

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI  
I TECHNIK INFORMACYJNYCH



# Dokumentacja aplikacji mobilnej KINO

stworzonej na potrzeby projektu w ramach przedmiotu Bezpieczeństwo  
oprogramowania testy penetracyjne

Maciej Krysiak

300574

Łukasz Śleboda

300519

Opiekun projektu:  
dr inż. Mariusz Sepczuk

Warszawa 2021

## Spis treści

1. Cel projektu.....	Błąd! Nie zdefiniowano zakładki.
2. Instrukcja obsługi.....	5
2.1. Panel Pinu.....	5
2.2. Panel Logowania.....	6
2.3. Panel Rejestracji.....	7
2.4. Pasek Nawigacji.....	8
2.5. Zakładka Repertuar.....	9
2.6. Zakładka Bilety.....	10
2.7. Zakładka Promocje.....	11
2.8. Zakładka Konto.....	12
2.9. Podgląd Biletu.....	13
2.10. Podgląd Promocji.....	14
2.11. Podgląd Seansu.....	15
2.12. Podgląd Sali.....	16
3. API.....	17
3.1. Authenticate.....	17
3.1.1. POST /register.....	17
3.1.2. POST /update.....	17
3.1.3. POST /login.....	18
3.2.1. GET /File/Image/{filename}.....	20
3.2.2. GET /File/QR/{filename}.....	20
3.2.3. GET /File/Occasions/{filename}.....	20
3.3.1. GET /Occasions.....	21
3.4.1. GET /Screenings/Get.....	22
3.4.2. GET /Screenings/SeatsAvaliable.....	22
3.5.1. GET /Tickets/Types.....	23
3.5.2. GET /Tickets/MyTickets/{email}.....	24
3.5.3. GET /Tickets/Buy.....	25
3.5.4. GET /Tickets/filtered/{sql}.....	25
4. Instalacja aplikacji Kinowej.....	26
4.1. Baza danych.....	26
4.2. Serwer.....	26
4.3. Aplikacja Mobilna.....	26
5. Dziury w aplikacji.....	27
5.1. Login i hasło w komentarzu aplikacji.....	27
5.2. Możliwość zmiany parametrów kupowanego biletu.....	27
5.3. SQL INJECTION.....	27

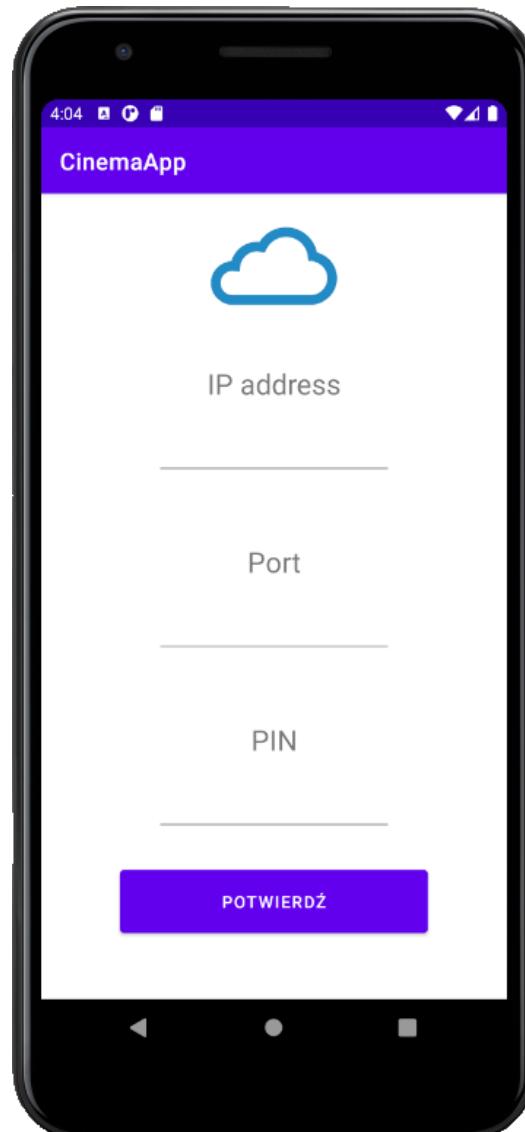
<b>5.4.</b>	<b><i>Authentication bypass.....</i></b>	<b><i>27</i></b>
<b>5.5.</b>	<b><i>Tymczasowy zapis pliku z biletem do pliku.....</i></b>	<b><i>27</i></b>
<b>5.6.</b>	<b><i>Możliwość podsłuchania intentu z danymi uwierzytelniania.....</i></b>	<b><i>27</i></b>
<b>5.7.</b>	<b><i>Zapis danych uwierzytelniania do Logu.....</i></b>	<b><i>27</i></b>
<b>5.8.</b>	<b><i>Zahardcodowany PIN dostępu do aplikacji.....</i></b>	<b><i>27</i></b>

## **1. Cel projektu.**

Celem projektu było zrealizowanie aplikacji mobilnej na system operacyjny Android w architekturze klient – serwer obsługującej kino. Program z założenia miał zawierać pewne luki bezpieczeństwa i służyć celom edukacyjnym na rzecz przedmiotu „BOT”.

## 2. Instrukcja obsługi

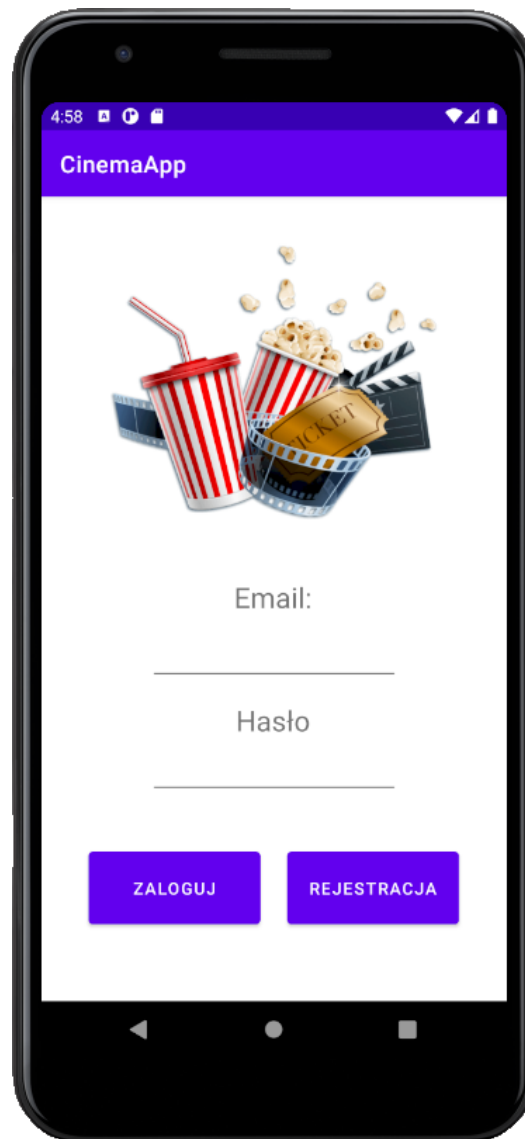
### 2.1. Panel Pinu



Rys 1

Pierwszy z widoków aplikacji wymaga od użytkownika wprowadzenia adresu IP, oraz portu na którym działa serwer aplikacji. Dodatkowo należy uwierzytelnić się wprowadzając kod PIN. PIN zapisany na stałe jest w kodzie aplikacji i ustawiony na wartość "1234". Aplikacja posiada zabezpieczenie polegające na konieczności wprowadzenia wszystkich danych, oraz ogranicza pola "port" i "pin" w taki sposób, że przyjmują one jedynie wartości numeryczne.

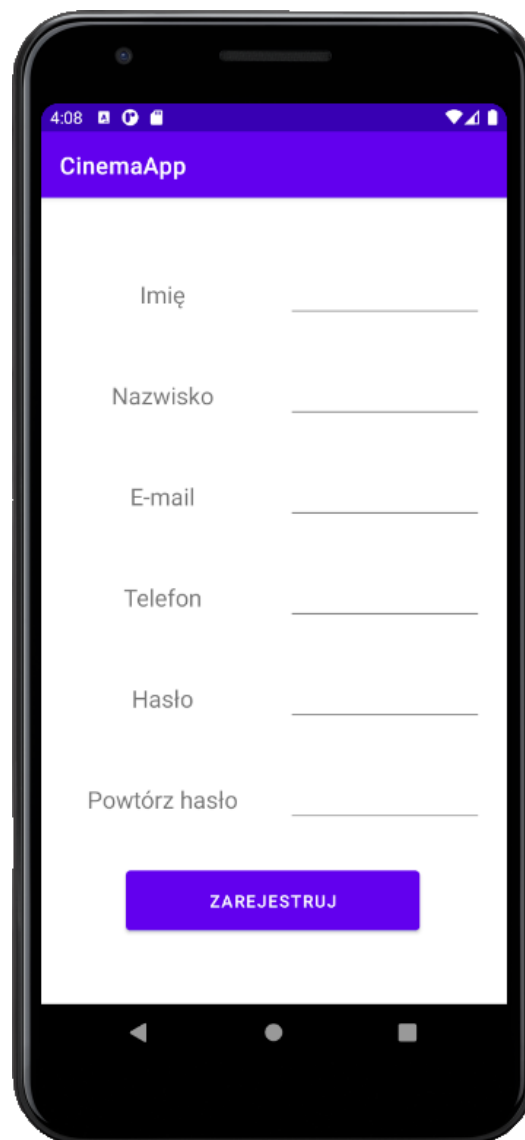
## 2.2. Panel Logowania



Rys 2

Na tym etapie dokonywana jest weryfikacja danych. Użytkownik wprowadza swój email i hasło, a następnie klikając przycisk Zaloguj wysyła żądanie do serwera korzystając z [API-1.3]. Po otrzymaniu pozytywnej odpowiedzi od serwera następuje wyświetlenie dymku z potwierdzeniem oraz zalogowanie użytkownika. Jeśli osoba korzystająca z aplikacji nie posiada jeszcze konta w serwisie może utworzyć je korzystając z panelu, do którego prowadzi przycisk "Rejestracja".

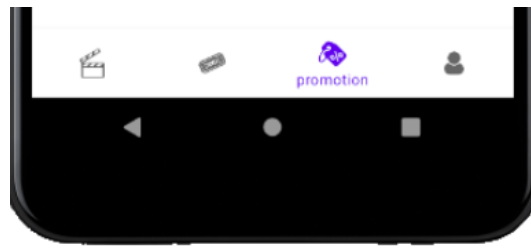
### 2.3. Panel Rejestracji

The image shows a smartphone screen with the CinemaApp registration interface. At the top, there is a purple header bar with the text "CinemaApp". Below the header, the screen is white and contains a registration form. The form consists of six input fields, each with a label to its left: "Imię", "Nazwisko", "E-mail", "Telefon", "Hasło", and "Powtórz hasło". Each label is followed by a horizontal line representing the input field. At the bottom of the form, there is a purple rectangular button with the white text "ZAREJESTRUJ". The smartphone's status bar at the very top shows the time "4:08" and various icons. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Rys 3

Panel rejestracji pozwala na utworzenie konta nowego użytkownika. W tym celu konieczne jest wypełnienie formularza rejestracji zwracając uwagę na poprawny format adresu email, oraz podanie dwa razy identycznego hasła i naciśnięcie przycisku "Zarejestruj" [API-1.1]. Zarówno w przypadku powodzenia, jak i niepowodzenia użytkownik otrzyma w dymku informacje zwrotną. Zostanie również automatycznie przeniesiony do ekranu logowania, a pola email i hasło wypełnią się automatycznie.

## 2.4. Pasek Nawigacji

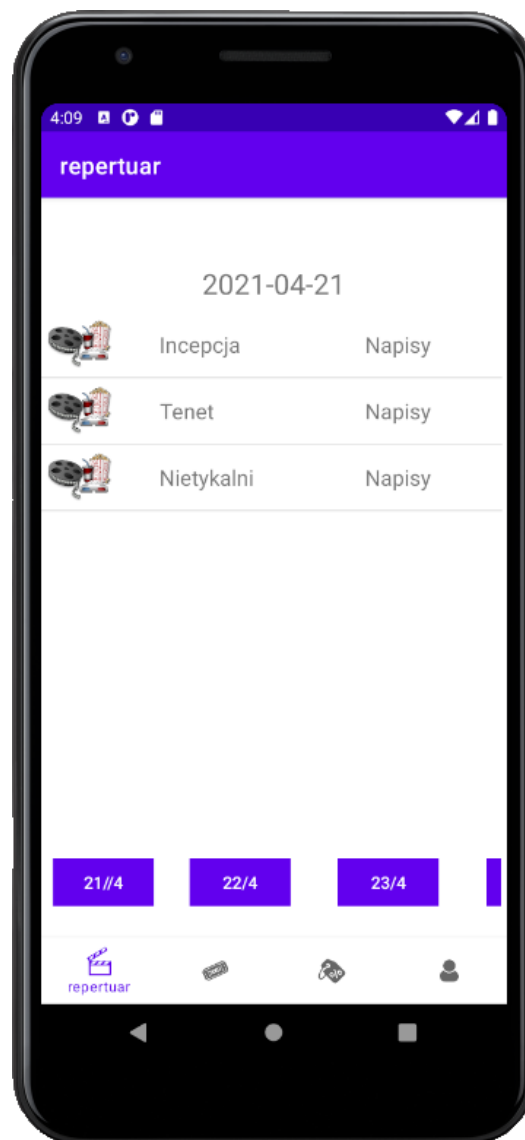


Rys 4

Po zalogowaniu na konto użytkownika mamy możliwość poruszania się po czterech dostępnych zakładkach (Repertuar, Bilety, Promocje, Konto). Można tego dokonać za pomocą paska nawigacji znajdującego się w dolnej części ekranu.



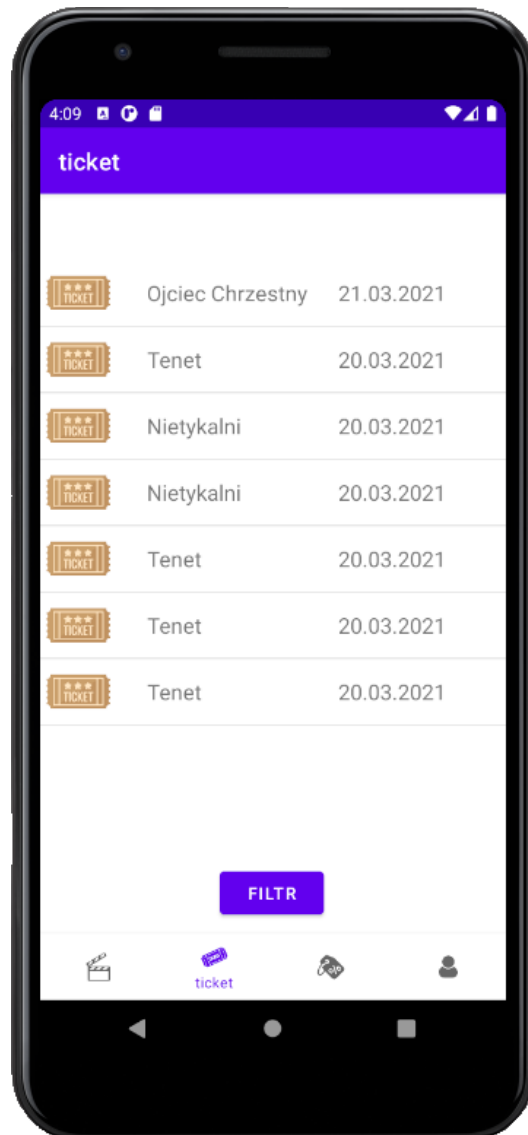
## 2.5. Zakładka Repertuar



Rys 5

Na szczycie podglądu repertuaru wyświetlana jest data informująca na jaki dzień tygodnia wyświetlane są filmy. Lista filmów zawiera podstawowe informacje, takie jak tytuł filmu, oraz typ tłumaczenia. Pobierana jest z serwera przy wykorzystaniu [API 4.1]. Dolny pasek z niebieskimi przyciskami pozwala na wybór konkretnego dnia tygodnia, na jaki chcielibyśmy przejrzeć repertuar. Zasięg dostępności filmów wynosi 7 dni w przód. Użytkownik ma możliwość wyboru jednego z filmów z listy poprzez kliknięcie na niego w celu podejrzenia szczegółów seansu.

## 2.6. Zakładka Bilety



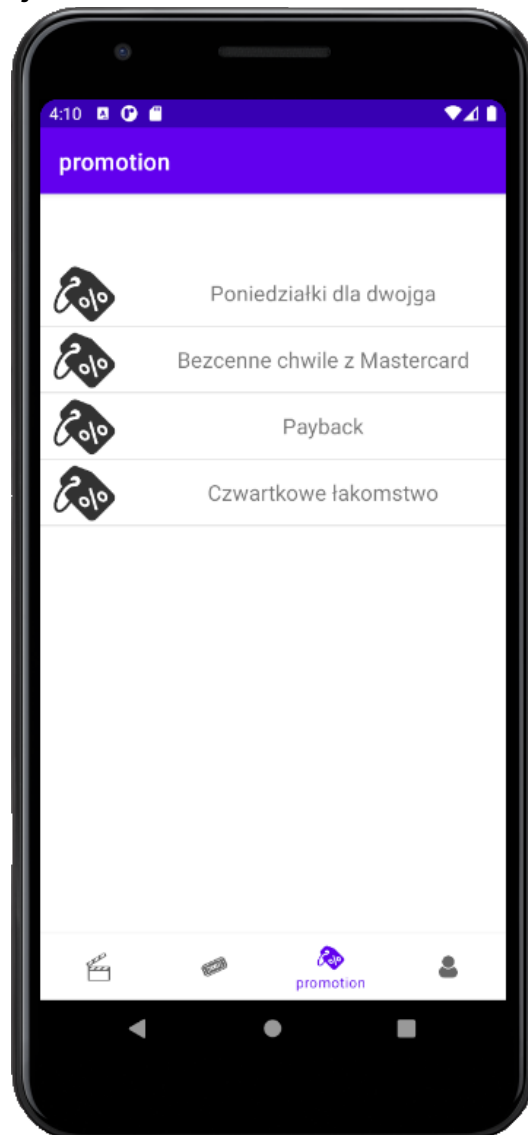
Rys 6

W tej zakładce użytkownik może znaleźć wszystkie bilety na seanse, jakie zakupił w naszej aplikacji. Pobierane są one przy wykorzystaniu [API-5.2]. Wszystkie z nich znajdują się na liście, którą można filtrować korzystając z ukrytego pod przyciskiem "FILTR" filtru. Podając w nim tytuł, datę, oraz godzinę możemy znaleźć konkretny interesujący nas bilet [API-5.4].



Rys 7

## 2.7. Zakładka Promocje



Rys 8

W zakładce promocje możemy znaleźć aktualne oferty, promocje i zniżki oferowane przez kino. Wyniki prezentowane na liście otrzymujemy korzystając z [API-3.1].

## 2.8. Zakładka Konto

account

Imię Lukasz

Nazwisko Sleboda

Telefon 789456123

E-mail lukasz@wp.pl

Hasło

Hasło

POTWIERDŹ ZMIANY

account

Rys 9

Zakładka konto pozwala na podejrzanie danych użytkownika uzyskiwanych dzięki [API-1.4], oraz ich modyfikacje [API-1.2]. Jedynym nieedytowalnym polem jest email na którym założone jest konto. Zmiana danych odbywa się poprzez edycję pól tekstowych, podanie dwa razy nowego, bądź też starego hasła i naciśnięciu przycisku “potwierdź zmiany”. O powodzeniu, bądź też niepowodzeniu poinformowani zostaniemy krótką wiadomością w dymku.

## 2.9. Podgląd Biletu



Rys 10

Do podglądu biletu dostajemy się poprzez wybranie konkretnej pozycji z listy w zakładce "bilety". Do każdego z nich dołączony jest unikatowy kod QR[API-2.2] pozwalający na weryfikację biletu przy wejściu na salę. Aby go otworzyć należy kliknąć napis "qr" na dole widoku biletu i poczekać na przekierowanie do strony.

## 2.10. Podgląd Promocji



Rys 11

Do szczegółowego widoku promocji dostajemy się poprzez wybór odpowiedniej pozycji na liście promocji. Możemy tutaj przeczytać jej opis, oraz obejrzeć grafikę promocyjną [API-2.3].

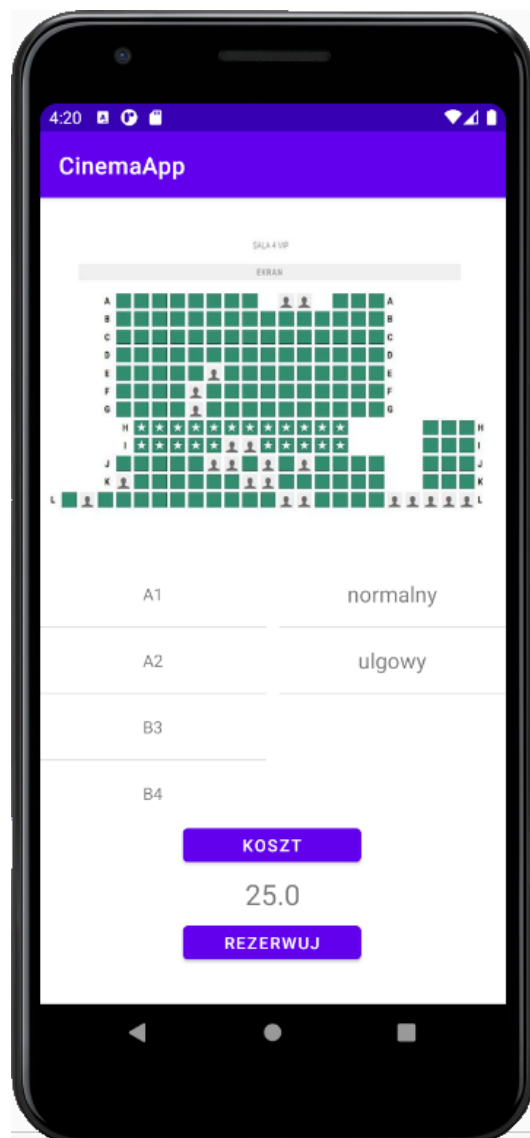
## 2.11. Podgląd Seansu



Rys 12

Do podglądu filmu dostajemy się poprzez wybór odpowiedniej pozycji na liście w panelu "Repertuar". Możemy tutaj między innymi przeczytać jego opis, obejrzeć okładkę[API-2.1], oraz wybrać interesującą nas godzinę w celu późniejszej rezerwacji na nią biletu. Wybranie jej z listy spowoduje przeniesienie do kolejnego ekranu umożliwiającego wybór konkretnego miejsca z sali.

## 2.12. Podgląd Sali

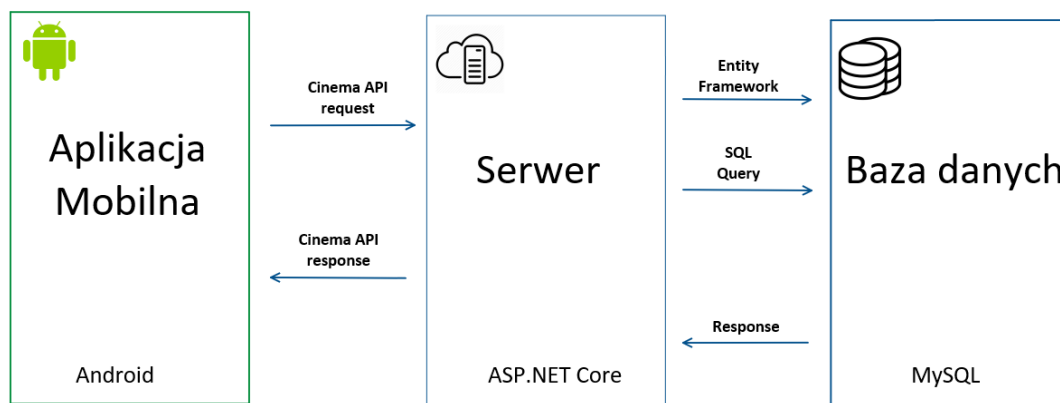


Rys 13

Widok sali jest ostatnim z ekranów, jeśli chodzi o uzupełnienie formularza rezerwacji. Mamy w nim możliwość wyboru konkretnego miejsca na sali, oraz typu biletu (normalny, ulgowy). Te pobrane z serwera zostają z wykorzystaniem w kolejności [API-4.2] oraz [API-5.1]. Koszt zamówienia możemy podejrzeć klikając w przycisk "koszt", a rezerwacji dokonujemy za pomocą przycisku "rezerwuj"[API-5.3]. O powodzeniu, bądź problemie zostaniemy poinformowani w dymku z informacją zwrotną.



### 3. API



Rys 14 – Architektura aplikacji

#### 3.1. Authenticate

Interfejs *Authenticate* służy do autoryzacji użytkownika i uzyskiwania wrażliwych danych dotyczących Klienta aplikacji.

##### 3.1.1. **POST** /register

API **register** służy do rejestracji użytkownika w bazie danych aplikacji. W celu dokonania poprawnej rejestracji, w ciele metody należy zamieścić następujące dane JSON:

```
{
  "name": "string",
  "lastName": "string",
  "mail": "user@example.com",
  "userName": "string",
  "password": "string",
  "phone": "string"
}
```

Wszystkie parametry są wymagane. Oznacza to, że wartość null jest niedozwolona. Ponadto, pole mail musi być mailem (czyli musi przyjmować format [example@domain.ex](mailto:example@domain.ex)), a pole phone musi przyjmować format numeru telefonu (same cyfry) o maksymalnej długości 12 znaków.

##### 3.1.2. **POST** /update

API **update** służy do aktualizacji danych użytkownika w bazie danych aplikacji. W celu zaktualizowania informacji o kliencie, w *body* metody należy zamieścić następujące dane JSON:

```
{
  "name": "string",
  "lastname": "string",
  "phone": "string",
  "password": "string",
  "email": "user@example.com"
}
```

```
}
```

**W nagłówkach** żądania należy zamieścić dane w formacie `"key": "value"`

`"email": "email@example.com"`

Wszystkie parametry są wymagane. Oznacza to, że wartość null jest niedozwolona. Pole email musi być aktualnym mailem użytkownika (nie można go zmienić, wprowadzenie nieprawidłowego adresu email nie zmieni maila w bazie). Pole phone musi przyjmować format numeru telefonu (same cyfry) o maksymalnej długości 12 znaków.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem,
- **Kod 404** - jeśli system napotkał błąd.

### 3.1.3. **POST** /login

API **login** służy do uwierzytelniania użytkownika w aplikacji. W celu autentykacji, w *body* metody należy zamieścić następujące dane JSON:

```
{  
  "mail": "user@example.com",  
  "password": "string"  
}
```

Wszystkie parametry są wymagane. Oznacza to, że wartość null jest niedozwolona. Ponadto, pole mail musi być mailem (czyli musi przyjmować format [example@domain.ex](mailto:example@domain.ex)).

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem, a dodatkowo w *body*:  

```
{  
  "token": "string",  
  "expiration": "string"  
}
```

Token zawarty w *body* metody będzie służył potem do autoryzacji użycia innych interfejsów API aplikacji,
- **Kod 404** - jeśli system napotkał błąd.

### 3.1.4. **GET** /info/{email}

API **info** służy do uzyskania danych użytkownika w aplikacji. W tym celu należy umieścić w adresie HTTP maila użytkownika, którego dane chcemy wyświetlić.

**W nagłówkach** żądania należy zamieścić dane w formacie `"key": "value"`

`"email": "email@example.com"`

`"Authorization": "Bearer <token>"`

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem, a dodatkowo w body:

```
{  
  "email": "email@example.com",  
  "name": "string",  
  "lastname": "string",  
  "phone": "string",  
  "password": ""  
}
```

Pole "password" jest puste. Serwer nie zwraca aktualnego hasła użytkownika.

- **Kod 404** - jeśli system napotkał błąd.

## 3.2. File

Interfejs *File* służy do uzyskiwania dostępu do plików znajdujących się na serwerze aplikacji.

### 3.2.1. **GET** /File/Image/{filename}

API **File/Image** służy do pobrania obrazka z serwera. W adresie żądania HTTP należy umieścić nazwę pliku.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się plik, a w hederach informacja o typie pliku.
- **Kod 404** - jeśli system napotkał błąd.

### 3.2.2. **GET** /File/QR/{filename}

API **File/QR** służy do pobrania kodu QR z serwera. W adresie żądania HTTP należy umieścić nazwę pliku.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się kod QR, a w hederach informacja o typie pliku.
- **Kod 404** - jeśli system napotkał błąd.

### 3.2.3 **GET** /File/Occasions/{filename}

API **File/Occasions** służy do pobrania obrazka z serwera. W adresie żądania HTTP należy umieścić nazwę pliku.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się kod QR, a w hederach informacja o typie pliku.
- **Kod 404** - jeśli system napotkał błąd.

### 3.3. Occasions

Interfejs *Occasions* służy do uzyskiwania dostępu do informacji związanych z okazjami.

#### 3.3.1. GET /Occasions

API **Occasions** służy do pobrania kodu QR z serwera. W adresie żądania HTTP należy umieścić nazwę pliku.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się JSON o następującej formie:

```
[ {  
  "title": "string",  
  "description": "string",  
  "image": "string",  
  "discount": "string",  
  "price": "string"  
}]
```
- **Kod 404** - jeśli system napotkał błąd.

### 3.4. Screenings

Interfejs *Screenings* służy do uzyskiwania dostępu do informacji związanych z repertuarem kina.

#### 3.4.1. **GET** /Screenings/Get

API **Screenings/Get** służy do pobrania repertuaru filmów granych w kinie danego dnia.

Przykład zastosowania API:

`https://<server_address>/Screenings/Get?date=<date>`

gdzie **date**, to data w formacie **dd/mm/yyyy**, z której żąda się repertuaru.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się JSON o następującej formie:

```
[ {
  "title": "string",
  "description": "string",
  "image": "string",
  "discount": "string",
  "price": "string"
}]
```
- **Kod 404/400** - jeśli system napotkał błąd.

#### 3.4.2. **GET** /Screenings/SeatsAvaliable

API **Screenings/SeatsA** służy do pobrania wolnych miejsc na dany seans filmowy.

Przykład zastosowania API:

`https://<server_address>/Screenings/SeatsAvaliable?MovieId=<int>&Time=<string>&Date=<string>`

gdzie:

- **Date**, to data seansu w formacie **dd/mm/yyyy**,
- **Time**, to godzina seansu w formacie **hh:mm**.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się JSON o następującej formie:

```
[ {
  "id": int,
  "seatRow": "string",
  "seatColumn": int
}]
```
- **Kod 404/400** - jeśli system napotkał błąd.

### 3.5. Tickets

Interfejs *Tickets* służy do uzyskiwania dostępu do informacji związanych z biletami użytkownika.

#### 3.5.1. **GET** /Tickets/Types

API **Tickets/Types** służy do pobrania typów biletów możliwych do kupienia w kinie.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się JSON o następującej formie:

```
[ {  
  "id": int,  
  "description": "string",  
  "discount": int  
}]
```
- **Kod 404/400** - jeśli system napotkał błąd.

### 3.5.2. **GET** /Tickets/MyTickets/{email}

API **Tickets/MyTickets** służy do pobrania biletów użytkownika. W adresie żądania HTTP należy umieścić mail użytkownika.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem. W body odpowiedzi znajduje się JSON o następującej formie:

```
[ {  
  "price": "string",  
  "screening": {  
    "movie": {  
      "id": int,  
      "title": "string",  
      "producer": "string",  
      "description": "string",  
      "subtitles": boolean,  
      "dubbing": boolean,  
      "imageName": "string"  
    },  
    "time": "string",  
    "date": "string"  
  },  
  "reservationType": {  
    "id": int,  
    "description": "string",  
    "discount": int  
  },  
  "seat": {  
    "id": int,  
    "seatRow": "string",  
    "seatColumn": int  
  },  
  "qrCode": "string",  
  "email": "string"  
}]
```

- **Kod 404/400** - jeśli system napotkał błąd.



### 3.5.3. **GET** /Tickets/Buy

API **Tickets/Buy** służy do kupowania biletów.

**W nagłówkach** żądania należy zamieścić dane w formacie *"key": "value"*

"email": "email@example.com"

"Authorization": "Bearer <token>"

Przykład zastosowania API:

https://<server\_address>/Tickets/Buy?movieId=<int>&date=<string>&time=<string>&seatId=<int>&reservationTypeId=<int>

gdzie:

- **movieId** - Id filmu, na który żąda się kupić biletu, int,
- **date** - data seansu w formacie **dd/mm/yyyy**,
- **time** - godzina seansu w formacie **hh:mm**,
- **reservationTypeId** - Id typu biletu, int

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem,
- **Kod 404/400** - jeśli system napotkał błąd.

### 3.5.4. **GET** /Tickets/filtered/{sql}

API **Tickets/Buy** służy do filtrowania biletów. W adresie żądania HTTP należy umieścić zapytanie sql to filtrowania.

API to zwraca:

- **Kod 200 OK** - jeśli akcja zakończyła się sukcesem, body odpowiedzi identyczne, jak w punkcie 5.2.
- **Kod 404/400** - jeśli system napotkał błąd.

## 4. Instalacja aplikacji Kinowej

### 4.1. Baza danych

W celu zainstalowania lokalnej bazy danych, należy:

- pobrać MySQL server w wersji 8.0.18. lub nowszej,
- stworzyć bazę danych, używając skryptu z pliku **DB/DatabaseScript.sql**.

### 4.2. Serwer

W celu uruchomienia serwera aplikacji lokalnie, na tym samym urządzeniu co aplikacja mobilna, należy przygotować maszynę z systemem Windows i postępować zgodnie z instrukcją:

- zainstalować najnowszą wersję środowiska Visual Studio,
- podczas instalacji Visual Studio zainstalować pakiet ASP.NET Core (aplikacje webowe),
- zainstalować rozszerzenie Visual Studio Conveyor, postępując zgodnie z instrukcją w linku: (<https://marketplace.visualstudio.com/items?itemName=vs-publisher-1448185.ConveyorbyKeyoti>),
- W linku z poprzedniego punktu w „Step 3” widać miejsce, w którym należy szukać adresu IP i portu serwera aplikacji,
- w pliku **CinemaServer/cofig/json/database.json** wprowadzić dane uwierzytelniania bazy danych (punkt 4.1.), z której będzie korzystał server.
- Uruchomić projekt.

Dodatkowe: informacja o problemie z integracją serwera ASP.NET z aplikacją na system operacyjny androida:

<https://medium.com/@ryan.samarajeewa/connect-your-android-app-to-a-local-asp-net-core-app-without-the-headaches-725dfb16e061>

### 4.3. Aplikacja Mobilna

W celu uruchomienia aplikacji klienckiej potrzebny będzie telefon, bądź też symulator z systemem operacyjnym Android w wersji 11 lub nowszej. W kolejnych krokach należy:

- Wgrać plik CinemaApp.apk,
- Uruchomić plik apk z poziomu menedżera plików telefonu bądź symulatora,
- Postępować zgodnie ze wskazówkami na ekranie,
- Znaleźć aplikację w menu i uruchomić klikając na ikonkę.

## 5. Dziury w aplikacji

### 5.1. Login i hasło w komentarzu aplikacji

Login i hasło do jednego z kont użytkownika zostały pozostawione jako komentarz w pliku "LoginActivity.java" w funkcji *onLoginClick()*.

### 5.2. Możliwość zmiany parametrów kupowanego biletu.

Polecenie zakupu biletów jest poleceniem typu GET, a przez to wszystkie parametry dotyczące zamówienia znajdują się w linku. Pozwala to na jego przechwycenie i modyfikację, a w efekcie zamówienie biletu z inną ceną.

### 5.3. SQL INJECTION

W API 5.4. wysyłane jest żądanie GET z zapytaniem SQL wpisanym bezpośrednio w link. Oznacza to, że można wpisać w niego dowolne zapytanie związane z biletami i tym sposobem np. otrzymać listę wszystkich biletów zakupionych w aplikacji przez różnych użytkowników wraz z ich danymi.

### 5.4. Authentication bypass

Pominięcie ekranu logowania pozwala osobie atakującej na pobranie bez uwierzytelniania i możliwość przejścia okazji oferowanych przez kino.

### 5.5. Tymczasowy zapis pliku z biletem do pliku

Jako, że bilety wyświetlane są przy użyciu komponentu WebView, tymczasowy plik HTML zakodowany w bajtach zapisywany jest do pliku *TICKETcache.txt*. Istnieje więc możliwość podejrzenia go, nawet po wylogowaniu i zamknięciu aplikacji.

### 5.6. Możliwość podsłuchania intentu z danymi uwierzytelniania

W naszej aplikacji istnieje możliwość podsłuchania intentu z wrażliwymi danymi użytkownika (email i hasło). Jest to możliwe w momencie zakładania nowego konta użytkownika. Dane te są umieszczane w intencie w pliku RegisterActivity, a następnie odbierane przez LoginActivity i automatycznie wpisywane w panel logowania.

### 5.7. Zapis danych uwierzytelniania do Logu

Dane logowania takie jak hasło i email zapisywane są do logu w momencie kliknięcia guzika "Zaloguj" na ekranie logowania.

### 5.8. Zahardcodowany PIN dostępu do aplikacji

Hash pinu do aplikacji został zapisany na stałe w aplikacji w pliku "hashDecoder.java" w linii dziewiątej. Ma to miejsce w pliku "LoginActivity.java" w funkcji *onLoginClick()*.

### 5.9. Niezabezpieczone kody QR biletów

Linki do biletów w postaci zapytania GET nie wymagają autoryzacji z wykorzystaniem tokenu sesji. Oznacza to, że taki kod możemy otrzymać wywołując odpowiedni link

w przeglądarce lub używając odpowiedniego hiperłącza w aplikacji do biletu innego użytkownika uzyskanego poprzez wykorzystanie podatności 5.3.

<b>Podatność</b>	<b>Stan Wykonania</b>
Login i hasło w komentarzu aplikacji	Wykonane
Możliwość zmiany parametrów kupowanego biletu.	Wykonane
Sztucznie uszkodzony filtr biletów bazujący na zapytaniach SQL	Wykonane
Authentication bypass	Wykonane
Tymczasowy zapis pliku z biletem do pliku	Wykonane
Możliwość podsłuchania intentu z danymi uwierzytelniania	Wykonane
Zapis danych uwierzytelniania do Logu	Wykonane
Zahardcodowany PIN dostępu do aplikacji	Wykonane
Niezabezpieczone kody QR biletów	Wykonane

*Tabela 15 – Status wykonania podatności*