

AGH

cinemaex

DOKUMENTACJA PROJEKTU
BAZY DANYCH 1

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 **cinemaX**. Serwis ten powinien obsługiwać rezerwacje dla klientów i pozwolić administratorom na manipulację 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
- Dodawanie i usuwanie seansów filmów znajdujących się w bazie

Użytkownik:

- Przeglądanie filmów i seansów
- Przeglądanie aktorów
- Przeglądanie reżyserów
- Rezerwacja biletów
- Podgląd wykonanych rezerwacji

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 diagramów (konceptualny)

2.1. Budowa i analiza diagramu przepływu danych (DFD)

Różnicę między diagramami stanowi dostęp do dodawania nowych danych, który udostępniony jest dla konta o roli administratora.

DIAGRAM PRZEPŁYWU DANYCH DLA UŻYTKOWNIKA

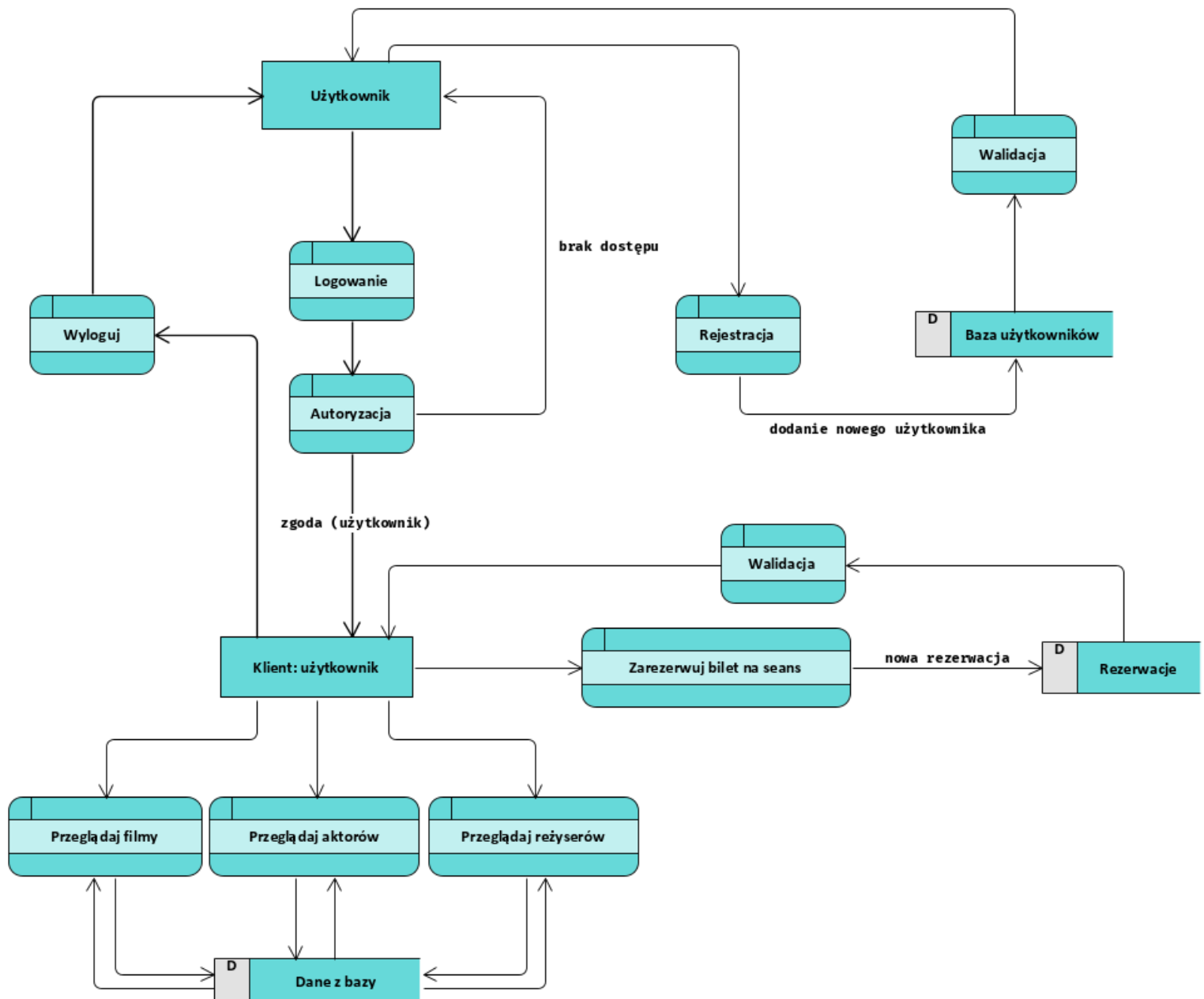
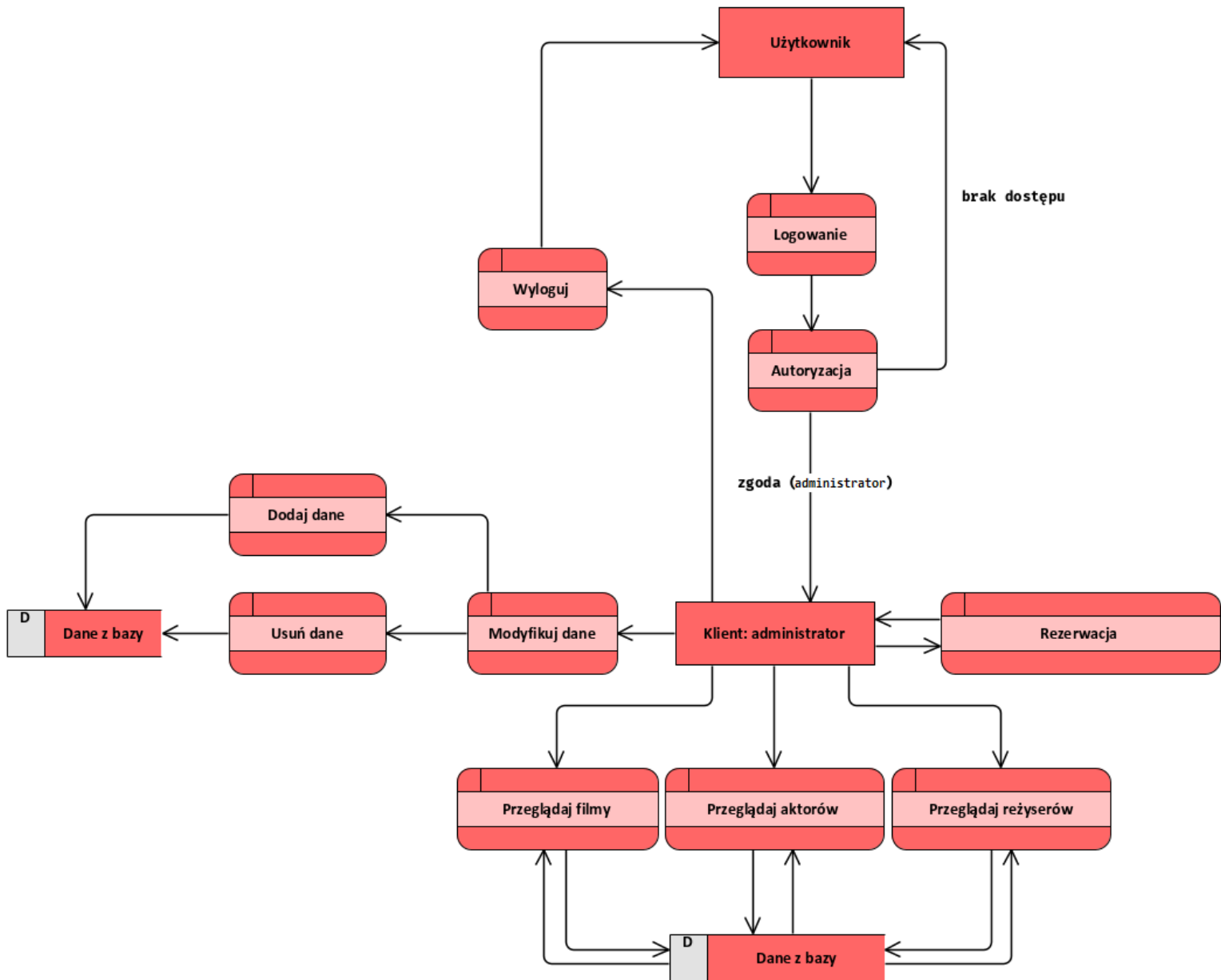


DIAGRAM PRZEPŁYWU DANYCH DLA ADMINISTRATORA



2.2. Zdefiniowanie encji (obiektów) oraz ich atrybutów

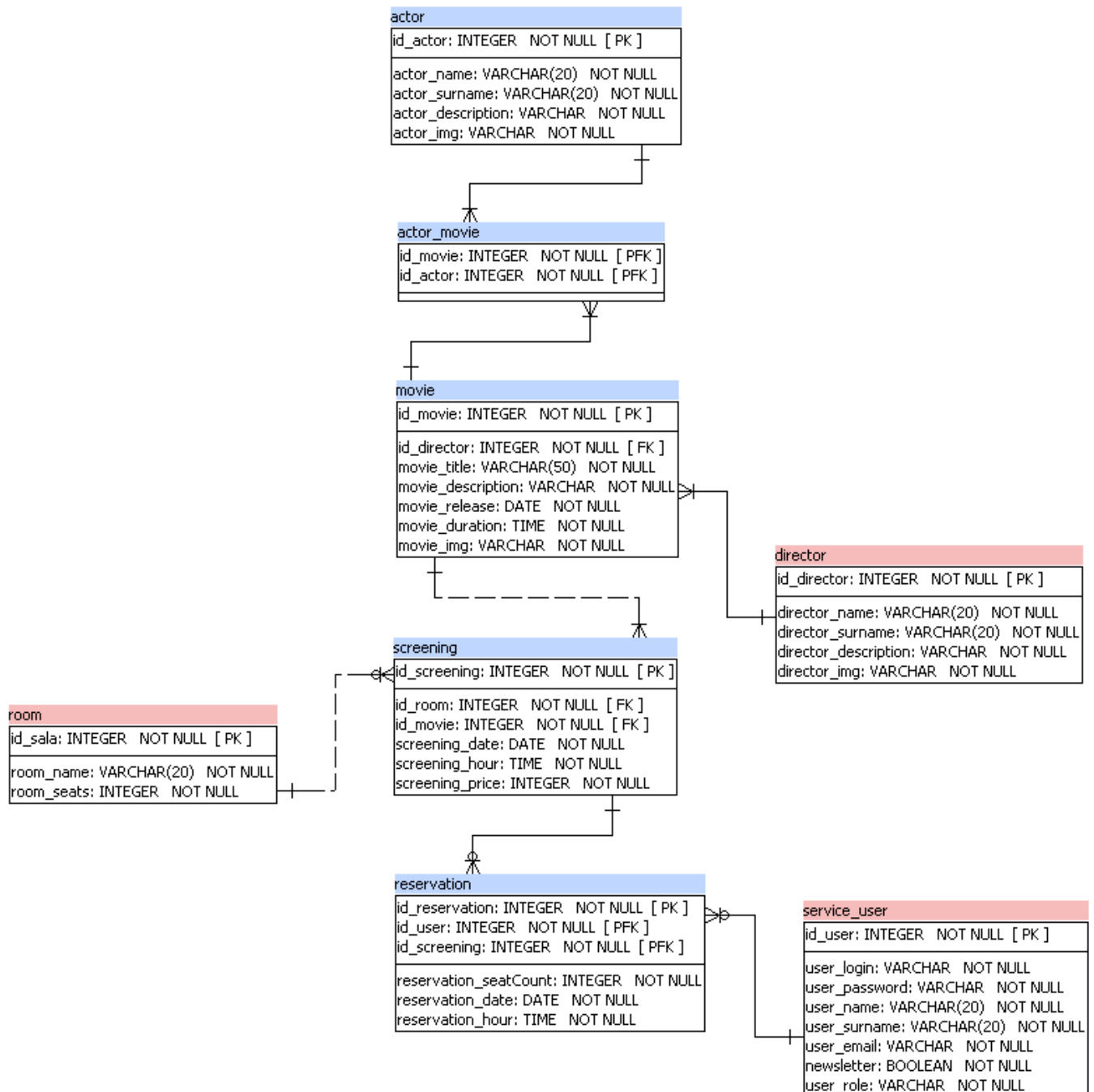
- Schemat **cinemaex**

- Typ wyliczeniowy enum **role_type** służący do zdecydowania czy konto ma uprawnienia **standardowe** lub **administratorskie**.
 - **'standard'** - nadaje uprawnienia standardowe, pozwala na przeglądanie danych
 - **'admin'** - nadaje uprawnienia administratorskie, pozwala na wprowadzanie danych do tabel
- Tabela **service_user** przechowuje dane na temat użytkowników:
 - **id_user SERIAL PRIMARY KEY** - unikalny identyfikator użytkownika,
 - **user_login VARCHAR** - login użytkownika,
 - **user_name VARCHAR** - imię użytkownika,
 - **user_surname VARCHAR** - nazwisko użytkownika,
 - **user_email VARCHAR** - email użytkownika, służy on między innymi do logowania, może być wykorzystany do przesyłania newslettera,
 - **newsletter BOOLEAN** - czy użytkownik chce otrzymywać newsletter serwisu,
 - **user_role role_type** - rola użytkownika
- Tabela **movie** przechowuje dane na temat filmów:
 - **id_movie SERIAL PRIMARY KEY** - unikalny identyfikator filmu,
 - **movie_title VARCHAR** - tytuł filmu,
 - **movie_description VARCHAR** - opis filmu,
 - **movie_release VARCHAR** - data premiery filmu,
 - **movie_duration VARCHAR** - czas trwania filmu,
 - **movie_img VARCHAR** - link do plakatu filmu
- Tabela **screening** przechowuje dane na temat seansów:
 - **id_screening SERIAL PRIMARY KEY** - unikalny identyfikator seansu,
 - **id_room INTEGER** - identyfikator sali, w której odbywa się seans,
 - **id_movie INTEGER** - identyfikator filmu odtwarzanego na seansie,
 - **screening_date VARCHAR** - data seansu,
 - **screening_hour VARCHAR** - godzina seansu,
 - **screening_price INTEGER** - cena za bilet

- Tabela **reservation** przechowuje dane na temat rezerwacji:
 - **id_reservation SERIAL PRIMARY KEY** - unikalne id rezerwacji,
 - **id_user INTEGER** - identyfikator użytkownika rezerwującego bilet,
 - **id_screening INTEGER** - id seansu na który rezerwujemy bilet,
 - **reservation_seatCount INTEGER** - ilość zarezerwowanych miejsc,
 - **reservation_date VARCHAR** - data utworzenia rezerwacji,
 - **reservation_hour VARCHAR** - godzina utworzenia rezerwacji
- 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
- Tabela **actor_movie** to tabela asocjacyjna łącząca w relacji **n do m** aktorów z filmami:
 - **id_movie INTEGER** - identyfikator filmu w którym występuje aktor,
 - **id_actor INTEGER** - identyfikator aktora występującego w filmie
- Tabela **room** przechowuje dane na temat sal występujących w kinie:
 - **id_sala SERIAL PRIMARY KEY** - unikalny identyfikator sali,
 - **room_name VARCHAR** - nazwa sali,
 - **room_seats INTEGER** - ilość miejsc w sali

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

2.3. Zaprojektowanie relacji pomiędzy encjami (ERD)



III. Projekt logiczny

3.1. Projektowanie tabel, kluczy, widoków

Projektowane tabele, klucze i indeksy znajdują się w folderze **database**, w pliku **db-create.sql**. Opis tabel zawarty został w punkcie **2.2**.

Każda tabela zawiera atrybut **id**, służący do jednoznacznego określenia konkretnej encji. Wyjątkiem jest tabela **aktor_film**, która jest tabelą asocjacyjną przechowującą wspólne dane: **id_film** oraz **id_aktor**. Jest tak, ponieważ jeden aktor może grać w wielu filmach, a w jednym filmie może grać wielu aktorów.

Widoki znajdujące się w bazie:

- **actor_full** - widok prezentujący pełne dane aktora, służące do przedstawienia go na jego pełnej podstronie
- **actor_preview** - widok prezentujący kilka podstawowych danych aktora, służących do przedstawienia go na ogólnej liście
- **director_full** - widok prezentujący pełne dane reżysera, służące do przedstawienia go na jego pełnej podstronie
- **director_preview** - widok prezentujący kilka podstawowych danych reżysera, służących do przedstawienia go na ogólnej liście
- **movie_full** - widok prezentujący pełne dane filmu, służące do przedstawienia go na jego pełnej podstronie
- **movie_preview** - widok prezentujący kilka podstawowych danych reżysera, służących do przedstawienia go na ogólnej liście
- **reservation_preview** - widok łączący kilka tabel, służący do przedstawienia rezerwacji w podstronie z dokonanymi rezerwacjami
- **screening_preview** - widok prezentujący kilka podstawowych danych o seansie, służących do przedstawienia go na ogólnej liście

3.2 Analiza zależności funkcyjnych i normalizacja tabel

Pierwsza postać normalna (1NF)

Wszystkie tabele powiązane z bazą została zdefiniowana według klucza podstawowego - **id**. Wszystkie dane są **atomowe**. Odpowiednie zestawy danych powiązane ze sobą znajdują się w tych samych tabelach.

Druga postać normalna (2NF)

Każda z tabel przechowuje dane dla konkretnej klasy obiektów.

Odpowiednie tabele zostały powiązane ze sobą **kluczem obcym**.

Trzecia postać normalna (3NF)

Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.

3.3 Zaprojektowanie operacji na danych

Wszystkie funkcje i operacje na danych przedstawione są w folderze **functions**, znajdującym się w folderze **database**.

Funkcje z folderu **getters**:

- Funkcja **get_sorted_screenings** - funkcja pobierająca z bazy posortowane względem daty i godziny (w tej kolejności) seanse filmowe
- Funkcja **get_new_screenings** - funkcja pobierająca z bazy dwa najbliższe datą seanse filmowe, po to by wyświetlić je na panelu głównym użytkownika
- Funkcja **get_movie_title** - funkcja pobierająca z bazy tytuły i identyfikatory filmów, po to by móc je wprowadzić w element **'select'**, w panelu administratora. Pozwala nam to później wybrać jaki film ma się pojawić w dodawanym seansie.
- Funkcja **get_director_name** - funkcja pobierająca z bazy imiona i nazwiska reżyserów, po to by móc je wprowadzić w element **'select'**, w panelu administratora. Pozwala nam to później wybrać jaki reżyser ma się pojawić w dodawanym filmie.
- Funkcja **get_full_movie(id_movie int)** - funkcja pobierająca z bazy pełne dane o filmie, którego identyfikator dostanie jako argument
- Funkcja **get_full_actor(id_actor int)** - funkcja pobierająca z bazy pełne dane o aktorze, którego identyfikator dostanie jako argument
- Funkcja **get_full_director(id_director int)** - funkcja pobierająca z bazy pełne dane o reżyserze, którego identyfikator dostanie jako argument

Funkcje z folderu **utils**:

- Funkcja **deleteMovie** - funkcja zwracająca wyzwalacz, który jest włączany w momencie usunięcia reżysera - wyzwalacz dba o to, by wraz z reżyserem usunięte zostały wszystkie filmy, które on wyreżyserował

Wyzwalacz **delete_director** - czuwający nad potencjalnym usunięciem reżysera

- Funkcja **deleteScreening** - funkcja zwracająca wyzwalacz, który jest włączany w momencie usunięcia filmu - wyzwalacz dba o to, by wraz z filmem usunięte zostały wszystkie seanse, na których miał być odtwarzany

Wyzwalacz **delete_movie** - czuwający nad potencjalnym usunięciem filmu

Funkcje z folderu **validation**:

- Funkcja **validate_new_actor** - funkcja walidująca wprowadzone dane dla nowego aktora

Wyzwalacz **validate_new_actor_trigger** - włącza funkcję po dodaniu nowego aktora

- Funkcja **validate_new_director** - funkcja walidująca wprowadzone dane dla nowego reżysera

Wyzwalacz **validate_new_director_trigger** - włącza funkcję po dodaniu nowego reżysera

- Funkcja **validate_new_movie** - funkcja walidująca wprowadzone dane dla nowego filmu

Wyzwalacz **validate_new_movie_trigger** - włącza funkcję po dodaniu nowego filmu

- Funkcja **validate_new_screening** - funkcja walidująca wprowadzone dane dla nowego seansu

Wyzwalacz **validate_new_screening_trigger** - włącza funkcję po dodaniu nowego seansu

Walidacja loginu i rejestracji odbywa się po stronie serwera. Wydaje mi się że rozsądniej by było jakby cała walidacja odbywała się po stronie serwera, natomiast by dodać do bazy funkcjonalności to właśnie tam przenieść resztę.

Więcej funkcjonalności i kwerend do obsługi bazy można znaleźć w folderze **server** w plikach serwerowych obsługujących poszczególne ścieżki w **Rest API**.

IV. Projekt funkcjonalny

4.1 Interfejsy do prezentacji, edycji i obsługi danych

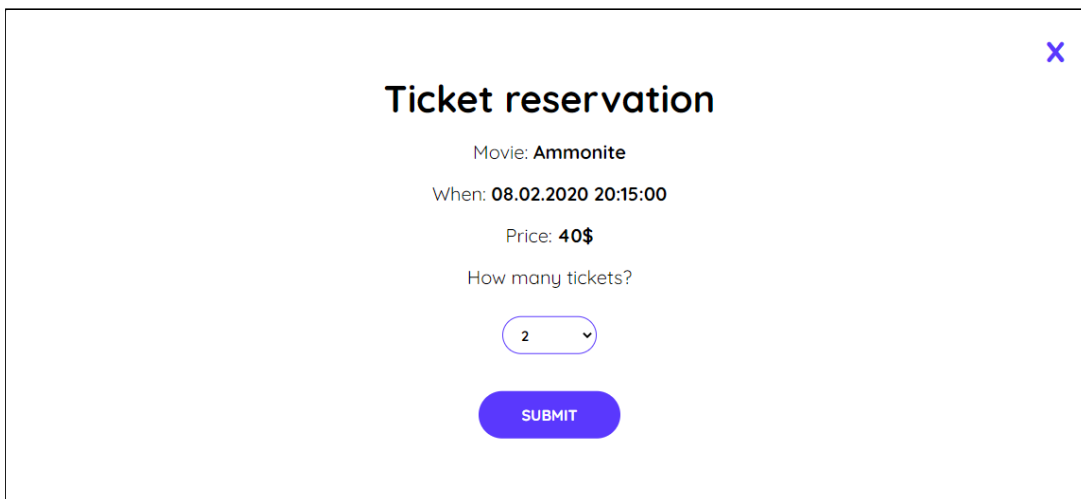
Edycja i obsługa danych jest możliwa jedynie z poziomu konta posiadającego uprawnienia **administratorskie**.

Użytkownik o **standardowych** uprawnieniach może natomiast rezerwować bilety na seanse oraz rezygnować z rezerwacji.

Uprawnienia standardowe i administratorskie:

- Rezerwacja biletu

Do rezerwacji biletów wyspecyfikowany został tzw. modal - czyli wyskakujące okienko, które pokazuje się po kliknięciu w przycisk **"Tickets"** pod wybranym seansem:



Ticket reservation

Movie: Ammonite

When: 08.02.2020 20:15:00

Price: 40\$

How many tickets?

2

SUBMIT

Widzimy tutaj podstawowe informacje na temat seansu, oraz mamy opcję wyboru jak wiele biletów chcemy zarezerwować (z limitem do 3).

- **Przeglądanie i usunięcie rezerwacji**

Dokonane rezerwacje możemy przeglądać w zakładce reservations. Tam też mamy opcję, by zrezygnować z rezerwacji, poprzez kliknięcie przycisku **“Cancel”**.

Your reservations

Ammonite

Date: 08.02.2020
Hour: 20:15:00
Reservation ID: 4

Reservation date: 7/1/2021
Reservation hour: 12:40:44
Reservation count: 3

CANCEL

Uprawnienia administratorskie:

- Dodawanie i usuwanie aktorów

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

Actors Panel

Directors Panel

Movies Panel

Screenings Panel

Add an actor

Name

Surname

Description

Image URL

Add an actor

Delete an actor

Jeffrey Dean Morgan

Delete an actor

- Dodawanie i usuwanie reżyserów

Administrator 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.

Actors Panel

Directors Panel

Movies Panel

Screenings Panel

Add a director

Name

Surname

Description

Image URL

Add director

Delete a director

Cary Joji Fukunaga

Delete a director

- Dodawanie i usuwanie filmów

Administrator może manipulować danymi dotyczącymi filmów. Może je dodawać i usuwać. Za pomocą wyzwalaczy dbamy o to by wraz z usuniętymi filmami, usuwane były seanse, na których miały być odtwarzane.

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

- Dodawanie i usuwanie seansów

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

Actors Panel

Directors Panel

Movies Panel

Screenings Panel

Add a screening

Room

No Time To Die

Date (DD:MM:YY)

Hour (HH:MM)

Price

Add a screening

Delete a screening

Ammonite 08.02.2020 20:15:00

Delete a screening

4.2 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.

- **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.

- **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.

- **Listę seansów**

W zakładce **Screenings** możemy oglądać wszystkie seanse znajdujące się w bazie oraz poprzez kliknięcie w plakat filmu z seansu możemy zobaczyć dodatkowe informacje na jego temat.

- **Lista naszych zamówień (rezerwacji)**

W zakładce **Reservations** możemy oglądać wszystkie zrobione przez nas rezerwacje, które znajdują się w bazie. Jak wyżej było wspomniane, możemy z nich również zrezygnować klikając przycisk **Cancel**.

4.3 Zdefiniowanie panelu sterowania aplikacji

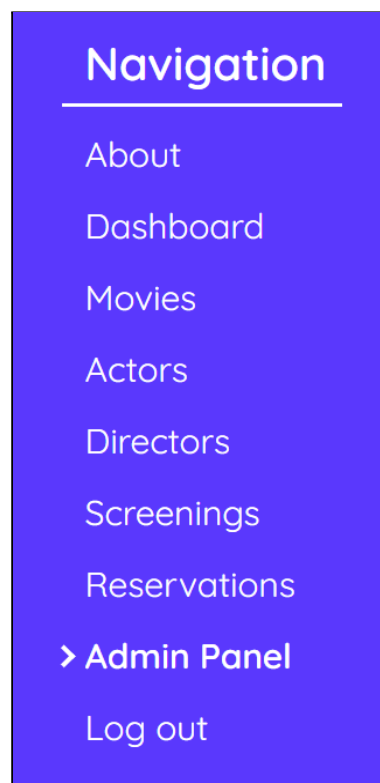
Głównym panelem sterowania jest nawigacja strony. By ją otworzyć, należy kliknąć przycisk w prawym górnym rogu strony. Jest on dostępny zawsze, natomiast w zależności od tego czy jesteśmy **zalogowani** czy też nie, mamy dostęp do innej ilości podstron. Panel jest w formie **wertykalnej**, w celu lepszej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

Gdy jesteśmy zalogowani w roli **standardowego** użytkownika to mamy dostęp do wszystkiego **oprócz panelu administratora**.

Gdy jesteśmy zalogowani w roli **administratora** to w panelu sterowania (**nawigacji**) mamy dostęp do wszystkich części strony.



V. Dokumentacja

5.1 Wprowadzanie danych

Wartości znajdujące się w bazie zostały dodane za pomocą insertów (znaleźć je można w pliku **insert.sql** w folderze **database**) oraz za pomocą formularzy, które były testowane dodawaniem i usuwaniem nowych rekordów.

Więcej danych wprowadzać możemy ręcznie poprzez formularze (tylko z konta z uprawnieniami **administratora**).

5.2 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.herokuapp.com/>
Proszę o chwilę cierpliwości ponieważ **heroku** potrzebuje chwili na pierwsze załadowanie. Włączenie strony po raz kolejny zajmie już dużo mniej.
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 (**tworzenie konta przez stronę nie zezwala na stworzenie konta o uprawnieniach administratora**) lub też zalogować się do istniejącego.

Przykładowe konta:

- Uprawnienia standardowe:
MAIL: **user@cinemaex.com**
HASŁO: **user**
- Uprawnienia administratorskie:
MAIL: **admin@cinemaex.com**
HASŁO: **admin**

4. Po zalogowaniu/zarejestrowaniu ukazuje nam się głównym Panel użytkownika, który prezentuje **dwa najbliższe seanse**. Z tego ekranu można przejść do wszystkich seansów za pomocą przycisku **SEE ALL**, lub przejść do innych ekranów za pomocą nawigacji (**prawy górny róg**).
5. Obsługa nawigacji: po kliknięciu przycisku, ukazuje nam się po animacji niebieski ekran, na którym możemy zobaczyć wszystkie ekrany:

- **About** - przekierowuje do strony internetowej autora aplikacji, nie jest faktyczną częścią projektu,
- **Dashboard** - ekran przedstawiający panel wejściowy użytkownika,
- **Movies** - ekran na którym ujrzyć 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 ujrzyć 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 ujrzyć 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.

- **Screenings** - ekran na którym ujrzyć możemy wszystkie seanse z bazy danych. Możemy się po nim poruszać za pomocą scroll'a. Pod każdym z seansów znajdują się dwa przyciski:

1. **TICKETS** - otwiera okienko, w którym możemy zobaczyć dokładne dane na temat seansu i możemy wybrać ilość rezerwowanych biletów. Możemy dokonać rezerwacji za pomocą przycisku **SUBMIT**, możemy również z niego wyjść klikając w **X** w prawym górnym rogu.

2. **INFO** - przenosi do strony filmu odtwarzanego na danym seansie. Pozwala dowiedzieć się więcej na jego temat.

- **Reservations** - ekran na którym ujrzyć możemy wszystkie dokonane przez nas rezerwacje. Klikając w przycisk **CANCEL** znajdujący się w opisie rezerwacji - rezygnujemy z niej, usuwając ją z bazy.
- **Admin Panel - (dostępny jedynie gdy konto posiada uprawnienia admina)** - ekran na którym ujrzyć 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, Screenings Panel).

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

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

5.3 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 **node-postgres** czyli klienta PostgreSQL dla Node.js. 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 wewnątrz React'owych (Javascript'owych) komponentów.,

Po stronie serwera:

- **node.js** - Javascript po stronie serwera
- **express.js** - framework Node.js dla łatwiejszego pisania aplikacji serwerowych
- **heroku** - PaaS - platforma w chmurze dla hostowania aplikacji
- **JWT** - Json Web Token - technologia pozwalająca na ograniczanie dostępu do strony i autoryzację
- **bcrypt** - szyfrowanie haseł

Po stronie bazy:

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

Obsługa żądań

Serwer zawiera dane potrzebne do logowania do bazy danych w pliku **server/dbdata.json** - z ich pomocą tworzy nowe połączenie z bazą w pliku **server/database.js**. Instancję jako moduł importujemy z pliku **server/index.js** gdzie obsługiwane są ścieżki. Tam też **tworzony** i **włączany** jest sam serwer. Możemy tam zobaczyć też inne ścieżki, które odprowadzają do innych plików w folderze **routes**. W tym folderze znajdują się pliki obsługujące ścieżki zasobów, które chcemy pobierać do klienta. Przykładowo w **server/index.js** mamy ścieżkę **'api/actors'**, i z niej odwołanie do pliku **routes/actors.js**. W tym pliku znajdziemy obsługę dalszej części ścieżki, przykładowo: **'api/actors/name'**, **'api/actors/:id'** itp. Obsługa polega na odebraniu żądania, przeprowadzeniu go przez jakieś oprogramowanie pośredniczące (**middleware**, w naszym przypadku jest to albo autoryzacja z JWT, albo walidacja) i wysłanie odpowiedzi w postaci żądanych danych lub też błędu, w razie jego wystąpienia.

Autoryzacja

Do autoryzacji wykorzystywany jest **JWT Token**. Jest on tworzony w momencie logowania lub rejestracji - funkcja która go tworzy znajduje się w pliku **server/utils/jwtGenerator.js**. Zawiera w środku zaszyfrowane (wybrany 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 kopiowany i przesyłany wraz zapytaniem do serwera. Wszystkie ścieżki po stronie serwera korzystają z **middleware**, które nazywa się **authorization**:

```
router.delete('/:id', authorization, async (req, res) => {
```

Jego działanie można zobaczyć w pliku **server/middleware/authorization.js**. Jego działanie pobiera z nagłówka zapytania **token** (skopiowany przez klienta z **localStorage**) i funkcjami przygotowanymi przez paczkę **JWT** sprawdza (**verify**) czy jest on prawidłowy. Jeżeli jest, wywoływana jest funkcja **next()**, przesyłająca zapytanie dalej do jego obsługi.

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, chcące 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 destrukuryzując 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 elementarty które pobraliśmy z bazy:

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

Error zawiera potencjalny błąd

- **używany jedynie gdy błąd wystąpi**

5.4 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 Express.js - <http://expressjs.com/en/api.html>
- Dokumentacje pomniejszych bibliotek
- Wikipedia - <https://en.wikipedia.org/wiki/> (informacje o aktorach, filmach itp.)