poprawnym programem TypeScript. Aplikacje napisane w TypeScript mogą stać się, w wyniku transpilacji, kodami języka JavaScript. Więcej o opisywanej tutaj technologii znajdziesz na <https://www.typescriptlang.org/>.


```
var data = {
  items: [{ text: 'JavaScript', checked: false },
    { text: 'PYTHON', checked: false },
    { text: 'PHP', checked: false }],
  title: 'Lista ulubionych języków programowania',
  newItem: ''
};

new Vue({
  el: '#app',
  data: data,
  methods: {
    addItem: function () {
      var text;
      text = this.newItem.trim();
      if (text) {
        this.items.push({
          text: text,
          checked: false
        });
        this.newItem = '';
      }
    }
  }
});
```

Rys. 26.11. Framework Vue.js – przykładowy kod

CIEKAWOSTKI

Vue.js (strona domowa projektu: <https://vuejs.org/>) to biblioteka JavaScript, którą można podłączyć do tworzonego projektu niemal na każdym jego etapie. Jest ona lekka i stosunkowo prosta do opanowania, szczególnie dla osób dobrze znających język programowania JavaScript. Jest jednym z trzech najpopularniejszych i najczęściej wykorzystywanych narzędzi opartych na technologii JavaScript. Pozostałe dwa to React.js i Angular.js. Na rys. 26.11 przedstawiono kod, który w połączeniu z kilkoma dodatkowymi linijkami znajdującymi się wewnątrz HTML-a tworzy listę aktywnych elementów (zadań do wykonania, ulubionych produktów itp.).

Według kilku rankingów popularności programiści interesują się przede wszystkim trzema rozwiązaniami, w kolejności: React, Angular, Vue.js. Na czwartym miejscu należy umieścić Svelte, który jest bardzo młodym rozwiązaniem, w ciągu ostatnich miesięcy bardzo zyskującym na popularności.

Poniżej znajdziesz krótką charakterystykę kilku ciekawych frameworków:

- Angular jest rozwiązaniem dostarczającym najbardziej kompleksowych, całościowych rozwiązań, a zarazem bardzo trudny do opanowania, szczególnie przez początkujących programistów.
- React jest (według rankingów) najpopularniejszym spośród wszystkich frameworków JavaScript, wymaga zwykle wsparcia zewnętrznymi modułami, ale za to jest zdecydowanie łatwiejszy do nauczenia i użytkowania niż Angular (choć nie jest rozwiązaniem równie kompleksowym).

```
<body>
  <div id="root"></div>

  <script type="text/babel">
    class App extends React.Component {
      render() {
        return <h1>Witaj w świecie komponentów
          REACT!</h1>
      }
    }

    ReactDOM.render(<App />,
      document.getElementById('root'))
  </script>
</body>
```

Rys. 26.12. Framework React – przykładowy kod

- Vue.js jest zdecydowanie najprostszy i najłatwiejszy do nauki (wśród trzech najbardziej popularnych frameworków JS) i umożliwia dołączenie go do niemal dowolnego projektu webowego właściwie „z marszu”.

- jQuery jest biblioteką JS, która kiedyś królowała w internecie. Obecnie jest często wykorzystywana, ale jako biblioteka służy głównie do manipulacji na HTML-owym DOM (ang. *Document Object Model*). Poza tym jest ona bardzo wolna i jej zastosowanie spowalnia szybkość funkcjonowania stron internetowych i aplikacji.
- Svelte jest jednym najbardziej rozwojowych frameworków JavaScript (według wielu specjalistów z zakresu technologii webowych). Ceniony za prostotę kodu, szybkość i możliwości rozwojowe dla budowanych aplikacji.

W tej części książki wymieniono zaledwie kilka frameworków związanych z technologiami CSS, PHP, JavaScript. Tego typu rozwiązań są prawdopodobnie tysiące. Mamy tutaj na myśli nie tylko frameworki z wymienionych trzech technologii, ale także z innych języków programowania, jak chociażby Django, będący bodajże najbardziej rozpoznawalnym obecnie frameworkiem języka Python, służącym również do tworzenia aplikacji webowych.

Bogactwo i różnorodność dotyczące frameworków może bardzo dobrze zostać zobrazowane przez GITHUB (oficjalna strona projektu: <https://github.com/> – portal z projektami programistycznymi rozwijanymi przez społeczność międzynarodową). Przykładowo: wpisanie w wyszukiwarce hasła (stan na luty 2021 r.):

- „framework” – uzyskanie informacji: „487,223 repository results”, czyli można znaleźć ponad 400 tys. repozytoriów związanych z zagadnieniem framework;
- „framework JavaScript” – uzyskanie informacji: „16,878 repository results”, czyli można znaleźć prawie 17 tys. repozytoriów związanych z wyszukiwanym zagadnieniem;
- „framework PHP” – uzyskanie informacji: „30,000 repository results”, czyli można znaleźć 30 tys. repozytoriów związanych z wyszukiwanym zagadnieniem;
- „framework CSS” – uzyskanie informacji: „15,441 repository results”, czyli można znaleźć ponad 15 tys. repozytoriów związanych z wyszukiwanym zagadnieniem.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wskaż technologię mogącą ułatwić tworzenie wyglądu strony WWW. Czego można użyć zamiast CSS i wraz z nim?
2. Wymień najpopularniejsze frameworki JavaScript.
3. Wymień najpopularniejsze frameworki PHP.
4. Wymień najpopularniejsze frameworki CSS.
5. Wyjaśnij, czym jest biblioteka jQuery.
6. Czy frameworku React.js można, twoim zdaniem, użyć w ramach strony (aplikacji) opartej na PHP i relacyjnej bazie danych? Poszukaj stosownych informacji w internecie. Odpowiedź uzasadnij.

27

Plan prac programistycznych nad SOAW – etapy realizacji i harmonogram projektu

ZAGADNIENIA

- Nowoczesne technologie – alternatywne planowanie projektu biznesowego
- Lean Canvas – szybkie i kompleksowe planowanie projektu technologicznego
- Kanban – przedstawienie zadań za pomocą tablicy
- Plan prac programistycznych – etapy realizacji i harmonogram projektu

27.1. Nowoczesne technologie – alternatywne planowanie projektu biznesowego

W tej części podręcznika uzyskasz informacje na temat dwóch popularnych metodologii związanych z procesami planowania i zarządzania projektem: Lean Canvas i tablica Kanban. Dotyczy to zarówno podejścia:

- **biznesowego** – Lean Canvas zawiera elementy finansowe: budżetowanie, planowanie i wynik finansowy.

Lean Canvas będzie omówione jako metoda szybkiej analizy opłacalności przedsięwzięcia gospodarczego. Dzięki tej metodzie można bardzo szybko określić sensowność realizacji planowanego projektu. Szczególnie często jest ona wykorzystywana w analizie przedsięwzięć związanych z nowoczesnymi technologiami;

- **organizacyjnego** – Kanban (oraz w nieco mniejszym stopniu Lean Canvas) to narzędzie do organizowania przedsięwzięć, w tym także projektów informatycznych.

Kanban może być wykorzystane do zorganizowania prostego w formie, ale równocześnie bardzo skutecznego systemu do zarządzania zadaniami. Oczywiście także podczas tworzenia aplikacji SOAW znalazłyby się obszary projektu, których realizację moglibyśmy przedstawiać jako tablicę Kanban.

27.2. Lean Canvas – szybkie i kompleksowe planowanie projektu technologicznego

Biznesplan na kartce papieru to jedno z możliwych tłumaczeń wyrażenia Lean Canvas. Idea metodologii Lean Canvas jest prosta: w bardzo krótkim czasie przygotować biznesplan (a raczej jego podstawowy zarys), aby szybko podjąć decyzję inwestycyjną. Ma on szerokie zastosowanie do biznesów i projektów w fazie start-up. Szczególnie dobrze nadaje się do zastosowania w branży nowoczesnych technologii. Jest spójny ideowo z filozofią Agile.

Istnieje wiele wersji i modyfikacji metody Lean Canvas. Zaprezentujemy tutaj podejście, które wypracowano na przestrzeni ostatnich kilku lat w trakcie pracy doradczej i trenerskiej.

PROBLEM	ROZWIĄZANIE	WARTOŚĆ	PRZEWAGA KONKURENCYJNA NIEUCZCIWA PRZEWAGA	KLIENCI
	KLUCZOWE WSKAŹNIKI		SPOSOBY DOTARCIA DYSTRYBUCJA	
KOSZTY	STRUKTURA KOSZTÓW	PRZYCHODY		

Rys. 27.1. Lean Canvas – tabela obrazująca podstawowe metody

Lean Canvas jest pisany i rysowany na kartce papieru. Zwykle jego analiza rozpoczyna się od górnej lewej części do środka, potem prawe kolumny, a dopiero po nich kolumna środkowa. Na końcu w Lean Canvas rozpatruje się jego dolne obszary, czyli część finansową. W ten sposób buduje się zintegrowaną biznesową narrację planowanego projektu. Zazwyczaj ta analiza jest wykonywana dla nowego biznesu lub projektu. Można ją wykonać w jeden dzień. Efektem jej przeprowadzenia jest decyzja – realizujemy przedsięwzięcie (wstępna decyzja) czy szukamy kolejnego pomysłu.

Lean Canvas: PROBLEM

Zwykle trzy problemy – trzy punkty.

Jaki problem klientów chcesz rozwiązać swoimi produktami?

Może jest alternatywne rozwiązanie?

Lean Canvas: ROZWIĄZANIE

Jak rozwiązujesz problem swoich klientów – twój produkt, usługa.

Wypisz trzy najważniejsze cechy twojego produktu lub usługi.

Lean Canvas: KLUCZOWE WSKAŹNIKI

Najważniejsze miary, które powiedzą, czy osiągasz cele.

Jak klienci postrzegają ciebie?

Jakie jest ich pierwsze wrażenie?

Co o tobie mówią? Czy wracają? Czy płacą?

Ile procent klientów płaci (będzie chciało zapłacić)?

Lean Canvas: WARTOŚĆ / UNIKALNA WARTOŚĆ

Realne korzyści, jakie odniesie klient, korzystając z twojego produktu / usługi.

Jakie problemy twój produkt pozwoli mu rozwiązać?

Stwórz dwa proste zdania wskazujące, co robi twój projekt / produkt / usługa.

Lean Canvas: PRZEWAGA KONKURENCYJNA

Nieuczciwa przewaga.

Wyróżnik twojego biznesu, który ciężko skopiować konkurencji.

Prawdziwa przewaga, której nie może „podrobić” konkurencja.

Lean Canvas: SPOSOBY DOTARCIA, DYSTRYBUCJA

Sposoby dotarcia do klientów.

Jakimi kanałami?

Lean Canvas: KLIENCI

Do kogo będziesz kierować swoją ofertę?

Wymieniasz grupy, które mają te problemy / potrzeby.

Podział demograficzny albo behawioralny (zachowanie).

Lean Canvas: KOSZTY

Koszty związane z projektem.

Struktura kosztów.

Badania rynku, promocja, reklama.

Wynagrodzenia, biuro, komputer, prawnicy, księgowość.

Lean Canvas: PRZYCHODY

Jak i ile będziesz na produkcie / usłudze zarabiać.

27.3. Kanban – przedstawienie zadań za pomocą tablicy

Jest to metoda, która pierwotnie miała pomagać (i zapewne ciągle znacznie pomaga) w sterowaniu produkcją. Opracowano ją w Japonii w latach pięćdziesiątych XX wieku. W języku japońskim Kanban oznacza ‘szyld’ bądź ‘tabliczkę informacyjną’.



Rys. 27.2. Prosta aplikacja Kanban wykonana w vanilla JavaScript – widok poziomy

Metoda Kanban pierwotnie opierała się na kartach poszczególnych wyrobów, ich cyrkulacji i analizie. Obecnie, oprócz pierwotnego znaczenia, stała się określeniem systemu informacyjnego wraz z system planowania, podziału i kontroli czynności zadań produkcyjnych. Jej ideą jest organizowanie procesu wytwórczego w taki sposób, aby każda komórka organizacyjna wyprodukowała tyle, ile w danej chwili jest potrzebne.

W ujęciu projektowym Kanban bardzo często jest przedstawiana jako trzy tablice:

- Zadania do wykonania,
- Zadania w trakcie wykonywania,
- Zadania wykonane.

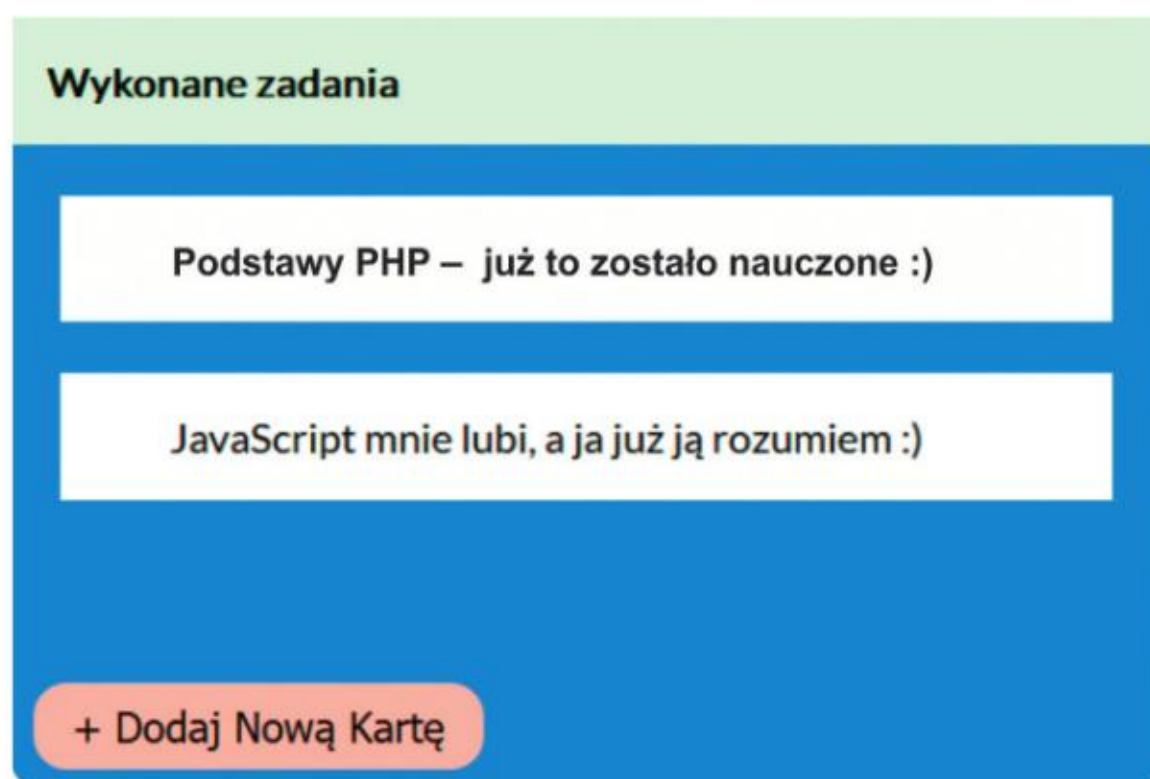
W trakcie trwania projektu poszczególne czynności są zapisywane po kolei do danej tablicy. Rozpoczyna się od „Zadania do wykonania”. Gdy zmieni się status zadania, zostaje ono przeniesione do kolejnej tablicy.



Rys. 27.3. Prosta aplikacja Kanban wykonana w vanilla JavaScript – karta „Do zrobienia”



Rys. 27.4. Prosta aplikacja Kanban wykonana w vanilla JavaScript – karta „Pracuję nad tym”



Rys. 27.5. Prosta aplikacja Kanban wykonana w vanilla JavaScript – karta „Wykonane zadania”

27.4. Plan prac programistycznych: etapy realizacji i harmonogram projektu

Etapy realizacji i harmonogram projektu

Tuż po podpisaniu umowy z klientem będą realizowane etapy zgodnie z założeniami dotyczącymi „cyklu życia produktu”.

Istnieje wiele różnorodnych podejść do „cyklu życia projektu”. W dużej mierze jest to związane m.in. z:

- różnorodnością produktów,
- rodzajami planowanych celów projektu,
- metodologią zarządzania,
- umiejscowieniem projektu.

W naszym przypadku mamy do czynienia z produktem technologicznym, który może się bardzo szybko zestarzeć. Z tego powodu stosujemy podejście oparte na filozofii Agile. Powinno to doprowadzić do zminimalizowania procesu koncepcyjnego i (w miarę możliwości) produkcyjnego.

Fazy cyklu życia projektu – podejście Agile

I. Tworzenie koncepcji graficzno-opisowej

- idea projektu – identyfikacja oczekiwań właściciela pomysłu biznesowego,
- określenie „historyjkowe” pożądanego stanu przyszłego projektu (planowanej aplikacji),
- analiza możliwości realizacji projektu wraz z określeniem najbardziej optymalnego zestawu technologii.

II. Zdefiniowanie projektu

- struktura zarządzania projektem – dobranie zespołu: kierownik projektu, projektanci, deweloperzy (programiści), ewentualnie pozostali członkowie zespołu realizującego projekt (przy bardzo rozbudowanym przedsięwzięciu),
- sporządzenie wstępnego harmonogramu,
- oszacowanie kosztów,
- analiza możliwości realizacyjnych projektu (wykonalności).

III. Realizacja projektu – pierwsza iteracja

- realizacja pierwszej wersji projektu,
- spotkanie – wprowadzenie korekt, ustalenie zakresu kolejnego etapu prac,
- rozwiązywanie bieżących problemów,
- testowanie projektu.

IV. Realizacja projektu – kolejne iteracje

- realizacja zaplanowanych podczas poprzedniej iteracji prac nad projektem,
- spotkanie – wprowadzenie korekt, ustalenie zakresu kolejnego etapu prac,
- rozwiązywanie bieżących problemów,
- testowanie projektu.

V. Etapy n-te

- pętla w realizacji projektów nowoczesnych technologii,
- powtarzanie kolejnych iteracji aż do uzyskania aplikacji spełniającej oczekiwania i potrzeby klienta – właściciela pomysłu biznesowego.

VI. Etap ostatni – zakończenie projektu

- oddanie działającej aplikacji klientowi (właścicielowi) i przekazanie odpowiedzialności za jej użytkowanie,
- uzyskanie pisemnej akceptacji potwierdzającej skuteczną realizację projektu,
- dokonanie wewnętrznej oceny pracy zespołu projektowego,
- opracowanie raportu dotyczącego parametrów biznesowych zrealizowanego projektu: czasu, kosztów i jakości,
- zarchiwizowanie dokumentacji projektowej.

Najważniejszym wnioskiem płynącym z tej części podręcznika jest to, że gdy mamy do czynienia z projektami tworzącymi i rozwijającymi nowoczesne projekty webowe, powinniśmy przede wszystkim mieć na uwadze założenia filozofii Agile. Dotyczy to zarówno etapu planowania, jak i budowania, a także dalszego rozwijania oprogramowania internetowego.



SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij, czym jest:
 - Kanban.
 - Lean Canvas.
2. Wyjaśnij pętle (iteracje) w realizacji projektów nowoczesnych technologii.

28

Zasoby ludzkie do realizacji SOAW

ZAGADNIENIA

- Analiza projektu uwzględniająca potrzeby technologiczne
- Dobór kadry – „człowiek orkiestra” czy specjalista ukierunkowany na daną technologię
- Pozostała kadra projektowa

28.1. Analiza projektu uwzględniająca potrzeby technologiczne

Poprzednie rozdziały tej książki wskazują, że potrzeby technologiczne projektu SOAW w zakresie zespołu projektowego powinny obejmować posiadanie przez personel kompetencji z zakresu:

- PHP,
- JavaScript,
- HTML 5,
- CSS,
- SQL,
- Visual Studio Code,
- XAMPP,
- Inkscape,
- GIMP,
- systemu kontroli wersji (GIT),
- obsługi tzw. standardowych programów (np. system operacyjny Windows, przeglądarki internetowe).

Wszyscy pracownicy techniczni projektu (programiści) powinni znać: Visual Studio Code, XAMPP i wymienione na ostatniej pozycji programy standardowe. Całość projektu powinna być prowadzona w takim samym środowisku, więc znajomość wymienionych programów jest dla wszystkich obowiązkowa.

Zanim przejdziemy do języków programowania i doboru kadry w tym zakresie, zastanówmy się jeszcze nad kontrolą wersji.

System kontroli wersji w obszarze informatyki to oprogramowanie, które potrafi śledzić zmianę w kodzie źródłowym i służy pomocą programistom współpracującym przy jednym projekcie w identyfikowaniu i łączeniu zmian dokonanych w plikach przez poszczególne osoby tworzące kod w różnym czasie. Jeżeli kod projektu byłby tworzony przez

więcej niż jedną osobę, to zastosowanie tego typu oprogramowania jest niezbędne. Nawet gdyby tylko jeden programista był odpowiedzialny za napisanie całej aplikacji, wówczas także warto wdrożyć w projekcie opisywany rodzaj oprogramowania. Zagadnienie to w postaci aplikacji szczegółowo omówiono w jednym z poprzednich rozdziałów:

- GIT – szczegóły projektu: <https://git-scm.com>,
- GITHUB Desktop – szczegóły projektu: <https://desktop.github.com/>.

Możemy je bez problemu prawnego wykorzystać w przygotowywanym projekcie, gdyż Git to wolne oprogramowanie – opublikowano je na licencji GNU General Public License version 2.0 (strona warunków licencji: <https://opensource.org/licenses/GPL-2.0>).

28.2. Dobór kadry – „człowiek orkiestra” czy specjalista ukierunkowany na daną technologię

Ważne, aby każdy z pracowników znał w stopniu podstawowym wszystkie wymienione technologie: PHP, JavaScript, HTML 5, CSS, SQL. Pozostaje decyzja, czy jedna osoba ma być specjalistą wysokiej klasy w całym obszarze, czy tylko w określonych technologiach.

```
<title>Człowiek orkiestra</title>
<style>
  /* Zmienna w CSS */
  body{
    --czerwony: red;
  }
  h1 {
    color: var(--czerwony);
  }
</style>
</head>

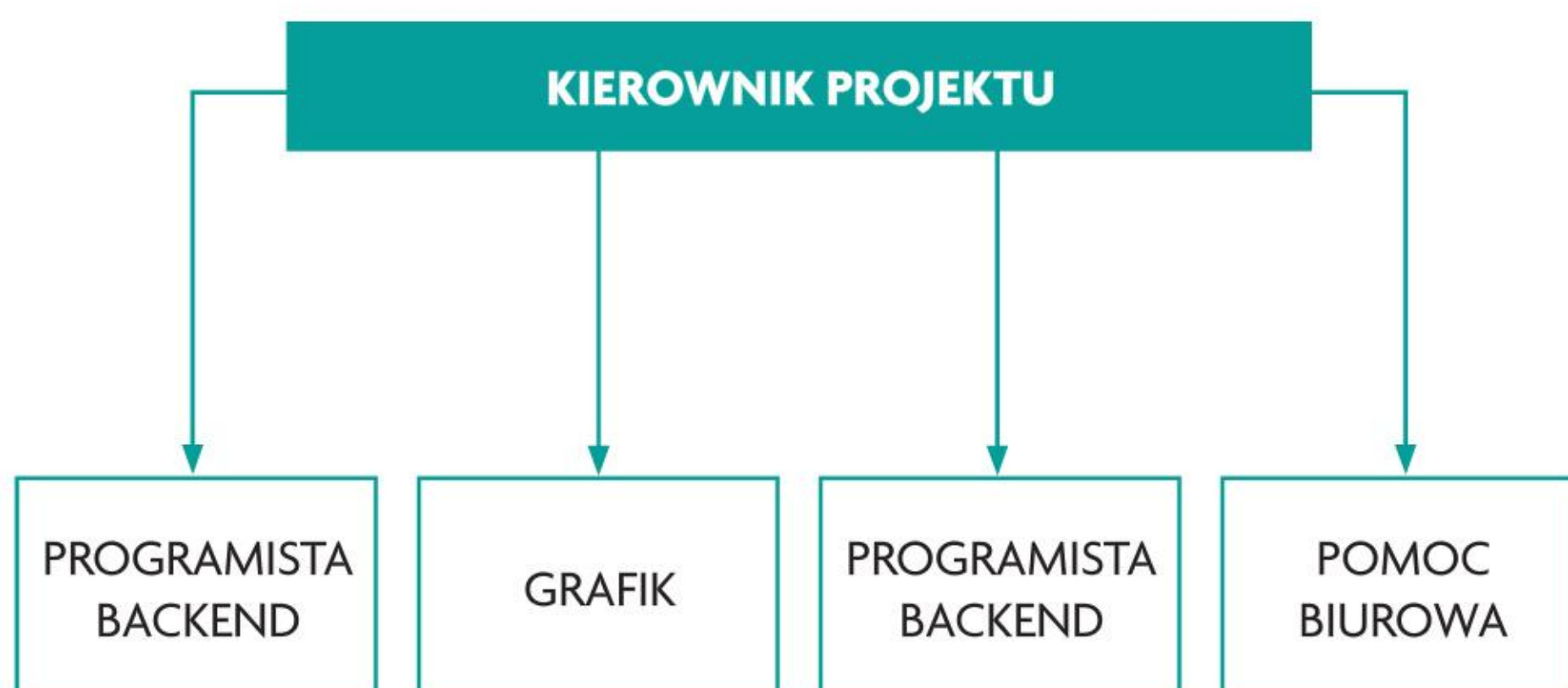
<body>
  <h1>Zmienne w PHP, JavaScript oraz CSS</h1>
  <!-- Zmienna w PHP -->
  <?php $zmiennaPHP = "Programuj z JS" ?>

  <script>
    // Zmienna w JavaScript
    let ZmiennaJavaScript = "Programuj z JS";
  </script>
</body>

</html>
```

Rys. 28.1. Zmienne w trzech językach programowania w jednym pliku: CSS, PHP, JavaScript

Oczywiście możemy zatrudnić osobę znającą się na wszystkim. W naszym przypadku oznacza to zaawansowany poziom wiedzy zarówno w zakresie technologii frontendowych, jak i backendowych. Takie rozwiązanie jest bardzo kuszące, ale zwykle, nawet przy niewielkich projektach, byłoby lepiej, aby współdziałały ze sobą co najmniej dwie osoby. Daje to choćby poczucie wzajemnej kontroli. Niewątpliwie istnieje też zagrożenie, że w przypadku problemów z funkcjonowaniem aplikacji – szczególnie na styku frontendu z backendem – pojawi się problem odpowiedzialności. Od ustalenia, kto zawinił, powinna być jednak kolejna osoba projektowa – pewnego rodzaju nadzorca, nazywany zwykle menedżerem projektu bądź kierownikiem projektu. W ten sposób przechodzimy do tzw. kadry projektowej – pozostałej.



Rys. 28.2. Przykładowa struktura zarządzania projektem – kierownik i podlegli mu pracownicy

28.3. Pozostała kadra projektowa

Pozostała kadra projektu (oprócz programistów) jest równie ważna do osiągnięcia sukcesu. Szczególnie istotny jest menedżer projektu, czyli osoba, która „zgodnie z duchem Agile” powinna się zajmować:

- organizacją pracy zespołu wraz z utrzymywaniem kontaktów z klientem;
- wspieraniem (doradztwem) mentalnym i merytorycznym działań pozostałych członków zespołu;
- decyzyjnym – w razie różnicy zdań (i ewentualnych konfliktów) menedżer powinien przeważyć szalę decyzyjną na jedną ze stron, ale nie powinien się narzucać jako autorytarny nadzorca.

Opisany standard w realizacji profesjonalnych projektów jest charakterystyczny dla branży informatycznej. Nie jest on jednak ani idealny, ani jedyny możliwy do zastosowania. W realizowanych projektach bardzo często pojawiają się inne stanowiska, np.:

- właściciel produktu (menedżer produktu),
- grafik,
- asystenci – pomoc biurowa, czyli osoby wspierające pozostały personel.



SPRAWDŹ SWOJĄ WIEDZĘ

1. Wytlumacz, dlaczego w projekcie będzie potrzebny co najmniej jeden programista backendowy.
2. Podaj powody zatrudnienia w projekcie programisty dobrze posługującego się zagadnieniami frontendowymi.
3. Jakiego rodzaju specjaliści będą potrzebni poza programistami, aby zrealizować projekt SOAW? Odpowiedź uzasadnij.
4. Co to jest „system kontroli wersji”?

29

Zarządzanie projektem

ZAGADNIENIA

- Zarządzanie projektem w projektach technologicznych
- Zarządzanie personelem w projektach technologicznych
- Zarządzanie kontaktami z klientami

29.1. Zarządzanie projektem w projektach technologicznych

Zarządzanie w zakresie realizacji projektów nowoczesnych technologii znacznie się różni od zarządzania tradycyjnymi projektami biznesowymi. Szczególnie istotne jest podejście do założonych celów. Zwykle cel główny – ogólny, czyli np. działająca aplikacja SOAW, mająca funkcje logowania, autorejestracji oraz dodawania i usuwania ogłoszeń, jest elementem niezmiennym. Założenie początkowe staje się efektem końcowym (choć wcale nie musi się tak stać). Nie można tego powiedzieć o celach cząstkowych. Podczas kolejnych iteracji poszczególne elementy aplikacji mogą ulegać znacznym zmianom, związanym ze zmianą wizji cząstkowej właściciela aplikacji i z nieprzewidzianymi elementami technologicznymi. Brak uwzględnienia wszystkich zmiennych jest związany m.in. z:

- rozpiętością, liczbą i różnorodnością poszczególnych technologii,
- szybkością rozwoju środowisk programistycznych,
- tempem i poziomem zagłębienia się w szczegóły w trakcie analizy projektu (dla projektów technologicznych wykonywanie analiz typu Lean Canvas).

Oczywiście w naszym bardzo prostym projekcie prawdopodobieństwo, że zmieni się w trakcie jego realizacji jeden z wymienionych czynników, jest niewielkie. Nasz projekt jest pewnym uproszczonym wzorcem; chodzi o ogólne schematy, które można wdrożyć także w realizacji dużych i bardzo dużych aplikacji webowych (a także innych z zakresu nowoczesnych technologii).

29.2. Zarządzanie personelem w projektach technologicznych

Zarządzanie projektem i personelem w projektach technologicznych zwykle różni się od tradycyjnego podejścia projektowego. Ma to związek ze specyfiką projektów i osób w nich zaangażowanych. Szczególnie istotnymi ogniwami zespołu są programiści, którzy bardzo często wymagają odmiennego traktowania niż np. pracownicy produkcyjni i biurowi. Wiąże się to m.in. z:

- rodzajem pracy – programowanie wymaga bardzo wysokiego poziomu skupienia i kreatywności – kierownik / menedżer powinien umieć zapewnić odpowiedni komfort pracy (także od strony sprzętu IT);
- specyfiką charakterologiczną – programiści są bardzo często osobami „zamkniętymi w sobie”, niejednokrotnie – pomimo posiadania potężnej wiedzy – boją się sięgać po mało konwencjonalne rozwiązania; zadaniem kierownika jest odpowiedni sposób zmotywowania do pracy i wyzwolenie kreatywności u osób programujących;
- rynkiem pracy – zatrudnienie można znaleźć dość łatwo, a im lepszy programista, tym większe ryzyko, że odejdzie, dlatego zadaniem kierownika jest odpowiedni sposób umotywowania do pracy nad danym projektem i u konkretnego pracodawcy.

Wyzwolenie wysokiego poziomu kreatywności ma istotne znaczenie nie tylko w przypadku programistów. Pozostali członkowie zespołu również powinni zostać tak pokierowani, aby wydobyć z nich ich najlepsze cechy. Dzięki temu praca zgodna z filozofią Agile będzie bardziej efektywna dla całego zespołu i powinna się przełożyć na zadowolenie klienta. Oczywiście mamy tutaj na myśli nie tyle pisanie kodu, ile pozostałe czynności, które trzeba rozważyć w trakcie realizacji projektu.

29.3. Zarządzanie kontaktami z klientami

Zaspokojenie potrzeb i oczekiwań klienta jest elementem kluczowym do odniesienia sukcesu biznesowego. Aby w pełni to osiągnąć, klientowi trzeba poświęcić czas niezbędny do zrozumienia motywów, które nim kierują. Podstawą jest uważne wsłuchiwanie się w oczekiwania klienta i kierowanie rozmowy na elementy niezbędne z punktu widzenia skuteczności realizacji projektu. Warto postępować zgodnie z ogólnie znaną zasadą: „Mowa jest srebrem, a milczenie złotem”.



Rys. 29.1. Komunikacja interpersonalna – wpływ zakłóceń na skuteczność przekazu

Aby skutecznie komunikować się z klientem, warto poznać zasady procesu komunikacji interpersonalnej.

Proces komunikowania się

Komunikacja interpersonalna to przekazywanie informacji między dwoma stronami (a czasem nawet większą liczbą stron). Mogą to być osoby bądź też grupy osób. Do zaistnienia komunikacji są potrzebne:

- informacja – treść, która ma być przekazana; zauważ, że ten punkt przekłada się na to, co masz do powiedzenia podczas spotkania z klientem;
- strony przekazu – co najmniej dwie, między którymi ma być przekazana informacja.

Jeżeli ma to być komunikacja skuteczna, dodatkowo muszą zaistnieć następujące bardzo ważne elementy:

- język przekazu – taki sam u obu stron lub zbliżony, tak aby nastąpiło wzajemne zrozumienie;
- obszar doświadczenia – wspólny dla obu stron;
- chęć porozumienia – jeżeli nie chcemy czegoś słyszeć i zrozumieć, to po prostu tak właśnie zrobimy;
- brak zakłóceń – odpowiednie warunki do usłyszenia przekazu.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest zarządzanie projektem?
2. Scharakteryzuj zasady zarządzania personelem w projektach technologicznych.
3. Co to jest komunikacja interpersonalna?
4. Opisz, na czym polega skuteczny kontakt z klientami.

VII. Programowanie zaawansowanych aplikacji webowych

- Omówienie wyglądu aplikacji SOAW – HTML, CSS
- SOAW – system logowania jako element serwisu
- SOAW – interakcja w aplikacji, „ciasteczkowy projekt”, kodowanie vanilla JavaScript
- SOAW – interakcja w aplikacji, przykłady kodu, biblioteka jQuery

30

Omówienie wyglądu aplikacji SOAW – HTML, CSS

ZAGADNIENIA

- Wykorzystywane technologie – dodatkowe elementy
- Strona tytułowa – wygląd
- Widok mobilny – Viewport
- Programowanie strony tytułowej – HTML, CSS, JavaScript, PHP
- Header – nagłówek aplikacji
- Content – zawartość główna aplikacji
- Footer – stopka aplikacji

30.1. Wykorzystywane technologie – dodatkowe elementy

Zapewne pamiętasz, jakie technologie będą wykorzystywane w projekcie SOAW. Przypomnimy, że oprócz języków programowania: HTML, CSS, JavaScript, PHP, SQL, PDO (łączenie frontendu z backendem) będziemy używali oprogramowania typu *open source* – Visual Studio Code (VSC), XAMPP.

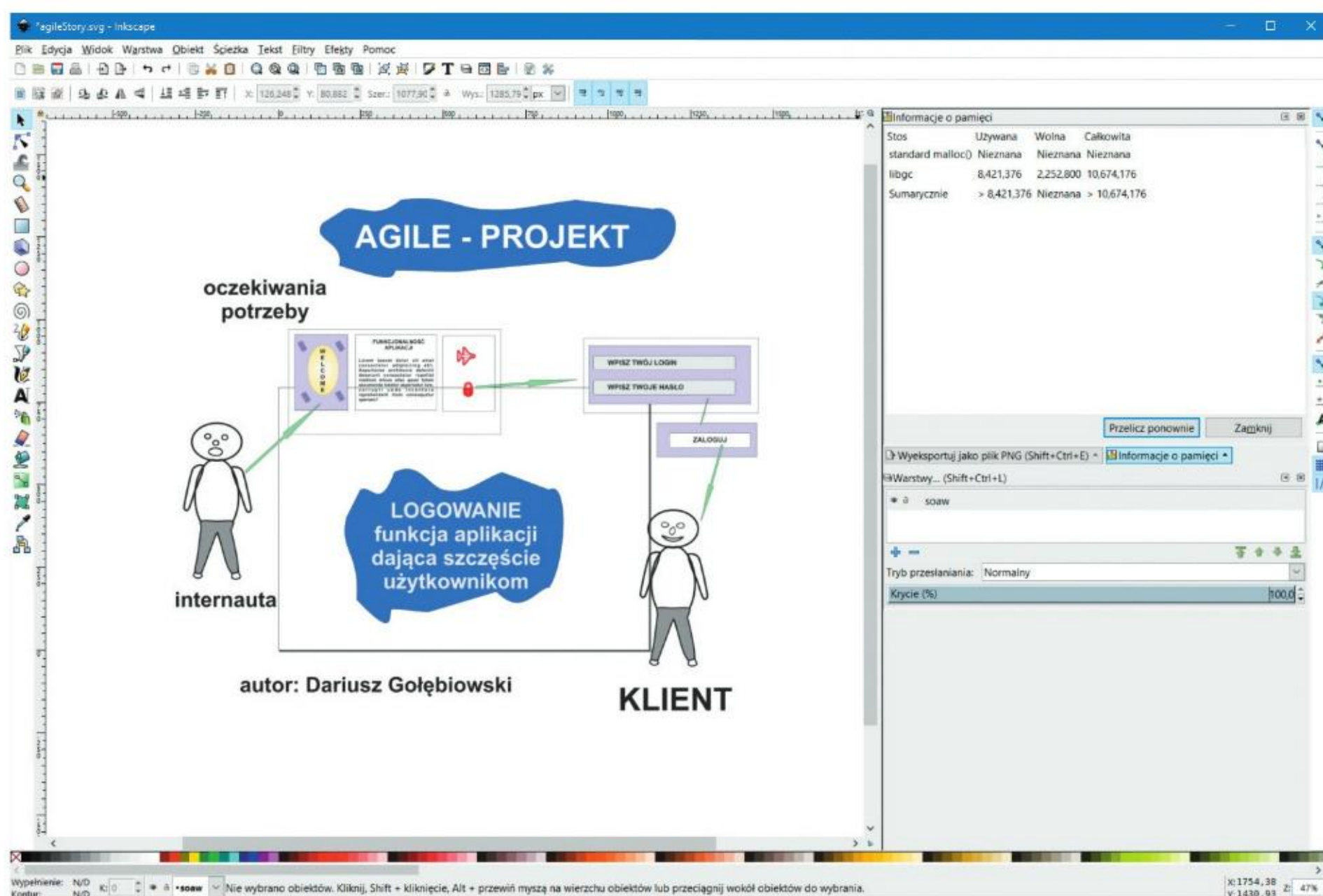
Grafika i obróbka zdjęć

Drobne elementy graficzne można opracować w programach opartych na filozofii *open source*, a ściślej – w bezpłatnych programach:

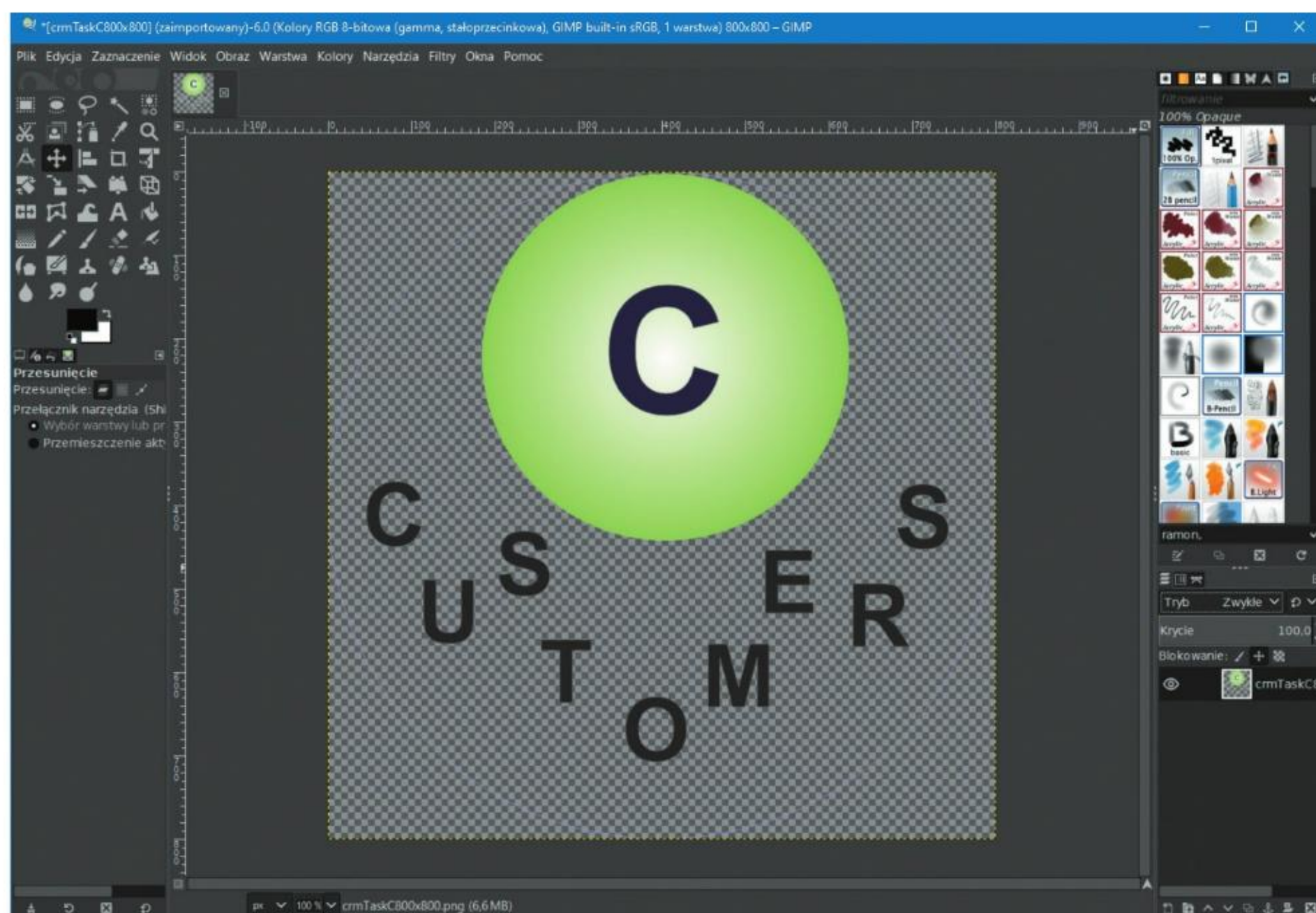
- Inkscape – program do tworzenia grafiki wektorowej (<https://inkscape.org/>),
- GIMP – program do edycji grafiki rastrowej (<https://www.gimp.org/>).

Kolejny istotny element to program Visual Studio Code (źródło: <https://code.visualstudio.com/>). W tym miejscu nie omawiamy szczegółów skonfigurowania tego programu ani też żadnej innej aplikacji do pisania kodów programistycznych. Oczywiście nie musisz używać VSC do pracy nad kodami, na rynku jest sporo ciekawych rozwiązań – tzw. **IDE** (ang. *Integrated Development Environment*).

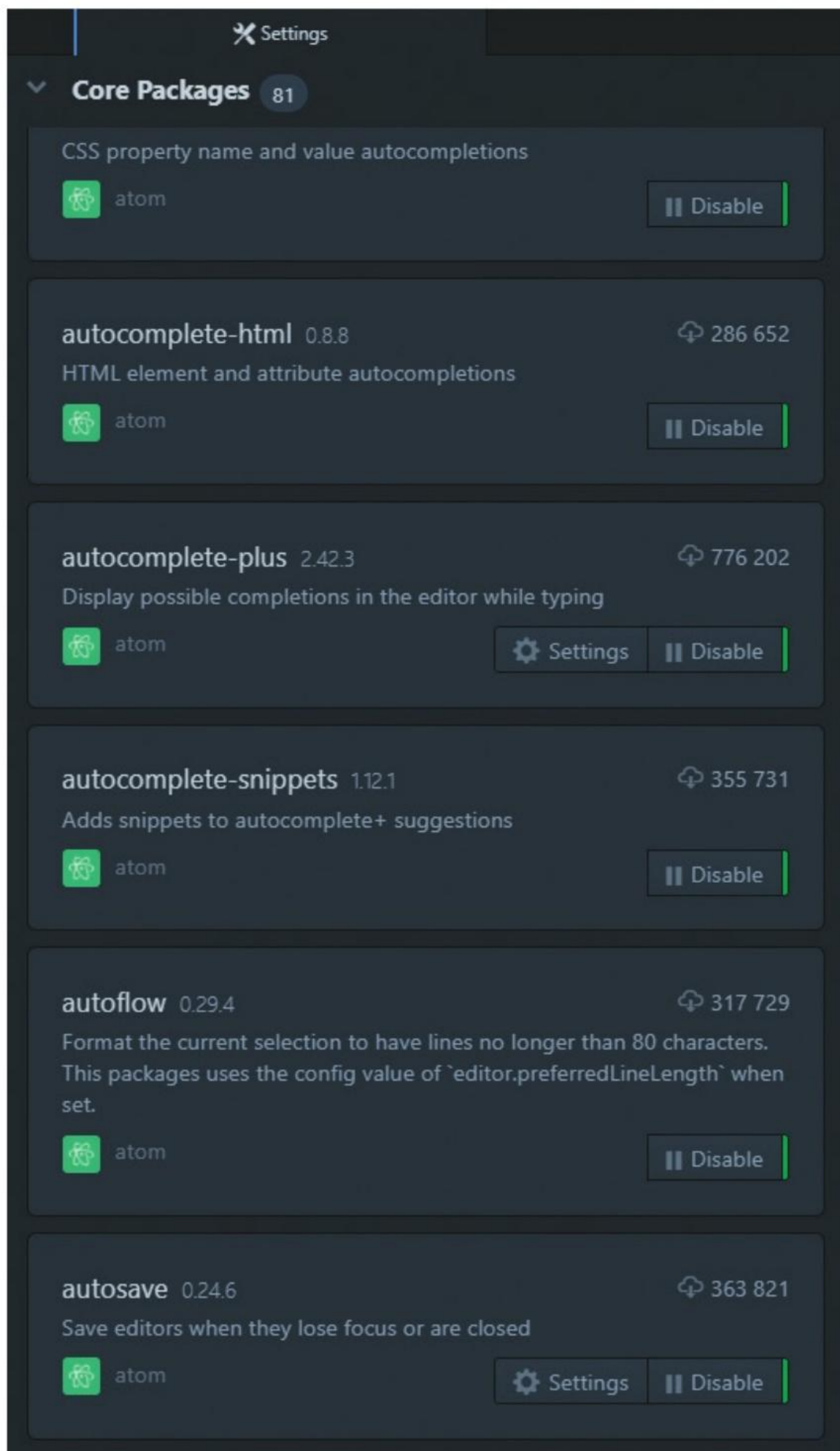
IDE to zintegrowane środowisko programistyczne – program lub pakiet programów (dlatego środowisko) służących do tworzenia, modyfikowania, testowania, konserwacji i utrzymania (bardziej fachowo: *maintenance*) oprogramowania. Programów będących IDE jest dużo, zarówno płatnych, jak i bezpłatnych. Najciekawsze bezpłatne to: Visual Studio Code, Brackets, Atom, NetBeans, Eclipse, CodeLite.



Rys. 30.1. Inkscape – widok zrzutu ekranowego interfejsu programu i obszaru roboczego programu



Rys. 30.2. GIMP – widok interfejsu programu i obszaru roboczego programu



Rys. 30.3. Aplikacja Atom.js – wycinek tzw. Packages (widok zainstalowanych modułów)

Używając IDE, warto pomyśleć o dodaniu (do VSC lub podobnego używanego programu) kilku podstawowych modułów, które znacznie zwiększą jego funkcjonalność w zakresie wykorzystywanych technologii. Poprawi to komfort pracy, co przełoży się na szybsze pisanie funkcjonalnego, wysokiej jakości i ładnego kodu. Odpowiednie moduły umożliwią nie tylko uzyskiwanie podpowiedzi co do kodu, lecz także spowodują jego poprawne ułożenie na stronie (wcięcia, odstępy itp.).

Przydatnych modułów istnieją tysiące, ale na początek wymienimy tylko kilka, za to najbardziej użytecznych pod względem tworzonego projektu (używanych języków programowania). Należy wybierać dodatki, patrząc na kryterium liczby instalacji (ang. *install count*). Zwykle jest tak, że im więcej osób używa danego modułu, tym wyższa jest jego jakość.

Przykładowe użyteczne moduły do VSC:

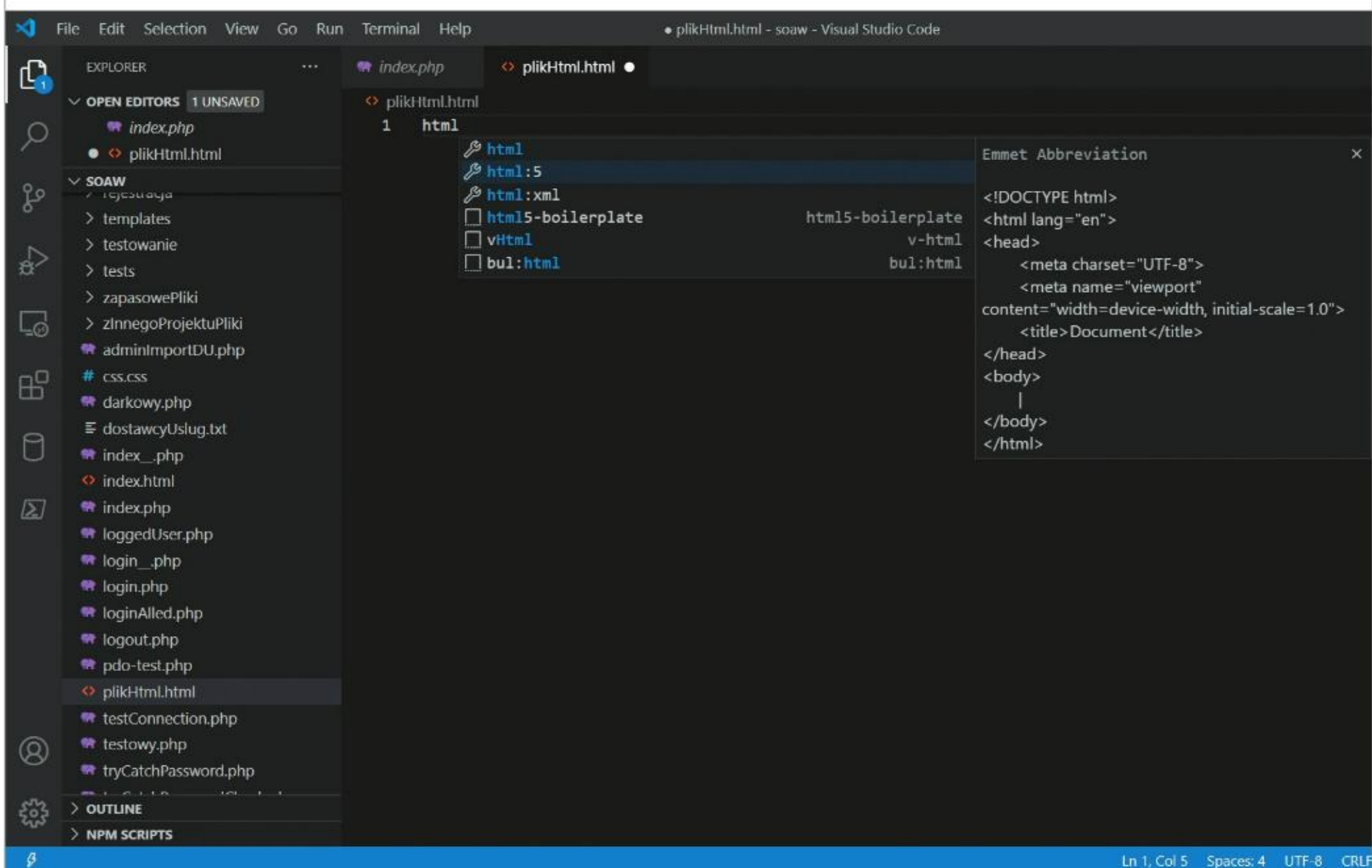
- Beautify – oficjalna strona projektu na github
<https://github.com/HookyQR/VSCoDeBeautify>,
- HTML CSS Support – oficjalna strona projektu na github
<https://github.com/ecmel/vscode-html-css>,
- PHP IntelliSense – oficjalna strona projektu na github
<https://github.com/felixfbecker/vscode-php-intellisense>,
- PHP Server – oficjalna strona projektu na github
<https://github.com/brapifra/vscode-phpserver>,
- IntelliSense for CSS class – oficjalna strona projektu na github
<https://github.com/brapifra/vscode-phpserver>,
- JS-CSS-HTML Formatter – oficjalna strona projektu na github
<https://github.com/lonefy/vscode-js-css-html-formatter>,
- Live Server – oficjalna strona projektu na github
<https://github.com/ritwickdey/vscode-live-server>,
- JavaScript (ES6) code snippets – oficjalna strona projektu na github
<https://github.com/xabikos/vscode-javascript>
- SQLTools – oficjalna strona projektu na github
<https://github.com/mtxr/vscode-sqltools>,
- Babel JavaScript – oficjalna strona projektu na github
<https://github.com/michaelgmcd/vscode-language-babel>.

Ostatni element to kwestia obsługi programu do pisania kodów, np. wspomnianego już VSC. Musisz poświęcić trochę czasu, aby poznać daną aplikację w zakresie podstawowej obsługi i dodatkowych, bardzo przydatnych funkcji. Przykładem mogą być skróty klawiszowe, które bardzo przyspieszają pracę, np. jeden z najczęściej używanych ALT+Z, który zawija tekst na ekranie.

Szczegółów warto poszukać w internecie, gdzie są wszystkie niezbędne informacje zarówno co do obsługi programu do pisania kodów, jak i dodatkowych modułów rozszerzających jego podstawową funkcjonalność.

30.2. Strona tytułowa – wygląd

Zbudujemy stronę tytułową z wykorzystaniem VSC. Najpierw tworzymy plik `index.html`, a następnie wpisujemy w nim (pole edycyjne pliku) `html`. W oddzielnym okienku program podpowiada opcje – wybierzemy `html:5`.



Rys. 30.4. Tworzenie strony HTML 5 na podstawie szablonu dostępnego (już zaprogramowanego) w VSC

W ten łatwy sposób stworzyliśmy podstawową strukturę strony głównej dla naszego projektu. Gdy chcemy uzyskać inną wersję językową niż angielska, np. polską, to zamieniamy `<html lang="en">` na `<html lang="pl">`.

W znaczniku `<head>` uzupełniamy jego zawartość. W znaczniku `<title>` wpisujemy nazwę projektu (rys. 30.5).

W kodach znajduje się m.in. podlinkowanie pliku `style.css` (stworzymy go wkrótce, w nowym katalogu o nazwie `css`). Dzięki temu nasz projekt będziemy budowali zgodnie z najnowszymi trendami, w których poszczególne technologie, implementowane w projekcie webowym, umieszczamy w oddzielnych katalogach.

W head znajduje się także informacja na temat strony kodowej tworzonego projektu "UTF-8" i o ustawieniach dla urządzeń mobilnych: `meta name="viewport"`.


```

<> plikHtml.html    <> index.html X
index.html > html
1  <!DOCTYPE html>
2  <html lang="pl">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width,
          initial-scale=1.0">
7      <title>Serwis Ogłoszeniowy Aplikacja Webowa</title>
8      <link rel="stylesheet" href="css/style.css">
9  </head>
10

```

Rys. 30.5. Kod strony HTML – deklaracje HTML 5, języka i nagłówka (head) zawierający podstawowe ustawienia

30.3. Widok mobilny – Viewport

Istnieje wiele rozwiązań, dzięki którym witryna internetowa będzie zoptymalizowana do wyświetlania na urządzeniach mobilnych. Jednym z nich jest umieszczenie w kodzie HTML znacznika:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Właściwość **width** (szerokość) kontroluje wielkość widocznego pola. Można jej przypisać konkretną liczbę, np. `width=1200`, albo specjalne słowo kluczowe związane z technologią mobilną – **device-width**. Oznacza ono nadanie wielkości ekranu dla danego urządzenia w pikselach, przy zoomie na poziomie 100%. Dodatkowo można zastosować polecenia **height** i **device-height**. Warto je wykorzystać, gdy wielkość danego elementu zależy od wysokości ekranu.

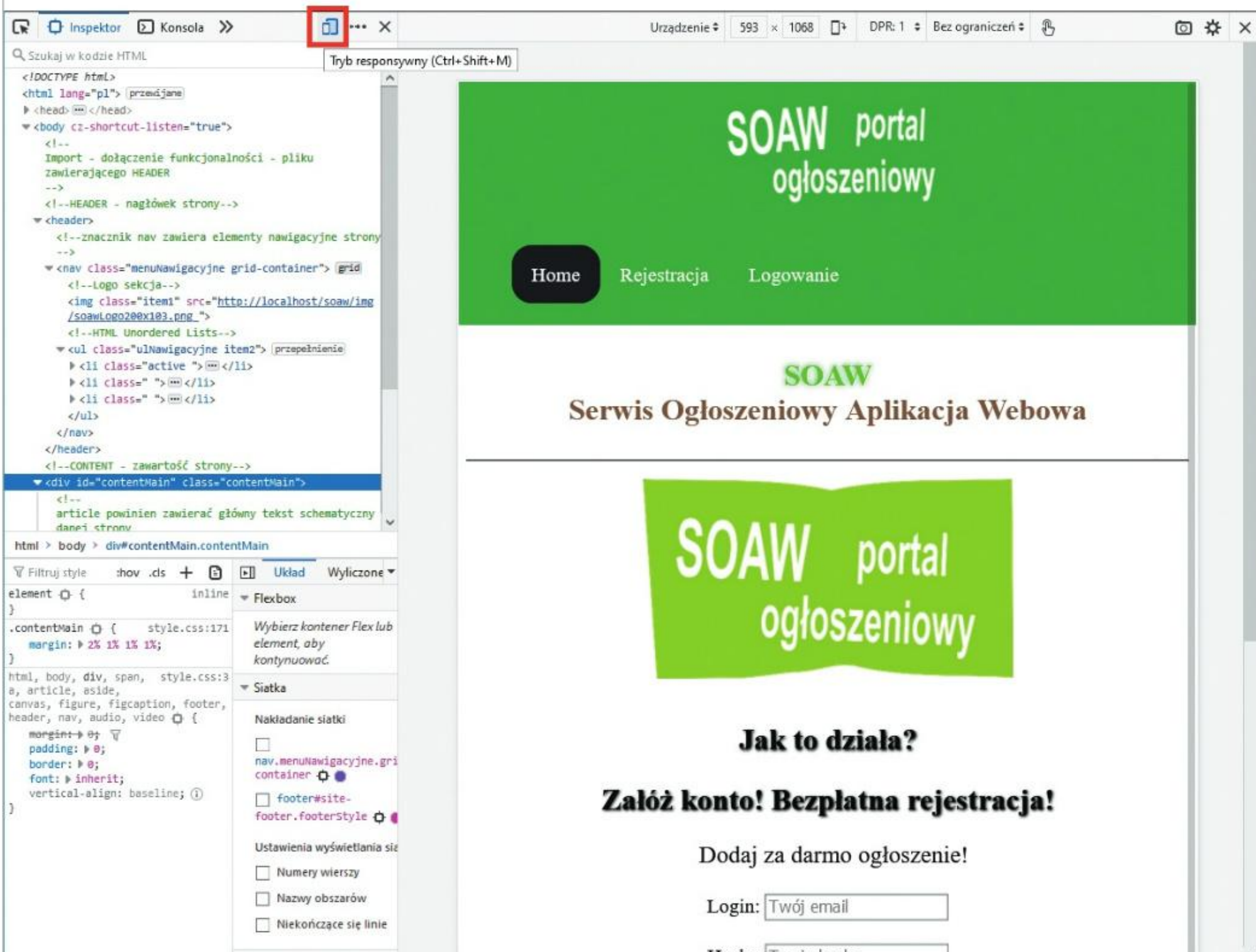
Trzeba też pamiętać o właściwości **initial-scale**, która kontroluje poziom zbliżenia po załadowaniu strony. W naszym przypadku mówimy o standardowym przybliżeniu, czyli równym 1.

Użyte ustawienia sprawią, że strona nie będzie skalowana, a użytkownik będzie miał możliwość jej przybliżenia lub oddalenia. Strona będzie się rozciągała na 100% szerokości ekranu, na którym będzie wykorzystana.

Zmień ustawienia powyższych elementów, by sprawdzić, jaki będzie widok dla urządzeń mobilnych. Na rys. 30.6 znajdziesz (zaznaczone czerwonym kolorem) miejsce włączania widoku mobilnego z poziomu Firefox Developer Tools. Narzędzia deweloperskie w przeglądarkach internetowych są włączane (zwykle) przez naciśnięcie klawisza F12. Standardowo panel deweloperski pokazuje się na dole ekranu (pod stroną WWW),

ale możesz ustawić wyświetlanie w innym miejscu. Na przykład tak jak we wspomnianym przykładzie – z lewej strony. Zapewne zwróciłeś uwagę, że strona na zdjęciu jeszcze nie jest responsywna. Menu nie wyświetla się poprawnie, a pola tekstowe trzeba dopracować.

Najpierw stworzymy strukturę HTML, a potem za pomocą CSS zajmiemy się wyglądem. Weźmiemy pod uwagę także responsywność i przyjazność dla urządzeń mobilnych.



Rys. 30.6. Panel deweloperski – Firefox Developer Tools (kolorem zaznaczona ikona włączająca widok mobilny)

W internecie znajdziesz bardzo dużo samouczków na temat wykorzystania panelu deweloperskiego. Szczególnie cenny jest stworzony przez twórców z fundacji Mozilla: <https://developer.mozilla.org/en-US/docs/Tools>. Jeżeli używasz innej przeglądarki niż Firefox, wówczas trzeba poszukać stron zawierających analogiczne poradniki, np.:

- Chrome
https://developer.chrome.com/docs/extensions/reference/devtools_panels/,
- Edge
<https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/>.

! UWAGA

Odświeżanie widoku strony

W trakcie pracy nad designem (wyglądem) stron projektu zazwyczaj chcemy zobaczyć w przeglądarce wykonane zmiany w plikach CSS. Zwykle musisz wyczyścić pamięć podręczną używanej przeglądarki, aby zobaczyć zmiany na stronie, a i to nie zawsze gwarantuje, że widok się odświeży. Rozwiązaniem alternatywnym jest praca w trybie prywatnym. Pamiętaj, żeby za każdym razem po zmianach w CSS i zapisaniu ich zamknąć okno z trybem prywatnym i ponownie je otworzyć.

30.4. Programowanie strony tytułowej – HTML, CSS, JAVASCRIPT, PHP

Stronę tytułową projektu SOAW zbudujemy w taki sposób, aby stanowiła wzorzec (często określany jako template, czyli szablon) dla kolejnych podstron. W tym celu wyodrębnimy elementy powtarzające się na każdej podstronie:

- **header** – będzie zawierał nagłówek strony, od lewej: logo SOAW i główne menu nawigacyjne;
- **content** – zawartość strony;
- **footer** – stopka strony.

Na rys. 30.7 przedstawiono postać strony głównej projektu (między znacznikami `<body>` `</body>`).

```
<body>
  <!-- HEADER - nagłówek strony -->
  <header>
    [Logo SOAW i menu nawigacyjne]
  </header>
  <!-- CONTENT - zawartość strony -->
  <div id="contentMain" class="contentMain">
    [Zawartość strony]
  </div>
  <!-- FOOTER - stopka strony -->
  <footer>
    [Stopka strony]
  </footer>
</body>
</html>
```

Rys. 30.7. Strona główna projektu SOAW – struktura

30.5. Header

Zacznijmy omawianie wnętrza body od sekcji header. Chcemy, aby od lewej strony wyświetlało się logo SOAW, a następnie menu nawigacyjne. Na rys. 30.8 przedstawiono właściwy kawałek kodu HTML wykonujący wyświetlanie obrazka i tworzący element funkcjonalny.

```
<!-- HEADER - nagłówek strony -->
<header>

  <!-- znacznik nav zawiera elementy nawigacyjne strony -->
  <nav class="menuNawigacyjne grid-container">

    <!-- Sekcja logo -->
    

    <!-- HTML Unordered Lists -->
    <ul class="ulNawigacyjne item2">
      <li class="active"><a href="#home">Home</a></li>
      <li><a href="#link1">Link 1</a></li>
      <li><a href="#link2">Link 2</a></li>
      <li><a href="#link3">Link 3</a></li>
      <li><a href="#link4">Link 4</a></li>
    </ul>

  </nav>

</header>
```

Rys. 30.8. Strona główna projektu SOAW – kody dla obszaru header

Logo SOAW przygotowano za pomocą Inkscape. Oczywiście możesz dodać dzieło wykonane własnoręcznie. Żeby uruchomić kod, najpierw utworzymy katalog `img`, w którym będziemy umieszczali niezbędne grafiki. Podział projektu na katalogi tematyczne jest rozwiązaniem bardzo praktycznym i powszechnie stosowanym, ułatwiającym zarządzanie rozwojem zasobów niezbędnych do stworzenia aplikacji. Następnie do wymienionego katalogu wkleimy przygotowany wcześniej plik z logo.

Jeżeli masz LiveServer w swoim VSC, możesz od razu włączyć stronę index.html i sprawdzić, czy wszystko działa. Strona powinna się uruchomić najprawdopodobniej pod adresem: `http://127.0.0.1:5501/index.html` w przeglądarce ustawionej jako główna dla twojego systemu operacyjnego.

Zwróć uwagę, że w kodzie została stworzona struktura menu z wykorzystaniem znaczników `<nav>` i `<ul>`. Aby nadać projektowi właściwy wygląd, stworzymy plik stylów CSS `style.css` (w katalogu `css`), w którym na początek wyzerujemy niektóre ustawienia początkowe przeglądarek internetowych.

```
/* Reset CSS START */

html,
body,
div,
span,
a,
article,
aside,
canvas,
figure,
figcaption,
footer,
header,
nav,
audio,
video {
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    vertical-align: baseline;
}
```

Rys. 30.9. Przykładowy reset ustawień początkowych stylów CSS przeglądarek internetowych

Następnie w `style.css` dodamy kilka linijek kodu, by uzyskać efekt sticky, czyli przyklejania się menu do góry strony podczas skrołlowania w dół (rys. 30.10).


```
header {  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    background-color: #333333; /* rgb(48, 12, 12); */  
    position: -webkit-sticky;  
    /* Safari */  
    position: sticky;  
    top: 0;  
}
```

Rys. 30.10. Kod powodujący efekt sticky (przyklejania do górnej krawędzi ekranu podczas scrolowania, czyli przewijania kółkiem myszki)

Aby nawigacja wyświetlała się w oczekiwany przez nas sposób, czyli:

- w poziomie,
 - z usunięciem efektu wypunktowania,
 - z ustawieniem kierunku przepływu elementów w bloku header na lewy,
- potrzebujemy dodać kolejne linijki kodu, tak jak na rys. 30.11.

```
/* Wyświetlanie nawigacji w poziomie START */  
  
header ul {  
    /* usunięcie efektu wypunktowania */  
    list-style-type: none;  
}  
  
/* Ustawienie kierunku przepływu elementów w bloku header na lewy */  
  
header ul li {  
    float: left;  
}
```

Rys. 30.11. Kod powodujący wyświetlenie nawigacji w oczekiwany przez nas sposób


```
/* Wskazanie właściwego sposobu wyświetlania elementów */
```

```
header ul li a {  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}
```

```
header ul li a:hover {  
    border: 0.1em solid #111;  
    border-radius: 1em;  
}
```

Rys. 30.11. Kod powodujący wyświetlenie nawigacji w oczekiwany przez nas sposób – ciąg dalszy

Wróćmy jeszcze do sekcji header. Znajdziesz tam klasę grid-container, która wymaga dodatkowego wyjaśnienia. Mimo że nie będziemy omawiali szczegółów CSS, warto wspomnieć o grid. Najpierw popatrz na kod w style.css (rys. 30.12).

```
/* Ten element powoduje, że wszystkie zawarte w nim bloki ustawiają  
się w grupy po dwa w jednym wierszu, dopasowując się do szerokości  
zawartego w każdym z nich contentu (zawartości). */
```

```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto;  
    grid-gap: 10px;  
    padding: 10px;  
}
```

```
.grid-container>div {  
    text-align: center;  
    padding: 20px 0;  
    font-size: 30px;  
}
```

Rys. 30.12. Kod CSS powodujący wyświetlenie kolejnych bloków (kolumn) po dwa w każdym wierszu

Warto zwrócić uwagę, że w znaczniku: `<nav class="menuNawigacyjne grid-container">` znajdują się dwa bloki: `img` i `ul` (rys. 30.13).

```
<!-- znacznik nav zawiera elementy nawigacyjne strony -->
<nav class="menuNawigacyjne grid-container">

  <!-- Sekcja logo -->
  

  <!-- HTML Unordered Lists -->
  <ul class="ulNawigacyjne item2">
    <li class="active"><a href="#home">Home</a></li>
    <li><a href="#link1">Link 1</a></li>
    <li><a href="#link2">Link 2</a></li>
    <li><a href="#link3">Link 3</a></li>
    <li><a href="#link4">Link 4</a></li>
  </ul>

</nav>
```

Rys. 30.13. Dwa bloki – `img` i `ul`

Właściwości powodują, że wymienione dwa elementy znajdą się obok siebie.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto;
  grid-gap: 10px;
  padding: 10px;
}
```

Rys. 30.14. Kod CSS – właściwość `display`, wartość `grid`

Dodatkowo uzyskujemy automatyczne dopasowanie zawartości do szerokości zawartej w każdym z nich. Techniki oparte na grid i grid-template-columns są bardzo ważne, gdyż ułatwiają pracę z elementami blokowymi na stronie WWW i ustawianie poszczególnych elementów HTML we właściwy sposób na stronach tworzonej aplikacji. Tutaj mamy do czynienia z bardzo prostym kawałkiem kodu, dotyczącym zaledwie dwóch elementów w jednym wierszu. Jest to niewielki procent możliwości wykorzystania tej techniki CSS. Spróbuj np. zamienić zapis:

```
grid-template-columns: auto auto;
```

na

```
grid-template-columns: 80% 20%.
```

Spójrz, jak zmieniły się kolumny. Teraz pierwsza ma 80% szerokości elementu nadrzednego (mniej więcej strony), a druga zaledwie 20%.

Zauważ, że dzięki temu, dostarczając dla urządzeń mobilnych, np.:

```
/* Style dla urządzeń mobilnych */  
  
@media screen and (max-width: 768px) {  
  .grid-container {  
    display: grid;  
    grid-template-columns: auto;  
    min-width: 100%;  
    grid-gap: 10px;  
    padding: 2px;  
  }  
  .grid-container>div {  
    text-align: center;  
    padding: 2px 0;  
    font-size: 20px;  
  }  
}
```

Rys. 30.15. Kod CSS – właściwość display, wartość grid – ustawienie dla urządzeń mobilnych

uzyskamy to, że każda z kolumn będzie zawarta na całej szerokości strony, czyli znajdują się one pod sobą. Dokładnie tak, jak chcielibyśmy, żeby wyglądała strona mobilna budowanej aplikacji.

Więcej informacji na temat grid znajdziesz m.in. na stronie MDN: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout.

30.6. Content – zawartość główna

Podstawowa część informacyjna projektowanej aplikacji będzie się znajdowała wewnątrz bloku (tzw. główna zawartość). W zależności od adresu html – strony WWW, do której przejdzie użytkownik aplikacji, będzie serwowana inna treść informacyjno-graficzna. Będzie to jedyny element, który znajdzie się bezpośrednio na stronie index.php (już wkrótce zamienimy index.html na index.php). Pozostałe elementy strony będą importowane za pomocą PHP z innych plików (tzw. templatek – szablonów).

```
<!-- CONTENT - zawartość strony START -->
<div id="contentMain" class="contentMain">
  <article>
    <h1 class="h1Style"><span>SOAW</span> <br>
    Serwis Ogłoszeniowy Aplikacja Webowa</h1>
    <hr>
    <div class="soawContent">
      <div class="soawContentLeft">
        
      </div>
      <div class="soawContentRight">
        <h2 class="h2Style">Jak to działa?</h2>
        <h2 class="h2Style">
          Załącz konto! Bezpłatna rejestracja!
        </h2>
        <p>&nbsp; Dodaj za darmo ogłoszenie!</p>
      </div>
    </div>
  </article>
</div>
```

Rys. 30.16. Kod HTML (strona index.html) – kod początkowy głównej zawartości – część 1


```

<form action="#" method="POST"
      class="underHeaderInputs">
  Login: <input type="text"
           placeholder="Twój email" name="email">
  <br>
  <br> Hasło: <input type="text"
           placeholder="Twoje hasło" name="password"><br>
  <br> <a href="#">Rejestracja</a>
  &nbsp;
  <input class="inputLogIn" type="submit"
        value="Zaloguj"/>
</form>

<p class="underHeaderInputs">LOGIN</p>
<p class="underHeaderInputs">
  <a href="login.php">Zaloguj Się</a>
</p>
</div>
</div>
</article>
</div>
<!-- CONTENT - zawartość strony END -->

```

Rys. 30.17. Kod HTML (strona index.html) – kod początkowy głównej zawartości – część 2

30.7. Footer – stopka aplikacji

W stopce możemy umieścić informacje o firmie, prawach autorskich i inne istotne z punktu widzenia budowanego projektu. Uzupełnimy ją w jednym z dalszych etapów prac nad tworzoną aplikacją.

```

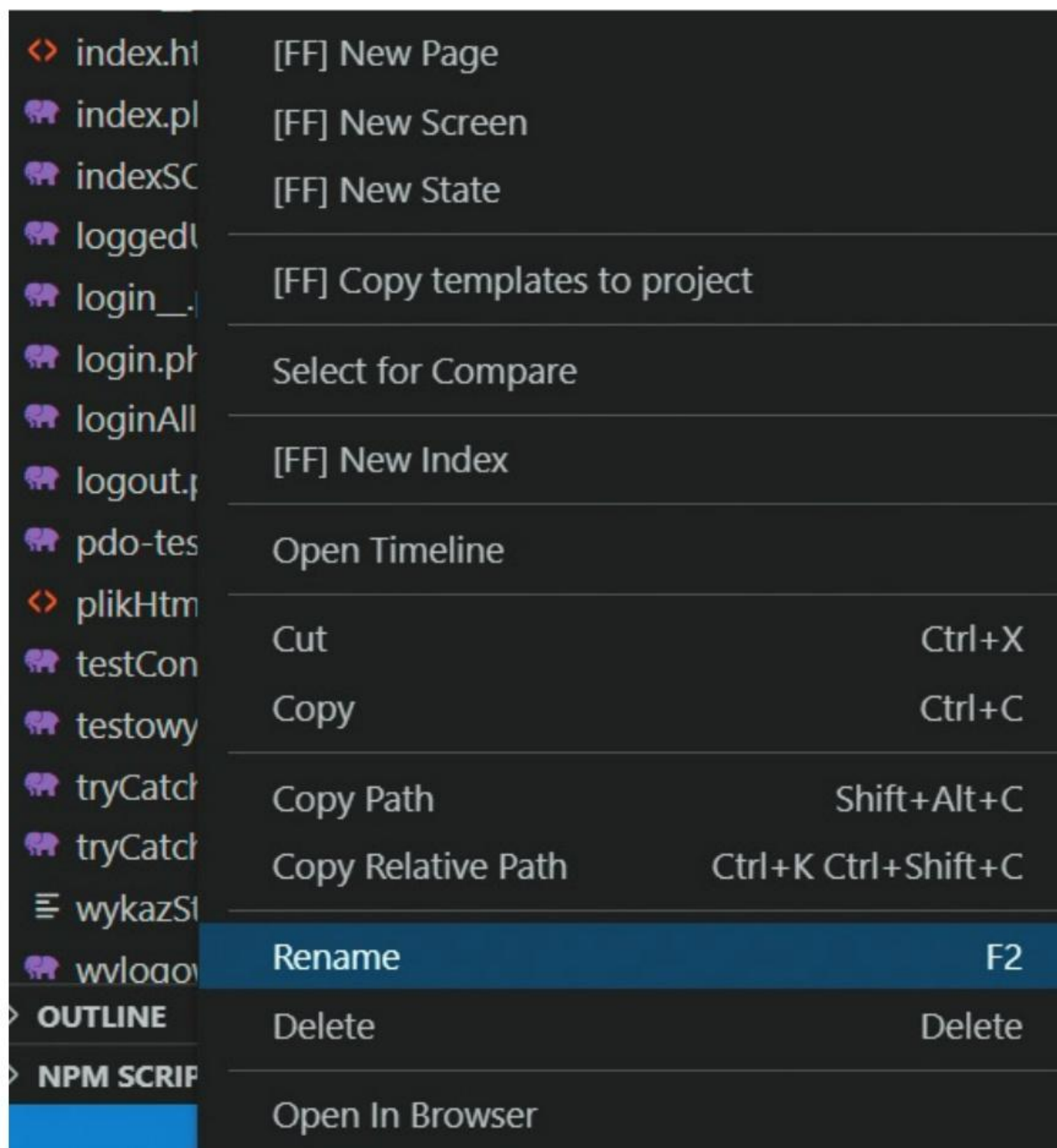
<!-- FOOTER - stopka strony -->
<footer>
  <h3>To jest stopka, umieścimy tutaj informacje o firmie
  oraz linkowanie do plików (np. informacje o budowanym
  projekcie i jego twórcach.</h3>
</footer>

```

Rys. 30.18. Kod HTML (strona index.html) – stopka

Zamiana index.html na index.php

Struktura HTML jest już gotowa, zmienimy więc nazwę pliku z index.html na index.php. Można tego dokonać w lewym panelu VSC. Najedź myszką na nazwę pliku index.html, kliknij w prawy przycisk, wówczas pojawi się okienko z dodatkowym menu. Wybierz: **Rename** – zmień nazwę pliku. To samo możesz uzyskać, gdy wciśniesz na klawiaturze F2.



Rys. 30.19. Visual Studio Code – Rename (zmiana nazwy pliku)

W ten sposób otrzymamy plik, który jest wyjściowym elementem tworzonego projektu. Wkrótce przemyślimy, które części pliku będą się powtarzały na poszczególnych podstronach. Jeżeli np. nagłówek będzie taki sam na stronie głównej i podstronach, wówczas przeniesiemy kod danej części strony do oddzielnego pliku (tutaj headerUser.php). Utworzymy także katalog templates, w którym umieścimy wszystkie pliki z powtarzającymi się elementami strony.

Stronę `index.php` podzielimy na poszczególne części podczas omawiania PHP. Na tym etapie zajmiemy się treścią strony: co poza menu i logowaniem chcielibyśmy przekazać użytkownikom portalu SOAW. Jednymi z najważniejszych elementów w budowie każdej aplikacji są prosty, bardzo przyjazny (dla użytkownika, a nie dla programisty) interfejs i zrozumiałe teksty, które precyzyjnie wyjaśnią miejsce, w którym internauta się obecnie znajduje. W naszym projekcie umieściliśmy już kilka tekstów, m.in. zawierające informacje o celu powstania portalu SOAW.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij, do czego służą:
 - Gimp,
 - Inkscape.
2. Wymień co najmniej trzy dodatkowe moduły VSC. Znajdź w internecie szczegółowe wyjaśnienia, do czego służą i jak mogą pomóc w tworzeniu kodu programistycznego.
3. Co to jest Viewport?
4. Czy wewnątrz pliku `index.html` będzie działał kod PHP? Odpowiedź uzasadnij.
5. Wymień co najmniej jeden powód podziału kodu strony PHP na: header, content i footer.
6. Chcesz jasno i zrozumiale poinformować klienta o tym, co może zyskać, zostając członkiem społeczności SOAW. Jakie informacje proponujesz umieścić na stronie głównej portalu? Wymyśl cztery wartościowe zdania, składające się przynajmniej z pięciu wyrazów.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przygotuj logo dla projektu SOAW:
 - naszkicuj na kartce znak graficzny i/lub tekstowy;
 - za pomocą INKSCAPE wykonaj logo, a następnie wyeksportuj go do formatu `.png` lub `.svg`;
 - w programie GIMP dopracuj stworzony plik.

31

SOAW – system logowania jako element serwisu ogłoszeniowego (logika i bezpieczeństwo aplikacji)

ZAGADNIENIA

- SOAW – krótkie podsumowanie dotychczasowych prac
- Dostęp do portalu – logika logowania
- Bezpieczeństwo aplikacji – planowanie systemu zabezpieczeń

31.1. SOAW – krótkie podsumowanie dotychczasowych prac

W ramach wspólnej nauki zdobyliśmy podstawową wiedzę na temat:

- HTML i CSS – zbudowany szkielet strony;
- XAMPP/MariaDB i język SQL – założyliśmy bazę 'soaw', w niej stworzyliśmy tabelę 'user' (w części opisującej typy oparte na relacyjnej bazie danych SQL).

Wspomniane technologie umożliwiły nam stworzenie zaledwie statycznej strony WWW i bazy danych, która w żaden sposób nie jest powiązana ze stroną. Stanowi to problem, ponieważ budując aplikację webową, chcielibyśmy skorzystać z zaplecza zawierającego dane. Aby to osiągnąć, sięgniemy po kolejny potężny język programowania w obszarze internetu – skryptowy PHP (rys. 31.1).

```
// Połączenie z bazą danych z użyciem funkcji mysqli()
$polaczenie = @new mysqli($host, $db_user, $db_password, $db_name);

if ($polaczenie->connect_errno!=0)
{
    echo "Error: ".$polaczenie->connect_errno;
}
else
{
    $login = $_POST['login'];
    $haslo = $_POST['haslo'];

    $login = htmlentities($login, ENT_QUOTES, "UTF-8");

    if ($rezultat = @$polaczenie->query(
        sprintf("SELECT * FROM uzytkownicy WHERE user='%s'",
            mysqli_real_escape_string($polaczenie,$login))))
    {
```

Rys. 31.1. PHP – nowoczesny i potężny język programowania

CIEKAWOSTKA

Kod języka programowania PHP przedstawiony na rys. 31.1 zawiera funkcjonalność dostępu do bazy danych z wykorzystaniem funkcji **mysqli()**. Jest to najprostszy sposób połączenia strony internetowej z zasobami i możliwościami bazy danych typu SQL. W zdecydowanej większości przypadków uczniowie poznają właśnie tę metodę. Aby zrozumieć, że o ile faktycznie funkcja **mysqli** jest dobra jako narzędzie edukacyjne, o tyle w przypadku rzeczywistych projektów komercyjnych raczej nie nadaje się do powszechnego wykorzystania, choćby ze względu na trudności w jej zabezpieczeniu. Komercyjną alternatywą (i to ciągle bezpłatną) mogą być np. rozwiązania wykorzystujące programowanie obiektowe:

- PDO (ang. *PHP Data Objects*), <https://www.php.net/manual/en/book.pdo.php>;
- DOCTRINE – jest to zestaw wielu bibliotek PHP do kompleksowej obsługi komunikacji z bazami danych, <https://www.doctrine-project.org/>.

Struktura języka PHP jest bardzo prosta – kod należy wpisać wewnątrz:

<?ph tutaj piszemy kod ?>.

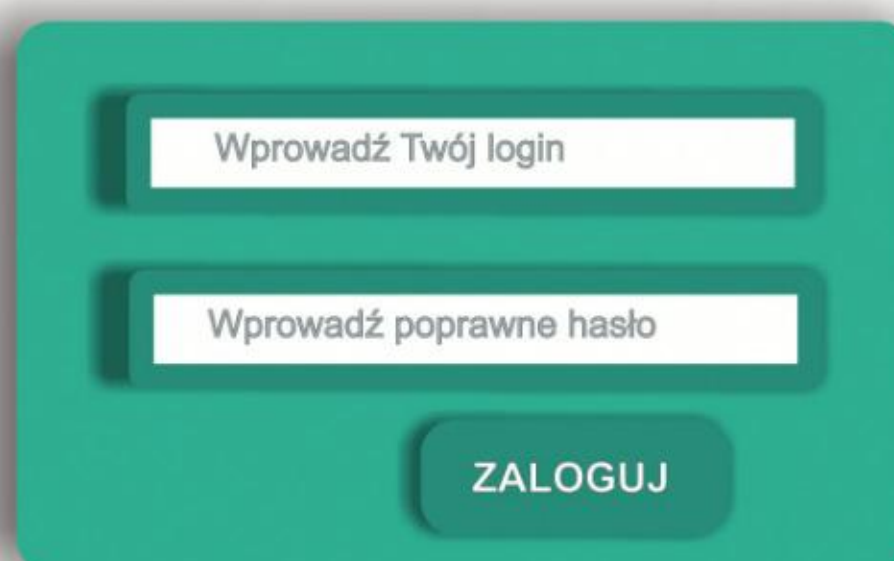
Istnieją także wersje skrócone znaków, np. **<?>**, ale nie są one zalecane. W celu pogłębienia wiedzy na temat tego języka warto zajrzeć na:

- <https://www.php.net/> – główna, oficjalna strona w internecie zawierająca dokumentację języka PHP,
- <https://pear.php.net/> – repozytorium dodatkowych rozszerzeń i aplikacji do PHP,
- <https://pecl.php.net/> – dodatkowe moduły PHP.

31.2. Dostęp do portalu – logika logowania

Programowanie opiera się na logice działania, którą można rozpisywać za pomocą kodu programistycznego w danym języku programowania albo tzw. pseudokodu. Na początek, zanim użyjemy PHP czy SQL, opiszemy, co chcemy osiągnąć, programując dostęp do bazy danych.

LOGOWANIE DO PORTALU



Panel logowania do aplikacji internetowej. Ma on tło w kolorze ciemnozielonym. W środku znajdują się dwa białe pola tekstowe: górne z napisem "Wprowadź Twój login" i dolne z napisem "Wprowadź poprawne hasło". Poniżej tych pól znajduje się przycisk z napisem "ZALOGUJ".

Rys. 31.2. Panel logowania do aplikacji internetowej

Logowanie do portalu SOAW powinno polegać na zaprogramowaniu zdarzeń:

- Użytkownik wpisuje do pola input właściwe dane. Zwykle są to:
 - unikatowy login – u nas jest nim adres e-mail (unikatowość w przypadku adresu e-mail jest spełniona, gdyż nie ma dwóch takich samych),
 - hasło.
- Wpisane do HTML-owych inputów dane są przechwycone przez PHP do zmiennych `$_POST` i podlegają wstępnej weryfikacji (walidacja i sanityzacja – oba pojęcia wyjaśnimy później).
- Po zweryfikowaniu wprowadzonych danych aplikacja łączy się z bazą i szuka ich odpowiedników w bazie danych.
- Porównanie przez aplikację danych zweryfikowanych z danymi otrzymanymi z bazy i podjęcie decyzji o odrzuceniu użytkownika lub o wpuszczeniu go do środka (skuteczne zalogowanie) albo decyzja o przyznaniu określonego rodzaju uprawnień (admin lub użytkownik – w naszym wypadku tego typu przypadków nie będziemy musieli programować).
- Po zalogowaniu internauty aplikacja powinna zapamiętać zalogowanego użytkownika i umożliwić mu przechodzenie między stronami dostępnymi dla danego rodzaju uprawnień bez potrzeby ponownego logowania.

Pora na praktyczne zaprogramowanie aplikacji SOAW z wykorzystaniem wymienionych zasobów: sql, html i css.

31.3. Bezpieczeństwo aplikacji – planowanie systemu zabezpieczeń

Bezpieczeństwo stron i aplikacji internetowych to bardzo ważny element pracy dla każdego programisty. Tworząc projekt webowy, trzeba myśleć kategoriami potencjalnych niebezpieczeństw związanych z internetem.



Rys. 31.3. Bezpieczeństwo aplikacji

Istnieje wiele sposobów zabezpieczeń, ale jeszcze więcej metod na złamanie tego, co tworzymy. Najczęściej spotykane metody to:

- włamanie jako zamierzone przestępstwo – haker stara się wykraść dane;
- „trening czyni mistrza” – wielu adeptów świata programowania w ten sposób testuje swoją wiedzę, „włamują się, bo mogą”, zwykle nie dlatego, żeby zrobić coś złego, ale bardziej dla zabawy i nauki (pamiętaj, że włamanie bez wiedzy właściciela, nawet „w dobrej wierze”, to ciągle przestępstwo);
- testowanie zabezpieczeń – płacone włamywanie się do projektów informatycznych celem sprawdzenia zabezpieczeń; uzgodnione dwustronną umową (i zapewne bardzo dużym wynagrodzeniem) nie jest przestępstwem.

Planowane zabezpieczenia

Poniżej podajemy kilka przykładowych zabezpieczeń, które jednak nie wyczerpują tematu, gdyż jest on bardzo obszerny:

- pisanie kodu programistycznego w zgodzie z „najlepszymi zasadami sztuki programistycznej”;
- mocne hasła administracyjne – używanie przez twórców aplikacji mocnych haseł, czyli m.in. zawierających co najmniej osiem znaków, w tym jedna duża litera, jedna mała litera, cyfra, przynajmniej dwa znaki specjalne;
- mocne hasła użytkowników – wymuszenie na użytkownikach używania mocnych haseł (nasi „już założeni” użytkownicy w bazie '**soaw**' mają ograniczenie jedynie co do liczby znaków);
- walidacja i sanitizacja – wyjaśnienie poniżej;
- hashowanie hasła – na etapie dodawania do bazy, a więc tzw. CRUD;
- csrf – na etapie dodawania do bazy, stosowane podczas programowania CRUD.

CIEKAWOSTKA

Duża część analityków uważa, że III wojna światowa trwa od wielu lat. Jej areną nie są jednak tradycyjnie rozumiane obszary, tylko sieć internetowa. Wrogie działania skupiają się na atakach hakerskich, które są nastawione m.in. na wykradanie tajemnic:

- rządowych,
- technologicznych,
- biznesowych,
- dotyczących danych osobowych.

Walczy tak naprawdę każdy z każdym. Poniżej znajdziesz adresy kilku przykładowych stron internetowych, które odnotowują „cyfrowe działania wojenne”:

- <https://threatmap.bitdefender.com/>,
- <https://threatmap.checkpoint.com/>,
- <https://cybermap.kaspersky.com/>,
- <https://threatmap.fortiguard.com/>,
- <https://www.fireeye.com/cyber-map/threat-map.html>.

Działanie	Instrukcja SQL	Metody HTTP
Create utworzenie / dodanie nowych informacji	INSERT	PUT / POST
Read odczytanie / wyświetlenie istniejących informacji	SELECT	GET
Update modyfikowanie / edycja informacji w bazie	UPDATE	POST / PUT / PATCH
Delete usuwanie istniejących informacji	DELETE	DELETE

Rys. 31.4. CRUD – Create Read Update Delete (podstawowe informacje)

Walidacja i sanityzacja

PHP dostarcza bardzo użyteczną i gotową do użycia funkcję `filter_var`. Przetworzy ona dostarczoną jej zmienną według określonych założeń (filtrów). Obsłuży ona:

- sanityzację – `filter_var($email, FILTER_SANITIZE_EMAIL)`,
- walidację – `filter_var($email, FILTER_VALIDATE_EMAIL)`.

Sanityzacja powoduje usunięcie wszystkich niedozwolonych znaków z adresu e-mail. Mamy więc do czynienia z weryfikacją danych wejściowych po stronie klienta. Oczyszczenie danych przed ich przesłaniem w kierunku bazy danych jest zawsze ważne (działanie sanizacyjne jest dobrym i bezpiecznym przyzwyczajeniem), ale szczególnego znaczenia nabiera, gdy chcemy umieścić dane w bazie projektu.

Na rys. 31.5 przedstawiono, jak może to wyglądać w kodzie.

```
// Filtr FILTER_SANITIZE_EMAIL usunie niedozwolone
znaki z adresu e-mail.
$emailAfterSanitize = filter_var($email,
FILTER_SANITIZE_EMAIL);
```

Rys. 31.5. Kod sanityzujący dane wejściowe

Walidacja to kolejny element bezpieczeństwa. Sprawdza, czy ciąg wprowadzonych znaków przypisany do zmiennej jest prawidłowym adresem e-mail.

Skoro chcemy otrzymać od użytkownika właściwy adres e-mail (czyli strukturę typu `programista@soaw.pl`), to aplikacja musi być na tyle inteligentna, aby go o tym poinformować. Ten element jest obsługiwany przez specjalną funkcję PHP i nazywa się walidacją (rys. 31.6).

```
// Aplikacja sprawdzi, czy po usunięciu
nieprawidłowych znaków, string jest taki sam jak
wprowadzony ciąg znaków w INPUT i czy tym samym
$email jest prawidłowym adresem email- sprawdza to
FILTER_VALIDATE_EMAIL
if ($emailAfterSanitize == $email && filter_var
($emailAfterSanitize, FILTER_VALIDATE_EMAIL)) {
    $testPassed = true;
    echo ' o.k. ';
} else {
    $testPassed = false;
    echo 'to nie jest prawidłowy adres email';
}
```

Rys. 31.6. Przykładowy kod walidujący (sprawdzający) poprawność wpisanego adresu e-mail

Jeżeli po sanityzacji zmienna `$email` jest w dalszym ciągu taka sama jak przed sanityzacją (`$emailAfterSanitize == $email`) i jest poprawnym adresem e-mail – kod `&& filter_var($emailAfterSanitize, FILTER_VALIDATE_EMAIL)`, oznacza to, że został wpisany poprawny string. Pozwala nam to np. ustawić tzw. flagę na wartość `true`, czyli program będzie mógł kontynuować swój bieg.

Dzięki temu np. napisana logika zaakceptuje wpisany login (u nas jest nim e-mail) i przejdzie do analizy zgodności hasła.

Gdyby się jednak okazało, że zmienna `$email` jest inna niż przed sanityzacją lub też wpisany string nie jest poprawnym adresem e-mail, to wykonany byłby kod:

```
$testPassed = false;
echo 'to nie jest prawidłowy adres email';
```

wyświetlony w nawiasie klamrowym po `else`.



SPRAWDŹ SWOJĄ WIEDZĘ

1. Jaka technologia (język programowania) pozwoli na połączenie bazy danych z front-endem?
2. Opisz logikę logowania do aplikacji SOAW.
3. Wymień planowane elementy systemu zabezpieczeń aplikacji SOAW.



SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Na koniec rozdziału 7. w części SPRAWDŹ SWOJE UMIEJĘTNOŚCI wykonałeś bazę danych, a w niej tabelę. Stwórz plik index.php. Napisz w nim kod w języku programowania PHP, który korzystając z wymienionej baz danych:
 - połączy się z nią (przez funkcję mysqli);
 - pobierze z niej wszystkie rekordy;
 - wyświetli wszystkie rekordy – każdy w kolejnym wierszu – na stronie internetowej.

32

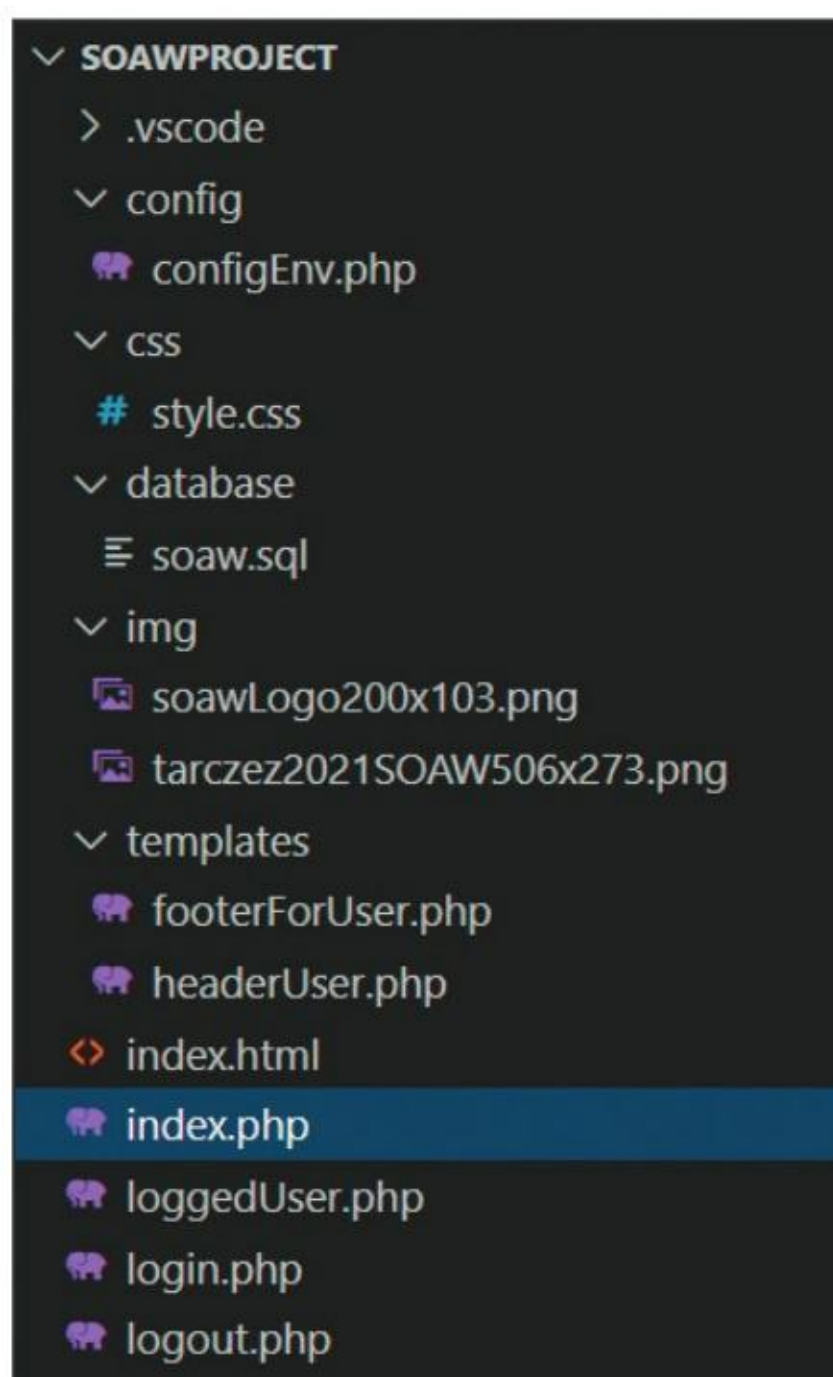
SOAW: przygotowanie systemu logowania – HTML i PHP

ZAGADNIENIA

- Katalog templates – początek zmian w HTML
- PHP – dołączenie zawartości pliku do wnętrza innego pliku
- Logowanie php – plik login.php
- Plik index.php a login.php – przepływ danych
- Testowanie PHP
- HTML i php – adres e-mail (logika 'and')
- Skrypty PHP wewnątrz HTML

32.1. Katalog templates – początek zmian w HTML

Rozpoczniemy tę część podręcznika od stworzenia w głównym katalogu projektu 'soaw' nowego katalogu 'templates'. Przeniesiemy do niego wszystkie pliki z powtarzającymi się elementami strony (header i footer).



Rys. 32.1. Struktura plików projektu SOAW – widok w VSC

Jako pierwszy stworzymy we wskazanym folderze plik `headerUser.php`, do którego przeniesiemy kod odpowiadający za nagłówek strony. Nagłówek będzie taki sam na stronie głównej projektu (dla osób niezalogowanych) i podstronach (dla zalogowanych). Łatwiej nim będzie można zarządzać, jeżeli znajdzie się tylko w jednym miejscu. Wyobraź sobie, że twój klient będzie chciał dodać element w nagłówku, np. poziomy baner reklamowy, informujący o promocyjnych cenach usług SOAW. Jeżeli każda strona projektu będzie zawierała, niezależnie od innych, kod obszaru header, to zmiany trzeba by wykonać na każdej podstronie oddzielnie.

Jeżeli przeniesiemy kod danej części strony do oddzielnego pliku – tutaj `headerUser.php`, to gdyby była potrzebna zmiana w tym obszarze strony, wystarczy zaktualizować tylko jeden plik. W naszym projekcie wytniemy z pliku `index.php` drobny kawałek kodu (rys. 32.2) i i wkleimy go do nowo utworzonego pliku `headerUser.php` (wewnątrz katalogu `'templates'`).

```
<!-- HEADER - nagłówek strony -->
<header>

    <!-- znacznik nav zawiera elementy nawigacyjne strony -->
    <nav class="menuNawigacyjne grid-container">

        <!-- Sekcja logo -->
        

        <!-- HTML Unordered Lists -->
        <ul class="ulNawigacyjne item2">
            <li class="active"><a href="#home">Home</a></li>
            <li><a href="#link1">Link 1</a></li>
            <li><a href="#link2">Link 2</a></li>
            <li><a href="#link3">Link 3</a></li>
            <li><a href="#link4">Link 4</a></li>
        </ul>

    </nav>

</header>
```

Rys. 32.2. Wycięcie i przeniesienie kodu wspólnego dla wielu stron do pliku `headerUser.php`

Taką samą zmianę (wycięcie i wklejenie do nowego pliku) wykonamy dla kolejnego fragmentu kodu (rys. 32.3).

```
<footer id="site-footer" class="footerStyle">
  <div class="footerLeft">
    <h2 class="font-italic">SOAW</h2>
    <h2 class="display-5 font-italic">www.adresSOAW.pl</h2>
    <h3>Serwis Ogłoszeniowy Aplikacja Webowa</h3>
  </div>
  <div class="footerRight">
    <h3><span style='font-size:20px;'>&#x260F; GSM 500 xxx
    yyy</span></h3>
    <h3><span id="footerSpanFirma" style='font-size:20px;'>&
    #9993; biuro@adresSOAW.pl</span></h3>
    <h3 class="footer-copyright">&copy;
    2021 <a href="https://adresSOAW.pl/">powered by
    SOAW</a>
  </h3>
  </div>
</footer>
```

Rys. 32.3. Wycięcie i przeniesienie kodu wspólnego dla wielu stron do pliku footerForUser.php

Przeniesiemy go do nowego pliku footerForUser.php, który znajduje się wewnątrz katalogu ‘templates’.

Wykonując dwie wymienione operacje, uzyskaliśmy modułową budowę strony, niejako dostosowując się do jednej z zasad programowania: „dziel i rządź”. Jeżeli masz duży problem (zagadnienie, zadanie, funkcję itp.), to podziel go na mniejsze kawałki (jak najmniejsze), a następnie połącz wszystkie w jeden element. W ten modułowy sposób są zbudowane największe ogólnościowe portale, ale także małe i średnie projekty informatyczne.

32.2. PHP – dołączenie zawartości jednego pliku do wnętrza innego pliku

PHP ma setki przydatnych funkcji, w tym aż cztery na dołączanie zawartości plików do wnętrza innych plików. Skorzystamy z jednej, bodaj najczęściej stosowanej – **include_once**.

Najpierw użyjemy jej we właściwym miejscu dla nagłówka pliku **index.php** (rys. 32.4).


```
<!-- Import - dołączenie funkcjonalności - pliku  
zawierającego FOOTER -->  
<?php include_once 'templates/footerForUser.php';  
?>
```

Rys. 32.4. Dołączenie zawartości pliku headerUser.php do pliku index.php

A następnie także dla stopki (rys. 32.5).

```
<!-- Import - dołączenie funkcjonalności - pliku  
zawierającego FOOTER -->  
<?php include_once 'templates/footerForUser.php';  
?>
```

Rys. 32.5. Dołączenie zawartości pliku footerForUser.php do pliku index.php

Polecenie `include_once` zdecydowanie musisz zapamiętać – będziemy wykorzystywali je również w dalszej części projektu. Informacje o różnicach między tym poleceniem a `include_once`, `require_once`, `require` znajdziesz w materiałach źródłowych PHP:

- <https://www.php.net/manual/en/function.require-once.php>,
- <https://www.php.net/manual/en/function.require.php>,
- <https://www.php.net/manual/en/function.include-once.php>,
- <https://www.php.net/manual/en/function.include.php>.

32.3. Logowanie php – plik login.php

Jeżeli w twoim projekcie nie ma pliku `login.php`, to koniecznie musisz go utworzyć. Zrób to w katalogu głównym projektu SOAW. Zbudujemy go z dwóch podstawowych części:

- kod PHP – będziemy pisali go krok po kroku, zaczynając od góry strony `login.php`, a pod nim, czyli pod kodem PHP, znajdzie się:
- kod HTML, w którym także będziemy osadzali PHP.

Co do kodu HTML, najlepiej skopiować całą zawartość kodu HTML z pliku `index.php`, czyli od rozpoczynającego strukturę strony HTML znacznika `<!DOCTYPE html>` do kończącego stronę WWW znacznika `</html>`. Pamiętaj, że wygląd strony jest przykładowy. Warto popracować nad atrakcyjnym wyglądem, tak aby był to oryginalny design.

Na rys. 32.6–32.9 przedstawiono wygląd kodu strony `login.php`.


```

<?php
session_start();

?>

<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
    <title>Serwis Ogłoszeniowy Aplikacja Webowa</title>
    <link rel="stylesheet" href="css/style.css">
</head>

```

Rys. 32.6. Kod strony login.php – góra strony (miejsce na wpisywanie kodów PHP i początek struktury HTML 5)

Jest to stan początkowy. Jeżeli pojawiły się na tym etapie jakieś błędy, to w kolejnych krokach – podczas programowania PHP będą się one nawarstwiały i zapewne twój projekt nie będzie działał poprawnie.

```

<body>
    <!-- Dołączenie nagłówka, czyli strony headerUser.php -->
    <?php include 'templates/headerUser.php'; ?>

    <!-- CONTENT - zawartość strony -->
    <div id="contentMain" class="contentMain">
        <!-- aside powinien zawierać tekst dodatkowy -->
        <form action="login.php" method="POST"
            class="loginPodHeader">
            Login: <input type="text" placeholder="Your nick"
                name="email">
            <br> Hasło: <input type="text" placeholder="Twoje
                hasło" name="password">

            <br> <a href="rejestracja/createUser.php ?
                ">Rejestracja</a>
            &nbsp;
            <input class="inputLogIn" type="submit" value="Zaloguj"
            />
        </form>

```

Rys. 32.7. Kod strony login.php – początek sekcji **<body>** HTML 5, dołączenie nagłówka i formularz do zalogowania


```

<!-- article powinien zawierać główny tekst schematyczny
danej strony -->
<article>
<h1 class="h1Style"><span>SOAW</span> <br>Serwis
Ogłoszeniowy Aplikacja Webowa</h1>
    <hr>
    <div class="soawContent">
        <div class="soawContentLeft">
            
        </div>
        <div class="soawContentRight">
            <h2 class="h2Style">Jak to działa?</h2>
            <h2 class="h2Style">Założ konto! Bezpłatna
            rejestracja!</h2>
        </div>
    </div>
</article>

<!-- aside powinien zawierać tekst dodatkowy -->
<aside>
    <hr>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing
elit. Modi eum illum deleniti, voluptate sunt, enim
repudiandae exercitationem dolore corporis dolores
possimus dolorum veritatis perspiciatis ullam magni
assumenda eos ad obcaecati.</p>
</aside>
</div>

```

Rys. 32.8. Kod strony login.php – sekcja z zawartością opisową strony

```

<!-- FOOTER - stopka strony -->
<?php include 'templates/footerForUser.php';
?>
</body>

</html>

```

Rys. 32.9. Kod strony login.php (dół strony) – dołączenie stopki i zakończenie kodów HTML 5

32.4. Plik index.php a login.php – przepływ danych

Budowę systemu logowania do portalu SOAW zaczniemy od strony głównej, na której w kodzie umieściliśmy formularz umożliwiający szybkie logowanie. Jeżeli użytkownik kliknie na stronie index.php w **Zaloguj**, wówczas skrypt kieruje otrzymane dane na stronę – login.php, na której nastąpi weryfikacja poprawności wprowadzonych danych.

SOAW Serwis Ogłoszeniowy Aplikacja Webowa



Jak to działa?

Zalóż konto! Bezpłatna rejestracja!

Dodaj za darmo ogłoszenie!

Login:

Hasło:

[Rejestracja](#)

Rys. 32.10. Widok logowania na pierwszej stronie aplikacji SOAW

Na stronie login.php aplikacja musi rozpocząć procedurę sprawdzenia poprawności wprowadzonych danych. Oczywiście zostanie ona uruchomiona po wpisaniu danych przez użytkownika i kliknięciu **Zaloguj**. Jeżeli termin „Login” na przycisku do logowania nie będzie ci odpowiadał, zawsze możesz go zmienić na inny, np. „Zaloguj”.

Na stronę login mają zostać przeniesione dane wprowadzone przez użytkownika. Jeżeli będą one poprawne, to użytkownik przejdzie na specjalną stronę, którą dopiero stworzymy. Dostęp do niej będą mieli wyłącznie użytkownicy zalogowani. Jeżeli wprowadzone dane będą niepoprawne, wówczas wyświetli się zawartość HTML-owa strony, zawierająca pola do ponownego wprowadzenia loginu i hasła. Umieszczone tam będą także informacje dla użytkownika, których celem będzie pomoc we wprowadzeniu

właściwych hasła i loginu (o ile użytkownik takie dane ma). Jeżeli nimi nie dysponuje, czyli nie zarejestrował się jeszcze w aplikacji, wówczas może kliknąć przycisk „Rejestracja”. Przeniesie się wówczas na stronę, na której będzie mógł założyć bezpłatne konto do aplikacji SOAW.

32.5. Testowanie PHP

Przechodzimy do pracy nad zawartością pliku login.php, do którego są wysyłane dane pobierane z formularza na stronie głównej SOAW.

W programowaniu ważne jest testowanie drobnych elementów. Przede wszystkim chodzi o sprawdzanie, czy od początku pracy z plikiem nasz ekosystem technologiczny działa. Przykładem może być sprawdzenie istnienia zmiennej email w nowo stworzonym pliku:

```
<?php
if (isset($_POST['email'])) {
    echo ,Istnieje zmienna: ,. $_POST[,email'];
}
else {
    echo ,Brak zmiennej';
}
?>
```

Ustaw w formularzu – action = „#” – spójrz na kod:

```
<form action="#" method="POST" class="underHeaderInputs">
    Login: <input type="text" placeholder="Twój email"
    name="email"><br>
    <br> Hasło: <input type="text" placeholder="Twoje hasło"
    name="password"><br>
    <br> <a href="#">Rejestracja</a>
    &nbsp;
    <input class="inputLogIn" type="submit" value="Zaloguj"/>
</form>
```

Rys. 32.11. Kod formularza ustawiony na niewysyłanie danych z pól input poza plik index.php

Powyższy kod spowoduje, że po wpisaniu danych i naciśnięciu przycisku **Zaloguj** dane nie zostaną przesłane dalej (poza plik index.php), tylko „obsłużone” wewnątrz tego pliku. Uruchom w przeglądarce adres localhost/soaw (czyli stronę index.php). Teraz dzięki kodowi z następnej strony (rys. 32.12) na stronie wyświetli się napis „Brak zmiennej”.


```
<?php
if (isset($_POST['email'])) {
    echo 'Istnieje zmienna: ' . $_POST['email'];
} else {
    echo 'Brak zmiennej';
}
?>
```

Rys. 32.12. Kod PHP wyświetlający napis na stronie „Istnieje zmienna”, ale dopiero po kliknięciu **Zaloguj**



Rys. 32.13. Napis wyświetlany na stronie po wejściu na nią

Jeżeli masz kod wpisany zgodnie z instrukcjami podanymi w podręczniku, to ujrzysz pod nawigacją (widok strony index.php) dwa pola umożliwiające wpisanie loginu i hasła. Wpisz w te dwie pozycje dane, a następnie kliknij **Zaloguj**. W pole Login wpisano: „Wpisany Email”, a w Hasło – „Cokolwiek”. Po wprowadzeniu danych do inputów kliknij **Zaloguj**.

Jeżeli wszystko wcześniej zostało zrobione poprawnie, to na stronie index.php funkcja echo wypisze tekst: „Istnieje zmienna: Wpisany Email” – spójrz na rys. 32.14.



Rys. 32.14. Napis wyświetlany na stronie index.php po wpisaniu danych do inputów i kliknięciu **Zaloguj**

Wykonaliśmy jeden z najprostszych sposobów przetestowania tworzonej funkcjonalności aplikacji. Wykorzystaliśmy polecenie **echo** . . , które bardzo często nazywa się w PHP funkcją. W rzeczywistości funkcją wcale nie jest, choć działa podobnie.

Echo:

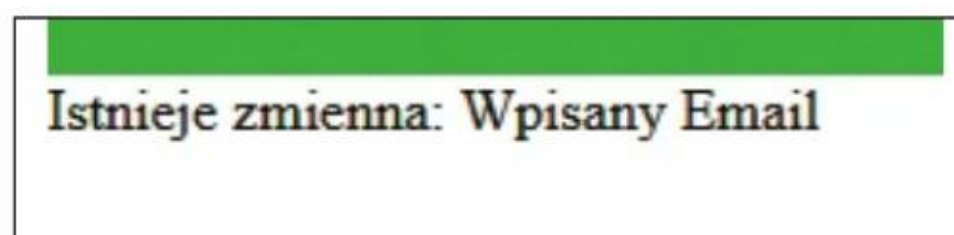
- jest konstrukcją języka PHP (teraz już wiadomo, dlaczego nie trzeba wpisywać nawiasów po niej);
- wyświetla elementy wpisane po spacji aż do napotkanego znaku ; (średnik – jest końcem polecenia w języku PHP);
- ma także składnię skrótów PHP, czyli oznacza, że można od razu – po otwierającym tagu – wstawić znak równości: **<?=\$zmienna?>**.

Wiele przydatnych informacji o konstrukcji językowej echo znajdziesz na <https://www.php.net/manual/en/function.echo.php>.

Wykonaliśmy test i wiemy już, że PHP działa właściwie, więc możemy przystąpić do dalszej pracy. W kolejnym rozdziale zajmiemy się właściwym programowaniem w PHP. Będziemy także tworzyć skrypty wewnątrz znaczników HTML-owych.

32.6. HTML i PHP – adres e-mail, logika ‘and’

Nasza aplikacja ma już pola do wpisania danych: loginu i hasła. Zajmiemy się ich obsługą programistyczną. Zaczniemy od analizy loginu, którym jest w SOAW adres e-mail. Każdy adres e-mail ma określoną strukturę i wystarczy zadbać o to, żeby aplikacja sprawdziła, czy wprowadzony ciąg tekstowy jest zgodny ze standardem e-mail, czyli czy jest poprawnym adresem e-mail. Można to zrobić za pomocą jednej z funkcji wbudowanych języka PHP. Zanim dojdziemy do właściwej funkcji, rozszerzymy test, który wcześniej tworzyliśmy. Osoba, która dokładnie przetestowała zapisaną logikę testującą, zauważyła, że jeżeli nic nie wpisaliśmy w pole Login, to i tak pojawiał się na stronie napis „Istnieje zmienna: Wpisany Email” (rys. 32.15).



Rys. 32.15. Napis wyświetlany na stronie index.php po wpisaniu danych do inputów i kliknięciu **Zaloguj**

Musimy sprawić, aby dopiero wpisanie co najmniej jednego znaku spowodowało uruchomienie logiki. Przekładając powyższy tok rozumowania na kod PHP, na początek zapiszmy prosty warunek logiczny w pliku index.php, czyli:

- OR – lub – znak **||** ,
- AND – i – znak **&&**.


```
if (isset($_POST['email']) && strlen($_POST['email'])  
> 0)  
{  
    echo "Ten napis pokaże się, gdy zostanie  
    wprowadzony co najmniej jeden znak w polu input -  
    login oraz zostanie kliknięty przycisk Zaloguj ";  
}  
else  
{  
    echo 'Brak wpisanego przynajmniej jednego znaku -  
    wprowadź poprawny login, którym musi być Twój  
    adres email zarejestrowany w naszej bazie';  
}
```

Rys. 32.16. Kod PHP, który uruchomi kod w części **if { }**, gdy zostanie wprowadzony co najmniej jeden znak w polu input – login i zostanie kliknięty przycisk **Zaloguj**

Zwróć uwagę na umieszczoną w kodzie funkcję **strlen()**, która zwróci długość znaków w ciągu znaków, wpisanych jako jej argument, czyli w nawiasach. Uwaga – funkcja **strlen()** bierze pod uwagę także puste znaki i dolicza je do sumy. Oznacza to, że trzeba pomyśleć nad takim fragmentem kodu, który sam będzie usuwał puste spacje, co zabezpieczy nas przed klientami, którzy nie są zbyt uważni podczas wpisywania znaków.

Wróćmy do zapisanej logiki – wskazuje ona, że gdy zmienna **email** będzie ustanowiona i (bardzo ważne: „i”, a nie „lub”) długość tej zmiennej jest większa od zera (popatrz na kawałek kodu: **strlen(\$_POST['email'])**), czyli użytkownik wpisał coś do inputa i spróbował to wysłać, wówczas będzie wykonany kod po **if**. Jeżeli oba wskazane warunki równocześnie nie są spełnione, wówczas zostanie wykonany kod po **else**, czyli pojawi się napis:

Brak wpisanego przynajmniej jednego znaku – wprowadź poprawny login, którym musi być Twój adres email zarejestrowany w naszej bazie.

Ten element rozbudujemy za pomocą PHP w kolejnym dziale. Wykonamy pobieranie danych z inputów, by sprawdzić ich poprawność i zgodność z odpowiednikami zawartymi w bazie danych.

32.7. Skrypty PHP wewnątrz HTML

HTML i PHP są ze sobą powiązane. Już wkrótce będziemy używali skryptów PHP zaczynających się od `<?php` i kończących się na `?>`, ale wpisywanych wewnątrz HTML. Wykorzystamy je również, jak się przekonasz, zamiast HTML lub jego niektórych elementów. Spójrz na przykład kodu na rys. 32.17.

```
<a
  href="<?php echo ($_SESSION
    ['HTTP_SERVER_DOMAIN'] . 'rejestracja/
    createUser.php' == 'http://' . $_SERVER
    ['HTTP_HOST'] . $_SERVER['PHP_SELF'] ?
    "#" : $_SESSION['HTTP_SERVER_DOMAIN'] .
    'rejestracja/createUser.php') ?
  ">Rejestracja</a>
&nbsp;
<input class="inputLogIn" type="submit"
value="Zaloguj" />
```

Rys. 32.17. Kod PHP wewnątrz HTML

Powyższy skrypt PHP znajduje się wewnątrz atrybutu HTML – **href**. Wykorzystując uproszczoną wersję instrukcji warunkowej, czyli operator trójkowy, w zależności od adresu, pod którym zostanie wyświetlony, pokaże aktywne linkowanie do adresu **'rejestracja/createUser.php'** lub nie pozwoli nigdzie dalej przejść – opcja **'#'**.

Przeanalizuj podane kody, a potem dokonaj w nich kilku zmian. Zobacz, co się wtedy wydarzy. Pamiętaj! Aby zostać programistą, trzeba pisać, pisać i jeszcze raz pisać kody.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wy tłumacz, dlaczego pliki określonego rodzaju, np. szablony elementów strony, są przenoszone do jednorodnych tematycznie katalogów.
2. Wyjaśnij, jak działa `include_once`.
3. Opisz poszczególne części kodu zawartego wewnątrz pliku `login.html`.
4. Jaki znasz najprostszy sposób na testowanie poprawnego działania PHP?
5. Wyjaśnij przepływ danych między plikami `index.php` a `login.php`.
6. Zasada „dziel i rządz” jest często wykorzystywana w tworzeniu projektów informatycznych. Opisz korzyści, jakie mogą powstać w wyniku jej zastosowania w tworzeniu projektów webowych.
7. Wy tłumacz różnicę między dwoma pojęciami z logiki: „and” i „lub”. Jakich znaków w programowaniu używa się do zaprogramowania logiki zgodnej z wymienionymi pojęciami?
8. Omów zasady pisania skryptów PHP wewnątrz HTML.

33

SOAW: programowanie systemu logowania – PHP

ZAGADNIENIA

- Uruchomienie sesji PHP
- Flaga w programowaniu
- Logowanie PHP – sanityzacja loginu
- Walidacja loginu w praktyce
- Walidacja hasła

33.1. Uruchomienie sesji PHP

W PHP jest sposób na przechowywanie i przekazywanie informacji (w zmiennych), które mają być używane na wielu stronach WWW w ramach tej samej domeny internetowej. Nazywa się to „**sesją PHP**”. W naszym przypadku uruchomiona sesja umożliwi zarządzanie danymi aplikacji wpisanymi przez użytkownika do pól input. Po przekazaniu danych z tablic \$_POST będziemy mogli zapisać je w pliku login.php do zmiennej sesyjnej, czyli określanej jako:

```
$_SESSION['nazwa_danej_zmiennej_sesyjnej'];
```

Użycie `$_SESSION[ ]` także na innych stronach aplikacji spowoduje, że będziemy mogli posługiwać się danymi niezbędnymi do zbudowania skutecznego systemu logowania.

```
index.php > ...
1      <?php
2
3      session_start();
4
```

Rys. 33.1. Kod uruchamiający sesję PHP

Funkcja `session_start()`

Dane sesyjne mogą zostać zapamiętane i być udostępniane dla wielu stron .php. Jest jeden warunek zapamiętywania informacji w ramach sesji – wpisanie na początku pliku .php, tuż po znaczniku rozpoczynającym plik `<?php`, wywołania sesji, czyli wpisanie funkcji `session_start()`. Każda taka strona, dla której uruchomimy mechanizm sesji, zapamięta ustawienia, np. logowanie, ustawianie i przekazywanie zmiennych.

SESJA PHP

zmienna sesyjna `$_SESSION['email']` – będzie dostępna na stronach:

`index.php`, `login.php`, `rejestracja.php`

gdyż została na nich uruchomiona sesja PHP `session_start();`,

ale nie będzie dostępna wskazana zmienna na stronie

`o-nas.php` – brak uruchomionej sesji PHP

www.soaw.com.pl/login.php

`session_start();`

www.soaw.com.pl/index.php

`session_start();`

www.soaw.com.pl/o-nas.php

`$_SESSION['email'];`

www.soaw.com.pl/rejestracja.php

`session_start();`

Rys. 33.2. Sesja w języku PHP

W języku PHP zmienne sesji mogą być ustawiane za pomocą tzw. zmiennej globalnej: `$_SESSION`. Możesz o niej więcej poczytać np. pod adresem <https://www.php.net/manual/en/reserved.variables.session.php>.

33.2. Flaga w programowaniu

Kod naszego projektu będziemy tworzyli na sposób strukturalny, wykorzystując dostępne w PHP instrukcje warunkowe. Niezmiernie istotnym elementem będzie zastosowanie **flagi**, która będzie oznaczała po prostu pewną zmienną. Przypiszemy do niej za każdym razem jedną z dwóch wartości logicznych: `true` (prawda) lub `false` (fałsz). Wykonując kolejne kroki kodu wykorzystujące np. instrukcje `if`, ustawimy flagę na `true`, gdy program

może pójść dalej, lub na false, gdy trzeba się zatrzymać z realizacją kolejnych etapów. Przykładowo: jeżeli użytkownik wpisze niepoprawny adres e-mail (np. zabraknie w nim znaku @), wówczas flagę ustawimy na false, co zastopuje bieg programu i poinformuje o błędzie, który trzeba będzie wyświetlić użytkownikowi na stronie projektu.

W kodzie aplikacji SOAW ustawimy flagę, czyli po prostu zmienną, którą nazwiemy `$flagaTest`. W naszym przypadku będzie ona:

- równa true, gdy nasz kod przejdzie pozytywnie test;
- równa false, gdy nasz kod nie przejdzie pozytywnie testu.

```
if (isset($_POST['email']) && strlen($_POST['email']) > 0)
{
    // echo 'Login wprowadzono poprawnie';
    //Udana walidacja? Załóżmy, że tak, czyli ustalmy flagę
    $flagaTest = true;
}
```

Rys. 33.3. Wykorzystanie flagi w programowaniu

Kolejne fragmenty kodu powinieneś wpisywać pod liniijką z kodem ustawiającym zmienną `$flagaTest`. Sugeruję przechwycić zmienną email, czyli stworzymy zmienną `$email`, której przypiszemy wartość, wykorzystując zmienną `$_POST['email']`, która jest tzw. literałem wpisanym przez użytkownika do pola **Login**.

```
// Ustaw zmienną $email
$email = trim($_POST['email']);
```

Rys. 33.4. Przechwycenie zmiennej email – funkcja `trim()`

Znak `$` postawiony przed nazwą (bez jakiegokolwiek spacji) wskazuje nam, że jest to nazwa zmiennej. Zatem przechwyciliśmy zmienną `$_POST['email']` do zmiennej PHP `$email`.

33.3. Logowanie PHP – sanitizacja loginu

Wróćmy do loginu, którym jest poprawnie wpisany adres e-mail. Stwórzmy kod wykonujący sprawdzanie poprawności wpisanego adresu e-mail, korzystając z pojęć sanitizacja i walidacja. Pamiętajmy, że sanitizacja usuwa wszystkie niedozwolone znaki z adresu e-mail. Fachowo jest określana jako „Weryfikacja danych wejściowych po stronie klienta”.

W kodzie projektu SOAW wygląda to tak, jak przedstawiono na rys. 33.5.

```
// Funkcja filter_var () filtruje zmienną z  
określonym filtrem.  
// Sprawdź czy wprowadzony tekst jest poprawnym  
adresem email.  
// Filtrowanie FILTER_SANITIZE_EMAIL usuwa wszystkie  
niedozwolone znaki z adresu e-mail.  
  
$emailAfterSanitize = filter_var($email,  
FILTER_SANITIZE_EMAIL);
```

Rys. 33.5. Przechwycenie zmiennej email – sanityzacja

33.4. Walidacja loginu (adresu e-mail)

Kod aplikacji dokonujący walidacji loginu sprawdzi, czy ciąg wprowadzonych przez użytkownika znaków przypisany do zmiennej `$email` jest prawidłowym adresem e-mail, tzn. czy ma budowę charakterystyczną dla adresu e-mail.

```
// WALIDACJA EMAIL  
  
if ($emailAfterSanitize == $email && filter_var  
($emailAfterSanitize, FILTER_VALIDATE_EMAIL)) {  
    $flagaTest = true;  
    // echo ' o.k. ' ;  
}  
else  
{  
    $flagaTest = false;  
  
    // Ustawienie zmiennej sesyjnej, która po  
    wyświetleniu we właściwym miejscu poinformuje  
    użytkownika o zaistniałym problemie  
    $_SESSION['e_emailLogin'] = "Login - musi to być  
    poprawny adres email!";  
}
```

Rys. 33.6. Walidacja adresu e-mail (loginu)

Zapisany kod powoduje, że jeżeli po sanityzacji zmienna `$email` jest ciągle taka sama, jak była przed sanityzacją (`$emailAfterSanitize == $email`), i jest poprawnym adresem e-mail `&& filter_var($emailAfterSanitize, FILTER_VALIDATE_EMAIL)`, to zapewne oznacza, że użytkownik wpisał poprawny string. Jest to pozytywna wiadomość, przeszliśmy bowiem kolejny test ustawiający flagę na wartość `true` (rys. 33.7).

```
// Walidacja zakończyła się sukcesem
$flagaTest = true;
```

Rys. 33.7. Test zaliczony – flaga ustawiana na `true`

Logika kodu zaakceptuje login i przejdzie do kolejnego kroku, jakim będzie zgodność hasła.

Trzeba jednak pamiętać, że jeżeli zmienna `$email` będzie inna niż przed sanityzacją bądź wpisany przez użytkownika ciąg nie będzie poprawnym adresem e-mail, to zostanie wykonany kod, który występuje w nawiasie klamrowym po `else`. Powinna się wówczas wyświetlić stosowna informacja o błędzie. Obsłuży to zmienna sesyjna `$_SESSION['e_emailLogin']`, która się wyświetli, gdy użytkownik wpisze ciąg znaków niebędący poprawnym adresem e-mail, czyli ciąg, który nie spełnia walidacji dla `FILTER_VALIDATE_EMAIL`. Aby wyświetlić zmienną, umieścimy ją w kodzie HTML wewnątrz formularza, pod polem dla wpisywania loginu. Używając kodu PHP wewnątrz HTML, stworzymy logikę PHP powodującą jej warunkowe wyświetlenie – gdy użytkownik wpisze i będzie próbował wysłać nieprawidłowy adres e-mail (rys. 33.8).

```
<?php
    if (isset($_SESSION['e_email']))
    {
        echo $_SESSION['e_email'];
    }
?>
```

Rys. 33.8. Logika warunkowa – kod obsługujący pokazanie zmiennej będącej informacją dla klienta

W tym miejscu trzeba dodać, że wewnątrz instrukcji warunkowej `if( )` po klauzuli `else{ }` można rozważyć jeszcze dodanie flagi:

```
$testPassed = false;
```

która nie pozwoli na wykonanie zapytania do bazy SOAW.

! UWAGA

Jeżeli oba warunki są spełnione równocześnie:

```
if
($emailAfterSanitize == $email &&
filter_var($emailAfterSanitize, FILTER_VALIDATE_EMAIL))
{
```

to wykonaj dalej kod:

```
    $testPassed = true;
    // echo , o.k. , ;
}
```

a gdy nie został spełniony choćby jeden z wymienionych warunków:

```
else
{
```

to wykonaj kod:

```
    $_SESSION[,'e_emailLogin'] = „Login - musi to być poprawny
adres email!!!”;
    $testPassed = false;
}
```

33.5. Walidacja hasła

Podstawowym założeniem tej części podręcznika jest poprawna walidacja wpisanego przez użytkownika Loginu. Ta część aplikacji, która będzie opisana poniżej, już wie, że adres e-mail spełnia założone standardy.

Od strony kodowej mamy już pierwszą daną obsłużoną we właściwy sposób. Teraz pora na drugą pozycję, którą wpisuje użytkownik – hasło. O ile w przypadku wpisywania adresu e-mail dokonaliśmy walidacji i sanityzacji, o tyle w przypadku hasła nie ma konieczności wykonywania obu czynności. Wystarczy walidacja.

Najpierw musimy zapisać walidowane hasło w postaci zmiennej z przypisaniem, przechwycone z odpowiedniego pola formularza (rys. 33.9).

```
// WALIDACJA HASŁA
// Funkcja PHP trim() usunie zbędne spacje
// z początku i z końca wpisanego ciągu znaków

$password = trim($_POST['password']);
```

Rys. 33.9. Przechwycenie hasła z pola input – użycie funkcji `trim()`

W kodzie znajduje się funkcja PHP `trim()`, która zajmie się problemem pustych spacji. Zakładamy, że w środku hasła użytkownik nie poda spacji. Trzeba pamiętać, że na początku i (szczególnie) na końcu może się zdarzyć takie przypadkowe kliknięcie spacji.

Funkcja PHP `trim()` potrafi usunąć zbędne spacje z początku i z końca wpisanego (jako argument wewnątrz nawiasów) ciągu znaków.

Wcześniej w naszym kodzie funkcji `trim()` użyliśmy podczas oprogramowywania przechwycenia loginu do zmiennej.

Hasło pozyskane z HTML-owego pola input jest **stringiem** (ciągami znaków). Zrobimy pewne startowe założenie co do długości hasła. Oczywiście łatwiej i szybciej odgadnąć jedną literę bądź cyfrę niż 10 dowolnych znaków. Pominiemy kwestie siły hasła (a tym bardziej „siły znaku” + i znaku C), czyli kombinacji różnego rodzaju znaków.

```
// Logika PHP sprawdzająca czy wpisany ciąg znaków
// posiada odpowiednią długość
if (
    (strlen($password) < 8) ||
    (strlen($password) > 12)
)
{
    echo 'POZA ZAKRESEM';
    $flagaTest = false;

    // Ustawimy tutaj (nieco później) -
    // zmienną, która będzie komunikatem
    // ostrzegawczo-informacyjnym
}
```

Rys. 33.10. Kod PHP – logika sprawdzająca, czy wpisany ciąg znaków ma odpowiednią długość

W budowanej aplikacji przyjmiemy, że jako hasło użytkownik może wpisać ciąg dowolnych znaków z ograniczeniem, że hasło musi mieć co najmniej 8 i maksymalnie 12 znaków. Oczywiście we własnym kodzie można dokonać dowolnych zmian, np. co najmniej 5 i maksymalnie 15 znaków. Mechanizm działania kodu PHP pozostanie taki sam. Nie zaprogramujemy wymuszania na uczestniku minimalnej liczby znaków. To nie jest niezbędne, ale w ramach rozwoju projektu można dodać stosowną funkcjonalność podczas dalszych prac rozwojowych.

Powyższe założenia wskazują, że jeżeli wpisane hasło będzie zawierać nie więcej niż 12 i nie mniej niż 8 znaków, to taki ciąg znaków zostanie zaakceptowany z punktu widzenia liczby niezbędnych znaków. Nie oznacza to, że podane hasło jest zgodne z hasłem, które jest zapisane w bazie. Nastąpi jedynie sprawdzenie kryterium właściwej długości hasła.

Po sprawdzeniu poprawności hasła (a wcześniej także loginu) planujemy przejście:

- na poziom dla użytkownika zalogowanego lub
- jeżeli nie ma stosownych danych – do rejestracji nowego konta, gdyż użytkownik nie jest zarejestrowany.

Logika walidacji hasła

- **Etap 1 – poprawność wprowadzonego hasła (pod względem założonej liczby znaków)**

Jeżeli hasło nie jest poprawne, wówczas poproś użytkownika o ponowne wprowadzenie danych (tym razem właściwych).

Jeżeli hasło jest poprawne, to przejdź do etapu 2.

- **ETAP 2 – czy użytkownik znajduje się w bazie `soaw`**

Jeżeli użytkownik o danej nazwie (adresie e-mail) nie istnieje, poinformuj o tym na stronie.

Jeżeli użytkownik znajduje się w bazie, przejdź do etapu 3.

- **ETAP 3 – poprawność hasła dla danego użytkownika**

Jeżeli hasło nie jest poprawne, poproś o ponowne wprowadzenie hasła.

Jeżeli hasło jest poprawne, przejdź do etapu 4.

- **ETAP 4 – poinformuj użytkownika, że jest zalogowany, np. przez przeniesienie na określoną stronę dostępną tylko dla użytkowników zalogowanych.**

Pozostaje nam jedynie napisanie kodów odpowiadających funkcjonalnością powyższym etapom, czego efektem końcowym będzie zalogowanie do aplikacji SOAW.

Sprawdzenie poprawności wprowadzonego hasła

W kodzie wykorzystamy:

- funkcję PHP `strlen()`,
- flagę – `$flagaTest`,
- zmienną sesyjną `$_SESSION['e_haslo']`, dzięki której wyświetlimy użytkownikowi stosowny komunikat (o ile będzie taka potrzeba).

Na bazie użytego kodu, gdy hasło nie spełnia warunku liczby znaków z przedziału od 8 do 12, ustawiamy zmienne:

- `$testPassed = false` (informującą przeglądarkę internetową, że musi się tutaj zatrzymać);
- `$_SESSION['e_haslo'] = "Hasło musi mieć od 8 do 12 znaków, wpisz ponownie hasło o wymaganej długości znaków"` (będącą komunikatem ostrzegawczo-informacyjnym).

Liczba zmiennych informacyjnych rośnie (będzie ich więcej) i po powrocie na index.php przydałoby się ich wyczyszczenie (usunięcie). Metod (w tym miejscu możliwych do umieszczenia) pozwalających na wykonanie tego zadania jest wiele. Skorzystamy z pozornie najbardziej oczywistego rozwiązania – dodamy stosowne kody PHP w pliku index.php i login.php tuż po rozpoczęciu sesji `session_start()`; (rys. 33.12).

```
// WALIDACJA DŁUGOŚCI HASŁA
// Gdy ciąg znaków wpisany do input -
// Hasło, będzie mniejszy niż 8 lub większy
// niż 12, to flaga będzie ustawiona na false
// i program zatrzyma swój bieg.

if (
    (strlen($password) < 8) ||
    (strlen($password) > 12)
)
{
    $flagaTest = false;

    // Ustawiamy zmienną, która będzie
    // komunikatem ostrzegawczo-informacyjnym
    $_SESSION['e_haslo'] = "Hasło musi
    posiadać od 8 do 12 znaków!";
}
```

Rys. 33.11. Kod PHP – logika sprawdzająca, czy ciąg znaków ma odpowiednią długość i ustawia stosowną informację dla użytkownika

Wraz z rozwojem aplikacji zapewne trzeba będzie usunąć kilka zmiennych sesyjnych, które nie będą w danym momencie już potrzebne.

Opisane podejście do czyszczenia zmiennych nie jest jednak optymalne. Przeanalizujmy to zagadnienie: użytkownik wprowadzi poprawny adres e-mail za pierwszym razem, ale wpisze niewłaściwe hasło – wówczas podczas drugiej próby wpisania danych w input pojawi się komunikat dotyczący błędnego wpisania adresu e-mail.

Skutecznym rozwiązaniem tego problemu może być przeniesienie czyszczenia zmiennej sesyjnej w inny obszar, np. na początek (wewnątrz) instrukcji warunkowej `if() {}` danego bloku (czyli odpowiednio loginu i hasła).


```
<?php

session_start();

// Usuwanie informacji dla użytkownika
powstałych przy przechodzeniu ze strony głównej
(z nieudanej próby logowania)

if (isset($_SESSION['e_email']))
    unset($_SESSION['e_email']);

if(isset($_SESSION['e_haslo']))
    unset($_SESSION['e_haslo']);
```

Rys. 33.12. Kod PHP – usunięcie zbędnych zmiennych

```
if (isset($_POST['email']) &&
    strlen($_POST['email']) > 0)
{
    // echo 'Login wprowadzono poprawnie';
    //Udana walidacja? Załóżmy, że tak,
    //czyli ustalmy flagę
    $flagaTest = true;

    // TESTOWANIE ROZWIĄZANIA
    if (isset($_SESSION['e_email'])) {
        unset($_SESSION['e_email']);
    }
    // ciąg dalszy pierwszego wyrażenia if

    // Ustaw zmienną $email
    $email = trim($_POST['email']);
```

Rys. 33.13. Kod PHP – umieszczenie w miejscu alternatywnym (usunięcia zbędnych zmiennych)

Utworzyliśmy komunikat dla użytkownika. Warto go pokazać. W tym celu wewnątrz formularza HTML pod polem input przeznaczonym do wpisywania hasła dodamy kod PHP zawierający logikę, która umożliwi wyświetlenie komentarza (rys. 33.14).

```
<br> Hasło:  
<input type="text" placeholder="Twoje  
hasło" name="password">  
<br>  
<?php  
if (isset($_SESSION['e_haslo']))  
{  
    echo $_SESSION['e_haslo'];  
}  
?>
```

Rys. 33.14. Kod PHP wewnątrz HTML – logika odpowiedzialna za wyświetlenie informacji dla użytkownika

Wykonaliśmy skuteczny mechanizm dotyczący danych:

- pobranie od użytkownika,
- sanityzacja,
- walidacja,
- sprawdzenie poprawności pod względem rodzaju,
- sprawdzenie poprawności pod względem długości znaków.

Kolejnym etapem będzie połączenie z bazą danych. Po uzyskaniu dostępu sprawdzimy, czy użytkownik, który wprowadził dane do pól input, jest faktycznie zarejestrowany w naszej bazie. Dowiemy się, czy adres e-mail, który wpisał, jest w bazie i czy podał właściwe hasło.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij, czy każdy kod języka programowania PHP musi mieć zapis początku skryptu '`<?php`' oraz jego zakończenia, czyli '`?>`'.
2. Co to jest uruchomienie sesji PHP?
3. Wyjaśnij pojęcie flagi w programowaniu.
4. Wyjaśnij mechanizm walidujący hasło w aplikacji webowej.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wykonaj poniższe polecenia.
 - A. Na kartce papieru (lub w edytorze tekstu) napisz pseudokod, który będzie odpowiedzialny za połączenie aplikacji webowej z bazą danych.
 - B. Pseudokod z punktu A zapisz jako skrypt w języku programowania PHP.
 - C. Sprawdź, czy przygotowany program działa właściwie – czy istnieje aktywne połączenie z bazą danych.
 - D. Jeżeli nie działa, to najpierw samodzielnie poszukaj błędów.
 - E. Gdy nie będziesz w stanie uruchomić połączenia, zajrzyj do kodów zawartych w tym podręczniku i właściwie dopasuj kod.
2. Napisany kod PHP nie jest optymalny. Zastanów się, w którym miejscu i jak dokonać zmian. Przemyśl ten problem w zakresie optymalnego umieszczenia kodu odpowiadającego za usunięcie zbędnych zmiennych sesyjnych.

34

SOAW – obiektowy dostęp do bazy danych z wykorzystaniem PDO

ZAGADNIENIA

- Konfiguracja środowiska połączeniowego
- Podstawy PDO (PHP Data Objects)
- Połączenie z bazą danych z wykorzystaniem PDO – podstawy teoretyczne
- Testowanie połączenia z bazą danych z wykorzystaniem PDO
- Sprawdzenie istnienia odpowiednich danych: loginu i hasła w bazie – logika
- Instrukcja try – catch php – obsługa błędów
- query() – metoda PDO
- Sprawdzenie poprawności hasła
- Metody PDO: prepare(), execute(), fetch() i symbole zastępcze =?

34.1. Konfiguracja środowiska połączeniowego

Na początek warto przypomnieć etapy, które trzeba zrealizować, by uzyskać poprawne potwierdzenie hasła:

- poprawność wprowadzonego hasła (pod względem założonej liczby znaków),
- obecność użytkownika w bazie 'soaw',
- poprawność hasła dla danego użytkownika,
- poinformowanie użytkownika, że został zalogowany.

Etap pierwszy już zrealizowaliśmy. Przypomnę, że w części dotyczącej typów danych dokonaliśmy pierwszego wpisu, używając polecenia **INSERT INTO** (częścią języka zapytań SQL) wewnątrz phpMyAdmin.



CIEKAWOSTKA

Wydaje się, że w zakresie rozwoju: stron WWW, małych portali firmowych i usługowych, sklepów online i portali edukacyjnych połączenie technologii:

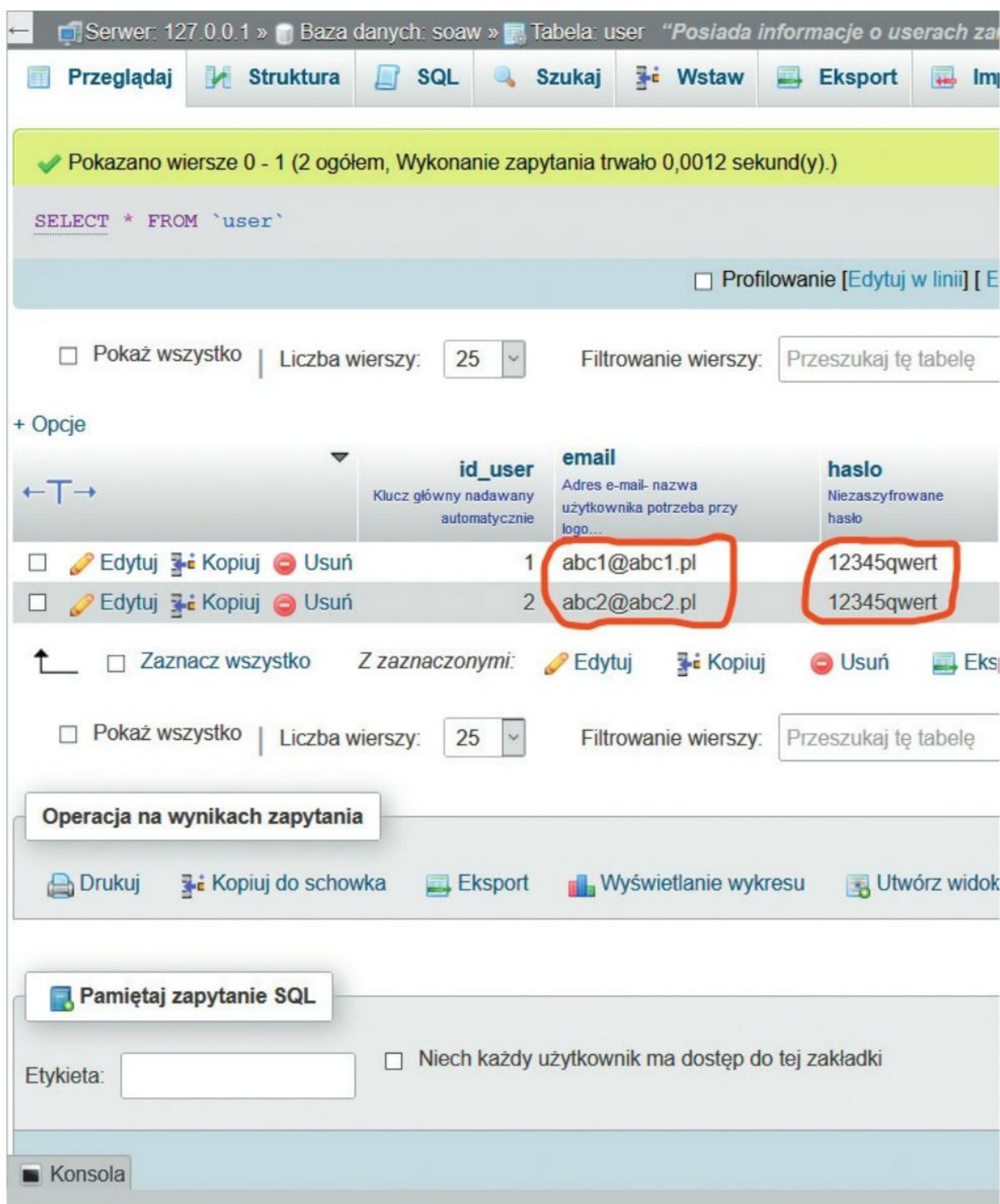
- PHP,
- SQL,
- serwer internetowy (np. Apache bądź Nginx)

jest najpopularniejszym rozwiązaniem wśród małych i średnich firm inwestujących w biznes internetowy.

Dzieje się tak zapewne z co najmniej dwóch powodów:

- niskiego poziomu skomplikowania omawianych technologii,
- niskiego poziomu kosztów związanych z dostępem do wskazanych rozwiązań.

Warto zajrzeć do tego menedżera baz danych i przypomnieć sobie zapisane hasła. Można to zrobić, gdyż nie wprowadziliśmy haseł zaszyfrowanych, a jedynie zwykłe ciągi znakowe. Stosowne informacje powinny się także znajdować w pliku zawierającym informacje dotyczące tworzenia bazy danych. W naszym przypadku to soaw.sql. Na następnych stronach znajdziesz wszystkie kody SQL niezbędne do stworzenia właściwej bazy danych, a w niej odpowiednich tabel projektowych.



Server: 127.0.0.1 » Baza danych: soaw » Tabela: user "Posiada informacje o userach za"

Przeglądaj Struktura SQL Szukaj Wstaw Eksport Im

✓ Pokazano wiersze 0 - 1 (2 ogółem, Wykonanie zapytania trwało 0,0012 sekund(y).)

SELECT * FROM `user`

☐ Profilowanie [Edytuj w linii] [E]

☐ Pokaż wszystko | Liczba wierszy: 25 Filtrowanie wierszy: Przeszukaj tę tabelę

+ Opcje

	id_user	email	haslo
	Klucz główny nadawany automatycznie	Adres e-mail- nazwa użytkownika potrzeba przy logo...	Niezaszyfrowane hasło
☐ Edytuj ☐ Kopiuj ☐ Usuń	1	abc1@abc1.pl	12345qwert
☐ Edytuj ☐ Kopiuj ☐ Usuń	2	abc2@abc2.pl	12345qwert

☐ Zaznacz wszystko Z zaznaczonymi: ☐ Edytuj ☐ Kopiuj ☐ Usuń ☐ Eks

☐ Pokaż wszystko | Liczba wierszy: 25 Filtrowanie wierszy: Przeszukaj tę tabelę

Operacja na wynikach zapytania

☐ Drukuj ☐ Kopiuj do schowka ☐ Eksport ☐ Wyświetlanie wykresu ☐ Utwórz widok

☐ Pamiętaj zapytanie SQL

Etykieta: ☐ Niech każdy użytkownik ma dostęp do tej zakładki

Konsola

Rys. 34.1. phpMyAdmin – widok danych dostępowych użytkowników


```

-----
-- Schema soaw
--
-- Serwis Ogłoszeniowy Aplikacja Webowa (soaw)
CREATE SCHEMA IF NOT EXISTS `soaw` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
USE `soaw`;

```

Rys. 34.2. Widok kodów pozwalających na stworzenie zasobów SQL projektu SOAW – część 1 stworzenia bazy

```

-----
-- Table `soaw`.`user` - stworzenie tabeli user
--
CREATE TABLE IF NOT EXISTS `soaw`.`user` (
  `id_user` INT NOT NULL AUTO_INCREMENT COMMENT 'Klucz główny nadawany
  automatycznie',
  `email` VARCHAR(30) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'
  NOT NULL COMMENT 'Adres e-mail- nazwa użytkownika potrzeba przy logowaniu.',
  `haslo` VARCHAR(256) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'
  NOT NULL COMMENT 'Niezaszyfrowane hasło',
  `imie` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci' NOT

```



```

NULL COMMENT 'Imię użytkownika',
`nazwisko` VARCHAR(30) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'
NOT NULL COMMENT 'Nazwisko użytkownika',
`telefon` VARCHAR(12) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'
NULL COMMENT 'Telefon użytkownika 9-io cyfrowy',
PRIMARY KEY (`id_user`),
UNIQUE (`email`))
ENGINE = InnoDB
COMMENT = 'Posiada informacje o userach zarejestrowanych w programie.';

```

Rys. 34.3. Widok kodów pozwalających na stworzenie zasobów SQL projektu SOAW – część 2 stworzenia bazy

```

-- -----
-- Dodawanie użytkowników do tabeli `soaw`.`user`
-- -----
INSERT INTO `user` (`id_user`, `email`, `haslo`, `imie`, `nazwisko`,
`telefon`) VALUES
(null, 'abc1@abc1.pl', '12345qwert', 'Jarek', 'Lubokotny', '509000000'),
(null, 'abc2@abc2.pl', '12345qwert', 'Andrew', 'Dudilsky', '1111111111');

```

Rys. 34.4. Widok kodów pozwalających na stworzenie zasobów SQL projektu SOAW – część 3 stworzenia bazy


```
-----  
-- Table `soaw`.`admin` - Tabela 2. admin  
-----  
  
CREATE TABLE IF NOT EXISTS `soaw`.`admin` (  
  `id_admin` INT NOT NULL AUTO_INCREMENT COMMENT 'Klucz główny przydzielony  
  automatycznie',  
  `login` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'  
  NOT NULL COMMENT 'Nazwa administratora potrzebna przy logowaniu',  
  `haslo` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci'  
  NOT NULL COMMENT 'Hasło niezaszyfrowane',  
  PRIMARY KEY (`id_admin`))  
ENGINE = InnoDB  
COMMENT = 'Posiada informacje o administratorach zarejestrowanych w programie.'  
;
```

Rys. 34.5. Widok kodów pozwalających na stworzenie zasobów SQL projektu SOAW – część 4 stworzenia bazy

Na rys. 34.3 przedstawiono polecenie języka programowania SQL – CREATE SCHEMA, za którego pomocą jest tworzona baza danych o nazwie `soaw`. Na rys. 34.3 zaprezentowano polecenie języka programowania SQL – CREATE TABLE. Dzięki niemu w bazie danych o nazwie `soaw` jest tworzona tabela o nazwie `user`. Tabela będzie miała kolumny:

- 'id_user',
- 'email',
- 'haslo',
- 'imie',
- 'nazwisko',
- 'telefon'.

Na rys. 34.4 przedstawiono polecenie języka programowania SQL – INSERT INTO, za którego pomocą do tabeli o nazwie 'user' są dodawane dwa rekordy (ostatnie dwie linijki po VALUES), a na rys. 34.5 pokazano polecenie języka programowania SQL – CREATE TABLE. Po jego użyciu w bazie danych o nazwie 'soaw' jest tworzona tabela o nazwie 'user'. Tabela będzie miała kolumny:

- 'id_admin',
- 'login',
- 'haslo'.

Podstawowym zadaniem tego etapu będzie sprawdzenie, czy w bazie danych 'soaw' istnieje użytkownik o danej nazwie. W naszym przypadku oznacza to wyszukania:

- właściwej tabeli – 'user',
- właściwej kolumny – 'email',
- ciągu znaków.

Gdy szukany ciąg znaków zostanie znaleziony w kolumnie 'email' tabeli 'user', to będziemy mogli przejść do sprawdzenia poprawności hasła, czyli do etapu 3.

Może się zdarzyć, że danego adresu e-mail w kolumnie 'email' tabeli 'user' nasz kod nie znajdzie. Wtedy musimy doprowadzić do poinformowania użytkownika, że jest problem i jeśli chce kontynuować, to musi wprowadzić poprawny, tzn. zarejestrowany login. Warto także zasugerować rozwiązanie alternatywne, jakim jest rejestracja w naszym portalu nowego konta (specjalnie dla niego).

Plik konfiguracyjny – do połączenia z bazą danych

W głównym folderze projektu SOAW przygotujemy dodatkowy katalog, który będzie przechowywał pliki konfiguracyjne. Nazwijmy go 'config', a w nim stwórzmy pierwszy plik konfiguracyjny, czyli taki, który określi ramy środowiska informatycznego aplikacji SOAW związane z bazą danych.

We wskazanym katalogu utwórz plik connectUser.php – będziemy w nim przechowywali parametry połączenia z bazą danych projektu 'soaw'. Na rys. 34.6 znajdziesz zawartość pliku z ustawieniami.


```
<?php

    $host = "localhost";
    $db_user = "root";
    $db_password = "";
    $db_name = "soaw";
    $dsn      = "mysql:host=$host;
                dbname=$db_name;
                charset=utf8mb4";
    $options = array(
                PDO::ATTR_ERRMODE =>
                PDO::ERRMODE_EXCEPTION
            );

?>
```

Rys. 34.6. Kody pliku connectUser.php – ustawienie środowiska informatycznego aplikacji SOAW

Dane, które zawarto w connectUser.php, musimy udostępnić innym plikom. Zrobimy to przez dołączenie pliku z konfiguracją do danego pliku (np. do login.php) za pomocą polecenia PHP:

```
include_once 'config/connectUser.php' ;
```

Pracując dalej nad kodem, musisz dopasować dane dostępowe do takich, które są właściwe dla twojego XAMPP-a. W naszym przypadku:

użytkownik nazywa się 'root', więc mamy

```
$db_user = "root";
```

nie ma hasła, dlatego mamy pusty cudzysłów:

```
$db_password = "";
```

baza nazywa się 'soaw', dlatego w kodzie jest

```
$db_name = "soaw";
```

34.2. Podstawy PDO (PHP Data Objects)

W programowaniu istnieje wiele sposobów połączenia z relacyjną bazą danych. Zwykle zaczyna się naukę od funkcji MySQLi. Skorzystamy z rozwiązania nowszego, opartego na PDO.


```

$pdo = new PDO('mysql:host=localhost;dbname=hotel', 'root', '');
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$stmt = $pdo->query('SELECT id, ile_osob, klima, niepalacy FROM rooms');
echo '<ul>';
foreach($stmt as $row)
{
    echo '<li>'. $row['ile_osob'].': '. $row['klima'].': '. $row['niepalacy']. '</li>';
}

$stmt = $pdo->prepare("SELECT id, ile_osob, klima, niepalacy FROM rooms");
$stmt->execute();

/* Pobierz wszystkie pozostałe wiersze w zestawie wyników */
print("Pobierz wszystkie pozostałe wiersze w zestawie wyników :\\n");
$result = $sth->fetchAll();
print_r($result);

$stmt->closeCursor();
echo '</ul>';

```

Rys. 34.7. PDO – przykład połączenia z relacyjną bazą danych, a także pobrania i wyświetlenia w przeglądarce pozyskanych danych celem wyświetlenia ich na stronie WWW



CIEKAWOSTKA

W języku programowania PHP istnieje kilka sposobów połączenia z relacyjnymi bazami danych. Jednym z nich jest PDO – nowoczesny interfejs napisany w OOP (ang. *object-oriented programming*, programowanie obiektowe). Na rys. 34.7 znajdziesz przykład wykorzystania PDO w kodzie PHP.

Struktura projektu

Połączymy się z bazą danych, wykorzystując:

- pakiet XAMPP (sugeruję, aby ten pakiet był włączany w Windows z pełnymi uprawnieniami administracyjnymi, najbezpieczniej z użyciem prawego klawisza myszki z opcją „Uruchom jako administrator”);
- PDO – jest to rozszerzenie języka PHP (biblioteka), które udostępnia jednolity interfejs baz danych, opierając się na projektowaniu obiektowym (klasy z metodami wspólnymi dla różnych silników zarządzania bazą danych).

PHP Data Objects obsługuje wiele narzędzi bazodanowych, m.in. MariaDB, Postgres, MSSQL, MySQL.

Więcej informacji na temat PDO znajdziesz na:

- stronie WIKIBOOKS – https://pl.wikibooks.org/wiki/PHP/Biblioteka_PDO,
- oficjalnej stronie projektu – <https://www.php.net/manual/en/book.pdo.php>.

Praktyczne wykorzystanie PDO sprowadza się do stworzenia połączenia z bazą danych z wykorzystaniem tzw. programowania obiektowego. Aby tego dokonać, najpierw stworzyliśmy skrypt, który ustawia parametry połączeniowe. Dotyczy to zawartości pliku `config/connectUser.php`. Poniżej znajdziesz krótkie wyjaśnienie do zapisanego kodu.

```
$host = "localhost"; // wskazujemy host aplikacji, czyli adres, pod którym
                        działa XAMPP (lokalny serwer)

$db_user = "root";    // dane Użytkownika bazy danych: name

$db_password = "";    // dane Użytkownika bazy danych: password, gdy nie
                        ma hasła, wstawiamy pusty string po =

$db_name = "soaw";    // nazwa bazy danych, z którą chcemy się połączyć

$dsn = "mysql:host=$host;dbname=$db_name;charset=utf8mb4";
```

Ostatnia linijka z powyższych wyjaśnień, ta zaczynająca się od `$dsn`, reprezentuje specjalny ciąg znaków, które identyfikują rodzaj serwera DB (tutaj: `mysql`), host, na którym jest uruchomiona usługa (tutaj: `localhost`), nazwę bazy, z którą chcemy się połączyć (tutaj: „soaw”), i kodowanie znaków obsługujące m.in. polskie fonty. Istnieją jeszcze

inne parametry połączeń – tzw. opcjonalne. Po szczegóły warto sięgnąć do dokumentacji PHP, np. <https://www.php.net/manual/en/book.pdo.php> (oficjalna strona dokumentacji języka PHP).

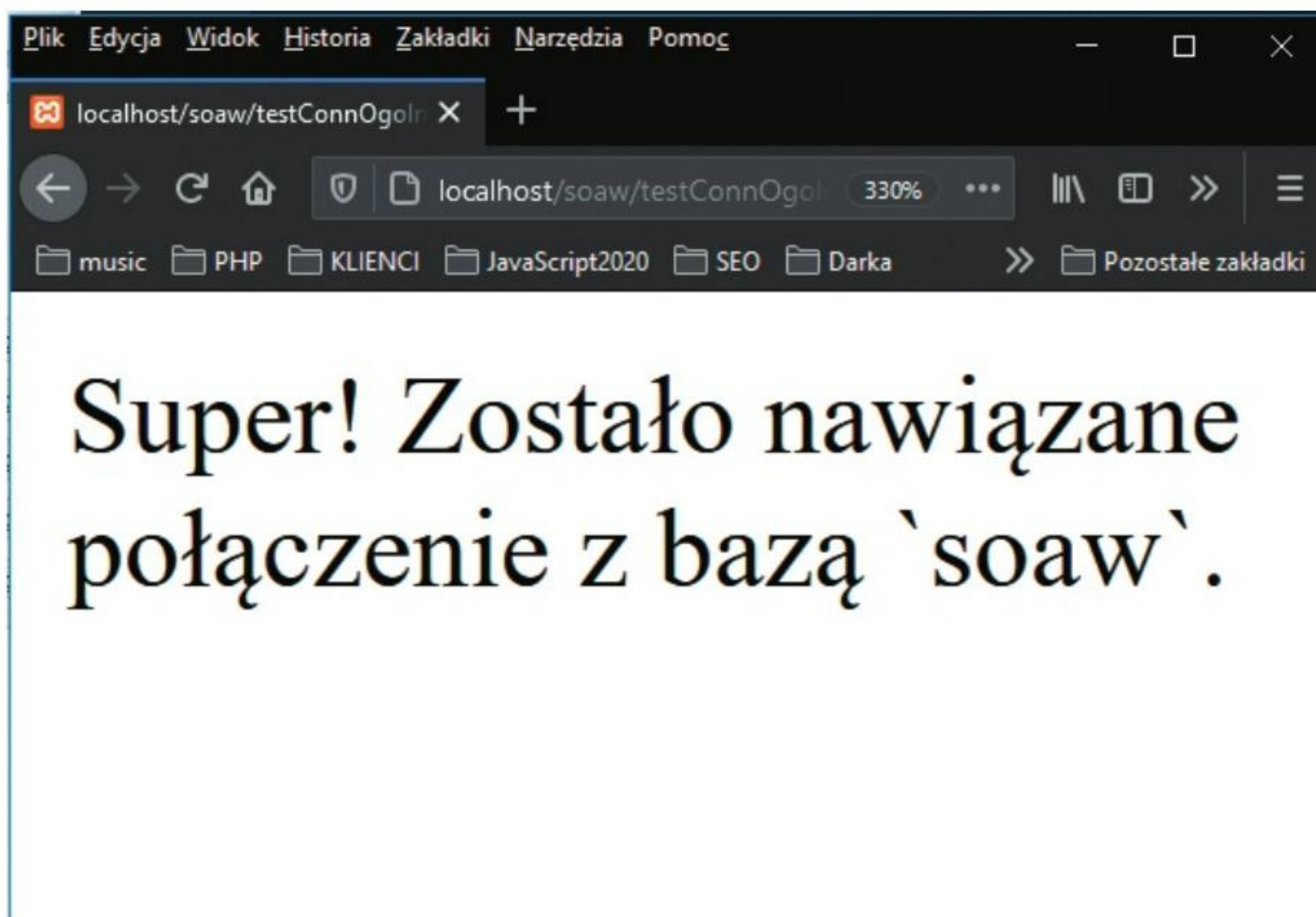
34.3. Połączenie z bazą danych z wykorzystaniem PDO – podstawy teoretyczne

Sprawdźmy, czy uda się nam połączyć z relacyjną bazą danych projektu – `soaw`. W tym celu napiszemy kod w oparciu o PDO – nowe zapytanie PDO:

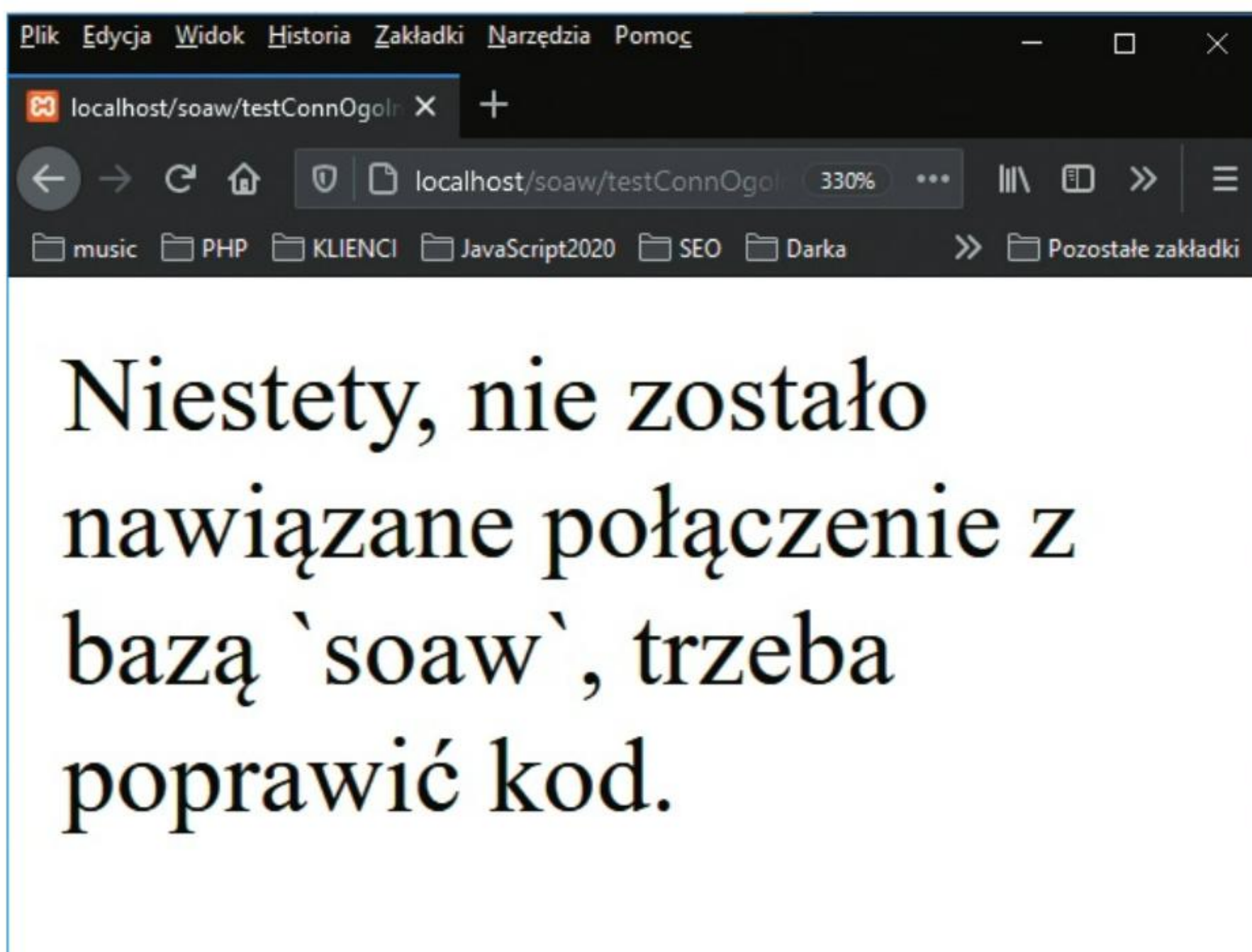
```
$connection = new PDO($dsn, $db_user, $db_password, $options);  
if($connection) {  
    echo "Super! Zostało nawiązane połączenie z bazą 'soaw'.";  
}  
  
else  
{  
    echo "Niestety, nie zostało nawiązane połączenie z bazą  
    'soaw', trzeba poprawić kod.";  
}
```

```
$connection = new PDO($dsn, $db_user, $db_password,  
$options);  
  
if($connection)  
{  
    echo "Super! Zostało nawiązane połączenie z bazą  
    `soaw`.";  
}  
else  
{  
    echo "Niestety, nie zostało nawiązane połączenie  
    z bazą `soaw`, trzeba poprawić kod.";  
}
```

Rys. 34.8. PDO – prosty test połączenia z relacyjną bazą danych



Rys. 34.9. PDO – pozytywny wynik testu połączenia z relacyjną bazą danych



Rys. 34.10. PDO – negatywny wynik testu połączenia z relacyjną bazą danych

Jeżeli na stronie pojawił się napis „Super! Zostało nawiązane połączenie z bazą 'soaw'”, możesz kontynuować dalsze tworzenie projektu SOAW. Jeżeli zaś na stronie wyświetlił się napis „Niestety, nie zostało nawiązane połączenie z bazą 'soaw', trzeba poprawić kod”, musisz odszukać błąd. Problemy mogą wystąpić w kilku obszarach:

- kod – sprawdź, czy nie ma np. literówki, być może w jednym miejscu masz wpisane: `'$connection'`, a w innym `'$cconnection'`;
- parametry połączenia – sprawdź, czy na pewno w pliku `connectUser.php` ustawiono właściwe informacje dotyczące nazwy: hosta, bazy danych, użytkownika bazy itd.;
- oprogramowanie (środowisko) – być może XAMPP nie jest włączony (albo tylko jeden z jego elementów); warto sprawdzić, czy Apache i MySQL są uruchomione;
- baza o nazwie 'soaw' – czy w XAMPP istnieje baza o nazwie projektu.

Kolejnym zagadnieniem wartym odnotowania jest kwestia zamknięcia połączenia. Jest ono dodatkowym obciążeniem aplikacji. Gdy baza zostanie już odpytana i uzyskamy potrzebną odpowiedź, trzeba je zamknąć. Utrzymywanie aktywnego połączenia zbyt długo może stanowić potencjalne niebezpieczeństwo. Oczywiście możemy wdrożyć po stronie serwera automatyczne zamykanie sesji (i połączenia) po określonym czasie bezczynności. Mimo wszystko zaleca się dodać zabezpieczenie także po stronie frontendu. Warto zajrzeć do dokumentacji PDO: „Po pomyślnym połączeniu z bazą danych wystąpienie klasy PDO jest zwracane do skryptu. Połączenie pozostaje aktywne przez cały okres istnienia tego obiektu PDO. Aby zamknąć połączenie, trzeba zniszczyć obiekt, upewniając się, że wszystkie pozostałe odwołania do niego zostały usunięte. Można to zrobić, przypisując wartość null do zmiennej, która przechowuje obiekt. Jeśli nie zrobisz tego jawnie, PHP automatycznie zamknie połączenie po zakończeniu skryptu”.

UWAGA

„Jeśli nadal istnieją inne odwołania do tego wystąpienia PDO (np. z wystąpienia `PDOStatement` lub innych zmiennych odwołujących się do tego samego wystąpienia PDO), muszą one zostać również usunięte (np. przez przypisanie null do zmiennej, która odwołuje się do obiektu klasy `PDOStatement`)”.

34.4. Testowanie połączenia z bazą danych z wykorzystaniem PDO

Aby przetestować pierwsze połączenie z relacyjną bazą danych z wykorzystaniem PDO, można stworzyć w katalogu głównym projektu nowy plik o nazwie `testConnSoaw.php`. Nie będzie on nam potrzebny w przyszłości, gdyż jest to plik dodatkowy. Zapiszemy w nim tylko mały kawałek kodu PHP, za którego pomocą połączymy się z bazą 'soaw' i wyświetlimy z wykorzystaniem pętli `foreach` w przeglądarce internetowej (na stronie `testConnSoaw.php`) dane pobrane z bazy danych. Tworzony teraz plik (a w nim kod PHP) nie jest częścią projektu z punktu widzenia potencjalnego klienta, dla którego

tworzymy projekt SOAW. Jest to jedynie test dla zespołu deweloperskiego, sprawdzający poprawność konfiguracji środowiska, na którym tworzymy SOAW.

Testy sprawdzające ogólną poprawność konfiguracji są niezmiernie ważne. Wyobraź sobie, że jesteś deweloperem i stworzyłeś kilka tysięcy linii kodu, uruchamiasz go i... nic nie działa. Błędy w całym kodzie możesz szukać nawet kilka dni. Z tego powodu warto sprawdzać środowisko połączenia z bazą i inne bazowe elementy. Poleca się testowanie małych kawałeczków projektu, kilku linii początkowych kodu zamiast pisania od razu całego projektu. Takie działanie na pewno zaoszczędza programistom mnóstwa stresu.

Zacniemy od stworzenia obsługi połączenia testowego, aby potem sprawnie i bez nadmiernego stresu poradzić sobie z wykonaniem skutecznego połączenia w naszym projekcie.

Pora na wykorzystanie wiedzy z zakresu podstaw programowania obiektowego. Wykorzystamy także mały element z potężnych możliwości biblioteki PDO. Zacnijmy od wyjaśnienia zasad działania połączenia z bazą. Na rys. 34.11 znajdziesz całość kodu, który zostanie omówiony na następnej stronie.

- **Krok 1 – new PDO(..)**

new PDO(..) – stworzymy nowy obiekt PDO, czyli będzie to nowe połączenie z bazą danych.

- **Krok 2 – \$connection**

Stworzenie zmiennej **\$connection** i przypisanie do niej stworzonego połączenia. Umożliwi to operator dostępu do metod i właściwości obiektu; w PHP jest nim myślnik, a obok niego znak większości, czyli **->**.

W nawiasie znajdują się parametry połączenia, ustawiliśmy je w pliku **connectUser.php** znajdującym się w katalogu **config**. Czyli nasza zmienna będzie miała postać:

```
$connection = new PDO($dsn, $db_user, $db_password, $options);
```

- **Krok 3 – if \$connection**

Sprawdzenie, czy istnieje zmienna **\$connection**, czyli czy zostało ustanowione połączenie z serwerem i bazą **'soaw'**.

- **Krok 4 – \$connectionResult**

Przypisanie do nowej zmiennej **\$connectionResult** obiektu połączenia, czyli **\$connection**.

- **Krok 5 – query()**

Wykorzystując metodę **query**, pobranie w obiektowy sposób (z wykorzystaniem operatora **->**) z obiektu **\$connection** danych z bazy.

[illegible]

Rys. 34.11. Kody pliku testConnSoaw.php – testowanie działania PDO w ramach środowiska informatycznego aplikacji SOAW

- **Krok 6 – SQL**

Zapytanie SQL

```
"SELECT * FROM user"
```

Za pomocą kodu języka SQL zostaną zwrócone z tabeli **'user'** wszystkie rekordy (informuje o tym gwiazdka *).

- **Krok 7 – foreach() { }**

Wykorzystując połączenie języka skryptowego PHP i języka znaczników HTML, pobrane rekordy z bazy na stronie WWW zostaną pokazane (wyświetlone) pod adresem <http://localhost/soaw/testConnSoaw.php>.

W wyniku powyższych kroków na stronie internetowej pod adresem <http://localhost/soaw/testConnSoaw.php> powinny pokazać się informacje z tabeli **'user'** bazy danych **'soaw'**, wyglądające podobnie jak na rys. 34.12.

Dane pobrane z bazy:

	id	adres email	hasło	Imię i Nazwisko	Numer telefonu
•	1	abc1@abc1.pl	12345qwert	Jarek Lubokotny	509000000
•	2	abc2@abc2.pl	12345qwert	Andrew Dudilsky	111111111

Rys. 34.12. Kody pliku testConnSoaw.php – testowanie działania PDO w ramach środowiska informatycznego aplikacji SOAW

Doszło do połączenia z bazą danych **'soaw'**. Zostały pobrane wszystkie rekordy, a następnie ich zawartość została (w pętli) wyświetlona na stronie WWW. Całość wyświetla się w dość brzydkiej postaci:

- wypunktowania nienumerowanego,
- prostego HTML-a,
- tzw. twardej spacji – **' '**.

Pamiętajmy, że to był jedynie test i szkoda czasu, aby poprawić wygląd czegoś, co dalej nie będzie już potrzebne.

Kod, który napisaliśmy do obsługi bazy danych, działa sprawnie, ale jest pewnym uproszczeniem. Brakuje w nim kilku elementów, m.in. nasze zapytanie powinno potrafić obsłużyć błędy. Pokażemy to w trakcie pracy nad właściwym kodem projektu.

34.5. Sprawdzenie istnienia odpowiednich danych – loginu i hasła w bazie z wykorzystaniem PDO

Wykorzystamy wiedzę z zakresu PDO i przeanalizujemy kod, który łączy frontend projektu SOAW z jego bazą danych. Umożliwi on nam sprawdzenie, czy istnieje dany użytkownik i czy hasło, które podał, pasuje do tego, które jest przechowywane w bazie danych.

Popatrz na kod na rys. 34.13. Przeanalizuj go dokładnie. Narysuj na kartce papieru powyższą grafikę za pomocą strzałek i tekstu tak, aby pokazać zasady działania logiki. Musisz być w stanie wyjaśnić własnymi słowami kolejność i zależność zdarzeń przedstawionych w powyższych kodach.

Czy użytkownik w bazie istnieje?

Sprawdzimy, czy dany użytkownik istnieje w naszej bazie danych, a dokładniej – dokonamy weryfikacji adresu e-mail. Poszukamy go w bazie danych w tabeli **'user'** w kolumnie **'email'**.

Co ważne, istnieje kilka sposobów na osiągnięcie wskazanego celu. Przykładowo: można pobrać wszystkie kolejne pozycje z kolumny **'email'**, a potem po prostu porównać wpisany string z każdym z wyników pobierania. Jest to rozwiązanie dobre, ale przy bardzo dużych projektach, zawierających setki tysięcy (albo miliony) rekordów danych, byłoby bardzo obciążające. Dlatego skorzystamy z innego rozwiązania – w pytaniu SQL umieścimy warunek sprawdzający. Dokonamy w nim testu: „**Czy string wpisany przez użytkownika znajduje swój odpowiednik w kolumnie 'email'?**”.

Sprawdzimy wyłącznie: „**Ile rekordów jest zwróconych jako odpowiedź?**”.

Gdy otrzymamy zero rekordów, to znaczy, że dany adres e-mail nie ma swojego odpowiednika w bazie projektu SOAW. Jeżeli rekordów będzie więcej niż zero, to dany adres e-mail jest zarejestrowany (znajduje się już w bazie **'soaw'** w tabeli **'user'**). Będzie to oznaczało, że aplikacja będzie mogła przejść do kolejnego kroku, czyli sprawdzenia poprawności hasła. Zauważ, że opisany sposób testowania danego ciągu znaków nie obciąża tak bardzo całego ekosystemu utrzymującego aplikację, co ten pierwszy wskazany. Jest to bardzo istotne przy olbrzymich projektach. Mimo że tworzony przez nas jest mały, to i tak warto pomyśleć o tzw. skalowalności. Trzeba założyć, że w przyszłości aplikacja będzie rozwijana i wtedy rozwiązanie będzie już gotowe, bez potrzeby dokonywania zmian. Warto wspomnieć o użytym pojęciu – ekosystem utrzymujący aplikację; chodzi o:

- serwer fizyczny (czyli po prostu komputer),
- infrastrukturę przesyłową (internet, kable itd.),
- oprogramowanie serwerowe,
- aplikacje zainstalowane na serwerze,
- dodatkowe oprogramowanie wspomagające działanie aplikacji webowych.


```
if (
    isset($_POST['email']) &&
    strlen($_POST['email']) > 0)
{
    if ($flagaTest == true) {
        // POŁĄCZENIE Z BAZĄ DANYCH Z WYKORZYSTANIEM PDO
        try {
            $connection = new PDO($dsn, $db_user,
                $db_password, $options);
            if (!$connection) {
                throw new Exception($connection->error);
            } else {
                // Kod sprawdzający użytkownika o danej
                nazwie oraz zgodność podanego hasła.
            }
        } catch (Exception $e) {
            echo 'Błąd serwera! .. (tutaj także może być
                szczegółowa informacja o błędach)';
        }
        // HASŁO - PASSWORD - SPRAWDZENIE

        if ($testEmail == true) {
            // LOGIN - EMAIL - SPRAWDZANIE
        }
    }
} else {
    $_SESSION['e_email'] = 'brak Login - wprowadź
        poprawny login, którym musi być Twój adres email
        zarejestrowany w naszej bazie';
}
?>
```

Rys. 34.13. Plik login.php – ogólna struktura kodu pokazująca jedynie szkielet logiczny aplikacji

Opisanego sprawdzenia informacji o istnieniu danego adresu e-mail w bazie dokonamy z użyciem wspomnianej już technologii PDO. Wykorzystamy w niej instrukcję **try - catch**, która obsłuży przechwytywanie błędów (gdyby jakieś powstały).

34.6. Instrukcja try-catch PHP – obsługa błędów

Struktura instrukcji **try-catch** wygląda następująco:

```
try
{
    // tutaj program ma wykonać jakieś działanie, np. pobranie danych z bazy SQL
}
catch (Exception $e)
{
    echo 'Masz problem 😊, wystąpił wyjątek nr ' . $e->getCode() . ', komunikat błędu: ' . $e->getMessage() ;
}
?>
```

Sekcja **catch (Exception \$e) { }**, zarządza tzw. obsługą błędów, a wewnątrz nawiasów klamrowych po **try** wpisujemy działanie do wykonania. U nas będzie to próba połączenia z bazą danych i pobranie określonych informacji.

Musimy sobie nieco przybliżyć tzw. obsługę wyjątków w PHP (rys. 34.14).

Gdyby się zdarzyło, że wyjątek nie został przechwycony, wtedy mamy duży problem, co zakończy się wyświetleniem błędu krytycznego ze stosownym komunikatem, zwykle: „Niełapany wyjątek”.

CIEKAWOSTKA

Pod koniec 2020 roku premierę miała nowa wersja PHP – linii 8, określanej jako PHP 8.x. W trakcie powstawania podręcznika najnowszą wersją jest PHP 8.0.3. Oczywiście wersja PHP 7.4 w dalszym ciągu działa i jest aktywnie wspierana, w tym regularnie są poprawiane odnotowywane błędy. Co ważne, te działania będą prowadzone tylko do 28 listopada 2022 roku. Szczegóły można znaleźć na oficjalnej stronie projektu języka programowania PHP: <https://www.php.net/supported-versions.php>.

Czasu na ewentualną zmianę pozostaje całkiem dużo, ale jeżeli planujesz rozpocząć przedsięwzięcie programistyczne, warto od razu rozważyć skorzystanie z wersji PHP 8.x, która jest już w pełni stabilna i przede wszystkim (ponoć) lepsza oraz bardziej nowoczesna od linii 7.x.



Rys. 34.14. „Obsługa wyjątków” w PHP

PHP Exception

Klasa Exception, czyli klasa podstawowa wszystkich wyjątków, jest jedną z wielu już predefiniowanych klas. Składa się ze zbioru metod do zarządzania informacjami dotyczącymi danego wyjątku. Skoro jest to klasa, czyli mamy do czynienia z programowaniem obiektowym, to dostęp do metod klasy uzyskujemy, używając operatora – np. `$exception->getMessage()`.

Poniżej znajdziesz kilka przykładowych metody klasy Exception:

- **getMessage()** – zwraca komunikat błędu, jest zdecydowanie najczęściej wykorzystywaną metodą spośród grupy Exception;

- `getCode()` – zwraca kod błędu;
- `getLine()` – zwraca numer linii, w której wyjątek został rzucony;
- `getPrevious()` – zwraca poprzedni wyjątek;
- `getTrace()` – jedna z obszerniejszych metod, zwraca tablicę zawierającą informacje o funkcjach i metodach wywołanych w momencie rzucenia wyjątku (tzw. ślad stosu).

W przypadku zainteresowania szczegółami warto zajrzeć do oficjalnej dokumentacji PHP – <https://www.php.net/manual/en/language.exceptions.php>.

```
$connection = new PDO($dsn, $db_user,
$badPassword, $options);
```

Rys. 34.15. Plik testConnSoaw.php – błędnie wpisane hasło

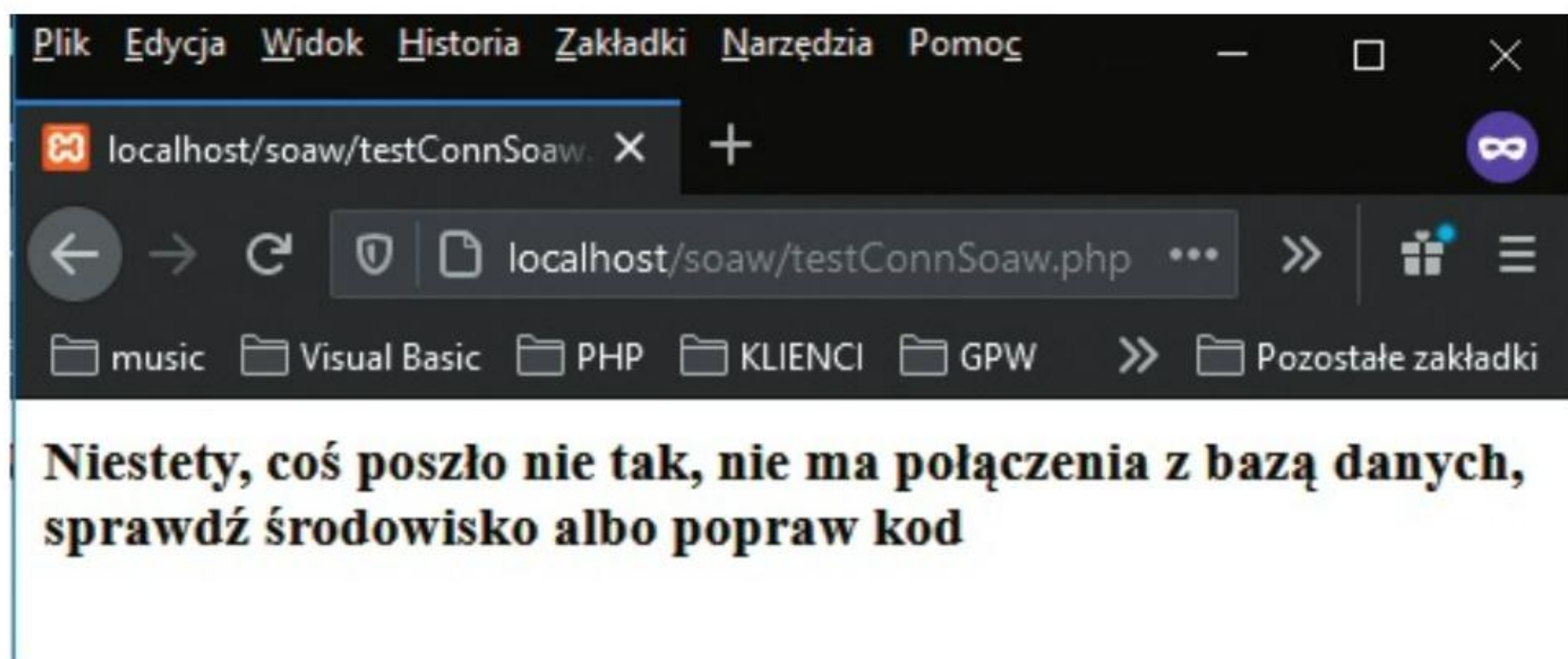
Aby wykonać test poprawności działania, możemy w naszym przykładzie:

- zrobić błąd w nazwie (rys. 34.15),
- zakomentować liniijkę zawierającą stworzenie nowego obiektu bazy danych (rys. 34.16).

```
// $connection = new PDO($dsn, $db_user,
$db_password, $options);
```

Rys. 34.16. Plik testConnSoaw.php – zakomentowanie linijki kodu PHP

Na stronie internetowej <http://localhost/soaw/testConnSoaw.php> w przeglądarce uzyskamy wówczas napis informujący o błędzie (rys. 34.17).



Rys. 34.17. Plik testConnSoaw.php (widok w przeglądarce internetowej) – błąd połączenia z bazą

34.7. query() – metoda PDO

Kolejnym krokiem w tworzonym kodzie jest pozyskanie informacji z bazy danych projektu. Aby to uzyskać, wykonamy metodę `query` na obiekcie PDO – `$connection`:

```
$connection->query( ... ).
```

W pliku `login.php` projektu ta część kodu wygląda tak jak na rys. 34.18.

```
try {
    $connection = new PDO($dsn, $db_user,
        $db_password, $options);
    if (!$connection) {
        throw new Exception($connection->error);
    } else {
        // Połączenie zostało uzyskane.

        // LOGIN - EMAIL - SPRAWDZANIE
        $testEmail = false;
        // Jeżeli jest w bazie adres email taki jak
        // wpisany przez użytkownika do Input, to
        // powstanie zmienna, czyli można to sprawdzić
        // poprzez if(true) count(*)
        $czyJestRekord = $connection->query("SELECT
            * FROM user WHERE email = '$email'");
```

Rys. 34.18. Plik `login.php` – kod PHP wysyłający zapytanie do relacyjnej bazy danych z wykorzystaniem PDO

Za pomocą metody `query()` wysyłamy zapytanie pobierające dane. Treść stworzonego zapytania jest przekazywana do metody `query()` jako argument:

```
"SELECT * FROM user WHERE email = '$email' ".
```

Kod pełnego zapytania:

```
$connection->query("SELECT * FROM user WHERE email = '$email' ").
```

W bibliotece PDO jest mnóstwo metod. Używana przez nas `query()` jest jedną z najprostszych, być może właśnie dlatego dość często używają jej programiści. Zapewne dzieje się tak dlatego, że stosunkowo łatwo jest zrozumieć jej sposób zastosowania, a jej użytkowanie nie jest zbyt skomplikowane. Warto podkreślić, że lepiej ją zastosować jako

tw. przygotowanie zapytania z podstawieniem zmiennych. Kod aplikacji jest wtedy bezpieczniejszy i bardziej odporny na wrogie działania. Nie będziemy szczegółowo na ten temat tutaj dyskutować, zwłaszcza że istnieją dwie przesłanki, które przemawiają za pozostawieniem kodu w obecnym stanie:

- jedynie odpytujemy bazę danych, a nie dokonujemy w niej żadnych zmian, które ingerowałyby w jej wnętrze (np. w dane);
- dla osób poznających PDO obecny stan kodu jest bardziej zrozumiały, na wersję bardziej zaawansowaną przyjdzie czas podczas budowy dalszej części projektu SOAW (w przypadku zainteresowania szczegółami poszukaj informacji np. pod adresem <https://www.php.net/manual/en/book.pdo.php>).

Sprawdzimy, czy jest więcej niż zero wierszy spełniających warunek podany w zapytaniu SQL (rys. 34.19).

```
if ($czyJestRekord->rowCount() > 0) {  
    // Jest taki rekord (email) w bazie  
    $testEmail = true;  
}
```

Rys. 34.19. Plik login.php – kod PHP sprawdzający za pomocą biblioteki PDO, czy warunek zadany w zapytaniu SQL jest spełniony

Efektem powyższego kodu będzie:
gdy rekord z zadaniem adresem e-mail znajduje się w bazie

```
if ($czyJestRekord->rowCount() > 0)
```

to warunek uzyska wartość true, czyli wykonany będzie kod znajdujący się w pierwszych po **if** nawiasach klamrowych. Tam wstawimy kolejną flagę. Tym razem nazwijmy ją **\$testEmail** i ustawmy na true. Ten kod umożliwi przejście do kolejnych linii naszej aplikacji.

Wcześniej w kodzie umieszczono flagę **\$testEmail** i ustawiono ją na false. Można użyć sformułowania, że jest to stan początkowy dla logiki kodu (rys. 34.20).

Powyższe wytłumaczenia wskazują, że dopiero pozytywna realizacja warunku:

```
if ($czyJestRekord->rowCount() > 0)
```

spowoduje, że program przejdzie do kolejnego kroku. Sprawdzi poprawność wpisanego ciągu dla hasła.

Zanim przejdziemy dalej, warto stworzyć zmienne sesyjne (rys. 34.21).


```
try {  
    $connection = new PDO($dsn, $db_user,  
        $db_password, $options);  
    if (!$connection) {  
        throw new Exception($connection->error);  
    } else {  
        // Połączenie zostało uzyskane.  
  
        // LOGIN - EMAIL - SPRAWDZANIE  
        $testEmail = false;  
    }  
}
```

Rys. 34.20. Plik login.php (kod PHP) – flaga \$testEmail ustawiona na false (stan początkowy dla logiki kodu)

```
if ($czyJestRekord->rowCount() > 0) {  
    // Jest taki rekord (email) w bazie  
    $testEmail = true;  
  
    // Ustaw zmienną sesyjną  
    $_SESSION['emailVarSession'] = $emailAfterSanitize;  
    // Jeżeli była wcześniej ustawiona zmienna sesyjna  
    // o złym adresie email, to usuń ją  
    if (isset($_SESSION  
        ['e_zlyemaillogowania'])) {  
        unset($_SESSION['e_zlyemaillogowania']);  
    }  
  
    if (isset($_SESSION['e_email'])) {  
        unset($_SESSION['e_email']);  
    }  
  
} else {  
    // echo "<br>" . "Nie ma takiego Loginu (email) w  
    // bazie" . "<br>";  
    $_SESSION['e_zlyemaillogowania'] = "Nie ma takiego  
    loginu (adresu email) w naszej bazie. Podałeś/aś  
    błędny email, spróbuj ponownie."  
    $testEmail = false;  
}
```

Rys. 34.21. Plik login.php – kod PHP (zmienne sesyjne)

Stworzonych zmiennych sesyjnych można użyć w kodzie HTML – wewnątrz formularza. Dodatkowy kod PHP wykorzystujący logikę:

```
if (isset( $ZmiennaSesyjna )  
    {  
        echo $ZmiennaSesyjna;  
    }
```

umożliwi wyświetlenie odpowiedniego komentarza na stronie WWW (rys. 34.22).

```
<form action="login.php" method="POST" class="loginPodHeader">  
    Login: <input type="text" placeholder="Your nick"  
    name="email">  
    <br>  
    <?php  
    if (isset($_SESSION['e_email']))  
    {  
        echo $_SESSION['e_email'];  
    }  
    ?>  
    <br>  
    <?php  
    if (isset($_SESSION['e_emailLogin']))  
    {  
        echo $_SESSION['e_emailLogin'];  
    }  
    ?>  
    <br>  
    <?php  
    if (isset($_SESSION['e_zlyemaillogowania']))  
    {  
        echo $_SESSION['e_zlyemaillogowania'];  
    }  
    ?>
```

Rys. 34.22. Widok kodu HTML, wewnątrz znacznika **<form>** – zaprogramowanie warunkowego wyświetlania zmiennych sesyjnych

Zmienne sesyjne zostały stworzone po to, aby wyświetliły się, gdy użytkownik popełni błąd. Wówczas uzyska informację (pod danym polem input), co zrobił źle (rys. 34.23).

The screenshot shows a web browser window displaying the SOAW portal login page. The page has a green header with the text "SOAW portal ogłoszeniowy". Below the header are three buttons: "Home", "Rejestracja", and "Logowanie". The "Logowanie" button is highlighted. Below the buttons are two input fields: "Login: Your nick" and "Hasło: Twoje hasło". Below the "Login" field is a red-bordered box containing the text "Login - musi to być poprawny adres email!". Below the "Hasło" field is a red-bordered box containing the text "Hasło musi posiadać od 8 do 12 znaków!". Below these fields are two buttons: "Rejestracja" and "Zaloguj". At the bottom of the page, the text "SOAW" is displayed in green, followed by "Serwis Ogłoszeniowy Aplikacja Webowa" in brown.

Rys. 34.23. Widok w przeglądarce strony login.php po wprowadzeniu niepoprawnych loginu i hasła – zmienne sesyjne umożliwiają wyświetlenie komunikatu ostrzegawczo-informującego

34.8. Sprawdzenie poprawności hasła

Logika dla tego etapu wygląda tak:

- sprawdź poprawność hasła dla danego użytkownika;
- jeżeli hasło jest poprawne, przejdź do etapu 4;
- jeżeli hasło nie jest poprawne, poproś o ponowne wprowadzenie hasła.

Żeby zweryfikować poprawność hasła dla danego użytkownika, musimy sprawdzić hasło użytkownika – czy dla danego adresu e-mail wprowadzone hasło jest takie samo, jak zapisano w tabeli '**user**' w kolumnie '**haslo**'.

34.9. Metody PDO: `prepare()`, `execute()`, `fetch()` i symbole zastępcze =?

Wspomnieliśmy już o mankamentach metody PDO `query()` powodujących, że nie jest – m.in. ze względów bezpieczeństwa – zalecana. Warto w takim razie poznać kolejne elementy PDO:

- metoda `prepare()`,
- metoda `execute()`,
- metoda `fetch()`,
- symbole zastępcze.

Poniżej znajdziesz kod pokazujący zastosowanie metody, która nie wysyła od razu do bazy całego zapytania (tak jak `query()`), tylko definiuje na początek jego szkielet.

```
$zapytanie = $polaczenie->prepare($zapytanie);
```

```
// HASŁO - PASSWORD - SPRAWDZENIE
if ($testEmail == true) {
    $checkPassword =
        $connection->prepare(
            'SELECT haslo FROM user WHERE email = ?');
    $checkPassword->execute([$email]);
    $passwordFromDb = $checkPassword->fetch();
}
```

Rys. 34.24. Plik login.php, kod PHP – PDO użycie do połączenie z bazą: `Prepare()`, `Execute()`, `Fetch()`, podstawiania

Zapytanie jest wykonywane dopiero po wywołaniu metody `execute()`:

```
$zapytanie->execute();
```

Jest to zupełnie inne podejście do omawianego tematu. Pozwala na dokonywanie niedostępnych dotychczas działań, jak np.:

- przygotowanie zapytania – bez jego natychmiastowego wysyłania;
- parametryzowanie zapytań.

Ważne, że nowe podejście pozwoli znacznie zaoszczędzić czas połączenia z serwerem bazy. Jeżeli mamy mały projekt, zyski będą niewielkie, ale gdy aplikacja się rozrośnie, każda sekunda będzie miała ogromne znaczenie.

Pora na element pomagający zabezpieczać przed dostępem niepożądanych osób do bazy danych z wykorzystaniem PDO, czyli na podmienienie dokonane w kodzie – popatrz na elementy podkreślone w kodzie PHP (rys. 34.25).

```
$checkPassword =  
    $connection->prepare(  
        'SELECT haslo FROM user WHERE email = ?');  
    $checkPassword->execute([$email]);  
    $passwordFromDb = $checkPassword->fetch();
```

Rys. 34.25. Plik login.php, kod PHP – PDO podstawienie (podczas wywołania egzekucji zapytania), podstawiany jest znak zastępczy

Pokazana technika nie jest jedyna, ale w naszym projekcie w zupełności wystarczy. Służy głównie bezpieczeństwu. Pamiętaj, że istnieją wersje bardziej zaawansowane.

Reszta kodu jest już prosta, pokażemy ją w kilku krokach, aby zrozumieć poszczególne zapisy. Najpierw logika sprawdzi, czy wpisane hasło jest takie samo jak hasło w bazie (rys. 34.26).

```
// LOGIN - EMAIL - SPRAWDZANIE  
if ($passwordFromDb) {  
    // Szukane hasło - sprawdzenie czy wpisane  
    // hasło jest takie samo jak hasło w bazie  
    $searchedPassword = $passwordFromDb[0];  
  
    // ustawiamy zmienną  
    $_SESSION['loggedInToApp'] = true;  
    header('location: loggedInUser.php');
```

Rys. 34.26. Plik login.php, kod PHP – sprawdzenie, czy wpisane hasło jest takie samo jak hasło w bazie

Jeżeli hasła się zgadzają, wówczas ustawiamy zmienną informującą o sukcesie i za pomocą funkcji PHP **header()** przenosimy użytkownika pod określony adres url, w tym przypadku localhost/soaw/loggedInUser.php. Stosowny plik trzeba oczywiście utworzyć w głównym folderze projektu SOAW. Pozostaje kontynuacja „napoczętego” **if()** { (rys. 34.27).

```
} else {  
    $_SESSION['e_falsePasswordInDatabase'] =  
        "Błąd! Podałeś/aś błędne hasło, spróbuj  
        ponownie."  
}
```

Rys. 34.27. Plik login.php, kod PHP – logika pokazująca, co zrobi program, gdy wpisane hasło nie będzie takie samo jak hasło w bazie

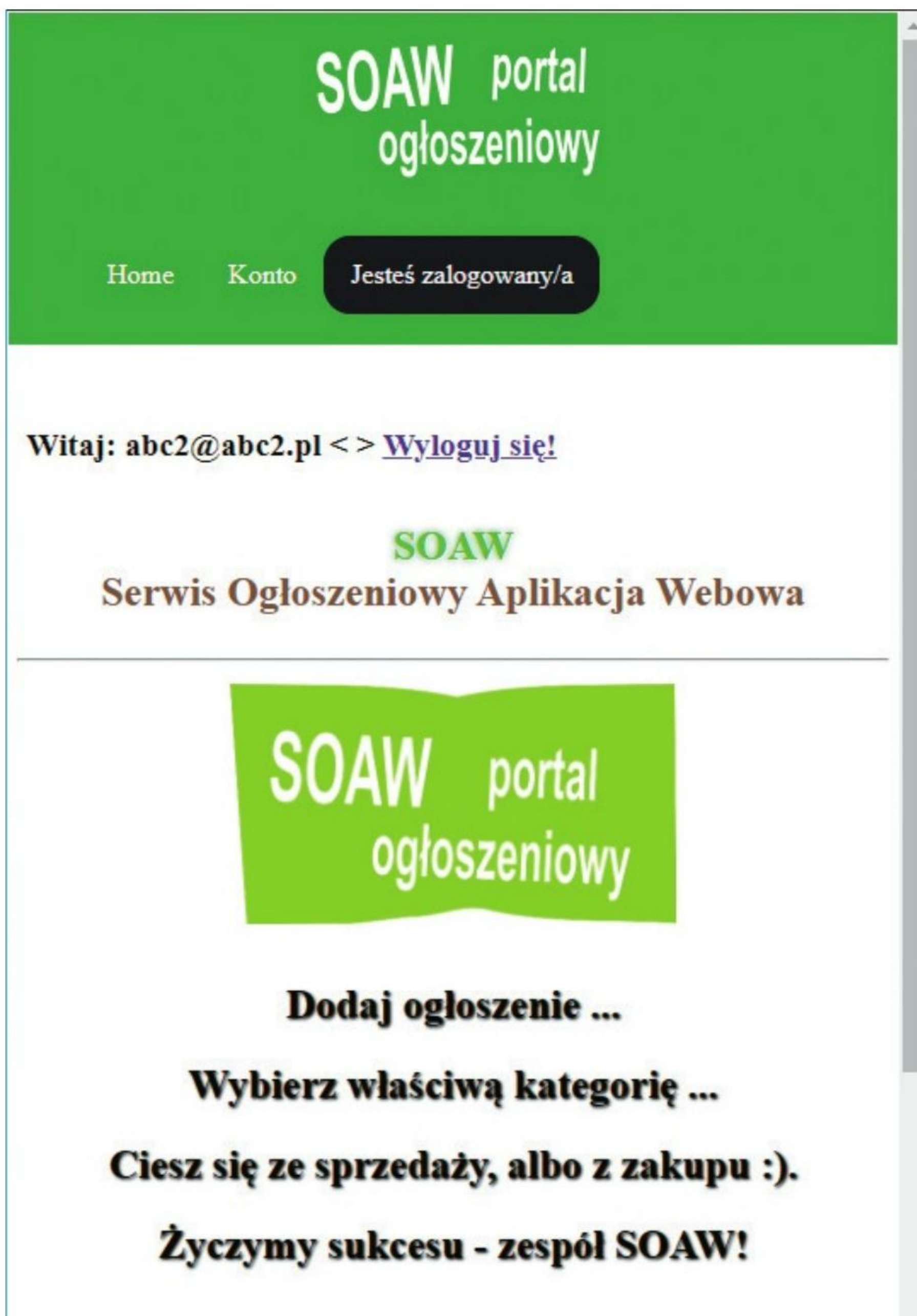
Spójrz na rys. 34.28. Kod z niego powinien znaleźć się na końcu bloku PHP. Gdy kod będzie już w całości wpisany, trzeba się upewnić, że:

- wszystkie nawiasy się zamykają, czyli w każdym bloku kodu są nawiasy otwierający i zamykający;
- PHP został otwarty na początku pliku znakiem **<?php** i zamknięty przed kodem HTML, czyli **?>**.

```
}  
}  
} else {  
    $_SESSION['e_email'] = 'brak Login - wprowadź poprawny  
    login, którym musi być Twój adres email zarejestrowany  
    w naszej bazie';  
}  
?>
```

Rys. 34.28. Plik login.php, kod PHP – logika pokazująca, co zrobi program, gdy nie będzie wpisany poprawny login

Po poprawnym zalogowaniu aplikacja musi poinformować użytkownika, że jest zalogowany – np. przeniesie na określoną stronę, dostępną tylko dla użytkowników zalogowanych: loggedInUser.php. Zadaniem programisty będzie samodzielnie zapisać taki kod w nowo utworzonym pliku, aby zapamiętał ustawienia i pokazywał, że użytkownik jest zalogowany. Efekt ma być mniej więcej taki jak na rys. 34.29.



Rys. 34.29. Widok w przeglądarce internetowej pliku loggedUser.php

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij pojęcie konfiguracji środowiska połączeniowego w PHP i SQL.
2. Co to jest PDO?
3. Opisz zasady połączenia frontendu z backendem. Przedstaw to na kartce papieru jako pseudokod.
4. Dlaczego testowanie pisanych kawałków kodu powinno stanowić nieodłączną pracę programisty? Odpowiedź uzasadnij.
5. Rozpisz na kartce papieru strukturę instrukcji PHP try-catch.
6. Wy tłumacz ideę połączenia z bazą danych za pomocą metody PDO **query()**.
7. Wy tłumacz zastosowaną w SOAW logikę sprawdzenia poprawności:
 - loginu,
 - hasła.
8. Wyjaśnij, do czego służą metody PDO:
 - **prepare()**,
 - **execute()**,
 - **fetch()**.
9. Wyjaśnij zasady stosowania symboli zastępczych typu **=?** w PHP i PDO.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przygotuj w języku programowania PHP skrypt sprawdzający, która jest godzina. Następnie dodaj do niego funkcjonalność, aby:
 - A. w godzinach 00.00 – 12.00 odmówił połączenia z bazą danych, a na stronie pojawiła się informacja napisana wielkimi literami 'TERAZ ŚPIĘ – PROSZĘ NIE PRZESZKADZAĆ, WRÓĆ PO 12.00';
 - B. w godzinach 12.01 – 23.59 połączył się z bazą danych, a na stronie pojawiła się informacja typu: 'PÓJDE SPAĆ O GODZINIE 00.00, czyli za „xx” minut’.
2. Przerób skrypt z zadania 1. w taki sposób, aby w miejscu „xx” pojawiało się automatyczne odliczanie minut do czasu zamknięcia połączenia („pójścia spać przez skrypt”).
3. Przygotuj system logowania w języku programowania PHP:
 - A. na kartce papieru (lub w edytorze tekstu) napisz pseudokod, który będzie odpowiedzialny za system logowania do aplikacji webowej;
 - B. przełóż stworzony opisowy kod na skrypt PHP. Możesz wykorzystać dowolną bazę danych z XAMPPA (lub zupełnie nową). Pamiętaj o niezbędnych elementach, np.:
 - sanityzacja loginu,
 - walidacja loginu;
 - C. przetestuj, czy przygotowany program działa właściwie;
 - D. jeżeli nie działa, to najpierw samodzielnie wyszukaj błędy;
 - E. gdy nie będziesz w stanie uruchomić logowania, zajrzyj do kodów zawartych w podręczniku i uruchom własny system logowania.

4. Napisz w języku programowania PHP skrypt, który będzie na danej stronie WWW:
- sprawdzał, ile osób jest na niej obecnych;
 - sumował obecność przez cały dzień;
 - zapisywał do bazy danych typu SQL na koniec dnia sumę osób, która odwiedziła stronę;
 - wyświetlał, ile osób było na stronie poprzedniego dnia.

Pamiętaj, aby najpierw zaprojektować i stworzyć bazą danych typu SQL (np. w MariaDB), którą wykorzystasz w opisanym projekcie.

35

Interakcja w aplikacji SOAW – „ciasteczkowy projekt”, kodowanie vanilla JavaScript

ZAGADNIENIA

- Ciasteczka, RODO – podstawy
- Dodawanie skryptów JavaScript
- Zdarzenia na stronie internetowej
- JavaScript – kodowanie ciasteczek
- JavaScript – kodowanie logiki wyświetlania ciasteczek

35.1. Ciasteczka, RODO – podstawy

RODO to potoczna nazwa dokumentu prawnego obowiązującego w Unii Europejskiej. Dotyczy ochrony danych osobowych (danych osób fizycznych). Wprowadzono je w krajach Unii Europejskiej 25 maja 2016 r. na podstawie Rozporządzenia Parlamentu Europejskiego I Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych). Faktyczne stosowanie zasad RODO datuje się jednak na 25 maja 2018 r. Dwuletni okres przejściowy między wspomnianymi datami miał służyć dostosowaniu przepisów krajowych państw członkowskich Unii do nowego prawa.

Przepisy związane z RODO powinny chronić dane osobowe osób fizycznych. Nie będziemy omawiać szczegółów związanych z ustawą o RODO, z naszego punktu widzenia liczy się samo pojęcie ochrony danych osobowych, co ma olbrzymie znaczenie w internecie, gdyż ten obszar ludzkiego życia jest narażony na wiele zagrożeń. Ponadto są to zagrożenia, których nie widać; mimo że wirtualne, to potencjalne straty przez nie powodowane są już bardzo realne.

Jednym z wymogów wynikających z wymienionego rozporządzenia jest informowanie użytkowników stron internetowych o możliwym przechwytywaniu i/lub przechowywaniu ich danych. W przypadku aplikacji SOAW skupimy się na tzw. ciasteczkach, które stanowią potencjalne zagrożenie dla każdego internauty, gdyż mogą przechwytywać różnego rodzaju informacje istotne z punktu widzenia prywatności osób fizycznych. Skupimy się na dostosowaniu SOAW do RODO przez umieszczenie właściwej informacji na naszej stronie. Opracujemy specjalny tekst, który poinformuje użytkownika o tym, że SOAW używa technologii cookies, czyli ciasteczek. Całość „ciasteczkowego projektu” zrealizujemy w JavaScript.

4.5.2016

PL

Dziennik Urzędowy Unii Europejskiej

I

(Akty ustawodawcze)

ROZPORZĄDZENIA

ROZPORZĄDZENIE PARLAMENTU EUROPEJSKIEGO I RADY (UE) 2016/679

z dnia 27 kwietnia 2016 r.

w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych)

(Tekst mający znaczenie dla EOG)

PARLAMENT EUROPEJSKI I RADA UNII EUROPEJSKIEJ,

uwzględniając Traktat o funkcjonowaniu Unii Europejskiej, w szczególności jego art. 16,

uwzględniając wniosek Komisji Europejskiej,

po przekazaniu projektu aktu ustawodawczego parlamentom narodowym,

uwzględniając opinię Europejskiego Komitetu Ekonomiczno-Społecznego ⁽¹⁾,uwzględniając opinię Komitetu Regionów ⁽²⁾,stanowiąc zgodnie ze zwykłą procedurą ustawodawczą ⁽³⁾,

Rys. 35.1. Strona tytułowa Rozporządzenia Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych)

W przypadku dodatkowego zainteresowania, czym są ciasteczka, możesz skorzystać z materiałów dodatkowych (np. https://pl.wikipedia.org/wiki/HTTP_cookie). Na MDN znajduje się bardzo dobre opracowanie, w którym przedstawiono omawiane zagadnienie w nieco bardziej fachowy (zaawansowany) sposób (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>).

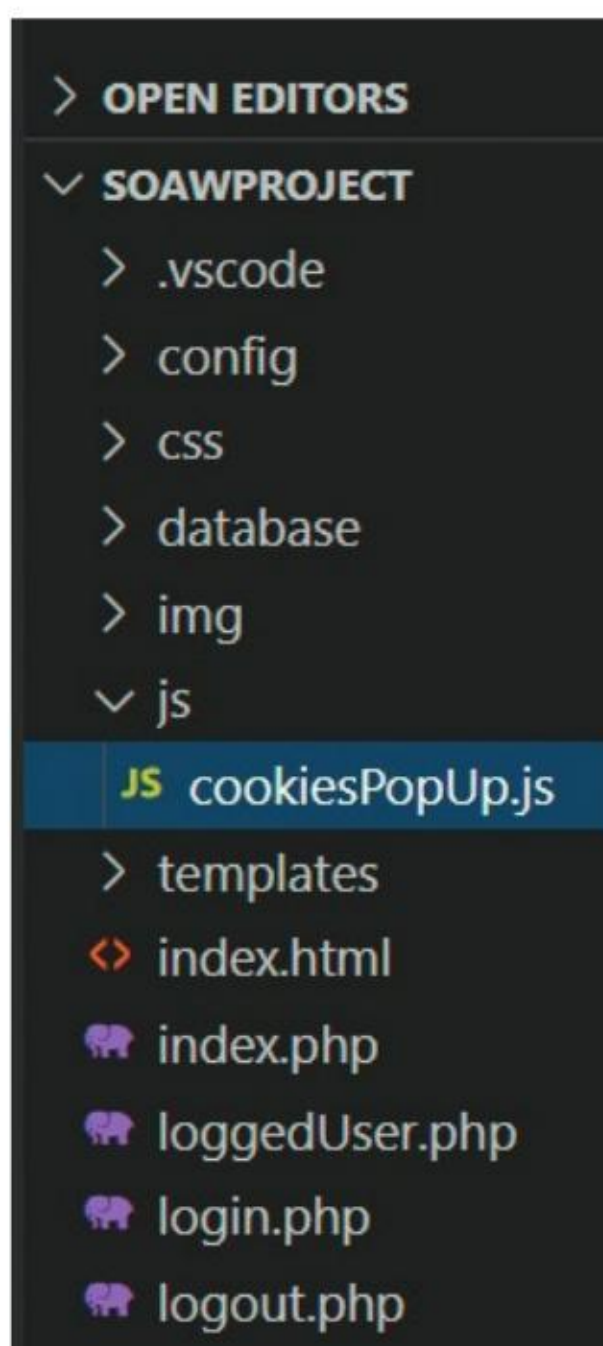


Rys. 35.2. Zdjęcie poglądowe – „ciasteczka”

35.2. Dodawanie skryptów JavaScript

Pracę nad „ciasteczkowym projektem” rozpoczniemy od stworzenia w katalogu głównym naszego projektu:

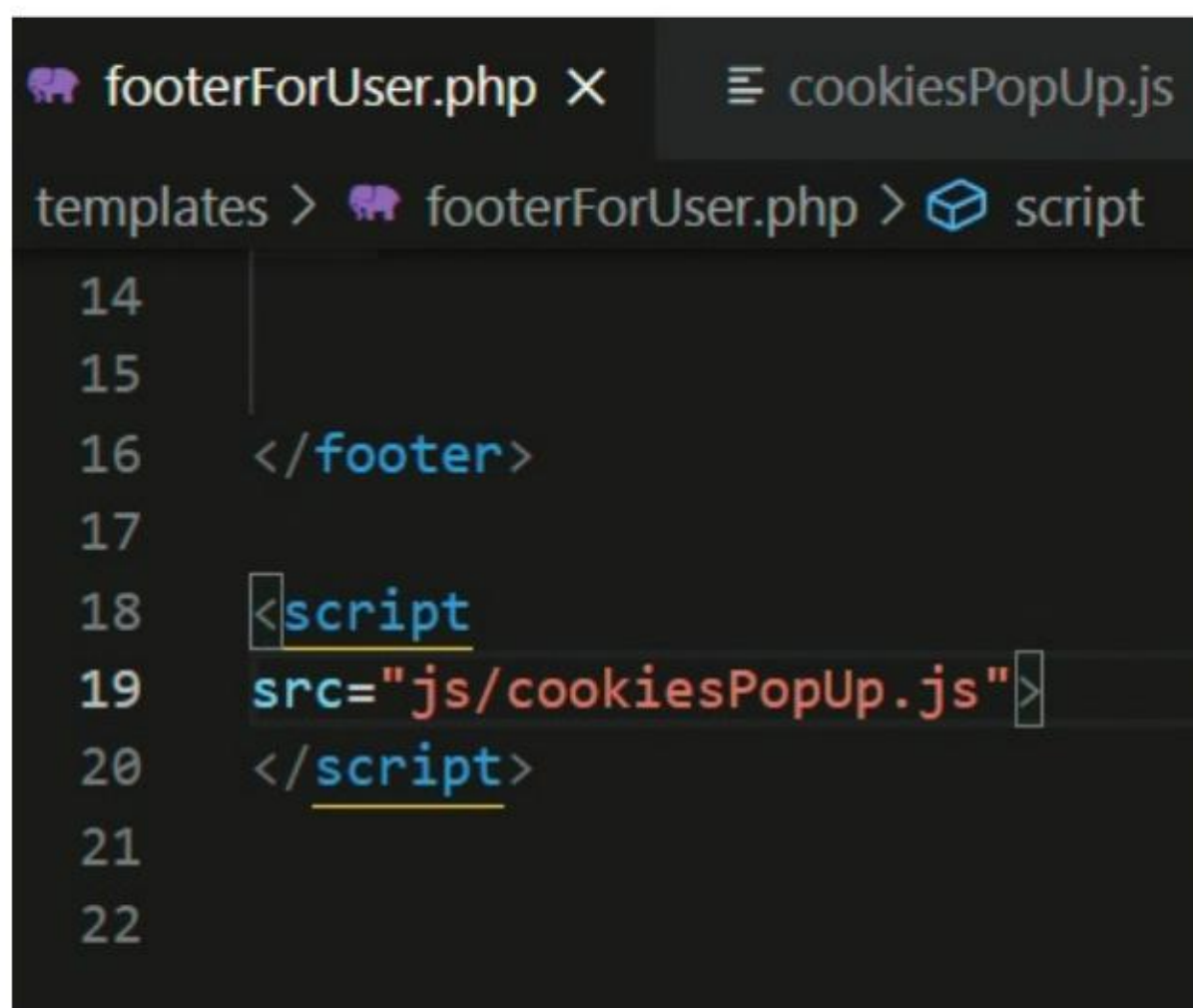
- katalogu o nazwie – ‘js’,
- pliku wewnątrz katalogu ‘js’, o nazwie cookiesPopUp.js.



Rys. 35.3. Widok struktury katalogów i plików projektu SOAW – zaznaczono nowy plik cookies PopUp.js

Aby kod JavaScript zadziałał, musimy go dodać do strony internetowej. Zanim zaczniemy programować „ciasteczkowy projekt”, proponuję zapoznanie się z niewielką ilością teorii dotyczącej JavaScript. Jest to potężny i bardzo nowoczesny język programowania, wykorzystywany zarówno po stronie frontendu, jak i backendu. Na tym etapie naszego projektu będziemy wykorzystywali cechy tego języka jedynie do programowania interakcji na stronach internetowych, czyli będzie to typowy frontend.

Plik, w którym będziemy pisali kod JavaScript, już utworzyliśmy. Aby na naszej stronie WWW zadziałały kody w nim zawarte, musimy we właściwy sposób podlinkować wskazany plik do tworzonego projektu. Wykonamy linkowanie na dole strony tuż przed body. Związane jest to bezpośrednio ze specyfiką przeglądarek internetowych i tzw. silnikiem JavaScript. Skrypt JS musi mieć dostęp do wszystkich elementów strony, czyli powinien być aktywowany dopiero po załadowaniu całej struktury drzewa DOM. Gdyby tak się nie stało, otrzymalibyśmy błąd. Błąd JS to poważna sprawa – bardzo często jeden powoduje awarię całej strony internetowej.



```
footerForUser.php X cookiesPopUp.js
templates > footerForUser.php > script
14
15
16 </footer>
17
18 <script
19   src="js/cookiesPopUp.js">
20 </script>
21
22
```

Rys. 35.4. Podlinkowanie pliku JavaScript wewnątrz pliku footerForUser.php do realizowanego projektu

DOM

Model obiektowy dokumentu jest API dla dokumentów HTML i XML, czyli budowy strony internetowej z drzewa znaczników HTML. Zrozumienie struktury DOM jest niezmiernie istotne w programowaniu w JavaScript. Warto sięgnąć do materiałów szczegółowych na temat DOM, np.:

- <https://www.w3.org/DOM/DOMTR#dom3>,
- <https://dom.spec.whatwg.org/>,
- https://developer.mozilla.org/pl/docs/Dokumentacja_Gecko_DOM.

Aby mieć pewność, że wszystkie elementy strony WWW już się załadowały (zanim zadziała skrypt JS), linkowanie do pliku JavaScript umieścimy na dole strony. Otwórz plik `footerForUser.php`, a następnie dodaj w nim na samym dole (pod znacznikiem zamykającym blok stopki – `</footer>`) taki kod jak na rys. 35.4.

Wróćmy jeszcze do aktywowania (linkowania) kodów JavaScript na stronie internetowej. Możemy to wykonać na kilka sposobów. Poniżej opisano podstawowe z nich:

- JavaScript jako załącznik – dołączamy oddzielny plik z rozszerzeniem `.js` w sekcji body HTML-a – umieszczany tuż przed końcem ciała strony, czyli przed znacznikiem `</body>`:

```
<script src="skryptJavaScript.js"></script>
```

Rozwiązanie to jest prawdopodobnie najczęściej stosowane, choć w ostatnich czasach nie aż tak często (głównie za sprawą `async` i `defer`). Przed skorzystaniem z niego powinna się już załadować strona (choć pełnej gwarancji nie ma), więc skrypt może się odwoływać do elementów HTML i CSS załadowanych wcześniej w przeglądarce;

- JavaScript jako załącznik – dołączamy oddzielny plik z rozszerzeniem `.js` w sekcji `head` HTML-a:

```
<script src="skryptJavaScript.js"></script>;
```

- skrypt umieszczany w znaczniku:

```
<script> tutaj będzie kod JavaScript </script>
```

w podobnym miejscu jak w pierwszym punkcie, czyli tuż przed znacznikiem zamykającym – `</body>`;

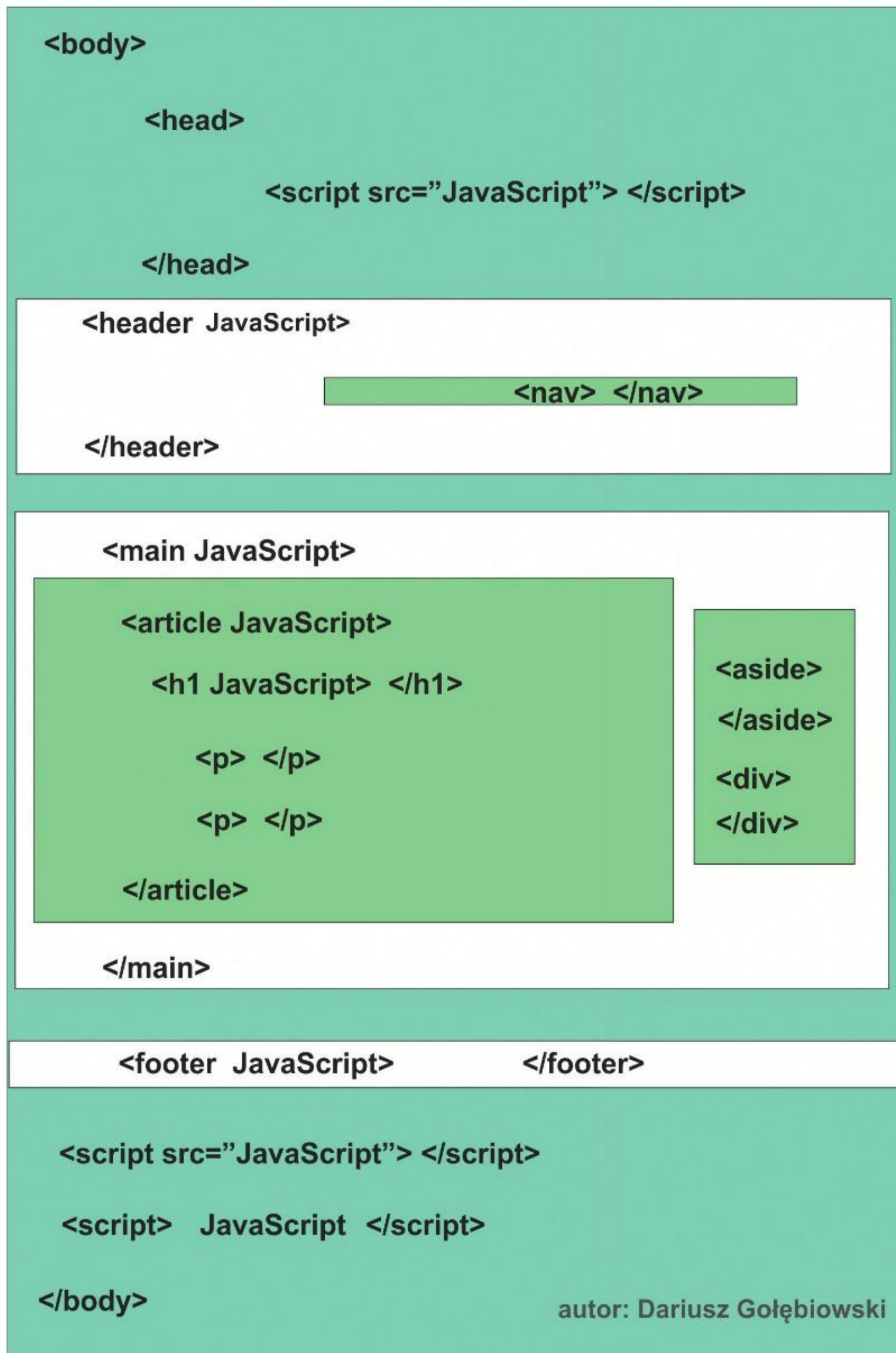
- skrypt lub właściwie pojedyncze polecenia umieszczamy wewnątrz znaczników, np.:

```
<div onclick = "location.href='./templates';" onmouseover =  
"firstPopUp(this)" onmouseout="firstPDown(this)" id="popUpEle-  
ment" class="popUpElement">..  
</div>.
```

CIEKAWOSTKA

JavaScript ma wiele wbudowanych metod, dzięki którym można nie tylko pobierać poszczególne elementy z drzewa HTML – DOM, lecz także tworzyć je od podstaw. Służy do tego m.in. metoda `createElement()`. Poniżej przedstawiono jej przykładowe zastosowanie – funkcja umożliwi stworzenie znacznika `div`:

```
function dodajDiv() {  
    const nowyDiv = document.createElement("div");  
    const nowyContent = document.createTextNode("Witaj w div  
    stworzonym przez JavaScript");  
    nowyDiv.appendChild(nowyContent);  
}
```

Rys. 35.5. Umieszczanie elementów języka JavaScript w ramach języka znaczników HTML

35.3. Zdarzenia na stronie internetowej

Gdy powstawał internet, tworzone pierwsze strony WWW. Początkowo zawierały one wyłącznie tekst i wizualnie nie były interesujące. Później do tekstu dołączano obrazki i zdjęcia. Wizualnie atrakcyjne i bardzo przydatne dla użytkowników powstały dopiero wtedy, kiedy na stronach WWW zagościły zdarzenia i obsługujące je JavaScript.

 Home Rejestracja Logowanie	• Home • Rejestracja • Logowanie SOAW Serwis Ogłoszeniowy Aplikacja Webowa
SOAW Serwis Ogłoszeniowy Aplikacja Webowa	Jak to działa? Załącz konto! Bezpłatna rejestracja! Dodaj za darmo ogłoszenie!
 Jak to działa? Załącz konto! Bezpłatna rejestracja! Dodaj za darmo ogłoszenie! Login: Twój email Hasło: Twoje hasło Rejestracja Zaloguj	Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi eum illum deleniti, voluptate sunt, enim repudiandae exercitationem dolore corporis dolores possimus dolorum veritatis perspiciatis ullam magni assumenda eos ad obcaecati. SOAW www.adresSOAW.pl Serwis Ogłoszeniowy Aplikacja Webowa ☎ GSM 500 xxx yyy ✉ biuro@adresSOAW.pl © 2021 powered by SOAW

Rys. 35.6. Strona projektu SOAW: po prawej – wykonana w najstarszej technologii, czyli jest to zwykły tekst; po lewej – zawiera style CSS, obrazki, elementy aktywne – linki i pola input

Zdarzenia to na stronie internetowej podstawa interakcji webowych. Dzięki nim można prawdopodobnie oprogramować każde zachowanie użytkownika strony i/lub wynik jego zachowania.

Zdarzenia WWW można zdefiniować jako czynności, które użytkownik wykonuje podczas odwiedzin na stronie. Najczęściej wykonywane są te związane z operowaniem myszką komputera, np.:

- scrollowanie strony internetowej,
- przesunięcie kursora myszki nad obrazek,
- kliknięcie w zdjęcie, link, dany obszar HTML (np. jakiegoś diva) czy menu,
- aktywowanie formularza przez ustawienie na nim kursora myszki, wysłanie formularza.


```
var batonStartOdID = document.getElementById  
('batonStartId');  
batonStartOdID.style.width = szerokoscOkna * 0.15 + 'px';  
batonStartOdID.style.height = wysokoscOkna * 0.1 + 'px';  
if (true)  
{  
    var strusiek1IdVar = document.getElementById  
('strusiek1Id');  
  
    batonStartOdID.addEventListener('click', przedAnimuj);  
}
```

Rys. 35.7. Zdarzenie – przykład kodu JavaScript, który po kliknięciu w postać gry spowoduje uruchomienie animacji

Możliwych zdarzeń są setki, oczywiście nie wszystkie są związane z użyciem myszki czy jej przycisków. Istnieje wiele innych, które będą aktywowane przez użytkownika, np. związane z czynnościami:

- uruchomienie animacji slidera,
- załadowanie strony internetowej,
- naciśnięcie określonego klawisza klawiatury,
- zakończenie animacji CSS,
- przesuwanie strony za pomocą strzałek na klawiaturze.

Wykaz wszystkich zdarzeń, które może obsłużyć JavaScript, znajdziesz na stronie MDN. Na rys. 35.8 przedstawiono przykładowe kategorie zdarzeń.

Jest ich bardzo dużo, ale w praktyce używa się najczęściej zaledwie kilkunastu (co najwyżej kilkudziesięciu). Co ważne, niemal każdy większy skrypt internetowy obsługujący interakcję nie może obejść się bez zdarzeń. To one kreują interaktywność, tak bardzo pożądaną zarówno przez właścicieli witryn, jak i ich klientów (użytkowników internetu).



CIEKAWOSTKA

Poniżej przedstawiono sposób na dodanie nowo powstałego elementu – nowyDiv – przed istniejący element strony:

```
const istniejacyElementBody = document.getElementById("istniejacyId");  
document.body.insertBefore(nowyDiv, istniejacyElementBody);
```


Most common categories

Resource events

Event Name	Fired When
<code>error</code>	A resource failed to load.
<code>abort</code>	The loading of a resource has been aborted.
<code>load</code>	A resource and its dependent resources have finished loading.
<code>beforeunload</code>	The window, the document and its resources are about to be unloaded.
<code>unload</code>	The document or a dependent resource is being unloaded.

Network events

Event Name	Fired When
<code>online</code>	The browser has gained access to the network.
<code>offline</code>	The browser has lost access to the network.

Focus events

Event Name	Fired When
<code>focus</code>	An element has received focus (does not bubble).
<code>blur</code>	An element has lost focus (does not bubble).
<code>focusin</code>	An element is about to receive focus (does bubble).
<code>focusout</code>	An element is about to lose focus (does bubble).

WebSocket events

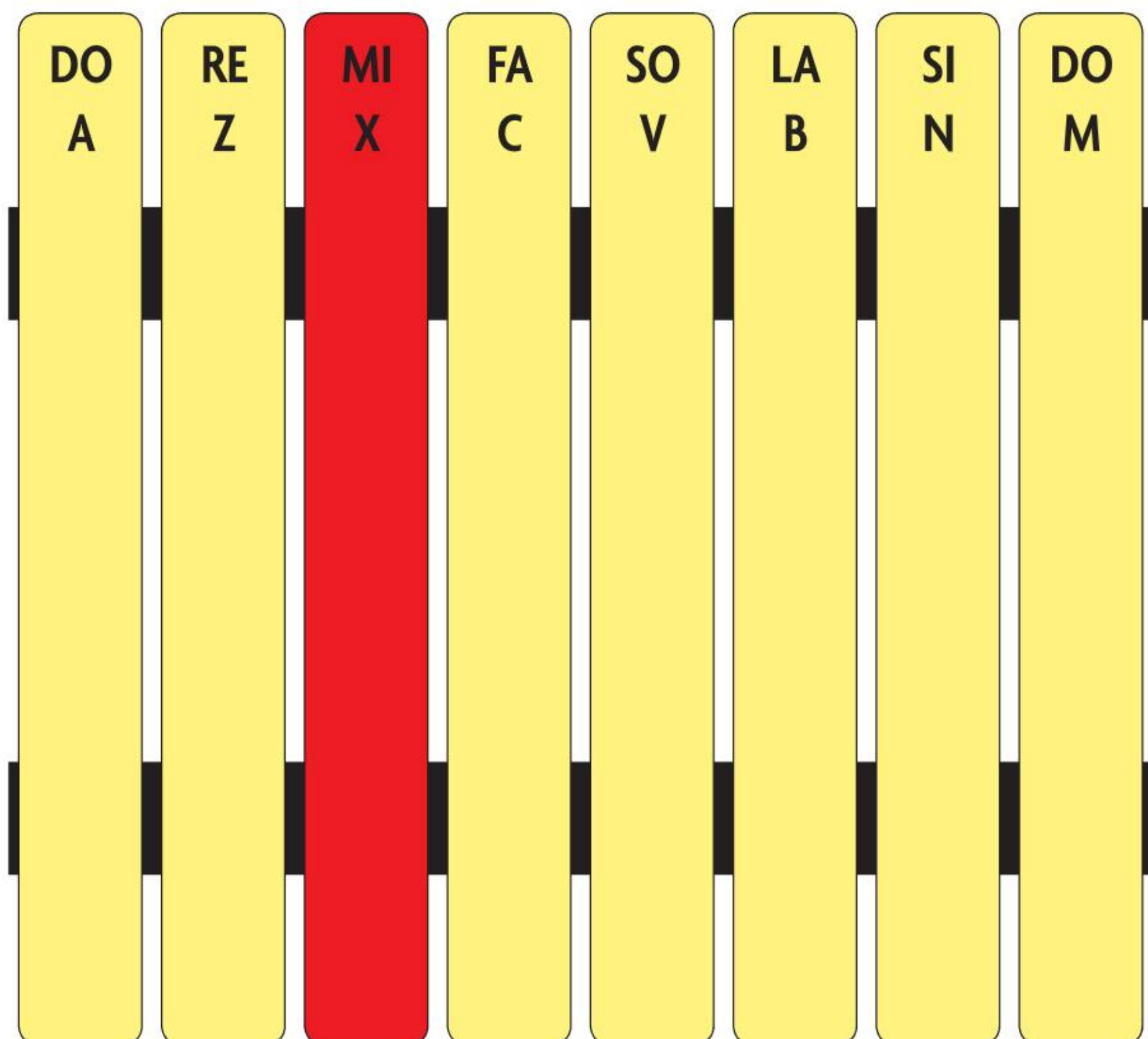
Event Name	Fired When
<code>open</code>	A WebSocket connection has been established.
<code>message</code>	A message is received through a WebSocket.
<code>error</code>	A WebSocket connection has been closed with prejudice (some data couldn't be sent for example).
<code>close</code>	A WebSocket connection has been closed.

Session History events

Event Name	Fired When
<code>pagehide</code>	A session history entry is being traversed from.
<code>pageshow</code>	A session history entry is being traversed to.
<code>popstate</code>	A session history entry is being navigated to (in certain cases).

Rys. 35.8. Zdarzenia – przykład kilku kategorii

Źródło: <https://developer.mozilla.org/en-US/docs/Web/Events> [dostęp: 16.02.2020].



Rys. 35.9. Zdarzenia – przykład interakcji: gra Cymbałki, kliknięcie myszką w elementy instrumentu lub naciśnięcie właściwego klawisza spowodują uruchomienie dźwięków

Stworzony przez programistę kod JavaScript, nazywany po prostu skryptem, jest uruchamiany w tradycyjnym środowisku internetowym w wyniku zaistnienia jakiegoś zdarzenia, dlatego są one tak ważne w prawie każdym większym projekcie internetowym.

35.4. JavaScript – kodowanie ciasteczka

Na następnych stronach znajdują się informacje o zdarzeniach, które są obsługiwane w dużej mierze za pomocą metody `addEventListener()`. Wiele osobom mającym do czynienia z językiem JavaScript wydaje się, że tylko ta metoda jest podstawą w obsłudze zdarzeń na stronie WWW. W naszym projekcie podejście będzie nieco odmienne. Nie zastosujemy właściwie zdarzeń, poza trzema przyciskami. Cały „ciasteczkowy projekt” zbudujemy wyłącznie za pomocą JavaScript, bez CSS-ów i HTML-a.

Zanim przejdziemy do kodowania, musimy wyjaśnić sposób tworzenia wykorzystywanych nazw (funkcji, zmiennych itp.).

Po pierwsze, zwykle tworzymy nazwy według metodologii camelCase (rys. 35.10), czyli używamy systemu notacji ciągów tekstowych, który łączy ze sobą kolejne wyrazy, tylko pierwszy wpisując małą literą, a każdy następny – wielką literą.


```

if (klawiatura[1] == kliknietyKlawisz)
{
    poszukiwanyNumerek = 1 * 1;
    console.log('to jest 1:' + 1);
    console.log('to jest poszukiwanyNumerek:' + poszukiwanyNumerek);

    var dzwiek = ['sounds/cymbal1.mp3', 'sounds/cymbal2.mp3',
        'sounds/cymbal3.mp3', 'sounds/cymbal4.mp3', 'sounds/cymbal5.
        mp3', 'sounds/cymbal6.mp3', 'sounds/cymbal7.mp3', 'sounds/
        cymbal8.mp3'];
    var audio = new Audio(dzwiek[poszukiwanyNumerek]);
    audio.play();

    klawiszeAll[poszukiwanyNumerek].style.background = 'red';
}
else
{
    infoZlyKlawisz.innerHTML = "The rights keys with sounds that
    are:" + "<br>" + "DO: A; RE: Z; MI: X; FA: C; SO: V; LA: B; SI:
    N; DO: M";
};

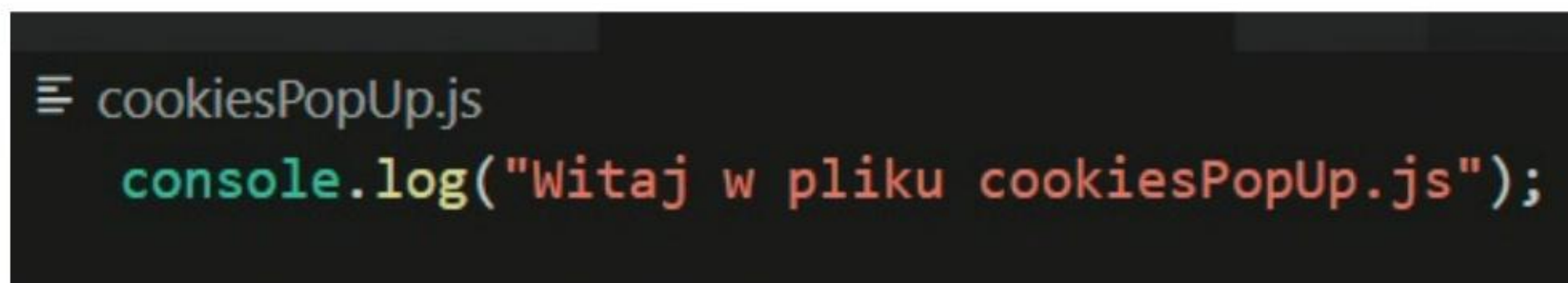
```

Rys. 35.10. Kod JavaScript, notacja camelCase (podkreślono przykładowe nazwy zgodne z tą notacją)

Po drugie, używamy nazw opisowych, które czasami mogą się wydawać zbyt długie. Staramy się nazywać funkcje i zmienne (a czasami także inne elementy kodu JavaScript) tak, aby można było z samej nazwy wywnioskować, do czego dany element kodu służy, co robi, jakie jest jego podstawowe działanie.

Na początek sprawdźmy, czy plik `cookiesPopUp.js` jest właściwie podlinkowany do naszego projektu. W tym celu wpisz w pliku polecenie, które będzie wywołane przez konsolę:

```
console.log(„Witaj w pliku cookiesPopUp.js”);
```



Rys. 35.11. Kod JavaScript – sprawdzenie w konsoli poprawności konfiguracji środowiska

Po zapisaniu pliku otwórz stronę projektu `http://localhost/soaw/` w swojej ulubionej przeglądarce internetowej (serwer i bazę powinieneś już mieć uruchomione). Naciśnij F12, by uruchomić panel dewelopera. W konsoli powinien pojawić się napis „Witaj w pliku `cookiesPopUp.js`”. Jeżeli efekt jest podobny do tego na rys. 35.12, to możesz wyczyścić zawartość pliku `cookiePopUp.js`.



Rys. 35.12. Kod JavaScript – sprawdzenie w konsoli poprawności konfiguracji środowiska

Zapiszemy pierwszą linię kodu, przeznaczoną do realizacji „ciasteczkowego projektu”. Stworzymy zmienną `divCookies`, przypisując do niej metodę JavaScript, która stworzy zupełnie nowy element blokowy HTML – **div** (rys. 35.13).

Kolejnym krokiem będzie stworzenie funkcji z komentarzem w środku. Dalsze elementy funkcji będziemy umieszczali pod komentarzem.


```
≡ cookiesPopUp.js  
var divCookies = document.createElement("div");
```

Rys. 35.13. Kod JavaScript – stworzenie bloku div, czyli elementu HTML

```
function createcookiesDivEndBaton() {  
    // Jeżeli nie ma ciasteczka - tworzymy wyskakującego div'a  
    z batonem  
}
```

Rys. 35.14. Kod JavaScript – pusta funkcja z komentarzem

Zanim napiszemy kolejne bloki kodu, pokażemy, jak ma wyglądać efekt. Co istotne, podczas programowania musimy pamiętać o wyglądzie projektu (kolory, czcionki, ułożenie na stronie itp.). Ma to być wyskakujący nad stronę element graficzny:

- zlokalizowany na dole strony,
- możliwy do zamknięcia przez akceptację ciasteczek,
- przekazujący informację o możliwości przejścia do podstrony opisującej politykę RODO portalu SOAW,
- umożliwiający osobom, które nie wiedzą, czym są ciasteczka, przejście do innej strony, na której będą mogły uzyskać stosowną wiedzę.

Na rys. 35.15 przedstawiono, jak powinien wyglądać efekt naszej pracy.



Rys. 35.15. Kod JavaScript – stworzenie bloku div, czyli elementu HTML

Kolejna część kodu Kod JavaScript spowoduje przypisanie do zmiennej textCookies bloku z tekstem przez wykorzystanie metody createTextNode(). Stworzyliśmy w ten sposób tekst, który wyświetli się na naszej stronie w informacji o cookies. Oczywiście możesz w to miejsce wstawić dowolny tekst.

Wykorzystując wcześniej stworzone w kodzie JavaScript elementy HTML, dodamy do strony projektu SOAW z wykorzystaniem metody `appendChild()` dwa elementy:

- `divCookies.appendChild(textCookies)` – tekst do stworzonego wcześniej diva (tzn. do zmiennej `divCookies`),
- `document.body.appendChild(divCookies)` – do dokumentu, czyli strony internetowej, cały `divCookies`, tzn. wraz z zawartością.

```
var textCookies = document.createTextNode('Drogi  
użytkowniku, aplikacja SOAW używa technologii  
cookies, czyli tzw. ciasteczek. Korzystając z SOAW,  
wyrażasz zgodę na ich używanie, ale dzieje się to  
zgodnie z ustawieniami Twojej przeglądarki. Możesz  
zmienić ww. ustawienia w dowolnym momencie.');
```

```
divCookies.appendChild(textCookies);  
document.body.appendChild(divCookies);
```

Rys. 35.16. Kod JavaScript – stworzenie zmiennej, która tworzy blok z tekstem (`createTextNode()`) i dodanie stworzonych elementów HTML do strony (`appendChild()`)

Następna linijka kodu JavaScript ustawi za pomocą metody `setAttribute()` atrybut `'id'` dla zmiennej `divCookies` na `,ciasteczkowyNapis'`. Kolejnych jedenaście wierszy kodu odpowiada za wygląd wyskakującego okienka z informacją o cookies. Po słowie „style” stawiamy kropkę, a po niej właściwość CSS zapisaną w camelCase (rys. 35.17).

```
divCookies.setAttribute('id', 'ciasteczkowyNapis');
divCookies.style.position = "fixed";
divCookies.style.fontSize = "17" + "px";
divCookies.style.bottom = "5" + "%";
divCookies.style.right = "5" + "%";
divCookies.style.maxWidth = "87" + "%";
divCookies.style.color = "3" + "%";
divCookies.style.background = "#FDFDFD";
divCookies.style.border = '0.4em solid green';
divCookies.style.zIndex = "100002";
divCookies.style.padding = "0.3" + "%";
divCookies.style.borderRadius = "0.3" + "em";
```

Rys. 35.17. Kod JavaScript – ustawienie atrybutu `id` i wyglądu dla stworzonej wcześniej zmiennej `divCookies`

Teraz napiszemy kod JavaScript, który stworzy przycisk potwierdzający włożenia ciasteczka do pamięci. Na rys. 35.15 będzie to widoczny na dole strony przycisk „Akceptuj”. Elementy kodu już tłumaczyliśmy, musisz teraz napisać, pozmienić, potestować, przekonać się, która metoda, zmienna, właściwość wykonują dane elementy naszego projektu.

```
// Baton potwierdzający włożenia ciasteczka do pamięć
var batonCookies = document.createElement("button");
var textBatonCookies = document.createTextNode("Akceptuj");
batonCookies.appendChild(textBatonCookies);
divCookies.appendChild(batonCookies);
batonCookies.setAttribute('id', 'dodajCiasteczko');
batonCookies.style.background = "#5A7B7C";
batonCookies.style.color = "#fff";
batonCookies.style.margin = "10" + "px";
```

Rys. 35.18. Kod JavaScript – stworzenie przycisku potwierdzającego włożenia ciasteczka do pamięci

Ostatnie dwa bloki kodu JavaScript w funkcji `createcookiesDivEndBaton()` stworzą dwa kolejne przyciski, odpowiednio:

- „Polityka Prywatności – zobacz, co robimy z ciasteczkami”;
- „Ciasteczka – informacje o technologii na Wikipedii”.

```
// baton polityka prywatności
var batonPolitykaPrywatnosci = document.createElement
("button");
var textBatonPolityka = document.createTextNode("Polityka
Prywatności - zobacz co robimy z ciasteczkami");
batonPolitykaPrywatnosci.appendChild(textBatonPolityka);
divCookies.appendChild(batonPolitykaPrywatnosci);
batonPolitykaPrywatnosci.setAttribute('id',
'pokazPolitykaPrywatnosci');
batonPolitykaPrywatnosci.style.margin = "10" + "px";

// baton Więcej o ciasteczkach
var batonCiasteczkaWiecej = document.createElement
("button");
var textBatonPolityka = document.createTextNode
("Ciasteczka- informacje o technologii na Wikipedii");
batonCiasteczkaWiecej.appendChild(textBatonPolityka);
divCookies.appendChild(batonCiasteczkaWiecej);
batonCiasteczkaWiecej.setAttribute('id',
'pokazCiasteczkaWiecej');
batonCiasteczkaWiecej.style.margin = "10" + "px";
```

Rys. 35.19. Kod JavaScript – zaprogramowanie przycisków „Polityka Prywatności – zobacz, co robimy z ciasteczkami” i „Ciasteczka – informacje o technologii na Wikipedii”

Zakończyliśmy budować wygląd „ciasteczkowego projektu”. Proponuję wywołać stworzoną funkcję, żeby zobaczyć efekty pracy.

Wywołanie funkcji w języku JavaScript:

```
createcookiesDivEndBaton();
```

Wynikiem powinien być gotowy wygląd „ciasteczkowego projektu”. Po kliknięciu w przyciski niewiele się jednak na stronie dzieje... Uzupełnimy to w kolejnej części podręcznika.

35.5. JavaScript – kodowanie logiki wyświetlania ciasteczek

W poprzedniej lekcji zaprogramowaliśmy za pomocą JavaScript wizualną część „ciasteczkowego projektu” wraz z przyciskami i tekstem informacyjnym dla użytkownika SOAW. Pora na wykonanie funkcjonalności obsługującej zdarzenia. Najpierw wyjaśnijmy mechanizm działania planowanego kodu. Praca będzie się składała z kilku etapów:

1. Stworzymy zmienną o określonej nazwie, która będzie nazwą ciasteczka.
2. Sprawdzimy, czy ciasteczko o takiej nazwie już jest zarejestrowane w przeglądarce:
 - gdy jest – oznacza to, że informacja o ciasteczkach, którą stworzyliśmy, nie może się wyświetlić,
 - gdy nie ma – informacja o ciasteczkach musi się wyświetlić.

Zacznijmy kodowanie. Przypiszemy nazwę „ciasteczkoSOAW” do zmiennej **ciasteczkoSzukane**. Potem stworzymy zmienną **const**, która w JavaScript oznacza typ zmiennej, która jest stałą. Nie ma takiego przymusu, ale przyjęło się, że w przypadku zmiennych **const** ich nazwy są pisane wielkimi literami (bądź chociaż pierwsza wielką). Zastosowaliśmy to drugie rozwiązanie. Do Cookies przypisaliśmy metodę **cookie**, która pobiera istniejące ciasteczko (rys. 35.20).

```
// Pobieranie COOKIES - sprawdzamy czy jest ciasteczko SOAW
// document.cookie - pobiera/ustawia listę ciasteczek
// powiązanych z dokumentem.
var ciasteczkoSzukane = "ciasteczkoSOAW";

const Cookies = document.cookie.split(/; */);
```

Rys. 35.20. Kod JavaScript – zastosowanie metody cookie

Gdy kod „projektu ciasteczkowego” będzie ukończony, wpisz poniżej powyższego kodu:

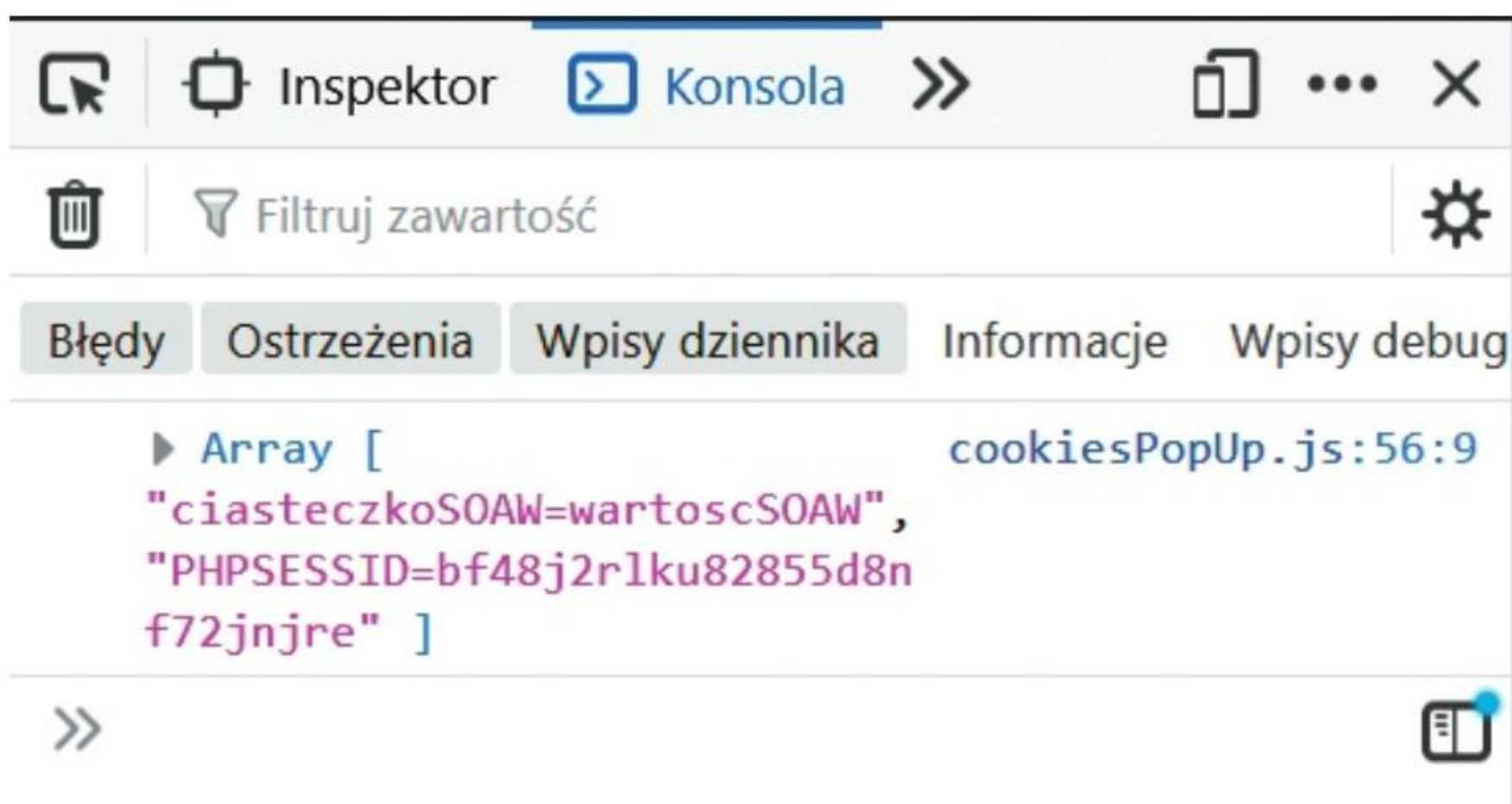
```
console.log(Cookies);
```

Zobacz w przeglądarce, co zostanie napisane w konsoli. W naszym projekcie, ale już po akceptacji ciasteczek, wygląda to tak, jak przedstawiono na rys. 35.21.



CIEKAWOSTKA

Przygotowanie własnego kodu obsługującego „ciasteczka” jest bardzo ciekawym przedsięwzięciem. Warto samemu napisać działający kod. Gdy brakuje jednak czasu na tworzenie jego elementów od podstaw, wtedy warto sięgnąć po framework, który będzie odpowiadał celom i założeniom danego projektu, lub poszukać innego rozwiązania, które ktoś udostępnił na otwartej, bezpłatnej licencji. Tego typu oprogramowanie można znaleźć m.in. na portalu <https://github.com/>.



Rys. 35.21. Kod JavaScript – pokazanie w konsoli, czym jest zmienna Cookies

Następnym elementem w projekcie będzie funkcja o nazwie **sprawdzCookie()**, którą podzielimy na kilka części i omówimy każdą oddzielnie.

Najpierw musimy poszukać, czy w przeglądarce są ciasteczka. Jeżeli są, trzeba znaleźć ich nazwy.

```
function sprawdzCookie(name) {
  if (document.cookie !== "") {
    const Cookies = document.cookie.split(/; */);
    var CookieNames = [];

    for (let i = 0; i < Cookies.length; i++) {
      const CookieName = Cookies[i].split("=")[0];
      CookieNames[i] = CookieName;
      const CookieVal = Cookies[i].split("=")[1];
    }
  }
}
```

Rys. 35.22. Kod JavaScript, funkcja **sprawdzCookie()** – część 1

Następnym krokiem będzie oprogramowanie stworzonych przycisków. Kliknięcie przycisku „Akceptuj” spowoduje usunięcie informacji o ciasteczkach. Do pozostałych przycisków zostaną podane linki prowadzące do odpowiednich stron. Po kliknięciu jednego z przycisków użytkownik zostanie przeniesiony na właściwą stronę.


```
if (divCookies) {  
} else {  
    createcookiesDivEndBaton();  
    /* (name, val, days, path, domain, secure) */  
    document.querySelector('#dodajCiasteczko').addEventListener  
    ('click', function() {  
        wstawCookie('ciasteczkoSOAW', 'wartoscSOAW');  
        document.getElementById('ciasteczkowyNapis').remove();  
    });  
    document.querySelector('#pokazPolitykaPrywatnosci').  
    addEventListener('click', function() {  
        window.location.href = "http://localhost/soaw/  
polityka-prywatnosci.html";  
    });  
    document.querySelector('#pokazCiasteczkaWiecej').  
    addEventListener('click', function() {  
        window.open("https://pl.wikipedia.org/wiki/HTTP\_cookie",  
            '_blank');  
    });  
}
```

Rys. 35.23. Kod JavaScript, funkcja `sprawdzCookie()` – część 2


```
    } else {  
        createcookiesDivEndBaton();  
        document.querySelector('#dodajCiasteczko').addEventListener  
        ('click', function() {  
            wstawCookie('ciasteczkoSOAW', 'wartoscSOAW');  
            document.getElementById('ciasteczkowyNapis').remove();  
        });  
        document.querySelector('#pokazPolitykaPrywatnosci').  
        addEventListener('click', function() {  
            window.location.href = "http://localhost/soaw/  
            polityka-prywatnosci.html";  
        });  
        document.querySelector('#pokazCiasteczkaWiecej').addEventListener  
        ('click', function() {  
            window.open("https://pl.wikipedia.org/wiki/HTTP_cookie",  
            '_blank');  
        });  
    }  
}
```

Rys. 35.24. Kod JavaScript, funkcja `sprawdzCookie()` – część 3


```
} else {  
    createcookiesDivEndBaton();  
    document.querySelector('#dodajCiasteczko').addEventListener  
    ('click', function() {  
        wstawCookie('ciasteczkoSOAW', 'wartoscSOAW');  
        document.getElementById('ciasteczkowyNapis').remove();  
    });  
    document.querySelector('#pokazPolitykaPrywatnosci').addEventListener  
    ('click', function() {  
        window.location.href = "http://localhost/soaw/  
        polityka-prywatnosci.html";  
    });  
    document.querySelector('#pokazCiasteczkaWiecej').addEventListener  
    ('click', function() {  
        window.open("https://pl.wikipedia.org/wiki/HTTP_cookie",  
        '_blank');  
    });  
}
```

Rys. 35.25. Kod JavaScript, funkcja `sprawdzCookie()` – część 4


```
sprawdzCookie(ciasteczkoSzukane);  
// document.cookie = "nazwaCookie=wartoscCookie;  
expires=dataWygasniecia; path=/; secure"  
// Ustawienie własnego ciasteczka  
function wstawCookie(name, val) {  
    if (navigator.cookieEnabled) { //czy ciasteczka są włączone  
        const CookieName = encodeURIComponent(name);  
        const CookieVal = encodeURIComponent(val);  
        let cookieText = CookieName + "=" + CookieVal;  
  
        document.cookie = cookieText;  
    }  
}
```

Rys. 35.26. Kod JavaScript – wywołanie funkcji `sprawdzCookie()` i funkcja `wstawCookie`

Pamiętaj, że w środowisku internetowym masz do dyspozycji przeglądarkę z panelem dewelopera. Użyj `console.log()`, aby wyświetlić poszczególne zmienne. W ten sposób przetestujesz napisany program, ale także własny sposób rozumienia przedstawionych kodów.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to są ciasteczka w informatyce?
2. Wyjaśnij, co to jest RODO.
3. Jaka metoda JavaScript potrafi stworzyć elementy strony internetowej, czyli znaczniki HTML? Podaj przykład.
4. Wymień przynajmniej jedną metodę JavaScript, która dodaje elementy strony internetowej, czyli znaczniki HTML do drzewa DOM.
5. Wymień rodzaje zmiennych JavaScript i opisz, czym się różnią.
6. Napisz na kartce papieru funkcję JavaScript, która znajdzie na stronie internetowej pierwszy dostępny znacznik `div` (lub dowolny inny element HTML) i wyświetli w nim dowolny ciąg tekstowy.
7. Wyjaśnij, do czego służą metody JavaScript:
 - `appendChild()`,
 - `createElement()`,
 - `setAttribute()`,
 - `createTextNode()`,
 - `addEventListener()`.



SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Stwórz plik index.html (stronę HTML), zawierający wewnątrz znacznika **<body> ... </body>** – znacznik **<div> ... </div>**, a w nim znaczniki:

- **<h1>Zabawy z językiem programowania JavaScript</h1>**,
- **<script> .. </script>**.

Wewnątrz znacznika **<script> .. </script>** napisz skrypt w języku programowania JavaScript (wykorzystując metody tego języka poznane w tej części podręcznika lub inne) i:

- A. stwórz nowy element **<div> .. </div>**;
 - B. dodaj do elementu **<div> .. </div>** stworzonego w punkcie A atrybut **id="stworzony"**;
 - C. wstaw do elementu **<div> .. </div>** stworzonego w punkcie A nowy element tekstowy, a następnie dodaj do niego tekst „Tekst stworzony przez język programowania JavaScript”;
 - D. dołącz element **<div> .. </div>** stworzony w punkcie A do wnętrza elementu **<body> ... </body>**.
2. Będziemy kontynuowali pracę na kodem przygotowanym w zadaniu 1. Wykonaj kolejne polecenia, opierając się na kodzie z powyższego zadania:
 - A. za pomocą języka HTML stwórz tuż przed sekcją **<script> .. </script>** przycisk o **id="zmienTekst"**; ma się na nim wyświetlić tekst „Zmień tekst”;
 - B. utwórz wewnątrz sekcji **<script> .. </script>** zdarzenie, które po kliknięciu przycisku **id="zmienTekst"** (utworzony w punkcie A) zmieni na stronie HTML tekst:

„Tekst stworzony przez język programowania JavaScript”

na tekst

„Tekst został zmieniony przez język programowania JavaScript”.

3. Stwórz plik index.html (stronę HTML), zawierający wewnątrz znacznika **<body> ... </body>** znacznik **<div> ... </div>**, a w nim znaczniki:

- A. **<h1>IT CORPORATION: WITAMY</h1>**;
- B. **<script src="script.js"></script>**.

W katalogu głównym projektu utwórz plik o nazwie script.js. W języku programowania JavaScript zapisz w nim kod, który zamieni wyraz ‘WITAMY’ z punktu A na napis:

- a) **'WITAMY O PORANKU'** – gdy użytkownik wejdzie na stronę index.html przed 8.00, ale nie wcześniej niż po 4.00;
- b) **'DZIEŃ DOBRY'** – gdy użytkownik wejdzie na stronę index.html między 8.00 a 12.00;
- c) **'MIŁEGO POPOŁUDNIA'** – gdy użytkownik wejdzie na stronę index.html między 12.00 a 20.00,
- d) **'DOBRANOC'** – gdy użytkownik wejdzie na stronę index.html między 20.00 a 4.00.

36

SOAW: interakcja w aplikacji – przykłady kodu, biblioteka jQuery

ZAGADNIENIA

- Wprowadzenie do biblioteki jQuery
- Przykłady składni jQuery
- Projekt menu rozwijanego – kod HTML
- Projekt menu rozwijanego – kod CSS
- Projekt menu rozwijanego – kod jQuery

36.1. Wprowadzenie do biblioteki jQuery

jQuery to biblioteka oparta na JavaScript. Ułatwia ona i upraszcza pisanie kodu programistycznego. Wymagany kod do osiągnięcia tego samego efektu w jQuery jest zdecydowanie krótszy niż napisany w vanilla JavaScript. Niestety, mniejsza ilość kodu przekłada się na spadek jego wydajności, gdyż jQuery „obciąża” projekt, do którego jest dołączana. Często używa się jej do tworzenia animacji na stronach internetowych, a także do dokonywania dynamicznych zapytań AJAX (np. do relacyjnej bazy danych SQL). W przeciwieństwie do „czystej” JavaScript jQuery nie jest dostępna bezpośrednio w silniku przeglądarek internetowych. Tworząc projekt z tą biblioteką, trzeba ją dołączyć do danej strony, którą ma obsługiwać. Można to zrobić na co najmniej dwa sposoby.

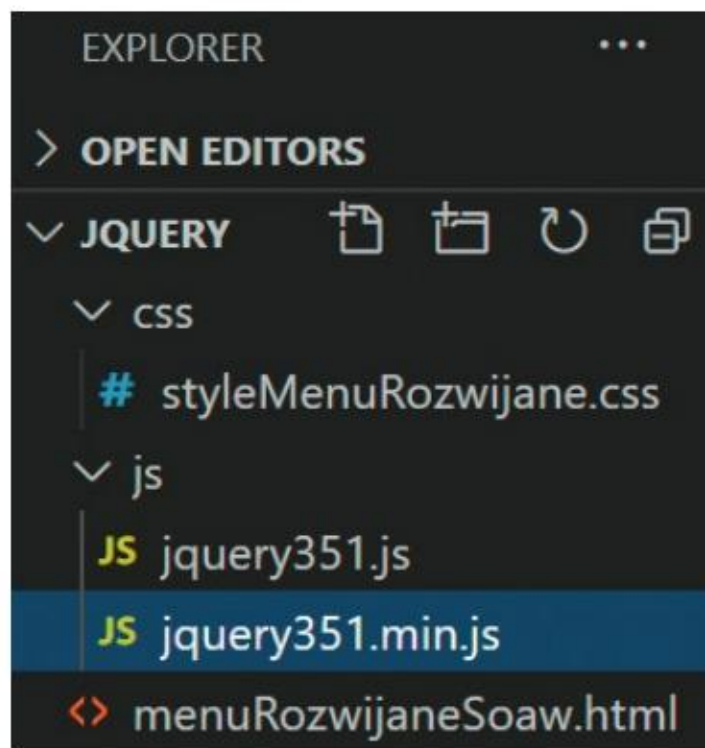
Pobranie jQuery w postaci pojedynczego pliku

Plik z zawartością jQuery można pobrać z oficjalnej strony jQuery (dla bezpieczeństwa najlepiej pobrać właśnie w ten sposób – <https://jquery.com/download/>). Pobrany plik trzeba dołączyć do projektu, tzn. wkleić do odpowiedniego katalogu. Jeżeli pracujemy nad projektem i być może będziemy musieli zmodyfikować jakiś element biblioteki, wówczas trzeba wybrać pełen plik, tutaj – jquery351.js.

Jeżeli nie będzie potrzeby dokonywania zmian w bibliotece (tak jest w zdecydowanej większości przypadków), wówczas najlepiej jest podlinkować plik zminifikowany, czyli jquery351.min.js (rys. 36.1). Ta nazwa oznacza, że będzie to zapis z usunięciem wszystkich spacji i podziału na wiersze. Tego typu plik będzie dużo „lżejszy”. Dla przykładu poniżej podajemy „wagę” obu plików:

- jquery351.js – 115 KB,
- jquery351.min.js – 88 KB.

Różnica jest znaczna. Oznacza to, że strona używająca pliku jquery351.min.js będzie się ładowała i zapewne także działała zdecydowanie szybciej niż strona, do której byłby podlinkowany plik jquery351.js.



Rys. 36.1. VSC – widok struktury katalogów i plików, w tym jquery351.min.js

```
jquery351.min.js > ...
/*! jQuery v3.5.1 | (c) JS Foundation and other contributors
| jquery.org/license */
!function(e,t){"use strict";"object"==typeof module&&
"object"==typeof module.exports?module.exports=e.document?t(e,
!0):function(e){if(!e.document)throw new Error("jQuery
requires a window with a document");return t(e)}:t(e)}
("undefined"!=typeof window?window:this,function(C,e){"use
strict";var t=[],r=Object.getPrototypeOf,s=t.slice,g=t.flat?
function(e){return t.flat.call(e)}:function(e){return t.
concat.apply([],e)},u=t.push,i=t.indexOf,n={},o=n.toString,
v=n.hasOwnProperty,a=v.toString,l=a.call(Object),y={},
m=function(e){return"function"==typeof e&&"number"!=typeof e.
nodeType},x=function(e){return null!=e&&e===e.window},E=C.
document,c={type:!0,src:!0,nonce:!0,noModule:!0};function b(e,
t,n){var r,i,o=(n=n||E).createElement("script");if(o.text=e,t)
for(r in c)(i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.
setAttribute(r,i);n.head.appendChild(o).parentNode.removeChild
(o)}function w(e){return null==e?e+"":"object"==typeof e||
"function"==typeof e?n[o.call(e)]||"object":typeof e}var f="3.
5.1",S=function(e,t){return new S.fn.init(e,t)};function p(e)
{var t=!!e&&"length"in e&&e.length,n=w(e);return!m(e)&&!x(e)&&
("array"===n||0===t||"number"==typeof t&&0<t&&t-1 in e)}S.
fn=S.prototype={jquery:f,constructor:S,length:0,
toArray:function(){return s.call(this)},get:function(e)
{return null==e?s.call(this):e<0?this[e+this.length]:this[e]},
```

Rys. 36.2. Widok początku pliku jquery351.min.js – zminifikowanego, zawierającego bibliotekę jQuery

Kolejnym krokiem będzie podlinkowanie biblioteki do strony naszego projektu. Jest to plik z rozszerzeniem .js, czyli można go podlinkować w jeden z takich samych sposobów, jak każdy inny plik zawierający kod JavaScript. Na rys. 36.3 znajdziesz linkowanie pliku jquery351.min.js na górze kodu HTML – w sekcji **head**.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Serwis Ogłoszeniowy Aplikacja Webowa</title>

  <link rel="stylesheet" href="css/style.css">

  <script src="js/jquery351.min.js"></script>

</head>

<body>
```

Rys. 36.3. Linkowanie pliku jQuery sekcji **<head>** kodu HTML

Podłączenie jQuery z zewnętrznego serwera

Najpopularniejszym sposobem dołączenia jQuery do strony jest podlinkowanie tej biblioteki z zewnętrznego serwera. Istnieje kilka adresów, które udostępniają pliki różnego rodzaju bibliotek, z których korzystają programiści z całego świata. Oto kilka przykładowych i chyba najbardziej popularnych:

- <https://developers.google.com/speed/libraries#jquery>,
- <https://cdnjs.com/libraries?q=jquery>,
- https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview#jQuery_Releases_on_the_CDN_0,
- <https://www.jsdelivr.com/package/npm/jquery>.

W realizowanym projekcie rozwijanego menu skorzystamy z adresu:

```
<script src="https://ajax.googleapis.com/ajax/libs/
jquery/3.5.1/jquery.min.js"></script>
```



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Serwis Ogłoszeniowy Aplikacja Webowa</title>

  <link rel="stylesheet" href="css/style.css">

  <script src="https://ajax.googleapis.com/ajax/libs/
    jquery/3.5.1/jquery.min.js"></script>

</head>

<body>
```

Rys. 36.4. Linkowanie jQuery w **<head>** kodu HTML za pomocą zewnętrznego serwera zawierającego wskazaną bibliotekę

Biblioteka jQuery ma potężne możliwości programistyczne, a był taki czas (kilka lat temu), że zdecydowanie wyprzedzała inne biblioteki w zakresie wykorzystania przy projektach internetowych, głównie ze względu na prostotę w użyciu. Wraz z rozwojem kilku innych, szybszych od niej, technologii opartych na JavaScript jej znaczenie (szczególnie w ostatnich dwóch, trzech latach) zdecydowanie spadło. Na niekorzyść w częstotliwości jej użytkowania wpłynął także rozwój języka JavaScript, zwłaszcza powstanie wersji ES6 i wielu frameworków.

Pełną specyfikację jQuery wraz z najnowszymi jej wersjami znajdziesz pod adresem: <https://jquery.com/>. Można tam także zapoznać się z poradnikami pomagającymi opanować sposób użytkowania i funkcjonalność jQuery. Szczegóły API jQuery można odszukać pod adresem: <https://api.jquery.com/>.

Tworząc kod w JavaScript czy też w jego uproszczonej wersji – jQuery, odwołujemy się do elementów drzewa DOM, dokonując np. zmiany ich właściwości. Obok odwołań do elementów HTML, JavaScript i jQuery doskonale sprawdzają się w zmianach elementów arkusza stylów, np. zamiana koloru czcionki bloku header z koloru określonego w arkuszu CSS na inny wybrany przez nas.

36.2. Przykłady składni jQuery

```
<script>

// ukrywa bieżący element:
$(this).hide();

// ukrywa wszystkie elementy <h3>
$("h3").hide();

// ukrywa wszystkie elementy z class="artykul"
$(".artykul").hide();

// ukrywa element o id = "glowny"
$("#glowny").hide();

</script>
```

Rys. 36.5. Biblioteka jQuery – przykładowe polecenia wpisane w znacznik `<script>`

jQuery (szczegóły projektu: <https://jquery.com/>) to biblioteka programistyczna, która ułatwia pracę z językiem programowania JavaScript. Na rys. 36.5 znajdziesz jej zastosowanie do animacji – ukrywania elementów na stronie. Zauważ, że jedna linijka kodu potrafi ukryć dany element. Gdybyśmy chcieli zaprogramować takie działanie w vanilla JavaScript, to musielibyśmy napisać kilka razy więcej kodu lub skorzystać z pomocy animacji CSS. Z tego powodu wielu programistów częściej korzysta z jQuery niż z czystej JavaScript.

Gotowy dokument zdarzeń

Metody jQuery są umieszczane wewnątrz:

```
$(document).ready(function() {
    // Metody jQuery – tutaj ...
});
```

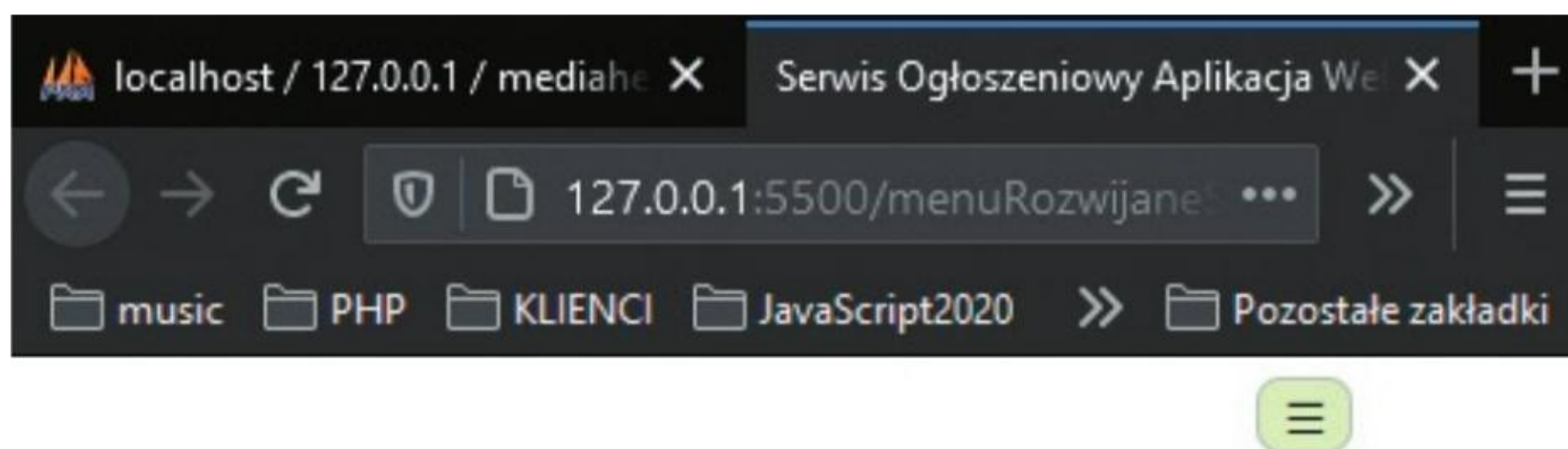
`$(document).ready` oznacza, że gdy strona już się załadowała, to dopiero wtedy jest wykonywana komenda (funkcja, polecenie) jQuery. Rozwiązuje to problem, który sygnalizowaliśmy podczas dołączania kodu JavaScript. Opisywany kod spowoduje, że jQuery zadziała dopiero po załadowaniu wszystkich elementów strony HTML.

Istnieje też krótszy zapis gotowości dokumentu do realizacji poleceń jQuery:

```
$(function() {
    // Metody jQuery – tutaj ...
});
```

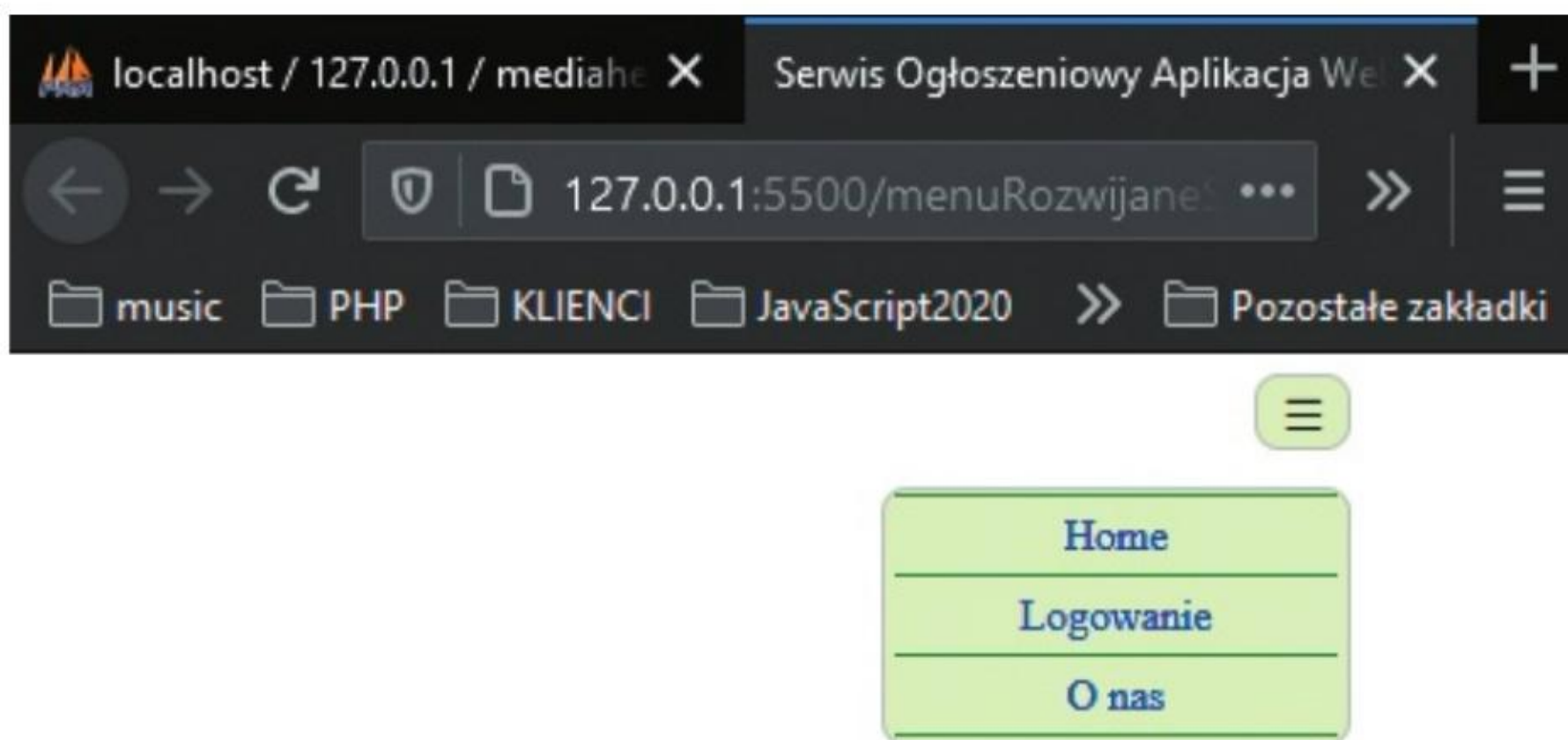
36.3. Projekt menu rozwijanego – kod HTML

Wspólnie z biblioteką jQuery napiszemy kody, które wyświetlą – najpierw małą ikonkę na stronie WWW – tzw. hamburger button.



Rys. 36.6. Budowane menu – hamburger button

Po kliknięciu przycisku rozwinie się menu. Oczywiście pomoże w tym kod jQuery, który wkrótce napiszemy.



Rys. 36.7. Budowane menu – rozwinięte menu

Zauważ, że dysponując mechanizmem powyższego menu rozwijanego, możesz, wklejając jego kody (HTML, CSS i jQuery), dodać takie menu do projektu SOAW. To właśnie będzie twoje zadanie po zakończeniu lektury tej książki.


```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Serwis Ogłoszeniowy Aplikacja Webowa Menu
    Rozwijane</title>

  <link rel="stylesheet" href="css/
    styleMenuRozwijane.css">

  <script src="https://ajax.googleapis.com/ajax/libs/
    jquery/3.5.1/jquery.min.js"></script>

</head>
```

Rys. 36.8. Budowane menu – początek kodu HTML

```
<body>

  <div id="hamburgerButton" class="hamburgerButton">
    ≡ </div>

  <div id="hamburgerMenu" class="hamburgerMenu">
    <ul class="ulHamburgerMenu">
      <li><a href="#">Home</a></li>
      <li><a href="#">Logowanie</a></li>
      <li><a href="#">O nas</a></li>
    </ul>
  </div>
```

Rys. 36.9. Budowane menu – kod HTML (wewnątrz body)

36.4. Projekt menu rozwijanego – kod CSS

Kody CSS są bardzo proste i zostały złożone z popularnych elementów języka CSS. Aby łatwiej je analizować, podzieliliśmy całość na trzy grafiki (rys. 36.10–36.12). Ich zawartość musi się znaleźć w pliku .css.

```
.nav {  
    width: 100%;  
    padding: 10px 0;  
    background-color: #701919;  
    text-align: center;  
    border-top: 1px solid #c34f4f;  
    border-bottom: 1px solid #c34f4f;  
    overflow: hidden;  
    z-index: 100;  
}  
  
.hamburgerButton {  
    padding: 2px;  
    text-align: center;  
    background-color: #e5eecc;  
    border: solid 1px #c3c3c3;  
    border-radius: 0.6em;  
    font-weight: bold;  
    width: 5%;  
    position: absolute;  
    right: 15%;  
}
```

Rys. 36.10. Budowane menu – kod CSS (część 1)

Rys. 36.11. Budowane menu
– kod CSS (część 2)

```
.hamburgerMenu {
    padding: 2px;
    text-align: center;
    background-color: #e5eccc;
    border: solid 1px #c3c3c3;
    max-width: 20%;
    position: absolute;
    right: 15%;
    top: 3%;
    display: none;
    min-width: 170px;
    margin-top: 3%;
    border-radius: 0.6em;
}

.ulHamburgerMenu {
    margin: 0 2px;
    padding: 0;
}
```

Rys. 36.12. Budowane menu
– kod CSS (część 3)

```
.hamburgerMenu li {
    list-style-type: none;
    border-top: 1px solid green;
    text-decoration: none;
    padding: 3%;
}

.hamburgerMenu li:last-child {
    border-bottom: 1px solid green;
}

.hamburgerMenu li:hover {
    background: rgb(218, 233, 218);
}

.hamburgerMenu li a {
    text-decoration: none;
}
```


36.5. Projekt menu rozwijanego – kod jQuery

Został nam ostatni element – kod jQuery. Może być on wpisywany, podobnie jak JavaScript, wewnątrz znaczników `<script>` `</script>` lub też w oddzielnym pliku o rozszerzeniu `.js`. Wybraliśmy pierwsze ze wskazanych powyżej rozwiązań, gdyż jest łatwiejsze i w tym konkretnym przypadku mniej pracochłonne. Kody jQuery umieściliśmy na dole strony HTML, tuż przed zamykającym znacznikiem `</body>` (rys. 36.13).

```
<script>
    $(document).ready(function() {
        $("#hamburgerButton")
            .click(function() {
                $("#hamburgerMenu")
                    .toggle("slow");
            });
    });
</script>
</body>
</html>
```

Rys. 36.13. Budowane menu – kod jQuery

Rozpiszmy użyty kod jQuery:

<code>\$ ("#hamburgerButton")</code>	– chwytamy element HTML, który ma <code>id="#hamburgerButton"</code> ,
<code>.click(function() {</code>	– ustawiamy na elemencie z powyższego wiersza funkcję, która po kliknięciu elementu o <code>id="#hamburgerButton"</code> rozwinie element o <code>id=#hamburgerMenu</code> ,
<code>\$ ("#hamburgerMenu")</code>	
<code>.toggle("slow");</code>	– rozwinięcie nastąpi powoli, o czym mówi nam właściwość „slow”, zamiast niej możemy napisać „fast” albo po prostu podać wartość liczbową i powrócić do wskazanych wartości liczbowych, są one równoważne:


```
'fast' = 200 milisekund
'slow' = 600 milisekund
});
```

SPRAWDŹ SWOJĄ WIEDZĘ

1. Zaimplementuj menu rozwijane do aplikacji SOAW. Zrób to tak, żeby dla widoku:
 - desktop (zwykły ekran komputera) – wyświetlało się menu poziome, które stworzyliśmy we wcześniejszych rozdziałach;
 - mobile (ekran urządzenia przenośnego) – wyświetlało się menu rozwijalne, czyli będzie widać tylko hamburger button, a po jego kliknięciu rozwinię się zawartość menu. Wykorzystaj zapytania o media CSS.
2. Wymień podstawowe różnice między vanilla JavaScript a biblioteką jQuery.
3. Podaj przykłady składni jQuery.
4. Wyjaśnij kod HTML tworzący menu rozwijalne.
5. Wyjaśnij kod CSS tworzący menu rozwijalne.
6. Co oznacza konstrukcja: `$(document).ready(function() { });`? Do czego można ją zastosować?
7. Ile czasu animowałoby się menu, gdyby w kodzie, który tworzyliśmy, zamiast `toggle(,slow')` użyto `toggle(1000)`?
8. Co mógłbyś powiedzieć o zapisie w jQuery `.click(function() { });`, gdybyś znalazł ją w kodzie? Do czego on służy?

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Stwórz plik index.html (stronę HTML), zawierający wewnątrz znacznika `<body> ... </body>` znacznik `<div> ... </div>`, a w nim znaczniki:
 - A. `<h1>IT CORPORATION: WITAMY</h1>`;
 - B. `<script src="script.js"></script>`.

W katalogu głównym projektu utwórz plik o nazwie script.js. Za pomocą biblioteki jQuery zapisz w nim kod, który zamieni wyraz 'WITAMY' z punktu A na napis:

- a) 'WITAMY O PORANKU' – gdy użytkownik wejdzie na stronę index.html przed 8.00, ale nie wcześniej niż po 4.00;
- b) 'DZIEŃ DOBRY' – gdy użytkownik wejdzie na stronę index.html między 8.00 a 12.00;
- c) 'MIŁEGO POPOŁUDNIA' – gdy użytkownik wejdzie na stronę index.html między 12.00 a 20.00;
- d) 'DOBRANOC' – gdy użytkownik wejdzie na stronę index.html między 20.00 a 4.00.

Pamiętaj o właściwym podlinkowaniu biblioteki jQuery.

2. Stwórz plik index.html (stronę HTML), zawierający wewnątrz znacznika `<body> ... </body>` znacznik `<div> ... </div>`, a w nim znaczniki:

- A. `<buton id="start">Rozpocznij animację</button>`;
- B. `<div id="czerwony">czerwony</div>`;
- C. `<div id="zielony">zielony</div>`;
- D. `<div id="niebieski">niebieski</div>`;
- E. `<buton id="end">Zakończ animację</button>`;
- F. `<script src=" animacja.js"></script>`.

Dołącz do pliku index.html plik animacja.css, a następnie stwórz ten ostatni z wymienionych. W pliku animacja.css napisz kody w języku CSS, które spowodują, że elementy `<div> ... </div>` utworzone w punktach B, C i D będą kołami o:

- kolorach obramowania (border) takich samych jak ich nazwy;
- grubości obramowania – 10 px;
- rodzaju obramowania – dowolnym;
- wielkości przekątnej – 100 px;
- położeniu – „przyklejonym” do lewej krawędzi strony i ułożonym jeden pod drugim z marginesem 10 px.

W katalogu głównym projektu utwórz plik o nazwie animacja.js. Za pomocą biblioteki jQuery zapisz w nim kod, który spowoduje, że:

- a) po kliknięciu przez użytkownika lewym przyciskiem myszki w przycisk o id="start" uruchomi się animacja elementów o id="czerwony", id="zielony", id="niebieski";
- b) elementy będą się przesuwaly od lewej krawędzi do prawej, a po osiągnięciu celu zatrzymają się na 2 sekundy, po czym ruszą w powrotną drogę, do lewej krawędzi strony;
- c) każdy kolejny element ruszy w drogę dopiero 2 sekundy po pierwszym;
- d) wędrówka elementów od jednego brzegu strony do drugiego będzie trwała do momentu kliknięcia przez użytkownika lewym przyciskiem myszki w przycisk o id="stop".

Pamiętaj o właściwym podlinkowaniu biblioteki jQuery.

DODATEK

Projekt SOAW w podstawowej wersji, czyli obejmującej kody PHP, składa się z plików:

1. index.php;
2. login.php;
3. templates/headerUser.php (pamiętaj o umieszczeniu headerUser.php w katalogu templates);
4. templates/footerForUser.php (pamiętaj o umieszczeniu footerForUser.php w katalogu templates);
5. config/connectUser.php;
6. config/configEnv.php.

Znajdziesz tu pełne kody plików wskazanych w punktach 1–4, które zostały omówione w podręczniku.

Zgodnie z informacjami zawartymi w podręczniku musisz w pliku connectUser.php (po utworzeniu w katalogu config) umieścić parametry konfiguracyjne połączenia z własną bazą danych. Szczegóły znajdziesz na rys. 34.6.

Zwróć uwagę, że najpierw trzeba założyć w XAMPP bazę o nazwie '**soaw**' i w niej tabelę '**user**'. Szczegóły dotyczące SQL znajdziesz na rys. 34.2–34.4.

Co do pliku configEnv.php, będziesz musiał wykonać poniższe zadania.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. W projekcie są wykorzystywane obrazki:
 - soawLogo200x103.png – logo projektu SOAW,
 - tarczez2021SOAW506x273.png – element uzupełniający design strony głównej projektu.

Aby strony projektowe właściwie się wyświetlały, oba wymienione pliki muszą znaleźć się w katalogu img (umieszczonym w pliku głównym projektu). Twoim zadaniem jest przygotowanie stosownych plików. W nazwach plików graficznych znajdziesz ich wymiary:

- soawLogo200x103.png – 200 px × 103 px;
 - tarczez2021SOAW506x273.png – 506 px × 273 px.
2. W katalogu config utwórz plik configEnv.php. Umieść w nim co najmniej jedną zmienną związaną z lokalizacją projektu na serwerze (jeżeli XAMPP, to adres localhost):

```
$_SESSION['HTTP_SERVER_DOMAIN']
```

i przypisz jej właściwy adres na serwerze.

Podczas rozwiązywania tego zadania pamiętaj, że localhost prowadzi do katalogu htdocs, a projekt istnieje w katalogu soaw.

Projekt SOAW – kody PHP

index.php

```
1  <?php
2
3  session_start();
4
5  if (!isset($_SESSION['HTTP_SERVER_DOMAIN'])) {
6      include_once 'config/configEnv.php';
7  }
8
9  // Wyczyszczenie zmiennych informujących o błędach
10
11 if (isset($_SESSION['e_email'])) {
12     unset($_SESSION['e_email']);
13 }
14
15 if (isset($_SESSION['e_haslo'])) {
16     unset($_SESSION['e_haslo']);
17 }
18
19 ?>
20
21 <!DOCTYPE html>
22 <html lang="pl">
23
24 <head>
25     <meta charset="UTF-8">
26     <meta name="viewport"
27         content="width=device-width, initial-scale=1.
28         0">
29     <title>Serwis Ogłoszeniowy Aplikacja Webowa</
30     title>
31     <link rel="stylesheet" href="css/style.css">
32 </head>
33
34 <body>
35
36     <?php isset($_SESSION['loggedInToApp']) ?
37         include_once 'templates/headerUserLogged.php'
38         : include_once 'templates/headerUser.php'; ?>
39
40     <div id="contentMain" class="contentMain">
41
42         <article>
```



```
38 <h1 class="h1Style"><span>SOAW</span>
39 <br>Serwis Ogłoszeniowy Aplikacja
40 Webowa</h1>
41 <hr>
42
43 <div class="soawContent">
44     <div class="soawContentLeft">
45         
49     </div>
50     <div class="soawContentRight">
51         <h2 class="h2Style">Jak to
52         działa?</h2>
53         <h2 class="h2Style">Założ
54         konto! Bezpłatna
55         rejestracja!</h2>
56         <p>&nbsp; Dodaj za darmo
57         ogłoszenie!</p>
58
59         <?php if ($_SESSION
60             ['HTTP_SERVER_DOMAIN'] .
61             'rejestracja/createUser.php'
62             != 'http://' . $_SERVER
63             ['HTTP_HOST'] . $_SERVER
64             ['PHP_SELF'] & $_SESSION
65             ['HTTP_SERVER_DOMAIN'] .
66             'login.php' != 'http://' .
67             $_SERVER['HTTP_HOST'] .
68             $_SERVER['PHP_SELF']) : ?>
69             <form action="login.php"
70                 method="POST"
71                 class="underHeaderInputs">
72                 Login: <input
73                     type="text"
74                     placeholder="Twój
75                     email" name="email">
```



```
55 <br> <?php if (isset
56 ($_SESSION['e_email'])
57 ) {
58     echo
59     $_SESSION
60     ['e_email']
61     ];
62 }
63 ?>
64 <br> Hasło: <input
type="text"
placeholder="Twoje
hasło"
name="password">
65 <br> <?php if (isset
66 ($_SESSION['e_haslo'])
67 ) {
    echo
    $_SESSION
    ['e_haslo']
    ];
    }
    ?>
    <br> <a href="<?php
echo ($_SESSION
['HTTP_SERVER_DOMAIN']
. 'rejestracja/
createUser.php' ==
'http://' . $_SERVER
['HTTP_HOST'] .
$_SERVER['PHP_SELF']
? '#' : $_SESSION
['HTTP_SERVER_DOMAIN']
. 'rejestracja/
createUser.php') ?
>">Rejestracja</a>
    &nbsp;
    <input
class="inputLogIn"
type="submit"
value="Zaloguj" />
</form>
```



```
68      <?php elseif ($_SESSION
        ['HTTP_SERVER_DOMAIN'] .
        'login.php' == 'http://' .
        $_SERVER['HTTP_HOST'] .
        $_SERVER['PHP_SELF']) : ?>
69          <p
            class="underHeaderInputs">
            LOGIN</p>
70      <?php else : ?>
71          <p
            class="underHeaderInputs">
            <a href="<?php echo
            ($_SESSION
            ['HTTP_SERVER_DOMAIN'] .
            'login.php') ?>">Zaloguj
            Sie</a>
72          </p>
73      <?php endif; ?>
74  </div>
75 </div>
76
77 </article>
78
79 <aside>
80
81     <hr>
82     <p>Lorem ipsum dolor sit amet
        consectetur adipisicing elit. Modi
        eum illum deleniti, voluptate sunt,
        enim repudiandae exercitationem
        dolore corporis dolores possimus
        dolorum veritatis perspiciatis ullam
        magni assumenda eos ad obcaecati.</p>
83 </aside>
84
85 </div>
86
87 <?php include_once 'templates/footerForUser.
    php';
88 ?>
89
90 </body>
```


login.php

```
1  <?php
2  session_start();
3  include_once 'config/connectUser.php';
4  include_once 'config/configEnv.php';
5
6  if (isset($_SESSION['e_zlyemaillogowania'])) {
7      unset($_SESSION['e_zlyemaillogowania']);
8  }
9
10 if (isset($_SESSION['e_email'])) {
11     unset($_SESSION['e_email']);
12 }
13
14 if (isset($_SESSION['e_zlehaslologowania'])) {
15     unset($_SESSION['e_zlehaslologowania']);
16 }
17 if (isset($_SESSION['e_falsePasswordInDatabase']))
18 ) {
19     unset($_SESSION['e_falsePasswordInDatabase']);
20 }
21
22 if (isset($_POST['email']) && strlen($_POST
23 ['email']) > 0) {
24
25     $flagaTest = true;
26
27     $email = trim($_POST['email']);
28
29     $emailAfterSanitize = filter_var($email,
30 FILTER_SANITIZE_EMAIL);
31
32     if ($emailAfterSanitize == $email &&
33 filter_var($emailAfterSanitize,
34 FILTER_VALIDATE_EMAIL)) {
35         $flagaTest = true;
36     } else {
37         $flagaTest = false;
38         $_SESSION['e_emailLogin'] = "Login - musi
39 to być poprawny adres email!";
40     }
41
42     $password = trim($_POST['password']);
```



```
37
38     if ((strlen($password) < 8) || (strlen
39         ($password) > 12)) {
40         $flagaTest = false;
41
42         $_SESSION['e_haslo'] = "Hasło musi
43             posiadać od 8 do 12 znaków!";
44     }
45     if ($flagaTest == true) {
46
47         try {
48             $connection = new PDO($dsn, $db_user,
49                 $db_password, $options);
50             if (!$connection) {
51                 throw new Exception
52                     ($connection->error);
53             } else {
54
55                 $testEmail = false;
56
57                 $czyJestRekord =
58                     $connection->query("SELECT * FROM
59                         user WHERE email = '$email'");
60
61                 if ($czyJestRekord->rowCount() >
62                     0) {
63                     $testEmail = true;
64                     $_SESSION['emailVarSession']
65                         = $emailAfterSanitize;
66                     if (isset($_SESSION
67                         ['e_zlyemaillogowania'])) {
68                         unset($_SESSION
69                             ['e_zlyemaillogowania']);
70                     }
71
72                     if (isset($_SESSION['e_email']
73                         )) {
74                         unset($_SESSION['e_email']
75                             );
76                     }
77                 } else {
```



```
68         $_SESSION
        ['e_zlyemaillogowania'] =
        "Nie ma takiego loginu
        (adresu email) w naszej
        bazie. Podałeś/aś błędny
        email, spróbuj ponownie.";
69         $testEmail = false;
70     }
71 }
72 } catch (Exception $e) {
73     echo '<span style="color:red;">Błąd
        serwera! Przepraszamy za
        niedogodności i prosimy o wizytę w
        innym terminie!</span>';
74     echo '<br />Informacja developerska:
        ' . $e;
75 }
76
77 if ($testEmail == true) {
78     $hasloHash = password_hash($password,
        PASSWORD_DEFAULT);
79
80     $checkPassword = $connection->prepare
        ('SELECT haslo FROM user WHERE email
        = ?');
81     $checkPassword->execute([$email]);
82     $passwordFromDb =
        $checkPassword->fetch();
83
84     if ($passwordFromDb) {
85         $searchedPassword =
        $passwordFromDb[0];
86         if (password_verify($password,
        $searchedPassword) ||
        $searchedPassword == $password) {
87             $_SESSION['loggedInToApp'] =
        true;
88             header('location: loggedUser.
        php');
89         } else {
```



```
90         $_SESSION
           ['e_zlehaslologowania'] =
           "Podałeś/aś błędne hasło,
           spróbuj ponownie.";
91     }
92 } else {
93     $_SESSION
           ['e_falsePasswordInDatabase'] =
           "Błąd! Podałeś/aś błędne hasło,
           spróbuj ponownie.";
94 }
95 }
96 }
97 } else {
98     $_SESSION['e_email'] = 'brak Login - wprowadź
           poprawny login, którym musi być Twój adres
           email zarejestrowany w naszej bazie';
99 }
100 ?>
101
102 <!DOCTYPE html>
103 <html lang="pl">
104
105 <head>
106     <link rel="stylesheet" href="css/style.css">
107 </head>
108
109 <body>
110     <?php include 'templates/headerUser.php'; ?>
111     <div id="contentMain" class="contentMain">
112         <form action="login.php" method="POST"
           class="loginPodHeader">
113             Login: <input type="text"
           placeholder="Your nick" name="email">
114             <br> <?php if (isset($_SESSION
           ['e_email'])) {
115                 echo $_SESSION['e_email'];
116             }
117             ?>
118             <br> <?php if (isset($_SESSION
           ['e_emailLogin'])) {
```



```
119         echo $_SESSION
120             ['e_emailLogin'];
121     } ?>
122     <br> <?php if (isset($_SESSION
123         ['e_zlyemaillogowania'])) {
124         echo $_SESSION
125             ['e_zlyemaillogowania'];
126     }
127     ?>
128     <br> Hasło: <input type="text"
129         placeholder="Twoje hasło"
130         name="password">
131     <br> <?php if (isset($_SESSION
132         ['e_haslo'])) {
133         echo $_SESSION['e_haslo'];
134     }
135     ?>
136     <br> <?php if (isset($_SESSION
137         ['e_haslo2'])) {
138         echo $_SESSION['e_haslo2']
139             ;
140     }
141     ?>
142     <br> <?php if (isset($_SESSION
143         ['e_zlehaslologowania'])) {
144         echo $_SESSION
145             ['e_zlehaslologowania'];
146     }
147     ?>
148     <br> <a href="<?php echo ($_SESSION
149         ['HTTP_SERVER_DOMAIN'] . 'rejestracja/
150         createUser.php' == 'http://' .
151         $_SERVER['HTTP_HOST'] . $_SERVER
152         ['PHP_SELF'] ? "#" : $_SESSION
153         ['HTTP_SERVER_DOMAIN'] . 'rejestracja/
154         createUser.php') ?>">Rejestracja</a>
155     &nbsp;
156     <input class="inputLogIn"
157         type="submit" value="Zaloguj" />
158 </form>
```



```
143     <article>
144     <h1 class="h1Style"><span>SOAW</span>
    <br>Serwis Ogłoszeniowy Aplikacja Webowa</
    h1>
145         <hr>
146
147         <div class="soawContent">
148             <div class="soawContentLeft">
149                 
153             </div>
154             <div class="soawContentRight">
155                 <h2 class="h2Style">Jak to
156                 działa?</h2>
157                 <h2 class="h2Style">Założ
158                 konto! Bezpłatna
159                 rejestracja!</h2>
160             </div>
161         </div>
162     </article>
163
164     <aside>
165         <hr>
166         <p>Text do przygotowania.</p>
167     </aside>
168 </div>
169
170 <?php include 'templates/footerForUser.php';
171 ?>
172
173 <?php include 'templates/scriptsJs.php'; ?>
174 </body>
</html>
```


templates/headerUser.php

```
1 <header>
2
3 <nav class="menuNawigacyjny grid-container">
4
5 
6
7 <ul class="ulNawigacyjny item2">
8 <li
9 class="<?php echo ($_SERVER
['PHP_SELF'] == $_SESSION
['HTTP_SUBHOST'] . 'index.php' ?
'active' : '') ?> ">
10 <a href="<?php echo $_SESSION
['HTTP_SERVER_DOMAIN'] . 'index.
php'; ?>">Home</a></li>
11 <li
12 class="<?php echo ($_SERVER
['PHP_SELF'] == $_SESSION
['HTTP_SUBHOST'] . 'rejestracja/
createUser.php' ? 'active' : '') ?
> ">
13 <a
14 href="<?php echo ($_SESSION
['HTTP_SERVER_DOMAIN'] .
'rejestracja/createUser.php'
== 'http://' . $_SERVER
['HTTP_HOST'] . $_SERVER
['PHP_SELF'] ? "#" : $_SESSION
['HTTP_SERVER_DOMAIN'] .
'rejestracja/createUser.php')
?>">Rejestracja</a>
15 </li>
16 <li
17 class="<?php echo ($_SERVER
['PHP_SELF'] == $_SESSION
['HTTP_SUBHOST'] . 'login.php' ?
'active' : '') ?> ">
18 <a
```



```
19 href="<?php echo ($_SESSION  
   ['HTTP_SERVER_DOMAIN'] .  
   'login.php' == 'http://' .  
   $_SERVER['HTTP_HOST'] .  
   $_SERVER['PHP_SELF'] ? "#" :  
   $_SESSION  
   ['HTTP_SERVER_DOMAIN'] .  
   'login.php') ?>">Logowanie</  
   a></li>  
20 </ul>  
21  
22 </nav>  
23 </header>
```


templates/footerForUser.php

```
1  <footer id="site-footer" class="footerStyle">
2      <div class="footerLeft">
3          <h2 class="font-italic">SOAW</h2>
4          <h2 class="display-5 font-italic">www.
      adresSOAW.pl</h2>
5          <h3>Serwis Ogłoszeniowy Aplikacja Webowa</
      h3>
6      </div>
7      <div class="footerRight">
8          <h3><span style='font-size:20px;'>&
      #x260F; GSM 500 xxx yy</span></h3>
9          <h3><span id="footerSpanFirma"
      style='font-size:20px;'>&#9993;
      biuro@adresSOAW.pl</span></h3>
10         <h3 class="footer-copyright">&copy;
11             2021 <a href="https://adresSOAW.pl/"
              ">powered by SOAW</a>
12         </h3>
13     </div>
14
15 </footer>
16
17 <script src="js/cookiesPopUp.js"></script>
18
```


NETOGRAFIA

<http://brackets.io/>
<http://git-scm.com/>
<https://agilemanifesto.org/>
<https://atom.io/>
<https://code.visualstudio.com/>
<https://creativecommons.org/licenses/by/4.0/>
https://developer.chrome.com/docs/extensions/reference/devtools_panels/
<https://developer.mozilla.org/en-US/docs/Tools>
<https://developer.mozilla.org/pl/>
<https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/>
<https://dom.spec.whatwg.org/>
<https://github.com>
<https://mariadb.com/>
<https://netbeans.org/>
[https://opensource.org/licenses/GPL-2.0\)](https://opensource.org/licenses/GPL-2.0)
<https://pear.php.net/>
<https://pecl.php.net/>
<https://visualstudio.microsoft.com/pl/>
<https://www.apachefriends.org/pl/index.html>
<https://www.eclipse.org/>
<https://www.libreoffice.org/>
<https://www.php.net/>
<https://www.phpmyadmin.net/>
<https://www.uodo.gov.pl/>
<https://www.w3.org/>

Strony dotyczące oprogramowania:

<http://www.wymeditor.org/>
<https://airtable.com/>
<https://angular.io/>
<https://blossom.co/>
<https://bulma.io/>
<https://ckeditor.com/>
<https://cksource.com/>
<https://clarle.github.io/yui3/yui/docs/editor/>
<https://clickup.com/>
<https://getbootstrap.com/>
<https://hive.com/>
<https://inkscape.org/>

<https://jquery.com/>
<https://laravel.com/>
<https://pl.reactjs.org/>
<https://semantic-ui.com/>
<https://svelte.dev/>
<https://symfony.com/>
<https://trac.edgewall.org/>
<https://trello.com/pl/>
<https://vuejs.org/>
<https://www.assembla.com/>
<https://www.atlassian.com/software/jira>
<https://www.bugzilla.org/>
<https://www.dolibarr.org/>
<https://www.gimp.org/>
<https://www.inflectra.com/>
<https://www.mantisbt.org/>
<https://www.pivotaltracker.com/>
<https://www.redmine.org/>
<https://www.targetprocess.com/>
<https://www.tiny.cloud/docs/quick-start/>
<https://www.yiiframework.com/>

Przykładowe moduły do VSC:

<https://github.com/brapifra/vscode-phpserver>
<https://github.com/brapifra/vscode-phpserver>
<https://github.com/ecmel/vscode-html-css>
<https://github.com/felixfbecker/vscode-php-intellisense>
<https://github.com/HookyQR/VSCodeBeautify>
<https://github.com/lonefy/vscode-js-css-html-formatter>
<https://github.com/michaelgmcd/vscode-language-babel>
<https://github.com/mtxr/vscode-sqltools>
<https://github.com/ritwickdey/vscode-live-server>
<https://github.com/xabikos/vscode-javascript>

Biblioteki programistyczne online:

<https://cdnjs.com/libraries?q=jquery>
<https://developers.google.com/speed/libraries#jquery>
https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview#jQuery_Releases_on_the_CDN_0
<https://www.jsdelivr.com/package/npm/jquery>

WYKAZ PODSTAWOWYCH POJĘĆ W JĘZYKACH: POLSKIM, ANGIELSKIM I NIEMIECKIM

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
aktualizacja	update	die Aktualisierung
algorytm	algorithm	der Algorithmus
analiza	analysis	die Analyse
architektura klient–serwer	client-server model	das Klient-Server-Modell
archiwizacja	data archiving	die Datenversicherung / die Archivierung
atrybut	attribute	das Attribut
baza danych	database	die Datenbank
bezpieczeństwo danych	data safety	die Datensicherheit
biznesplan	business plan	der Geschäftsplan/ Businessplan
blok tekstu	block of text	der Textblock
bramka logiczna	logic gate	die Logic-Gatter
czas pracy	work time	die Arbeitszeit
czcionka	font	die Schriftart
dane osobowe	personal data	die Personaldaten
dokument	document	das Dokument / die Urkunde
filtr	filter	der Filter
formatowanie	formatting	die Formatierung
formularz	form	das Formular
funkcja	function	die Funktion
grafika rastrowa	raster graphics	die Bitmapgrafik / die Rastergrafik / die Pixelgrafik
grafika wektorowa	vector graphics	die Vektorgrafik
grupa robocza	workgroup	die Arbeitsgruppe
hasło	password	das Passwort
hierarchia	hierarchy	die Hierarchie
hipertekstowy język znaczników	hypertext markup language	die Hypertext-Auszeichnungssprache / die Web-Programmiersprache
host	host	der Host
ikonka	icon	die Ikone
implementacja	implementation	die Implementierung

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
indeksowanie	indexing	die Indexierung
instrukcja obsługi	manual	das Handbuch
interfejs	interface	das Interface
iteracja	iteration	die Iteration / die Wiederholung
język skryptowy	scripting language	die Scriptsprache
język SQL	SQL language	die Strukturierte Abfragesprache
kaskadowy arkusz stylów	cascading style sheet	der Kaskadenstilentwurf
klient	client	der Kunde / der Benutzer
kolorystyka	colour scheme	die Farbzusammenstellung
konkatenacja	concatenation	die Verkettung
konkurencja	competition	die Konkurrenz
konto użytkownika	user account	das Benutzerkonto
kosztorys	estimated budget / cost estimation	die Berechnung / der Kosten- voranschlag
kwalifikacje zawodowe	professional qualifications	die Berufsqualifikationen
licencja	licence	die Lizenz
logowanie	login	die Anmeldung
marketing	marketing	das Marketing
menu nawigacyjne	navigation menu	das Navigationsmenü
metoda	method	die Methode
miejsce pracy	work place	der Arbeitsplatz
motyw	theme	das Motiv
obiekt	object	das Objekt
obliczenie	computation	die Berechnung
obraz cyfrowy	digital image	das digitale Bild
ochrona	protection	der Schutz
ograniczenie	limitation	die Begrenzung
organizacja pracy	organisation of work	die Arbeitsplanung
pętla	loop	die Schleife
planista	planner	der Planer
poczta elektroniczna	e-mail	elektronische Post
powiadomienie	notification	die Mitteilung
praca zdalna	remote work	die Heimarbeit

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
pracownik	worker / employee / laborer	der Arbeiter
priorytet	priority	die Priorität
proces	process	der Prozess
profil użytkownika	user profile	das Benutzerprofil
programista	coder / developer	der Programmierer
projekt	project	das Projekt
prośba	request	die Anfrage
protokoły	protocols	die Protokolle
przeglądarka	browser	der Browser
przepis prawny	regulation	die Rechtsverordnung
przycisk	button	der Knopf / die Taste
przyjazny dla użytkownika	user-friendly	anwenderfreundlich / benutz- erfreundlich
relacyjna baza danych	relational database	die relationale Datenbank
roboczogodzina	man-hour	die Betriebsstunde
rozdzielczość	resolution	die Auflösung
sekwencja	sequence	die Sequenz
skalowanie	calibration	das Skalieren / die Skalierung
stanowisko pracy	work station	der Arbeitsplatz
strukturalny język zapytań	the structured language of queries	die Strukturierte Abfragesprache
system bazy danych	database system	das Datenbanksystem
system zarządzania bazą danych	database management system	das Datenbankverwaltungssystem
tabela	table	die Tabelle
testowanie	testing	der Softwaretest
typ danych	data type	der Datentyp
układ graficzny	typography	die Typografie
walidacja	validation	die Validierung
wiersz tabeli	table row	die Tabellenzeile
witryna internetowa	website	die Website
właściwości pola	field properties	die Feldeigenschaften
właściwości tabeli	table properties	die Tabelleneigenschaften
wprowadzenie na rynek	launch	der Launch / der Markteintritt
współpraca	cooperation	die Zusammenarbeit

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
współpracownik	colleague	der Mitarbeiter
wykaz	list	die Liste
wyszukać i usuwać usterki	to debug	die Fehler beseitigen (ein Programm debuggen)
zapis (rekord)	record	das [Daten]satz
zapytanie (kwerenda)	query	die Abfrage
zasada	principle	das Prinzip
zawód	profession	der Beruf
zdarzenie	happening	das Ereignis
zmienna	variable	die Variable
znak interpunkcyjny	punctuation mark	das Satzzeichen
zorientowany obiektowo	object-oriented	objektorientiert
żargon	jargon	der [Fach]jargon

Źródła ilustracji

Okładka: stół z projektami (Chaosamran_Studio/Shutterstock.com)

Tekst główny: s. 14 (nazwy technologii internetowych) Sashkin/Shutterstock.com; s. 18 (tło skryptowe) whiteMocca/Shutterstock.com; s. 23 (monitory) fad82/Shutterstock.com, (serwer) Aayam 4D/Shutterstock.com, (urządzenia mobilne) fad82/Shutterstock.com; s. 25 (tło skryptowe) whiteMocca/Shutterstock.com; s. 26 (tło skryptowe) whiteMocca/Shutterstock.com, (podwórko) Francesca Pianzola/Shutterstock.com; s. 53 (dzbanek i kubek z kawą) JohannMG/Shutterstock.com; s. 55 (tło skryptowe) whiteMocca/Shutterstock.com; s. 57 (liczby) tostphoto/Shutterstock.com; s. 59 (tło skryptowe) whiteMocca/Shutterstock.com; s. 66 (tło skryptowe) whiteMocca/Shutterstock.com; s. 80 (tło skryptowe) whiteMocca/Shutterstock.com; s. 82 (tło skryptowe) whiteMocca/Shutterstock.com; s. 121 (uczeń) Elnur/Shutterstock.com, (student) ESB Professional/Shutterstock.com, (pracownicy IT) G-Stock Studio/Shutterstock.com; s. 144 (biuro) Rawpixel.com/Shutterstock.com, (księgowość) Standret/Shutterstock.com, (produkcja) Gorodenkoff/Shutterstock.com, (telefon) Vasin Lee/Shutterstock.com, (mężczyzna) Rido/Shutterstock.com; s. 168 (znaczenie klientów) Rawpixel.com/Shutterstock.com; s. 169 (zależność sprzedaży od zadowolenia klienta) Rawpixel.com/Shutterstock.com; s. 171 (laptop) Goran Bogicevic/Shutterstock.com, (aplikacja) PureSolution/Shutterstock.com, (mężczyzna) Rido/Shutterstock.com; s. 178 (piłka) KP Photograph/Shutterstock.com; s. 181 (tło skryptowe) whiteMocca/Shutterstock.com; s. 183 (internauta) ASDF_MEDIA/Shutterstock.com, (laptop) Goran Bogicevic/Shutterstock.com, (aplikacja) PureSolution/Shutterstock.com, (mężczyzna) Rido/Shutterstock.com; s. 212 (tło skryptowe) whiteMocca/Shutterstock.com; s. 214 (tło skryptowe) whiteMocca/Shutterstock.com; s. 216 (tło skryptowe) whiteMocca/Shutterstock.com; s. 234 (zakłócenia w komunikacji interpersonalnej) pathdoc/Shutterstock.com; s. 258 (bezpieczeństwo aplikacji) sitthiphong/Shutterstock.com; s. 321 (cookies) TarikVision/Shutterstock.com

Wydawnictwa Szkolne i Pedagogiczne oświadczają, że podjęły starania mające na celu dotarcie do właścicieli i dysponentów praw autorskich wszystkich zamieszczonych utworów. Wydawnictwa Szkolne i Pedagogiczne, przytaczając w celach dydaktycznych utwory lub fragmenty, postępują zgodnie z art. 27¹ ustawy o prawie autorskim. Jednocześnie Wydawnictwa Szkolne i Pedagogiczne oświadczają, że są jedynym podmiotem właściwym do kontaktu autorów tych utworów lub innych podmiotów uprawnionych w wypadkach, w których twórcy przysługuje prawo do wynagrodzenia.