Dokumentacja projektowa - PAINT

Zespół

- Piotr Brzeziński
- Wojciech Kołodziejak
- Korneliusz Litman
- Mateusz Maj
- Bartłomiej Rasztabiga
- Marcin Zasuwa

Temat

Aplikacja webowa służąca do komunikacji uzytkownikami w stylu tzw. "instant messaging" (np. WhatsApp, Messenger, Telegram, Discord, Slack, itp.). Aplikacja powinna umożliwiać komunikację w czasie rzeczywistym, a także przesyłanie plików.

Wymagania funkcjonalne

- Rejestracja i logowanie użytkowników: Aplikacja pozwala na tworzenie indywidualnych kont użytkowników, które mogą być logowane za pomocą unikalnych danych uwierzytelniających, zapewniając bezpieczny dostęp do platformy i jej funkcji.
- Wysyłanie wiadomości prywatnych: Użytkownicy mogą bezpośrednio komunikować się z innymi, wysyłając wiadomości prywatne, które są dostępne tylko dla odbiorcy, umożliwiając bezpośrednią, osobistą komunikację.
- Wysyłanie wiadomości grupowych: Aplikacja pozwala na tworzenie konwersacji grupowych, umożliwiając użytkownikom wysyłanie wiadomości do wielu osób naraz, co promuje kolaborację i dyskusję zbiorową.
- Wysyłanie i pobieranie plików: Użytkownicy mogą przesyłać i pobierać pliki różnego typu, takie jak dokumenty, zdjęcia czy filmy, umożliwiając łatwą wymianę informacji i materiałów pomiędzy użytkownikami.
- Status aktywności: Aplikacja wyświetla status aktywności użytkowników, czy są w danym momencie online

Podział na moduły

- Backend:
  - język: Kotlin (JVM)
  - framework: Spring Boot
  - inne technologie: Spring Security, Spring Data, Spring WebSockets, Spring Messaging
- Frontend:
  - język: TypeScript
  - framework: React.js z frameworkiem Vite
  - inne technologie: Tailwind CSS, React Router
- Baza danych: MongoDB

Aplikacja została wdrożona na prywatnym klastrze Kubernetes

Przydział odpowiedzialności w ramach zespołu

Piotr Brzeziński - Frontend, UI/UX Wojciech Kołodziejak - Backend, Baza Danych Korneliusz Litman - Backend, Baza Danych Mateusz Maj - Frontend, UI/UX Bartłomiej Rasztabiga - Backend, Kierownik projektu Marcin Zasuwa - Frontend, Infrastruktura

Prezentacja projektu

Demo projektu zostało nagrane przy użyciu dwóch instancji przeglądarki, każdej zalogowanej na konto innego użytkownika. Na nagraniu można zaobserwować możliwość wysyłania wiadomości do pojedynczych użytkowników oraz konwersacji grupowych, a także wysyłanie dowolnych plików.

https://youtu.be/P9zfKfwgTDU

Przykładowi użytkownicy

Aby przetestować działanie systemu, można założyć nowe konto lub wykorzystać jedno z poniższych:

Nazwa użytkownika	Hasło
user	user
user1	user1
user2	user2

Backend

Backend został podzielony na następujące moduły:

- conversation: Zawiera obsługe konwersacji i wiadomości
- files: Zawiera obsługę plików
- users: Zawiera informacje o użytkownikach, umożliwia rejestrację użytkowników

Uwierzytelnianie i autoryzacja użytkowników jest oparta o aplikację Keycloak. https://www.keycloak.org

Backend konfigurowany jest za pomocą zmiennych środowiskowych. Aplikacja jest budowana za pomocą Gradle. Wynikiem budowania jest plik jar, który jest umieszczany w obrazie Dockerowym, który jest wykorzystywany do deployu na klastrze Kubernetesowym. Aplikacja jest bezpieczna, nie ma możliwości dostępu do danych innego użytkownika, a każdy endpoint jest zabezpieczony.

Endpointy

- Konwersacje
  - `POST /conversations` - Endpoint służący do rozpoczynania nowej konwersacji pomiędzy co najmniej dwoma użytkownikami. Przyjmuje listę ID użytkowników jako część żądania i zwraca ID nowo utworzonej konwersacji.
  - `GET /conversations` - Endpoint służący do pobierania listy konwersacji, do których należy zalogowany użytkownik. Zwraca listę konwersacji, w których bierze udział aktualny użytkownik.
- Wiadomości
  - `GET /conversation/{conversationId}/messages` - Endpoint do pobierania wiadomości z określonej konwersacji, zwraca listę wiadomości wraz z informacjami o autorze, treści i dacie utworzenia.
  - `POST /conversation/{conversationId}/messages` - Endpoint do tworzenia nowych wiadomości w danej konwersacji, zwraca ID nowo utworzonej wiadomości.
- Pliki
  - `GET /files/{fileName}` - Endpoint służący do pobierania plików o podanej nazwie z serwera. Zwraca plik jako załącznik do pobrania w formacie ByteArrayResource.
  - `POST /conversation/{conversationId}/files` - Endpoint służący do wysyłania plików w konwersacji. Zwraca ID nowo utworzonej wiadomości zawierającej wysłany plik.
- Użytkownicy
  - `POST /users` - Endpoint służący do tworzenia nowego użytkownika. Oczekuje żądania z nazwą użytkownika i hasłem. Zwraca ID nowo utworzonego użytkownika.
  - `GET /users` - Endpoint do pobierania listy wszystkich użytkowników. Zwraca listę użytkowników zarejestrowanych w systemie.

Frontend

Aplikacja webowa została zbudowana przy pomocy Frameworka Vite z biblioteką React.js.

Do tworzenia spójnego i responsywnego interfejsu użytkownika wykorzystywane są biblioteki Tailwind oraz DaisyUI.

Całość zakodowana jest przy pomocy języka TypeScript, który jest rozszerzeniem JavaScriptu o zaawansowany system typów i interfejsów.

Endpointy backendowe są odpytywane przy użyciu bibliotek Axios i react-query.

Wdrożenie

W katalogu `k8s` projektu znajdują się pliki konfiguracyjne przy użyciu których wdrożono aplikację na prywatny klaster Kubernetes.

Środowisko wdrożniowe składa się z następujących elementów:

- System operacyjny: Ubuntu 22.04
- Specyfikacja maszyny: 4vCPU, 12GB RAM, 50GB SSD
- Lokalizacja: Sztokholm, Szwecja
- Klaster Kubernetes: MicroK8s 1.27

Aby wdrożyć elementy na własny klaster, wystarczy użyć poniższej komendy, znajdując się w katalogu `k8s`:

```
kubectl apply -f . -n "nazwa namespace'u"
```

Oprócz plików konfiguracyjnych Kubernetesa, w katalogu `k8s` znajduje się też plik `keycloak-realm-config.json`, zawierający konfigurację "realmu" w aplikacji Keycloak, której użycie zostało opisane powyżej.