

A G H

cinemaeX

DOKUMENTACJA PROJEKTU
ZAAWANSOWANE TECHNIKI INTERNETOWE

Tomasz Gajda

I. Projekt koncepcji, założenia

1.1. Zdefiniowanie tematu projektu

Tematem mojego projektu jest stworzenie serwisu internetowego dla pierwszego na świecie **kina 12D** o nazwie **cinemaeX**. Serwis ten powinien obsługiwać rezerwacje dla klientów i pozwolić administratorom na manipulacje bazą danych przechowującą informacje ważne dla funkcjonowania usługi.

1.2. Analiza wymagań

Baza danych powinna posiadać takie funkcjonalności jak:

- Rejestracja w serwisie
- Logowanie do serwisu
- Dodawanie i usuwanie nowych reżyserów do bazy
- Dodawanie i usuwanie nowych aktorów do bazy
- Dodawanie i usuwanie nowych filmów do bazy

Użytkownik:

- Przeglądanie filmów
- Przeglądanie aktorów
- Przeglądanie reżyserów

1.3. Zaprojektowanie funkcji

Podstawowymi funkcjami tego serwisu jest łatwe przeglądanie danych z bazy oraz dodawanie i usuwanie rekordów z odpowiednich tabel. W bazie powinny pojawić się również systemy walidujące i przyjemny dla oka interfejs.

II. Projekt bazy

2.1. Zdefiniowanie encji (obiektów) oraz ich atrybutów

- Tabela **users** przechowuje dane na temat użytkowników:
 - **id SERIAL PRIMARY KEY** - unikalny identyfikator użytkownika,
 - **username** - login użytkownika,
 - **name VARCHAR** - imię użytkownika,
 - **surname VARCHAR** - nazwisko użytkownika,
 - **email VARCHAR** - email użytkownika, służy on między innymi do logowania, może być wykorzystany do przesyłania newslettera,
 - **newsletter BOOLEAN** - czy użytkownik chce otrzymywać newsletter serwisu,
- Tabela **movie** przechowuje dane na temat filmów:
 - **id_movie SERIAL PRIMARY KEY** - unikalny identyfikator filmu,
 - **id_director FOREIGN KEY** - unikalny identyfikator reżysera filmu,
 - **movie_title VARCHAR** - tytuł filmu,
 - **movie_description VARCHAR** - opis filmu,
 - **movie_release VARCHAR** - data premiery filmu,
 - **movie_duration VARCHAR** - czas trwania filmu,
 - **movie_image VARCHAR** - link do plakatu filmu
- Tabela **director** przechowuje dane na temat reżyserów:
 - **id_director SERIAL PRIMARY KEY** - unikalny identyfikator reżysera,
 - **director_name VARCHAR** - imię reżysera,
 - **director_surname VARCHAR** - nazwisko reżysera,
 - **director_description VARCHAR** - opis reżysera,
 - **director_img VARCHAR** - link do zdjęcia reżysera
- Tabela **actor** przechowuje dane na temat aktorów:
 - **id_actor SERIAL PRIMARY KEY** - unikalny identyfikator aktora,
 - **actor_name VARCHAR** - imię aktora,
 - **actor_surname VARCHAR** - nazwisko aktora,
 - **actor_description VARCHAR** - opis aktora,
 - **actor_img VARCHAR** - link do zdjęcia aktora

Do prezentacji **dat i godzin** wykorzystałem typ **varchar** zamiast **date**, ponieważ oszczędza to wiele zbędnych operacji na danych po stronie serwera i klienta.

III. Projekt funkcjonalny

3.1 Interfejsy do prezentacji, edycji i obsługi danych

Edycja i obsługa danych jest możliwa jedynie z poziomu panelu admina.

- Dodawanie i usuwanie aktorów

Użytkownik może manipulować danymi dotyczącymi aktorów. Może ich dodawać i usuwać.

The screenshot shows a user interface for managing actors. At the top, there are four tabs: 'Actors Panel' (highlighted with a blue border), 'Directors Panel', 'Movies Panel', and 'Screenings Panel'. Below the tabs, the 'Actors Panel' section is active, displaying the 'Add an actor' form. This form includes fields for 'Name', 'Surname', 'Description' (with a text area), and 'Image URL'. A large purple 'Add an actor' button is centered below these fields. Below this section, the 'Delete an actor' section is shown, featuring a dropdown menu with 'Jeffrey Dean Morgan' selected and a purple 'Delete an actor' button at the bottom.

- **Dodawanie i usuwanie reżyserów**

Użytkownik może manipulować danymi dotyczącymi reżyserów. Może ich dodawać i usuwać. Za pomocą wyzwalaczy dbamy o to by wraz z usuniętymi reżyserami, usuwane były filmy, które zostały przez nich wyreżyserowane.

The screenshot shows a user interface for managing directors. At the top, there are four tabs: "Actors Panel", "Directors Panel" (which is highlighted in blue), "Movies Panel", and "Screenings Panel".

Add a director: This section contains four input fields: "Name", "Surname", "Description", and "Image URL". Below these fields is a blue button labeled "Add director".

Delete a director: This section features a dropdown menu containing the name "Cary Joji Fukunaga". Below the dropdown is a blue button labeled "Delete a director".

- **Dodawanie i usuwanie filmów**

Administrator może manipulować danymi dotyczącymi filmów. Może je dodawać i usuwać.

[Actors Panel](#) [Directors Panel](#) [Movies Panel](#) [Screenings Panel](#)

Add a movie

Title

Description

Cary Joji Fukunaga

▼

Release Date

Duration

Image URL

Add a movie

Delete a movie

No Time To Die

▼

Delete a movie

3.2 Wizualizacja danych

Dane można oglądać z poziomu każdej roli - możemy oglądać:

- Listę aktorów

W zakładce **Actors** możemy oglądać wszystkich aktorów znajdujących się w bazie oraz poprzez kliknięcie w obrazek aktora możemy zobaczyć dodatkowe informacje na jego temat.

The screenshot shows the 'Actors' section of the cinemaex website. At the top left is a back arrow icon, at the top right is a menu icon. The title 'Actors' is centered above three actor profiles. Each profile consists of a small portrait photo followed by the actor's name. Below the profiles are three horizontal navigation arrows.

Actor	Description
Jeffrey Dean Morgan	American actor of film and television. He is known for his roles as John Winchester in the fantasy horror series <i>Supernatural</i> (2005–2007; 2019), Denny Duquette in the medical drama series <i>Grey's Anatomy</i> (2006–2009), the Comedian in the superhero film <i>Watchmen</i> (2009), Jason Crouse in the political drama series <i>The Good Wife</i> (2015–2016), Negan in the horror drama series <i>The Walking Dead</i> (2016–present), and Harvey Russell in <i>Rampage</i> (2018).
Andrew Lincoln	British actor best known for his role as Rick Grimes in the television series <i>The Walking Dead</i> .
Chris Hemsworth	Australian actor known for his roles in films like <i>Thor</i> , <i>Avengers: Endgame</i> , and <i>Guardians of the Galaxy</i> .

The screenshot shows the detailed profile page for Jeffrey Dean Morgan. At the top left is a back arrow icon. The title 'Jeffrey Dean Morgan' is prominently displayed. Below the title is a section titled 'Additional Information' containing a biography. To the right of the text is a large portrait photo of Jeffrey Dean Morgan.

Additional Information

American actor of film and television. He is known for his roles as John Winchester in the fantasy horror series *Supernatural* (2005–2007; 2019), Denny Duquette in the medical drama series *Grey's Anatomy* (2006–2009), the Comedian in the superhero film *Watchmen* (2009), Jason Crouse in the political drama series *The Good Wife* (2015–2016), Negan in the horror drama series *The Walking Dead* (2016–present), and Harvey Russell in *Rampage* (2018).

- Listę reżyserów

W zakładce **Directors** możemy oglądać wszystkich reżyserów znajdujących się w bazie oraz poprzez kliknięcie w obrazek reżysera możemy zobaczyć dodatkowe informacje na jego temat.

cinema^eX.

☰

←

Directors



David Lynch Martin Scorsese Steven Soderbergh

←

David Lynch

[Additional Information](#)

Born in 1946 in Missoula, Montana, David Lynch was raised in small-town America. After high school, he went to Boston to attend the School of the Museum of Fine Arts. Shortly after that, he planned a three-year trip to Europe to work on his art, but didn't take to it and left after 15 days. In 1977, he released his first film *Eraserhead* (1977), which, although not critically acclaimed, was noticed by many people, including Francis Ford Coppola, who was rumored to have screenings of it for his cast and crew on the *Apocalypse Now* (1979) set.



- Listę filmów

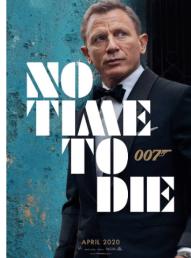
W zakładce **Movies** możemy oglądać wszystkie filmy znajdujące się w bazie oraz poprzez kliknięcie w plakat filmu możemy zobaczyć dodatkowe informacje na jego temat.

cinema^eX.

≡

←

Movies



No Time To Die

Release date: 2020-04-02
Duration: 110 min



Wander

Release date: 2020-11-09
Duration: 98 min



Ammonite

Release date: 2020-10-16
Duration: 106 min

←

Wander

Release: 2020-11-09 Duration: 98 min Director: TBA TBA

Arthur Bretnik is a mentally unstable conspiracy theorist and private eye with a traumatic past. After being hired to investigate a possible murder cover up in the small town of Wander, Arthur is plunged into a world of lies and deceit, as he quickly suspects the murder may be part of the same "conspiracy cover up" that caused the death of his daughter. Increasingly paranoid, Arthur's sanity is tested as he attempts to filter fact from fiction and solve the case, all the while questioning if he is a pawn in a much bigger game.

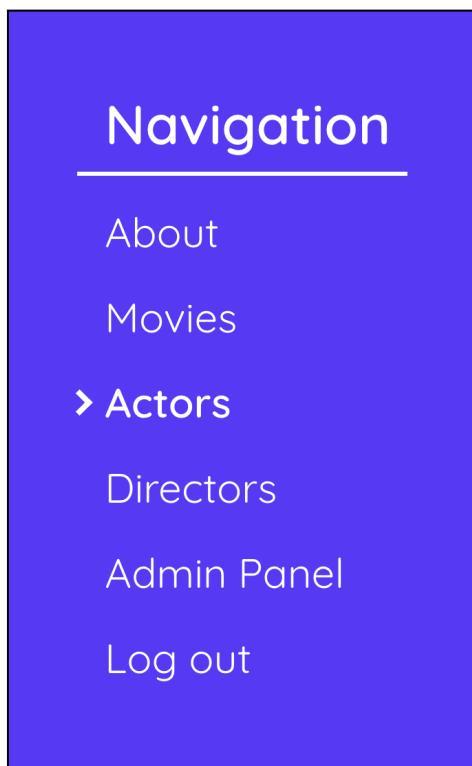


3.3 Zdefiniowanie panelu sterowania aplikacji

Głównym panelem sterowania jest nawigacja strony. By ją otworzyć, należy kliknąć przycisk w prawym górnym rogu strony. Jest on dostępny zawsze, natomiast w zależności od tego czy jesteśmy **zalogowani** czy też nie, mamy dostęp do innej ilość podstron. Panel jest w formie **wertykalnej**, w celu lepszej responsywności.

(Strona jest responsywna i powinna wyglądać dobrze na ekranach różnych rozdzielczości)

Gdy jesteśmy niezalogowani możemy dostać się tylko do **panelu logowania** lub do **panelu rejestracji**.



IV. Dokumentacja

4.1 Wprowadzanie danych

Więcej danych wprowadzać możemy ręcznie poprzez formularze ze strony panelu admina.

4.2 Dokumentacja użytkownika

Strona jest **responsywna**, co oznacza, że dostosowana jest do różnych rozmiarów i można na nią wygodnie wejść z przeglądarki internetowej różnych urządzeń - **komputera, tableta** czy też **telefonu**.

1. Użytkownik wchodzi na stronę o linku: <https://cinemaex.netlify.app/>
Proszę o chwilkę cierpliwości ponieważ **heroku** potrzebuje chwili na pierwsze załadowanie. Włączenie strony po raz kolejny zajmie już dużo mniej. Więcej o deploymencie opiszę w następnym rozdziale.
2. Użytkownik po wejściu na stronę ma tylko dwie opcje przejścia do dalszej części aplikacji: **logowanie** lub **rejestrację**. Do tych ekranów możemy przejść klikając jeden z przeznaczonych do tego przycisków lub przechodząc do nich z ekranu nawigacji otwieranego poprzez przycisk w prawym górnym rogu.
3. Użytkownik może założyć nowe konto lub też zalogować się do istniejącego.

Przykładowe konto:

MAIL: **user@cinemaex.com**

HASŁO: **user**

4. Po zalogowaniu/zarejestrowaniu ukazuje nam się lista filmów dostępnych w ramach aplikacji. Z tego ekranu można przejść do innych za pomocą nawigacji (**prawy górny róg**).
5. Obsługa nawigacji: po kliknięciu przycisku, ukazuje nam się po animacji niebieski ekran, na którym możemy zobaczyć wszystkie dostępne ścieżki:

- **About** - przekierowuje do strony internetowej autora aplikacji, nie jest faktyczną częścią projektu,
- **Movies** - ekran na którym ujrzeć możemy wszystkie filmy z bazy danych. Możemy się po nim poruszać za pomocą scroll'a.

Dodatkową funkcją jest możliwość kliknięcia w każdy z filmów, co otworzy nam ekran pojedynczego filmu z dokładniejszymi informacjami na jego temat.

- **Actors** - ekran na którym ujrzeć możemy wszystkich aktorów z bazy danych. Możemy się po nim poruszać za pomocą scroll'a.

Dodatkową funkcją jest możliwość kliknięcia w każdego z aktorów, co otworzy nam ekran pojedynczego aktora z dokładniejszymi informacjami na jego temat.

- **Directors** - ekran na którym ujrzeć możemy wszystkich reżyserów z bazy danych. Możemy się po nim poruszać za pomocą scroll'a.

Dodatkową funkcją jest możliwość kliknięcia w każdego z reżyserów, co otworzy nam ekran pojedynczego reżysera z dokładniejszymi informacjami na jego temat.

- **Admin Panel** - ekran na którym ujrzeć formularze służące do dodawania i usuwania rekordów z bazy. Za pomocą przycisków na górze ekranu możemy przełączać się pomiędzy odpowiednimi panelami:

(Actors Panel, Directors Panel, Movies Panel).

Każdy **“podpanel”** przedstawia możliwości modyfikacji odpowiedniej części bazy danych.

- **Log out** - pozwala na wylogowanie się ze swojego konta i powrót do ekranu logowania.

4.3 Opracowanie dokumentacji technicznej

Technologie wykorzystane w projekcie:

Komunikacja pomiędzy serwerem a klientem odbywa się za pomocą stylu architektury **REST** (Representational state transfer). Po stronie serwera mamy określone ścieżki do których odwołujemy się od strony klienta po to by **żądać** dane ze strony serwera. Połączenie pomiędzy **serwerem** a **bazą** jest utrzymywane poprzez **Springa**. Pozwala nam ono na wysyłanie zapytań do bazy i pobieranie z niej danych. Inne technologie:

Po stronie klienta:

- **React** - biblioteka języka Javascript, służąca do budowania interfejsów użytkownika w postaci poszczególnych komponentów,
- **Axios** - klient HTTP przeznaczony do wykonywania zapytań do serwera,
- **Styled-components** - biblioteka pozwalająca na korzystanie z CSS'owych stylów wewnętrz React'owych (Javascript'owych) komponentów,

Po stronie serwera:

- **Java**
- **Spring Boot**
- **heroku** - PaaS - platforma w chmurze dla hostowania aplikacji
- **JWT** - Json Web Token - technologia pozwalająca na ograniczanie dostępu do strony i autoryzację

Po stronie bazy:

- **PostgreSQL** - relacyjna baza danych
- **ElephantSQL** - DaaS - baza w chmurze

Obsługa żądań

Aplikacja serwerowa składa się z kilku rodzajów obiektów - są to **kontrolery, modele, requesty oraz repozytoria**. Kontrolery przedstawiają ścieżki dostępne w ramach naszego REST API. Czy to ścieżka "**/api/movies/**" zwracająca całą listę dostępnych filmów, czy to "**/api/movies/delete/1/**" usuwająca pierwszy film z bazy - oba zachowania opisane są w kontrolerze stworzonym dla filmów.

Modele przedstawiają zawartość tabel odpowiadających głównym podmiotom aplikacji tj. **User'om, filmom i reżyserom**.

Requesty pokazują czego oczekujemy od klienta w zapytaniach. Dzięki nim wiemy jak wypełnić zawartość request'u by osiągnąć zamierzony wynik.

Repozytoria natomiast trzymają wszystkie zapisane w tabelach dane i dzięki nim możemy się dowiedzieć czy poszukiwany przez nas element znajduje się w bazie czy też nie.

Autoryzacja

Do autoryzacji wykorzystywany jest **JWT Token** obsługiwany przez bibliotekę Spring i jej dodatki. Token jest tworzony w momencie logowania lub rejestracji. Zawiera w środku zaszyfrowane (wybranym przez nas **secretem**) ID użytkownika. Wygenerowany **token** przesyłany jest do klienta w momencie logowania lub też rejestracji, zawiera on wszystkie informacje potrzebne do udzielenia dostępu do reszty strony. **Token** po stronie klienta jest zapisywany w **localStorage**, skąd przy każdym żądaniu jest kopowany i przesyłany wraz zapytaniem do serwera.

Zapytania ze strony klienta

Po stronie klienta korzystamy z narzędzia o nazwie **Axios**, ponieważ jest on najbardziej wspierany przez większość przeglądarek oraz jest dość prosty w użyciu. Przy każdej zmianie strony wysyłamy zapytanie do serwera korzystając z określonej ścieżki. Przykładowo, poniżej znajduje się przykładowe zapytanie do serwera, chcąc dostać w

odpowiedzi listę obiektów aktorów. W nagłówku przesyłamy również **token**, który przejdzie przez **middleware** służący do autoryzacji.

```
const getActors = (): Promise<ActorPreviews> => {
  return new Promise((resolve, reject) => {
    axios
      .get('/actors', {
        headers: {
          token: localStorage.token,
        },
      })
      .then(({ data }) => resolve(data))
      .catch((err) => reject(err));
  });
};
```

Dane pobierane są za pomocą Reactowego **hooka** **useAsync**, który pakuje dane do zmiennych destrukturyzując je na trzy zmienne.

```
const { status, value, error } = useAsync<ActorPreviews>(getActors);
```

Status zawiera informację o statusie zapytania - możliwe opcje :

- **idle, pending, success, error**

Value zawiera elementy które pobraliśmy z bazy:

- **wypełnia się dopiero gdy status jest 'success'**

Error zawiera potencjalny błąd

- **używany jedynie gdy błąd wystąpi**

4.4 Instrukcja lokalnego uruchomienia

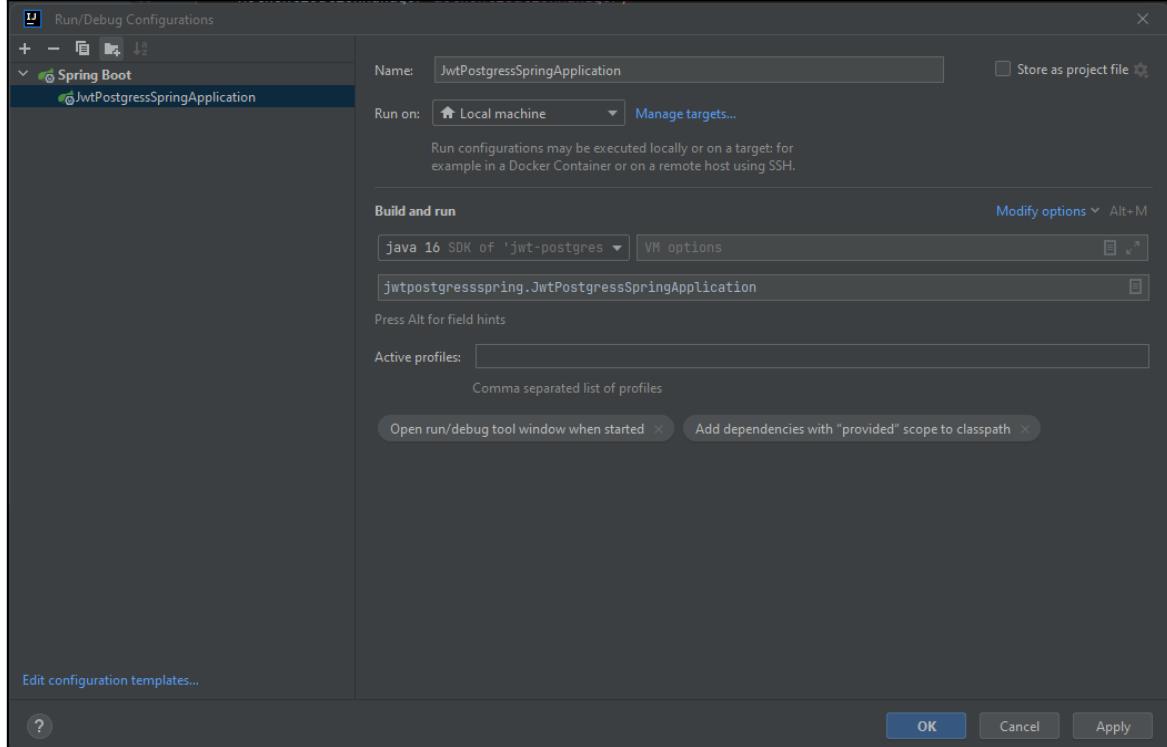
Uruchomienie aplikacji klienta

Wchodzimy w folder **client** i korzystamy z komendy **npm install** by pobrać wszystkie potrzebne nam zależności i biblioteki. Po tym używamy komendy **npm start** by uruchomić lokalny serwer. Na adresie **localhost:3000** dostępny będzie interfejs aplikacji.

Uruchomienie aplikacji serwerowej

W celu włączenia serwera potrzebne jest środowisko IntelliJ IDEA oraz zainstalowanym pakietem Java SDK w wersji **16**.

Główną klasą projektu jest **jwtpostgressspring.JwtPostgressSpringApplication**, dlatego środowisko uruchomieniowe należy skonfigurować w sposób przedstawiony na poniższym zdjęciu:



Następnie uruchamiając naszą główną klasę uruchamiamy projekt i możemy na nim pracować.

V. Wdrożenie

5.1 Wdrożenie aplikacji klienta

Interfejs aplikacji został wdrożony za pomocą platformy **Netlify**, która udostępnia bardzo przyjemne API konsolowe pozwalające w bardzo prosty sposób udostępnić statyczną zawartość w postaci dostępnej w przeglądarce strony internetowej z URL posiadającym dopisek **netlify.app**.

5.2 Wdrożenie aplikacji serwerowej

Do wdrożenia aplikacji serwerowej skorzystałem z **Heroku** (proszę o ewentualne odjęcie punktów za wykorzystanie Heroku zamiast np. Azure). Nie uważam że jest to najlepszy wybór, ale spędziłem niestety sporo czasu próbując wdrożyć aplikację za pomocą Azure czy też AWS, jednak pojawiało się z tym wiele różnorakich problemów, które były dość ciężkie do załatwania. Dlatego też postanowiłem powołać się na klasykę i udostępnić nasze REST API w formie aplikacji udostępnionej przez Heroku.

VI. Literatura

6.1 Literatura

Do stworzenia aplikacji skorzystałem z:

- Treści laboratoriów
- Dokumentacja Postgresql - <https://www.postgresql.org/docs/9.2/index.html>
- Dokumentacja React - <https://reactjs.org/docs/getting-started.html>
- Dokumentacja Spring Boot -
<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- Dokumentacje pomniejszych bibliotek
- Wikipedia - <https://en.wikipedia.org/wiki/> (informacje o aktorach, filmach itp.)