**Aplikacja do recenzji filmów**

**Autor: Janusz Wdówka 236460**

Spis treści

[1. Zarys projektu 3](#_Toc150275784)

[2. Diagram komunikacji 3](#_Toc150275785)

[3. Utworzenie kontenera Mongo DB 4](#_Toc150275786)

[4. Inicjalizacja projektu Java 6](#_Toc150275787)

[4.1. Połączenie z bazą danych 7](#_Toc150275788)

[4.2. Utworzenie klas entity 8](#_Toc150275789)

[4.3. Utworzenie połączenia z bazą 9](#_Toc150275790)

[4.4. Utworzenie struktury dla obiektu recenzji 13](#_Toc150275791)

[5. Test stworzonych metod 15](#_Toc150275792)

[5.1. Get all movies 15](#_Toc150275793)

[5.2. Get by ID 16](#_Toc150275794)

[5.3. Get by IMDB ID 17](#_Toc150275795)

[5.4. Create new Review 18](#_Toc150275796)

[6. FRONTEND 20](#_Toc150275797)

[6.1. utworzenie połączenia reacta z javą 21](#_Toc150275798)

[6.2. Wyświetlenie plakatów 22](#_Toc150275799)

[6.3. Poprawa wyglądu 25](#_Toc150275800)

[6.4. Odtwarzacz trailera 25](#_Toc150275801)

[6.5. Strona wystawiania opinii 26](#_Toc150275802)

[7. API 27](#_Toc150275803)

[7.1. Połącznie aplikacji backendowej z TMBD API 27](#_Toc150275804)

[7.2. Pobrania danych po tytule 29](#_Toc150275805)

[7.3. Pobrania po Id filmu z poprzedniego calla: 30](#_Toc150275806)

[7.4. Pobrania danych o zdjęciach filmu 31](#_Toc150275807)

[7.5. Pobrania danych o trailerach 32](#_Toc150275808)

[7.6. Tworzenie filmów 33](#_Toc150275809)

[8. Utworzenie konteneru backendowego i frontendowego 36](#_Toc150275810)

[8.1. Kontener aplikacji backendowej 36](#_Toc150275811)

[8.2. Kontener aplikacji frontendowej 37](#_Toc150275812)

# Zarys projektu

Celem projektu jest stworzenie aplikacji webowej do wystawiania recenzji filmów z wykorzystaniem poniższych technologii:

* React – frontend aplikacji odpowiedzialny za warstwę prezentacji aplikacji,
* Java z Springiem – backend aplikacji odpowiedzialny za komunikację między frontendem a bazą danych oraz połączenie z TMBD API,
* Mongo DB – system bazodanowy przechowujący informację,
* TMBD API – Zewnętrzne api przechowujące dane o filmach z IMBD.

Aplikacja jest stroną, gdzie każdy może wystawić recenzje o filmie pisząc co o nim uważa. Jest to odpowiednik filmWeba jednak bez ocen, aby użytkownicy mogli określić wyłącznie swoją opinie na podstawie konstruktywnych wypowiedzi.

# Diagram komunikacji

Poniżej został przedstawiony system komunikacji 3 osobnych kontenerów.

Obraz zawierający tekst, zrzut ekranu, diagram, Czcionka

Opis wygenerowany automatycznie

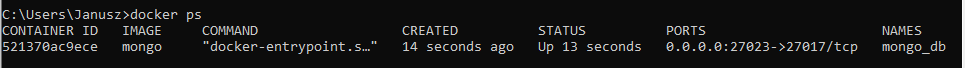
# Utworzenie kontenera Mongo DB

Aby utworzyć docker Mongo DB należy najpierw pobrać dane obrazu mongo, a następnie utworzyć docker, który będzie przechowywać nasze dane.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Widok utworzonego konteneru:

Link, przez który możemy połączyć się z bazą:

mongodb://localhost:27023

Aby połączyć się z bazą danych należy uruchomić kontener, a następnie aplikację do komunikacji z bazą MongoDB np. MongoDB Compass i tam wprowadzić dane do połączenia.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, komputer

Opis wygenerowany automatycznie

Po połączeniu się do bazy należy utworzyć nową bazę danych. Następnie stworzyć kolekcje i wgrać do niej na przykład dane z JSONa.

Widok po wgraniu przykładowych danych:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

# Inicjalizacja projektu Java

Aby zainicjalizować projekt w springu wykorzystujemy stronę <https://start.spring.io>, aby przyśpieszyć proces tworzenia. Projekt będzie wykorzystywać strukturę Maven i wersję 17 Javy.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Widok po zainicjalizowaniu pustego projektu (aby połączyć się z bazą należy użyć adresu: localhost:8080):

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Połączenie z bazą danych

Aby skonfigurować połączenie w application.properties należy ustawić wartości dla spring.data.mongodb.database które zawiera nazwę bazy do której się łączymy oraz spring.data.mongodb.uri, które zawiera link przez który się łączymy do bazy. W naszym przypadku będzie to:

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, Ikona komputerowa

Opis wygenerowany automatycznie

Aby zabezpieczyć dane tworzymy plik .env który zawiera te informacje i w application.properties podmieniamy te wartości na zmienne z pliku .env, który nie znajduje się w repozytorium, jednak dodajemy plik .env.example który zawiera zmienne które musimy zadeklarować w pliku .env.

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, Czcionka

Opis wygenerowany automatycznie



Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, oprogramowanie

Opis wygenerowany automatycznie

Aby móc odwoływać się do plików .env należy dodać w pom.xml dependency do pakietu obsługującego dany typ plików. Poniżej przykładowa implementacja jednego z takich pakietów:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Utworzenie klas entity

Aby móc komunikować się z bazą danych konieczne jest stworzenie odpowiednich klas które reprezentują dane tzw. klasy entity. Poniżej zostały ukazane klasy reprezentujące film i recenzję. Konieczne jest dodanie adnotacji @Document z nazwą kolekcji do której się odwołuje oraz dodanie wszystkich pól występujących w bazie danych na danym obiekcie.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

## Utworzenie połączenia z bazą

Następnym krokiem do komunikacji danych z bazą mongoDB należy stworzyć interfejsy z adnotacją @Repository oraz rozszerzeniem o MongoRepository<Nazwa klazy entity, ObjectId>.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wykorzystałem również w projekcie model Controller-Service. Klasa Controller zawiera adnotację @RequestMapping, który zawiera adres za pomocą którego łączymy w aplikacji string do danej klasy oraz @RestController, które służy do określenia że klasa obsługuje połączenia REST. Klasa ta dla ukrycia części implementacji odwołuje się do funkcji z klasy MovieService.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Klasa MovieService, za pomocą interfejsu MovieRepository pobiera i dodaje dane z bazy danych.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Przykład użycia (localhost:8080/api/movies):

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Aby pobierać dane z bazy mongoDB w specyficzny sposób należy stworzyć w MovieRepository nową funkcję które określają, co realizują w poniższym przypadku pobierają konkretny film na podstawie IMDBID.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Domyślnie funkcje findAll oraz findById są wbudowane w interfejs, który łączy się z bazą i ich implementować ręcznie.

Obraz zawierający tekst, zrzut ekranu, Czcionka

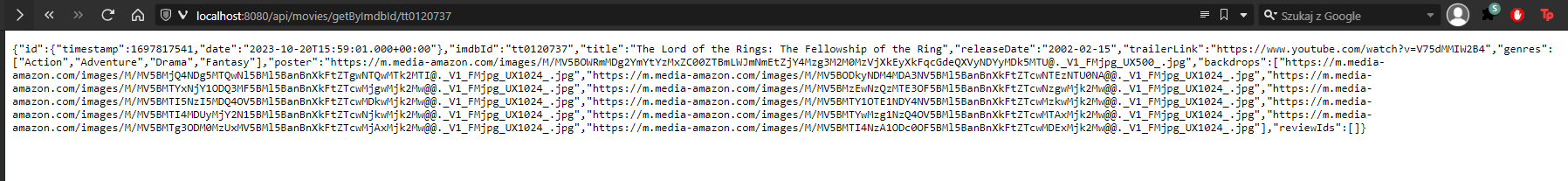
Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Przykład użycia:

* Localhost:8080/api/movies/getByImdbId/{ImbdId filmu}



* Localhost:8080/api/movies/getById/{Id filmu z bazy danych}



## Utworzenie struktury dla obiektu recenzji

Kolejnym etapem jest stworzenie takiej samej struktury dla obiektu Review. Aby stworzyć taki obiekt w bazie wykorzystujemy metodę insert z interfejsu oraz aktualizujemy obiekt Movie za pomocą MongoTemplate z używaniem metod odpowiadających klasycznemu query z SQLa.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

# Test stworzonych metod

## Get all movies

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Get by ID

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Get by IMDB ID

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Create new Review

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

# FRONTEND

Aby utworzyć podstawową strukturę projektu React należy wprowadzić poniższe komendy w terminalu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Następnie żeby uruchomić projekt należy wprowadzić komendę:

npm start

Dostęp do niej jest w poniższym linku:

<http://localhost:3000/>

## utworzenie połączenia reacta z javą

do kontrolerów w javie należy dodać nową adnotację @CrossOrigin, które określają dodatkowe źródła które, mogą się odwołać do kontrolerów.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

W reactcie należy utworzyć plik komunikacji, w którym określamy połączenie z aplikacją bacekndową.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Finalnie w App.js można zaimplementować logikę pobierania danych z bazy za pomocą kontrolera w Javie:



## Wyświetlenie plakatów

Aby wyświetlić pobrane dane o plakatach stworzyłem komponent Movie, który przechowuje dane o filmach.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Komponent ten jest przechowywany w komponencie Home, który reprezentuje główną cześć strony startowej.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Następnie w pliku App.js należy stworzyć początkową strukturę, która zawiera podstawowe ciało strony, które znajduje się w pliku index.js, która jest renderowana jako początek aplikacji.

App.js:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

index.js:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, plakat, fikcja, Film akcji

Opis wygenerowany automatycznie

## Poprawa wyglądu

Po ostylowaniu stworzonych komponentów home page wygląda następująco:

Obraz zawierający tekst, zrzut ekranu, niebo, Gra komputerowa

Opis wygenerowany automatycznie

## Odtwarzacz trailera

Aby wyświetlić filmy w aplikacji React wykorzystałem ReactPlayer, który umożliwia otwieranie filmów pod podanym adresem URL. Są one otwierane po kliknięciu w czerwony przycisk który był widoczny na grafice wyżej.

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

Obraz zawierający zrzut ekranu, śnieg, góra, natura

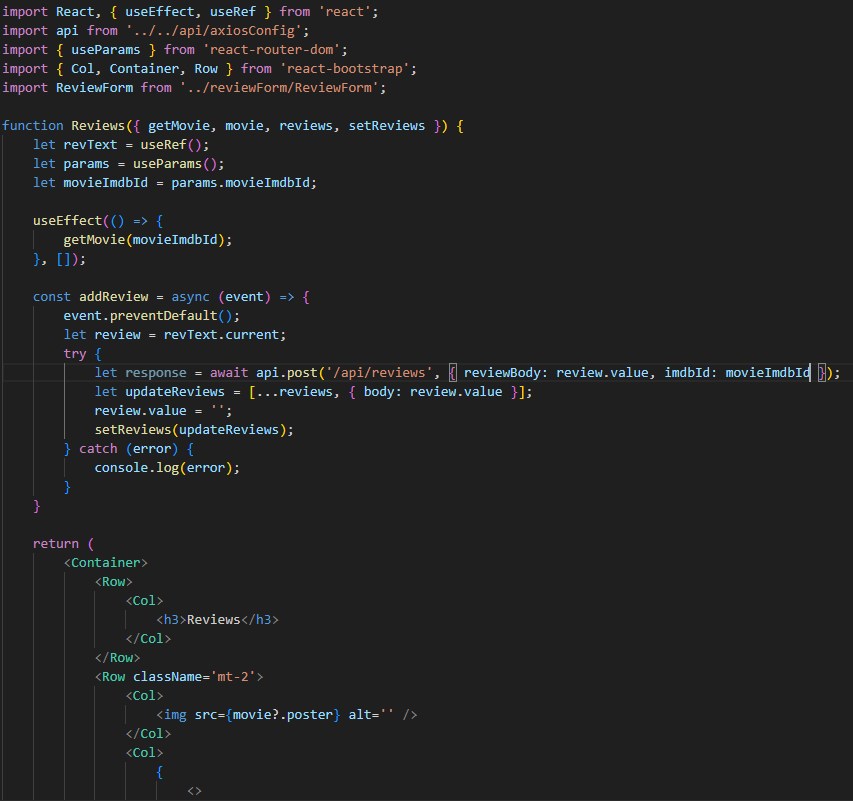
Opis wygenerowany automatycznie

## Strona wystawiania opinii

Do wystawiania opinii wykorzystałem Form z react-bootstrapa, który znajduje się w komponencie ReviewForm, który następnie jest dodawany do Reviews, który zawiera całą stronę do wystawiania recenzji o filmie i wyświetlania pozostałych opinii o nim.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie



Obraz zawierający zrzut ekranu, Oprogramowanie multimedialne, tekst, multimedia

Opis wygenerowany automatycznie

# API

Do pobierania danych z zewnętrznego API wykorzystałem TMBD API którego dokumentacja znajduje się pod poniższym adresem:

<https://www.themoviedb.org>

Do pobrania pełnej ilości potrzebnych danych do aplikacji koniecznie jest wykorzystanie 4 calli do tego API, które są opisane poniżej od kroku 7.2.

## Połącznie aplikacji backendowej z TMBD API

Aby móc uzyskać dostęp do systemu TMBD koniecznie jest założenie konta na platformie oraz wygenerowanie klucza do API.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Powyższy klucz API będzie nam służyć jak autentykator w systemie zewnętrznym.

Następnym krokiem jest stworzenie ponownie struktury Controller-Service. Controller jest otwarty na połączenia z aplikacji Reactowej.

Część kodu Controllera:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

W Klasie Service, wykonujemy połączenie pod odpowiedni adres, aby pobrać potrzebne dane za pomocą HttpRequestów, z których na koniec wyciągane jest ciało i przekazywane do Controllera a potem do aplikacji React. Oczywiście połączenia te wymagają autentykacji za pomocą klucza z systemu TMBD.

Część kodu klasy Service z pobieraniem danych.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

## Pobrania danych po tytule

Pierwszym z koniecznych do wykonani calli jest pobrania danych po tytule filmu. Z którego uzyskujemy takie dane jak tytuł, data premiery oraz adres plakatu. Uzyskujemy również ID filmu w systemie TMBD.

Aby wykonać to połączenie używamy linku o danym szablonie w aplikacji: localhost:8080/api/api/getByName/{Nazwa%Filmu}

title: title

releaseDate: release\_date

poster: <https://image.tmdb.org/t/p/w500> + poster\_path

Przykład pozyskanych danych:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Pobrania po Id filmu z poprzedniego calla:

Kolejnym krokiem jest pobranie IMDBID filmu oraz gatunków, jakie sobą reprezentuje dana ekranizacja.

Aby wykonać to połączenie używamy linku o danym szablonie w aplikacji: localhost:8080/api/api/getById/{TMDB\_ID\_FILMU}

imdbId: imdb\_id

genres: genres – z każdej wartości tablicy wyciągnięte pole name

Przykład pozyskanych danych:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Pobrania danych o zdjęciach filmu

Następnie możemy pobrać również zdjęcia, które posiada dany film. Do wykonania tego połączenia w naszej aplikacji wykorzystujemy połączenie o podanym szablonie:

localhost:8080/api/api/getImagesById/{TMDB\_ID\_FILMU}

backdrops: <https://image.tmdb.org/t/p/original> + file\_path

Przykład pozyskanych danych:

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Pobrania danych o trailerach

Ostatnim koniecznym połączeniem dla modelu danych w aplikacji jest pobranie danych o trailerach związanych z filmem. Do realizacji tego wykorzystywany jest połączenie o poniższym szablonie:

localhost:8080/api/api/getVideosById/{TMDB\_ID\_FILMU}

Przykład pozyskanych danych:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

trailerLink: <https://www.youtube.com/watch?v=> + key pierwszego filmu, z listy który ma site równy YouTube i type równy Trailer

Na podstawie tych danych film jest dodawany do bazy i można go otworzyć w aplikacji react.

## Tworzenie filmów

Aby utworzyć nowy film w bazie w aplikacji Reactowej w górnym panelu można wprowadzić nazwę filmu i następnie po kliknięciu enter zostanie uruchomiona funkcja openMovie(screen poniżej), która najpierw sprawdza czy film o podanej nazwie znajduje się bazie, jeśli tak otwiera go, w przeciwnym razie wywołuje całą logikę pobierania danych z systemu TMBD za mocą aplikacji backendowej a na końcu również za pomocą aplikacji Javowej wprowadza nowy film do bazy danych, co zostało opisane niżej.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Dla wykonania tej operacji klasa entity wymagała dodania nowego konstruktora, który przyjmował niezbędne dane do stworzenia filmu w bazie Mongo.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

React przekazuje do aplikacji Javowej Mapę<String, Object>, gdzie niektórymi z tych obiektów są List<String>, więc konieczne było przed wywołaniem operacji tworzenia przekonwertowania ich odpowiednio na listy. Na potrzeby tego stworzyłem funkcję convertToListOfStrings, która jest ukazana poniżej.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Tak przetworzone dane mogły wykonać operację insert do bazy MongoDB z nowym obiektem filmu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

# Utworzenie konteneru backendowego i frontendowego

## Kontener aplikacji backendowej

Aby utworzyć kontener aplikacji javovej w pliku env konieczne jest wprowadzenie zmiany w zmiennej MONGO\_URI jeżeli używamy przekierowań portów(co realizowałem wcześniej). Oto konieczna zmiana:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Następnym krokiem jest stworzenie Dockerfile w głównym katalogu projektu:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Oraz wygenerowanie pliku Jar całego projektu.

Z racji na używanie IntelliJ IDEA możemy skonfigurować również nazwę obrazu za pomocą IDE i kliknąć Run

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Kolejnym krokiem jest utworzenie kontenera aplikacji na podstawie stworzonego obrazu za pomocą poniższej komendy:



Następnie należy stworzyć sieć oraz dodać do niej zarówno kontener bazy MongoDB, jak i aplikacji Javoweji.



Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Dla bezpieczeństwa należy zmienić nazwę kontenera na taką, która nie zawiera dużych liter i zastąpić je podkreśleniami (nie zrobienie tego może powodowa w przyszłości problemy komunikacyjne między kontenerami).

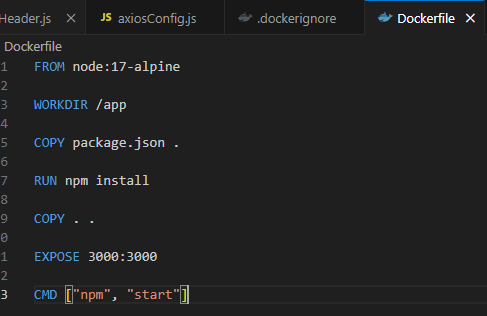


## Kontener aplikacji frontendowej

Podobnie jak w aplikacji Javowiej należy stworzyć w folderze głównym Dockerfile, jednak należy również utworzyć plik .dockerignore i dodać do niego ścieżkę /node\_modules.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie



Tak utworzony plik należy uruchomić w terminalu dla tworzenia nowego kontenera i dodać go do sieci wraz z otwarciem portu 3000.



Aplikacja do recenzji filmów jest gotowa do użytku. Wystarczy jedynie uruchomić wszystkie kontenery.