

# Lokalizacja projektu GitHub:

---

<https://github.com/kmichal661/ProjektJavaZaliczenie>

HTTPS clone:

<https://github.com/kmichal661/ProjektJavaZaliczenie.git>

## Logowanie do aplikacji

---

Aplikacja powinna mieć możliwość zalogowania kontem demo gdy jest uruchomiona lokalnie z bazą danych H2

W aplikacji docker baza danych jest zainicjalizowana lecz pusta.

Dane do logowania lokalnie:

```
email: test_user@email.com  
password: SecretPassword123!
```

## Uruchamianie aplikacji

---

### Uruchamianie za pomocą Dockera

Wymagania wstępne

- Zainstalowany Docker i Docker Compose.

Kroki

1. W katalogu głównym projektu uruchom następujące polecenie:

```
docker-compose up --build
```

2. Backend będzie dostępny pod adresem: <http://localhost:8080>.
3. Frontend będzie dostępny pod adresem: <http://localhost:3000>.

Przechowywanie plików

- Pliki przesyłane do aplikacji (np. obrazy nieruchomości) będą zapisywane w katalogu **uploads** w katalogu głównym projektu na Twoim komputerze.

---

## Uruchamianie lokalnie

## Wymagania wstępne

- Zainstalowana Java 21.
- Zainstalowany Node.js w wersji 18 lub nowszej.
- Zainstalowany Maven.

## Uruchamianie Backend

1. Przejdź do katalogu **demo**:

```
cd demo
```

2. Uruchom aplikację Spring Boot:

```
./mvnw spring-boot:run
```

3. Backend będzie dostępny pod adresem: <http://localhost:8080>.

## Uruchamianie Frontend

1. Przejdź do katalogu **FrontEnd/real-estate-app**:

```
cd FrontEnd/real-estate-app
```

2. Zainstaluj zależności:

```
npm install
```

3. Uruchom aplikację w trybie deweloperskim:

```
npm run dev
```

4. Frontend będzie dostępny pod adresem wskazanym w terminalu (domyślnie <http://localhost:5173>).

## Przechowywanie plików

- Pliki przesyłane do aplikacji (np. obrazy nieruchomości) będą zapisywane w katalogu **uploads** w katalogu głównym projektu.

---

## Uwagi

- W przypadku uruchamiania lokalnie upewnij się, że zmienna środowiskowa `UPLOAD_DIR` nie jest ustawiona, aby aplikacja korzystała z domyślnego katalogu `uploads`.
- W przypadku uruchamiania w Dockerze zmienna `UPLOAD_DIR` jest automatycznie ustawiana w pliku `docker-compose.yml`.

# Dokumentacja projektu: Backend

---

## Opis projektu

Projekt **Demo** to aplikacja internetowa stworzona w technologii Spring Boot, która umożliwia zarządzanie ofertami nieruchomości. Aplikacja obsługuje funkcje takie jak rejestracja użytkowników, logowanie, zarządzanie ofertami nieruchomości oraz przesyłanie i pobieranie obrazów związanych z ofertami.

---

## Funkcjonalności

### Użytkownicy

- **Rejestracja użytkownika:** Możliwość tworzenia nowych użytkowników z hasłem szyfrowanym za pomocą BCrypt.
- **Logowanie:** Uwierzytelnianie użytkowników za pomocą JWT (JSON Web Token).
- **Pobieranie listy użytkowników:** Endpoint do pobierania wszystkich użytkowników.
- **Wyszukiwanie użytkownika po e-mailu:** Możliwość wyszukiwania użytkownika na podstawie adresu e-mail.

### Oferty nieruchomości

- **Dodawanie ofert:** Tworzenie nowych ofert nieruchomości.
- **Pobieranie ofert:** Pobieranie listy wszystkich ofert.
- **Pobieranie szczegółów oferty:** Pobieranie szczegółowych informacji o konkretnej ofercie na podstawie jej ID.
- **Aktualizacja ofert:** Edytowanie istniejących ofert.
- **Usuwanie ofert:** Usuwanie ofert na podstawie ich ID.

### Obrazy nieruchomości

- **Dodawanie obrazów do oferty:** Możliwość przesyłania obrazów i przypisywania ich do konkretnej oferty.
  - **Pobieranie obrazów oferty:** Pobieranie listy obrazów przypisanych do konkretnej oferty.
  - **Serwowanie obrazów:** Pobieranie obrazów na podstawie ich nazwy pliku.
- 

## Technologie

### Backend

- **Spring Boot:** Framework do budowy aplikacji internetowych.
- **Spring Security:** Obsługa uwierzytelniania i autoryzacji.
- **Spring Data JPA:** Obsługa bazy danych z wykorzystaniem JPA.

- **H2 Database:** Wbudowana baza danych do celów testowych.
- **JWT (JSON Web Token):** Mechanizm uwierzytelniania użytkowników.

## Baza danych

- **H2 Database:** Używana jako baza danych w trybie plikowym.

## Język programowania

- **Java 21:** Wersja języka Java używana w projekcie.

---

# Struktura projektu

## Główne pakiety

- **com.example.demo.controller:** Zawiera kontrolery REST obsługujące żądania HTTP.
- **com.example.demo.entity:** Zawiera klasy encji reprezentujące tabele w bazie danych.
- **com.example.demo.repository:** Zawiera interfejsy repozytoriów JPA do komunikacji z bazą danych.
- **com.example.demo.security:** Zawiera klasy związane z bezpieczeństwem, takie jak konfiguracja Spring Security i filtry JWT.
- **com.example.demo.util:** Zawiera klasy pomocnicze, np. do generowania tokenów JWT.
- **com.example.demo.dto:** Zawiera klasy DTO (Data Transfer Object) używane do przesyłania danych.

---

# Dokumentacja projektu: FrontEnd

## Opis projektu

Real Estate App to aplikacja internetowa umożliwiająca użytkownikom przeglądanie, dodawanie, edytowanie oraz usuwanie ofert nieruchomości. Aplikacja została stworzona przy użyciu React, TypeScript oraz Chakra UI, a backend komunikuje się z aplikacją za pomocą REST API.

---

## Funkcjonalności

### 1. Rejestracja i logowanie użytkowników

- Użytkownicy mogą się rejestrować i logować, aby uzyskać dostęp do funkcji aplikacji.
- Tokeny autoryzacyjne są przechowywane w **localStorage**.

### 2. Dodawanie ofert

- Użytkownicy mogą dodawać nowe oferty nieruchomości, w tym zdjęcia, opis, cenę, adres i inne szczegóły.

### 3. Przeglądanie ofert

- Główna strona wyświetla listę dostępnych ofert w formie kart.

### 4. Szczegóły oferty

- Użytkownicy mogą przeglądać szczegóły wybranej oferty, w tym zdjęcia, opis, cenę i dane kontaktowe.

## 5. Usuwanie ofert

- Użytkownicy mogą usuwać swoje oferty.

## 6. Tryb jasny/ciemny

- Aplikacja obsługuje przełączanie między trybem jasnym a ciemnym.

---

# Struktura projektu

## Główne katalogi i pliki

- **src/**: Główny katalog aplikacji.
  - **components/**: Komponenty wielokrotnego użytku.
    - **OfferCard.tsx**: Komponent wyświetlający pojedynczą ofertę.
    - **NewOfferDialog.tsx**: Komponent dialogu do dodawania nowych ofert.
    - **Navbar.tsx**: Pasek nawigacyjny z opcją wylogowania i przełączania trybu kolorów.
    - **ui/**: Komponenty interfejsu użytkownika, takie jak **tooltip**, **toaster**, **provider**, i **color-mode**.
  - **Views/**: Widoki aplikacji.
    - **MainView.tsx**: Widok główny wyświetlający listę ofert.
    - **OfferEntityView.tsx**: Widok szczegółów pojedynczej oferty.
    - **Login.tsx**: Widok logowania.
    - **Register.tsx**: Widok rejestracji.
  - **services/**: Logika aplikacji, np. **AuthenticationContext.tsx** do zarządzania autoryzacją.
  - **App.tsx**: Główny komponent aplikacji.
  - **main.tsx**: Punkt wejścia aplikacji.
- **Dockerfile**: Plik konfiguracji Dockera do uruchamiania aplikacji w kontenerze.
- **vite.config.ts**: Konfiguracja Vite.
- **package.json**: Lista zależności i skryptów projektu.

---

# Technologie

- **React**: Biblioteka do budowy interfejsów użytkownika.
- **TypeScript**: Statyczne typowanie dla JavaScript.
- **Chakra UI**: Biblioteka komponentów UI.
- **Formik**: Zarządzanie formularzami.
- **React Router**: Nawigacja w aplikacji.
- **Vite**: Narzędzie do budowy aplikacji.

---

# Instalacja i uruchomienie

## Wymagania wstępne

- Node.js w wersji 18 lub nowszej.

## Screenshots

---

### Rejestracja użytkownika

Vite + React + TS

localhost:3000/register

REAL ESTATE APP

Please register

**Username**  
username  
Provide your username

**Email**  
test\_user@email.com  
Provide your email address

**Name**  
Michał Kubicki  
Provide your name

**Password**  
\*\*\*\*\*  
Provide your password

Register

Already have an account? [Login](#)

### Logowanie użytkownika

Vite + React + TS

localhost:3000/login

REAL ESTATE APP

Please login

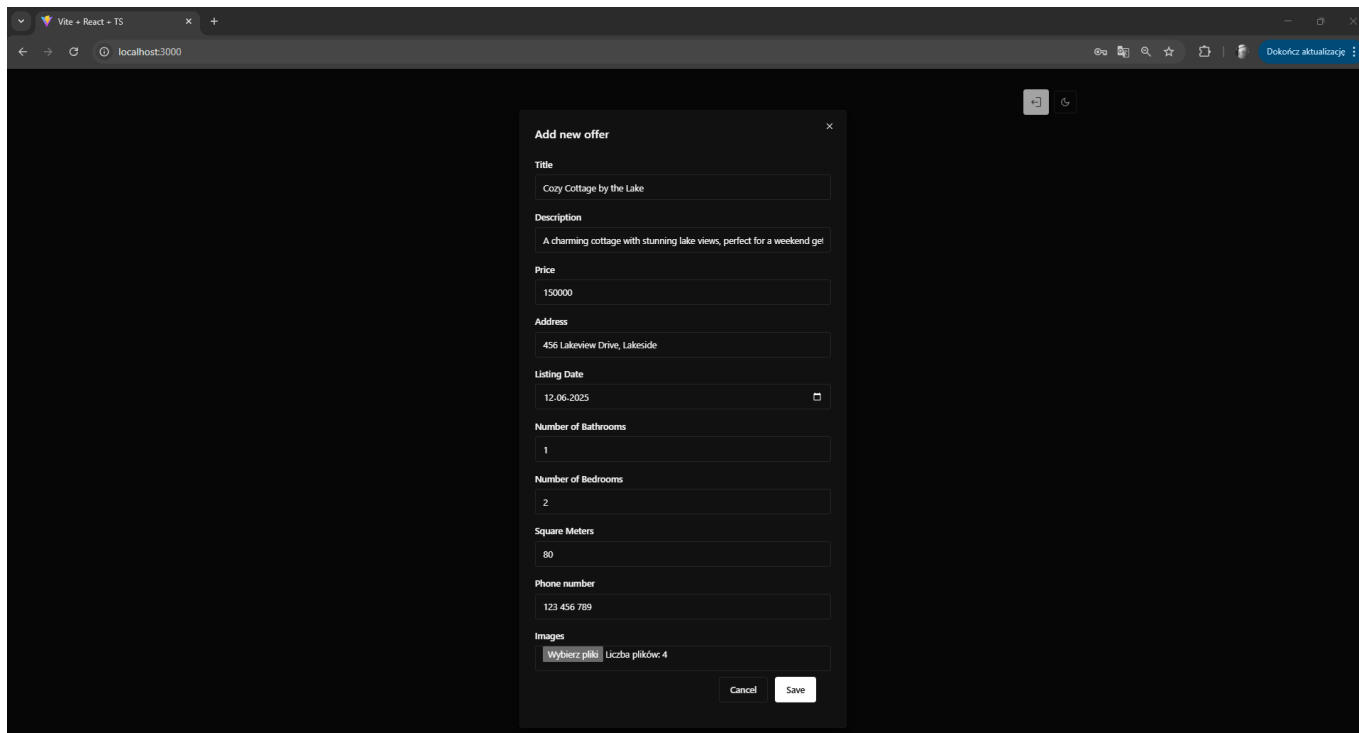
**Email**  
test\_user@email.com  
Provide your email address

**Password**  
\*\*\*\*\*  
Provide your password

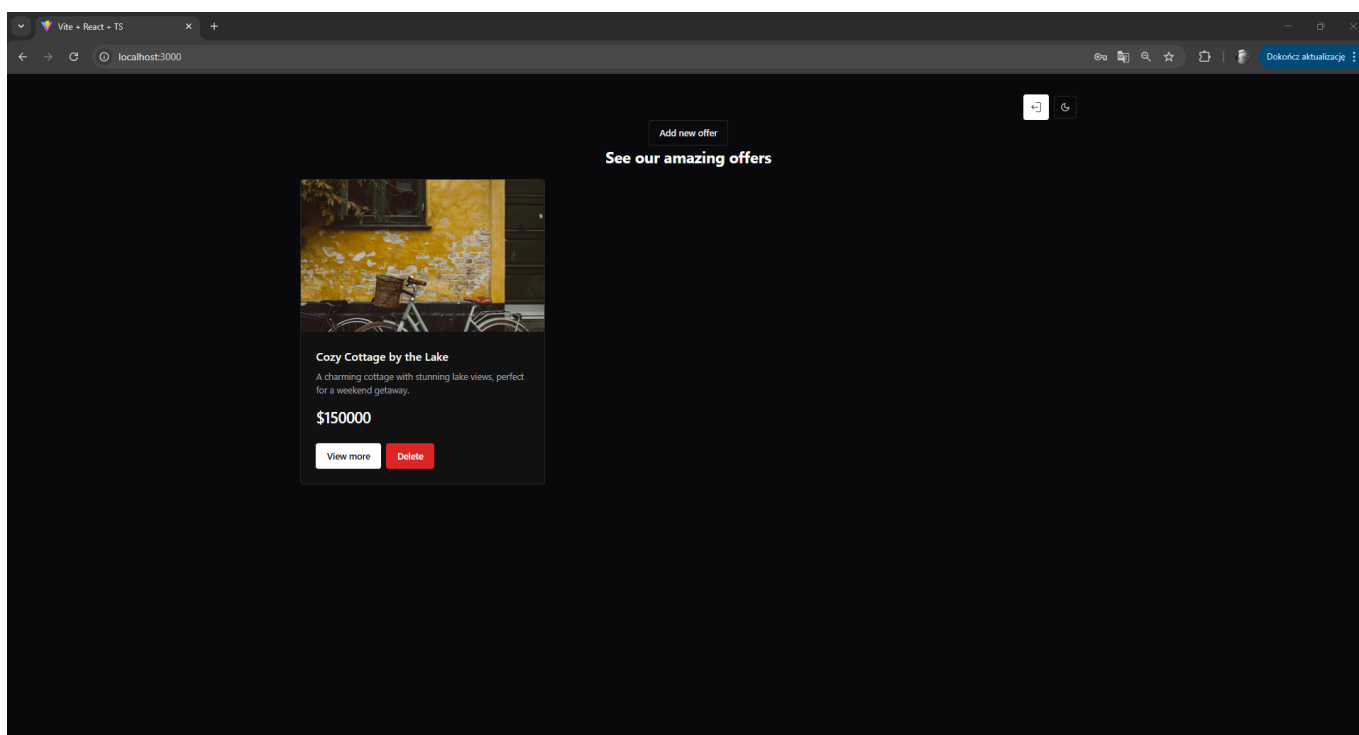
Login

Don't have an account? [Register](#)

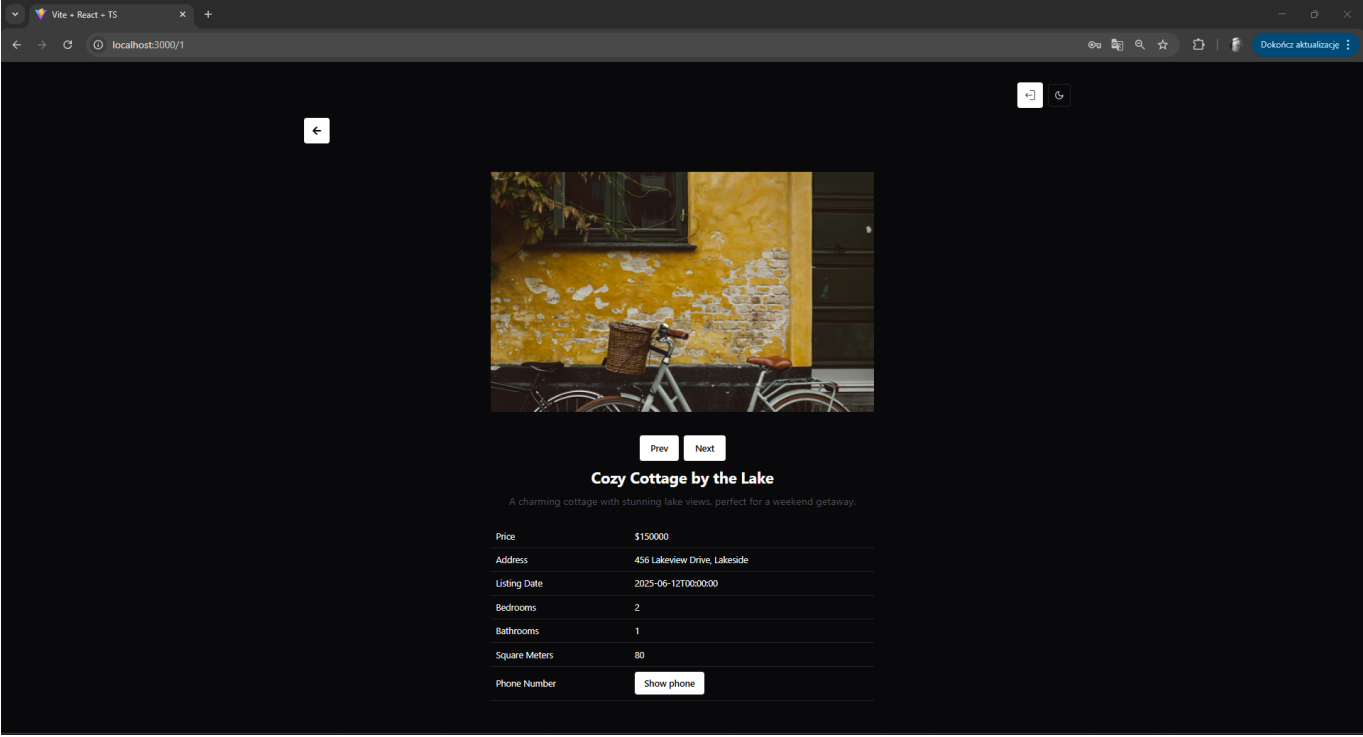
### Dodawanie oferty



## Przeglądanie ofert



## Szczegóły oferty



## Szczegóły oferty z numerem telefonu

