

# Politechnika Wrocławskaw

## Wydział Elektroniki, Fotoniki i Mikrosystemów

---

KIERUNEK: Automatyka i Robotyka (AIR)  
SPECJALNOŚĆ: Technologie informacyjne w systemach automatyki (ART)

## PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:  
Aplikacja webowa jako serwis społecznościowy  
udostępniający usługę mikroblogowania

AUTOR:  
Marcin Salamandra

PROMOTOR:  
dr hab. inż. Andrzej Rusiecki K30W04D03N



# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Istniejące rozwiązania . . . . .	3
1.2	Cel pracy . . . . .	4
1.3	Tematyka dalszej części pracy . . . . .	4
<b>2</b>	<b>Architektura i technologie</b>	<b>5</b>
2.1	Wykorzystane technologie - frontend . . . . .	6
2.2	Wykorzystane technologie - backend . . . . .	6
<b>3</b>	<b>Kluczowe zagadnienia</b>	<b>7</b>
3.1	Założenia aplikacji . . . . .	7
3.2	System Security . . . . .	8
3.2.1	Authentication - uwierzytelnianie . . . . .	8
3.2.2	Authorization - autoryzacja . . . . .	9
3.3	Paginacja . . . . .	10
<b>4</b>	<b>Interfejs aplikacji</b>	<b>13</b>
4.1	Rejestracja oraz logowanie . . . . .	13
4.2	Podział głównej części aplikacji na panele . . . . .	15
4.2.1	Panel menu . . . . .	15
4.2.2	Panel content . . . . .	16
4.2.3	Panel filter . . . . .	17
4.3	Widoki główne . . . . .	18
4.3.1	Lista użytkowników . . . . .	18
4.3.2	Pojedynczy użytkownik . . . . .	19
4.3.3	Edycja użytkownika . . . . .	19
4.3.4	Lista postów . . . . .	20
4.3.5	Pojedynczy post . . . . .	21
4.3.6	Tworzenie posta . . . . .	22
4.3.7	Edycja posta . . . . .	23
4.3.8	Lista komentarzy . . . . .	24
<b>5</b>	<b>Podsumowanie</b>	<b>25</b>
5.1	Możliwe rozszerzenia aplikacji . . . . .	25
5.2	Testy aplikacji . . . . .	25
5.3	Wnioski . . . . .	27
<b>Bibliografia</b>		<b>27</b>
<b>Spis rysunków</b>		<b>29</b>



# Rozdział 1

## Wstęp

Praca inżynierska traktuje o aplikacji webowej, która pełni rolę portalu społecznościowego. Skupia się on na usługach mikroblogowania i te udostępnia użytkownikom. Głównym założeniem takich usług jest fakt, iż użytkownicy wymieniają informacje za pomocą wpisów, które w dalszej części pracy nazywane są również postami, natomiast nie istnieje komunikacja poprzez bezpośrednie wysyłanie prywatnych wiadomości.

### 1.1 Istniejące rozwiązania

Portale społecznościowe zyskały bardzo dużą popularność wśród użytkowników internetu dzięki swoim innowacyjnym rozwiązaniami. Obecnie świat tych mediów dominuje kilka konkurencyjnych aplikacji, które rozwijane i utrzymywane są już przez lata. Z tego względu nowe aplikacje tego typu mają bardzo ciężkie zadanie w postaci przebicia się na tym rynku i zyskaniu nowych klientów.

Na pierwszym miejscu występuje *Twitter*, który opiera się na komunikacji poprzez posty użytkowników oraz komentarze do tych wpisów. Bardzo chętnie używaną funkcjonalnością jest również udostępnianie posta, zwiększająca jego zasięgi w aplikacji. Do poważnych wad można zaliczyć fakt, iż użytkownicy nie mają możliwości edytowania postów ani komentarzy. Aplikacja cieszy się skromnym i bardzo pozytywnym w odbiorze interfejsem użytkownika.

Kolejna aplikacja, która zdobiła rynek w ostatnich latach to *Instagram*. Podobnie jak *Twitter*, aplikacja pozwala użytkownikowi na dodawanie postów, jednakże nie jest to najważniejsza funkcjonalność. Innowacyjnym rozwiązaniem są tak zwane *relacje*, czyli krótkie, ale pełne treści filmy, które mają na celu pokazanie, czym dana osoba aktualnie się zajmuje. Istotny jest fakt, iż relacje znikają po upływie 24 godzin i nie są później dostępne, zatem liczy się bieżące śledzenie aplikacji, by nic nie ominęło użytkownika.

Jednym z najpopularniejszych portali jest również *Facebook*. W przeszłości był on zdecydomy przodującą aplikacją na rynku, jednakże w ostatnich latach portal zaliczył spadek. Aplikację charakteryzuje możliwość tworzenia grup użytkowników oraz bardzo rozbudowany system czatu, na którym również można tworzyć grupy. Jest to zdecydomy największa zaleta aplikacji.

## 1.2 Cel pracy

Celem pracy jest stworzenie aplikacji webowej o nazwie Feeflack, służącej użytkownikom jako portal społecznościowy, która będzie starała się naśladować istniejące już rozwiązania, zatem nie wprowadza ona innowacyjnych rozwiązań. Autor miał na celu pogłębienie swojej wiedzy z zakresu Security aplikacji webowych, a także poszerzenie znajomości relacyjnych baz danych oraz ich funkcjonalności.

## 1.3 Tematyka dalszej części pracy

Kolejne rozdziały odnoszą się do specyfikacji technicznej oraz osiągniętych efektów. Wybrane zostały technologie, którymi autor posługiwał się podczas implementacji systemu, szczegółowo opisano istotne elementy aplikacji oraz zaprezentowano końcowe widoki z aplikacji wraz z opisem dostępnych na nich funkcjonalności.

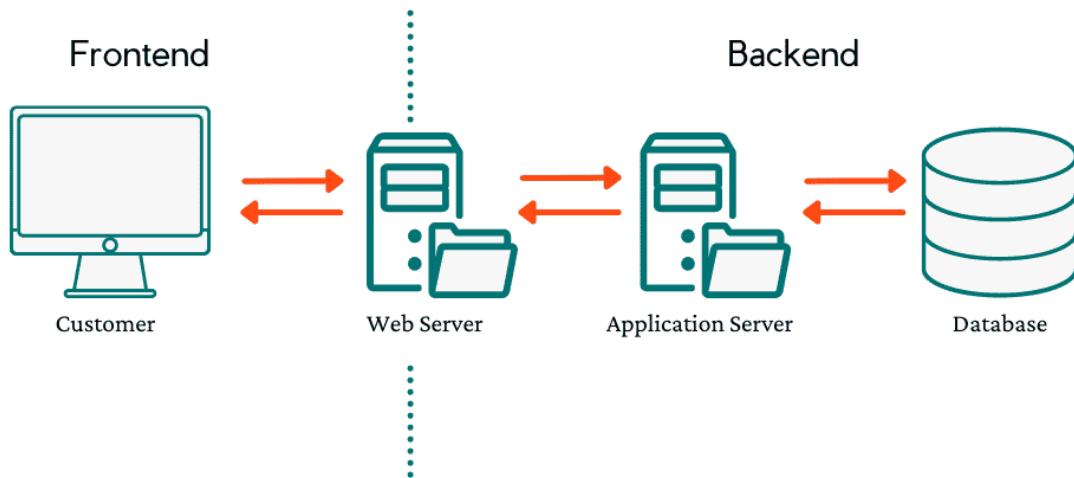
## Rozdział 2

# Architektura i technologie

Rozdział traktuje o architekturze aplikacji oraz technologiach wykorzystanych w celu jej implementacji. Wymieniona jest również literatura oraz dokumentacja dotycząca wybranych technologii.

Wybrana architektura dzieli aplikację na cztery główne sektory. Komunikują się one ze sobą w celu wymiany danych. Rola każdego z sektorów została opisana poniżej:

- Frontend - zwany również stroną klienta, to część aplikacji, z którą użytkownik ma bezpośredni kontakt. Są to przyciski, formularze, obrazy, ikony i inne elementy wizualne. Niektóre z tych elementów są interaktywne, np. reagują w pewien sposób na kliknięcie. Po stronie frontendu przeprowadzane są różne walidacje danych i zabezpieczenia, które służą tylko lepszemu doświadczeniu użytkownika na stronie internetowej, np. walidacja formularzy logowania oraz rejestracji. Frontend nie pobiera danych bezpośrednio z bazy danych. W tym celu komunikuje się on z serwerem aplikacji. Zapytania wysyłane są zgodnie z protokołem HTTP, a format przesyłanych danych to JSON.
- Web server - serwer, który komunikuje ze sobą frontend oraz backend aplikacji i umożliwia wymianę danych między tymi sektorami.
- Application server - serwer aplikacji, który przechowuje logikę biznesową aplikacji. Zanim serwer zacznie przetwarzać zapytanie i uruchomi logikę biznesową, przeprowadza on autoryzację zapytania (prawo do żądanego zasobów) oraz walidację danych przesyłanych z frontendu (zgodność oczekiwanych i odebranych danych oraz ich typów). Walidacja i zabezpieczenia przeprowadzone na serwerze aplikacji stanowią o bezpieczeństwie i zgodności danych. Logika biznesowa polega na pobieraniu, tworzeniu, edytowaniu oraz usuwaniu danych. Każda odpowiedź serwera zawiera w nagłówku status HTTP, który traktuje o powodzeniu całego zapytania.
- Database - baza danych aplikacji, w której przechowywane są dane oraz relacje między nimi.



Rysunek 2.1 Schemat przepływu danych w aplikacji

## 2.1 Wykorzystane technologie - frontend

- HTML - język znaczników, który umożliwia dodawanie elementów takich jak przyciski, obrazki, formularze itd..
- CSS - język służący do stylizowania elementów stworzonych w języku HTML. Służy jedynie do upiększania aplikacji.
- React.js - biblioteka stworzona w języku JavaScript, która ułatwia oraz przyspiesza tworzenie interfejsów użytkownika w aplikacjach webowych. W trakcie pracy korzystano z dokumentacji [3] biblioteki.
- TypeScript - typowany język oparty na języku JavaScript. Stosowano rozwiązania zawarte w dokumentacji technicznej [5] tej technologii.

## 2.2 Wykorzystane technologie - backend

- Java - język programowania, posiadający ogromną platformę o nazwie Spring, służącą do tworzenia aplikacji webowych. Oprogramowanie tworzone zgodnie z zasadami opisanymi w popularnym tytule *Clean Code* [7]
- Spring Boot - platforma stworzona w języku Java, która ułatwia oraz przyspiesza tworzenie części serwerowej w aplikacjach webowych. Platforma Spring Boot zawiera w sobie wiele istotnych modułów takich jak Spring Security (warstwa security aplikacji), czy Spring Data JPA (warstwa komunikacji z bazą danych). Podczas implementacji systemu korzystano z dokumentacji platformy [4] oraz literatury [8] dedykowanej tej technologii.
- PostgreSQL - relacyjna baza danych zyskująca w ostatnich latach ogromną popularność. Ten dialect języka SQL cechuje się bezpieczeństwem, szybkością i skalowalnością. W celu zaznajomienia się z tym językiem wykorzystano jego dokumentację techniczną [2]. Dodatkowo sposoby persystowania obiektów języka Java na encje zaczerpnięto z książki [6].

# Rozdział 3

## Kluczowe zagadnienia

Ten fragment pracy poświęcony jest omówieniu kluczowych zagadnień aplikacji. Do takich zagadnień zaliczają się jej założenia, system Security oraz paginacja.

### 3.1 Założenia aplikacji

Aplikacja powinna działać zgodnie z podstawowymi założeniami, zatem powinny być dostępne takie funkcjonalności jak:

- Stworzenie konta.
- Logowanie na istniejące konto.
- Formularze rejestracji oraz logowania informujące użytkownika o poprawności wprowadzonych danych.
- Odmowa dostępu do głównej zawartości aplikacji użytkownikom niezalogowanym, zatem każda próba dostania się na podstronę inną niż logowanie oraz rejestracja skutkuje przekierowaniem użytkownika na stronę logowania.
- Dostęp do pełnej zawartości aplikacji dla użytkowników zalogowanych.
- Przeglądanie różnych podstron typu najpopularniejsi użytkownicy, najnowsze posty.
- Edytowanie danych dotyczących własnego profilu, czyli opisu i zdjęcia profilowego.
- Usuwanie własnego konta.
- Tworzenie, edytowanie i usuwanie własnych postów zawierających opis oraz zdjęcie.
- Tworzenie, edytowanie i usuwanie własnych komentarzy pod różnymi postami.
- Wyszukiwanie użytkowników po ich nazwie.
- Obserwowanie użytkowników.
- Reagowanie na posty.

## 3.2 System Security

Portale społecznościowe, to aplikacje, w których istotną rolę odgrywa system Security, blokujący nieupoważnionym osobom dostęp do zasobów. Proces weryfikacji użytkowników rozróżnia dwa podprocesy - uwierzytelnianie oraz autoryzację.

### 3.2.1 Authentication - uwierzytelnianie

Uwierzytelnianie to proces mający na celu zweryfikowanie, czy dany użytkownik jest osobą, za którą się podaje. Proces ten ma miejsce podczas logowania użytkownika do systemu. Wygląda on następująco:

1. Użytkownik podaje swój login i hasło w aplikacji.
2. Frontend wysyła zapytanie do serwera z danymi.
3. Serwer odbiera dane i sprawdza, czy w bazie danych istnieje taki użytkownik.
4. Jeśli użytkownik podał niepoprawne dane, wtedy serwer zwraca stosowny komunikat i odmawia dostępu do aplikacji.
5. Jeśli użytkownik podał poprawne dane, wtedy serwer pozwala na dostęp do aplikacji i zwraca do frontendu JSON Web Token.

JSON Web Token nazywany jest w skrócie JWT. Jest to zahaszowany token, który przechowuje kluczowe informacje dotyczące procesu logowania.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VybmFtZTIIiLCJyb2xlcyI6WyJST0xFX1VTRVIiXSwiaXNzIjoiaHR0cDovL2xvY2FsIaG9zdDo4MDgwL2FwaS9sb2dpbiIsImV4cCI6MTY3MDY5NTE1MX0.fH0wq5xs01bKA00w6wauDRhsLjmCPyWYeNPjENWcZTs
```

Rysunek 3.1 Przykładowy JWT

HEADER: ALGORITHM & TOKEN TYPE
{ "typ": "JWT", "alg": "HS256" }
PAYOUT: DATA
{ "sub": "username2", "roles": [ "ROLE_USER" ], "iss": "http://localhost:8080/api/login", "exp": 1670695151 }

Rysunek 3.2 Odkodowane dane przykładowego JWT

Najważniejsze informacje po odkodowaniu tokenu to:

- Algorytm haszowania - **HS256**.
- Nazwa użytkownika - **username2**.
- Rola użytkownika w systemie - **ROLE\_USER**.
- Adres, z jakiego pochodzi token - **http://localhost:8080/api/login**.
- Data ważności - **10.12.2022r. godzina 18:59:11**.

Frontend zapisuje JWT użytkownika po otrzymaniu go od serwera w tzw. **Local Storage**. Jest to magazyn na różnego rodzaju dane, które potrzebne są w wielu miejscach aplikacji. JWT przydaje się w procesie autoryzacji, o którym traktuje następna sekcja.

### 3.2.2 Authorization - autoryzacja

Autoryzacja to proces polegający na zweryfikowaniu, czy użytkownik ma dostęp do żądanych zasobów. Proces ten występuje za każdym razem, gdy **zalogowany** użytkownik wysyła zapytanie do serwera. Do każdego zapytania poza logowaniem i rejestracją użytkownik potrzebuje posiadać JWT, co jednoznacznie świadczy o tym, iż dana osoba jest zalogowana do systemu. Brak JWT w nagłówku zapytania natychmiastowo skutkuje odmową dostępu do zasobów. Zapytania, w których występuje JWT, obsługiwane są w następujący sposób:

- Serwer odhaszowuje dane zapisane w JWT, który otrzymał od frontendu.
- Serwer sprawdza, czy użytkownik posiada odpowiednie uprawnienia, które są wymagane, aby otrzymać dostęp do żądanych zasobów.

Istnieją różne klasy uprawnień, które mogą zabezpieczać zasoby serwera. Najczęściej wykorzystywane uprawnienia polegają na tym, że:

- Użytkownik jest jedynie zalogowany do systemu, a zatem wystarczy, że posiada prawidłowy JWT, w którym zahaszowany jest jego login.
- Użytkownik jest zalogowany do systemu oraz posiada odpowiednią rolę, np. **ROLE\_USER** lub **ROLE\_ADMIN**. Taki sposób wykorzystywany jest bardzo często w aplikacjach posiadających różnego rodzaju płatne subskrypcje. Przykładowo w pewnej aplikacji każdy użytkownik po rejestracji otrzymuje rolę **USER**, ale taka rola upoważnia go jedynie do ograniczonej ilości zasobów. Po wykupieniu płatnej subskrypcji użytkownik otrzymuje rolę np. **SUBSCRIBER**, automatycznie dostając również dostęp do wszystkich zasobów, które wymagają tej roli. Jest to bardzo popularne, proste w implementacji i skuteczne rozwiązanie. Informacje o roli lub rolach zahaszowane są w JWT użytkownika.
- Użytkownik jest zalogowany do systemu oraz posiada bezpośrednie uprawnienia do zasobów, nazywane również **authorities** lub **permissions**. Jest to rozwinięcie drugiej klasy opierającej się na rolach. Uprawnienia są znacznie bardziej szczegółowe niż role. Przykładowo w pewnej aplikacji użytkownik może otrzymać blokadę na pisanie wiadomości na czacie za wulgarny język. W takim przypadku rozwiązanie polegające na rolach nie wystarczy, gdyż nie można odebrać użytkownikowi całej roli **USER**, ponieważ takie działanie skutkowałoby tym, że użytkownik straciłby dostęp do całej aplikacji. Jest to idealny przykład na zastosowanie bezpośrednich uprawnień np. uprawnienie **WRITE\_ON\_CHAT** bądź **SEND\_MESSAGE**. W przytoczonym przykładzie wystarczy jedynie usunąć konkretne uprawnienie z listy uprawnień użytkownika. W ten sposób użytkownik traci tylko określoną część uprawnień. Rozwiązania te są bardzo do siebie podobne w działaniu, natomiast stworzenie systemu opartego na bezpośrednich uprawnieniach jest znacznie bardziej wymagające. Bezpośrednie uprawnienia zapisane są w JWT użytkownika.

W przygotowanej aplikacji rozwiązanie wystarczające to klasa pierwsza, gdyż istnieje tylko jedna klasa użytkowników i wszyscy użytkownicy mają te same uprawnienia, jednak w celach naukowych zostało zastosowane rozwiązanie klasy drugiej i każdy użytkownik posiada rolę **ROLE\_USER**.

### 3.3 Paginacja

Drugim ważnym elementem portalu społecznościowego jest baza danych oraz optymalizacja zapytań wysyłanych przez klienta do serwera, gdyż portale społecznościowe muszą płynnie przetwarzać bardzo duże zbiory danych. Nowoczesna aplikacja powinna umożliwić użytkownikowi szybki dostęp do żądanego zasobu, a czas oczekiwania pomiędzy zapytaniami powinien być możliwie najkrótszy.

W wielu przypadkach aplikacje spowalniają swoje działanie przez zbyt obciążające zapytania do serwera, dlatego obecnie stosuje się zasadę, iż należy wysyłać do klienta tyle danych, ile jest on w stanie wyświetlić na raz, a większa ilość danych zostanie dosłana, gdy zajdzie potrzeba. Taki efekt zapewnia metoda zwana **paginacją** lub **stronicowaniem**.

Paginacja polega na tym, że użytkownik wysyłający zapytanie o pewną kolekcję danych, otrzymuje jedynie jej część. Takie działanie ma swoje wytlumaczenie, ponieważ jeśli danych jest dużo, to użytkownik nie jest w stanie wyświetlić ich wszystkich na ekranie swojego urządzenia. Z tego powodu zwracanie całej kolekcji nie jest konieczne.

Przesyłanie ograniczonych zbiorów danych skutkuje mniej obciążającymi zapytaniami, a więc serwer aplikacji szybciej obsługuje konkretne zapytanie. W przeciwnym wypadku, gdy serwer zwraca całą zawartość tabeli, wszystkie te dane muszą zostać zapisane w pamięci komputera. Po uwzględnieniu liczby użytkowników aplikacji, która w jednym momencie może być ogromna, okazuje się, iż serwer nie radzi sobie z obsługą wszystkich zapytań w rozsądny czasie. Skutkuje to opóźnieniami, które doświadczają klienci, co jest dla nich frustrujące i zniechęca ich do korzystania z danej aplikacji.

Paginacja odbywa się na etapie pobierania danych z bazy. Język SQL wspiera paginację danych i udostępnia odpowiednie funkcje. Paginacja zawiera kilka składowych:

- Atrybut **page** - określa, którą stronę danych należy pobrać, np. 38.
- Atrybut **size** - określa wielkość stron, na jakie zostanie podzielona kolekcja, np. 10.
- Atrybut **sort** - określa sposób oraz kierunek sortowania, np. id,DESC.

Przykładowe użycie paginacji:

- Kolekcja danych składająca się z 580 wierszy.
- Paginacja o wymienionych wyżej wartościach jej atrybutów.

Baza danych zwróci dane posortowane **malejąco** po kolumnie **id** z zakresu **[371-380]**, zakładając, że dane są numerowane od 1.

Paginacja bardzo często wykorzystywana jest w połączeniu z tabelami, pod którymi znajdują się przyciski "lewo" i "prawo". Dane wyświetlane aktualnie w tabeli to dane z ostatnio pobranej **strony**. Kliknięcie przycisku "prawo" skutkuje pobraniem **kolejnej** strony danych, natomiast "lewo" pobiera **poprzednią** stronę.



Rysunek 3.3 Przykładowa stopka tabeli wykorzystującej paginację

W przygotowanej aplikacji został zastosowany inny sposób paginowania danych. Język TypeScript umożliwia sprawdzenie, czy dany element HTML jest widoczny aktualnie na ekranie. W tym celu należy wykorzystać gotowe narzędzie - Intersection Observer [1]. Na jego podstawie zaprojektowano algorytm, który sprawdza, czy klient widzi na ekranie ostatni element paginowanej kolekcji. Jeśli tak jest, to frontend aplikacji automatycznie wysyła zapytanie po kolejną stronę paginowanej kolekcji. Takie rozwiązanie sprawia, iż użytkownik nie musi kliknąć przycisku pobierającego dane i wszystko dzieje się automatycznie bez jego wiedzy. Jest to funkcjonalność zapewniająca wygodne przeglądanie listy postów, komentarzy oraz użytkowników.



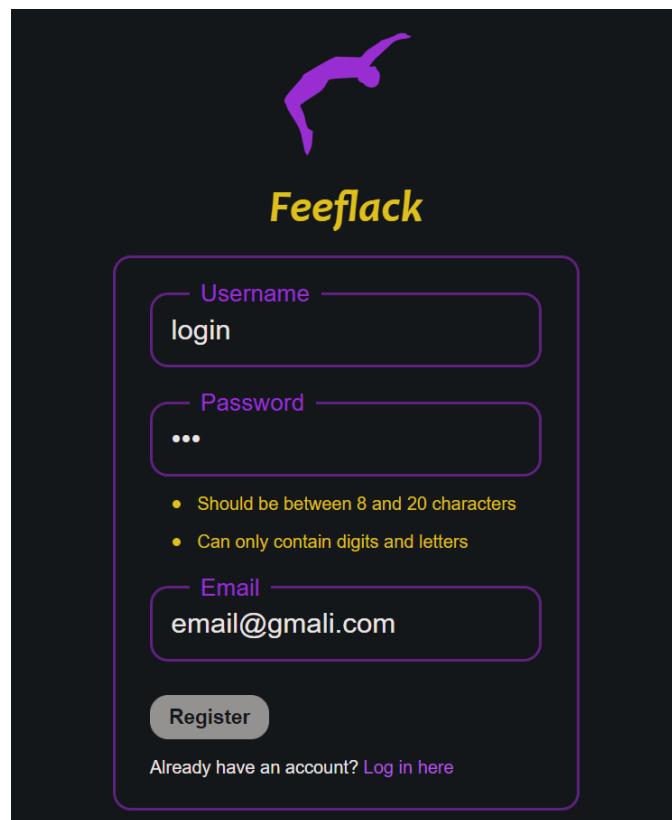
# Rozdział 4

## Interfejs aplikacji

Dalsza część pracy skupia na prezentacji zaimplementowanych funkcjonalności.

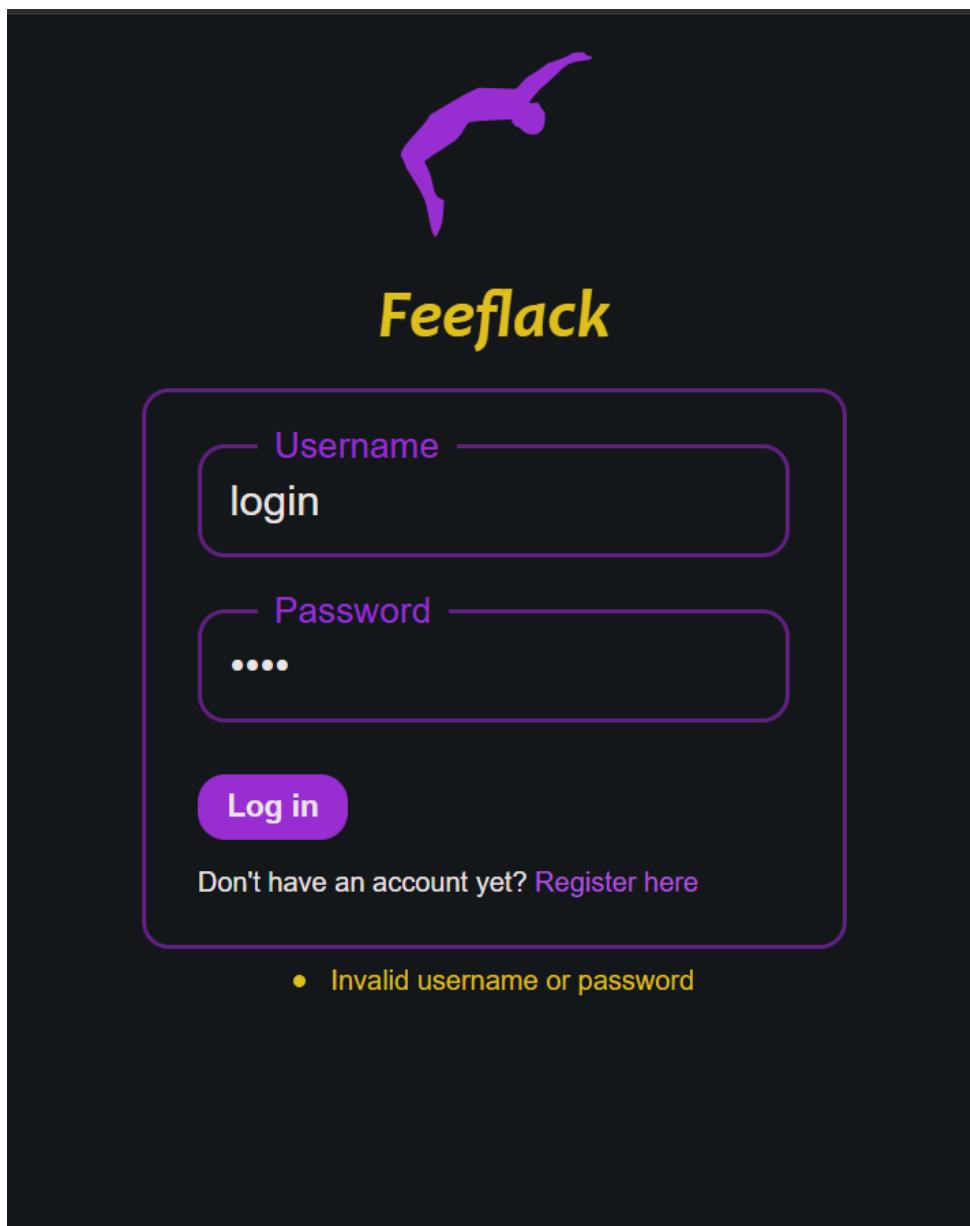
### 4.1 Rejestracja oraz logowanie

W celu rejestracji nowego konta użytkownik uzupełnia formularz, który informuje użytkownika o błędach wprowadzonych w konkretnych polach. Dodatkowo wyświetlany jest stosowny komunikat, jeśli podana nazwa użytkownika lub adres e-mail jest zajęty. Dostępny jest również odnośnik do formularza logowania, jeśli użytkownik posiada już konto. Rejestracja zakończona powodzeniem automatycznie przenosi użytkownika do formularza logowania. Przycisk **Register** jest aktywny wyłącznie wtedy, gdy wszystkie pola są uzupełnione i spełniają wszystkie wymagania.



Rysunek 4.1 Formularz rejestracji

Formularz logowania został przygotowany analogicznie jak formularz rejestracji. Formularz zwraca stosowny komunikat, jeśli w bazie danych nie istnieje użytkownik o uzupełnionych danych. Dodatkowo przycisk **Log in** jest aktywny wyłącznie wtedy, gdy oba pola są uzupełnione.



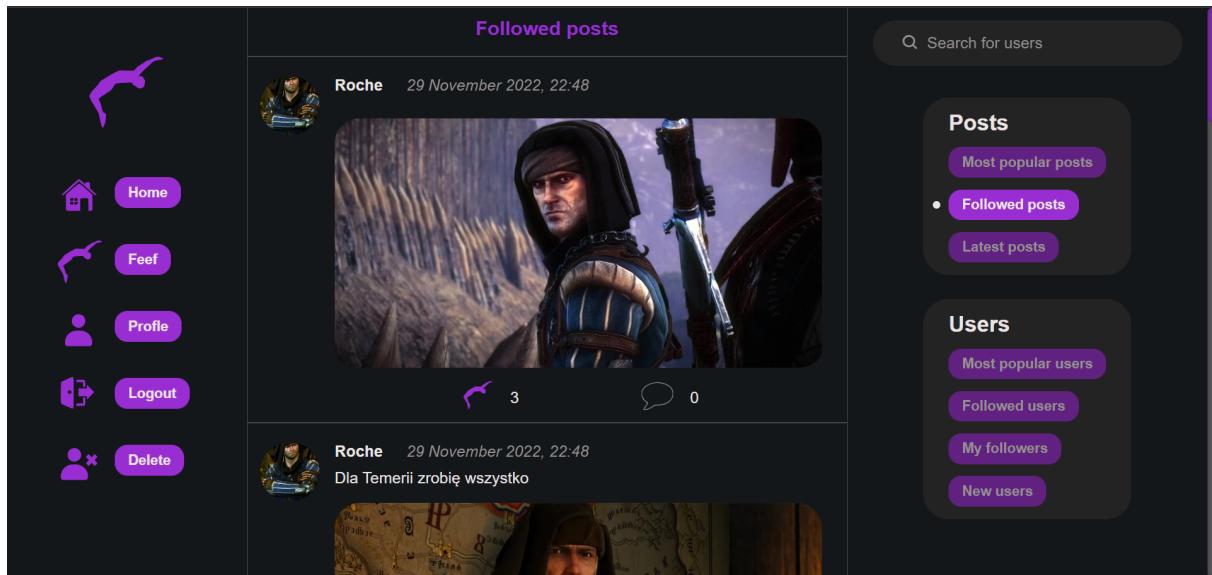
Rysunek 4.2 Formularz logowania

## 4.2 Podział głównej części aplikacji na panele

Widok aplikacji został podzielony na 3 panele:

- Panel po lewo - menu.
- Panel na środku - content.
- Panel po prawo - filter.

**Menu** oraz **filter** są statyczne i zawsze widoczne w takiej samej postaci. Panel **content** zmienia swoją zawartość w zależności od wyborów użytkownika.

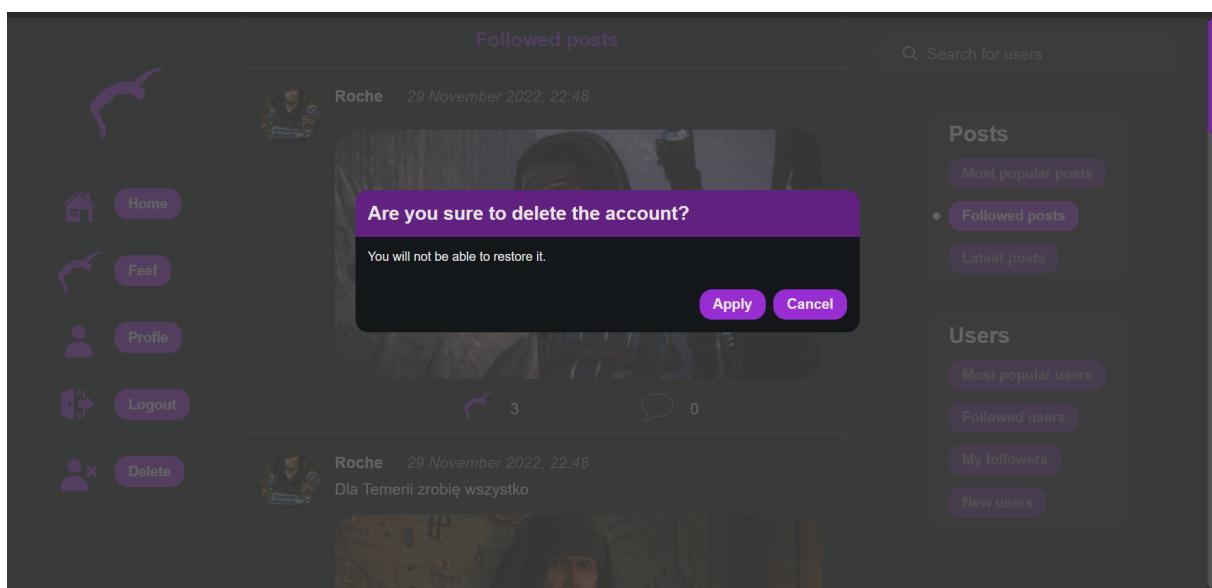


Rysunek 4.3 Podział aplikacji na panele

### 4.2.1 Panel menu

W panelu **menu** dostępne są jedynie przyciski z najważniejszymi funkcjonalnościami, które muszą być zawsze dostępne dla użytkownika:

- Home - przenosi użytkownika na domyślny widok aplikacji.
- Feef - przenosi do formularza tworzącego nowy post.
- Profile - przenosi użytkownika do jego profilu, gdzie może edytować swoje dane oraz posty.
- Logout - wylogowuje użytkownika z aplikacji i przenosi do formularza logowania.
- Delete - usuwa konto użytkownika i przenosi do formularza logowania. Kliknięcie przycisku wyświetla na ekranie ostrzeżenie, które wymaga potwierdzenia swojego wyboru. W tym czasie wszystkie pozostałe przyciski nie są interaktywne. Komunikat można zamknąć lub zaakceptować.



Rysunek 4.4 Komunikat wymagający potwierdzenia swojego żądania

#### 4.2.2 Panel content

W panelu **content** wyświetlana jest wybrana przez użytkownika zawartość. W aplikacji dostępne są główne widoki:

- Lista postów.
- Pojedynczy post.
- Tworzenie posta.
- Edycja posta.
- Lista komentarzy.
- Lista użytkowników.
- Pojedynczy użytkownik.
- Edycja użytkownika.

Dodatkowo użytkownik ma możliwość odwiedzenia poszczególnych podwidoków, które bazują na widokach głównych, przykładowo **Najnowsze posty** oraz **Najpopularniejsze posty** korzystają z widoku głównego **Lista postów**, różnią się jedynie danymi:

- Najpopularniejsze posty.
- Posty obserwowanych użytkowników.
- Najnowsze posty.
- Najpopularniejsi użytkownicy.
- Obserwujący danego użytkownika.
- Obserwowani przez danego użytkownika.
- Najnowsi użytkownicy.

Główny widok aplikacji to podwidok **Posty obserwowanych użytkowników**, gdyż przeglądanie postów znajomych jest głównym przeznaczeniem aplikacji.

### 4.2.3 Panel filter

Panel **filter** służy do filtrowania zawartości wyświetlanej w panelu **content**. Wybrany podwidok jest podświetlony na liście oraz oznaczony białą kropką. Dostępne w tym panelu opcje to:

- Wyszukiwarka użytkowników po ich nazwie. Wystarczy, że wpisana fraza znajduje się w nazwie użytkownika.
- Most popular posts - przenosi do podwidoku z najpopularniejszymi postami. Posty są sortowane malejąco po liczbie reakcji.
- Followed posts - przenosi do podwidoku z postami obserwowanych użytkowników.
- Latest posts - przenosi do podwidoku z najnowszymi postami. Posty są sortowane malejąco po dacie utworzenia posta.
- Most popular users - przenosi do podwidoku z najpopularniejszymi użytkownikami. Użytkownicy są sortowani malejąco po liczbie osób, która ich obserwuje.
- Followed users - przenosi do podwidoku z obserwowanymi przez danego użytkownika.
- My followers - przenosi do podwidoku z obserwującymi danego użytkownika.
- New users - przenosi do podwidoku z najnowszymi użytkownikami. Użytkownicy są sortowani po dacie utworzenia konta.

## 4.3 Widoki główne

### 4.3.1 Lista użytkowników

Jest to widok bazowy dla każdego podwidoku użytkowników. W panelu **content** użytkownicy wyświetlały się w kolumnie, jeden pod drugim. Użytkownik może zawierać:

- Zdjęcie profilowe oraz jego nazwę. Kliknięcie nazwy użytkownika przenosi do widoku **Pojedynczy użytkownik**.
- Datę utworzenia konta.
- Informację o liczbie obserwowanych i obserwujących.
- Opis.

Warto nadmienić, iż dostępne są również przyciski **Follow** oraz **Unfollow**, których wyświetlanie zależy od tego, czy zalogowany użytkownik obserwuje już danego użytkownika, czy też nie.

### Followers of Geralt

**Jaskier** Joined 29 November 2022 Unfollow

3 Followers 1 Following

Jaskier (czy też raczej wicehrabia Julian de Lettenhove) jest w świecie wiedźmina szeroko znany i popularnym bardem i trubadurem, „słynnym od Buiny po Jarugę po dworach, kasztelach, zajazdach, oberżach i zamczyskach”, autorem wielu wierszy, ballad, pieśni i gawęd. Jest przyjacielem głównego bohatera, Geralta, przebywającym z nim wiele przygód i często wpłatającym go w różnorakie kłopoty. Grywał między innymi na dworach królów Niedamira, Vizimira i Venzlava.

**Zoltan** Joined 29 November 2022 Unfollow

2 Followers 8 Following

Zoltan Chivay – krasnolud, weteran II wojny z Nilfgaardem, przyjaciel Geralta. Ich drogi zeszły się, kiedy Geralt ze swoją drużyną zmierzał z Brokilonu w stronę Jarugi, na południe. Za radą krasnoluda, Geralt, Jaskier i Milva przyłączyli się do kompanii i podążyli na wschód, razem z uciekinierami z Kernow. Właśnie od Zoltana, Geralt dostał swój miecz – sihill.

**Dijkstra** Joined 29 November 2022 Follow

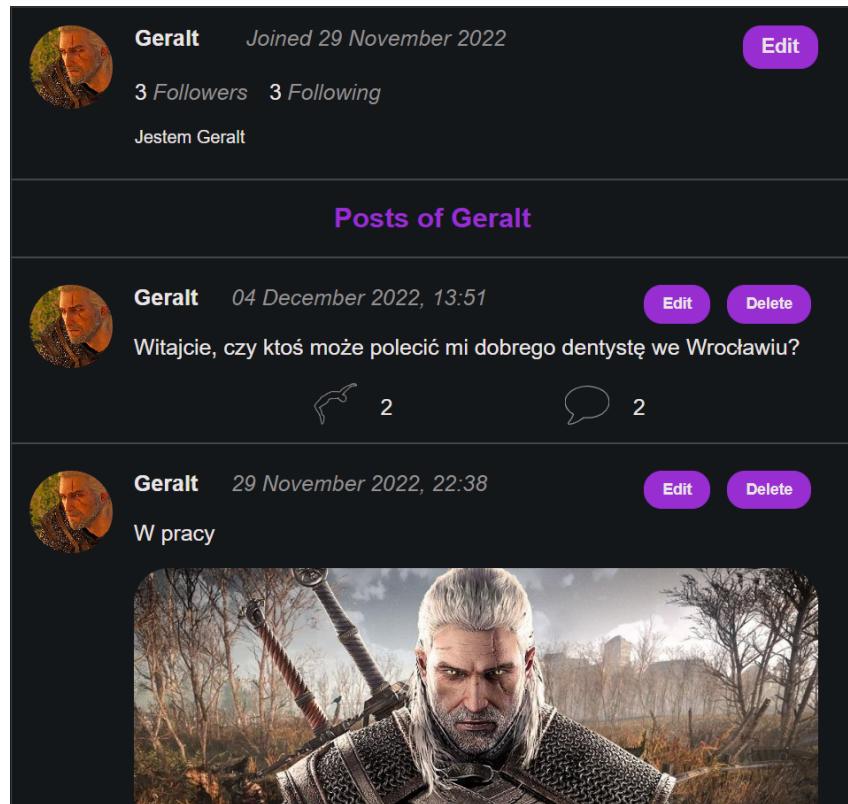
1 Followers 6 Following

Redański szpieg

Rysunek 4.5 Widok listy użytkowników

### 4.3.2 Pojedynczy użytkownik

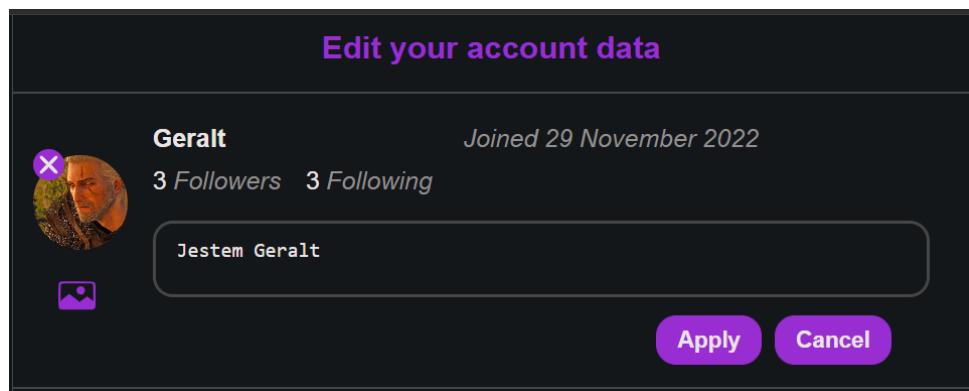
Jest to widok pojedynczego użytkownika. Składa się on z podstawowych informacji o użytkowniku oraz jego postach. Zalogowany użytkownik ma dostęp do przycisku **Edit**, dzięki któremu może edytować informacje o swoim profilu. Ma on również dostęp do edycji swoich postów.



Rysunek 4.6 Widok pojedynczego użytkownika

### 4.3.3 Edycja użytkownika

Na tej stronie użytkownik może edytować opis swojego profilu, usunąć lub zmienić swoje zdjęcie profilowe za pomocą odpowiednich przycisków.



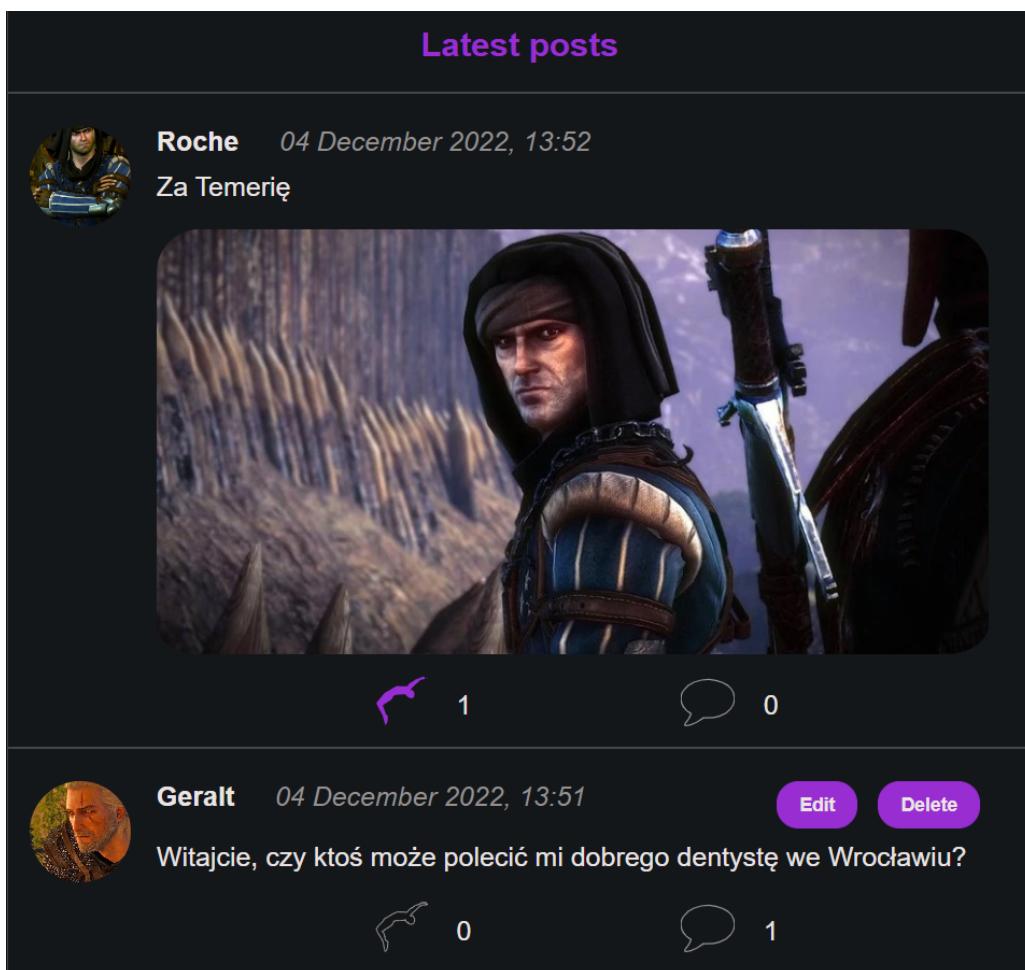
Rysunek 4.7 Widok edycji użytkownika

### 4.3.4 Lista postów

Jest to widok bazowy dla każdego podwidoku postów. W panelu **content** posty wyświetlały się w kolumnie, jeden pod drugim. Post może zawierać:

- Zdjęcie profilowe autora oraz jego nazwę. Kliknięcie nazwy użytkownika przenosi do widoku **Pojedynczy użytkownik**.
- Datę utworzenia.
- Opis.
- Zdjęcie.
- Ikonę **feeflack** oraz ikonę **komentarz**. Ikona feeflack służy do reagowania na posty. Jest ona zapełniona, jeśli zalogowany użytkownik zareagował już na danego posta. Ikona komentarz przenosi do widoku **Pojedynczy post**.

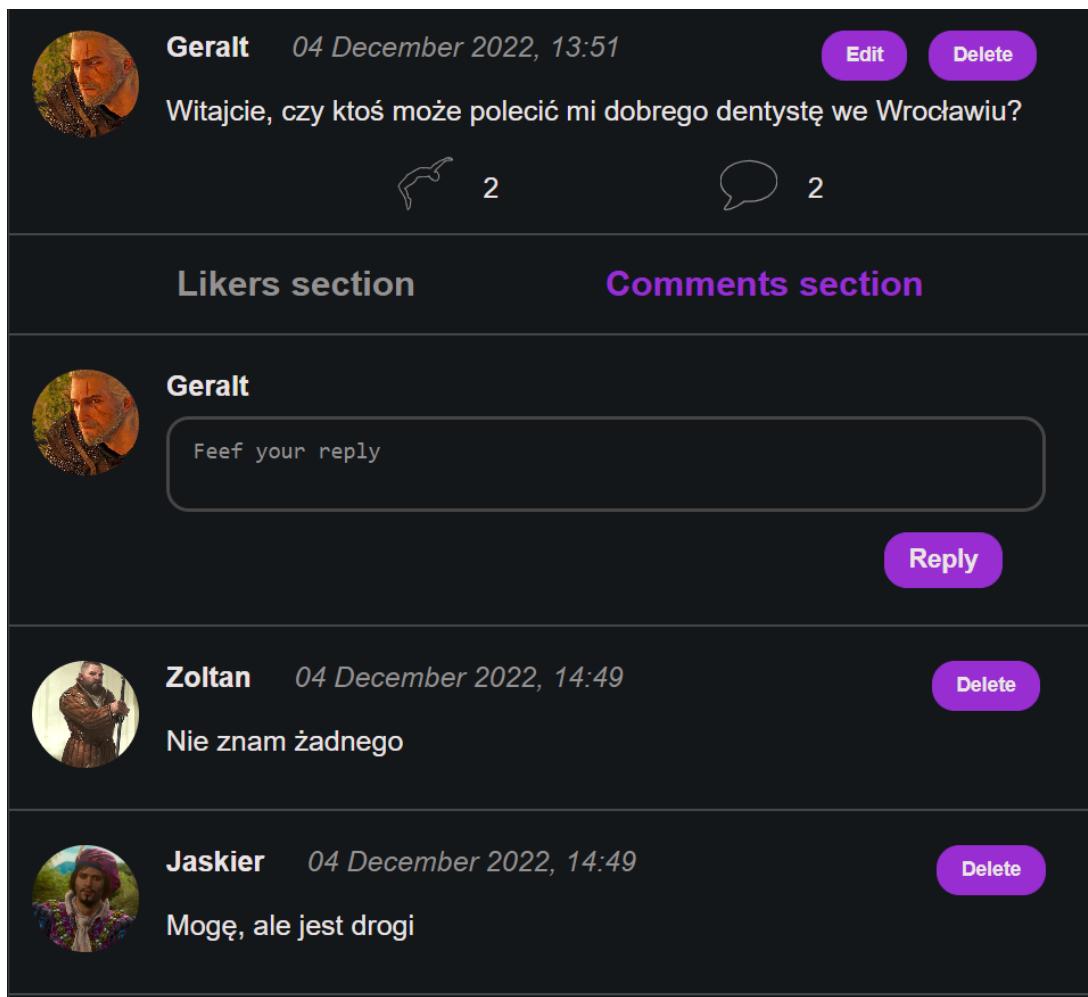
Warto nadmienić, iż dostępne są również przyciski **Edit** oraz **Delete**, jeśli dany post należy do zalogowanego użytkownika. Przycisk edit przenosi do widoku **Edycja posta**, natomiast delete usuwa posta.



Rysunek 4.8 Widok listy postów

### 4.3.5 Pojedynczy post

Jest to widok pojedynczego posta. Składa się on z podstawowych informacji o poście oraz sekcji komentarzy i osób reagujących. Sekcja komentarzy to widok **Lista komentarzy**, natomiast sekcja osób reagujących to widok **Lista użytkowników**. Użytkownik ma możliwość wyboru aktywnej sekcji oraz dostępne są wszystkie funkcjonalności z widoku **Lista postów**. W sekcji komentarzy znajduje się również formularz służący dodawaniu komentarzy. Autor posta może usunąć dowolny komentarz, natomiast pozostali użytkownicy mają możliwość usuwania tylko swoich komentarzy. Dodatkowo tylko autor może post usunąć oraz edytować.



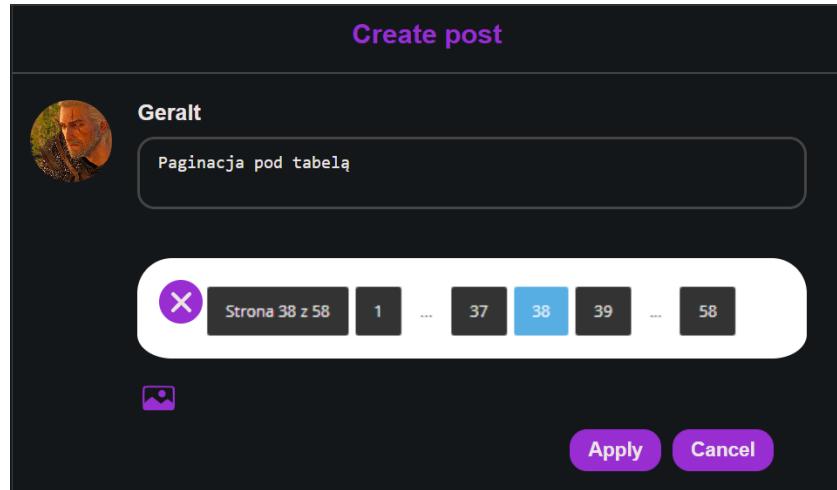
Rysunek 4.9 Widok pojedynczego posta z sekcją komentarzy



Rysunek 4.10 Widok pojedynczego posta z sekcją osób reagujących

### 4.3.6 Tworzenie posta

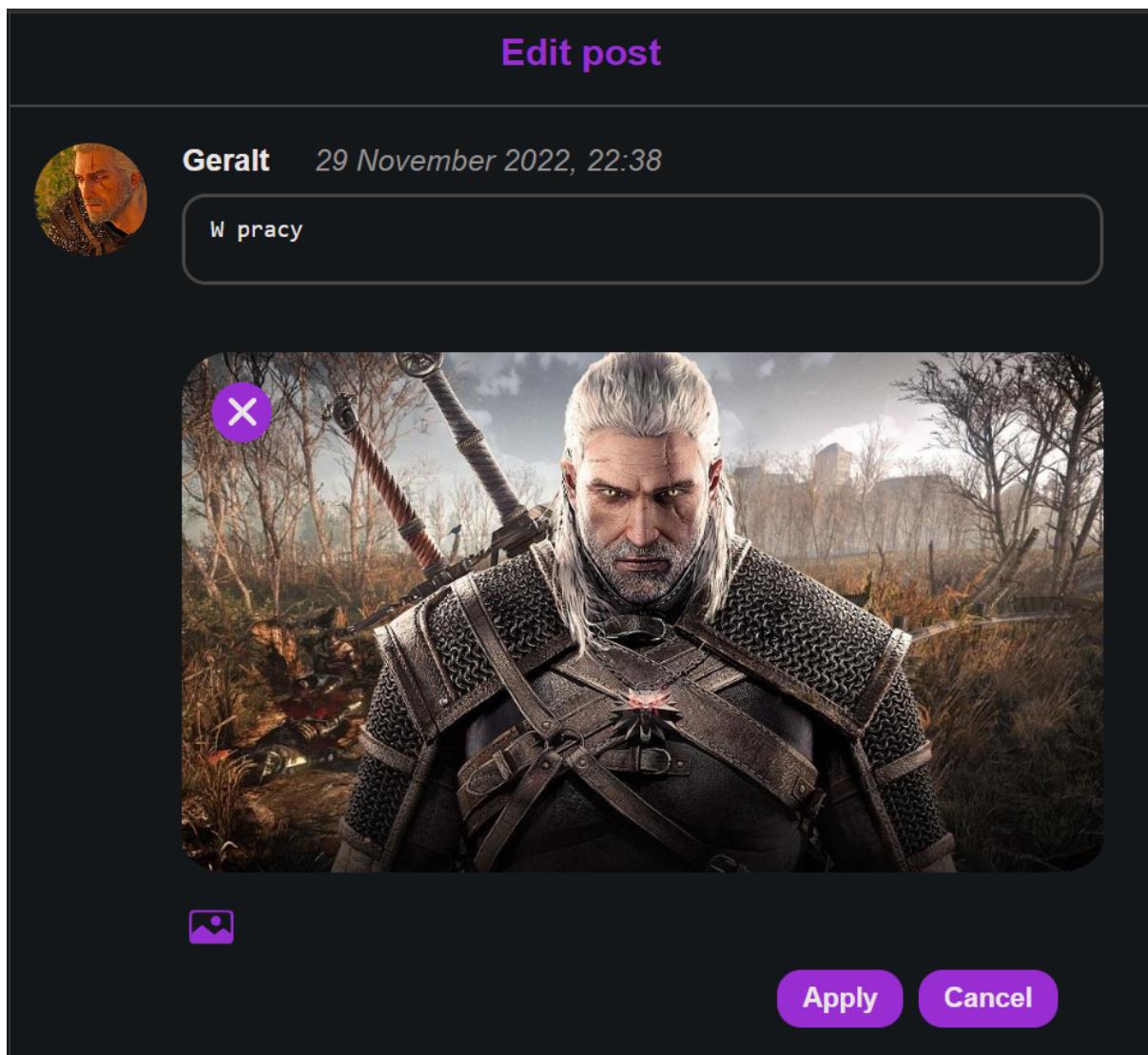
Widok przedstawiający formularz do tworzenia posta. Użytkownik ma możliwość wyboru zdjęcia ze swojego urządzenia za pomocą odpowiedniego przycisku oraz wprowadzenia opisu. Zdjęcie można anulować krzyżykiem.



Rysunek 4.11 Widok tworzenia posta

### 4.3.7 Edycja posta

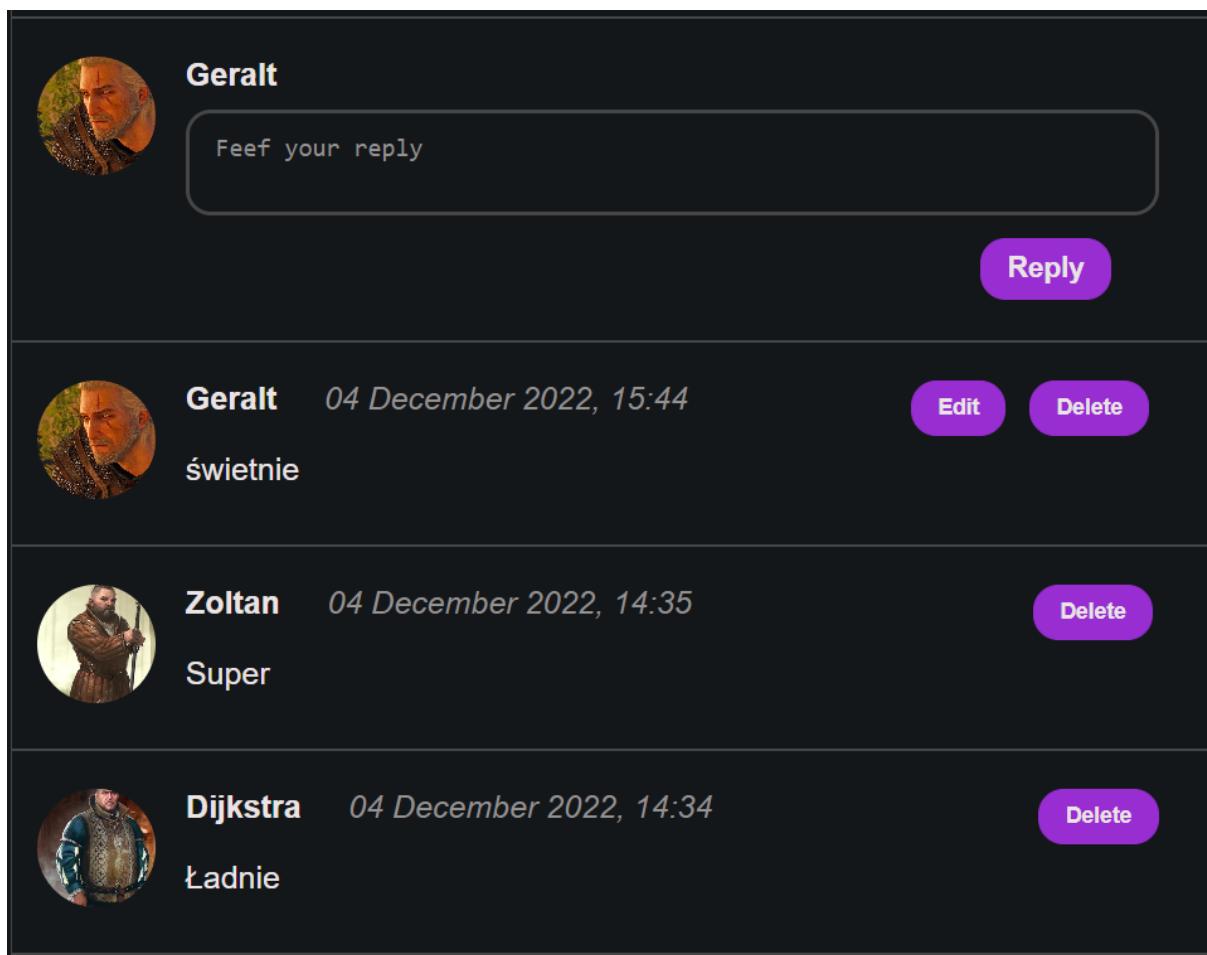
Widok przedstawiający formularz do edycji posta. Użytkownik ma możliwość wyboru zdjęcia ze swojego urządzenia za pomocą odpowiedniego przycisku oraz wprowadzenia opisu. Zdjęcie można anulować krzyżkiem.



Rysunek 4.12 Widok edycji posta

### 4.3.8 Lista komentarzy

Widok przedstawiający listę komentarzy pod postem. Użytkownik ma możliwość zostawienia komentarza poprzez uzupełnienie formularza. Widok ten zawsze występuje pod postem. Użytkownik ma możliwość usuwania i edytowania swoich komentarzy. Formularz edycji komentarzy wygląda analogicznie jak ich tworzenia.



Rysunek 4.13 Widok listy komentarzy

# Rozdział 5

## Podsumowanie

W podsumowaniu autor podaje możliwe rozszerzenia aplikacji, opisuje sposób jej testowania, a także potwierdza, iż założony cel projektu został osiągnięty.

### 5.1 Możliwe rozszerzenia aplikacji

Istnieje kilka pomysłów na rozbudowanie aplikacji, które nie zostały zrealizowane głównie z powodu niewystarczającej ilości czasu na implementację aplikacji:

- Weryfikacja zarejestrowanych użytkowników poprzez wysyłanie wiadomości e-mail do użytkownika ze specjalnym kodem weryfikacyjnym.
- Reagowanie nie tylko na posty, ale również komentarze, a także ich komentowanie.
- System powiadomień informujący o reakcjach i komentarzach.
- Dodanie tagów do postów i wyszukiwanie postów po tych tagach. Tag to tekst zawarty w opisie posta, który często zaczyna się od znaku „#”.
- Oznaczanie innych użytkowników w postach oraz komentarzach. Oznaczenie to tekst zawarty w opisie posta lub komentarza, który często zaczyna się od znaku „@”.
- Publikowanie nie tylko zdjęć, ale również filmów.

### 5.2 Testy aplikacji

Testy przeprowadzano jedynie na części serwerowej aplikacji. Testowano, czy wysyłając odpowiednio sformułowane zapytanie, serwer odpowiada w oczekiwany sposób. Do testów wykorzystano narzędzie **Postman**. Narzędzie to służy do wysyłania zapytań HTTP do dowolnego serwera. By wysłać zapytanie, należy podać:

- Adres URL serwera.
- Odpowiednie nagłówki zapytania. np. nagłówek **Authorization**, którego wartość to token JWT.
- Metodę HTTP, np. **GET** lub **POST**.
- Opcjonalne ciało zapytania, czyli dane, które należy przesłać na serwer, np. informacje o nowo utworzonym poście.

Interfejs narzędzia jest prosty i intuicyjny. Poniżej przedstawiono rysunki przykładowych testów, które w tym przypadku pobierają użytkownika.

W momencie wystąpienia błędów, które widać poniżej, serwer zwraca stosowny, klarowny komunikat, podając przyczynę oraz odpowiedni status HTTP, np. **FORBIDDEN** lub **NOT FOUND**. W przypadku, gdy cały proces zakończy się powodzeniem, serwer zwraca żądane zasoby.

Wszystkie pozostałe funkcjonalności testowano w sposób analogiczny. Testy nie wykazały błędów implementacyjnych i wszystkie przygotowane funkcjonalności działają tak, jak przewidywano.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections (selected), APIs, Environments, Mock Servers, Monitors, Flows, History.
- Top Bar:** Home, Workspaces, API Network, Explore, Search Postman, Upgrade.
- Current Collection:** Microblogging app.
- Current Endpoint:** users / get user (GET, http://localhost:8080/api/users/13).
- Headers Tab:** Selected, showing 7 headers.
- Body Tab:** PRETTY format, showing the following JSON response:
 

```

1  "httpStatus": "FORBIDDEN",
2  "message": "No request token"
      
```
- Bottom Status Bar:** Cookies, Capture requests, Runner, Trash.

Rysunek 5.1 Interfejs narzędzia Postman

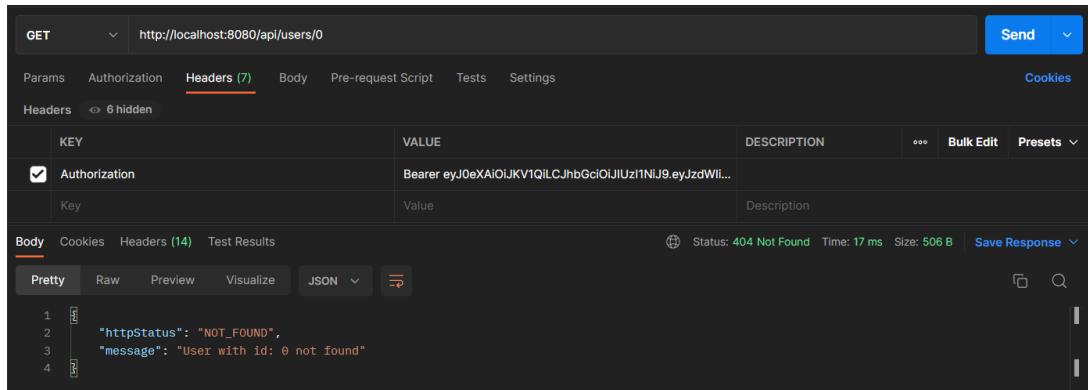
The screenshot shows the Postman interface with the following details:

- Top Bar:** GET, http://localhost:8080/api/users/13, Send.
- Headers Tab:** Selected, showing 7 headers.
- Body Tab:** PRETTY format, showing the following JSON response:
 

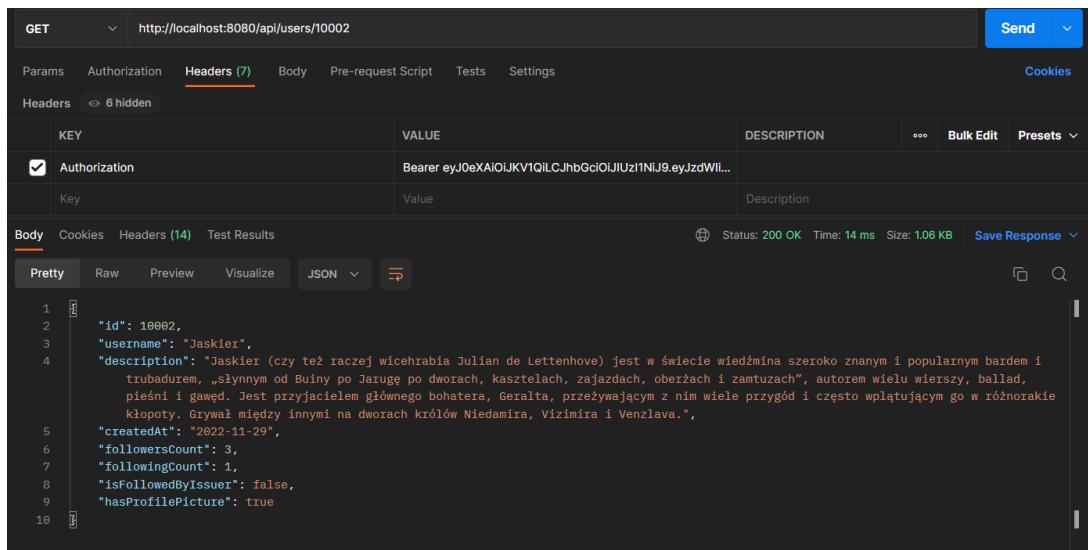
```

1  "httpStatus": "FORBIDDEN",
2  "message": "No request token"
      
```
- Bottom Status Bar:** Status: 403 Forbidden, Time: 8 ms, Size: 514 B, Save Response.

Rysunek 5.2 Próba pobrania użytkownika bez podania tokenu JWT



Rysunek 5.3 Próba pobrania nieistniejącego użytkownika



Rysunek 5.4 Próba pobrania użytkownika zakończona powodzeniem

## 5.3 Wnioski

Celem projektu było stworzenie portalu społecznościowego, który byłby odwzorowaniem istniejących, podobnych aplikacji. Ten cel udało się osiągnąć i aplikacja spełnia postawione wcześniej wymagania.

Użytkownicy mają możliwość pozyskiwania nowych znajomości poprzez system obserwacji. Każdy użytkownik może również dzielić się informacjami o swojej osobie, zawierając je na swoim profilu w postaci opisu lub postów. Ponadto istnieje możliwość komentowania postów, dzięki czemu występuje bezpośrednia interakcja pomiędzy użytkownikami. Na każdy post można również zareagować.

Autorowi udało się zgłębić swoją wiedzę dotyczącą możliwości baz danych oraz systemów Security, co również było celem projektu.

Podczas realizowania projektu występowały różne problemy. Najważniejszym z nich było wybranie sposobu wykorzystania paginacji bez konieczności używania przycisków. Ten problem udało się rozwiązać za pomocą narzędzia **Intersection Observer** dostępnego w języku TypeScript.



# Literatura

- [1] Intersection Observer documentation, 2022. [https://developer.mozilla.org/en-US/docs/Web/API/Intersection\\_Observer\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API).
- [2] PostgreSQL documentation, 2022. <https://www.postgresql.org/docs/14/index.html>.
- [3] React documentation, 2022. <https://reactjs.org/>.
- [4] Spring Boot documentation, 2022. <https://spring.io/projects/spring-boot>.
- [5] TypeScript documentation, 2022. <https://www.typescriptlang.org/docs/>.
- [6] C. Bauer, G. King, G. Gregory. *Java Persistence with Hibernate, 2nd Edition*. Helion, 2016.
- [7] R. C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Financial Times Prentice Hall, 2008.
- [8] C. Walls. *Spring Boot in Action*. Manning, 2016.



# Spis rysunków

2.1	Schemat przepływu danych w aplikacji	6
3.1	Przykładowy JWT	8
3.2	Odkodowane dane przykładowego JWT	9
3.3	Przykładowa stopka tabeli wykorzystującej paginację	11
4.1	Formularz rejestracji	13
4.2	Formularz logowania	14
4.3	Podział aplikacji na panele	15
4.4	Komunikat wymagający potwierdzenia swojego żądania	16
4.5	Widok listy użytkowników	18
4.6	Widok pojedynczego użytkownika	19
4.7	Widok edycji użytkownika	19
4.8	Widok listy postów	20
4.9	Widok pojedynczego posta z sekcją komentarzy	21
4.10	Widok pojedynczego posta z sekcją osób reagujących	22
4.11	Widok tworzenia posta	22
4.12	Widok edycji posta	23
4.13	Widok listy komentarzy	24
5.1	Interfejs narzędzia Postman	26
5.2	Próba pobrania użytkownika bez podania tokenu JWT	26
5.3	Próba pobrania nieistniejącego użytkownika	27
5.4	Próba pobrania użytkownika zakończona powodzeniem	27