

**POLITECHNIKA ŚLĄSKA
WYDZIAŁ INŻYNIERII MATERIAŁOWEJ**

**Kierunek: Informatyka Przemysłowa
Profil praktyczny
Rodzaj studiów:
stacjonarne I stopnia**

Projekt inżynierski

Radosław Loth

**TRÓJWARSTWOWA APLIKACJA
INTERNETOWA – DZIENNIK
TRENINGOWY.**

**Full-stack three-tier web application - Workout
Diary.**

**Kierujący pracą:
dr inż. Adrian Smagór**

**Recenzent:
dr inż. Maciej Sajkowski**

Katowice, czerwiec 2022 r.

Spis treści

Spis treści.....	2
1 Wstęp.....	4
2 Cel Pracy	4
3 Zakres Pracy	5
4 Projekt aplikacji.....	6
4.1 Warstwa klienta	6
4.1.1 Single-page application	7
4.1.2 HTML	7
4.1.3 CSS	7
4.1.4 TypeScript	7
4.1.5 Angular	7
4.1.6 Bootstrap.....	8
4.1.7 Font Awesome	8
4.1.8 Angular Toastr	8
4.2 Warstwa aplikacji.....	8
4.2.1 REST API.....	8
4.2.2 Java	9
4.2.3 Spring Boot.....	9
4.2.4 Json Web Token	9
4.2.5 Bcrypt	9
4.2.6 Lombok.....	9
4.2.7 OpenAPI	10
4.3 Warstwa danych.....	10
4.3.1 MySQL	12
4.3.2 SQL.....	12
4.3.3 Liquibase	12

5	Przedstawienie aplikacji.....	13
5.1	Wymagania funkcjonalne	13
5.2	Wymagania niefunkcjonalne	14
5.3	Scenariusze przypadków użycia	14
5.3.1	Autoryzacja.....	14
5.3.2	Obsługa dziennika	22
6	Podsumowanie	29
7	Wnioski	30
8	Literatura	31

1 Wstęp

Ponad połowa światowej populacji ludzi korzysta z Internetu mobilnego [1]. Duża część użytkowników codziennie uzyskuje dostęp do usług internetowych za pomocą urządzeń mobilnych, takich jak komputery przenośne, smartfony lub inteligentne zegarki. Istnieje bardzo dużo zastosowań dla powszechnego dostępu do sieci internetowej.

Istnieje wiele rodzajów aplikacji mobilnych oraz programów komputerowych. Mocno zauważalny jest trend, w którym popularyzowane są rozwiązania internetowe, gdzie dane użytkownika nie są przechowywane w pamięci lokalnej urządzenia, a zamiast tego w lokalizacjach sieciowych. Są to często używane usługi takie jak kalendarz, notatnik, poczta elektroniczna, jak również synchronizacja kontaktów, wiadomości i zdjęć. Zapisywanie danych użytkownika do lokalizacji sieciowych wpływa na komfort i wygodę korzystania z usług, ponieważ użytkownik otrzymuje aktualny stan aplikacji na wszystkich urządzeniach jednocześnie. Jest to możliwe dzięki rozwojowi sieci komputerowych, a w głównej mierze zwiększeniu dostępności oraz przepustowości Internetu mobilnego, który za sprawą popularyzacji rozwiązań chmurowych spowodował, że sposób korzystania z urządzeń przypomina popularny w latach 80 schemat wielu terminali oraz jednego systemu komputerowego.

Tak duża skala oraz popularność powodują, że potrzebne jest coraz więcej produktów, które zaspokajają potrzeby współczesnych użytkowników. W związku ze wspomnianym faktem oraz tym, że większość rozwiązań stosowanych w programach komputerowych jest powtarzalna, to popularność na rynku zdobywają technologie, które upraszczają proces inżynierii oprogramowania. Wspomniane technologie dają łatwy dostęp do prostego w użyciu, skutecznego i sprawdzonego szeregu rozwiązań programistycznych. Korzystanie z takich rozwiązań pozwala na szybsze wytwarzanie oprogramowania oraz, dzięki silnie rozbudowanym społecznościom skupionym wokół tego typu projektów, wpływają one na jego stabilność i bezpieczeństwo, ponieważ są testowane przez dużo szersze grupy specjalistów niż byłoby to możliwe bez ich zaangażowania. Niniejsza praca przedstawia propozycję wykorzystania oraz implementacji przoduujących rozwiązań i technologii programistycznych [2].

2 Cel pracy

Celem pracy jest budowa aplikacji internetowej pozwalającej na prowadzenie dziennika treningowego, który umożliwi wprowadzanie i prezentacje danych z wykorzystaniem dostępnych na rynku platform i bibliotek programistycznych. Aplikacja została zaprojektowana z myślą

o kontrolowaniu zadanych wartości w ustalonych aktywnościach. Program zapewnia kontrolę nad wprowadzanymi danymi oraz ich przedstawienie na wykresie.

Niniejsza praca odpowiada na zapotrzebowania współczesnego użytkownika i przedstawia propozycję usługi dostępnej z sieci internetowej, pozwalającej na prowadzenie dziennika treningowego, a jej głównym założeniem jest implementacja, zastosowanie oraz prezentacja przodujących na rynku rozwiązań, i platform programistycznych.

3 Zakres pracy

Praca obejmuje zagadnienia związane z inżynierią oprogramowania, a w szczególności programowania aplikacji internetowych. Praca przedstawia wybrane przez autora aspekty inżynierii oprogramowania, w których opisano architekturę, teorię i technologie programistyczne oraz uzyskany z ich pomocą rezultat.

Rozdział 4 zawiera przedstawienie oraz podstawy teoretyczne wykorzystanych rozwiązań programistycznych wymaganych do zrozumienia tematu pracy. W rozdziale tym omówiono architekturę trójwarstwową projektu. Rozdział 4 został podzielony na trzy główne podrozdziały odpowiadające konkretnej warstwie aplikacji.

W podrozdziale 4.1 omówiona została istota aplikacji jednostronicowej oraz to, w jaki sposób strona internetowa upodabnia sposób swojego działania do okienkowej aplikacji komputerowej. W dalszej części zostały omówione konkretne technologie wykorzystane do utworzenia aplikacji użytkownika.

W rozdziale 4.2 omówiono architekturę aplikacji, czyli programu komputerowego działającego na serwerze. Do tego serwera kierowane są zapytania wychodzące z aplikacji użytkownika. Na początku omówiona zostaje architektura oprogramowania, a następnie konkretne technologie wykorzystane do jego utworzenia.

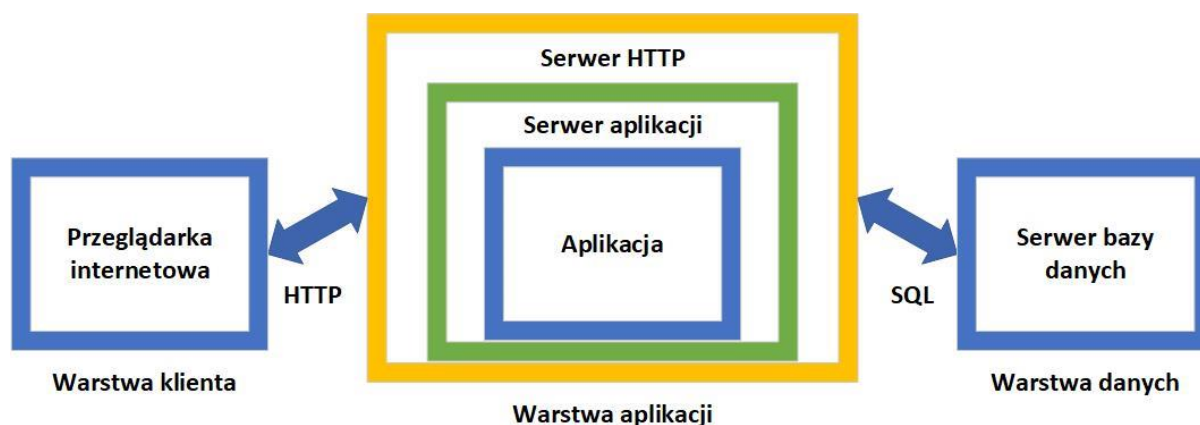
W podrozdziale 4.3 omówiona została warstwa danych. Dostęp do niej możliwy jest jedynie z warstwy aplikacji. Warstwa aplikacji stanowi warstwę pośredniczącą między aplikacją uruchomioną w przeglądarce internetowej użytkownika końcowego, a magazynem danych, który pracuje po stronie dostawcy usługi.

Rozdział 5 zawiera przedstawienie obsługi, wyglądu graficznego interfejsu użytkownika oraz sposobu działania aplikacji na przykładzie diagramów przedstawiających uproszczone scenariusze przypadków jej użycia. Sposób działania aplikacji przedstawiony jest za pomocą zrzutów ekranu graficznego interfejsu użytkownika aplikacji oraz wybranych fragmentów kodu i dzienników zdarzeń aplikacji, jak również bazy danych.

Podsumowanie pracy oraz wnioski końcowe ujęte zostały w rozdziale 5 i 6.

4 Projekt aplikacji

Praca przedstawia wykonanie kompletnej aplikacji internetowej w architekturze trójwarstwowej [3]. Aplikacja działa na podstawie osobistych kont użytkowników oraz jest dostępna z dowolnego urządzenia, które ma połączenie z siecią Internet oraz dostęp do przeglądarki internetowej.



Rysunek 4.1 Diagram architektury trójwarstwowej. Opracowanie własne.

Powyższy diagram (rysunek 4.1) przedstawia uproszczony schemat architektury oraz sposób działania aplikacji. Rozpoczynając od lewej strony rysunku widoczna jest warstwa klienta, która została zaprojektowana z użyciem platformy programistycznej Angular i uruchamiana jest po stronie klienta w przeglądarce internetowej. Warstwa klienta komunikuje się z warstwą aplikacji poprzez protokół HTTP. Warstwa aplikacji została zaprojektowana z użyciem platformy programistycznej Spring Boot i odpowiedzialna jest za przetwarzanie żądań ze strony warstwy klienta oraz zapisywanie i odczytywanie danych w warstwie danych na serwerze bazy danych, który umożliwia dostęp do systemu zarządzania relacyjnymi bazami danych MySQL.

4.1 Warstwa klienta

Udostępnia graficzny interfejs użytkownika, odpowiada za wymianę informacji z serwerem aplikacji oraz ich wyświetlanie. Została zaprogramowana w modelu Single-Page Application z użyciem języka programowania TypeScript, czyli nadzbioru języka JavaScript z użyciem platformy programistycznej Angular. Jest to warstwa najwyższego poziomu i jest uruchamiana w przeglądarce internetowej.

4.1.1 Single-Page Application

SPA (ang. Single-Page Application) to aplikacja, która jest pojedynczą stroną internetową i jest uruchamiana w przeglądarce internetowej. Charakteryzuje się dynamicznym renderowaniem zawartości strony, dzięki czemu strona nie wymaga przeładowywania w trakcie pracy. Strona startowa aplikacji jest jedyną stroną pobieraną w całości z serwera, a następnie jej zawartość jest dynamicznie podmieniana w kontekście interakcji z użytkownikiem. Zapytania HTTP do serwera aplikacji są wykonywane w sposób asynchroniczny co zapewnia możliwość wyświetlania nowego widoku bez konieczności przeładowywania całego dokumentu. Sposób działania aplikacji SPA sprawia wrażenie korzystania z aplikacji okienkowych [4][5].

4.1.2 HTML

Hipertekstowy język znaczników (ang. HyperText Markup Language), jest to język znaczników stosowany do opisu dokumentów tekstowych. Dokument HTML złożony jest z elementów opisywanych znacznikami, które można rozszerzać o atrybuty, między innymi klasy CSS [6][7][8].

4.1.3 CSS

Jest to język arkuszy stylów. Umożliwia selektywne stylizowanie komponentów HTML. Przez specjalne dyrektywy definiowane jest, w jaki sposób przeglądarka internetowa powinna zinterpretować wygląd dokumentu HTML [9][10][11].

4.1.4 TypeScript

Jest to wieloparadygmatowy, statycznie typowany język programowania opracowany przez firmę Microsoft. Język TypeScript jest nadzbiorem języka JavaScript kompilowanym do języka JavaScript [12][13].

4.1.5 Angular

Jest to otwarta źródłowa platforma programistyczna, wspierana i rozwijana przez Google. Służy do projektowania oraz tworzenia aplikacji internetowych typu SPA. Angular jest oparty o komponenty, które zapewniają enkapsulację elementów widoku, stylu oraz logiki aplikacji. Komponenty definiują widoki, czyli wydzielone elementy interfejsu użytkownika. Komponent składa się z plików takich jak typowa strona internetowa to znaczy z pliku HTML, CSS oraz TypeScript, który jest kompilowany do JavaScript. Dzięki podziałowi na komponenty kod jest

modułowy i raz zaprojektowane komponenty można wykorzystywać wielokrotnie w różnych miejscach aplikacji [14][15].

4.1.6 Bootstrap

Jest to otwartoźródłowy zbiór bibliotek CSS, który zapewnia możliwość korzystania z gotowych rozwiązań opartych o klasy CSS, jak również komponentów złożonych z plików HTML i JavaScript, co pozwala na uzyskanie efektów animacji. Są to takie komponenty jak: akordeon, alerty, formularze, grupy przycisków, karty, komunikaty, listy, listy rozwijane, paginacja, paski postępu i inne [16]. Biblioteka zapewnia proste tworzenie graficznego interfejsu użytkownika. W projekcie zostały wykorzystane różne implementacje biblioteki Bootstrap [17], w tym NG Bootstrap [18] oraz MDBBootstrap [19].

4.1.7 Font Awesome

Jest to zbiór skalowalnych ikon przeznaczonych do użytku na stronach internetowych. Pozwala na wykorzystanie odpowiednio przygotowanych plików SVG w plikach HTML [20].

4.1.8 Angular Toastr

Jest to otwartoźródłowa biblioteka przeznaczona dla platformy Angular. Udostępnia możliwość skorzystania z gotowego komponentu, który wyświetla komunikaty w postaci wyskakującego okienka [21].

4.2 Warstwa aplikacji

Jest to warstwa odpowiedzialna za przetwarzanie zapytań HTTP i stanowiąca warstwę pośrednią pomiędzy warstwą klienta, a warstwą danych służącą do komunikacji z bazą danych. Została zaprojektowana w modelu REST API z użyciem języka programowania Java oraz platformy programistycznej Spring Boot rozszerzającej możliwości języka Java.

4.2.1 REST API

REST (ang. Representational State Transfer) oznacza określoną architekturę tworzenia oprogramowania zgodnie ze zdefiniowanymi regułami określającymi definicję zasobów oraz dostępu do nich. API (ang. Application Programming Interface) oznacza zestaw reguł określający sposób komunikacji pomiędzy programami lub systemami komputerowymi. Interfejs REST API to interfejs API zaprojektowany zgodnie z założeniami projektowania REST [22][23].

4.2.2 Java

Jest to język programowania wysokiego poziomu. Kod źródłowy Javy jest kompilowany do kodu bajtowego i uruchamiany w maszynie wirtualnej Javy. Maszyna wirtualna Javy może być uruchamiana na wielu systemach operacyjnych, co uniezależnia Javę od architektury oraz zapewnia łatwą przenośność programów komputerowych zaprogramowanych w Javie [24].

4.2.3 Spring Boot

Jest to narzędzie zapewniające predefiniowaną konfigurację oraz kontener aplikacji dla platformy programistycznej Spring [25]. Spring to otwartoźródłowa platforma programistyczna, rozszerzająca możliwości Javy, która posiada mechanizm wstrzykiwania zależności zapewniający system zarządzania komponentami aplikacji oraz ich zależnościami, jak również inne mechanizmy takie jak konwersje typów, walidacja, obsługa wyjątków, zarządzanie zdarzeniami i inne. Biblioteka Spring Boot [26] wysuwa na pierwszy plan konwencję nad konfiguracją (ang. Convention Over Configuration), która jest paradygmatem architektury oprogramowania, zakładającym zmniejszenie ilości decyzji koniecznych do stworzenia konkretnego rozwiązania. Z użyciem Spring Boot możliwe jest wykorzystanie najlepszych praktyk Spring najmniejszym możliwym kosztem potrzebnym do konfiguracji.

4.2.4 Json Web Token

JWT (ang. Json Web Token) to otwarty standard internetowy systematyzujący bezpieczny sposób wymiany danych między dwiema stronami. Token JWT to kompaktowe rozwiązanie służące między innymi do autoryzacji, które może być przesłane w nagłówku żądania HTTP lub przez URL (ang. Uniform Resource Locator) [27].

4.2.5 Bcrypt

Bcrypt to funkcja przystosowana do haszowania haseł oparta o algorytm szyfrujący Eksblowfish. Służy do przetwarzania hasła na kryptograficzny klucz, który jest używany do autoryzacji [28][29].

4.2.6 Lombok

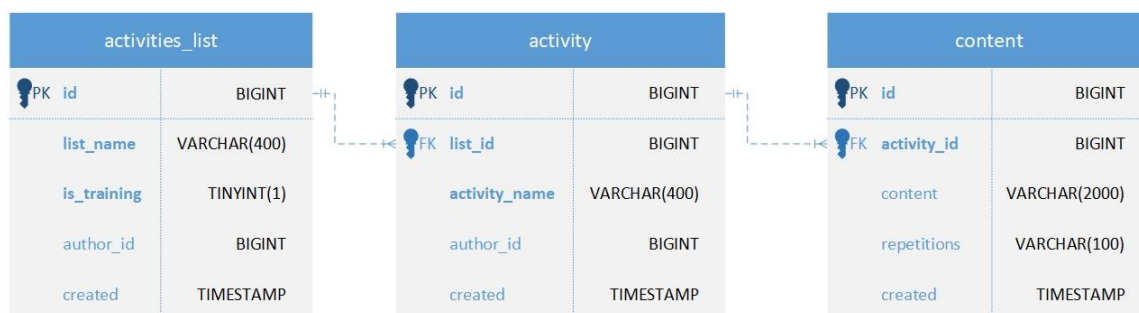
Jest to niewielka biblioteka Javy służąca do automatycznego generowania powtarzalnego kodu. Pozwala zastąpić powtarzalne części kodu na przykład akcesory pól klasy odpowiednimi adnotacjami. Stosowanie adnotacji zwiększa czytelność kodu i oszczędza czas [30].

4.2.7 OpenAPI

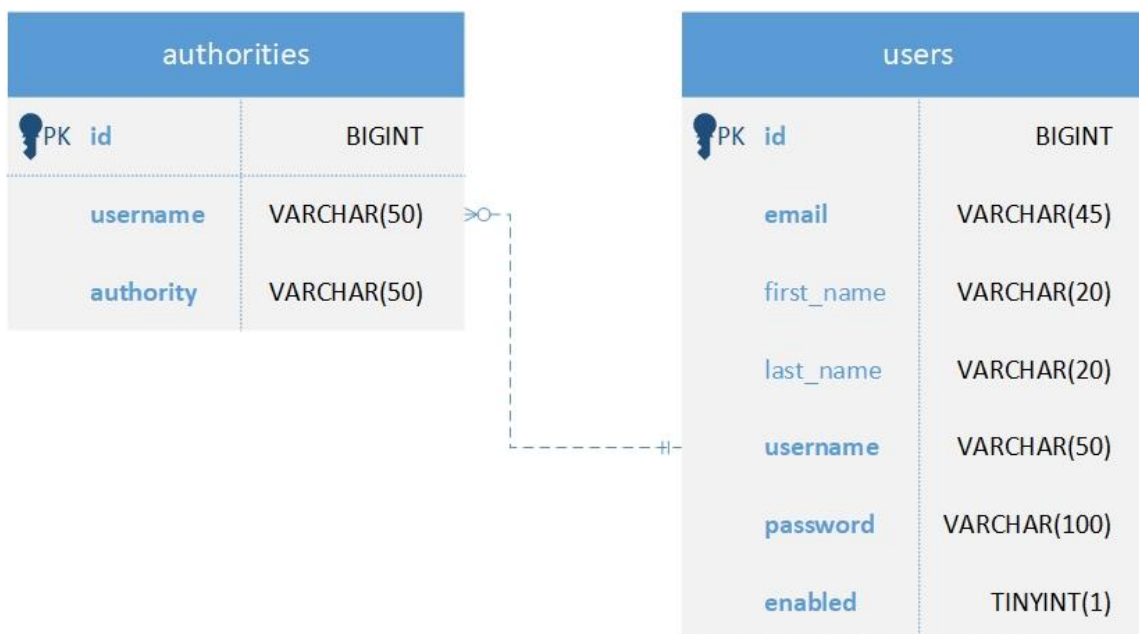
Jest to format opisu interfejsu API dla serwisów typu REST. Użycie OpenAPI pozwala zdefiniować interfejs aplikacji, czyli dostępne punkty końcowe i ich metody HTTP. Do implementacji standardu OpenApi została wykorzystana biblioteka Swagger, czyli zestaw otwartoźródłowych narzędzi, które z wykorzystaniem OpenAPI pomagają w utrzymaniu, projektowaniu, dokumentowaniu i testowaniu serwisu typu REST API [31].

4.3 Warstwa danych


Jest to warstwa oparta o system zarządzania relacyjną bazą danych MySQL i jest odpowiedzialna za przechowywanie informacji oraz zarządzania nimi. Dostęp do warstwy danych odbywa się za pomocą poleceń SQL wydawanych z warstwy aplikacji.




Rysunek 4.2 Schemat bazy danych aplikacji. Opracowanie własne.



Rysunek 4.3 Schemat bazy danych Spring Security. Opracowanie własne.

databasechangelog	
 PK ID	VARCHAR(255)
AUTHOR	VARCHAR(255)
FILENAME	VARCHAR(255)
DATAEXECUTED	DATETIME
ORDEREXECUTED	INT
EXECTYPE	VARCHAR(10)
MD5SUM	VARCHAR(35)
DESCRIPTION	VARCHAR(255)
COMMENTS	VARCHAR(255)
TAG	VARCHAR(255)
LIQUIBASE	VARCHAR(20)
CONTEXTS	VARCHAR(255)
LABELS	VARCHAR(255)
DEPLOYMENT_ID	VARCHAR(10)

databasechangeloglock	
 PK ID	INT
LOCKED	BIT(1)
LOCKGRANTED	DATETIME
LOCKEDBY	VARCHAR(255)

Rysunek 4.4 Schemat bazy danych Liquibase. Opracowanie własne.

Rysunki od 4.2 do 4.4 przedstawiają diagramy trzech wykorzystanych w projekcie baz danych. Wszystkie bazy danych wykorzystane w projekcie oparte są o model relacyjny.

Rysunek 4.2 przedstawia diagram, na którym widoczne są trzy tabele, activities_list, activity, content i odpowiadają one za dane zarządzane przez użytkownika serwisu.

W tabeli activities_list przechowywane są rekordy zawierające pola takie jak list_name określające nazwę zdefiniowanej listy, is_training określające czy lista powiązana jest z aktywnościami treningowymi lub nie, author_id jednoznacznie określające użytkownika, który jest autorem listy oraz created określające datę utworzenia rekordu.

W tabeli activity analogicznie do tabeli list przechowywane są informację o nazwie, autorze i dacie utworzenia, a dodatkowo przechowywany jest atrybut relacji jeden do wielu wskazujący wartość klucza tabeli activities_list. Pozwala to określić do której listy przypisana jest konkretna aktywność.

W tabeli content przechowywany jest atrybut relacji wiążący jeden do wielu, wskazujący na wartość klucza tabeli activity. Wspomniana tabela zawiera informację o zawartości wpisu oraz dacie utworzenia.

Rysunek 4.3 przedstawia dwie tabele authorities i users, które są niezbędne do funkcjonowania modułu autoryzacji, czyli rejestracji i logowania użytkowników zgodnie z modułem Spring Security [32]. W tej bazie danych przechowywane są informacje dotyczące użytkowników między innymi takiej jak nazwa użytkownika, hasło, adres email oraz uprawnienia użytkownika.

Rysunek 4.4 przedstawia tabele śledzenia zmian (ang. Tracking Tables) związane z obsługą biblioteki Liquibase [33].

Tabela DATABASECHANGELOGLOCK zapewnia, że tylko jedna instancja biblioteki Liquibase jest uruchomiona w danym momencie. Służy to uniknięciu konfliktów między współbieżnymi aktualizacjami na bazie danych.

Tabela DATABASECHANGELOG jest wykorzystywana do śledzenia wykonanego zestawu zmian, które zostały zrealizowane na bazie danych. Zapobiega to ponownemu wykonywaniu określonych zmian.

4.3.1 MySQL

Jest to system zarządzania relacyjnymi bazami danych oparty na języku SQL. W systemie relacyjnym dane są reprezentowane jako uporządkowane sekwencje elementów pogrupowanych w zbiory, gdzie każda ze składowych jest członkiem kolekcji wartości zawierających jednostkę danych o precyzyjnym znaczeniu [34].

4.3.2 SQL

Strukturalny język zapytań (ang. Structured Query Language) to język komputerowy używany do pracy z relacyjnymi bazami danych. Język SQL jest określony przez międzynarodowy standard ISO/IEC 9075:2016 [35][36].

4.3.3 Liquibase

Jest to biblioteka do zarządzania zmianami związanymi ze schematem bazy danych. Używana jest do śledzenia, zarządzania i stosowania zmian w schemacie bazy danych. Pozwala na automatyczne, proste w obsłudze odtwarzanie schematu bazy danych. Zapewnia integralność bazy danych, co pozwala na bezpieczne i bezproblemowe uruchamianie aplikacji w nowych środowiskach oraz łatwiejszą kontrolę nad zmianami, szczególnie podczas równoległego rozwoju aplikacji [37].

5 Przedstawienie aplikacji

Przygotowane oprogramowanie, będące przedmiotem niniejszego projektu, możliwe jest do uruchomienia na wszystkich najczęściej spotykanych i aktualnych platformach sprzętowych. W celu wdrożenia aplikacji potrzebna jest wirtualna maszyna Javy oraz serwer MySQL, natomiast użytkownik końcowy aplikacji potrzebuje jedynie urządzenia z dostępem do Internetu z zainstalowaną współczesną i aktualną przeglądarką internetową.

5.1 Wymagania funkcjonalne

- Autoryzacja
 - Rejestracja konta użytkownika.
 - Logowanie do konta użytkownika.
 - Wylogowanie z konta użytkownika.
- Trening i pomiary
 - Listy aktywności
 - Wyświetlanie tabeli list.
 - Dodanie listy aktywności.
 - Edytowanie nazwy listy aktywności.
 - Usuwanie listy aktywności.
 - Aktywności
 - Wyświetlanie tabeli aktywności.
 - Dodawanie aktywności.
 - Edytowanie nazwy aktywności.
 - Usuwanie aktywności.
 - Wpisy (pomiary)
 - Wyświetlanie tabeli wpisów.
 - Sortowanie tabeli wpisów.
 - Dodawanie wpisu.
 - Edycja wpisu.
 - Usuwanie wpisu.
- Wykresy
 - Edycja przedziału czasu.
- Strona główna

- Wyświetlanie wskaźników postępu.

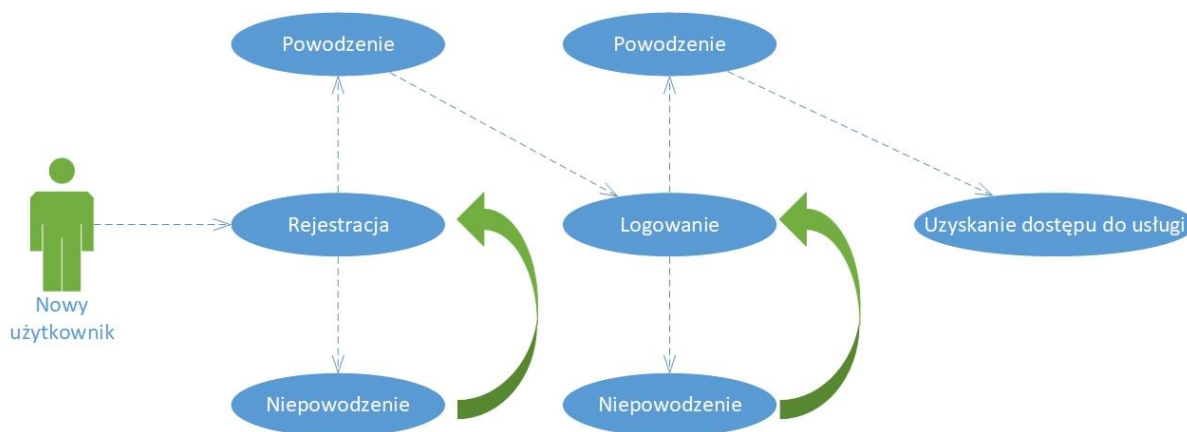
5.2 Wymagania niefunkcjonalne

- Technikalnia
 - Dostęp do Internetu.
 - Dostęp do przeglądarki internetowej.
- Użyteczność
 - Intuicyjny i atrakcyjny graficzny interfejs użytkownika.
- Przenośność
 - Oprogramowanie jest niezależne od systemu operacyjnego.
- Utrzymanie i rozwój
 - Kod źródłowy zgodny z praktykami dobrego programowania.
 - Modułowa architektura.

5.3 Scenariusze przypadków użycia

W tym podrozdziale zostały opisane przypadki użycia służące do dokumentacji funkcjonalności systemu bazujące na diagramach i specyfikacjach przypadków użycia.

5.3.1 Autoryzacja



Rysunek 5.1 Przypadek użycia – Autoryzacja do serwisu. Opracowanie własne.

Przypadek użycia: Autoryzacja w serwisie

Aktor: Użytkownik

Opis: Uzyskanie dostępu do serwisu.

Warunki wstępne: Użytkownik nie posiada konta użytkownika.

Warunki końcowe: Użytkownik posiada konto użytkownika oraz jest zalogowany do systemu.

Rezultat: Użytkownik ma dostęp do funkcji serwisu.

Główny scenariusz:

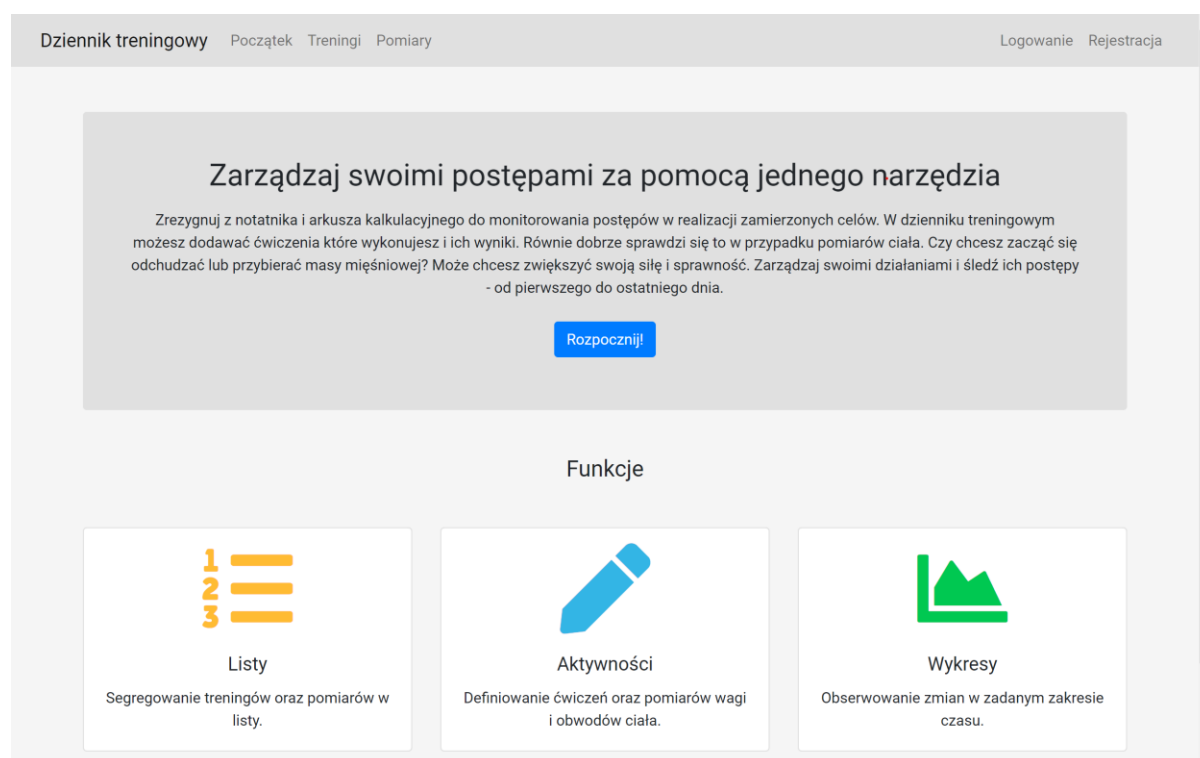
1. Użytkownik rejestruje konto.
2. Użytkownik loguje się na zarejestrowane konto.
3. Użytkownik uzyskuje dostęp do serwisu.

Alternatywny scenariusz:

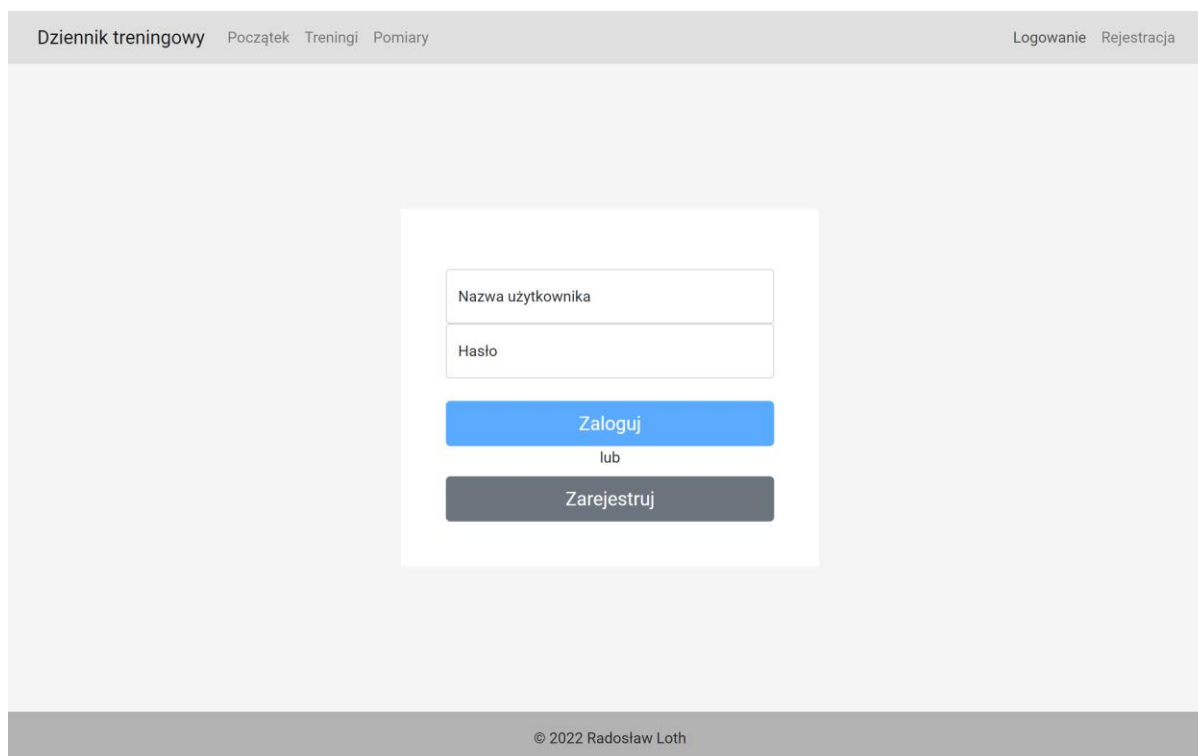
1. Użytkownik nie ma konta i nie otrzymuje dostępu do funkcji aplikacji.

Scenariusze wyjątków:

1. Użytkownik rejestruje konto, które już istnieje w bazie danych.
 - a. Wejście do punktu 1 scenariusza głównego.
2. Użytkownik loguje się nieprawidłowymi poświadczeniami.
 - a. Wejście do punktu 2 scenariusza głównego.



Rysunek 5.2 Widok strony startowej dla użytkowników niezalogowanych. Opracowanie własne.



Rysunek 5.3 Widok panelu logowania. Opracowanie własne.

Użytkownik serwisu uzyskuje dostęp do strony docelowej serwisu (rysunek 5.2) oraz do panelu logowania (rysunek 5.3). Pozostałe podstrony pozostają niedostępne dla niezalogowanych użytkowników. Próba przejścia do podstrony „Początek”, „Treningi”, „Pomiary” na pasku nawigacyjnym oraz przycisk „Rozpocznij!” na środku ekranu odsyła użytkownika do panelu logowania.

```
1 const routes: Routes = [  
2   {path: '', component: LandingPageComponent},  
3   {path: 'trainings', component: ActivitiesListsComponent, canActivate: [AuthGuardService]},  
4   {path: 'measurements', component: ActivitiesListsComponent, canActivate: [AuthGuardService]},  
5   {  
6     path: 'trainings/details',  
7     children: [{path: ':id', component: ActivityDetailsComponent, canActivate: [AuthGuardService]}]  
8   },  
9   {  
10    path: 'measurements/details',  
11    children: [{path: ':id', component: ActivityDetailsComponent, canActivate: [AuthGuardService]}]  
12  },  
13  {path: 'home', component: HomeComponent, canActivate: [AuthGuardService]},  
14  {path: 'login', component: LoginComponent},  
15  {path: 'lists', component: ActivitiesListsComponent, canActivate: [AuthGuardService]},  
16  {path: '**', component: PageNotFoundComponent}  
17 ];
```

Rysunek 5.4 Kod źródłowy warstwy klienta - tablica ścieżek.

Na rysunku 5.4 widoczny jest kod źródłowy aplikacji klienta, gdzie w linii 1 została skonfigurowana tablica ścieżek aplikacji. Ścieżki aplikacji po słowie *path* składają się na adres URL (ang. Uniform Resource Locator) [38] wyświetlany w aplikacji, a następnie po słowie *component* określany jest wyświetlany komponent aplikacji. Słowo *canActivate* pozwala wskazać serwis, który będzie określał czy aktualnie zalogowany użytkownik ma uprawnienia do aktywacji wskazanego

komponentu. Brak słowa *canActivate* oznacza, że komponent dostępny jest dla wszystkich użytkowników.

Dziennik treningowy Początek Treningi Pomiary Logowanie Rejestracja

Adres e-mail
radoslaw@loth.pl

Nazwa użytkownika
Radosław

Hasło

Zarejestruj

lub

Zaloguj

© 2022 Radosław Loth

Rysunek 5.5 Widok formularza rejestracji użytkownika.

Użytkownik wypełnia formularz rejestracyjny, jak zostało to przedstawione na rysunku 5.5. Pola formularza rejestracji podlegają walidacji. Przycisk „Zarejestruj” zostaje aktywowany w przypadku, kiedy dane wprowadzone do formularza będą zgodne z określonymi zasadami walidacji. Aktywny przycisk może zostać wciśnięty przez użytkownika. Po naciśnięciu przycisku zostaje wysłane zapytanie HTTP post do serwera aplikacji.

```
1 2022-05-11 22:46:29.376 DEBUG 784 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy : Securing POST /register
2 2022-05-11 22:46:29.376 DEBUG 784 --- [nio-8080-exec-2] s.s.w.c.SecurityContextPersistenceFilter : Set SecurityContextHolder to empty
  SecurityContext
3 2022-05-11 22:46:29.377 DEBUG 784 --- [nio-8080-exec-2] o.s.s.w.a.AnonymousAuthenticationFilter : Set SecurityContextHolder to anonymous
  SecurityContext
4 2022-05-11 22:46:29.378 DEBUG 784 --- [nio-8080-exec-2] o.s.s.w.a.i.FilterSecurityInterceptor : Authorized filter invocation [POST
  /register] with attributes [permitAll]
5 2022-05-11 22:46:29.378 DEBUG 784 --- [nio-8080-exec-2] o.s.security.web.FilterChainProxy : Secured POST /register
6 Hibernate: select user0_.id as idl_4, user0_.email as email2_4, user0_.enabled as enabled3_4, user0_.first_name as first_na4_4,
  user0_.last_name as last_nam5_4, user0_.password as password6_4, user0_.username as username7_4 from users user0_ where user0_.email=?
7 Hibernate: select user0_.id as idl_4, user0_.email as email2_4, user0_.enabled as enabled3_4, user0_.first_name as first_na4_4,
  user0_.last_name as last_nam5_4, user0_.password as password6_4, user0_.username as username7_4 from users user0_ where
  user0_.username=?
8 Hibernate: insert into users (email, enabled, first_name, last_name, password, username) values (?, ?, ?, ?, ?, ?)
9 Hibernate: insert into authorities (authority, username) values (?, ?)
10 2022-05-11 22:46:29.545 DEBUG 784 --- [nio-8080-exec-2] s.s.w.c.SecurityContextPersistenceFilter : Cleared SecurityContextHolder to
  complete request
```

Rysunek 5.6 Rejestr zdarzeń aplikacji serwerowej – rejestracja użytkownika.

Na rysunku 5.6 widoczne są zdarzenia aplikacji serwerowej. W linii 1 widoczne jest, że aplikacja odbiera zapytanie na punkcie końcowym „/register”. Linie od 1 do 5 związane są z autoryzacją i modulem Spring Security. W linii 6 widoczne są zapytania SQL wygenerowane poprzez mapowanie obiektowo relacyjne za pomocą platformy programistycznej Hibernate. W zapytaniach sprawdzane jest, czy w bazie danych istnieje już taki wpis, jaki użytkownik chce

zarejestrować. W pierwszym zapytaniu sprawdzane jest, czy nie występuje konflikt adresów email, a następnie nazwy użytkownika, ponieważ te informacje muszą być unikatowe dla każdego użytkownika. Po pomyślnym sprawdzeniu danych i upewnieniu się, że są one unikalne, wówczas tworzony jest nowy użytkownik oraz ustawiane są jego uprawnienia. Hasło użytkownika zapisywane jest w bazie danych w postaci kryptograficznego klucza stworzonego poprzez funkcję haszującą Bcrypt. W aplikacji została wykorzystana standardowa implementacja enkodera Bcrypt biblioteki Spring Security. Zapewnia to bezpieczeństwo w przypadku wycieku bazy danych, ponieważ nie ma możliwości odzyskania oryginalnego hasła na podstawie wartości funkcji haszującej.

```
1 mysql> SELECT * FROM progresstracker.users;
2 +-----+-----+-----+-----+-----+-----+-----+
3 | id | email | first_name | last_name | username | password | enabled |
4 +-----+-----+-----+-----+-----+-----+-----+
5 | 1 | radoslaw@loth.pl | NULL | NULL | Radoslaw | {bcrypt}$2a$10$odnJC0drueCNN0lobVovjuyqulj4hxAk9BfW04D/V1lZ8gNHAdI32 | 1 |
6 +-----+-----+-----+-----+-----+-----+-----+
7 1 row in set (0.00 sec)
```

Rysunek 5.7 Baza danych – tabela users.

Na rysunku 5.7 widoczna jest zawartość tabeli users. W kolumnie password można zobaczyć hasło utworzone funkcją skrótu kryptograficznego Bcrypt, które jest zapisane w bazie danych według schematu [39]:

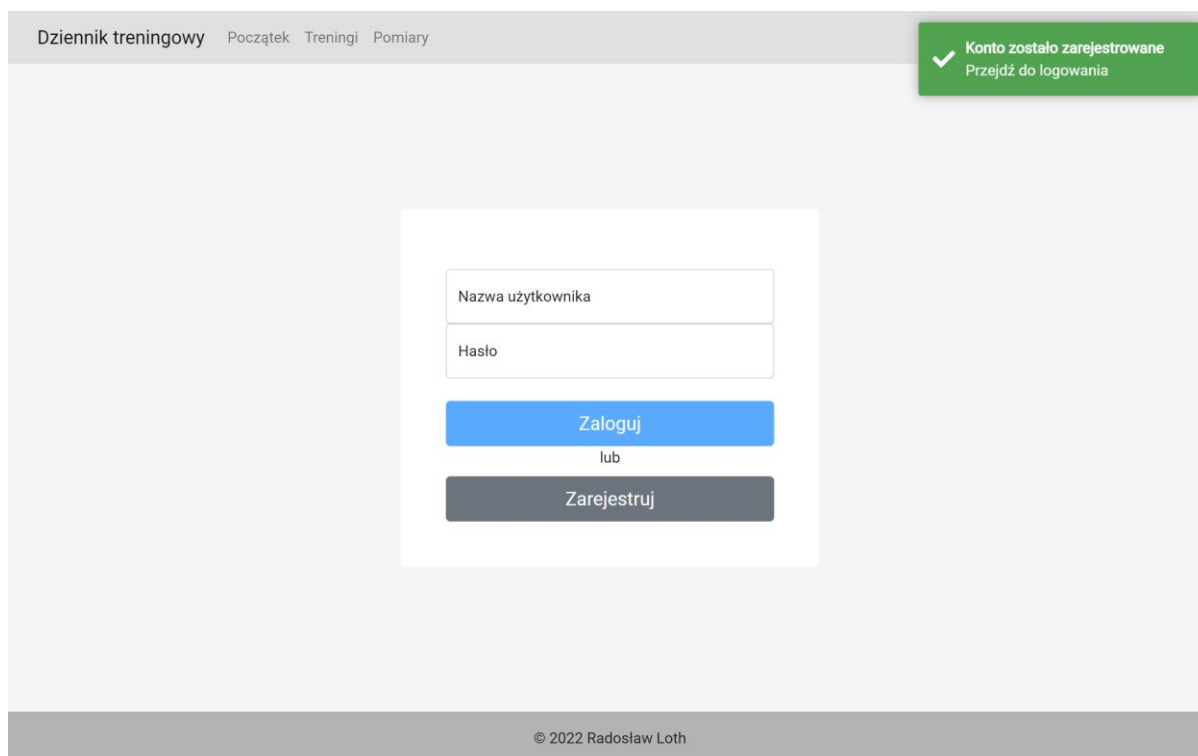
$$\{\alpha\} \$ < \beta > \$ < \gamma > \$ < \delta > < \varepsilon >$$

Schemat 5.1 Schemat hasła enkodowanego BCrypt

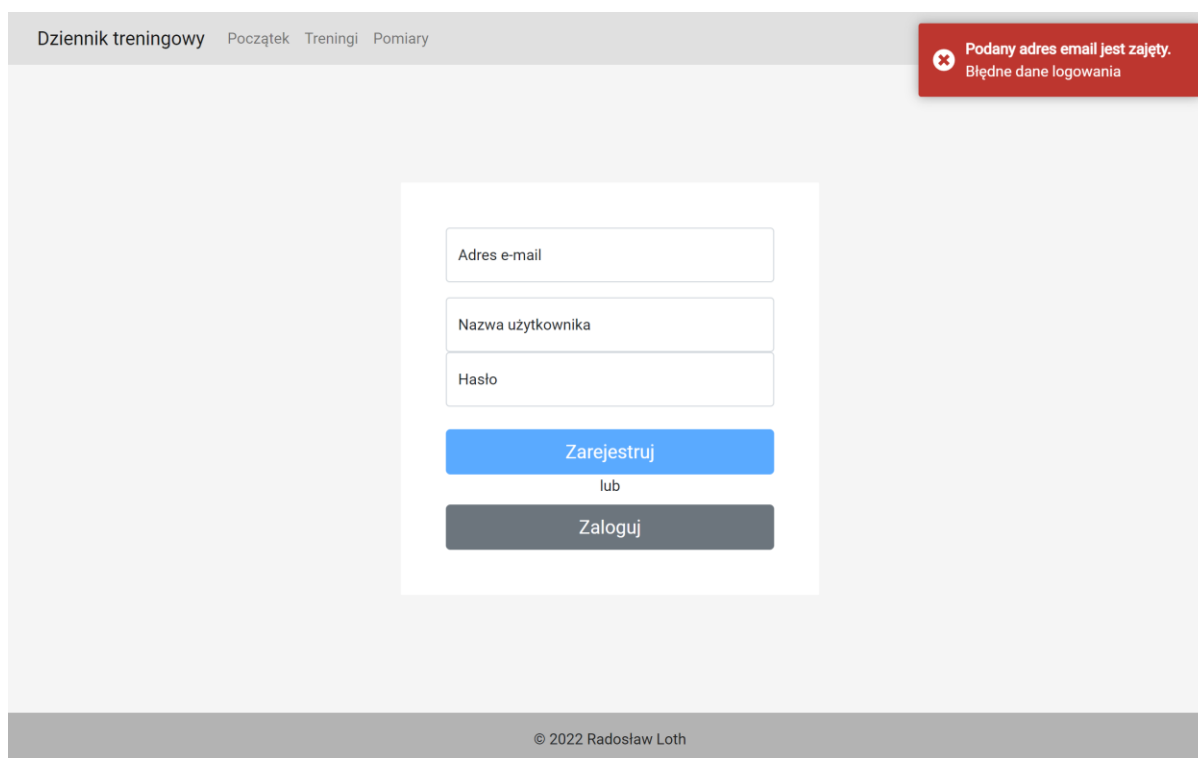
gdzie:

- α – Określenie funkcji skrótu,
- β – wersja algorytmu kryptograficznego,
- γ – współczynnik rozgałęzienia rekurencji,
- δ – sól algorytmu, czyli losowa 22 znakowa wartość powiększająca ciąg zaburzający podczas obliczania funkcji skrótu,
- ε – wartość funkcji kryptograficznej.

Aplikacja klienta dostaje odpowiedź od serwera, a następnie wyświetla wyskakujące okienko z informacją dla użytkownika.

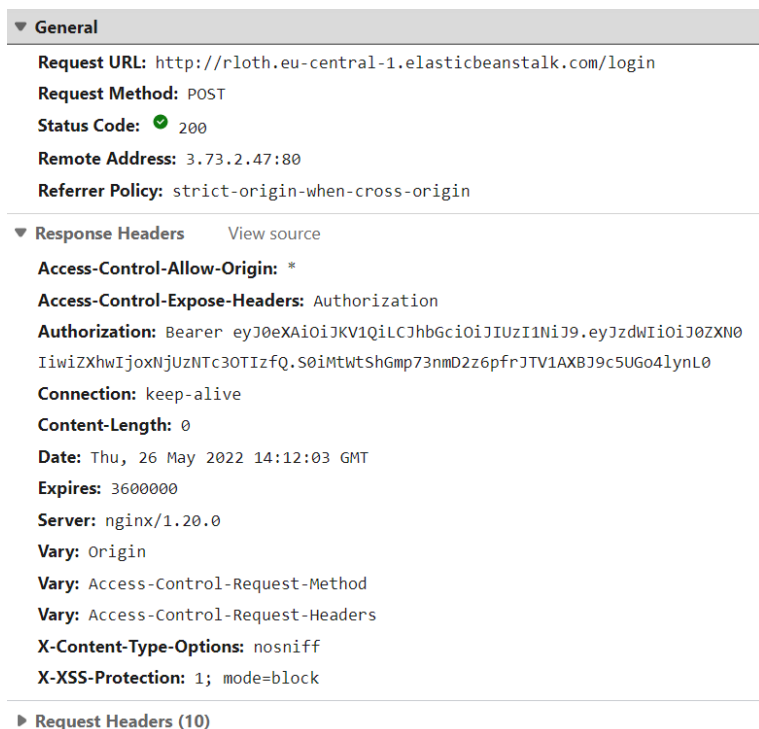


Rysunek 5.8 Widok panelu logowania – zarejestrowanie użytkownika.



Rysunek 5.9 Widok panelu rejestracji – błąd podczas rejestracji użytkownika.

Podczas udanej próby rejestracji konta użytkownika, użytkownik aplikacji zostaje przekierowany do strony logowania (rysunek 5.8). Przy nieudanej próbie rejestracji, użytkownik pozostaje na stronie rejestracji i zostaje wyświetlony komunikat o błędzie (rysunek 5.9).



Rysunek 5.10 Pomyślnie logowanie - odpowiedź serwera. Źródło: Chrome DevTools.

Użytkownik loguje się do serwisu. Dane dotyczące logowania zostają wysłane do warstwy aplikacji. W przypadku kiedy dane logowania są poprawne, wówczas serwer aplikacji zwraca kod odpowiedzi HTTP „200 OK” i zawarte w nagłówku *Authorization*, i *Expires* informacje takie jak token JWT oraz czas do utraty jego ważności (rysunek 5.10). W przypadku kiedy dane logowania nie są poprawne serwer zwraca kod odpowiedzi „401 Unauthorized”, a następnie zostaje wyświetlone wyskakujące okienko z powiadomieniem o błędzie.

```

1  onSubmit(form: NgForm) {
2    if (!form.valid) {
3      this.showError('Spróbuj ponownie', 'Błędny formularz')
4      return;
5    }
6    this.loginCredentials.password = form.value.password;
7    this.loginCredentials.username = form.value.username;
8    this.loginCredentials.email = form.value.email;
9    let authObs: Observable<HttpResponse<Object>>;
10   this.isLoading = true;
11   this.isLoginMode ? authObs = this.authService.login(this.loginCredentials)
12     : authObs = this.authService.signUp(this.loginCredentials);
13   authObs.subscribe(
14     resData => {
15       if (this.isLoginMode) {
16         this.isLoading = false;
17         this.showSuccess('Gratulacje', 'Zalogowano pomyślnie');
18         this.router.navigate(['/home']);
19       } else {
20         this.isLoading = false;
21         this.authService.toggleLoginMode(true);
22         this.router.navigate(['/login']);
23         this.showSuccess('Przejdź do logowania', 'Konto zostało zarejestrowane');
24       }
25     },
26     errorMessage => {
27       this.showError('Błędne dane logowania', errorMessage);
28       this.isLoading = false;
29     }
30   );
31   form.reset();
32 }

```

Rysunek 5.11 Kod źródłowy warstwy klienta – składanie formularza logowania i rejestracji.

```

1  login(loginCredentials: LoginCredentials) {
2    return this.http.post(`${environment.APIEndpoint}/login`, loginCredentials, {observe: 'response'})
3      .pipe(catchError(this.handleError), tap(resData => {
4        this.handleAuthentication(
5          loginCredentials.username,
6          resData.headers.get('Authorization'),
7          resData.headers.get('Expires'))
8      }));
9  }

```

Rysunek 5.12 Kod źródłowy warstwy klienta – logowanie.

```

1  handleAuthentication(username: string, token: string, tokenExpirationTime: string) {
2    const user = new User(username, token, new Date(new Date().getTime() + Number(tokenExpirationTime)));
3    this.user.next(user);
4    this.autoLogout(Number(tokenExpirationTime));
5    localStorage.setItem('user', JSON.stringify(user));
6  }

```

Rysunek 5.13 Kod źródłowy warstwy klienta - przechowywanie użytkownika.

```

1  autoLogout(expirationDuration: number) {
2    this.tokenExpirationTimer = setTimeout(() => {
3      this.logout();
4    }, expirationDuration)
5  }

```

Rysunek 5.14 Kod źródłowy warstwy klienta - wygaśnięcie ważności tokena.

```

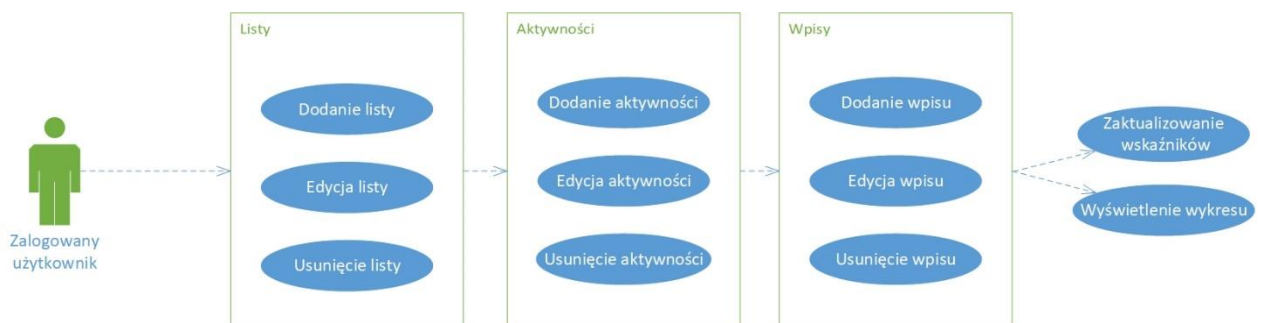
1  logout() {
2      this.user.next(null);
3      localStorage.removeItem('user');
4      this.router.navigate(['/login']);
5      if (this.tokenExpirationTimer) {
6          clearTimeout(this.tokenExpirationTimer);
7      }
8      this.tokenExpirationTimer = null;
9  }

```

Rysunek 5.15 Kod źródłowy warstwy klienta - wylogowanie.

Na rysunku 5.11 widoczny jest kod źródłowy metody *onSubmit* wykonywany podczas składania formularza logowania i rejestracji. W linii 11 wywoływana jest metoda *login* widoczna na rysunku 5.12. Metoda *login* w linii 2 wykonuje żądanie HTTP post do warstwy aplikacji, gdzie przekazywane są dane logowania. W linii 4 po otrzymaniu odpowiedzi z serwera aplikacji, nagłówki odpowiedzi HTTP takie jak *Authorization* oraz *Expires* wraz z nazwą użytkownika zostają przekazane do metody *handleAuthentication* widocznej na rysunku 5.13. W metodzie *handleAuthentication* w linii 2 tworzony jest obiekt użytkownika, składający się z nazwy użytkownika, tokena i czasu jego ważności. Następnie w linii 5 utworzony obiekt zapisywany jest do pamięci przeglądarki użytkownika aplikacji. Dodatkowo w linii 4 metoda *handleAuthentication* wywołuje metodę *autoLogout* (rysunek 5.14), gdzie przekazywany jest czas do wygaśnięcia tokena. Metoda *autoLogout* w linii 2 wywołuje funkcję języka JavaScript – *setTimeout*. W linii 3 wewnątrz funkcji *setTimeout* po upływie zadanego czasu, wywoływana jest metoda *logout* (rysunek 5.15). Metoda *logout* w linii 3 usuwa użytkownika z pamięci przeglądarki oraz w linii 4 przekierowuje na stronę logowania.

5.3.2 Obsługa dziennika



Rysunek 5.16 Przypadek użycia - korzystanie z dziennika. Opracowanie własne.

Przypadek użycia: Korzystanie z serwisu

Aktor: Użytkownik

Opis: Zarządzanie dziennikiem przez użytkownika.

Warunki wstępne: Użytkownik jest zalogowany.

Warunki końcowe: Użytkownik posiada aktualne dane w serwisie.

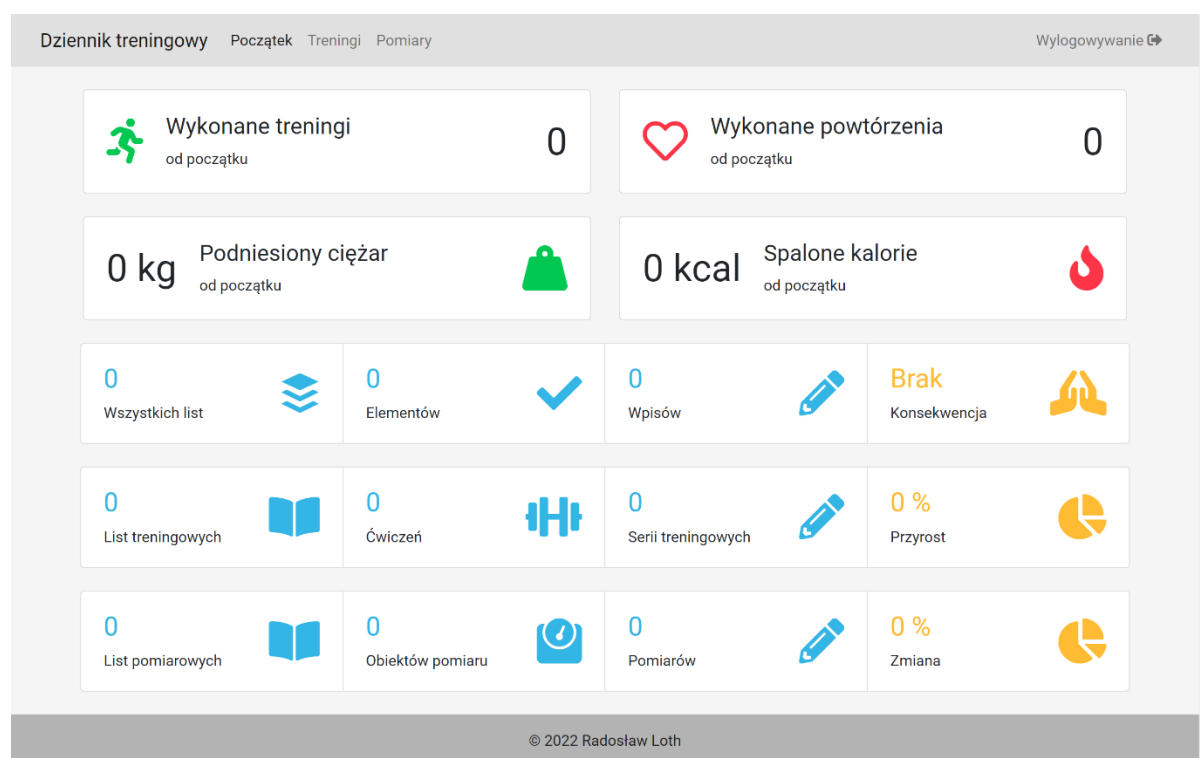
Rezultat: Użytkownik korzysta ze wskaźników na stronie głównej oraz wykresów opartych o dostarczone dane.

Główny scenariusz:

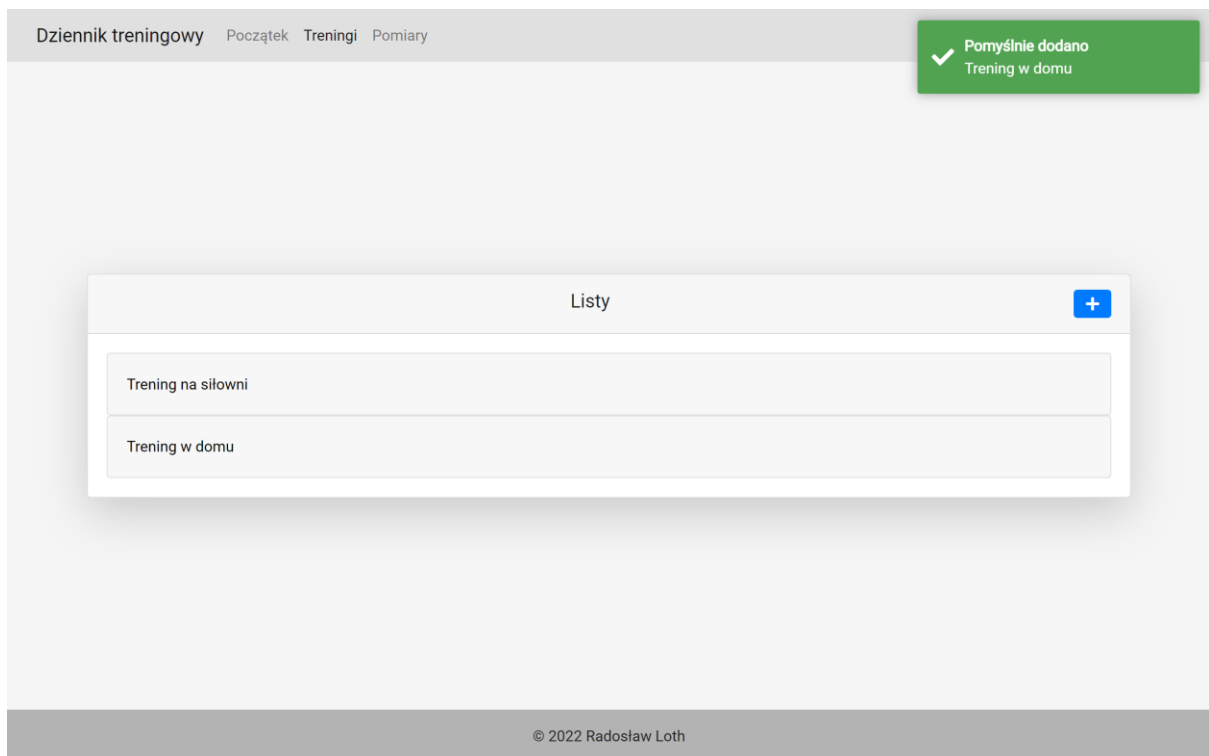
1. Użytkownik dodaje dane do serwisu.
2. Użytkownik korzysta ze wskaźników na stronie głównej oraz z wykresów.

Alternatywny scenariusz:

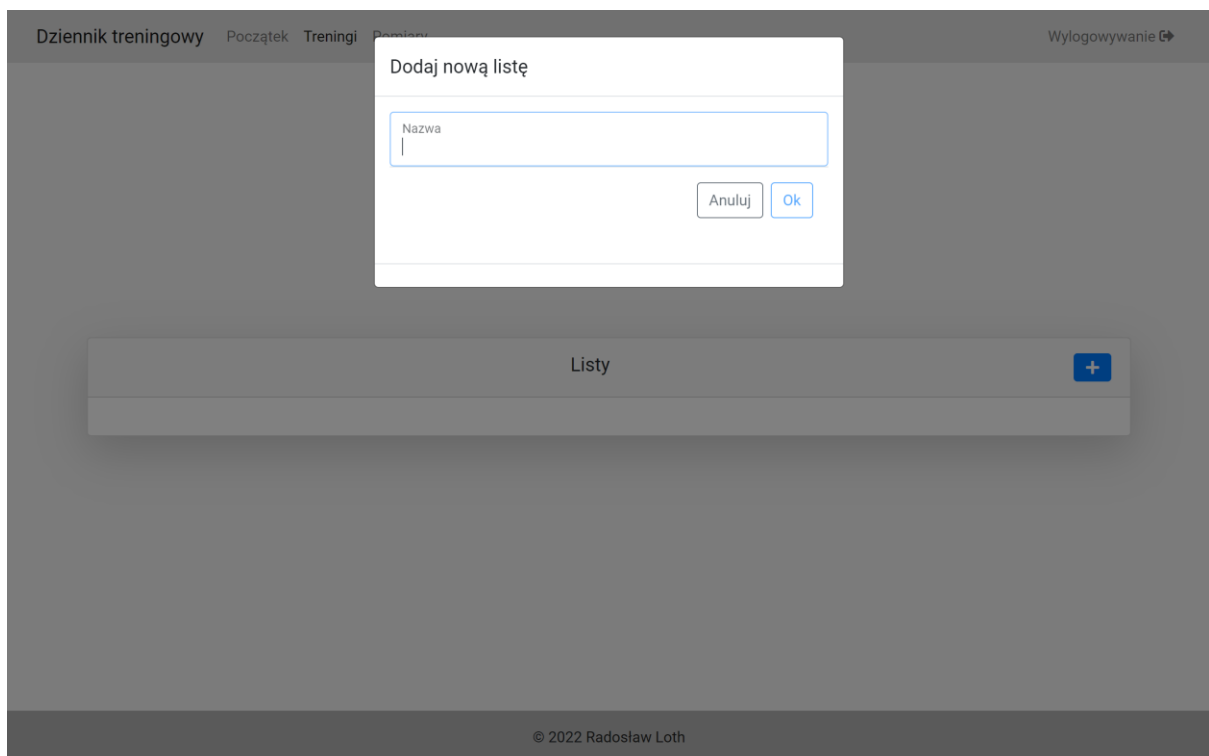
1. Użytkownik usuwa lub edytuje dane w serwisie.
2. Użytkownik korzysta ze wskaźników na stronie głównej oraz z wykresów.



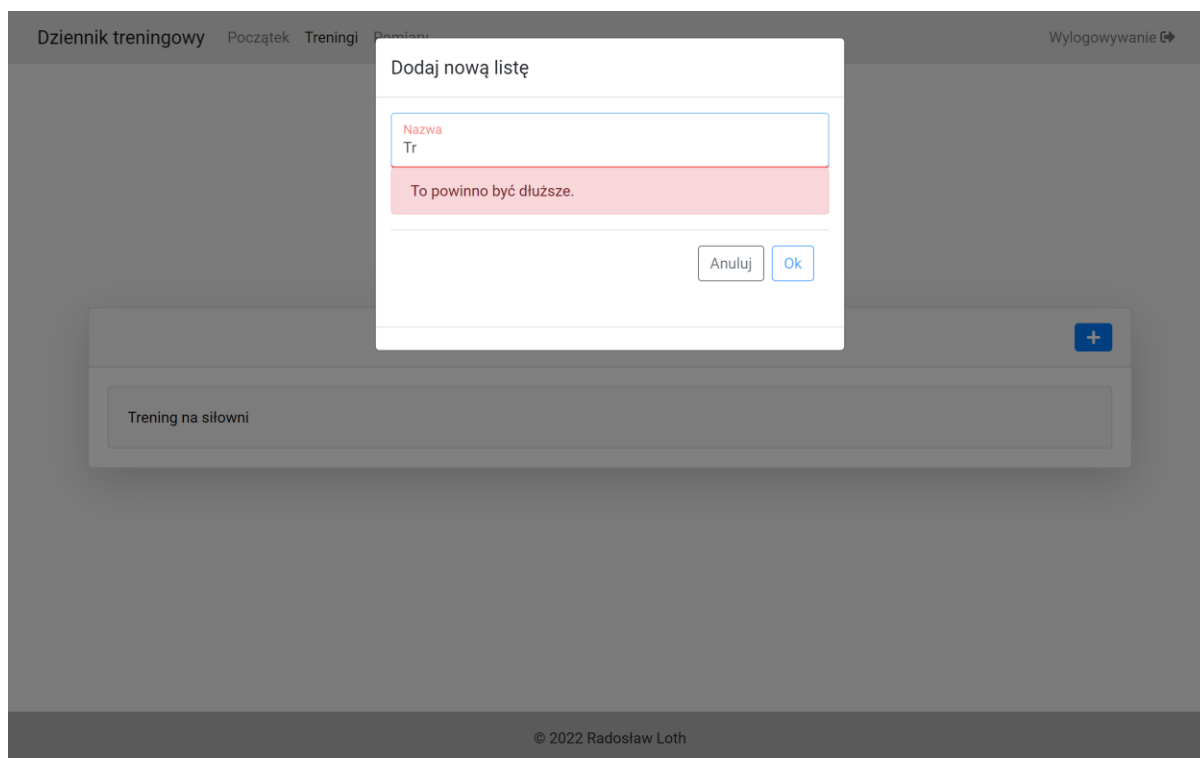
Rysunek 5.17 Widok strony głównej.



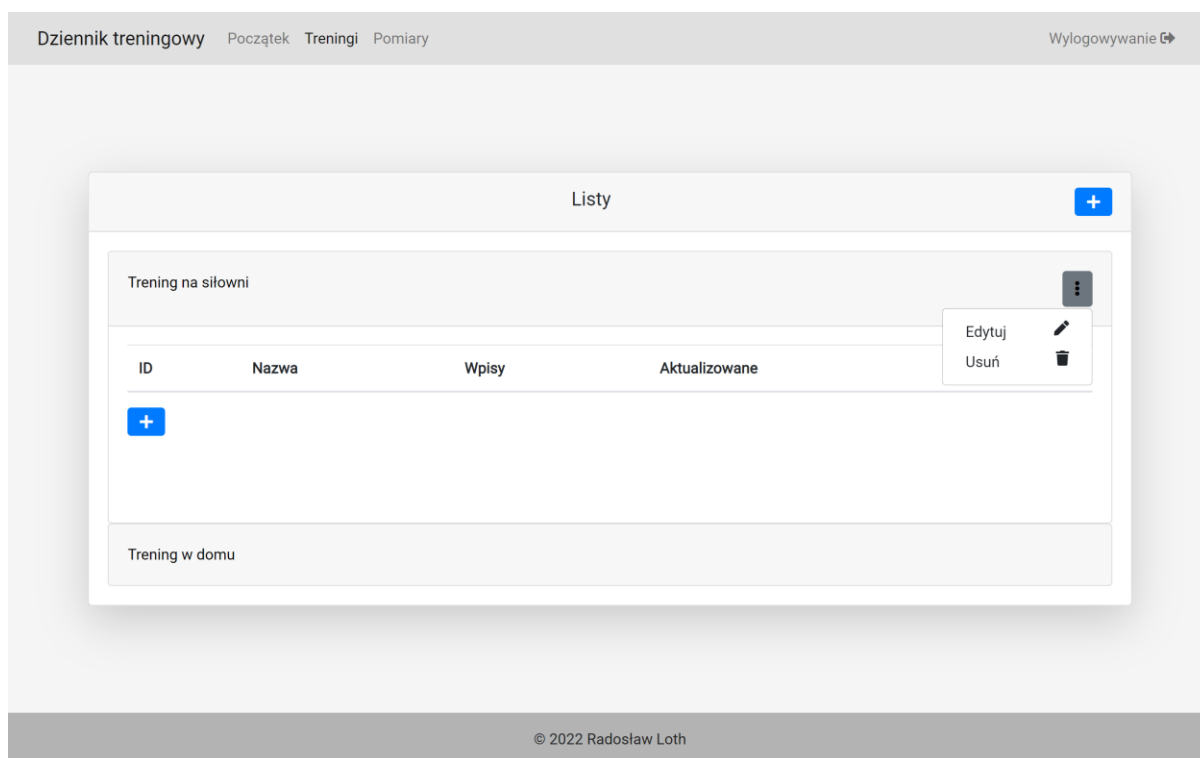
Rysunek 5.18 Widok list treningowych.



Rysunek 5.19 Widok formularza dodawania listy.



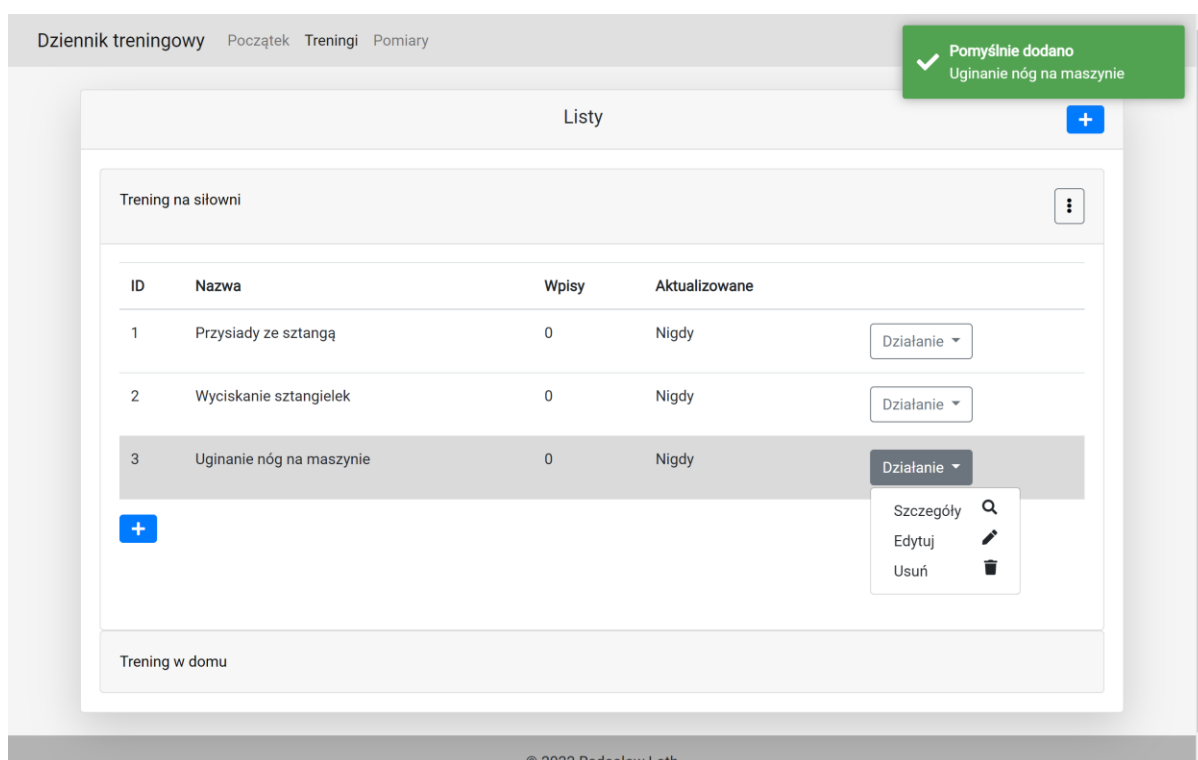
Rysunek 5.20 Widok formularza dodawania listy - walidacja pola.



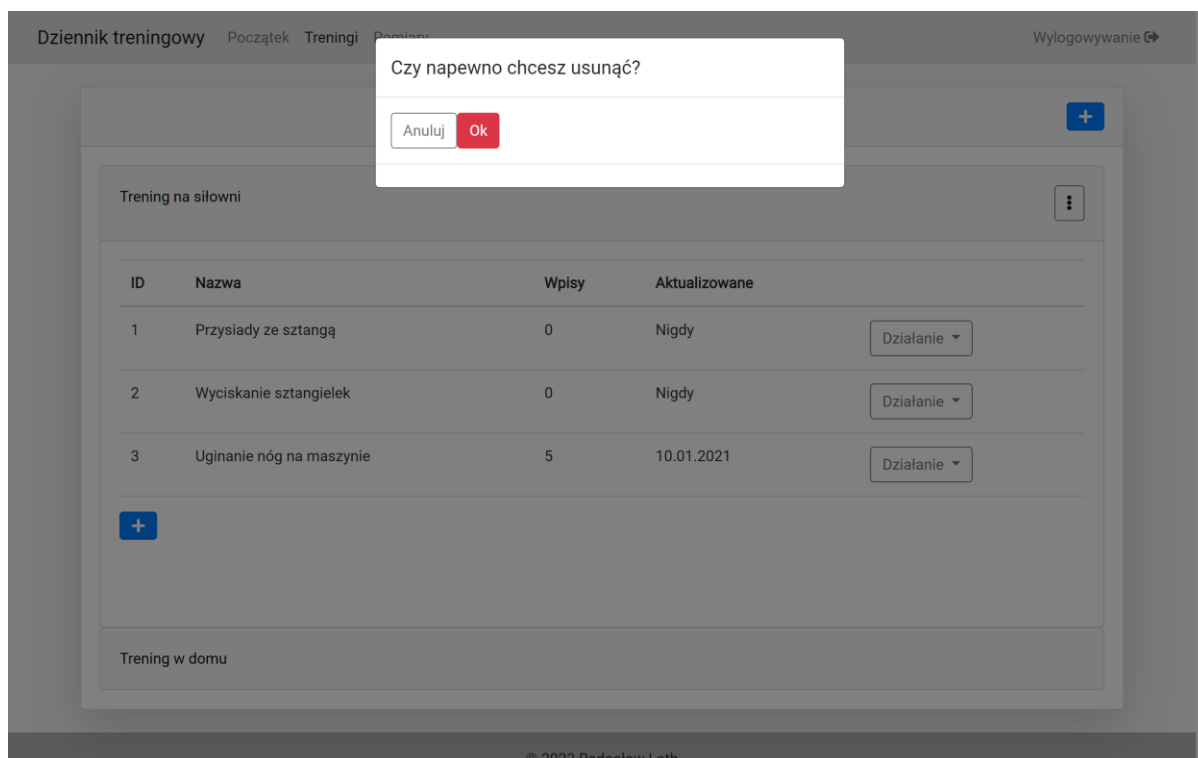
Rysunek 5.21 Widok list – lista rozwijana opcji.

Użytkownik po zalogowaniu przekierowany jest na stronę główną (rysunek 5.17). Na stronie głównej znajdują się wskaźniki informacyjne, które są obliczane na podstawie dodawanych przez użytkownika danych dotyczących treningów i pomiarów.

Na pasku nawigacyjnym u góry ekranu widoczne są zakładki „Treningi” oraz „Pomiary”. Po przejściu w zakładkę „Treningi” pojawia się widok list treningowych (rysunek 5.18). Na ekranie widoczny jest niebieski przycisk z symbolem „+” po naciśnięciu, którego użytkownik uzyskuje dostęp do formularza dodawania listy (rysunek 5.19). Wysłanie formularza zostało zabezpieczone poprzez walidację pola „Nazwa” (rysunek 5.20). Po rozwinięciu listy pojawia się przycisk, poprzez który możliwe jest edytowanie lub usuwanie istniejących list (rysunek 5.21). Po przejściu w zakładkę „Pomiary” sposób działania aplikacji jest bliźniaczo podobny do przedstawionego na przykładzie zakładki „Treningi”.

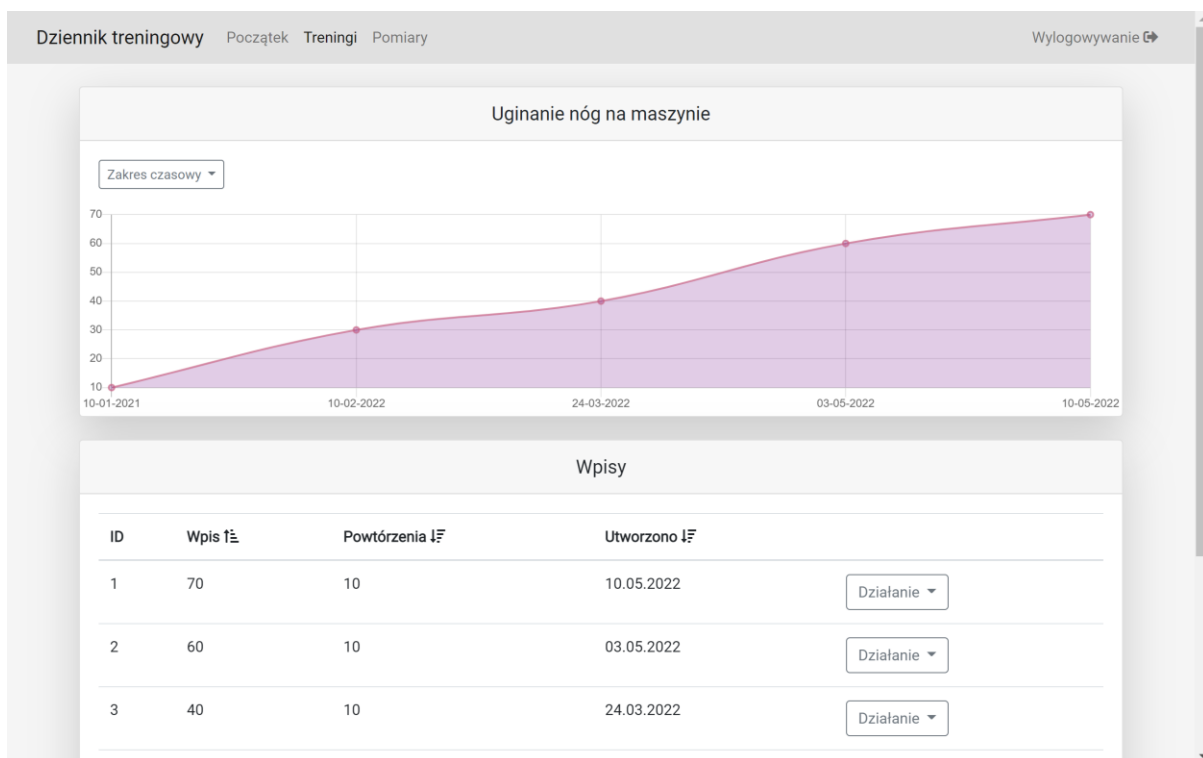


Rysunek 5.22 Widok aktywności – lista rozwijana opcji.

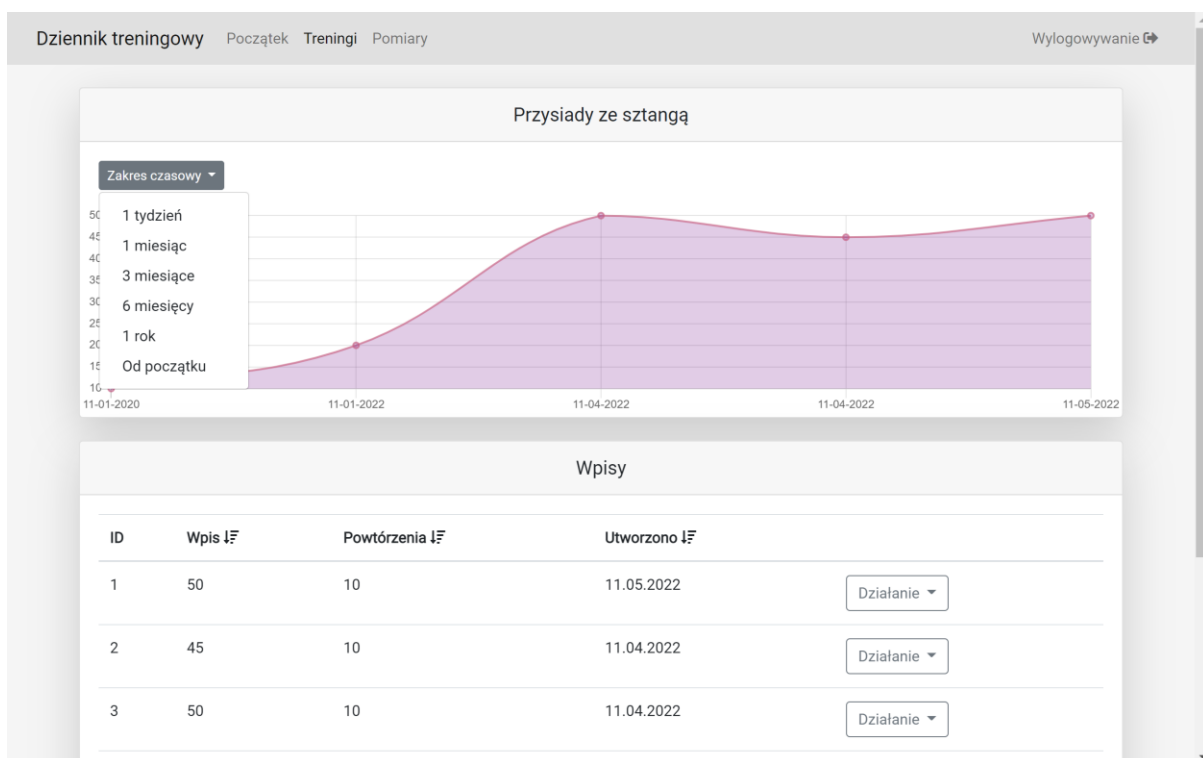


Rysunek 5.23 Widok okna modalnego potwierdzające usunięcie.

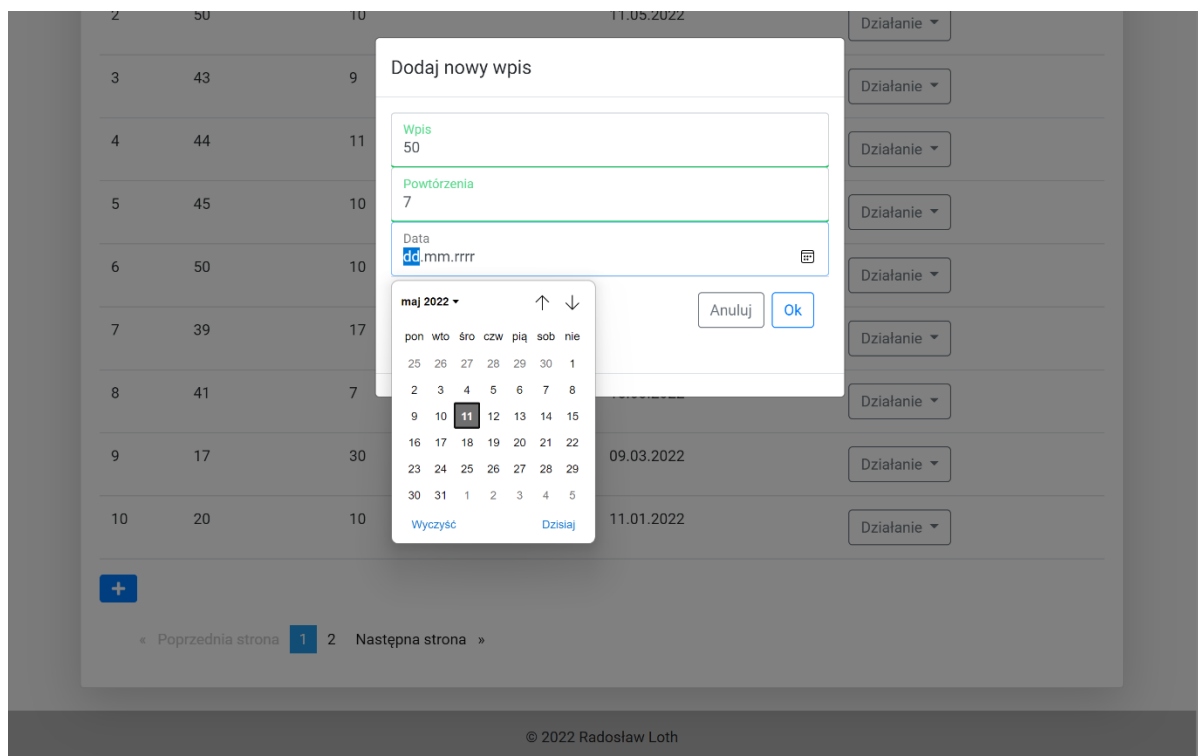
Użytkownik definiuje aktywności należące do wcześniej zdefiniowanej listy. Dodanie aktywności następuje za pomocą niebieskiego przycisku z symbolem „+” umieszczonego pod ostatnim elementem listy. Wyświetlenie formularza oraz jego walidacja jest analogiczna do dodawania list. Przy każdym elemencie wyświetlony jest przycisk z listą rozwijaną, udostępniający możliwość przejścia w szczegóły elementu, edytowanie elementu oraz jego usunięcie (rysunek 5.22). Przejście w szczegóły elementu jest również możliwe poprzez kliknięcie gdziekolwiek w obrębie podświetlonego pola, czyli wiersza tabeli odpowiadającemu wybranemu elementowi. Wybranie opcji „Szczegóły” przenosi użytkownika na podstronę, która zostanie omówiona w dalszej części pracy. Wybranie opcji „Edytuj” wyświetla okno modalne, analogiczne jak przy dodawaniu nowego elementu i zapewnia możliwość zmiany nazwy elementu. Wybranie opcji „Usuń” wywołuje okno modalne z dodatkowym potwierdzeniem usunięcia elementu (rysunek 5.23).



Rysunek 5.24 Widok szczegółów aktywności.



Rysunek 5.25 Widok szczegółów aktywności - lista rozwijana „Zakres czasowy”.



Rysunek 5.26 Widok formularza dodawania wpisu.

Użytkownik przechodzi do szczegółów związanych z wybranym elementem. Zostaje wyświetlony ekran składający się z kilku komponentów (rysunek 5.24). Jednym z nich jest komponent wyświetlający wykres na podstawie danych dodanych w komponencie poniżej. Komponent umożliwia dostosowanie zakresu wyświetlanych dat na wykresie zgodnie z opcjami wyświetlonymi na liście rozwijanej pod przyciskiem „Zakres czasowy” znajdującym się w lewym górnym rogu komponentu (rysunek 5.25). W komponencie poniżej wyświetlana jest tabela z uzupełnionymi przez użytkownika danymi. Tabela umożliwia sortowanie wpisów rosnąco oraz malejąco po naciśnięciu na etykietę odpowiadającą kolumnie, względem której tabela powinna być posortowana. Domyślnie dane wyświetlane są w kolejności chronologicznej. Dodawanie nowych wpisów odbywa się poprzez naciśnięcie niebieskiego przycisku, na dole tabeli, oznaczonego symbolem „+”. Po naciśnięciu przycisku wyświetla się okno modalne z formularzem dodawania wpisu (rysunek 5.26). Pola formularza objęte są walidacją, a zielony kolor wskazuje na poprawność wprowadzonych danych. Na dole ekranu widoczne są przyciski obsługujące stronicowanie wyników.

6 Podsumowanie

Celem pracy było wykorzystanie oraz przedstawienie przodujących na rynku rozwiązań i platform programistycznych na przykładzie przedstawionej aplikacji. Cel pracy został osiągnięty

poprzez implementację pełnego stosu technologicznego, składającego się na oprogramowanie złożone z serwera aplikacji, w tym serwera www, jak również bazy danych, narzędzi do zarządzania oraz aplikacji klienta. Do implementacji wyżej wymienionego stosu została wykorzystana wiedza dotycząca wielu technologii takich jak języki znaczników HTML i CSS, opisujące wygląd przechowywanej treści, języki programowania JavaScript i TypeScript, które odpowiadają za interaktywność w aplikacji klienta, język programowania Java potrzebny do stworzenia logiki działania aplikacji serwerowej, baza danych SQL wykorzystana do przechowywania danych, platformy programistyczne Spring i Angular zapewniające szkielet do budowy aplikacji, protokół HTTP służący do komunikacji między warstwami oraz zasady projektowania REST API. Struktura aplikacji, mechanizmy jej działania oraz zestawy komponentów, dostarczone przez wykorzystane biblioteki i platformy programistyczne, pozwoliły na efektywne wytworzenie w pełni funkcjonalnej aplikacji, przy możliwie najmniejszym obciążeniu czasowym, a przy jednoczesnej maksymalizacji jakości wytworzonego oprogramowania.

7 Wnioski

Wykonana praca przedstawia aktualne możliwości dostępnych na rynku bibliotek i platform programistycznych. W pracy zostało przedstawione wykorzystanie wspomnianych technologii na przykładzie aplikacji dziennika treningowego. Udowadnia to, że niskim nakładem pracy możliwe jest zbudowanie funkcjonalnej aplikacji internetowej, opierającej się na wykorzystaniu dostępnych na rynku rozwiązań w postaci bibliotek i platform programistycznych. Z wykorzystaniem niewielkiego nakładu pracy można przystosować tę aplikację do produkowania i konsumowania danych pochodzących z innych źródeł. Możliwe jest wykorzystanie API aplikacji do pobierania danych innych źródeł, przykładowo z czujników umieszczonych w urządzeniach ubieralnych użytkownika, takich jak buty lub opaski sportowe. Z pomocą takich czujników aplikacja może być rozwijana o funkcje, które w dokładniejszy sposób będą umożliwiały oszacowanie wydatku energetycznego użytkownika. Dalsza rozbudowa pracy pozwala na stworzenie oprogramowania, które w zautomatyzowany sposób mogłoby wpływać na zdrowie ludzi, przykładowo wspomagając ich w leczeniu otyłości. W tym celu należałoby pobierać dane wejściowe takie jak puls, ciśnienie, kroki, natlenienie krwi, waga i aktywność. Na podstawie tak wielu danych dostarczanych przez urządzenia ubieralne i inne urządzenia Internetu rzeczy możliwe byłoby określanie bilansu energetycznego użytkownika i jego kondycji zdrowotnej. Wówczas program komputerowy dawałby wskazówki użytkownikowi na temat tego co powinien robić, aby zachować postęp w zamierzonych celach zdrowotnych lub sylwetkowych. W podobny sposób w jaki została rozbudowana aplikacja

dziennika treningowego, możliwe byłoby rozwijanie aplikacji z zastosowaniem przemysłowym. W tym celu API aplikacji powinno konsumować dane z czujników przemysłowych, a warstwa prezentacji aplikacji powinna zostać dostosowana do konkretnego zastosowania.

8 Literatura

- [1] *The State of Mobile Internet Connectivity 2021*,
GSM Association 2021,
<https://www.gsma.com/r/wp-content/uploads/2021/09/The-State-of-Mobile-Internet-Connectivity-Report-2021.pdf>,
Dostęp z dnia: 16.05.2021
- [2] *Stack Overflow Annual Developer Survey*,
Stackoverflow 2021,
<https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies>, Dostęp z dnia: 16.05.2021
- [3] *Czym jest architektura trójwarstwowa*,
IBM 10.2020,
<https://www.ibm.com/pl-pl/cloud/learn/three-tier-architecture>,
Dostęp z dnia: 16.05.2022
- [4] *SPA (Single-page application)*,
Mozilla Developer Center 10.2021,
<https://developer.mozilla.org/en-US/docs/Glossary/SPA>,
Dostęp z dnia: 16.05.2022
- [5] *The Single Page Interface Manifesto*,
Jose Maria Arranz Santamaria 08.2015,
http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php,
Dostęp z dnia: 16.05.2022
- [6] *HTML*,
Web Hypertext Application Technology Working Group 08.2013,
<https://github.com/whatwg/html>,
Dostęp z dnia: 16.05.2022
- [7] *HTML Living Standard*,
Web Hypertext Application Technology Working Group 05.2022,

- <https://html.spec.whatwg.org/>,
Dostęp z dnia: 16.05.2022
- [8] *HTML & CSS*,
World Wide Web Consortium 2016,
<https://www.w3.org/standards/webdesign/htmlcss>,
Dostęp z dnia: 16.05.2022
- [9] *CSS Working Group Editor Drafts*,
World Wide Web Consortium 04.2013,
<https://github.com/w3c/csswg-drafts>,
Dostęp z dnia: 16.05.2022
- [10] *Cascading Style Sheets (CSS) Requirements for RFCs*,
H. Flanagan 12.2016,
<https://datatracker.ietf.org/doc/html/rfc7993>,
Dostęp z dnia: 16.05.2022
- [11] *CSS Snapshot 2021*,
World Wide Web Consortium 12.2021,
<https://www.w3.org/TR/css-2021/>,
Dostęp z dnia: 16.05.2022
- [12] *TypeScript*,
Microsoft,
<https://github.com/microsoft/TypeScript>,
Dostęp z dnia: 16.05.2022
- [13] *TypeScript Documentation*,
Microsoft 2012,
<https://www.typescriptlang.org/docs/>,
Dostęp z dnia: 16.05.2022
- [14] *Angular*,
Angular 09.2014,
<https://github.com/angular>,
Dostęp z dnia: 16.05.2022
- [15] *Angular Documentation*,
Google 2010,
<https://devdocs.io/angular~12/>,
Dostęp z dnia: 16.05.2022

- [16] *Bootstrap*,
Components,
<https://getbootstrap.com/docs/5.0/customize/components/>,
Dostęp z dnia: 16.05.2022
- [17] *Bootstrap*
Bootstrap 07.2011,
<https://github.com/twbs/bootstrap>,
Dostęp z dnia: 16.05.2022
- [18] *Ng-bootstrap*,
ng-bootstrap 09.2015,
<https://github.com/ng-bootstrap/ng-bootstrap>,
Dostęp z dnia: 16.05.2022
- [19] *MDBBootstrap*,
MDBBootstrap 07.2016,
<https://github.com/mdbootstrap>,
Dostęp z dnia: 16.05.2022
- [20] *FontAwesome*,
Font-Awesome 02.2012,
<https://github.com/FortAwesome/Font-Awesome>,
Dostęp z dnia: 16.05.2022
- [21] *Ngx-toastr*,
scttcper 07.2016,
<https://github.com/scttcper/ngx-toastr>,
Dostęp z dnia: 16.05.2022
- [22] *Standard API*,
Ministerstwo Cyfryzacji 05.2020,
<https://dane.gov.pl/media/ckeditor/2020/05/29/standard-api.pdf>,
Dostęp z dnia: 16.05.2022
- [23] *Interfejsy API REST*,
IBM Cloud Education 04.2021,
<https://www.ibm.com/pl-pl/cloud/learn/rest-apis>,
Dostęp z dnia: 16.05.2022
- [24] *JDK*,
OpenJDK 09.2018,

- <https://github.com/openjdk/jdk>,
Dostęp z dnia: 16.05.2022
- [25] *Spring Framework*,
Spring-projects 12.2010,
<https://github.com/spring-projects/spring-framework>,
Dostęp z dnia: 16.05.2022
- [26] *Spring Boot*,
Spring-projects 10.2012,
<https://github.com/spring-projects/spring-boot>,
Dostęp z dnia: 16.05.2022
- [27] *JSON Web Token (JWT)*,
M. Jones, Microsoft, J. Bradley, Ping Identity, N. Sakimura, NRI 03.2016,
<https://datatracker.ietf.org/doc/html/rfc7519>,
Dostęp z dnia: 16.05.2022
- [28] *Eksblowfish Algorithm*,
N. Provos, D. Mazieres 04.1999,
https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/provos/provos_html/node4.html,
Dostęp z dnia: 16.05.2022
- [29] *A Future-Adaptable Password Scheme*,
N. Provos, D. Mazieres 06. 1999,
<https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf>,
Dostęp z dnia: 16.05.2022
- [30] *Lombok*,
Project Lombok 06.2009,
<https://github.com/projectlombok/lombok>,
Dostęp z dnia: 16.05.2022
- [31] *OpenAPI Specification*,
SmartBear Software 2021,
<https://swagger.io/specification/>,
Dostęp z dnia: 16.05.2022
- [32] *Security Database Schema*,
B. Alex, L. Taylor, R. Winch, G. Hillert, Spring Security Reference 2015,

- <https://docs.spring.io/spring-security/site/docs/4.2.x/reference/html/appendix-schema.html>
- [33] *Tracking Tables*,
Liquibase Inc. 2022,
<https://docs.liquibase.com/concepts/tracking-tables/tracking-tables.html>,
Dostęp z dnia: 16.05.2022
- [34] *MySQL Server*,
MySQL 09.2014,
<https://github.com/mysql/mysql-server>,
Dostęp z dnia: 16.05.2022
- [35] *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*,
ISO/IEC 9075-1 12.2016,
<https://www.iso.org/standard/63555.html>
- [36] *The application/sql Media Type*,
Y. Shafranovich, BioFortis, Inc. 04.2013,
<https://datatracker.ietf.org/doc/html/rfc6922>,
Dostęp z dnia: 16.05.2022
- [37] *Liquibase*,
Liquibase 07.2011,
<https://github.com/liquibase/liquibase>,
Dostęp z dnia: 16.05.2022
- [38] *Uniform Resource Locators (URL)*,
T. Berners-Lee, CERN, L. Masinter, Xerox Corporation, M. McCahill, University of Minnesota 12.1994,
<https://datatracker.ietf.org/doc/html/rfc1738>,
Dostęp z dnia: 16.05.2022
- [39] *Kompendium bezpieczeństwa hasel – atak i obrona (część 1.)*,
A. Michalczyk 01.2013,
<https://sekurak.pl/kompendium-bezpieczenstwa-hasel-atak-i-obrona/>,
Dostęp z dnia: 16.05.2022