Politechnika Świętokrzyska
Wydział Zarządzania i Modelowania Komputerowego

Kierunek studiów:
Inżynieria danych

Damian Krzesimowski

Materiały dydaktyczne do przedmiotu

PROJEKTOWANIE APLIKACJI INTERNETOWYCH

opracowane w ramach realizacji Projektu
„Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści

1. Wprowadzenie do projektowania aplikacji internetowych

1.1. Słowem wstępu

Projektowanie aplikacji internetowych stanowi fundament współczesnych rozwiązań cyfrowych, łącząc warstwę wizualną z głębokimi procesami przetwarzania danych. Każda witryna czy aplikacja webowa musi sprostać wymaganiom użyteczności, wydajności i bezpieczeństwa, a także zapewnić sprawną komunikację między przeglądarką a serwerem. W niniejszym opracowaniu przedstawione zostaną wybrane koncepcje i technologie wykorzystywane na etapie projektowania front-endu oraz back-endu, ze szczególnym uwzględnieniem wymiany danych przez formularze. Zwrócimy uwagę głownie na mechanizmy weryfikacji danych, wspomniane zostaną także metody ich transmisji i dalszego przetwarzania zarówno po stronie klienta, jak i serwera.

1.2. Architektura klient-serwer

Model klient–serwer jest podstawą działania Internetu. Przeglądarka użytkownika odpowiada za prezentację zawartości: interpretację kodu HTML, stylowanie przez CSS oraz wykonanie logiki JavaScript. Serwer przejmuje żądania HTTP, wykonuje odpowiednie operacje na zasobach, bazie danych lub w logice biznesowej, a następnie zwraca rezultaty w postaci statycznych lub dynamicznych zasobów. Każda interakcja rozpoczyna się kliknięciem obiektu interaktywnego lub wypełnieniem formularza, co prowadzi do wygenerowania zapytania HTTP. Po stronie serwera następuje parsowanie parametrów, walidacja i przetwarzanie, a w efekcie wysłanie odpowiedzi. Przeglądarka odbiera dane i wprowadza zmiany w widoku strony. Zrozumienie przepływu danych w tym modelu jest nieodzowne. Standardowy cykl obejmuje nawiązanie połączenia TCP, wysłanie zapytania GET lub POST, przetworzenie żądania, wygenerowanie odpowiedzi, przesłanie nagłówków i treści oraz zamknięcie sesji lub jej utrzymanie w przypadku mechanizmu keep-alive. Wydajność całego procesu zależy od optymalizacji zarówno front-endu (minifikacja zasobów, cache'owanie), jak i back-endu (optymalizacja zapytań do bazy, równoważenie obciążenia).

1.3. Technologie front-end

Front-end to miejsce, w którym kształtuje się doświadczenie użytkownika. HTML5 dostarcza semantycznej struktury dokumentu, umożliwiając zdefiniowanie nagłówków, sekcji, formularzy i wielu innych elementów. Znaczniki semantyczne wspierają również dostępność dla osób korzystających z czytników ekranu. CSS3

odpowiada za warstwę wizualną, np. dzięki modułom Flexbox i Grid można projektować responsywne układy, które automatycznie dostosowują się do szerokości ekranu i rozdzielczości urządzenia. Zaawansowane właściwości animacji i transformacji pozwalają na subtelne dopełnienie interfejsu.
JavaScript to niezastąpione narzędzie do tworzenia interaktywności. Początkowo używany do prostych modyfikacji drzewa DOM, dziś stanowi podstawę skomplikowanych aplikacji jednopliktowych (SPA). Frameworki takie jak React, Vue czy Angular wprowadzają architektury oparte na komponentach, ułatwiając zarządzanie stanem i ponowne wykorzystanie kodu. Współpraca z ES6+ umożliwia korzystanie z modułów, asynchronicznych funkcji, destrukturyzacji i wielu innych nowoczesnych konstrukcji językowych. Bundlery i transpilatory, takie jak Webpack czy Babel, sprawiają, że kod front-endowy może być napisany w najnowszych standardach, a następnie dostarczony do przeglądarki w formie kompatybilnej z wieloma środowiskami.

1.4. Technologie back-end

Back-end odpowiada za logikę biznesową, operacje na danych i bezpieczeństwo. Istnieje kilka popularnych środowisk umożliwiających wykorzystanie JavaScript po stronie serwera, oferując asynchroniczne API i duże wsparcie społeczności programistów. Python, zaimplementowany w frameworkach Django lub Flask, ceniony jest za czytelność kodu i szybkie prototypowanie. Rozwiązania oparte na Javie, jak Spring Framework, czy platformie .NET, dostarczają zaawansowane narzędzia do budowy skalowalnych aplikacji korporacyjnych. Szeroko wykorzystywany jest także język PHP, będący podstawą dużych serwisów wymiany treści pomiędzy użytkownikiem, a bazami danych.
Bazy danych stanowią serce każdej aplikacji. W relacyjnym modelu MySQL i PostgreSQL zapewniają transakcyjność i silne więzy integralności. NoSQL-owe rozwiązania, takie jak MongoDB, umożliwiają elastyczny model dokumentowy i łatwe skalowanie w poziomie. Warstwa dostępu do danych często realizowana jest z użyciem bibliotek ORM, które mapują obiekty domenowe na tabele bazodanowe. Kiedy dane są gotowe do udostępnienia klientowi, back-end eksponuje je przez warstwę API. Architektura RESTful bazuje na dobrze znanych metodach HTTP, natomiast GraphQL pozwala precyzyjnie określić kształt i zakres zwracanych danych, co może zmniejszyć liczbę zapytań i ilość przesyłanych informacji. Mechanizmy uwierzytelniania i autoryzacji zapewniają, że tylko uprawnione podmioty mają dostęp do określonych zasobów. Sesje oparte na ciasteczkach mogą być uzupełnione tokenami JWT, co upraszcza skalowanie oraz współpracę z mikroserwisami. Standard OAuth2 umożliwia delegowanie procesu logowania do zewnętrznych dostawców, co w wielu sytuacjach podnosi poziom zaufania i wygody użytkowników.

1.5. Wymiana danych za pomocą formularzy

Formularze HTML stanowią podstawowy mechanizm pobierania danych od użytkownika stron internetowych. W każdym przypadku pojawienia się pola wyszukiwania, pól logowania lub różnego rodzaju pól wyboru w oknie rejestracji, formularze HTML grają główną rolę w interakcji. Pozwalają użytkownikowi przekazać dane, które mogą być przetwarzane, gromadzone lub przesłane na serwer. Formularze to niejako bramy dostępowe pomiędzy użytkownikiem i serwerem, wprowadzające element dynamiki i interaktywności. Są niezbędne na potrzeby zadań związanych z autentyfikacją, przesyłaniem danych, gromadzeniem opinii itp.
Budowa formularzy powinna być przemyślana od samego początku, zaczynając od określenia celu: rejestracji, logowania, składania zamówienia czy zgłaszania zapytania. Przejrzystość formularza leży u podstaw wysokiego wskaźnika konwersji i satysfakcji użytkownika. Etykiety muszą jasno opisywać dane, które należy wprowadzić, a placeholdery pełnią rolę uzupełniającą — nigdy nie zastępują stałych etykiet, aby zachować dostępność. Grupy pokrewnych pól, na przykład, dane kontaktowe lub informacje o płatności, powinny być wizualnie odseparowane, co wspomaga naturalny przebieg wypełniania.
Walidacja danych zaczyna się już w przeglądarce. Przeglądarki oferują funkcje weryfikacji, np. sprawdzenie formatu adresu email czy ograniczenie długości tekstu. Atrybuty HTML5 pozwalają na natychmiastowe blokowanie wysłania formularza, gdy pola nie spełniają założonych warunków. Gdy wymagania są bardziej złożone, JavaScript przejmuje inicjatywę, generując komunikaty obok poszczególnych pól. Weryfikacja asynchroniczna pozwala na sprawdzenie unikalności loginu czy poprawności podatkowego numeru identyfikacyjnego z zewnętrznymi serwisami w czasie rzeczywistym.
Walidacja po stronie serwera stanowi ostateczną barierę przeciwko wadliwym lub złośliwym danym. Serwer parsuje ciało żądania, a następnie wykonuje pełne sprawdzenie wszystkich reguł biznesowych i zabezpieczenia przed atakami typu SQL-injection oraz XSS. Dzięki spójnym zestawom reguł walidacyjnych zaimplementowanym w ramach używanego frameworka można utrzymać zasady w jednym miejscu, co ułatwia rozwój i utrzymanie aplikacji.
Metody przesyłania danych z formularza opierają się głównie na metodzie POST dla operacji modyfikujących stan systemu, a GET służy do pobierania zasobów bez efektu ubocznego. Dane formularza mogą być kodowane w formacie application/x-www-form-urlencoded, odpowiadającym standardowemu sposobowi przesyłania par klucz-wartość, lub w multipart/form-data, gdy wymagane jest dołączenie plików. W przypadku nowoczesnych aplikacji SPA często preferuje się wysyłanie danych jako JSON, co umożliwia precyzyjną kontrolę nad strukturą i zawartością przesyłanych informacji.

Agregacja danych odbywa się automatycznie przez przeglądarkę lub może zostać przejęta przez skrypty JavaScript. Obiekt FormData daje elastyczność w tworzeniu zestawu pól, włączając w to pliki, checkboxy czy listy wielokrotnego wyboru. Alternatywnie programista może ręcznie budować obiekt JSON, wywołując fetch z odpowiednimi nagłówkami i ciałem zapytania. Ważne jest, aby zarówno klient, jak i serwer uzgodnili format danych, co zapobiega nieporozumieniom i ułatwia proces debugowania.

1.6. Przetwarzanie danych po stronie klienta

Przetwarzanie danych tuż po wprowadzeniu przez użytkownika może znacznie zwiększyć komfort korzystania z serwisu. Lekka analiza wejścia w czasie rzeczywistym, kontrola zakresu wartości czy wyświetlanie sugestii w postaci autouzupełniania ułatwiają użytkownikowi szybsze uzupełnianie formularza. Mechanizmy Web Workers pozwalają na delegowanie bardziej skomplikowanych obliczeń do osobnych wątków, dzięki czemu interfejs pozostaje responsywny nawet przy rozbudowanych walidacjach lub transformacjach danych.
Wiele aplikacji korzysta także z lokalnego przechowywania w pamięci przeglądarki (localStorage, IndexedDB), aby zapobiegać utracie danych w przypadku odświeżenia strony lub przypadkowego zamknięcia karty. Dynamiczne podpowiedzi oparte na historii wpisywanych danych lub import z zewnętrznych baz (np. list krajów lub miast) skracają czas potrzebny na wypełnienie formularza.

1.7. Przetwarzanie danych po stronie serwera

Po stronie serwera cały zestaw danych trafia do komponentu odpowiedzialnego za mapowanie na obiekty domenowe i dalsze operacje. W Node.js często korzysta się z middleware'u body-parser, w Django z wbudowanego request body, a w innych frameworkach z dedykowanych modułów. Kolejno uruchamiane są reguły walidacji i sanitizacji, a dopiero po ich pomyślnym przejściu następuje zapis do bazy danych. W przypadku błędów serwer generuje odpowiedni kod statusu HTTP oraz treść JSON zawierającą szczegółowe informacje o nieprawidłowych polach, co pozwala klientowi precyzyjnie poinformować użytkownika o konieczności korekty.
W architekturze mikroserwisów formularze mogą publikować zdarzenia na kolejkę (Kafka, RabbitMQ), a kolejne serwisy przetwarzają je asynchronicznie. Takie podejście zwiększa odporność systemu i pozwala na płynną rozbudowę logiki biznesowej bez blokowania głównego wątku obsługi żądań użytkowników.

2. Statyczne formularze HTML5

Fundamentalna struktura formularza HTML zawarta jest między znacznikami <form>. Wewnątrz tego tagu można umieścić różnorodne kontrolki jak pola tekstowe, pola wyboru jednokrotnego i wielokrotnego oraz przyciski do przesłania formularza. Podstawowa struktura formularza wygląda następująco:

```
<form>
     <!--elementy formularza-->
</form>
```

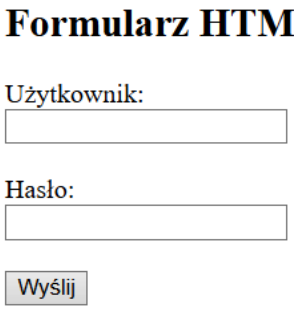
Zbudujmy stronę internetową w postaci formularza logowania złożonego z dwóch pól tekstowych oraz przycisku przekazania danych:

```
<html>
<head>
     <meta charset="utf-8">
     <title>Strona z formularzem HTML</title>
</head>
<body>
     <h2>Formularz HTML</h2>
     <form>
          <label for="username">Użytkownik:</label><br>
          <input type="text" id="username" name="username"><br><br>
          <label for="password">Hasło:</label><br>
          <input type="password" id="password"
name="password"><br><br>
          <input type="submit" value="Wyślij">
     </form>
</body>
</html>
```

Po zapisaniu niniejszego kodu HTML oraz jego uruchomieniu w przeglądarce pojawi się formularz jak na rys. 1.

[Tekst alternatywny. Formularz HTML do logowania. Nagłówek: „Formularz HTML”. Dwie etykiety i pola tekstowe: „Użytkownik:” – pole tekstowe dla nazwy użytkownika;

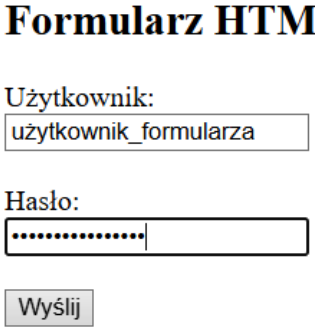
„Hasło:” – pole tekstowe dla hasła. Przycisk „Wyślij” na dole formularza.]



Rys. 1. Podstawowy formularz HTML.

Kod programu zawiera trzy różne elementy graficzne. Pierwszym jest pole wprowadzania tekstu jawnego: <input type="text" id="username" name="username">. Typ danych wprowadzanych to tekst, podano także identyfikator pola oraz jago nazwę. Dwa ostatnie parametry są niezbędne na potrzeby przetwarzania danych użytkownika, choć sam formularz bez nich będzie wyglądał identycznie jak na rys. 1. Drugim polem wprowadzania jest pole hasła, ten typ pozwala na ukrywanie znaków wprowadzanych przez użytkownika. Trzecim obiektem jest przycisk przesyłania danych „submit”. Różni się on od zwykłego przycisku „button” tym, że po jego aktywacji (naciśnięciu) próbuje domyślnie przesłać ciąg znaków do serwera. W przypadku zwykłego przycisku taka akcja wymaga uruchomienia oddzielnego skryptu. Po wprowadzeniu danych oraz aktywowaniu przycisku przesyłania danych w polu adresu pojawią się wprowadzone dane (formularz.html?username=użytkownik_formularza&password=to_jest_hasło_15) , jak na rys. 2.

[Tekst alternatywny. Formularz HTML. Nagłówek: Formularz HTML. Pierwsza etykieta i pole tekstowe: Użytkownik: (tekst „użytkownik_formularza”). Druga etykieta i pole hasła: Hasło: (ukryte znaki). Przycisk: Wyślij.]



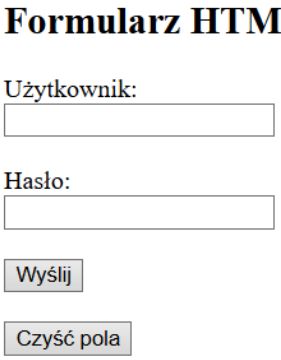
Rys. 2. Dane wprowadzone przez użytkownika.

Rozbudujmy formularz o kolejny przycisk czyszczący dane z pól formularza. Poniżej przycisku do wysyłania danych wprowadźmy następującą linię:

```
<br><br><input type="reset" value="Czyść pola">
```

Formularz będzie wyglądał, jak na rys. 3.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Dwie etykiety pola: Użytkownik: pole tekstowe; Hasło: pole tekstowe. Dwa przyciski na dole formularza: Wyślij; Wyczyść pola.]



Rys. 3. Formularz z przyciskiem czyszczącym dane.

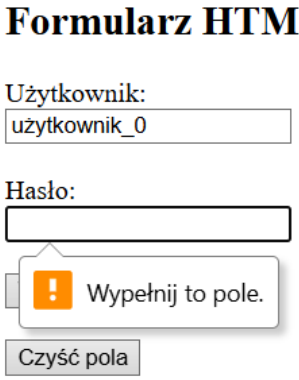
Przycisk czyszczenia danych „reset” po aktywacji wypełni i przestawi wszystkie elementy formularza na wartości domyślne. W naszym przypadku wartości domyślne pól tekstowych nie zostały zdefiniowane, zatem pola te będą wyczyszczone. Nadanie wartości domyślnej dla pola tekstowego jest możliwa poprzez wykorzystanie atrybutu „value”. Przykładowo, jeśli chcemy, aby wartością domyślną dla pola nazwy użytkownika na niniejszej stronie był ciąg znaków „użytkownik_0”, wystarczy uzupełnić wiersz definiujący odpowiednie pole tekstowe: <input type="text" id="username" name="username" value="użytkownik_0">

. Teraz po aktywacji przycisku czyszczenia pole tekstowe z nazwą użytkownika zostanie automatycznie wypełnione podanym ciągiem. Zaznaczyć należy, że ten sam ciąg znaków pojawi się w chwili pierwszego uruchomienia strony z formularzem oraz jej odświeżenia.
Obecny kod pozwala wywołać akcję przekazania danych na serwer nawet, gdy użytkownik niczego nie wprowadzi. Na potrzeby wymuszenia wprowadzenia danych przez użytkownika wykorzystuje się atrybut „required”. Przykładowo, aby użytkownik zdefiniował hasło należy zmodyfikować wiersz temu służący do postaci:

```
<input type="password" id="password" name="password"
required><br><br>
```

Teraz, gdy użytkownik spróbuje wysłać formularz bez hasła, otrzyma komunikat ostrzegawczy jak na rys. 4.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Dwie etykiety i pola: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: puste pole tekstowe.]



Rys. 4. Komunikat informacyjny o negatywnej walidacji pola.

Jest to komunikat informacyjny, który należy traktować jako pierwszy element walidacji danych. Walidacja polega na sprawdzeniu poprawności danych. W naszym przypadku sprawdzana jest sama obecność danych, bez kontroli poprawności ciągów wprowadzanych przez użytkownika. Zatem na chwilę obecną użytkownik może wprowadzić hasło np. jednoliterowe.
Rozszerzmy kod programu o kolejne pole wprowadzania: email. Pole to posiada wbudowaną walidację obecności znaku „@” oraz ciągu znaków przed nim i po nim. Pełny kod wygląda następująco:

```
<html>
<head>
     <meta charset="utf-8">
     <title>Strona z formularzem HTML</title>
</head>
<body>
     <h2>Formularz HTML</h2>
     <form>
          <label for="username">Użytkownik:</label><br>
          <input type="text" id="username" name="username"
value="użytkownik_0" required><br><br>
          <label for="password">Hasło:</label><br>
          <input type="password" id="password" name="password"
required><br><br>
          <label for="email">E-mail:</label><br>
```

```
            <input type="email" id="email" name="email"
required><br><br>
            <input type="submit" value="Wyślij">
            <br><br><input type="reset" value="Czyść pola">
      </form>
</body>
</html>
```

Natomiast formularz wraz z komunikatem walidacyjnym poprawności adresu e-mail przedstawiono na rys. 5.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Trzy etykiety i pola: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: pole maskowane (ukryte znaki); E-mail: pole tekstowe z wartością „we”. Poniżej pola E-mail komunikat błędu wraz z ikonką ostrzeżenia: „Uwzględnij znak „@” w adresie e-mail. W adresie „we” brakuje znaku „@”.” Na dole formularza przycisk „Czyść pola”.]

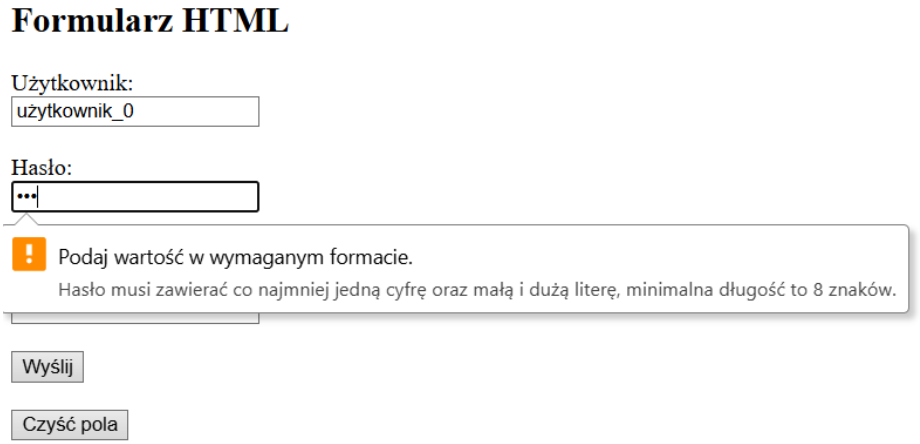
Rys. 5. Rozbudowany formularz z komunikatem walidacyjnym dla pola e-mail.

Wróćmy teraz po pola z hasłem i nałóżmy ograniczenia na jego wprowadzanie. Przede wszystkim odrzućmy hasła krótsze niż 8 znaków. Następnie wymuśmy użycie co najmniej jednej cyfry, jednej litery małej i jednej litery dużej. Do tego celu służy atrybut „pattern”, który w tym przypadku będzie wyglądał następująco:

```
<input type="password" id="password" name="password"
pattern="(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Hasło musi
zawierać co najmniej jedną cyfrę oraz małą i dużą literę, minimalna
długość to 8 znaków." required><br><br>
```

Wartość atrybutu „title” pojawia się jako opis w polu walidacyjnym, co przedstawiono na rys. 6.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Dwie etykiety i pola: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: pole maskowane (ukryte znaki). Poniżej pola hasła komunikat ostrzegawczy wraz z ikoną: „Podaj wartość w wymaganym formacie. Hasło musi zawierać co najmniej jedną cyfrę oraz małą i dużą literę, minimalna długość to 8 znaków.” Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



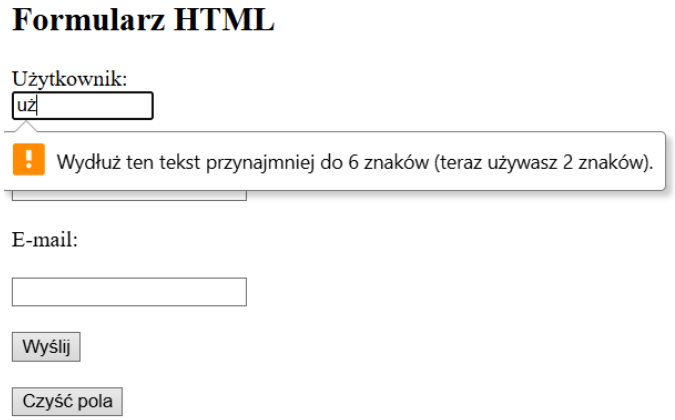
Rys. 6. Walidacja rozszerzona hasła wprowadzanego przez użytkownika.

Zaznaczyć należy, że treść główna wszystkich przedstawionych do tej pory wiadomości walidacyjnych jest wyświetlana w języku systemu lub języku zdefiniowanym w przeglądarce internetowej. Treści prezentowane w atrybutach „title” wyświetlane są jako łańcuchy tekstu pobierane bezpośrednio z kodu HTML. Wprowadźmy dodatkowe ograniczenia na nazwę użytkownika. Niech liczba znaków, które może wprowadzić użytkownik, mieści się w przedziale zamkniętym od 6 do 12 znaków. Dodatkowo niech pierwsza litera będzie duża. Ograniczmy także szerokość pola tekstowego wyświetlanego w formularzu na stałe do szerokości 12 znaków:

```
<input type="text" id="username" name="username"
value="użytkownik_0" minlength="6" maxlength="12" size="12"
required><br><br>
```

Po wprowadzeniu zbyt krótkiej nazwy użytkownika pojawi się komunikat jak na rys. 7.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Dwie etykiety i pola: Użytkownik: pole tekstowe wypełnione wartością „uż”; E-mail: puste pole tekstowe. Poniżej pola Użytkownik komunikat błędu wraz z ikonką ostrzeżenia: „Wydłuż ten tekst przynajmniej do 6 znaków (teraz używasz 2 znaków)”. Na dole formularza dwa przyciski: Wyślij, Czyść pola.]



Rys. 7. Komunikat walidacyjny dla pola nazwy użytkownika.

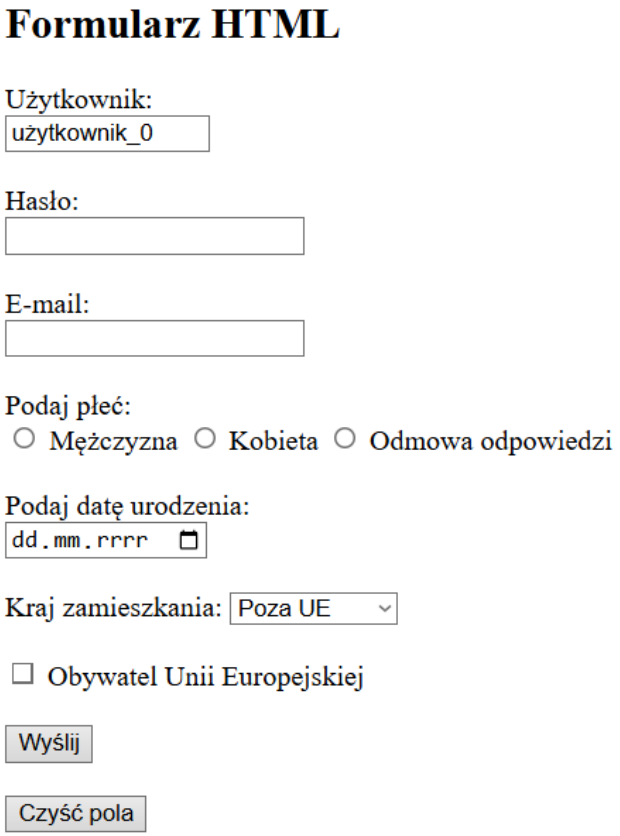
Rozbudujmy formularz o kolejne pola: płeć, datę urodzenia, kraj oraz obywatelstwo UE. Dla każdego elementu użyjemy innej kontrolki. Dla pierwszego będą to pola wyboru typu „radio”, dla drugiego pole typu „data”, dla trzeciego lista wybieralna natomiast dla czwartego „check box”. Kod formularza zostanie rozbudowany o następujące wiersze:

```
<label>Podaj płeć:</label><br>
<input type="radio" name="gender" value="male" id="male" required>
<label for="male">Mężczyzna</label>
<input type="radio" name="gender" value="female" id="female">
<label for="female">Kobieta</label>
<input type="radio" name="gender" value="refusal" id="refusal">
<label for="refusal">Odmowa odpowiedzi</label><br><br>
<label for="dofb">Podaj datę urodzenia:</label><br>
<input type="date" id="dofb" name="dofb" required><br><br>
<label for="country">Kraj zamieszkania: </label>
<select name="country" id="country">
      <option value="foreign">Poza UE</option>
      <option value="austria">Austria</option>
      <option value="belgium">Belgia</option>
      <option value="bulgaria">Bułgaria</option>
      <option value="croatia">Chorwacja</option>
      <option value="republic_of_cyprus">Cypr</option>
```

```
        <option value="czechia">Czechy</option>
        <option value="denmark">Dania</option>
        <option value="estonia">Estonia</option>
        <option value="finland">Finlandia</option>
        <option value="france">Francja</option>
        <option value="greece">Grecja</option>
        <option value="spain">Hiszpania</option>
        <option value="netherlands">Holandia</option>
        <option value="ireland">Irlandia</option>
        <option value="lithuania">Litwa</option>
        <option value="luxemburg">Luksemburg</option>
        <option value="latvia">Łotwa</option>
        <option value="malta">Malta</option>
        <option value="germany">Niemcy</option>
        <option value="poland">Polska</option>
        <option value="portugal">Portugalia</option>
        <option value="romania">Rumunia</option>
        <option value="slovakia">Słowacja</option>
        <option value="slovenia">Słowenia</option>
        <option value="sweden">Szwecja</option>
        <option value="hungary">Węgry</option>
        <option value="italy">Włochy</option>
</select><br><br>
<input type="checkbox" name="UE_citizen" id="UE_citizen">
<label for="UE_citizen">Obywatel Unii Europejskiej</label><br><br>
```

Formularz po zmianach wygląda jak na rys. 8.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Formularz HTML. Siedem etykiet i elementów: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: pole maskowane (ukryte znaki); E-mail: puste pole tekstowe; Podaj płeć: grupa przycisków radiowych z opcjami „Mężczyzna”, „Kobieta”, „Odmowa odpowiedzi”; Podaj datę urodzenia: pole typu data z placeholderem „dd.mm.rrrr”; Kraj zamieszkania: lista rozwijana z domyślną opcją „Poza UE”; Obywatel Unii Europejskiej: pole wyboru (checkbox). Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



Rys. 8. Formularz z dodatkowymi polami wyboru.

Pola wyboru typu „radio” domyślnie mają aktywną walidację zaznaczenia, tzn. musi być zaznaczone dokładnie jedno pole, pole daty również musi być wypełnione. Z kolei pole typu „check box” ma dwa stany: zaznaczone lub nie zaznaczone i nie musi być walidowane ponieważ w tym przypadku domyślną opcją jest „unchecked”. Ustawienie domyślne na zaznaczone można zrealizować za pomocą atrybutu checked=”checked”.

Wprowadźmy dodatkowe zmiany w wyglądzie i wykorzystajmy kaskadowe arkusze stylów (CSS) wewnątrz pliku HTML. Usuńmy nagłówek „h2” i wprowadźmy kilka modyfikatorów poprawiających wygląd formularza. Formatowanie dokumentu z wykorzystaniem CSS nie będzie tu omawiane, celem jest wyłącznie poprawa czytelności formularza. Pełny kod strony po zmianach wygląda następująco:

```
<html>
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-
scale=1.0">
	<title>Strona z formularzem HTML</title>
	<style>
		body {
			display: flex;
			justify-content: center;
			align-items: center;
			height: 100vh;
			margin: 0;
			background-color: #f0f0f0;
		}
		form {
			width: 500px;
			background-color: #fff;
			padding: 20px;
			border-radius: 8px;
			box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
		}
		fieldset {
			border: 1px solid black;
			padding: 10px;
			margin: 0;
		}
		legend {
			font-weight: bold;
			font-size: 1.2em;
			margin-bottom: 5px;
		}
		label {
			margin-bottom: 5px;
		}
		input[type="text"],
		input[type="email"],
		input[type="password"],
		input[type="date"] {
```

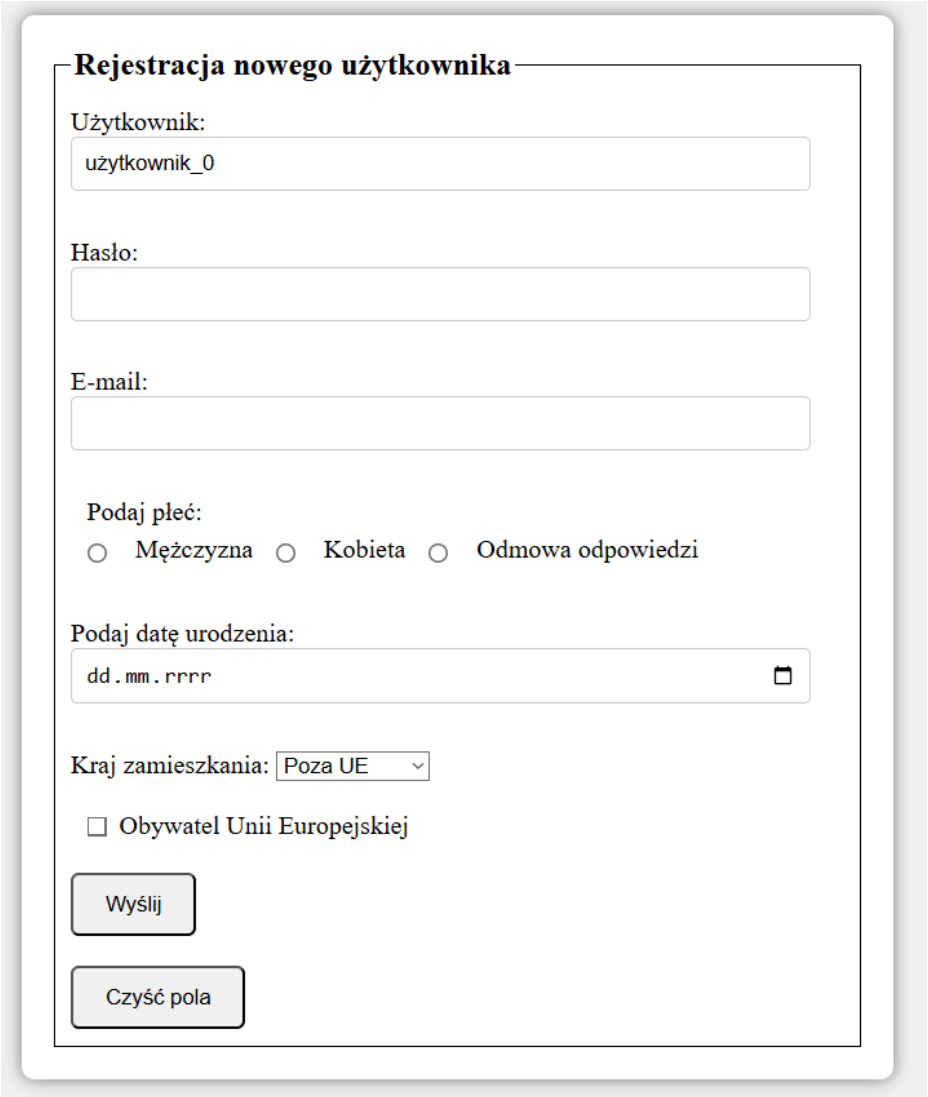
```
                width: calc(100% - 20px);
                padding: 8px;
                margin-bottom: 10px;
                box-sizing: border-box;
                border: 1px solid #ccc;
                border-radius: 4px;
            }
            .citizen_UE_check_country label{
                display: inline;
            }
            .gender-group {
                margin-bottom: 10px;
            }
            .gender-group label {
                display: inline-block;
                margin-left: 10px;
            }
            input[type="radio"] {
                margin-left: 10px;
                vertical-align: middle;
            }
            input[type="checkbox"] {
                margin-left: 10px;
                vertical-align: middle;
            }
            input[type="submit"],
            input[type="reset"]        {
                padding: 10px 20px;
                border-radius: 5px;
                cursor: pointer;
            }
        </style>
</head>
<body>
        <form>
        <fieldset>
        <legend>Rejestracja nowego użytkownika</legend>
            <label for="username">Użytkownik:</label><br>
```

```
            <input type="text" id="username" name="username"
value="użytkownik_0" minlength="6" maxlength="12" size="12"
required><br><br>
            <label for="password">Hasło:</label><br>
            <input type="password" id="password" name="password"
pattern="(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Hasło musi
zawierać co najmniej jedną cyfrę oraz małą i dużą literę, minimalna
długość to 8 znaków." required><br><br>
            <label for="email">E-mail:</label><br>
            <input type="email" id="email" name="email"
required><br><br>
            <div class="gender-group">
                  <label>Podaj płeć:</label><br>
                  <input type="radio" name="gender" value="male"
id="male" required>
                  <label for="male">Mężczyzna</label>
                  <input type="radio" name="gender" value="female"
id="female">
                  <label for="female">Kobieta</label>
                  <input type="radio" name="gender" value="refusal"
id="refusal">
                  <label for="refusal">Odmowa
odpowiedzi</label><br><br>
            </div>
            <label for="dofb">Podaj datę urodzenia:</label><br>
            <input type="date" id="dofb" name="dofb" required><br><br>
            <div class="citizen_UE_check_country">
            <label for="country">Kraj zamieszkania: </label>
            <select name="country" id="country">
                  <option value="foreign">Poza UE</option>
                  <option value="austria">Austria</option>
                  <option value="belgium">Belgia</option>
                  <option value="bulgaria">Bułgaria</option>
                  <option value="croatia">Chorwacja</option>
                  <option value="republic_of_cyprus">Cypr</option>
                  <option value="czechia">Czechy</option>
                  <option value="denmark">Dania</option>
                  <option value="estonia">Estonia</option>
                  <option value="finland">Finlandia</option>
                  <option value="france">Francja</option>
```

```
                <option value="greece">Grecja</option>
                <option value="spain">Hiszpania</option>
                <option value="netherlands">Holandia</option>
                <option value="ireland">Irlandia</option>
                <option value="lithuania">Litwa</option>
                <option value="luxemburg">Luksemburg</option>
                <option value="latvia">Łotwa</option>
                <option value="malta">Malta</option>
                <option value="germany">Niemcy</option>
                <option value="poland">Polska</option>
                <option value="portugal">Portugalia</option>
                <option value="romania">Rumunia</option>
                <option value="slovakia">Słowacja</option>
                <option value="slovenia">Słowenia</option>
                <option value="sweden">Szwecja</option>
                <option value="hungary">Węgry</option>
                <option value="italy">Włochy</option>
            </select><br><br>
            </div>
            <div class="citizen_UE_check_country">
                <input type="checkbox" name="UE_citizen"
id="UE_citizen">
                <label for="UE_citizen">Obywatel Unii
Europejskiej</label><br><br>
            </div>
            <input type="submit" value="Wyślij">
            <br><br><input type="reset" value="Czyść pola">
      </fieldset>
      </form>
</body>
</html>
```

Efektem powyższego kodu jest finalny formularz przedstawiony na rys. 9.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim. Nagłówek: Rejestracja nowego użytkownika. Siedem etykiet i elementów: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: pole maskowane (ukryte znaki); E-mail: puste pole tekstowe; Podaj płeć: grupa przycisków radiowych z opcjami „Mężczyzna”, „Kobieta”, „Odmowa odpowiedzi”; Podaj datę urodzenia: pole typu data z placeholderem „dd . mm . rrrr”; Kraj zamieszkania: lista rozwijana z domyślną opcją „Poza UE”; Obywatel Unii Europejskiej: pole wyboru (checkbox). Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



Rys. 9. Rozbudowany formularz z wykorzystaniem formatowania za pomocą kaskadowych arkuszy stylów.

Zwróćmy uwagę, że w podanym przykładzie zastosowano kilka metod walidacji danych wprowadzanych przez użytkownika bez korzystania z języka JavaScript. Również wszystkie elementy graficzne strony korzystają ze standardowego języka HTML. Jedną z metod walidacji było sprawdzenie, czy użytkownik wprowadził dane.

Kolejną metodą było sprawdzenie poprawności adresu e-mail, następnie stwierdzenie czy została wybrana data oraz jeden z przycisków „radio”. Te obiekty walidowane są domyślnie na poziomie HTML5. Zdefiniowane zostały także wymogi dotyczące hasła i tu również wykorzystane zostały metody HTML5.

3. Dynamicznie generowane formularze JavaScript

W poprzednim rozdziale zbudowany został formularz z wykorzystaniem wyłącznie HTML5. Jest to formularz statyczny, znaczy to że wszystkie jego elementy zostały zdefiniowane oddzielnie w dokumencie HTML. W tym rozdziale przedstawiony zostanie sposób dynamicznego generowania strony z formularzem za pomocą języka JavaScript. Wykorzystane zostaną dokładnie te same formatowania CSS zdefiniowane w nagłówku poprzedniego dokumentu, a wygląd i funkcjonalność formularza HTML5 zostanie w pełni odtworzona.
Rozpocznijmy od deklaracji trzech pól tekstowych: nazwa użytkownika, hasło oraz adres e-mail. Dodamy także dwa przyciski czyli „Wyślij” oraz „Czyść pola” działające identycznie, jak w formularzu HTML5. Z poprzedniego kodu programu usuńmy całą zawartość między znacznikami „body”, natomiast za znacznikiem „</html>” dodajmy następujący kod:

```
<script>
var br = document.createElement("br");
var fieldSet = document.createElement('fieldset');
var legendTag = document.createElement('legend');
legendTag.textContent = 'Rejestracja nowego użytkownika';

var formField = document.createElement("form");
formField.setAttribute('method','post');
formField.setAttribute('action','');

var labelUser = document.createElement('label');
labelUser.textContent = 'Użytkownik:';

var inputUser = document.createElement("input");
inputUser.setAttribute('type',"text");
inputUser.setAttribute('id',"username");
inputUser.setAttribute('name',"username");
inputUser.setAttribute('value',"użytkownik_0");
inputUser.setAttribute('minlength',"6");
inputUser.setAttribute('maxlength',"12");
```

```
inputUser.setAttribute('required', '');

var labelPassword = document.createElement('label');
labelPassword.textContent = 'Hasło:';

var inputPassword = document.createElement("input");
inputPassword.setAttribute('type',"password");
inputPassword.setAttribute('id',"password");
inputPassword.setAttribute('name',"password");
inputPassword.setAttribute('pattern',"(?=.*[0-9])(?=.*[a-z])(?=.*[A-
Z]).{8,}");
inputPassword.setAttribute('title',"Hasło musi zawierać co najmniej
jedną cyfrę oraz małą i dużą literę, minimalna długość to 8
znaków.");
inputPassword.setAttribute('required', '');

var labelEmail = document.createElement('label');
labelEmail.textContent = 'E-mail:';

var inputEmail = document.createElement("input");
inputEmail.setAttribute('type',"email");
inputEmail.setAttribute('id',"email");
inputEmail.setAttribute('name',"email");
inputEmail.setAttribute('required', '');

var buttonSubmit = document.createElement("input");
buttonSubmit.setAttribute('type',"submit");
buttonSubmit.setAttribute('value',"Wyślij");

var buttonReset = document.createElement("input");
buttonReset.setAttribute('type',"reset");
buttonReset.setAttribute('value',"Czyść pola");

formField.appendChild(fieldSet);
fieldSet.appendChild(legendTag);
fieldSet.appendChild(labelUser);
fieldSet.appendChild(inputUser);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelPassword);
```

```
fieldSet.appendChild(inputPassword);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelEmail);
fieldSet.appendChild(inputEmail);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(buttonSubmit);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(buttonReset);

document.body.appendChild(formField);
</script>
```

W efekcie uzyskamy formularz identyczny z tym na rys. 5 i o dokładnie tej samej funkcjonalności i wymaganiach co do pól (długość nazwy użytkownika, znaki w haśle). W powyższym kodzie obiekt formularza został zdefiniowany pod zmienną „formField". Obiekty służące do opisywania elementów graficznych zdefiniowane są jako elementy „label". Elementy interaktywne z użytkownikiem to „input" posiadające własne atrybuty. Rodzaje atrybutów oraz ich wartości są tożsame z HTML5, przykładowo, w HTML5 deklaracja pola wprowadzania adresu e-mail wygląda następująco:

```
<label for="email">E-mail:</label><br>
<input type="email" id="email" name="email" required><br><br>
```

Z kolei odpowiednikiem w JavaScript jest:

```
var labelEmail = document.createElement('label');
labelEmail.textContent = 'E-mail:';
var inputEmail = document.createElement("input");
inputEmail.setAttribute('type',"email");
inputEmail.setAttribute('id',"email");
inputEmail.setAttribute('name',"email");
inputEmail.setAttribute('required', '');
```

Warto zwrócić uwagę na rozdzielenie części deklaratywnej od części budowania dokumentu (DOM - Document Object Model). Budowanie dokumentu zaczyna się linią „formField.appendChild(fieldSet);", następnie definiowane są poszczególne

obiekty w kolejności ich występowania w formularzu. Elementy umieszczane są w kontenerach, np. elementy „input" oraz „label" znajdują się wewnątrz elementu „fieldSet", a ten umieszczony jest w kontenerze „form". Na końcu całość zostaje umieszczona między znacznikami „<body> </body>" dokumentu HTML za pomocą polecenia „document.body.appendChild(formField);" (konteneryzacja elementu „form" do wnętrza „body"). Warto zwrócić uwagę, że dla celów uzyskania identycznego wyglądu formularza zadeklarowano tag „
" jako oddzielną zmienną za pomocą „var br = document.createElement("br");". Wprowadzenie tego tagu do dokumentu odbywa się poleceniem „fieldSet.appendChild(br.cloneNode());". Tym samym zaprezentowano w jaki sposób pojedynczy obiekt może zostać klonowany w obrębie dokumentu bez konieczności deklarowania wielu identycznych zmiennych.
Pole do wprowadzania daty urodzin zadeklarowane zostanie następująco:

```
var labelDofb = document.createElement('label');
labelDofb.textContent = "Podaj datę urodzenia:";
var inputDofb = document.createElement("input");
inputDofb.setAttribute('type',"date");
inputDofb.setAttribute('id',"dofb");
inputDofb.setAttribute('name',"dofb");
inputDofb.setAttribute('required', '');
```

Aby pole to pojawiło się w formularzu należy uzupełnić kod skryptu o wiersze:

```
fieldSet.appendChild(labelDofb);
fieldSet.appendChild(inputDofb);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
```

Na potrzeby formatowania grupy pola „checkbox" zdefiniowano uprzednio oddzielną klasę CSS o nazwie „citizen_UE_check_country". Aby wykorzystać tą klasę należy utworzyć kontener „div" i dodać nazwę klasy metodą „classList":

```
var classCitizen_UE_check_country = document.createElement('div');
classCitizen_UE_check_country.classList.add('citizen_UE_check_countr
y');
```

Teraz następuje deklaracja elementów checkboxa:

```
var labelCheckbox = document.createElement('label');
labelCheckbox.textContent = "Obywatel Unii Europejskiej";
```

```
var checkbox = document.createElement("input");
checkbox.setAttribute('type',"checkbox");
checkbox.setAttribute('id',"UE_citizen");
checkbox.setAttribute('name',"UE_citizen");
```

Wykorzystanie uprzednio zadeklarowanej klasy CSS odbywa się na etapie budowania drzewa DOM poprzez deklarację położenia poszczególnych elementów w następujący sposób:

```
fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(checkbox);
classCitizen_UE_check_country.appendChild(labelCheckbox);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());
```

Tutaj element „checkbox” zawarty jest w kontenerz o nazwie „classCitizen_UE_check_country”, a ten zawarty jest wewnątrz kontenera o nazwie „fieldSet”. W podobny sposób skonteneryzowane zostaną kontrolki wyboru płci oraz kraju zamieszkania, przy czym na potrzeby wylistowania elementów wyboru wykorzystana zostanie tablica zmiennych typu „string”. Kontrolki „radio” zdefiniowane zostaną następująco:

```
var genderGroup = document.createElement('div');
genderGroup.classList.add('gender-group');

var labelGender = document.createElement('label');
labelGender.textContent = 'Podaj płeć:';

const genderOptions = [
     {value: 'male', id: 'male', text: 'Mężczyzna', required: true},
     {value: 'female', id: 'female', text: 'Kobieta'},
     {value: 'refusal', id: 'refusal', text: 'Odmowa odpowiedzi'}
];
```

Natomiast drzewo DOM zostanie zmodyfikowane w następujący sposób:

```
fieldSet.appendChild(genderGroup);
genderGroup.appendChild(labelGender);
genderGroup.appendChild(br.cloneNode());
```

```
genderOptions.forEach(opt => {
      var radio = document.createElement('input');
      radio.type = 'radio';
      radio.name = 'gender';
      radio.value = opt.value;
      radio.id = opt.id;
      if (opt.required) radio.required = true;
      genderGroup.appendChild(radio);

      var labelGenderList = document.createElement('label');
      labelGenderList.htmlFor = opt.id;
      labelGenderList.textContent = opt.text;
      genderGroup.appendChild(labelGenderList);
});
genderGroup.appendChild(br.cloneNode());
genderGroup.appendChild(br.cloneNode());
```

Warto zwrócić uwagę, że w tym przypadku wykorzystano inny sposób ustawiania atrybutów obiektów kontrolek. Mianowicie zamiast deklaracji „setAttribute” zastosowano odwołanie do właściwości obiektu. Wybór opcji z listy rozwijalnej przedstawia się bardzo podobnie. Deklaracja listy wybieralnej wraz z jej wypełnieniem predefiniowaną tablicą prezentuje się następująco:

```
var labelCountry = document.createElement('label');
labelCountry.textContent = "Kraj zamieszkania: ";

var countrySelect = document.createElement('select');
countrySelect.name = 'country';
countrySelect.id   = 'country'

const countries = [
      {value: 'foreign', text: 'Poza UE'},
      {value: 'austria', text: 'Austria'},
      {value: 'belgium', text: 'Belgia'},
      {value: 'bulgaria', text: 'Bułgaria'},
      {value: 'croatia', text: 'Chorwacja'},
      {value: 'republic_of_cyprus', text: 'Cypr'},
      {value: 'czechia', text: 'Czechy'},
      {value: 'denmark', text: 'Dania'},
```

```
    {value: 'estonia', text: 'Estonia'},
    {value: 'finland', text: 'Finlandia'},
    {value: 'france', text: 'Francja'},
    {value: 'greece', text: 'Grecja'},
    {value: 'spain', text: 'Hiszpania'},
    {value: 'netherlands', text: 'Holandia'},
    {value: 'ireland', text: 'Irlandia'},
    {value: 'lithuania', text: 'Litwa'},
    {value: 'luxemburg', text: 'Luksemburg'},
    {value: 'latvia', text: 'Łotwa'},
    {value: 'malta', text: 'Malta'},
    {value: 'germany', text: 'Niemcy'},
    {value: 'poland', text: 'Polska'},
    {value: 'portugal', text: 'Portugalia'},
    {value: 'romania', text: 'Rumunia'},
    {value: 'slovakia', text: 'Słowacja'},
    {value: 'slovenia', text: 'Słowenia'},
    {value: 'sweden', text: 'Szwecja'},
    {value: 'hungary', text: 'Węgry'},
    {value: 'italy', text: 'Włochy'}
];

countries.forEach(c => {
  var optionsCountries = document.createElement('option');
  optionsCountries.value = c.value;
  optionsCountries.textContent = c.text;
  countrySelect.appendChild(optionsCountries);
});
```

Obiekt zostanie zbudowany dzięki następującemu fragmentowi kodu:

```
fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(labelCountry);
classCitizen_UE_check_country.appendChild(countrySelect);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());
```

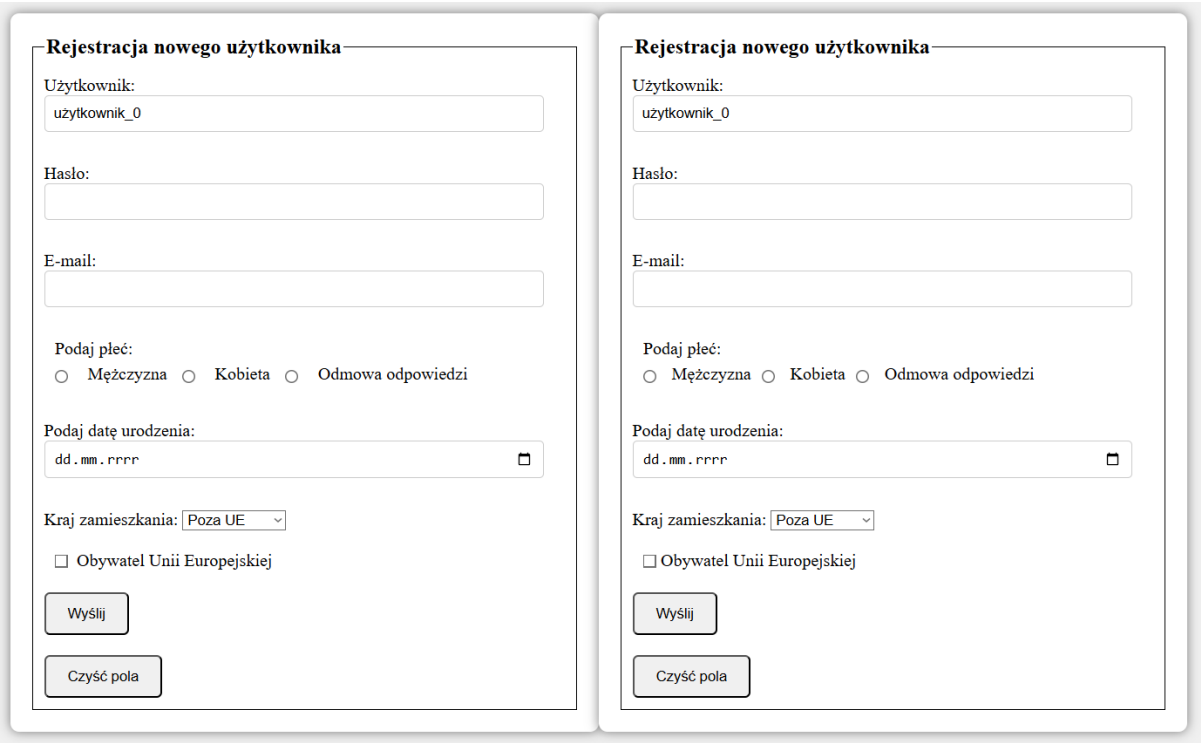
Zwróćmy uwagę na fakt zastosowania dokładnie tej samej klasy CSS, co w przypadku kontrolki „checkbox”. Ponieważ klasa ta została już dodana do obiektu „classList” wcześniej, tutaj wystarczy tylko się na nią powołać bez redeklaracji.

Istnieje także inny sposób wypełniania listy wybieralnej z pominięciem pętli oraz zmiennych tablicowych. Jest to sposób mniej wygodny w użyciu, zajmujący więcej linii kodu. W naszym przypadku część deklaratywna (pierwsze dwie opcje) wyglądałaby następująco:

```
var labelCountry = document.createElement('label');
labelCountry.textContent = "Kraj zamieszkania: ";

var countrySelect = document.createElement('select');
countrySelect.name = 'country';
countrySelect.id   = 'country'
var foreign = document.createElement('option');
foreign.value = 'foreign';
foreign.textContent = 'Poza UE';
var austria = document.createElement('option');
austria.value = ' austria';
austria.textContent = 'Austria';
countrySelect.appendChild(foreign);
countrySelect.appendChild(austria);
```

W tym momencie dokument HTML z dynamicznie generowaną treścią za pomocą JavaScript jest już gotowy. Formularz wygląda identycznie, jak w przypadku HTML5 (to zasługa głównie CSS) oraz spełnia wszystkie wymagania tam zdefiniowane. Na rys. 10. zaprezentowano obydwa formularze w jednym dokumencie HTML, po lewej stronie znajduje się formularz wyłącznie HTML5, po prawej stronie wyłącznie JavaScript.

[Tekst alternatywny. Formularz HTML z elementami w języku polskim przedstawiony dwukrotnie. Jeden po lewej stronie, jeden po prawej stronie. Obydwa są identyczne. Nagłówek: Rejestracja nowego użytkownika. Siedem etykiet i elementów: Użytkownik: pole tekstowe wypełnione wartością „użytkownik_0”; Hasło: pole maskowane (ukryte znaki); E-mail: puste pole tekstowe; Podaj płeć: grupa przycisków radiowych z opcjami „Mężczyzna”, „Kobieta”, „Odmowa odpowiedzi”; Podaj datę urodzenia: pole typu data z placeholderem „dd . mm . rrrr”; Kraj zamieszkania: lista rozwijana z domyślną opcją „Poza UE”; Obywatel Unii Europejskiej: pole wyboru (checkbox). Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



Rys. 10. Dwa formularze w dwóch różnych technologiach w jednym dokumencie HTML.

Pełna treść dokumentu HTML do dynamicznego wygenerowania zaprezentowanego formularza wygląda następująco:

```
<html>
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-
scale=1.0">
     <title>Strona z formularzem HTML</title>
     <style>
```

```
body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      background-color: #f0f0f0;
}
form {
      width: 500px;
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
}
fieldset {
      border: 1px solid black;
      padding: 10px;
      margin: 0;
}
legend {
      font-weight: bold;
      font-size: 1.2em;
      margin-bottom: 5px;
}
label {
      margin-bottom: 5px;
}
input[type="text"],
input[type="email"],
input[type="password"],
input[type="date"] {
      width: calc(100% - 20px);
      padding: 8px;
      margin-bottom: 10px;
      box-sizing: border-box;
      border: 1px solid #ccc;
      border-radius: 4px;
}
.citizen_UE_check_country label{
```

```
                display: inline;
            }
            .gender-group {
                margin-bottom: 10px;
            }
            .gender-group label {
                display: inline-block;
                margin-left: 10px;
            }
            input[type="radio"] {
                margin-left: 10px;
                vertical-align: middle;
            }
            input[type="checkbox"] {
                margin-left: 10px;
                vertical-align: middle;
            }
            input[type="submit"],
            input[type="reset"]        {
                padding: 10px 20px;
                border-radius: 5px;
                cursor: pointer;
            }
        </style>
</head>

<body>
</body>
</html>

<script>
var br = document.createElement("br");
var fieldSet = document.createElement('fieldset');
var legendTag = document.createElement('legend');
legendTag.textContent = 'Rejestracja nowego użytkownika';

var formField = document.createElement("form");
formField.setAttribute('method','post');
formField.setAttribute('action','');
```

```
var labelUser = document.createElement('label');
labelUser.textContent = 'Użytkownik:';

var inputUser = document.createElement("input");
inputUser.setAttribute('type',"text");
inputUser.setAttribute('id',"username");
inputUser.setAttribute('name',"username");
inputUser.setAttribute('value',"użytkownik_0");
inputUser.setAttribute('minlength',"6");
inputUser.setAttribute('maxlength',"12");
inputUser.setAttribute('required', '');

var labelPassword = document.createElement('label');
labelPassword.textContent = 'Hasło:';

var inputPassword = document.createElement("input");
inputPassword.setAttribute('type',"password");
inputPassword.setAttribute('id',"password");
inputPassword.setAttribute('name',"password");
inputPassword.setAttribute('pattern',"(?=.*[0-9])(?=.*[a-z])(?=.*[A-
Z]).{8,}");
inputPassword.setAttribute('title',"Hasło musi zawierać co najmniej
jedną cyfrę oraz małą i dużą literę, minimalna długość to 8
znaków.");
inputPassword.setAttribute('required', '');

var labelEmail = document.createElement('label');
labelEmail.textContent = 'E-mail:';

var inputEmail = document.createElement("input");
inputEmail.setAttribute('type',"email");
inputEmail.setAttribute('id',"email");
inputEmail.setAttribute('name',"email");
inputEmail.setAttribute('required', '');

var genderGroup = document.createElement('div');
genderGroup.classList.add('gender-group');

var labelGender = document.createElement('label');
labelGender.textContent = 'Podaj płeć:';
```

```
const genderOptions = [
     {value: 'male', id: 'male', text: 'Mężczyzna', required: true},
     {value: 'female', id: 'female', text: 'Kobieta'},
     {value: 'refusal', id: 'refusal', text: 'Odmowa odpowiedzi'}
];

var labelDofb = document.createElement('label');
labelDofb.textContent = "Podaj datę urodzenia:";

var inputDofb = document.createElement("input");
inputDofb.setAttribute('type',"date");
inputDofb.setAttribute('id',"dofb");
inputDofb.setAttribute('name',"dofb");
inputDofb.setAttribute('required', '');

var classCitizen_UE_check_country = document.createElement('div');
classCitizen_UE_check_country.classList.add('citizen_UE_check_countr
y');

var labelCountry = document.createElement('label');
labelCountry.textContent = "Kraj zamieszkania: ";

var countrySelect = document.createElement('select');
countrySelect.name = 'country';
countrySelect.id   = 'country'

const countries = [
     {value: 'foreign', text: 'Poza UE'},
     {value: 'austria', text: 'Austria'},
     {value: 'belgium', text: 'Belgia'},
     {value: 'bulgaria', text: 'Bułgaria'},
     {value: 'croatia', text: 'Chorwacja'},
     {value: 'republic_of_cyprus', text: 'Cypr'},
     {value: 'czechia', text: 'Czechy'},
     {value: 'denmark', text: 'Dania'},
     {value: 'estonia', text: 'Estonia'},
     {value: 'finland', text: 'Finlandia'},
     {value: 'france', text: 'Francja'},
     {value: 'greece', text: 'Grecja'},
```

```
    {value: 'spain', text: 'Hiszpania'},
    {value: 'netherlands', text: 'Holandia'},
    {value: 'ireland', text: 'Irlandia'},
    {value: 'lithuania', text: 'Litwa'},
    {value: 'luxemburg', text: 'Luksemburg'},
    {value: 'latvia', text: 'Łotwa'},
    {value: 'malta', text: 'Malta'},
    {value: 'germany', text: 'Niemcy'},
    {value: 'poland', text: 'Polska'},
    {value: 'portugal', text: 'Portugalia'},
    {value: 'romania', text: 'Rumunia'},
    {value: 'slovakia', text: 'Słowacja'},
    {value: 'slovenia', text: 'Słowenia'},
    {value: 'sweden', text: 'Szwecja'},
    {value: 'hungary', text: 'Węgry'},
    {value: 'italy', text: 'Włochy'}
];

countries.forEach(c => {
  var optionsCountries = document.createElement('option');
  optionsCountries.value = c.value;
  optionsCountries.textContent = c.text;
  countrySelect.appendChild(optionsCountries);
});

var labelCheckbox = document.createElement('label');
labelCheckbox.textContent = "Obywatel Unii Europejskiej";

var checkbox = document.createElement("input");
checkbox.setAttribute('type',"checkbox");
checkbox.setAttribute('id',"UE_citizen");
checkbox.setAttribute('name',"UE_citizen");

var buttonSubmit = document.createElement("input");
buttonSubmit.setAttribute('type',"submit");
buttonSubmit.setAttribute('value',"Wyślij");

var buttonReset = document.createElement("input");
buttonReset.setAttribute('type',"reset");
buttonReset.setAttribute('value',"Czyść pola");
```

```
formField.appendChild(fieldSet);
fieldSet.appendChild(legendTag);
fieldSet.appendChild(labelUser);
fieldSet.appendChild(inputUser);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelPassword);
fieldSet.appendChild(inputPassword);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelEmail);
fieldSet.appendChild(inputEmail);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());

fieldSet.appendChild(genderGroup);
genderGroup.appendChild(labelGender);
genderGroup.appendChild(br.cloneNode());

genderOptions.forEach(opt => {
      var radio = document.createElement('input');
      radio.type = 'radio';
      radio.name = 'gender';
      radio.value = opt.value;
      radio.id = opt.id;
      if (opt.required) radio.required = true;
      genderGroup.appendChild(radio);

      var labelGenderList = document.createElement('label');
      labelGenderList.htmlFor = opt.id;
      labelGenderList.textContent = opt.text;
      genderGroup.appendChild(labelGenderList);
});
genderGroup.appendChild(br.cloneNode());
genderGroup.appendChild(br.cloneNode());

fieldSet.appendChild(labelDofb);
fieldSet.appendChild(inputDofb);
fieldSet.appendChild(br.cloneNode());
```

```
fieldSet.appendChild(br.cloneNode());

fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(labelCountry);
classCitizen_UE_check_country.appendChild(countrySelect);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());

fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(checkbox);
classCitizen_UE_check_country.appendChild(labelCheckbox);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());

fieldSet.appendChild(buttonSubmit);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(buttonReset);

document.body.appendChild(formField);
</script>
```

4. Walidacja danych użytkownika za pomocą JavaScript

W poprzednich rozdziałach dane użytkownika oraz ich obecność były walidowane za pomocą metod HTML5 zaimplementowanych w interpretatorach dokumentów HTML. Pola były sprawdzane w chwili aktywacji przycisku „Wyślij”. Spróbujmy teraz rozbudować dokument HTML o funkcje pozwalające na walidację po wpisaniu danych przez użytkownika i informowanie go o tym w dodatkowy sposób. Zakładamy informowanie użytkownika o błędach poprzez zastosowanie czerwonego koloru obramowania pola z błędem. Jeśli pole jej wypełnione poprawnie kolor obramowania będzie zielony. Ta sama zasada dotyczy obramowania przycisku „Wyślij”. Ponadto poniżej pola wprowadzania użytkownik powinien znaleźć informację tekstową o błędzie. Wykorzystany zostanie dokument z dynamicznie generowanym formularzem, zmiany będą polegać na rozbudowie sekcji skryptu. Celem zachowania przejrzystość kodu i rozdzielenia części służącej budowaniu dokumentu od części walidacyjne dodatkowe wiersze kodu zostaną dodane pomiędzy wiersz „document.body.appendChild(formField);” a „</script>” natomiast nie zostaną wprowadzona żadne zmiany do istniejącego już kodu w dokumencie HTML.

Rozpocznijmy od podkolorowania obwódki przycisku „Wyślij”. Kolor ma się zmieniać w zależności od poprawności wypełnienia formularza. Ponadto będzie wyświetlane okno informacyjne po aktywowaniu przycisku „Wyślij” z poprawnie wypełnionym formularzem. Część walidacyjna zostanie umieszczona między wierszem „document.body.appendChild(formField);” a „</script>”. Na początku tworzymy referencje do elementów formularza i przycisków:

```
var form = formField;
var submitBtn = buttonSubmit;
var resetBtn = buttonReset;
```

Teraz definiujemy funkcję ustawiającą kolor przycisku „Wyślij”:

```
function updateSubmitButton() {
      if (form.checkValidity()) {
            submitBtn.style.borderColor = 'green';
      } else {
            submitBtn.style.borderColor = 'red';
      }
}
```

Tą funkcję można także zapisać następująco:

```
function updateSubmitButton() {
      submitBtn.style.borderColor = form.checkValidity() ? 'green' :
'red';
}
```

Na potrzeby wyboru właściwego koloru wykorzystywana jest metoda „checkValidity” zwracającą zmienną logiczną. Zrealizujmy wymuszenie walidacji i ustawienie koloru przycisku przy pierwszym uruchomieniu strony:

```
form.reportValidity();
updateSubmitButton();
```

Kolor przycisku aktualizujemy przy każdej zmianie formularza:

```
form.addEventListener('input', updateSubmitButton);
form.addEventListener('change', updateSubmitButton);
```

Zajmijmy się teraz przyciskiem „Czyść pola”, wymuszamy ponowną walidację i ustawiamy kolor:

```
resetBtn.addEventListener('click', function() {
	setTimeout(function() {
		form.reportValidity();
		updateSubmitButton();
	}, 0);
});
```

Zrealizujmy jeszcze obsługę wysłania formularza, jeśli wystąpią błędy pokazujemy je jak do tej pory. Jeśli formularz jest poprawny wyświetli się okno z informacją, a same pola zostaną zresetowane.

```
form.addEventListener('submit', function(e) {
	e.preventDefault();
	if (!form.checkValidity()) {
		form.reportValidity();
		return;
	}
	alert('Formularz został poprawnie wypełniony.');
	form.reset();
	form.reportValidity();
	updateSubmitButton();
});
```

Zajmijmy się teraz walidacją wszystkich pól formularza i zmianie koloru obramowania kontrolek w zależności od statusu walidacji. Dodajmy listę pól do walidacji, pomijamy przyciski „submit” i „reset”. Pomińmy także pole wyboru kraju oraz obywatelstwa UE. Grupa przycisków typu „radio” wymaga oddzielnej grupy:

```
var fieldsToValidate = [
	inputUser,
	inputPassword,
	inputEmail,
	inputDofb
];

var genderRadios = form.elements['gender'];
```

Funkcja ustawiająca obramówkę dla pojedynczego pola wygląda następująco:

```
function validateFieldBorder(field) {
      field.style.border = field.checkValidity()
            ? '2px solid green'
            : '2px solid red';
}
```

Funkcja walidująca grupę „radio” została zdefiniowana następująco:

```
function validateGenderGroup() {
      var isChecked = Array.from(genderRadios).some(r => r.checked);
      genderGroup.style.border = isChecked
            ? '2px solid green'
            : '2px solid red';
}
```

Zdefiniowana została funkcja uruchamiająca pełną walidację w następujący sposób:

```
function validateAllFields() {
      fieldsToValidate.forEach(validateFieldBorder);
      validateGenderGroup();
      updateSubmitButton();
}
```

Teraz podpinamy walidację na input i change:

```
fieldsToValidate.forEach(el => {
      el.addEventListener('input', validateAllFields);
      el.addEventListener('change', validateAllFields);
});
Array.from(genderRadios).forEach(r => {
      r.addEventListener('change', validateAllFields);
});
```

Zmieńmy obsługę czyszczenia formularza i wymuśmy walidację przy pierwszym załadowaniu strony:

```
resetBtn.addEventListener('click', function() {
      setTimeout(validateAllFields, 0);
```

```
});

validateAllFields();
```

Poinformujmy teraz użytkownika o błędach dodatkowym komunikatem pod polem do wprowadzania tekstu. Na początku zdefiniujmy kontenery na komunikaty o błędach. Wykorzystujemy tu pola „name” oraz pola „id”:

```
var errorContainers = {};
fieldsToValidate.forEach(function(field) {
      var err = document.createElement('div');
      err.style.color = 'red';
      err.className = 'error-message';
      field.insertAdjacentElement('afterend', err);
      errorContainers[field.name || field.id] = err;
});
var genderError = document.createElement('div');
genderError.style.color = 'red';
genderError.className = 'error-message';
genderGroup.insertAdjacentElement('afterend', genderError);
```

Zmodyfikujmy następująco funkcję ustawiającą obramówkę dla pojedynczego pola:

```
function validateFieldBorder(field) {
      if (field.checkValidity()) {
            field.style.border = '2px solid green';
            errorContainers[field.name || field.id].textContent = '';
      } else {
            field.style.border = '2px solid red';
            // pokazujemy natywny komunikat walidacji
            errorContainers[field.name || field.id].textContent =
field.validationMessage;
      }
}
```

Funkcja walidująca grupę „radio” również powinna być rozbudowana:

```
function validateGenderGroup() {
      var isChecked = Array.from(genderRadios).some(r => r.checked);
      genderGroup.style.border = isChecked
```

```
            ? '2px solid green'
            : '2px solid red';
      genderError.textContent = isChecked
            ? ''
            : 'Proszę zaznaczyć płeć';
}
```

Wprowadźmy dodatkową zmianę pozwalającą na zaprezentowanie danych wprowadzonych przez użytkownika w formie zapisu zgodnej z JSON. Dane w tym formacie będą wyświetlone w oknie komunikacyjnym po poprawnym wypełnieniu formularza i aktywacji przycisku „Wyślij". W tym celu rozbudujemy obsługę wysyłania formularza o zmienne zbierające dane oraz użyjemy metody JSON.stringify celem konwersji wartości JavaScript do formatu JSON:

```
form.addEventListener('submit', function(e) {
      e.preventDefault();
      if (!form.checkValidity()) {
            form.reportValidity();
            return;
      }
      var formData = {
            username: inputUser.value,
            password: inputPassword.value,
            email: inputEmail.value,
            gender: Array.from(genderRadios).find(r =>
r.checked).value,
            dofb: inputDofb.value,
            country: countrySelect.value,
            UE_citizen: checkbox.checked
      };
      alert(
            'Formularz został poprawnie wypełniony:\n' +
            JSON.stringify(formData, null, 2)
      );
      form.reset();
      setTimeout(validateAllFields, 0);
});
```

Dodajmy dodatkowy warunek poprawnego wysłania formularza związany z datą urodzenia. Niech skrypt sprawdzi wiek osoby wysyłającej i zablokuje możliwość

wysłania formularza dla osób poniżej 18 roku życia (oczywiście na podstawie wprowadzonej daty urodzenia). W tym celu rozbudujmy obsługę wysyłania formularza o następujący kod odejmujący obecny rok od roku podanego przez użytkownika:

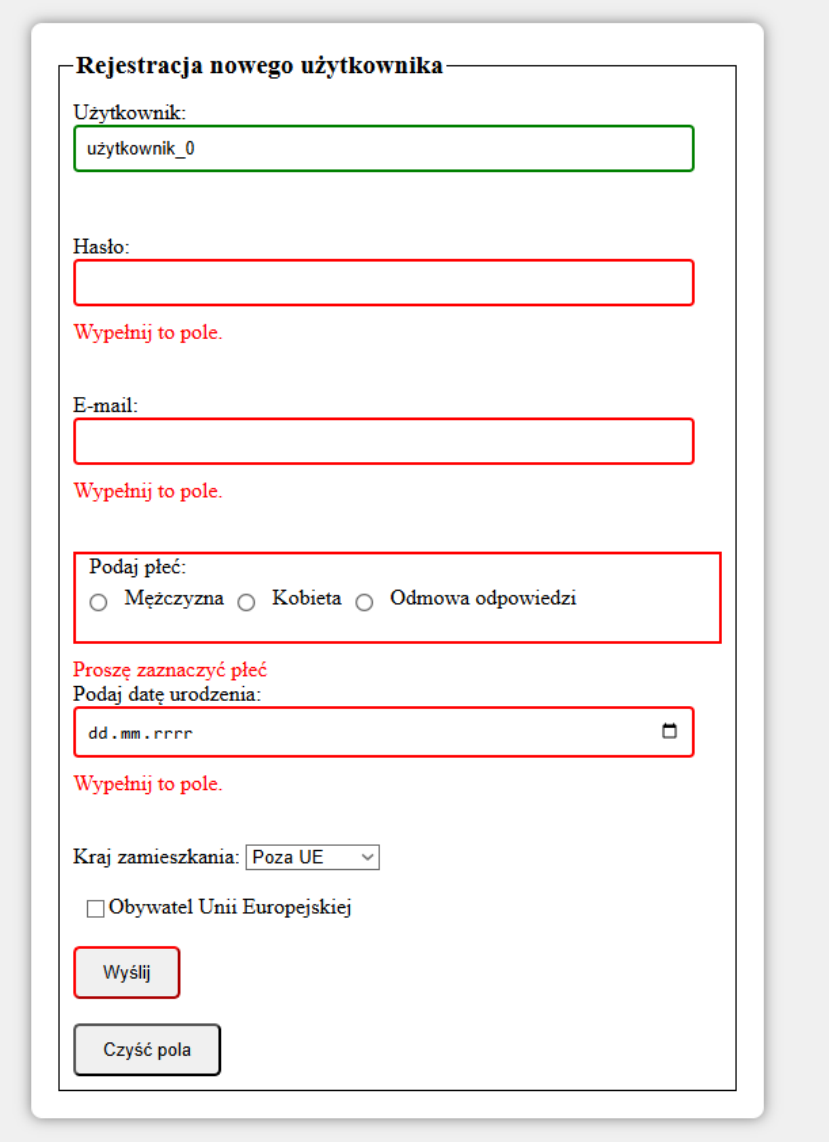
```
var birthYear = new Date(inputDofb.value).getFullYear();
var currentYear = new Date().getFullYear();
var ageYears = currentYear - birthYear;
if (ageYears < 18) {
     errorContainers['dofb'].textContent = 'Musisz mieć co najmniej
18 lat.';
     inputDofb.style.border = '2px solid red';
     return;
} else {
     errorContainers['dofb'].textContent = '';
     inputDofb.style.border = '2px solid green';
}
```

Zakończymy część walidacyjną możliwością zapisania lokalnie pliku z danymi w formacie JSON. Poniższy kod również umieścimy w sekcji przechwytu kliknięcia przycisku „Wyślij”:

```
var blob = new Blob([JSON.stringify(formData, null, 2)], {type:
'application/json'});
var url = URL.createObjectURL(blob);
var link = document.createElement('a');
link.href = url;
link.download = 'formData.json';
document.body.appendChild(link);
link.click();
document.body.removeChild(link);
URL.revokeObjectURL(url)
```

Formularz wraz z walidatorem jest skończony, zaraz po uruchomieniu strony użytkownik zobaczy formularz jak na rys. 11.

[Tekst alternatywny. Formularz HTML w języku polskim zatytułowany „Rejestracja nowego użytkownika”. Siedem etykiet i elementów: Użytkownik: pole tekstowe z wartością „użytkownik_0” wypełnione poprawnie (zaznaczone na zielono); Hasło: puste pole maskowane, obramowane na czerwono z komunikatem „Wypełnij to pole.”; E-mail: puste pole tekstowe, obramowane na czerwono z komunikatem „Wypełnij to pole.”; Podaj płeć: grupa przycisków radiowych („Mężczyzna”, „Kobieta”, „Odmowa odpowiedzi”), sekcja obramowana na czerwono z komunikatem „Proszę zaznaczyć płeć.”; Podaj datę urodzenia: pole typu data z placeholderem „dd.mm.rrrr”, obramowane na czerwono z komunikatem „Wypełnij to pole.”; Kraj zamieszkania: lista rozwijana z wybraną opcją „Poza UE”; Obywatel Unii Europejskiej: pole wyboru (checkbox) nieaktywne. Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



Rys. 11. Formularz z walidatorem oraz komunikatami o błędach zaraz po uruchomieniu strony.

Pełny kod skrypu JavaScript dla tego dokumentu HTML zaprezentowano poniżej:

```
<script>
var br = document.createElement("br");
var fieldSet = document.createElement('fieldset');
var legendTag = document.createElement('legend');
legendTag.textContent = 'Rejestracja nowego użytkownika';

var formField = document.createElement("form");
formField.setAttribute('method','post');
formField.setAttribute('action','');

var labelUser = document.createElement('label');
labelUser.textContent = 'Użytkownik:';

var inputUser = document.createElement("input");
inputUser.setAttribute('type',"text");
inputUser.setAttribute('id',"username");
inputUser.setAttribute('name',"username");
inputUser.setAttribute('value',"użytkownik_0");
inputUser.setAttribute('minlength',"6");
inputUser.setAttribute('maxlength',"12");
inputUser.setAttribute('required', '');

var labelPassword = document.createElement('label');
labelPassword.textContent = 'Hasło:';

var inputPassword = document.createElement("input");
inputPassword.setAttribute('type',"password");
inputPassword.setAttribute('id',"password");
inputPassword.setAttribute('name',"password");
inputPassword.setAttribute('pattern',"(?=.*[0-9])(?=.*[a-z])(?=.*[A-
Z]).{8,}");
inputPassword.setAttribute('title',"Hasło musi zawierać co najmniej
jedną cyfrę oraz małą i dużą literę, minimalna długość to 8
znaków.");
inputPassword.setAttribute('required', '');

var labelEmail = document.createElement('label');
labelEmail.textContent = 'E-mail:';
```

```
var inputEmail = document.createElement("input");
inputEmail.setAttribute('type',"email");
inputEmail.setAttribute('id',"email");
inputEmail.setAttribute('name',"email");
inputEmail.setAttribute('required', '');

var genderGroup = document.createElement('div');
genderGroup.classList.add('gender-group');

var labelGender = document.createElement('label');
labelGender.textContent = 'Podaj płeć:';

const genderOptions = [
     {value: 'male', id: 'male', text: 'Mężczyzna', required: true},
     {value: 'female', id: 'female', text: 'Kobieta'},
     {value: 'refusal', id: 'refusal', text: 'Odmowa odpowiedzi'}
];

var labelDofb = document.createElement('label');
labelDofb.textContent = "Podaj datę urodzenia:";

var inputDofb = document.createElement("input");
inputDofb.setAttribute('type',"date");
inputDofb.setAttribute('id',"dofb");
inputDofb.setAttribute('name',"dofb");
inputDofb.setAttribute('required', '');

var classCitizen_UE_check_country = document.createElement('div');
classCitizen_UE_check_country.classList.add('citizen_UE_check_countr
y');

var labelCountry = document.createElement('label');
labelCountry.textContent = "Kraj zamieszkania: ";

var countrySelect = document.createElement('select');
countrySelect.name = 'country';
countrySelect.id   = 'country'

const countries = [
```

```
    {value: 'foreign', text: 'Poza UE'},
    {value: 'austria', text: 'Austria'},
    {value: 'belgium', text: 'Belgia'},
    {value: 'bulgaria', text: 'Bułgaria'},
    {value: 'croatia', text: 'Chorwacja'},
    {value: 'republic_of_cyprus', text: 'Cypr'},
    {value: 'czechia', text: 'Czechy'},
    {value: 'denmark', text: 'Dania'},
    {value: 'estonia', text: 'Estonia'},
    {value: 'finland', text: 'Finlandia'},
    {value: 'france', text: 'Francja'},
    {value: 'greece', text: 'Grecja'},
    {value: 'spain', text: 'Hiszpania'},
    {value: 'netherlands', text: 'Holandia'},
    {value: 'ireland', text: 'Irlandia'},
    {value: 'lithuania', text: 'Litwa'},
    {value: 'luxemburg', text: 'Luksemburg'},
    {value: 'latvia', text: 'Łotwa'},
    {value: 'malta', text: 'Malta'},
    {value: 'germany', text: 'Niemcy'},
    {value: 'poland', text: 'Polska'},
    {value: 'portugal', text: 'Portugalia'},
    {value: 'romania', text: 'Rumunia'},
    {value: 'slovakia', text: 'Słowacja'},
    {value: 'slovenia', text: 'Słowenia'},
    {value: 'sweden', text: 'Szwecja'},
    {value: 'hungary', text: 'Węgry'},
    {value: 'italy', text: 'Włochy'}
];

countries.forEach(c => {
  var optionsCountries = document.createElement('option');
  optionsCountries.value = c.value;
  optionsCountries.textContent = c.text;
  countrySelect.appendChild(optionsCountries);
});

var labelCheckbox = document.createElement('label');
labelCheckbox.textContent = "Obywatel Unii Europejskiej";
```

```
var checkbox = document.createElement("input");
checkbox.setAttribute('type',"checkbox");
checkbox.setAttribute('id',"UE_citizen");
checkbox.setAttribute('name',"UE_citizen");

var buttonSubmit = document.createElement("input");
buttonSubmit.setAttribute('type',"submit");
buttonSubmit.setAttribute('value',"Wyślij");

var buttonReset = document.createElement("input");
buttonReset.setAttribute('type',"reset");
buttonReset.setAttribute('value',"Czyść pola");

formField.appendChild(fieldSet);
fieldSet.appendChild(legendTag);
fieldSet.appendChild(labelUser);
fieldSet.appendChild(inputUser);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelPassword);
fieldSet.appendChild(inputPassword);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(labelEmail);
fieldSet.appendChild(inputEmail);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());

fieldSet.appendChild(genderGroup);
genderGroup.appendChild(labelGender);
genderGroup.appendChild(br.cloneNode());

genderOptions.forEach(opt => {
      var radio = document.createElement('input');
      radio.type = 'radio';
      radio.name = 'gender';
      radio.value = opt.value;
      radio.id = opt.id;
      if (opt.required) radio.required = true;
      genderGroup.appendChild(radio);
```

```
      var labelGenderList = document.createElement('label');
      labelGenderList.htmlFor = opt.id;
      labelGenderList.textContent = opt.text;
      genderGroup.appendChild(labelGenderList);
});
genderGroup.appendChild(br.cloneNode());
genderGroup.appendChild(br.cloneNode());

fieldSet.appendChild(labelDofb);
fieldSet.appendChild(inputDofb);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());

fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(labelCountry);
classCitizen_UE_check_country.appendChild(countrySelect);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());

fieldSet.appendChild(classCitizen_UE_check_country);
classCitizen_UE_check_country.appendChild(checkbox);
classCitizen_UE_check_country.appendChild(labelCheckbox);
classCitizen_UE_check_country.appendChild(br.cloneNode());
classCitizen_UE_check_country.appendChild(br.cloneNode());

fieldSet.appendChild(buttonSubmit);
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(br.cloneNode());
fieldSet.appendChild(buttonReset);

document.body.appendChild(formField);

var form = formField;
var submitBtn = buttonSubmit;
var resetBtn = buttonReset;

var fieldsToValidate = [
      inputUser,
      inputPassword,
```

```
      inputEmail,
      inputDofb
];

var genderRadios = form.elements['gender'];

var errorContainers = {};
fieldsToValidate.forEach(function(field) {
      var err = document.createElement('div');
      err.style.color = 'red';
      err.className = 'error-message';
      field.insertAdjacentElement('afterend', err);
      errorContainers[field.name || field.id] = err;
});
var genderError = document.createElement('div');
genderError.style.color = 'red';
genderError.className = 'error-message';
genderGroup.insertAdjacentElement('afterend', genderError);

function updateSubmitButton() {
      submitBtn.style.borderColor = form.checkValidity() ? 'green' :
'red';
}

function validateFieldBorder(field) {
      if (field.checkValidity()) {
            field.style.border = '2px solid green';
            errorContainers[field.name || field.id].textContent = '';
      } else {
            field.style.border = '2px solid red';
            errorContainers[field.name || field.id].textContent =
field.validationMessage;
      }
}

function validateGenderGroup() {
      var isChecked = Array.from(genderRadios).some(r => r.checked);
      genderGroup.style.border = isChecked
            ? '2px solid green'
            : '2px solid red';
```

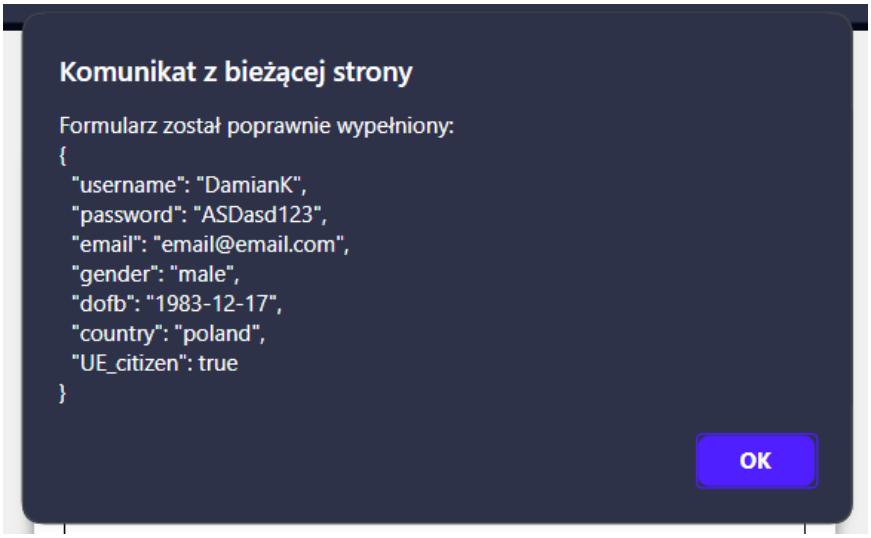
```
      genderError.textContent = isChecked
            ? ''
            : 'Proszę zaznaczyć płeć';
}

function validateAllFields() {
      fieldsToValidate.forEach(validateFieldBorder);
      validateGenderGroup();
      updateSubmitButton();
}

fieldsToValidate.forEach(el => {
      el.addEventListener('input', validateAllFields);
      el.addEventListener('change', validateAllFields);
});
Array.from(genderRadios).forEach(r => {
      r.addEventListener('change', validateAllFields);
});

form.addEventListener('submit', function(e) {
      e.preventDefault();
      if (!form.checkValidity()) {
            form.reportValidity();
            return;
      }

      var birthYear = new Date(inputDofb.value).getFullYear();
      var currentYear = new Date().getFullYear();
      var ageYears = currentYear - birthYear;
      if (ageYears < 18) {
            errorContainers['dofb'].textContent = 'Musisz mieć co
najmniej 18 lat.';
            inputDofb.style.border = '2px solid red';
            return;
      } else {
            errorContainers['dofb'].textContent = '';
            inputDofb.style.border = '2px solid green';
      }

      var formData = {
```

```
            username: inputUser.value,
            password: inputPassword.value,
            email: inputEmail.value,
            gender: Array.from(genderRadios).find(r =>
r.checked).value,
            dofb: inputDofb.value,
            country: countrySelect.value,
            UE_citizen: checkbox.checked
      };
      alert(
            'Formularz został poprawnie wypełniony:\n' +
            JSON.stringify(formData, null, 2)
      );

      var blob = new Blob([JSON.stringify(formData, null, 2)], {type:
'application/json'});
      var url = URL.createObjectURL(blob);
      var link = document.createElement('a');
      link.href = url;
      link.download = 'formData.json';
      document.body.appendChild(link);
      link.click();
      document.body.removeChild(link);
      URL.revokeObjectURL(url);

      form.reset();
      setTimeout(validateAllFields, 0);
});

resetBtn.addEventListener('click', function() {
      setTimeout(validateAllFields, 0);
});

validateAllFields();
</script>
```

Zaprezentowane metody zarówno budowania, jak i dynamicznego generowania oraz weryfikowania danych wprowadzonych prze użytkownika wraz z mechanizmami ich prezentacji i zapisywania lokalnie mogą być wykorzystane do zbudowania

dowolnego formularza WWW. Na rys. 12 zaprezentowano komunikat o poprawnym wypełnieniu formularza.

[Tekst alternatywny. Powiadomienie z bieżącej strony. Treść komunikatu: „Formularz został poprawnie wypełniony:”. W sekcji dane w formacie JSON: username: „DamianK”; password: „ASDasd123”; email: „email@email.com”; gender: „male”; dofb: „1983-12-17”; country: „poland”; UE_citizen: true. Na dole przycisk: OK.]



Rys. 12. Komunikat potwierdzający poprawne wypełnienie i wysłanie formularza.

Warto zaznaczyć, że dane w formacie JSON zostaną opcjonalnie zapisane na terminalu lokalnym pod domyślną nazwą „formData.json”.
Gdy formularz oraz wymogi dotyczące pól formularza zostaną zadeklarowane wyłącznie w HTML5, tak jak w pierwszym przypadku formularza, skrypt walidujący spełniający wszystkie wymogi tego rozdziału, wygląda następująco:

```
<script>
document.addEventListener('DOMContentLoaded', function() {
      var form       = document.querySelector('form');
      var submitBtn  = form.querySelector('input[type="submit"]');
      var resetBtn   = form.querySelector('input[type="reset"]');

      var inputUser     = document.getElementById('username');
      var inputPassword = document.getElementById('password');
      var inputEmail    = document.getElementById('email');
      var inputDofb     = document.getElementById('dofb');
      var countrySelect = document.getElementById('country');
      var checkbox      = document.getElementById('UE_citizen');
      var genderGroup   = document.querySelector('.gender-group');
```

```
	var genderRadios  = form.elements['gender'];

	var fieldsToValidate = [
		inputUser,
		inputPassword,
		inputEmail,
		inputDofb
	];

	var errorContainers = {};
	fieldsToValidate.forEach(function(field) {
		var err = document.createElement('div');
		err.style.color = 'red';
		err.className = 'error-message';
		field.parentNode.insertBefore(err, field.nextSibling);
		errorContainers[field.name || field.id] = err;
	});
	var genderError = document.createElement('div');
	genderError.style.color = 'red';
	genderError.className = 'error-message';
	genderGroup.parentNode.insertBefore(genderError,
genderGroup.nextSibling);

	function updateSubmitButton() {
		submitBtn.style.borderColor = form.checkValidity() ?
'green' : 'red';
	}

	function validateFieldBorder(field) {
		if (field.checkValidity()) {
			field.style.border = '2px solid green';
			errorContainers[field.name || field.id].textContent
= '';
		} else {
			field.style.border = '2px solid red';
			errorContainers[field.name || field.id].textContent
= field.validationMessage;
		}
	}
```

```
    function validateGenderGroup() {
        var isChecked = Array.from(genderRadios).some(r =>
r.checked);
        genderGroup.style.border = isChecked ? '2px solid green' :
'2px solid red';
        genderError.textContent = isChecked ? '' : 'Proszę
zaznaczyć płeć';
    }

    function validateAllFields() {
        fieldsToValidate.forEach(validateFieldBorder);
        validateGenderGroup();
        updateSubmitButton();
    }

    fieldsToValidate.forEach(function(el) {
        el.addEventListener('input',  validateAllFields);
        el.addEventListener('change', validateAllFields);
    });
    Array.from(genderRadios).forEach(r => {
        r.addEventListener('change', validateAllFields);
    });

    form.addEventListener('submit', function(e) {
        e.preventDefault();

        if (!form.checkValidity()) {
            form.reportValidity();
            return;
        }

        var birthYear   = new Date(inputDofb.value).getFullYear();
        var currentYear = new Date().getFullYear();
        if (currentYear - birthYear < 18) {
            errorContainers['dofb'].textContent = 'Musisz mieć
co najmniej 18 lat.';
            inputDofb.style.border = '2px solid red';
            return;
        } else {
            errorContainers['dofb'].textContent = '';
```

```
                inputDofb.style.border = '2px solid green';
            }

            var formData = {
                username: inputUser.value,
                password: inputPassword.value,
                email: inputEmail.value,
                gender: Array.from(genderRadios).find(r =>
r.checked).value,
                dofb: inputDofb.value,
                country: countrySelect.value,
                UE_citizen: checkbox.checked
            };

            alert(
                'Formularz został poprawnie wypełniony:\n' +
                JSON.stringify(formData, null, 2)
            );

            var blob = new Blob([JSON.stringify(formData, null, 2)],
{type: 'application/json'});
            var url  = URL.createObjectURL(blob);
            var link = document.createElement('a');
            link.href = url;
            link.download = 'formData.json';
            document.body.appendChild(link);
            link.click();
            document.body.removeChild(link);
            URL.revokeObjectURL(url);

            form.reset();
            setTimeout(validateAllFields, 0);
        });

        resetBtn.addEventListener('click', function() {
            setTimeout(validateAllFields, 0);
        });

        validateAllFields();
});
```

```
</script>
```

Zmiany dotyczą przede wszystkim metod pobierania identyfikatorów i wartości pól formularza. Poza tym mechanizmy walidacji, wyświetlania i zapisywania danych są identyczne.

5. Formularze i skrypty po stronie serwera z wykorzystaniem PHP

5.1. Współczesne wykorzystanie języka PHP

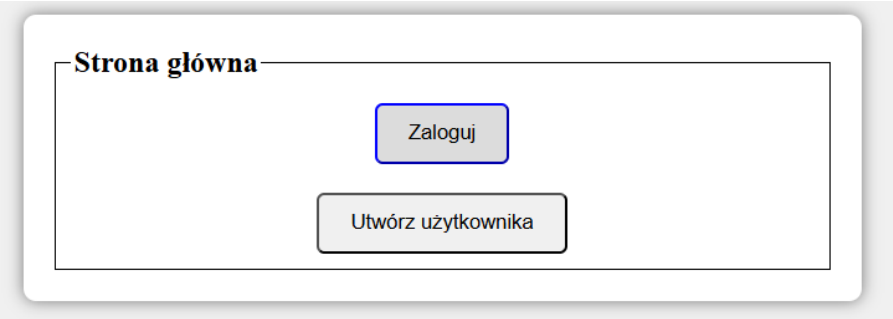
PHP to popularny język skryptowy wykonywany po stronie serwera, który bezpośrednio integruje się z kodem HTML. Umożliwia generowanie dynamicznych stron internetowych w locie, obsługę formularzy czy zarządzanie sesjami użytkowników. Dziś PHP napędza ogromną część światowego internetu, od prostych stron wizytówek po rozbudowane systemy e-commerce. Najczęściej wykorzystywane jest w tworzeniu serwisów opartych o gotowe systemy zarządzania treścią, takich jak WordPress czy Drupal.
Współczesne projekty korzystają z nowoczesnych frameworków PHP, na przykład Laravel, Symfony czy Zend Framework. Dzięki temu możliwe jest szybkie prototypowanie, ścisłe przestrzeganie wzorców projektowych i łatwa rozbudowa aplikacji. PHP umożliwia również tworzenie REST API, które komunikują się z front-endem napisanym w JavaScripcie, na przykład z wykorzystaniem Vue czy React. Dużą zaletą jest szeroka biblioteka gotowych pakietów dostępnych przez Composer oraz integracja z bazami danych MySQL, PostgreSQL czy SQLite.
Od wersji 7.0 wprowadzono znaczące przyspieszenia działania dzięki optymalizacji silnika Zend i wsparciu dla OPcache. W PHP 8.x pojawiło się kompilowanie JIT, ulepszone typowanie i nowoczesna składnia, co zbliża go do wzorców znanych z innych języków. Skalowalność zapewniają narzędzia takie jak Docker, Kubernetes czy platformy chmurowe, gdzie PHP można łatwo konteneryzować i automatyzować wdrażanie. Dzięki stabilności i kompatybilności wstecznej wiele starszych aplikacji wciąż działa na nowych wersjach silnika.
W przeciwieństwie do Node.js, które bazuje na asynchronicznym modelu zdarzeniowym, PHP tradycyjnie wykonuje kod synchronicznie, co upraszcza programowanie, lecz wymaga odpowiedniej konfiguracji serwera przy dużym ruchu. W porównaniu z Pythonem i Rubym, PHP oferuje zwykle łatwiejsze wdrożenie na standardowych serwerach WWW, choć niektóre rozwiązania wymagają większej liczby pakietów i środowisk uruchomieniowych. Mimo że inne języki często oferują nowocześniejsze biblioteki i ekosystemy, PHP wyróżnia się ogromną społecznością i

wsparciem hostingowym, co czyni go wciąż atrakcyjnym wyborem dla wielu projektów.

5.2. Hasła i logowanie, dane w plikach tekstowych

W poprzednich rozdziałach zdubowana została dwoma sposobami strona internetowa dość rozbudowanego formularza. Wykorzystany został zarówno język HTML5 wraz z jego metodami walidacyjnymi, jak i język JavaScript, który ponadto został wykorzystany do rozszerzonej walidacji danych pworwadzanych przez użytkownika. Po aktywacji (kliknięciu) przycisku „Wyślij" dane nie są jednak przesyłane na serwer, wszystkie operacje, tj. wyświetlenie informacji oraz zapis danych do pliku, odbywają się lokalnie. Rozbudujmy teraz serwis internetowy o kolejne podstrony. Stwórzmy stronę główną, która zawierać będzie wyłącznie dwa przyciski: „Zaloguj" oraz „Utwórz użytkownika". Drugi przycisk będzie prowadził na jedną ze stron formularza przygotowanych uprzednio. Pierwszy z kolei będzie prowadził na stronę logowania, którą będziemy sukcesywnie rozbudowywać z wykorzystaniem języka PHP. Stron głowna zaprezentowana jest na rys. 13.

[Tekst alternatywny. Prosta strona internetowa w języku polskim. Nagłówek: Strona główna. Dwie przyciski: Zaloguj; Utwórz użytkownika.]



Rys. 13. Strona główna służąca do uruchamiania zarówno strony z formularzem rejestracyjnym, jak i strony do logowania.

W stronie głównej zdecydowano się na wprowadzenie dodatkowego stylu dla akcji najechania kursorem na przycisk. Kod tej strony prezentuje się następująco:

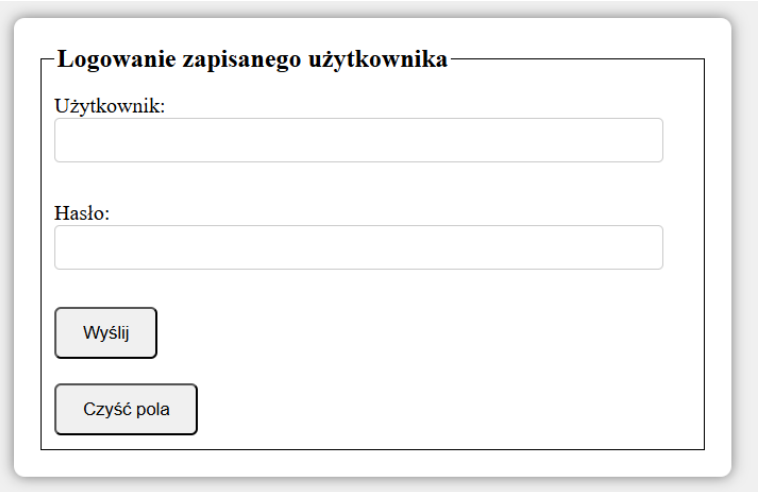
```
<html>
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-
scale=1.0">
     <title>Strona z formularzem HTML</title>
     <style>
```

```
            body {
                  display: flex;
                  justify-content: center;
                  align-items: center;
                  height: 100vh;
                  margin: 0;
                  background-color: #f0f0f0;
            }
            form {
                  width: 500px;
                  background-color: #fff;
                  padding: 20px;
                  border-radius: 8px;
                  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            }
            fieldset {
                  border: 1px solid black;
                  padding: 10px;
                  margin: 0;
            }
            legend {
                  font-weight: bold;
                  font-size: 1.2em;
                  margin-bottom: 5px;
            }
            input[type="button"]       {
                  padding: 10px 20px;
                  border-radius: 5px;
                  cursor: pointer;
            }
            input:hover, input:active {
                  border-color: blue;
                  background-color: gainsboro;
            }
      </style>
</head>

<body>
      <form>
      <fieldset>
```

```
            <legend>Strona główna</legend>
            <center>
            <a href="formularz_php.html" target="_self"     >
                  <input type="button" value="Zaloguj"><br><br>
            </a>
            <a href="formularz_html_css_js_walidacja_php.html"
target="_self">
                  <input type="button" value="Utwórz użytkownika">
            </a>
            </center>
      </fieldset>
      </form>
</body>
</html>
```

Część stylu CSS została pozostawiona bez zmian względem strony logowania celem zachowania spójności w obrębie serwisu. To samo będzie dotyczyć każdej innej strony, która zostanie przygotowania na potrzeby obsługi z wykorzystaniem PHP. Rozpocznijmy od przygotowania najprostrzej strony logowania z dwoma polami do wprowadzania nazwy użytkownika i hasła oraz dwoma przyciskami, z czego jeden do logowania, a drugi do czyszczenia pól. Przekopiujemy wymagania co do długości i formatowania zarówno nazwy użytkownika, jak i hasła. Strona ta ma wygląd zaprezentowany na rys. 14.

[Tekst alternatywny. Formularz HTML w języku polskim zatytułowany „Logowanie zapisanego użytkownika”. Dwie etykiety i pola: Użytkownik: pole tekstowe; Hasło: pole tekstowe. Na dole formularza dwa przyciski: Wyślij; Czyść pola.]



Rys. 14. Strona do logowania użytkownika.

Kod strony wykorzystauje dokładnie te same style formatowania CSS co w poprzednich stronach, dlatego też zaprezentowany zostanie wyłącznie blok „BODY”:

```
<body>
	<form method="post" action="user_check.php">
	<fieldset>
	<legend>Logowanie zapisanego użytkownika</legend>
		<label for="username">Użytkownik:</label><br>
		<input type="text" id="username" name="username"
minlength="6" maxlength="12" size="12" required><br><br>
		<label for="password">Hasło:</label><br>
		<input type="password" id="password" name="password"
pattern="(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Hasło musi
zawierać co najmniej jedną cyfrę oraz małą i dużą literę, minimalna
długość to 8 znaków." required><br><br>
		<input type="submit" value="Wyślij">
		<br><br><input type="reset" value="Czyść pola">
	</fieldset>
	</form>
</body>
```

Po kliknięciu przycisku „Wyślij” nastąpi wywołanie znajdującego się na serwerze skryptu „user_check.php”. Jego kod wygląda następująco:

```
<?php
$username = $_POST["username"];
$password = $_POST["password"];
$page_granted = file_get_contents('access_granted.html');
$page_denied = file_get_contents('access_denied.html');
if ($username <> "numerJeden" OR $password <> "hasloNumer1") {
	echo $page_denied;
} else {
	echo $page_granted;
}
?>
```

Dane do skryptu, czyli wprowadzone przez użytkownika login i hasło, zostaną przekazane metodą „POST”. W samym skrypcie odczytujemy dane zawarte w tablicy $_POST pod indeksem „username” oraz „password” i zapisujemy w zmiennych o tych samych nazwach. Następnie porównujemy wartości tych zmiennych z

wartościami zapisanymi w skrypcie. W tym miejscu możemy zaprogramować dodatkowe metody walidacji danych, jeśli nie zostały one zdefiniowane w pliku formularza. Przykładowo, aby sprawdzić, czy pola nazwy użytkownika i hasła zostały wypełnione wystarczy rozbudować powyższy kod w następujący sposób:

```
if ($username == "" OR $password == "") {
     echo $page_denied;
} else {
     if ($username <> "numerJeden" OR $password <> "hasloNumer1") {
          echo $page_denied;
     } else {
          echo $page_granted;
     }
}
```

Prostszym sposobem byłoby zdefiniowanie zmiennej logicznej, która przyjmowałaby wartości „true” i „false” w zależności od spełnienia warunków walidacyjnych narzuconych na dane użytkownika. Zmienna ta mogłaby mieć domyślnie wartość „true” i każde niespełnienie warunku w serii warunków zmieniłoby jej wartość na „false”. Ponadto od drugiego walidatora możnaby srawdzać także wartość tej zmiennej i rezygnować z walidacji dla wartości negatywnej celem oszczędzenia czasu pracy terminala serwerowego i szybszego przejścia do strony informującej o błędzie.
Kontynuując wątek zaprezentowanego kodu PHP jeżeli występuje zgodność, wczytujemy plik „access_granted.html”, w przeciwnym wypadku plik „access_denied.html”. Taka metoda zabezpieczenia dostępu jest bardzo prosta i nie zapewnia dużej poufności, choćby dlatego, że dane logowania przesyłane przez sieć nie są szyfrowane. Wymagane jest zatem szyfrowanie całego połączenia i dodatkowo samych danych. Ponadto możliwe jest w tej sytuacji uruchomienie strony „access_granted.html” z pominięciem ekranu logowania o ile zna się jej ścieżkę dostępu. Temu ostatniemu przypadkowi można zapobiec porzez wygenerowanie właściwej strony internetowej poprzez skrypt PHP. W takim przypadku strona internetowa będzie zapisana jako zmienna w postaci ciągu znaków tekstowych z wykorzystaniem operatora ciągu hermetycznego „<<<”. Pełny kod zmodyfikowanego pliku „user_check.php” prezentuje się następująco:

```
<?php
$username = $_POST["username"];
$password = $_POST["password"];
```

```
$page_granted = <<<CODE
<html>
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-
scale=1.0">
	<title>Udzielono dostepu</title>
	<style>
		body {
			display: flex;
			justify-content: center;
			align-items: center;
			height: 100vh;
			margin: 0;
			background-color: #f0f0f0;
		}
		form {
			width: 500px;
			background-color: #fff;
			padding: 20px;
			border-radius: 8px;
			box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
		}
		fieldset {
			border: 1px solid black;
			padding: 10px;
			margin: 0;
		}
		legend {
			font-weight: bold;
			font-size: 1.2em;
			margin-bottom: 5px;
		}
		label {
			margin-bottom: 5px;
		}
		input[type="button"] {
			padding: 10px 20px;
			border-radius: 5px;
			cursor: pointer;
```

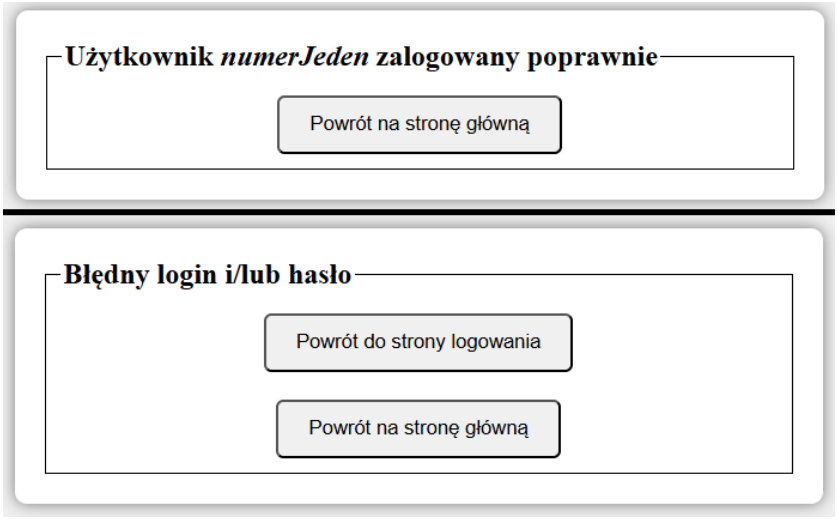
```
            }
            input:hover, input:active {
                  border-color: blue;
                  background-color: gainsboro;
            }
      </style>
</head>

<body>
      <form method="post" action="">
      <fieldset>
            <legend>Użytkownik <i>$username</i> zalogowany
poprawnie</legend>
            <center>
            <a href="index.html" target:_self>
                  <input type="button" value="Powrót na stronę
główną">
            </a>
            <center>
      </fieldset>
      </form>
</body>
</html>
CODE;

$page_denied = <<<CODE
<html>
<head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <title>Odmowiono dostepu</title>
      <style>
            body {
                  display: flex;
                  justify-content: center;
                  align-items: center;
                  height: 100vh;
                  margin: 0;
                  background-color: #f0f0f0;
```

```
            }
            form {
                  width: 500px;
                  background-color: #fff;
                  padding: 20px;
                  border-radius: 8px;
                  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            }
            fieldset {
                  border: 1px solid black;
                  padding: 10px;
                  margin: 0;
            }
            legend {
                  font-weight: bold;
                  font-size: 1.2em;
                  margin-bottom: 5px;
            }
            label {
                  margin-bottom: 5px;
            }
            input[type="button"] {
                  padding: 10px 20px;
                  border-radius: 5px;
                  cursor: pointer;
            }
            input:hover, input:active {
                  border-color: blue;
                  background-color: gainsboro;
            }
      </style>
</head>

<body>
      <form method="post" action="">
      <fieldset>
            <legend>Błędny login i/lub hasło</legend>
            <center>
            <a href="formularz_php.html" target:_self>
```

```
                <input type="button" value="Powrót do strony
logowania">
           </a>
           <br><br>
           <a href="index.html" target:_self>
                <input type="button" value="Powrót na stronę
główną">
           </a>
           <center>
      </fieldset>
      </form>
</body>
</html>
CODE;

if ($username <> "numerJeden" OR $password <> "hasloNumer1") {
      print("$page_denied");
} else {
      print("$page_granted");
}
?>
```

Zwróćmy uwagę, że w przypadku poprawnego logowania na stronie wyświetla się nazwa użytkownika pobrana jako zmienna metodą POST z formatowaniem „italic”. Strony internetowe z wykorzystaniem PHP można także generować linia po linii lub blok po bloku z wykorzystaniem polecenia „echo” (np.: echo(„<html>body> <h2>Strona WWW</h2> </body></html>”);). W przypadku prostych komunikatów ten sposób może okazać się prostszy dla programisty. W naszym przypadku bardziej słożone strony zdecydowano się na operator ciągu hermetycznego, ponieważ można wydzielić treść właściwej strony i modyfikować go w oderwaniu od skryptu. Obydwie wynikowe strony internetowe w zależności od wyniku porównania zaprezentowano na rys. 15.

[Tekst alternatywny. Komunikaty o rezultacie logowania. Dwa odrębne obramowane pola: W pierwszym polu tekst „Użytkownik numerJeden zalogowany poprawnie” oraz link „Powrót na stronę główną”; w drugim polu tekst „Błędny login i/lub hasło” oraz linki „Powrót do strony logowania” i „Powrót na stronę główną”.]



Rys. 15. Strona potwierdzająca dostęp (u góry) oraz strona informująca o błędnych danych logowania (na dole).

Powyższy skrypt umożliwiał wprowadzenie loginu i hasła dostępu do strony. Jednak był to jeden login i jedno hasło dla wszystkich użytkowników. Takie rozwiązanie w wielu sytuacjach może być niewystarczające. Często lepiej będzie wprowadzić dane oddzielnie dla każdej uprawnionej osoby. Skrypt będzie przez to bardziej skomplikowany, ale też bardziej funkcjonalny. Nazwy użytkowników oraz hasła przechowywane będą w zwykłym pliku tekstowym o nazwie „passwords.txt”. Każda linia tego pliku będzie zawierała dane w formacie: „nazwa_użytkownika:hasło”. Plik ten wypełnijmy przykładowymi danymi następująco:

```
numerJeden:hasloNumer1
numerDwa:hasloNumer2
numerTrzy:hasloNumer3
numerCztery:hasloNumer4
numerPiec:hasloNumer5
numerSzesc:hasloNumer6
numerSiedem:hasloNumer7
numerOsiem:hasloNumer8
numerDziewie:hasloNumer9
numerDziesie:hasloNumer10
```

Odpowiednia funkcja w skrypcie będzie dokonywała analizy i porównywania każdego wpisu z danymi wprowadonymi w formularzu logowania. Sam formularz logowania pozostanie taki sam, zmieni się jedynie kod PHP. Zmienione zostanie także wyrażenie warunkujące wyświetlenie właściwej strony po logowaniu:

```
function checkPass($pass, $user) {
	if (!$fd = @fopen("passwords.txt", "r")) return false;
	while (!feof ($fd)) {
		$line = trim(fgets($fd));
		if (($pos = strpos($line, ":")) === false) continue;

		$tempUser = substr($line, 0, $pos);
		if ($tempUser != $user) continue;

		$tempPass = substr($line, $pos + 1, strlen($line) + $pos);

		if ($tempPass != $pass) continue;
		else return true;
	}
	fclose($fd);
	return false;
}

if (!checkPass($password, $username)) {
	print("$page_denied");
} else {
	print("$page_granted");
}
```

Loginy i hasła zapisane są w pliku teksowym jawnie, oznacza to, że w przypadku udanego włamania na serwer dane te są narażone na kradzież i wykorzystanie poza kontrolą użytkowników i administratorów. Można temu zapobiec poprzez szyfrowanie lub haszowanie danych użytkownika. Zwiększa to poziom bezpieczeństwa ale uniemożlia ręczną modyfikację pliku tekstowego. Przykładowo, wykorzystajmy funkcję haszowania MD5 i istniejący formularz logowania. Wyłącznie dla celów przykładowych zmieńmy skrypt PHP na dopisujący użytkowników do pliku „passwords.txt” zamiast odczytywać dane. W takiej sytuacji usuwamy fukcję sprawdzającą dane i zastępujemy ją następującym kodem:

```
if (!$fd = @fopen("passwords.txt", "a")) {
```

```
    print("$page_denied");
    return;
}

$str = "\r\n".$username.":".md5($password);
fwrite($fd, $str);
fclose($fd);
print("$page_granted");
```

Wykorzystaliśmy te same strony informacyjne wyłącznie do celów realizacji metody haszowania. Wprowadźmy następujące dane: nazwa użytkownika: numerJedenas, hasło: hasloNumer11. Do pliku „passwords.txt" zostanie dopisany wiersz:

```
numerJedenas:037df263ef2b2671ff87222f08bad2f6
```

Login nie jest haszowany, z kolei hasło już tak. Weryfikacja odbywałaby się poprzez ponowne haszowanie danych wprowadzonych przez użytkownika i porównanie z wpisem. Kradzież pliku z hasłami nie umożliwi z kolei dostępu do zasobów chwonionych, ponieważ nie istnieje funkcja dehaszująca. Spróbujmy wykorzystać ten skrypt do dodania użytkownika z poziomu strony z formularzem, w naszym przypadku jest to strona „formularz_html_css_js_walidacja_php.html", w której uzupełnimy wiersz akcji formularza podając nazwę pliku PHP:

```
formField.setAttribute('action','user_add_file.php');
```

oraz dodamy wywołanie:

```
form.submit();
```

tuż po wyświetleniu komunikatu o poprawnym wypełnieniu formularza. Są to jedyne zmiany, zatem nadal możliwe jest zapisanie danych użytkownika w formacie JSON na dysku lokalnym terminala użytkownika. Teraz po kliknięciu przycisku „Wyślij" dodany zostanie nowy użytkownik o niehaszowanej nazwie i haszowanym haśle. Możliwe jest jednak dopisywanie użytkowników o tych samych nazwach, zatem niezbędny jest mechanizm sprawdzający istnienie użytkownika o zdublowanej nazwie, uniemożliwienie jego zapisanie i wyświetlenie stosowanej informacji użytkownikowi. Zmiany powinny objąć także generowane z poziomu PHP strony internetowe, a w zasadzie przekierowania do poszczególnych podstron oraz komunikaty informacyjne. Fragment skryptu umożliwiający powyższe operacje prezentuje się następująco:

```
function checkName($user) {
	if (!$fd = @fopen("passwords.txt", "r")) return false;
	while (!feof ($fd)) {
		$line = trim(fgets($fd));
		if (($pos = strpos($line, ":")) === false) continue;

		$tempUser = substr($line, 0, $pos);
		if ($tempUser != $user) continue;

		else return true;
	}
	fclose($fd);
	return false;
}

if (checkName($username)) {
	print("$user_denied");
} else {
	if (!$fd = @fopen("passwords.txt", "a")) {
		print("$user_denied");
		return;
	}

	$str = $username.":".md5($password)."\r\n";
	fwrite($fd, $str);
	fclose($fd);
	print("$user_granted");
}
```

Ponieważ hasła od tej pory będą haszowane, wymagane także będą zmiany w skrypcie sprawdzającym hasła z poziomu strony logowania. Zmiany polegają na wywołaniu funkcji „md5()” dla zmiennej „`$pass`” w wierszu „`if ($tempPass != md5($pass)) continue;`” wewnątrz funkcji „checkPass”. Poprzez formularz rejestracyjny wypełniono plik „password.txt” przykładowymi danymi:

```
userNumer1:numerJeden1
userNumer2:numerJeden2
userNumer3:numerJeden3
```

W efekcie uzyskano następujące wpisy:

```
userNumer1:879849b0638c656d94706656e1f9926e
userNumer2:31af6994dcfb68ea0e4e65929d3ab6ff
userNumer3:e275f4b099b3c4c9fd2e21cd9a0d2c38
```

Tak więc hasła użytkowników w pliku tekstowym są już w pewien sposób chronione, nadal pozostaje kwestia szyfrowania danych w medium transmisyjnym, która może być zrealizowana poprzez wybór odpowiedniego protokołu transmisyjnego (SSL) lub szyfrowanie danych poprzez skrypt na stronie z formularzem (dla danych tuż przed wysłaniem poleceniem „submit()") i deszyfracja po stronie serwera.
Pozostaje jeszcze kwestia zapisu na serwerze danych dodatkowych podawanych przez użytkownika. Aktualnie dane z formularza zapisywane są lokalnie do pliku JSON. Rozbudujmy skrypty JavaScript i PHP w taki sposób, aby wszystkie podawane przez użytkownika dane zapisywane były w tym formacie w oddzielnym pliku tekstowym na serwerze. Hasło nadal będzie zapisywane oddzielnie i będzie podlegać haszowaniu. Łącznikiem pomiędzy hasłem i danymi użytkownika będzie podana przez niego unikalna nazwa. Z kolei po poprawnym zalogowaniu użytkownik powinien zobaczyć swoje dane. Rozbudowujemy następująco dokument HTML z formularzem umieszczając dodatkowy kod tuż pod wierszami z komunikatem o poprawnym wypełnieniu formularza. Na początku zbieramy wszystkie dane, tworzymy obiekt bez zmiennej „password" i serializujemy go do JSON:

```
var formData = {
      username: inputUser.value,
      password: inputPassword.value,
      email: inputEmail.value,
      gender: Array.from(genderRadios).find(r => r.checked).value,
      dofb: inputDofb.value,
      country: countrySelect.value,
      UE_citizen: checkbox.checked
};

var {password, ...jsonPayload} = formData;
var jsonBody = JSON.stringify(jsonPayload);
```

Teraz przygotowujemy ukryte pole, które będzie wysyłne wraz z innymi polami formularza:

```
let hidden = form.querySelector('input[name="jsonData"]');
```

```
if (!hidden) {
      hidden = document.createElement('input');
      hidden.type = 'hidden';
      hidden.name = 'jsonData';
      form.appendChild(hidden);
}
hidden.value = jsonBody;
```

Zmienna „jsonData” zawiera interesujące nas dane. W pliku „user_add_file.php” dodajemy wiersz „`if (!isset($_POST['jsonData'])) return false;`” na początku funkcji „checkName”. Ma to na celu sprawdzenie poprawności odebrania danych w formacie JSON. Dodatkowo sprawdzamy istnienie pliku z danymi, nazwijmy go „user_data.json”. Jeśli weryfikacje wszystkich danych wejściowych będą pozytywne zapisujemy dane JSON do zmiennej i dopisujemy wiersz do pliku z danymi. Pełny kod dwóch odpowiedzialnych za wymienione zadania funkcji jest zaprezentowany następująco:

```
function checkName($user) {
      if (!$fd = @fopen("passwords.txt", "r")) return false;
      if (!$fdj = @fopen("user_data.json", "r")) return false;
      if (!isset($_POST['jsonData'])) return false;
      while (!feof ($fd)) {
            $line = trim(fgets($fd));
            if (($pos = strpos($line, ":")) === false) continue;

            $tempUser = substr($line, 0, $pos);
            if ($tempUser != $user) continue;

            else return true;
      }
      fclose($fd);
      fclose($fdj);
      return false;
}

if (checkName($username)) {
      print("$user_denied");
} else {
      if (!$fd = @fopen("passwords.txt", "a") OR !$fdj =
@fopen("user_data.json", "a")) {
```

```
            print("$user_denied");
            return;
        }

        $userString = $_POST['jsonData'];
        $str = $username.":".md5($password)."\r\n";
        $str_json = $userString."\r\n";
        fwrite($fd, $str);
        fwrite($fdj, $str_json);
        fclose($fd);
        fclose($fdj);
        print("$user_granted");
}
```

Utwórzmy troje użytkowników, plik z hasłami wygląda następująco:

```
Jan_Kowalski:ff75bac1953aafd787c45e779d8d6fb3
Anna_Nowak:c00b53dbd8f5cd151d180a2a839bdfbd
Henry_Smith:b358b55b869fdd8e346c7a5f9eb360ed
```

Plik z danymi użytkowników prezentuje się z kolei następująco:

```
{"username":"Jan_Kowalski","email":"jan@mail","gender":"male","dofb"
:"2000-10-28","country":"poland","UE_citizen":true}
{"username":"Anna_Nowak","email":"an@mail.es","gender":"female","dof
b":"2002-06-15","country":"spain","UE_citizen":true}
{"username":"Henry_Smith","email":"hesmi@us.edu","gender":"refusal",
"dofb":"2001-01-19","country":"luxemburg","UE_citizen":false}
```

Tak jak nadmieniono jedynym łącznikiem pomiędzy hasłem użytkownika, a jego danymi jest nazwa użytkownika. Pozwoli to na przechowywanie danych w dowolnej innej chronionej strukturze (np. szyfrowanej bazie danych) ponieważ w obecnej sytuacji dane nie są chronione na wypadek włamiania się na serwer i ręcznej manipulacji plików.
Zajmijmy się teraz wyświetlaniem danych po poprawnym zalogowaniu się użytkownika. Po poprawnej weryfikacji loginu i hasła odczytujemy plik z danymi JSON i szukamy właściwej nazwy użytkownika. Następnie odczytujemy pozostałe dane i wyświetlamy je w postaci listy na stronie internetowej „access_granted”. Wykorzystajmy mapowanie placeholderów po to, aby zminimalizować liczbę zmian w pliku. Możliwe byłoby także strony internetowej z zadeklarowanymi zmiennymi

poniżej deklaracji i nadania wartości tym zmiennym. Sekcja „body” dla strony wyświetlającej wynik zostanie zmodyfikowana następująco:

```
<body>
	<form method="post" action="">
	<fieldset>
		<legend>Użytkownik <i>$username</i> zalogowany
poprawnie</legend>
		<ul>
		<li>E-mail: {{email}}</li>
		<li>Płeć: {{gender}}</li>
		<li>Data urodzenia: {{dofb}}</li>
		<li>Kraj: {{country}}</li>
		<li>Obywatel UE: {{ueCitizen}}</li>
		</ul>
		<center>
		<br>
		<a href="index.html" target:_self>
			<input type="button" value="Powrót na stronę
główną">
		</a>
		<center>
	</fieldset>
	</form>
</body>
```

Z kolei zamiast wiersza `print("$page_granted");` w wyrażeniu sprawdzającym zmienną „checkPass” wprowadzamy kod opisany poniżej. Na początku otwieramy plik JSON, pojedyncze obiekty w kolejnych wierszach:

```
$handle = fopen("user_data.json", "r");
```

Teraz szukamy wiersza z pasującym „username”, przypisujemy znalezione dane do zmiennych, pomijamy także błędne linie:

```
$email = $gender = $dofb = $country = '';
$ueCitizen = 'no';
while (($line = trim(fgets($handle))) !== false) {
	if ($line === '') continue;
	$entry = json_decode($line, true);
```

```
    if (json_last_error() !== JSON_ERROR_NONE) {
        continue
    }
    if (isset($entry['username']) && $entry['username'] ===
$username) {
        $email = $entry['email'] ?? '';
        $gender = $entry['gender'] ?? '';
        $dofb = $entry['dofb'] ?? '';
        $country = $entry['country'] ?? '';
        $ueCitizen = !empty($entry['UE_citizen']) ? 'yes' : 'no';
        break;
    }
}
fclose($handle);
```

Teraz przygotowujemy tablicę zamienników:

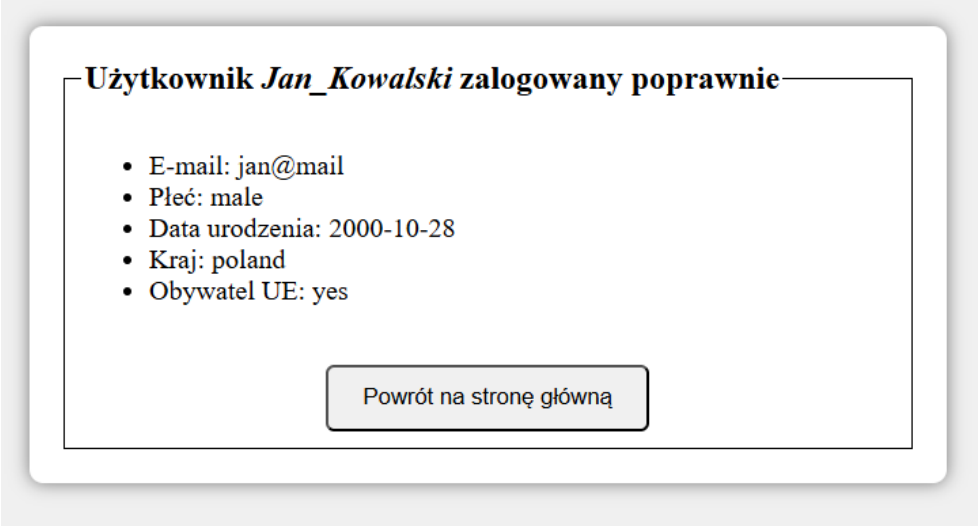
```
$replacements = [
    '{{email}}' => htmlspecialchars($email, ENT_QUOTES, 'UTF-8'),
    '{{gender}}' => htmlspecialchars($gender, ENT_QUOTES, 'UTF-8'),
    '{{dofb}}' => htmlspecialchars($dofb, ENT_QUOTES, 'UTF-8'),
    '{{country}}' => htmlspecialchars($country, ENT_QUOTES, 'UTF-
8'),
    '{{ueCitizen}}' => htmlspecialchars($ueCitizen, ENT_QUOTES,
'UTF-8'),
];
```

Zamiast print – podmieniamy znaczki i wyświetlamy:

```
echo str_replace(
    array_keys($replacements),
    array_values($replacements),
    $page_granted
);
```

Po wpisaniu przykładowych poprawnych danych uzyskamy widok, jak na rys. 16.

[Tekst alternatywny. Strona z potwierdzeniem poprawnego logowania w języku polskim. Nagłówek: Użytkownik Jan_Kowalski zalogowany poprawnie. Wypunktowana lista danych użytkownika: E-mail: jan@mail; Płeć: male; Data urodzenia: 2000-10-28; Kraj: poland; Obywatel UE: yes. Na dole przycisk/link: Powrót na stronę główną]



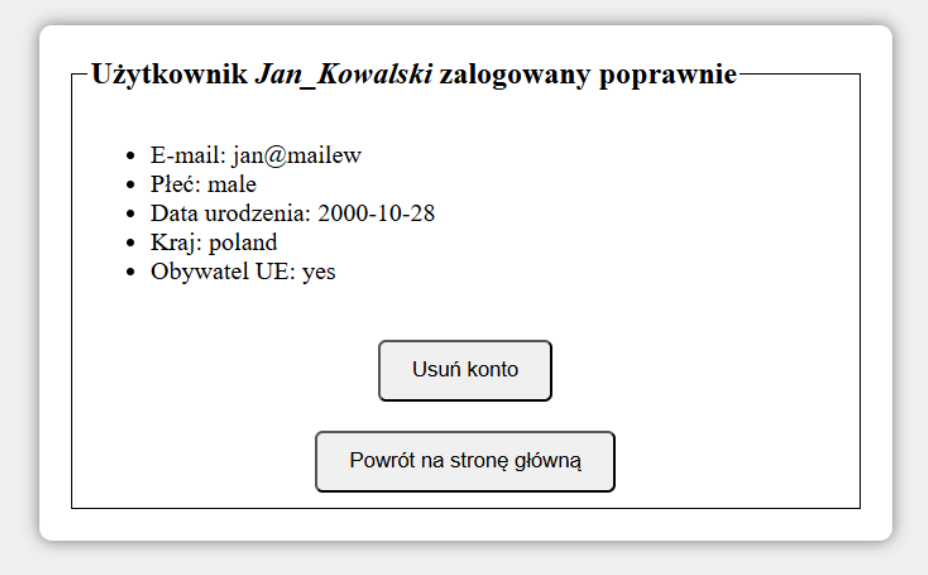
Rys. 16. Strona z danymi dla poprawnie zalogowanego użytkownika.

Zmodyfikujmy teraz tą stronę umożliwiając użytkownikowi na usunięcie konta. Dodajmy przycisk ze zdefiniowaną akcją usuwania w następujący sposób:

```
<input type="hidden" name="username" value="{$username}">
<input type="hidden" name="password" value="{$password}">

<button name="action" value="delete"
     onclick="return confirm('Na pewno usunąć konto?')">
Usuń konto
</button>
```

Strona informacyjna będzie zatem wyglądała jak na rys. 17.

[Tekst alternatywny: Strona profilu użytkownika w języku polskim. Górą nagłówek: „Użytkownik Jan_Kowalski zalogowany poprawnie.” Poniżej wypunktowana lista danych: E-mail: jan@mailew; Płeć: male; Data urodzenia: 2000-10-28; Kraj: poland; Obywatel UE: yes. Na dole dwa przyciski: Usuń konto; Powrót na stronę główną.]



Rys. 17. Strona informacyjna dla poprawnie zalogowanego użytkownika wraz z przyciskiem usuwającym danego użytkownika.

Na samej górze pliku PHP zadeklarujmy teraz dwie funkcje odpowiedzialne za usuwanie danych i zmieniamy metody pobierania danych tak, aby działały redirecty:

```
$username = $_REQUEST['username'] ?? '';
$password = $_REQUEST['password'] ?? '';

function deleteUser(string $user) {
     $lines = file("passwords.txt", FILE_IGNORE_NEW_LINES);
     $filtered = array_filter($lines, function($l) use($user){
          return strpos($l, "$user:") !== 0;
     });
     file_put_contents("passwords.txt", implode("\n", $filtered));

     $in  = fopen("user_data.json", "r");
     $out = fopen("user_data.tmp",  "w");
     while (($line = fgets($in)) !== false) {
          $entry = json_decode(trim($line), true);
          if (isset($entry['username']) && $entry['username'] ===
$user) {
               continue
```

```
            }
            fwrite($out, $line);
        }
        fclose($in);
        fclose($out);
        rename("user_data.tmp", "user_data.json");
}
```

Poniżej definicji stron HTML umieszczamy warunek usuwania konta oraz akcję:

```
if (isset($_POST['action']) && $_POST['action'] === 'delete') {
        if (!checkPass($password, $username)) {
            print("$page_denied");
            exit;
        }
        deleteUser($username);
        header('Location: index.html');
        exit;
}
```

Następuje tu ponowna weryfikacja uprawnień, po jej pozytywnym rozpatrzeniu i usunięciu konta użytkownik zostanie przekierowany na stronę główną. Użytkownik nie dostaje w tej chwili informacji o usunięciu konta. Wprowadźmy dodatkową zmianę w postaci wyświetlania okna potwierdzającego tą akcję zanim nastapi przekierowanie na stronę głowną. Zamieniamy linię `header('Location: index.html');` na:

```
echo "<script>
        alert('Konto $username zostało usunięte.');
        window.location.href = 'index.html';
</script>";
```

Wpisy dla danego użytkownika zarówno w pliku „passwords.txt” jak i „user_data.json” zostaną wyczyszczone. Pełny kod tak zmodyfikowanego pliku PHP wygląda następująco:

```
<?php
$username = $_REQUEST['username'] ?? '';
$password = $_REQUEST['password'] ?? '';

function deleteUser(string $user) {
```

```
      $lines = file("passwords.txt", FILE_IGNORE_NEW_LINES);
      $filtered = array_filter($lines, function($l) use($user){
            return strpos($l, "$user:") !== 0;
      });
      file_put_contents("passwords.txt", implode("\n", $filtered));

      $in  = fopen("user_data.json", "r");
      $out = fopen("user_data.tmp",  "w");
      while (($line = fgets($in)) !== false) {
            $entry = json_decode(trim($line), true);
            if (isset($entry['username']) && $entry['username'] ===
$user) {
                  continue;
            }
            fwrite($out, $line);
      }
      fclose($in);
      fclose($out);
      rename("user_data.tmp", "user_data.json");
}

$page_granted = <<<CODE
<html>
<head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <title>Udzielono dostepu</title>
      <style>
            body {
                  display: flex;
                  justify-content: center;
                  align-items: center;
                  height: 100vh;
                  margin: 0;
                  background-color: #f0f0f0;
            }
            form {
                  width: 500px;
                  background-color: #fff;
```

```
                padding: 20px;
                border-radius: 8px;
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            }
            fieldset {
                border: 1px solid black;
                padding: 10px;
                margin: 0;
            }
            legend {
                font-weight: bold;
                font-size: 1.2em;
                margin-bottom: 5px;
            }
            label {
                margin-bottom: 5px;
            }
            input[type="button"] {
                padding: 10px 20px;
                border-radius: 5px;
                cursor: pointer;
            }
            button {
                padding: 10px 20px;
                border-radius: 5px;
                cursor: pointer;
            }
            input:hover, input:active {
                border-color: blue;
                background-color: gainsboro;
            }
        </style>
</head>

<body>
        <form method="post" action="">
        <fieldset>
            <legend>Użytkownik <i>$username</i> zalogowany
poprawnie</legend>
            <ul>
```

```
            <li>E-mail: {{email}}</li>
            <li>Płeć: {{gender}}</li>
            <li>Data urodzenia: {{dofb}}</li>
            <li>Kraj: {{country}}</li>
            <li>Obywatel UE: {{ueCitizen}}</li>
            </ul>
            <center>
            <br>

            <input type="hidden" name="username" value="{$username}">
            <input type="hidden" name="password" value="{$password}">

            <button name="action" value="delete"
                  onclick="return confirm('Na pewno usunąć konto?')">
            Usuń konto
            </button>
            <br><br>
            <a href="index.html" target:_self>
                  <input type="button" value="Powrót na stronę
główną">
            </a>
            <center>
      </fieldset>
      </form>
</body>
</html>
CODE;

$page_denied = <<<CODE
<html>
<head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <title>Odmowiono dostepu</title>
      <style>
            body {
                  display: flex;
                  justify-content: center;
                  align-items: center;
```

```
                height: 100vh;
                margin: 0;
                background-color: #f0f0f0;
            }
            form {
                width: 500px;
                background-color: #fff;
                padding: 20px;
                border-radius: 8px;
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            }
            fieldset {
                border: 1px solid black;
                padding: 10px;
                margin: 0;
            }
            legend {
                font-weight: bold;
                font-size: 1.2em;
                margin-bottom: 5px;
            }
            label {
                margin-bottom: 5px;
            }
            input[type="button"] {
                padding: 10px 20px;
                border-radius: 5px;
                cursor: pointer;
            }
            input:hover, input:active {
                border-color: blue;
                background-color: gainsboro;
            }
        </style>
</head>

<body>
        <form method="post" action="">
        <fieldset>
            <legend>Błędny login i/lub hasło</legend>
```

```
            <center>
            <a href="formularz_php.html" target:_self>
                  <input type="button" value="Powrót do strony
logowania">
            </a>
            <br><br>
            <a href="index.html" target:_self>
                  <input type="button" value="Powrót na stronę
główną">
            </a>
            <center>
      </fieldset>
      </form>
</body>
</html>
CODE;

if (isset($_POST['action']) && $_POST['action'] === 'delete') {
      if (!checkPass($password, $username)) {
            print("$page_denied");
            exit;
      }
      deleteUser($username);
      echo "<script>
            alert('Konto $username zostało usunięte.');
            window.location.href = 'index.html';
      </script>";
      exit;
}

function checkPass($pass, $user) {
      if (!$fd = @fopen("passwords.txt", "r")) return false;
      if (!$fdj = @fopen("user_data.json", "r")) return false;
      while (!feof ($fd)) {
            $line = trim(fgets($fd));
            if (($pos = strpos($line, ":")) === false) continue;

            $tempUser = substr($line, 0, $pos);
            if ($tempUser != $user) continue;
```

```
            $tempPass = substr($line, $pos + 1, strlen($line) + $pos);
            if ($tempPass != md5($pass)) continue;
            else return true;
      }
      fclose($fd);
      fclose($fdj);
      return false;
}

if (!checkPass($password, $username)) {
      print("$page_denied");
} else {

      $handle = fopen("user_data.json", "r");

      $email = $gender = $dofb = $country = '';
      $ueCitizen = 'no';
      while (($line = trim(fgets($handle))) !== false) {
            if ($line === '') continue;
            $entry = json_decode($line, true);
            if (json_last_error() !== JSON_ERROR_NONE) {
                  continue;
            }
            if (isset($entry['username']) && $entry['username'] ===
$username) {
                  $email = $entry['email'] ?? '';
                  $gender = $entry['gender'] ?? '';
                  $dofb = $entry['dofb'] ?? '';
                  $country = $entry['country'] ?? '';
                  $ueCitizen = !empty($entry['UE_citizen']) ? 'yes' :
'no';
                  break;
            }
      }
      fclose($handle);

      $replacements = [
            '{{email}}' => htmlspecialchars($email, ENT_QUOTES, 'UTF-
8'),
```

```
            '{{gender}}' => htmlspecialchars($gender, ENT_QUOTES,
'UTF-8'),
            '{{dofb}}' => htmlspecialchars($dofb, ENT_QUOTES, 'UTF-
8'),
            '{{country}}' => htmlspecialchars($country, ENT_QUOTES,
'UTF-8'),
            '{{ueCitizen}}' => htmlspecialchars($ueCitizen,
ENT_QUOTES, 'UTF-8'),
      ];

      echo str_replace(
            array_keys($replacements),
            array_values($replacements),
            $page_granted
      );
}
?>
```

Program ten można także zmodyfikować w taki sposób, aby umożliwić zalogowanemu użytkownikowi zmianę danych. Akcja modyfikacji mogłaby być wyzwalana dodatkowym przyciskiem otwierającym okno (domyślnie ukryte) lub nową podstronę z uproszczonym formularzem. Zmiany przepisywane byłyby do pliku „user_data.json” na podobnej zasadzie, co uzuwanie, a więc poprzez przepisywanie poszczególnych wierszy do pliku tymczasowego z filtracją słów kluczowych i podmianą danych. Możnaby zatem wykorzystać fragment za to odpowiadający w funkcji „deleteUser” po jej przekopiowaniu do nowej funkcji o przykładowej nazwie „updateUser”:

```
function updateUser(string $user, string $email, string $country) {
      $in  = fopen("user_data.json", "r");
      $out = fopen("user_data.tmp", "w");

      while (($line = fgets($in)) !== false) {
            $entry = json_decode(trim($line), true);

            if (isset($entry['username']) && $entry['username'] ===
      $user) {
                  $entry['email'] = $email;
                  $entry['country'] = $country;
```

```
                fwrite($out, json_encode($entry,
            JSON_UNESCAPED_UNICODE) . "\n");
            } else {
                fwrite($out, $line);
            }
        }
        fclose($in);
        fclose($out
        rename("user_data.tmp", "user_data.json");
}
```

Tutaj również zaczynamy od otworzenia pliku właściwego i tymczasowego. Następnie przechodzimy linia po linii w poszukiwaniu właściwej nazwy użytkownika. Po jego znalezieniu szukamy słów kluczowych (tu zmieniamy email i kraj zamieszkania), po czym na końcu podmieniamy pliki. Dane są podmieniane niezależnie od tego czy byłyby zmodyfikowane prze użytkownika dlatego w przypadku tej funkcji należy zawsze przekazać skopiowane wartości domyślne. Zastosowanie mechanizmów walidacyjnych w formularzu zmiany danych z wpisanymy jako domyślne wartości aktualne w zupełności wystarczy. Warto nadmienić, że w podanych programach w zasadzie nie stosowano komunikatów o błędach. Na te potrzeby można użyć słowa „die” lub „throw new RuntimeException”, np. gdyby nie udało się otworzyć plików, stosowny komunikat zostałby wyświetlony dzięki:

```
if (!$in || !$out) {
    throw new RuntimeException("Nie można otworzyć plików do
odczytu/zapisu.");
}
```

Warto jest przygotowywać komunikaty o błędach w postaci okien lub wypełniania kontrolek tekstowych aby po pierwsze użytkownik był świadom, że coś jest nie tak, a po drugie aby programista mógł zlokalizować błąd. Dobrą praktyką jest także przygotowanie pliku z logami zapisującymi chronologicznie wszystkie operacje wyzwolone przez użytkownika i system. Plik ten najczęściej jest tekstowy i z regóły nie zawiera jawnie zapisanych danych wrażliwych.

5.3. Hasła i logowanie, dane w bazie danych

Język PHP jest bardzo często wykorzystywany do dostępu do bazy danych rezydującej na tym samym serwerze. Baza taka może służyć do przechowywania

zarówno prostych rekordów w postaci loginu i hasła (także zaszyfrowanego), jak i większej ilości danych o użytkowniku. Mogą to być dane zapisywane w naszym przykładzie w pliku „user_data.json”. Działania na bazie danych w naszym przypadku (niech to będzie MySQL) wyglądałyby następująco: jeżeli użytkownik poprawnie wypełni formularz rejestracyjny dane zostaną zapisane zarówno w pliku „user_data.json”, jak i bazie danych. W przypadku zalogowania się użytkownika obecnie dane są odczytywane z pliku JSON, możemy do celów demonstracyjnych rozbudować stronę o dane pobierane bezpośrednio z bazy danych. W przypadku usunięcia konta użytkownika poza obecnym mechanizmem operowania na plikach tekstowych przygotowana zostanie także funkcja usuwająca odpowiedniego wiersza w bazie danych. Z wykorzystaniem funkcji „updateUser” z poprzedniego podrozdziału można postarać się o rozbudowanie strony o mechanizm modyfikacji danych. Dane byłyby jednak modyfikowane zarówno na plikach tekstowych, jak i w bazie. Możliwa jest realizacja zarówno równoległa (niezależna) dla każdego typu repozytorium, jak i rzutowanie zmian z pliku na bazę lub odwrotnie. Rozpoczijmy jednak od utworzenia struktury w bazie danych MySQL. Tworzymy bazę „user_data_sql” z tabelą „userform” oraz sześcioma kolumnami o nazwach odpowiadających danym pobieranym z formularza. Struktura tabeli zaprezentowana jest na rys. 18.

[Tekst alternatywny: Zrzut ekranu z phpMyAdmin przedstawiający strukturę tabeli bazy danych. U góry nagłówki kolumn: Nazwa pola; Typ; Collation; Attributes; Null; Default; Comments; Extra; Action. Poniżej sześć wierszy z polami: user_SQL: text, utf8mb4_general_ci, Nie, brak, –; email_SQL: text, utf8mb4_general_ci, Nie, brak, –; gender_SQL: text, utf8mb4_general_ci, Nie, brak, –; birth_SQL: text, utf8mb4_general_ci, Nie, brak, –; country_SQL: text, utf8mb4_general_ci, Nie, brak, –; citizen_SQL: text, utf8mb4_general_ci, Nie, brak, –.]

	#	Nazwa	Typ	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**user_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej
☐	2	**email_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej
☐	3	**gender_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej
☐	4	**birth_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej
☐	5	**country_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej
☐	6	**citizen_SQL**	text	utf8mb4_general_ci		Nie	*Brak*			Change Drop Więcej

Rys. 18. Struktura tabeli do przechowywania danych zarejestrowanych użytkowników.

Teraz rozbudujmy programy o funkcje do wypełniania tej tabeli danymi. Zdefiniujmy nową funkcję `function insertUserToDb(array $data): bool`. Zestawiamy nowe połączenie TCP-owe z ominięciem socketów na Windows/XAMPP:

```
$conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
if ($conn->connect_errno) {
      error_log("DB connect error: " . $conn->connect_error);
      return false;
}
```

Przygotowujemy zapytanie:

```
$sql = "INSERT INTO usersform
           (user_SQL, email_SQL, gender_SQL, birth_SQL, country_SQL,
citizen_SQL)
           VALUES (?, ?, ?, ?, ?, ?)";
$stmt = $conn->prepare($sql);
if (!$stmt) {
      error_log("DB prepare error: " . $conn->error);
      $conn->close();
      return false;
}
```

Mapujemy wartości i typy:

```
$username = $data['username'];
$email = $data['email'];
$gender = $data['gender'];
$birth = $data['dofb'];
$country = $data['country'];
$citizen = $data['UE_citizen'] ? '1' : '0';

$stmt->bind_param(
      "ssssss",
      $username,
      $email,
      $gender,
      $birth,
      $country,
      $citizen
);
```

Wykonujemy zapytanie i zwaracy wynik:

```
$ok = $stmt->execute();
if (!$ok) {
      error_log("DB execute error: " . $stmt->error);
}

$stmt->close();
$conn->close();

return $ok;
```

Tuż przed linią `print("$user_granted");` wywołujemy funkcję bazodanową poprzedzając ją dekodowaniem wiersza JSON:

```
$userData = json_decode($_POST['jsonData'], true);

if (
      is_array($userData) &&
      isset(
            $userData['username'],
            $userData['email'],
            $userData['gender'],
            $userData['dofb'],
            $userData['country'],
            $userData['UE_citizen']
      )
) {
      $dbSuccess = insertUserToDb($userData);
}
```

Możemy w tym miejscu reagować na „$dbSuccess” poprzez np. automatyczne logowanie lub dodawanie okien alertów. Po wpisaniu przykładowych danych w podglądzie bazy danych uzyskamy widok jak na rys. 19.

[Tekst alternatywny: Zrzut ekranu z phpMyAdmin przedstawiający tabelę bazy danych z jedną wierszem danych. Nagłówki kolumn: user_SQL; email_SQL; gender_SQL; birth_SQL; country_SQL; citizen_SQL. Wiersz danych: user_SQL: Jan_Kowalski; email_SQL: janko@q; gender_SQL: male; birth_SQL: 1982-09-24; country_SQL: poland; citizen_SQL: 1.]

user_SQL	email_SQL	gender_SQL	birth_SQL	country_SQL	citizen_SQL
Jan_Kowalski	janko@q	male	1982-09-24	poland	1

Rys. 19. Wiersz w bazie danych z danymi przesłanymi z poprawnie wypełnionego formularza.

Baza danych może być już wypełniania, teraz czas na wyświetlanie danych z niej pobranych i zestawienie ich z danymi w pliku „user_data.json”. Do tego celu wykorzystamy plik PHP sprawdzający wynik logowania i wyświetlający dane. Rozbudujmy część „page_granted” o kod HTML wyświetlający dodatkowe dane poniżej listy z danymi z pliku JSON:

```
<b><i>Dane z bazy danych:</i></b>
<ul>
<li>E-mail: {{db_email}}</li>
<li>Płeć: {{db_gender}}</li>
<li>Data urodzenia: {{db_dofb}}</li>
<li>Kraj: {{db_country}}</li>
<li>Obywatel UE: {{db_ueCitizen}}</li>
</ul>
```

Wewnątrz wyrażenia `if (!checkPass($password, $username))` tuż przed przypisaniem „replacements” wprowadzamy kod do otwarcia azy danych, znalezienia użytkownika po jego nazwie, odczytania danych i zamknięcia bazy danych:

```
$db_email = $db_gender = $db_dofb = $db_country = '';
$db_ueCitizen = 'no';

$conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
if (!$conn->connect_errno) {
     $stmt = $conn->prepare(
          "SELECT email_SQL, gender_SQL, birth_SQL, country_SQL,
citizen_SQL
          FROM usersform
          WHERE user_SQL = ?"
```

```
        );
        if ($stmt) {
            $stmt->bind_param("s", $username);
            $stmt->execute();
            $stmt->bind_result(
                $db_email,
                $db_gender,
                $db_dofb,
                $db_country,
                $db_citizenFlag
            );
            if ($stmt->fetch()) {
                $db_ueCitizen = $db_citizenFlag === '1' ? 'yes' :
'no';
            }
            $stmt->close();
        }
        $conn->close();
}
```

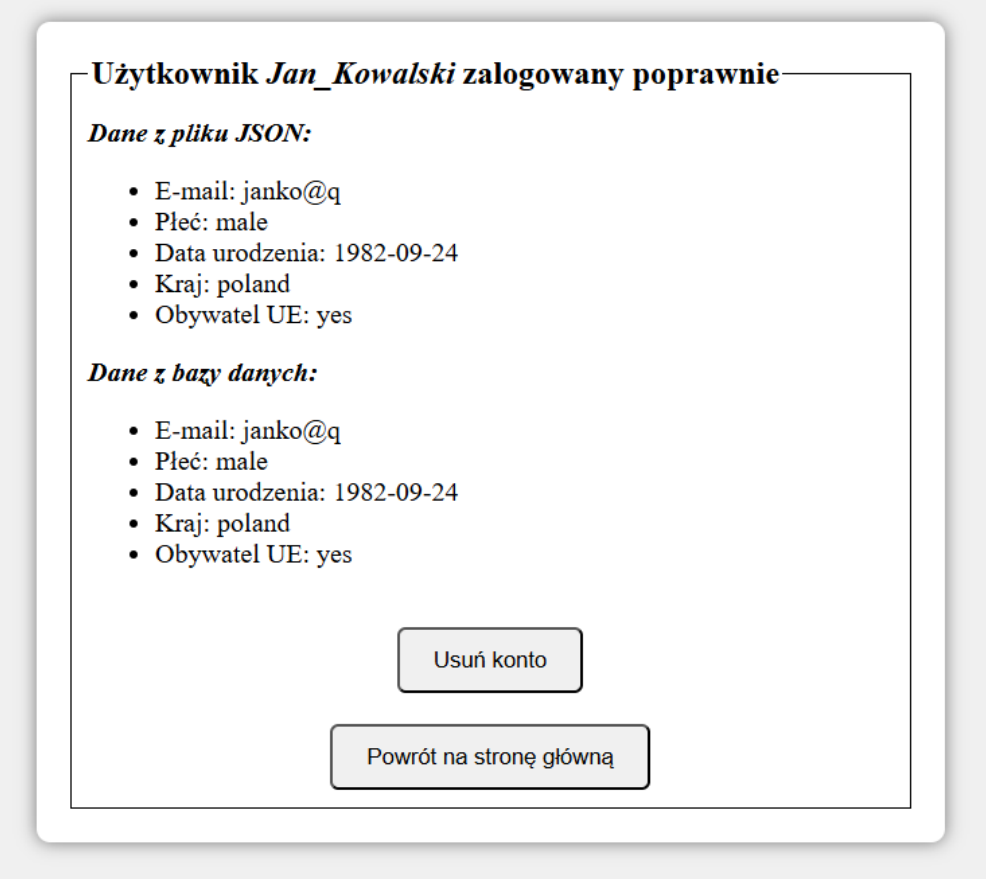
Rozbudowujemy także „replacements” na potrzeby dodatkowych pól w kodzie HTML:

```
$replacements = [
        '{{email}}' => htmlspecialchars($email, ENT_QUOTES, 'UTF-8'),
        '{{gender}}' => htmlspecialchars($gender, ENT_QUOTES, 'UTF-8'),
        '{{dofb}}' => htmlspecialchars($dofb, ENT_QUOTES, 'UTF-8'),
        '{{country}}' => htmlspecialchars($country, ENT_QUOTES, 'UTF-
8'),
        '{{ueCitizen}}' => htmlspecialchars($ueCitizen, ENT_QUOTES,
'UTF-8'),
        '{{db_email}}'    => htmlspecialchars($db_email, ENT_QUOTES,
'UTF-8'),
        '{{db_gender}}'   => htmlspecialchars($db_gender, ENT_QUOTES,
'UTF-8'),
        '{{db_dofb}}'     => htmlspecialchars($db_dofb, ENT_QUOTES,
'UTF-8'),
        '{{db_country}}'  => htmlspecialchars($db_country, ENT_QUOTES,
'UTF-8'),
        '{{db_ueCitizen}}'=> htmlspecialchars($db_ueCitizen,ENT_QUOTES,
'UTF-8'),
```

```
];
```

Po zalogowaniu uzyskamy widok jak na rys. 20.

[Tekst alternatywny. Strona z potwierdzeniem poprawnego logowania w języku polskim. Nagłówek: Użytkownik Jan_Kowalski zalogowany poprawnie. Wypunktowana lista danych z pliku JSON: E-mail: janko@q; Płeć: male; Data urodzenia: 1982-09-24; Kraj: poland; Obywatel UE: yes. Wypunktowana lista danych z bazy danych: E-mail: janko@q; Płeć: male; Data urodzenia: 1982-09-24; Kraj: poland; Obywatel UE: yes. Na dole przyciski: Usuń konto; Powrót na stronę główną.]



Rys. 20. Ekran do wyświetlania danych z pliku JSON oraz bazy danych.

Wymagane jest teraz takie rozbudowanie kodu PHP, aby realizowane było usuwanie użytkownika zarówno z plików TXT i JSON, jak i bazy danych. Zrealizować to można poprzez wprowadzenie następującego kodu w funkcji usuwającej „deleteUser” poniżej części usuwającej dane z pliku JSON:

```
$conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
if (!$conn->connect_errno) {
      $stmt = $conn->prepare(
```

```
            "DELETE FROM usersform WHERE user_SQL = ?"
      );
      if ($stmt) {
            $stmt->bind_param("s", $user);
            $stmt->execute();
            $stmt->close();
      }
      $conn->close();
}
```

Postarajmy się jeszcze bardziej rozbudować ten kod o dodatkowy przycisk „Zmień dane” pozwalający na update pola „email”. Zdefiniujmy funkcję zmeiniającą wartości w zadanych polach:

```
function updateUserData(string $user, string $newEmail): void
{
      $lines = file("user_data.json", FILE_IGNORE_NEW_LINES);
      foreach ($lines as &$line) {
            $entry = json_decode($line, true);
            if (isset($entry['username']) && $entry['username'] ===
$user) {
                  $entry['email'] = $newEmail;
                  $line = json_encode($entry);
                  break;
            }
      }
      file_put_contents("user_data.json", implode("\n", $lines) .
"\n");

      $conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
      if (!$conn->connect_errno) {
            $stmt = $conn->prepare(
                  "UPDATE usersform
                  SET email_SQL = ?
                  WHERE user_SQL = ?"
            );
            if ($stmt) {
                  $stmt->bind_param("ss", $newEmail, $user);
                  $stmt->execute();
                  $stmt->close();
```

```
            }
            $conn->close();
      }
}
```

Parametrami wejściowymi dla tej funkcji są nazwa użytkownika i nowa wartość pola email. Najpierw następuje upgrade w pliku JSON, a później w bazie danych. Przygotujmy zmienne pomocnicze:

```
$currentEmail = !empty($db_email) ? $db_email : $email;
$currentEmail = htmlspecialchars($currentEmail, ENT_QUOTES, 'UTF-
8');
```

Wyświetlmy formularz zmiany e-mail:

```
if (isset($_POST['action']) && $_POST['action'] === 'update') {
      echo <<<CODE
      <html>
      <head>
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width,
initial-scale=1.0">
            <title>Strona z formularzem HTML</title>
            <style>
                  body {
                        display: flex;
                        justify-content: center;
                        align-items: center;
                        height: 100vh;
                        margin: 0;
                        background-color: #f0f0f0;
                  }
                  form {
                        width: 500px;
                        background-color: #fff;
                        padding: 20px;
                        border-radius: 8px;
                        box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
                  }
                  fieldset {
```

```
                    border: 1px solid black;
                    padding: 10px;
                    margin: 0;
                }
                legend {
                    font-weight: bold;
                    font-size: 1.2em;
                    margin-bottom: 5px;
                }
                label {
                    margin-bottom: 5px;
                }
                input[type="text"],
                input[type="email"] {
                    width: calc(100% - 20px);
                    padding: 8px;
                    margin-bottom: 10px;
                    box-sizing: border-box;
                    border: 1px solid #ccc;
                    border-radius: 4px;
                }
                .citizen_UE_check_country label{
                    display: inline;
                }
                input[type="button"] {
                    padding: 10px 20px;
                    border-radius: 5px;
                    cursor: pointer;
                }
                button {
                    padding: 10px 20px;
                    border-radius: 5px;
                    cursor: pointer;
                }
            </style>
        </head>
        <body>
            <form method="post" action="">
                <input type="hidden" name="username"
value="{$username}">
```

```
                <input type="hidden" name="password"
value="{$password}">
                <label>Nowy e-mail:
                <input type="email" name="new_email" required
value="{$currentEmail}">
                </label>
                <button name="action" value="do_update">Zapisz
zmiany</button>
            </form>
      </body>
      </html>
      CODE;
      exit;
}
```

Zapisujemy zmiany i przekierowujemy spowrotem na stronę z danymi z ponownym wczytaniem JSON i bazy danych.

```
if (isset($_POST['action']) && $_POST['action'] === 'do_update') {
      $newEmail = trim($_POST['new_email'] ?? '');
      updateUserData($username, $newEmail);

      header("Location: {$_SERVER['PHP_SELF']}?username="
            .urlencode($username)."&password=".urlencode($password));
      exit;
}
```

Okno formularza zawiera w sobie wbudowany mechanizm walidacji HTML5 dzięki wykorzystaniu obiektu „input” typu „email”. Zmienna „currentEmail” musi być wcześniej odczytana, zatem przez powyższymi deklaracjami należy umieścić następujący kod do odczytywania wszystkich danych:

```
$handle = fopen("user_data.json", "r");
$email = $gender = $dofb = $country = '';
$ueCitizen = 'no';

while (($line = fgets($handle)) !== false) {
      $entry = json_decode(trim($line), true);
      if (isset($entry['username']) && $entry['username'] ===
$username) {
```

```
            $email = $entry['email'] ?? '';
            $gender = $entry['gender'] ?? '';
            $dofb = $entry['dofb'] ?? '';
            $country = $entry['country'] ?? '';
            $ueCitizen = !empty($entry['UE_citizen']) ? 'yes' : 'no';
            break;
      }
}
fclose($handle);

$db_email     = $db_gender = $db_dofb = $db_country = '';
$db_ueCitizen = 'no';

$conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
if (!$conn->connect_errno) {
      $stmt = $conn->prepare(
            "SELECT email_SQL, gender_SQL, birth_SQL, country_SQL,
citizen_SQL
            FROM usersform
            WHERE user_SQL = ?"
      );
      if ($stmt) {
            $stmt->bind_param("s", $username);
            $stmt->execute();
            $stmt->bind_result(
                  $db_email,
                  $db_gender,
                  $db_dofb,
                  $db_country,
                  $db_flag
            );
            if ($stmt->fetch()) {
                  $db_ueCitizen = ($db_flag === '1') ? 'yes' : 'no';
            }
            $stmt->close();
      }
      $conn->close();
}
```

Najpierw następuje odczytanie danych z pliku JSON, później z bazy danych. Rozbudujmy ten formularz o możliwość zmiany miejsca zamieszkania, a więc wyboru elementu z listy. Zmiany w funkcji „updateUserData" wyglądają następująco:

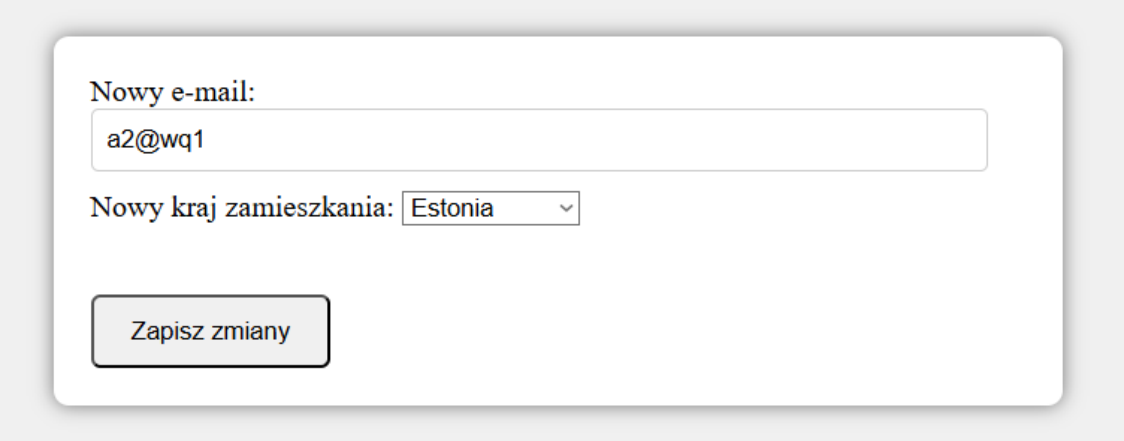
```
function updateUserData(string $user, string $newEmail, string
$newCountry): void
{
     $lines = file("user_data.json", FILE_IGNORE_NEW_LINES);
     foreach ($lines as &$line) {
          $entry = json_decode($line, true);
          if (isset($entry['username']) && $entry['username'] ===
$user) {
               $entry['email'] = $newEmail;
               $entry['country'] = $newCountry;
               $line = json_encode($entry);
               break;
          }
     }
     file_put_contents("user_data.json", implode("\n", $lines) .
"\n");

     $conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
     if (!$conn->connect_errno) {
          $stmt = $conn->prepare(
               "UPDATE usersform
               SET email_SQL   = ?,
                    country_SQL = ?
               WHERE user_SQL  = ?"
          );
          if ($stmt) {
               $stmt->bind_param("sss", $newEmail, $newCountry,
$user);
               $stmt->execute();
               $stmt->close();
          }
          $conn->close();
     }
}
```

Zmiany w pliku z formularzem HTML wymagają dodania stosownego pola wyboru wraz z krajami. Należy także Przechwycić nową wartość pola wybraną przez użytkownika funkcją warunkową wywołaną przyciskiem „Zapisz zmiany”:

```
if (isset($_POST['action']) && $_POST['action'] === 'do_update') {
      $newEmail = trim($_POST['new_email'] ?? '');
      $newCountry = trim($_POST['new_country'] ?? '');
      updateUserData($username, $newEmail, $newCountry);

      header("Location: {$_SERVER['PHP_SELF']}?username="
            .urlencode($username)."&password=".urlencode($password));
      exit;
}
```

Teraz po wybraniu opcji modyfikacji danych ukaże się okno, jak na rys. 21.

[Tekst alternatywny: Formularz aktualizacji danych użytkownika w języku polskim. Opis elementów formularza: Pole tekstowe „Nowy e-mail” z wartością a2@wq1; Pole wyboru „Kraj zamieszkania” ustawione na Estonia; Przycisk „Zapisz zmiany”.]



Rys. 21. Formularz zmiany danych.

Pełny kod programu PHP wygląda następująco:

```
<?php
$username = $_REQUEST['username'] ?? '';
$password = $_REQUEST['password'] ?? '';

function deleteUser(string $user) {
      $lines = file("passwords.txt", FILE_IGNORE_NEW_LINES);
      $filtered = array_filter($lines, function($l) use($user){
```

```
            return strpos($l, "$user:") !== 0;
      });
      file_put_contents("passwords.txt", implode("\n", $filtered));

      $in  = fopen("user_data.json", "r");
      $out = fopen("user_data.tmp",  "w");
      while (($line = fgets($in)) !== false) {
            $entry = json_decode(trim($line), true);
            if (isset($entry['username']) && $entry['username'] ===
$user) {
                  continue;
            }
            fwrite($out, $line);
      }
      fclose($in);
      fclose($out);
      rename("user_data.tmp", "user_data.json");

      $conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
      if (!$conn->connect_errno) {
            $stmt = $conn->prepare(
                  "DELETE FROM usersform WHERE user_SQL = ?"
            );
            if ($stmt) {
                  $stmt->bind_param("s", $user);
                  $stmt->execute();
                  $stmt->close();
            }
            $conn->close();
      }
}

function updateUserData(string $user, string $newEmail, string
$newCountry): void
{
      $lines = file("user_data.json", FILE_IGNORE_NEW_LINES);
      foreach ($lines as &$line) {
            $entry = json_decode($line, true);
            if (isset($entry['username']) && $entry['username'] ===
$user) {
```

```
                $entry['email'] = $newEmail;
                $entry['country'] = $newCountry;
                $line = json_encode($entry);
                break;
            }
        }
        file_put_contents("user_data.json", implode("\n", $lines) .
"\n");

        $conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
        if (!$conn->connect_errno) {
            $stmt = $conn->prepare(
                "UPDATE usersform
                SET email_SQL   = ?,
                    country_SQL = ?
                WHERE user_SQL  = ?"
            );
            if ($stmt) {
                $stmt->bind_param("sss", $newEmail, $newCountry,
$user);
                $stmt->execute();
                $stmt->close();
            }
            $conn->close();
        }
}

$page_granted = <<<CODE
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Udzielono dostepu</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
```

```
            margin: 0;
            background-color: #f0f0f0;
        }
        form {
            width: 500px;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        }
        fieldset {
            border: 1px solid black;
            padding: 10px;
            margin: 0;
        }
        legend {
            font-weight: bold;
            font-size: 1.2em;
            margin-bottom: 5px;
        }
        label {
            margin-bottom: 5px;
        }
        input[type="button"] {
            padding: 10px 20px;
            border-radius: 5px;
            cursor: pointer;
        }
        button {
            padding: 10px 20px;
            border-radius: 5px;
            cursor: pointer;
        }
        input:hover, input:active {
            border-color: blue;
            background-color: gainsboro;
        }
    </style>
</head>
```

```
<body>
     <form method="post" action="">
     <fieldset>
          <legend>Użytkownik <i>$username</i> zalogowany
poprawnie</legend>
          <b><i>Dane z pliku JSON:</b></i></b>
          <ul>
          <li>E-mail: {{email}}</li>
          <li>Płeć: {{gender}}</li>
          <li>Data urodzenia: {{dofb}}</li>
          <li>Kraj: {{country}}</li>
          <li>Obywatel UE: {{ueCitizen}}</li>
          </ul>
          <b><i>Dane z bazy danych:</i></b>
          <ul>
          <li>E-mail: {{db_email}}</li>
          <li>Płeć: {{db_gender}}</li>
          <li>Data urodzenia: {{db_dofb}}</li>
          <li>Kraj: {{db_country}}</li>
          <li>Obywatel UE: {{db_ueCitizen}}</li>
          </ul>
          <center>
          <br>

          <input type="hidden" name="username" value="{$username}">
          <input type="hidden" name="password" value="{$password}">

          <button name="action" value="update">Zmień dane</button>
          <button name="action" value="delete"
               onclick="return confirm('Na pewno usunąć konto?')">
          Usuń konto
          </button>
          <br><br>
          <a href="index.html" target:_self>
               <input type="button" value="Powrót na stronę
główną">
          </a>
          <center>
     </fieldset>
     </form>
```

```
</body>
</html>
CODE;

$page_denied = <<<CODE
<html>
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-
scale=1.0">
	<title>Odmowiono dostepu</title>
	<style>
		body {
			display: flex;
			justify-content: center;
			align-items: center;
			height: 100vh;
			margin: 0;
			background-color: #f0f0f0;
		}
		form {
			width: 500px;
			background-color: #fff;
			padding: 20px;
			border-radius: 8px;
			box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
		}
		fieldset {
			border: 1px solid black;
			padding: 10px;
			margin: 0;
		}
		legend {
			font-weight: bold;
			font-size: 1.2em;
			margin-bottom: 5px;
		}
		label {
			margin-bottom: 5px;
		}
```

```
            input[type="button"] {
                  padding: 10px 20px;
                  border-radius: 5px;
                  cursor: pointer;
            }
            input:hover, input:active {
                  border-color: blue;
                  background-color: gainsboro;
            }
      </style>
</head>

<body>
      <form method="post" action="">
      <fieldset>
            <legend>Błędny login i/lub hasło</legend>
            <center>
            <a href="formularz_php.html" target:_self>
                  <input type="button" value="Powrót do strony
logowania">
            </a>
            <br><br>
            <a href="index.html" target:_self>
                  <input type="button" value="Powrót na stronę
główną">
            </a>
            <center>
      </fieldset>
      </form>
</body>
</html>
CODE;

if (isset($_POST['action']) && $_POST['action'] === 'delete') {
      if (!checkPass($password, $username)) {
            print("$page_denied");
            exit;
      }
      deleteUser($username);
      echo "<script>
```

```
            alert('Konto $username zostało usunięte.');
            window.location.href = 'index.html';
      </script>";
      exit;
}

$handle = fopen("user_data.json", "r");
$email = $gender = $dofb = $country = '';
$ueCitizen = 'no';

while (($line = fgets($handle)) !== false) {
      $entry = json_decode(trim($line), true);
      if (isset($entry['username']) && $entry['username'] ===
$username) {
            $email = $entry['email'] ?? '';
            $gender = $entry['gender'] ?? '';
            $dofb = $entry['dofb'] ?? '';
            $country = $entry['country'] ?? '';
            $ueCitizen = !empty($entry['UE_citizen']) ? 'yes' : 'no';
            break;
      }
}
fclose($handle);

$db_email     = $db_gender = $db_dofb = $db_country = '';
$db_ueCitizen = 'no';

$conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
if (!$conn->connect_errno) {
      $stmt = $conn->prepare(
            "SELECT email_SQL, gender_SQL, birth_SQL, country_SQL,
citizen_SQL
            FROM usersform
            WHERE user_SQL = ?"
      );
      if ($stmt) {
            $stmt->bind_param("s", $username);
            $stmt->execute();
            $stmt->bind_result(
                  $db_email,
```

```
                $db_gender,
                $db_dofb,
                $db_country,
                $db_flag
            );
            if ($stmt->fetch()) {
                $db_ueCitizen = ($db_flag === '1') ? 'yes' : 'no';
            }
            $stmt->close();
        }
        $conn->close();
}

$currentEmail = !empty($db_email) ? $db_email : $email;
$currentEmail = htmlspecialchars($currentEmail, ENT_QUOTES, 'UTF-
8');

if (isset($_POST['action']) && $_POST['action'] === 'update') {
        echo <<<CODE
        <html>
        <head>
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width,
initial-scale=1.0">
            <title>Strona z formularzem HTML</title>
            <style>
                body {
                    display: flex;
                    justify-content: center;
                    align-items: center;
                    height: 100vh;
                    margin: 0;
                    background-color: #f0f0f0;
                }
                form {
                    width: 500px;
                    background-color: #fff;
                    padding: 20px;
                    border-radius: 8px;
                    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
```

```
            }
            fieldset {
                  border: 1px solid black;
                  padding: 10px;
                  margin: 0;
            }
            legend {
                  font-weight: bold;
                  font-size: 1.2em;
                  margin-bottom: 5px;
            }
            label {
                  margin-bottom: 5px;
            }
            input[type="text"],
            input[type="email"] {
                  width: calc(100% - 20px);
                  padding: 8px;
                  margin-bottom: 10px;
                  box-sizing: border-box;
                  border: 1px solid #ccc;
                  border-radius: 4px;
            }
            .citizen_UE_check_country label{
                  display: inline;
            }
            input[type="button"] {
                  padding: 10px 20px;
                  border-radius: 5px;
                  cursor: pointer;
            }
            button {
                  padding: 10px 20px;
                  border-radius: 5px;
                  cursor: pointer;
            }
      </style>
</head>
<body>
      <form method="post" action="">
```

```
                <input type="hidden" name="username"
value="{$username}">
                <input type="hidden" name="password"
value="{$password}">
                <label>Nowy e-mail:
                <input type="email" name="new_email" required
value="{$currentEmail}">
                </label>
                <div class="citizen_UE_check_country">
                     <label for="country">Nowy kraj zamieszkania:
</label>
                     <select name="new_country">
                          <option value="foreign">Poza UE</option>
                          <option value="austria">Austria</option>
                          <option value="belgium">Belgia</option>
                          <option
value="bulgaria">Bułgaria</option>
                          <option
value="croatia">Chorwacja</option>
                          <option
value="republic_of_cyprus">Cypr</option>
                          <option value="czechia">Czechy</option>
                          <option value="denmark">Dania</option>
                          <option value="estonia">Estonia</option>
                          <option
value="finland">Finlandia</option>
                          <option value="france">Francja</option>
                          <option value="greece">Grecja</option>
                          <option value="spain">Hiszpania</option>
                          <option
value="netherlands">Holandia</option>
                          <option value="ireland">Irlandia</option>
                          <option value="lithuania">Litwa</option>
                          <option
value="luxemburg">Luksemburg</option>
                          <option value="latvia">Łotwa</option>
                          <option value="malta">Malta</option>
                          <option value="germany">Niemcy</option>
                          <option value="poland">Polska</option>
```

```
                        <option
value="portugal">Portugalia</option>
                        <option value="romania">Rumunia</option>
                        <option
value="slovakia">Słowacja</option>
                        <option
value="slovenia">Słowenia</option>
                        <option value="sweden">Szwecja</option>
                        <option value="hungary">Węgry</option>
                        <option value="italy">Włochy</option>
                    </select>
                </div><br><br>
                <button name="action" value="do_update">Zapisz
zmiany</button>
            </form>
        </body>
        </html>
        CODE;
        exit;
}

if (isset($_POST['action']) && $_POST['action'] === 'do_update') {
        $newEmail = trim($_POST['new_email'] ?? '');
        $newCountry = trim($_POST['new_country'] ?? '');
        updateUserData($username, $newEmail, $newCountry);

        header("Location: {$_SERVER['PHP_SELF']}?username="
            .urlencode($username)."&password=".urlencode($password));
        exit;
}

function checkPass($pass, $user) {
        if (!$fd = @fopen("passwords.txt", "r")) return false;
        if (!$fdj = @fopen("user_data.json", "r")) return false;
        while (!feof ($fd)) {
            $line = trim(fgets($fd));
            if (($pos = strpos($line, ":")) === false) continue;

            $tempUser = substr($line, 0, $pos);
            if ($tempUser != $user) continue;
```

```
            $tempPass = substr($line, $pos + 1, strlen($line) + $pos);
            if ($tempPass != md5($pass)) continue;
            else return true;
      }
      fclose($fd);
      fclose($fdj);
      return false;
}

if (!checkPass($password, $username)) {
      print("$page_denied");
} else {

      $handle = fopen("user_data.json", "r");

      $email = $gender = $dofb = $country = '';
      $ueCitizen = 'no';
      while (($line = trim(fgets($handle))) !== false) {
            if ($line === '') continue;
            $entry = json_decode($line, true);
            if (json_last_error() !== JSON_ERROR_NONE) {
                  continue;
            }
            if (isset($entry['username']) && $entry['username'] ===
$username) {
                  $email = $entry['email'] ?? '';
                  $gender = $entry['gender'] ?? '';
                  $dofb = $entry['dofb'] ?? '';
                  $country = $entry['country'] ?? '';
                  $ueCitizen = !empty($entry['UE_citizen']) ? 'yes' :
'no';
                  break;
            }
      }
      fclose($handle);

      $db_email = $db_gender = $db_dofb = $db_country = '';
      $db_ueCitizen = 'no';
```

```
    $conn = new mysqli('127.0.0.1', 'root', '', 'user_data_sql');
    if (!$conn->connect_errno) {
        $stmt = $conn->prepare(
            "SELECT email_SQL, gender_SQL, birth_SQL,
country_SQL, citizen_SQL
            FROM usersform
            WHERE user_SQL = ?"
        );
        if ($stmt) {
            $stmt->bind_param("s", $username);
            $stmt->execute();
            $stmt->bind_result(
                $db_email,
                $db_gender,
                $db_dofb,
                $db_country,
                $db_citizenFlag
            );
            if ($stmt->fetch()) {
                $db_ueCitizen = $db_citizenFlag === '1' ? 'yes'
: 'no';
            }
            $stmt->close();
        }
        $conn->close();
    }

    $replacements = [
        '{{email}}' => htmlspecialchars($email, ENT_QUOTES, 'UTF-
8'),
        '{{gender}}' => htmlspecialchars($gender, ENT_QUOTES,
'UTF-8'),
        '{{dofb}}' => htmlspecialchars($dofb, ENT_QUOTES, 'UTF-
8'),
        '{{country}}' => htmlspecialchars($country, ENT_QUOTES,
'UTF-8'),
        '{{ueCitizen}}' => htmlspecialchars($ueCitizen,
ENT_QUOTES, 'UTF-8'),
        '{{db_email}}'    => htmlspecialchars($db_email,
ENT_QUOTES, 'UTF-8'),
```

```
            '{{db_gender}}'   => htmlspecialchars($db_gender,
ENT_QUOTES, 'UTF-8'),
            '{{db_dofb}}'     => htmlspecialchars($db_dofb,
ENT_QUOTES, 'UTF-8'),
            '{{db_country}}'  => htmlspecialchars($db_country,
ENT_QUOTES, 'UTF-8'),
            '{{db_ueCitizen}}'=>
htmlspecialchars($db_ueCitizen,ENT_QUOTES, 'UTF-8'),
      ];

      echo str_replace(
            array_keys($replacements),
            array_values($replacements),
            $page_granted
      );
}
?>
```

Zakończono zatem prezentowanie podstawowych operacji wymiany danych pomiędzy klientem i serwerem na żądania użytkownika. Dalsze prace mogą dotyczyć deklaracji konta administratora mającego specjalne uprawnienia usuwania, dodawania i modyfikacji użytkowników. Ponadto możliwe jest rozbudowanie zaprezentowanych skryptów celem zbudowania np. sklepu internetowego, forum internetowego, komunikatora lub nawet gry online. Należy także zadbać o zapisywanie informacji operacji dostępu do danych i aktywności użytkowników. W przypadku implementacji powyższych kodów na różne wersje języka PHP oraz różne bazy danych inne niż MySQL mogą być wymagane pewne zmiany zależnie od wymagań zawartych w dokumentacji.

6. Literatura

- ✓ materiały zwarte:
  - Bowers M, Synodinos D, Summer V. (2013). *HTML5 i CSS3 Zaawansowane wzorce projektowe*, Wydawnictwo Helion, Gliwice.
  - Lis M. (2003). *PHP 101 praktycznych skryptów*, Wydawnictwo Helion. Gliwice.
  - Zakas N. D. (2017). *Wydajny JavaScript*, Wydawnictwo APN Promise, Warszawa
- ✓ źródła internetowe:
  - Patel M. K. (2018). *HTML, CSS, Bootstrap, Javascript and jQuery*, (https://www.kufunda.net/publicdocs/html-css-bootstrap-javascript-and-jquery.pdf dostępny 29.07.2025).