

Střední průmyslová škola elektrotechnická  
a Vyšší odborná škola Pardubice

# **STŘEDNÍ PRŮMYSLOVÁ ŠKOLA ELEKTROTECHNICKÁ**

## **MATURITNÍ PRÁCE – WEBOVÉ STRÁNKY**

### **CINEMA TOWN**

březen 2024

Jan Rožek, 4.D



*„Prohlašuji, že jsem maturitní práci vypracoval(a) samostatně a použil(a) jsem literárních pramenů, informací a obrázků, které cituji a uvádím v seznamu použité literatury a zdrojů informací a v seznamu použitých obrázků a neporušil jsem autorská práva.*

*Souhlasím s umístěním kompletní maturitní práce nebo její části na školní internetové stránky a s použitím jejích ukázek pro výuku.“*

*V Pardubicích dne .....*

.....

*podpis*



## ZADÁNÍ MATURITNÍ PRÁCE

### Maturitní zkouška – profilová část – Maturitní projekt

<b>Obor:</b>	18-20-M/01 Informační technologie	<b>Školní rok:</b>	2023/2024
<b>Jméno a příjmení žáka:</b>	Jan Rožek	<b>Třída:</b>	4.D
<b>Téma maturitní práce:</b>	Systém na rezervaci sedadel v kině		
<b>Vedoucí maturitní práce:</b>	Mgr. Čestmír Bárta		
<b>Pracoviště vedoucího:</b>	SPŠE a VOŠ Pardubice		

**Kategorie maturitní práce:** Web

#### **Téma maturitní práce (popis):**

Vytvořte systém na rezervaci sedadel v kině. V aplikaci si bude moci uživatel prohlédnout nabídku filmů, které budou promítány v nejbližší době (do 14 dní). Registrovaný uživatel si bude moci rezervovat místo na promítání určitého filmu, zobrazit si rezervace a případně je zrušit (pokud to ještě půjde). Dále se bude moci uživatel přihlásit k odběru novinek (nově přidaných filmů).

#### **Hlavní body – specifikace maturitní práce:**

- 1) Rezervování sedadla na promítání
- 2) Poslání zakoupené vstupenky na email
- 3) Filtrování filmů
- 4) Dvoufázové ověření
- 5) Přihlášení se k odběru novinek -> uživatel bude informován, když se do systému přidá nový film
- 6) Zrušení rezervace

#### **Způsob zpracování maturitní práce:**

Maturitní práce musí být zpracována v souladu se zákonem č. 121/2000 Sb., autorský zákon.

Maturitní práce je tvořena praktickou částí (viz téma a specifikace maturitní práce výše) a písemnou prací.

Maturitní práce je realizována žákem převážně v rámci výuky ve 4. ročníku. Lze pokračovat na projektu z nižších ročníků.

Podrobné pokyny ke zpracování maturitní práce příslušné kategorie jsou uvedeny v dokumentu *MP\_Zpracovani-podrobne\_pokyny*.

Zpracování písemné práce musí odpovídat požadavkům uvedeným v dokumentu *MP\_Formalni\_stranka\_dokumentace*. Písemná práce musí být rovněž zpracována v souladu s normou pro úpravu písemností [ČSN 01 6910 (2014)], s citační normou [ČSN ISO 690], se

základními typografickými pravidly, pravidly sazby, gramatickými pravidly a pravidly českého pravopisu.

#### **Pokyny k rozsahu a obsahu maturitní práce:**

Rozsah a obsah praktické části maturitní práce je určen tématem a specifikací maturitní práce, rozsah písemné části maturitní práce je minimálně 15 normostran vlastního textu. Do uvedeného rozsahu se nezapočítávají úvodní listy (titulní list, prohlášení, zadání, anotace, obsah...), závěrečné listy (seznam literatury...) a přílohy. Podrobné pokyny k rozsahu a obsahu maturitní práce příslušné kategorie jsou uvedeny v dokumentu *MP\_Zpracovani-podrobne\_pokyny*.

#### **Kritéria hodnocení maturitní práce a její obhajoby:**

Maturitní práce a její obhajoba u maturitní zkoušky je hodnocena bodově. Celkový dosažený počet bodů je součtem bodů přidělených vedoucím práce a oponentem (maximální dosažitelný počet je 100 bodů). Výsledná známka u maturitní zkoušky je stanovena přepočtem celkového počtu bodů na známku pomocí tabulky uvedené níže.

V případě nesplnění tématu maturitní práce a v případě plagiátorství bude práce hodnocena stupněm „nedostatečný“.

#### **Tabulka bodového vyjádření hlavních kritérií a obhajoby:**

- praktická část – zpracování tématu maturitní práce .....[max. 25 bodů]
- písemná část – zpracování, formální a obsahová stránka .....[max. 15 bodů]
- obhajoba maturitní práce před zkušební komisí .....[max. 10 bodů]

Podrobná kritéria pro hodnocení maturitní práce příslušné kategorie jsou uvedena v dokumentu *MP\_Kriteria\_hodnoceni*.

#### **Tabulka přepočtu celkového počtu bodů na známku:**

[0 .. 60 bodů] .....	[5]
[61 .. 70 bodů] .....	[4]
[71 .. 80 bodů] .....	[3]
[81 .. 90 bodů] .....	[2]
[91 .. 100 bodů] .....	[1]

#### **Požadavek na počet vyhotovení maturitní práce a její odevzdání:**

Kompletní maturitní práce (praktická i písemná) se odevzdává ve stanoveném termínu vedoucímu maturitní práce. Písemná práce se odevzdává v jednom tištěném vyhotovení obsahující podepsaný přenosný nosič CD/DVD/SD s písemnou prací a sadou kompletních dat v elektronické podobě dle pokynů v dokumentu *MP\_Zpracovani-podrobne\_pokyny*.

**Termín odevzdání maturitní práce:** 5. dubna 2024

**Délka obhajoby maturitní práce před zkušební maturitní komisí:** 15 minut

Pardubice 23. října 2023

  
Mgr. Petr Mikuláš, ředitel školy

## *Anotace*

*Webová aplikace pro rezervování sedadel na promítání filmů. Bude umožňovat správu kin, filmů, promítání, žánrů filmů a možnost úpravy oprávnění u některých rolí. Přihlašování a registrace, bude řešena dvoufázovým ověřením za pomoci emailů.*

*Klíčová slova: rezervování sedadel, dvoufázové ověřování*

## *Annotation*

*Web application for movie screening seat reservation. It will allow management of theaters, movies, screenings, movie genres and the ability to edit permissions for some roles. Login and registration will be handled by two-phase verification using emails.*

*Keywords: seat reservation, two-phase verification.*

# Obsah

## ÚVOD

<b>1.1</b>	<b>CINESTAR</b>	<b>11</b>
1.1.1	Kladné stránky	11
1.1.2	Záporné stránky	12
<b>1.2</b>	<b>CINEMA CITY</b>	<b>12</b>
1.2.1	Kladné stránky	13
1.2.2	Záporné stránky	13
<b>1.3</b>	<b>KINO SVĚTOZOR</b>	<b>13</b>
1.3.1	Kladné stránky	14
1.3.2	Záporné stránky	14
<b>2</b>	<b>NÁVRH PROJEKTU</b>	<b>15</b>
<b>2.1</b>	<b>CÍLOVÉ SKUPINY</b>	<b>15</b>
<b>2.2</b>	<b>POPIS PROJEKTU</b>	<b>15</b>
<b>2.3</b>	<b>USE CASE DIAGRAM</b>	<b>15</b>
<b>2.4</b>	<b>DATABÁZE</b>	<b>16</b>
<b>3</b>	<b>POUŽITÉ TECHNOLOGIE</b>	<b>17</b>
<b>3.1</b>	<b>JAVASCRIPT</b>	<b>17</b>
3.1.1	TypeScript	17
3.1.2	React	17
3.1.3	Node.js	18
3.1.4	NPM	19
<b>3.2</b>	<b>JAVA</b>	<b>19</b>
3.2.1	Spring / Spring Boot	19
3.2.2	Maven	19
<b>3.3</b>	<b>MONGODB</b>	<b>19</b>
<b>4</b>	<b>BACKEND PROJEKTU</b>	<b>21</b>
<b>4.1</b>	<b>STRUKTURA KÓDU</b>	<b>21</b>
4.1.1	Controllery	22
4.1.2	Modely	23



4.1.3	<i>Servisy</i>	23
4.1.4	<i>Repositáře</i>	24
<b>4.2</b>	<b>PRINCIP KOMUNIKACE</b>	<b>25</b>
<b>4.3</b>	<b>KONCOVÉ BODY A HTTP METODY</b>	<b>25</b>
4.3.1	<i>Controllery se standartním fungováním</i>	26
4.3.2	<i>Controllery s deaktivovanými koncovými body</i>	26
4.3.3	<i>Controllery s poupraveným chováním</i>	27
4.3.4	<i>AuthController</i>	29
<b>5</b>	<b>FRONTEND PROJEKTU</b>	<b>30</b>
<b>5.1</b>	<b>STRUKTURA KÓDU</b>	<b>30</b>
5.1.1	<i>Volání API</i>	30
5.1.2	<i>Modely</i>	31
5.1.3	<i>Stránky</i>	31
<b>5.2</b>	<b>ROUTOVÁNÍ</b>	<b>32</b>
<b>6</b>	<b>POPIS JEDNOTLIVÝCH ČÁSTÍ APLIKACE</b>	<b>33</b>
<b>6.1</b>	<b>ÚVODNÍ STRÁNKA / PROGRAM</b>	<b>33</b>
<b>6.2</b>	<b>REGISTRAČNÍ A PŘIHLAŠOVACÍ FORMULÁŘE</b>	<b>34</b>
<b>6.3</b>	<b>DETAIL FILMU</b>	<b>36</b>
<b>6.4</b>	<b>MOJE REZERVACE</b>	<b>39</b>
<b>6.5</b>	<b>SPRÁVA</b>	<b>39</b>
6.5.1	<i>Přehled žánrů a spol.</i>	40
6.5.2	<i>Editování a přidávání žánrů a spol.</i>	41
	<b>ZÁVĚR</b>	<b>42</b>
	<b>SEZNAM PŘÍSTUPOVÝCH ÚDAJŮ</b>	<b>43</b>
	<b>SEZNAM POUŽITÉ LITERATURY A ZDROJŮ OBRÁZKŮ</b>	<b>44</b>
	<b>SEZNAM OBRÁZKŮ</b>	<b>46</b>
	<b>PŘÍLOHY</b>	<b>48</b>

## Úvod

Jako maturitní projekt jsem se rozhodl vytvořit zjednodušenou aplikaci pro rezervování sedadel na promítání filmů, která umožní kompletní správu kin včetně sálů, filmů a jejich promítání. Také bude umět posílat uživatelům, kteří jsou přihlášení, k odběru novinky (nově přidané filmy).

Tato aplikace by neměla sloužit sama o sobě k zisku peněz. Byla by doplňkem k existujícímu podnikání ve filmovém průmyslu. Poskytla by jednoduše dosažitelné informace pro zákazníky stejně jako umožnila jednoduchou správu zaměstnancům za to zodpovědným.

Ke konci studia na střední škole jsem chtěl zhotovit nějakou webovou aplikaci, za kterou by se nemusela (s přivřenýma očima a s přihlédnutím, že na tom budu pracovat sám) stydět žádná větší firma jak po vzhledové, tak po funkční stránce.

Pro tento konkrétní projekt jsem se rozhodl, protože mi jsou filmy celkem blízké, protože jsem si nedokázal vybrat zaměření internetového obchodu, který se stal pro maturitní práce standardem.

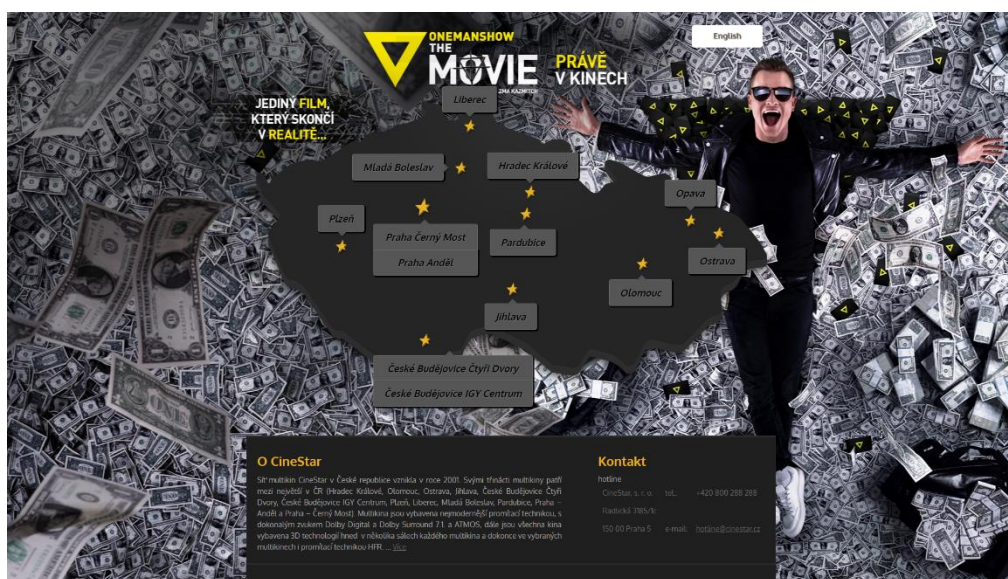
Aplikace bude rozdělena do tří serverů, Node.js s pomocí knihovny React pro frontend, Spring Boot pro backend a MongoDB jako databáze zajistí uchovávání dat. Pro toto řešení jsem se rozhodl, protože bych se chtěl zdokonalit ve výše uvedených technologiích a to, protože v nich vidím největší budoucnost.

## Analýza obdobných webových stránek

Analýzu je důležité provádět při každém vývoji nové aplikace zejména proto, abychom zjistili, jakou můžeme nabídnout konkurenční výhodu, čemu se vyhnout, anebo co je dobré z ostatních projektů převzít, případně vylepšit.

### 1.1 CineStar

Adresa: [cinestar.cz](http://cinestar.cz)



Obrázek 1 CineStar (zdroj: vlastní)

CineStar je síť multikin s největší hustotou pokrytí v republice. Účelem webové stránky je informovat širokou veřejnost o programu kin (případně i jejich poloze) a samozřejmě umožňuje rezervaci míst na určitá promítání filmů. Po prvním vstoupení na stránku se ukáže mapa republiky s možností vybrat kino jehož program uživatele zajímá. Stránka s detaily kin se pak liší jen nejspodnější částí, kde se nachází program kina s jeho adresou a kontaktem. Toto řešení mi nepřijde nejlepší, protože uživatele, byť jen poprvé, zdržuje od vybrání filmu. Dále bych ocenil možnost v detailu filmu mít možnost provést rezervaci.

#### 1.1.1 Kladné stránky

- Přehledný program
- V programu jsou náhledy na filmy

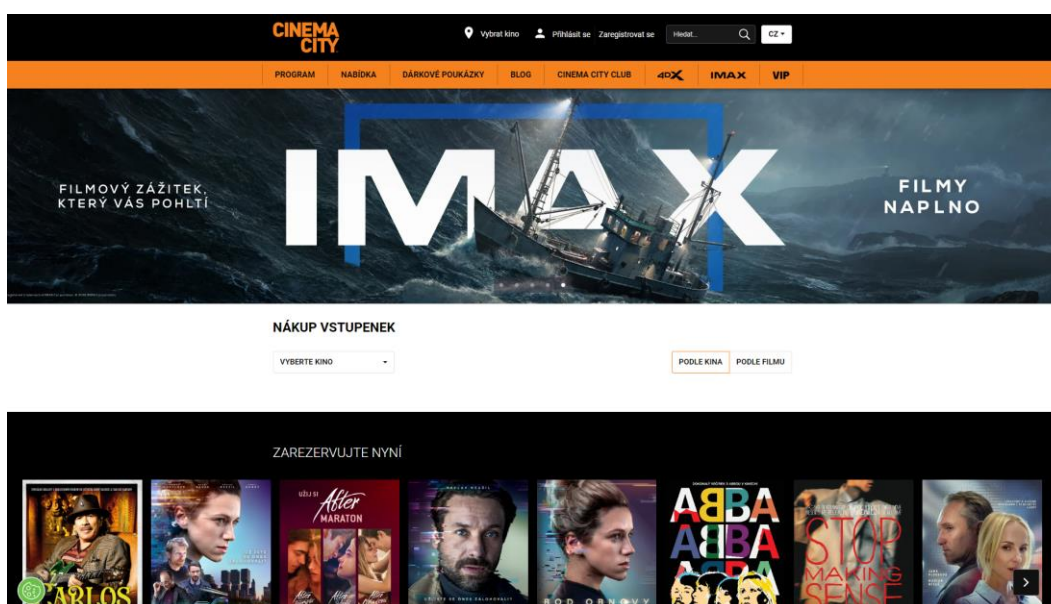
- Filmy je možné filtrovat pomocí kategorií

### 1.1.2 Záporné stránky

- Složitější cesta k programu
- V detailu filmu není možné provést rezervaci
- Je možné si prohlédnout filmy promítané jen ve vybraném kině

## 1.2 Cinema City

Adresa: [cinemacity.cz](http://cinemacity.cz)



Obrázek 2 Cinema City (zdroj: vlastní)

Cinema City je nadnárodní síť multikin s kořeny v Izraeli. Účel webové stránky se stejný jako u CineStar, a to informovat a rezervovat. Na úvodní stránce je možnost si prohlédnout nabídku filmů. Z počátku všechny filmy v různých kategoriích (promítáme, budeme promítat, speciální projekce a rodinné filmy), nebo při výběru kina už konkrétní program na určitý den (s výjimkou projekcí naplánovaných na dlouho dopředu). U každého filmu je jeho náhled (při kliku na něj se přehraje trailer na film) a nejpodstatnější informace o filmu. Posledním zobrazením filmů je „Podle filmu“ kde se po vybrání konkrétního filmu zobrazí všechna promítání tohoto filmu ve vybraný den.

### 1.2.1 Kladné stránky

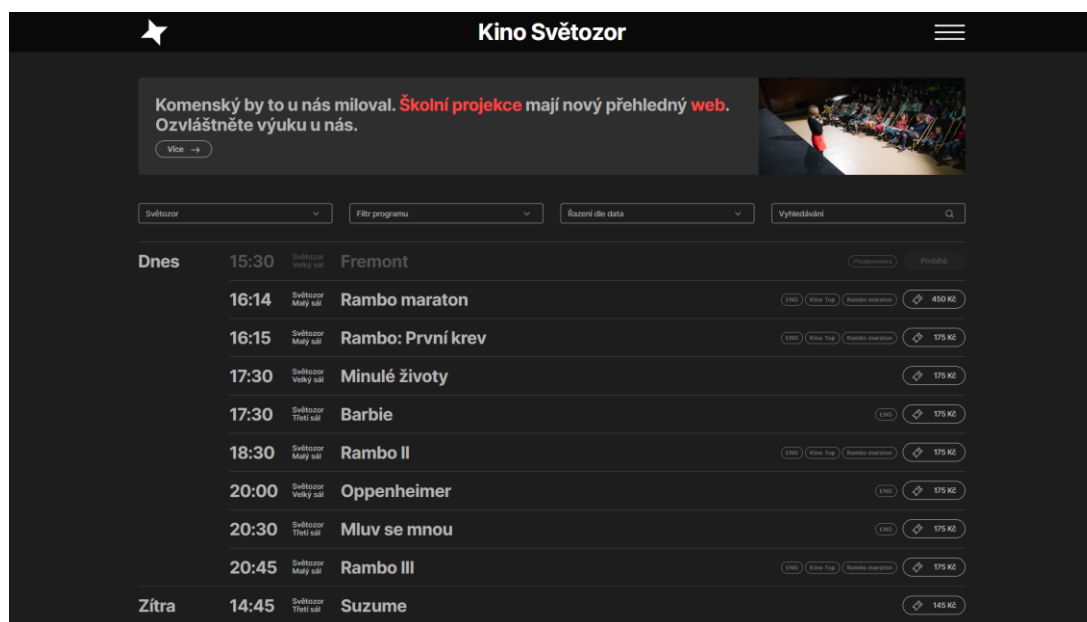
- Trailery na filmy
- Je možné rezervovat promítání z detailu filmy
- Možnost prohlédnout si celou nabídku filmů

### 1.2.2 Záporné stránky

- Po vybrání kina se není možné vrátit do stavu nevybraného kina
- Vyhledávání filmu podle žánru je možné jen při vybraném kině
- Program by mi více vyhovoval jako one page s odkazy v hlavičce na jednotlivé dny

## 1.3 Kino Světozor

Adresa: [kinosvetozor.cz](https://kinosvetozor.cz)



Obrázek 3 Kino Světozor (zdroj: vlastní)

Stránka spojuje program několika menších pražských kin. I proto si může stránka dovolit být poměrně jednoduchá a přehledná. Nejdůležitější částí stránky je filtrovací formulář a program, který je umístěný hned pod ním. Program je možné přepínat mezi časovým (ten je vidět na ukázce) a filmovým (skládá se z dlaždic s náhledem na film a jeho

jménem). Detail filmu se zobrazí jako popup s termíny projekce, ale rezervovat je možné jen nejbližší termín promítání.

### **1.3.1 Kladné stránky**

- V detailu filmu je možnost přehrát si trailer
- Přehledný program
- Jednoduchý a funkční design

### **1.3.2 Záporné stránky**

- Není možné vyhledávat podle žánrů
- V detailu filmu jsem byl chvíli zmatený při vybírání konkrétního promítání
- Na stránce není moc obsahu pro inspiraci

## 2 Návrh projektu

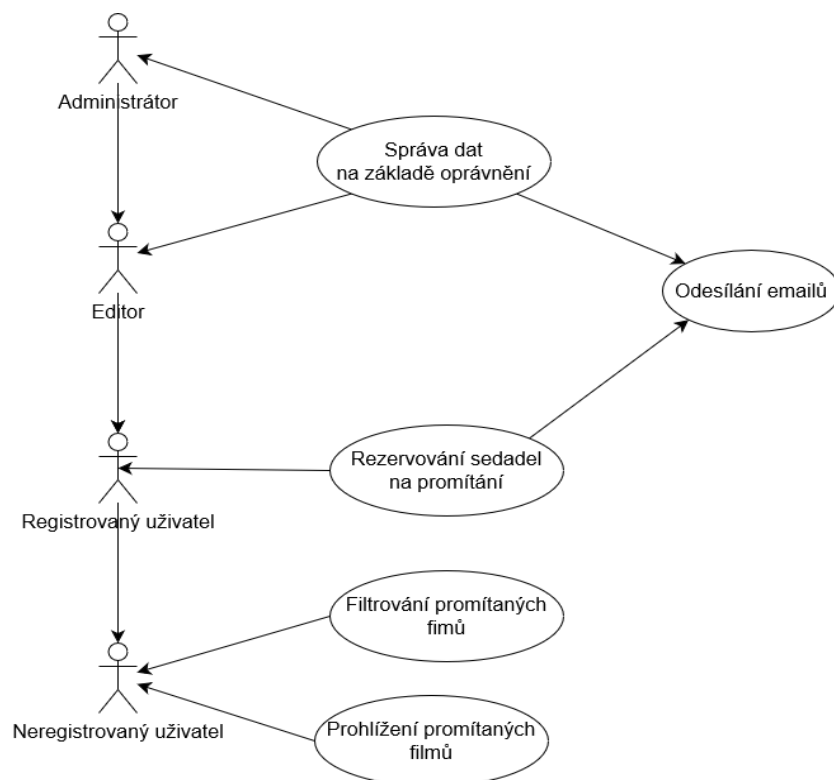
### 2.1 Cílové skupiny

Lidé, kteří rádi chodí do kina, aby se podívali na film.

### 2.2 Popis projektu

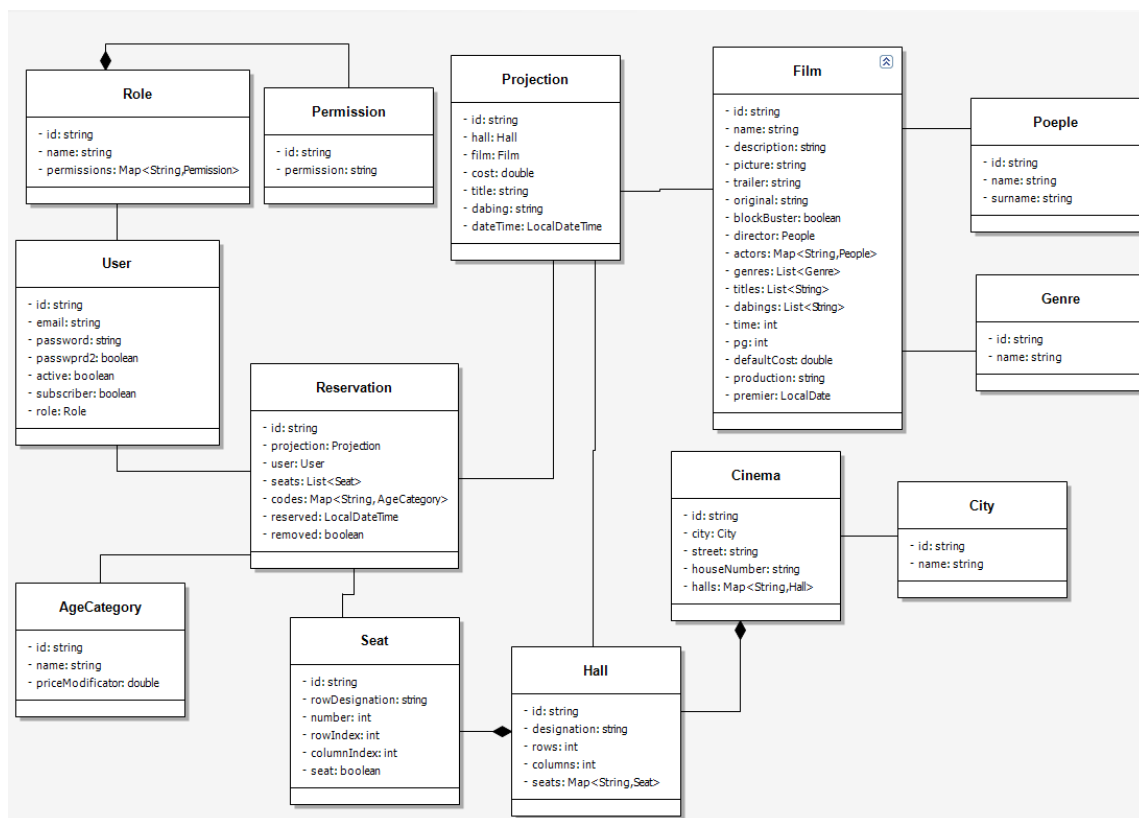
Projekt jako celek bude rozdělen do tří samostatných částí (serverů). O ukládání dat se bude starat databázový server MongoDB, přístup k němu bude zprostředkovávat backendová část napsaná v jazyku Java za použití frameworku Spring Boot skrze REST API. O prezentaci dat uživateli se bude starat JavaScriptový kód s využitím knihovny React, který bude spuštěn v prostředí Node.js. Toto řešení by do budoucna umožnilo snadnou rozšiřitelnost funkcionality webové aplikace i například vytvoření mobilní aplikace (bylo by třeba jen napsat aplikaci, protože API už by bylo hotové).

### 2.3 Use Case diagram



Obrázek 4 Use Case diagram

## 2.4 Databáze



Obrázek 5 Struktura databáze UML Class diagram

Na výše uvedeném diagramu je vidět struktura jednotlivých kolekcí v databázi (kolekce v NoSQL databázích představuje to co v relačních databázích tabulka), do kolekce se ukládají data ve formě dokumentů (dokument v NoSQL databázích představuje pak to co by v relačních databázích byl jeden záznam). Protože NoSQL nemá žádnou standardizovanou formu pro modelování grafů (jako ERD relačních databází), uvádím zde UML Class diagram, který výstižně zachycuje strukturu dat, kterou jsem použil.

Výše uvedený diagram reprezentuje strukturu databáze přesně až na jednu výjimku. Diagram např. říká, že v dokumentu Role jsou uloženy dokumenty Permission. Toto řešení by ale mělo hned několik problémů počínaje porušením pravidel normalizace dat, přes zbytečně vysoké nároky na kapacitu databáze až po velmi náročnou editaci. Proto je ukládání dokumentů do dokumentů řešeno nahrazením dokumentu záznamem DBRef s parametrem kolekce, ve které se nachází nahrazený dokument a parametrem id nahrazeného dokumentu.



## 3 Použité technologie

### 3.1 JavaScript

JavaScript je objektově orientovaný, interpretovaný jazyk (překládaný za běhu) s C-like syntaxí a dynamickým typováním, který využívá funkcionální paradigma (umožňuje uložení funkcí do proměnných). Jeho účel byl doplnit značkovací jazyky HTML a CSS na webových stránkách (manipulace s domem, složitější chování částí webu apod.) Dnes ale najde uplatnění i jako plnohodnotný programovací jazyk, v kterém může být napsaný i například program.

#### 3.1.1 TypeScript

Je nadstavba JavaScriptu, kterou vyvíjí Microsoft. Jedná se o samostatný programovací jazyk, který se kompiluje do JavaScriptu. Jeho přínos spočívá v zavedení statického typování do JavaScriptu, pohlídlá volání nedefinovaných vlastností, zavádí třídy, rozhraní a výčtové typy. Přichází i s genericitou a union typy.

#### 3.1.2 React

React je open-source JavaScriptovou knihovnou, pro tvorbu UI (user interface), za jehož vývojem stojí společnost Meta a komunita individuálních vývojářů. React funguje na bázi tzv. „komponent“. Ty mají své vstupní vlastnost a vnitřní stavy, což usnadňuje vytváření znovupoužitelného kódu.

React dokáže automaticky reagovat na změnu stavů a překreslovat aplikaci jen když je to potřeba. Komponenty mohou být vytvořeny buď jako funkce nebo jako třídy.

Pokud je komponenta vytvořena jako funkce, pak musí vrátet jeden JSX tag (JavaScript XML). Pokud je vytvořena jako třída, tak musí obsahovat metodu „render“, která vrací jeden JSX tag.

JSX má dost podobnou syntaxi jako HTML 5 s následujícími rozdíly:

- Je možné do něj vkládat reactové komponenty jako tagy

- Vlastnosti tagů mohou mít částečně jiný název (class je className, onclick je onClick )
- Mezi složené závorky {} je možné vkládat JavaScriptová data
- Každý tag musí být ukončen
- Všechny tagy jsou párové a jdou zapsat zkráceným zápisem
- Existuje prázdný tag, který se nevykreslí na stránku. Má využití, například když chceme, aby komponenta vracela více jak jeden tag.

JSX může vypadat následovně:

```

    <>
      /** Komentář */
      /** Vložení funkce k eventu onClick */
      <div className='management-reference' onClick={() =>
navigate(url)}>
        /** ukázka zkráceného zápisu */
        <div className='management-reference-img'
style={imgBg} />
        /** vložení komponenty filter */
        <Filter />
        /** důkaz o párovosti všech tagů */
        <br></br>
        /** vložení textu do odstavce */
        <p>{text}</p>
      </div>
    </>

```

*Příklad kódu Reactu*

Webové rozhraní vytvořené pomocí reactu je pak SPA (Single Page Application – celá aplikace je obsažena v jednom HTML a jedno JS souboru).

### 3.1.3 Node.js

Je prostředí, které umožňuje spouštění JavaScriptového kódu mimo webový prohlížeč. Jeho primárním účelem je tvorba serverové části webových aplikací s důrazem na vysokou škálovatelnost (schopností obsloužit mnoho připojených klientů současně). Té dosahuje tím, že jednotlivé požadavky klientů zachytává jedno vlákno, která je pak distribuuje mezi asynchronní vlákna. Asynchronní vlákna pak požadavky zpracovávají.

### **3.1.4 NPM**

Node.js package Manager je správce JavaScriptových balíčků, který usnadňuje jejich instalaci a údržbu.

## **3.2 Java**

Je objektově orientovaný programovací jazyk se striktním typováním a C-like syntaxí. Programy napsané v Javě jsou platformě nezávislé (i na operačním systému, tak na mikroprocesoru). Tuto vlastnost má díky JVM (Java Virtual Machine). Implementuje automatickou správu paměti (garbage collector).

### **3.2.1 Spring | Spring Boot**

Jedná se o Framework pro jazyk Java, jehož cílem bylo ulehčení vývoje webových aplikací v Javě.

Čistý Spring nepoužívám, protože by to vyžadovalo nastavení tzv. servlet kontejneru (webového serveru). Nastavení servlet kontejneru se obecně považuje za náročnější akci do, které jsem se nechtěl pouštět a místo toho jsem použil knihovnu Spring Boot, která má nastavený kontejner už implementovaný (tomcat). Dalším důvodem pro využití Spring Bootu pro mě bylo to, že Spring pouze definuje rozhraní, zatímco ve Spring Bootu jsou už implementována.

### **3.2.2 Maven**

Je správce balíčků, který je možné použít pro Java projekty. Zařizuje přidání Spring Bootu a jiných knihoven do projektu.

## **3.3 MongoDB**

Jako databázi jsem zvolil dokumentově orientovanou NoSQL databázi MongoDB. MongoDB neukládá data v tabulkách a už vůbec ne v řádcích. Data jsou ukládána ve formátu BSON (Binary JSON) a jsou organizována ve formě dokumentů (v relačních databázích bychom to nazvali řádek), které jsou seskupeny do kolekcí (v relačních databázích bychom

řekli tabulek). Každý dokument ale může mít jinou vnitřní strukturu (aplikace si musí sama hlídat jaká data do databáze ukládá).

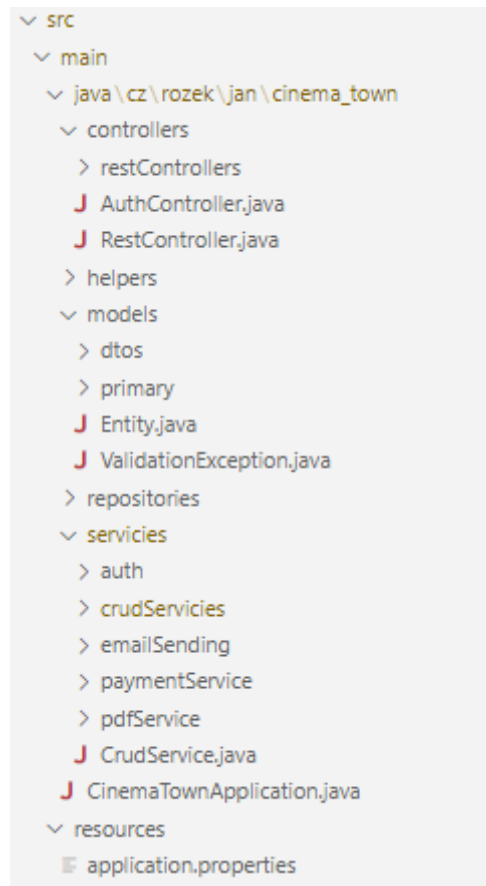
Zatímco vztahy v relačních databázích jsou řešeny pomocí cizích klíčů MongoDB zná dvě vazby mezi objekty. Objekt může obsahovat jiný objekt anebo může obsahovat referenci (odkaz). Druhý způsob je podobný cizím klíčům relačních databází, ale přichází s problémem integrity dat, kterou musí opět vyřešit aplikace.

## 4 Backend projektu

Backendová část je napsaná jako REST API (Application Programming Interface) v programovacím jazyce Java s použitím frameworku Spring Boot, který využívá pro uchování dat NoSQL databázi, MongoDB.

### 4.1 Struktura kódu

Aplikace odpovídá třívrstvé architektuře. Prezentační vrstvu představují třídy Controller a DTO (zástupné třídy, které nesou data. Slouží pro jednodušší komunikaci s frontendem). Aplikační vrstva je tvořena Modely třídami Service. Datovou vrstvu představují rozhraní Repository.



Obrázek 6 Struktura backendové části

### 4.1.1 Controllery

Controller registruje metody pro obsluhu určitého dotazu k určité cestě na webovém serveru (URI). Až na třídu AuthController, všechny dědí z abstraktní třídy RestController. RestController definuje základní zpracování HTTP požadavků GET, POST, PUT a DELETE. Spolu s nimi deklaruje ještě dvě konstanty modifikátorem dostupnosti protected pro získávání JWT z hlaviček HTTP požadavků. Poslední vlastnosti ResrControlleru, které v něm jsou definovány jsou dvě proměnné rovněž s dostupností protected. První z nich je generická proměnná service, která musí rozšiřovat abstraktní třídu CrudService. Závislost na toto komplexní entitu je přidána pomocí návrhového vzoru Dependenci Injection, za pomoci anotace @Autowired. Stejným způsobem je vytvořena i závislost na entitu třídy ObjectMapper. Instance této třídy převádí data, která metody vracejí na textové řetězce formátu JSON. Data, která controller zpracovává jsou určena genericky a musí implementovat rozhraní Entity.

Toto řešení pomocí abstraktní třídy RestController řeší ohromnou redundanci kódu a také dovoluje nechat většinu ostatních Controllerů téměř prázdných, aniž by zabránila snadné úpravě chování pro určitý typ dat. V konkrétním controlleru je třeba definovat kromě typu dat i konkrétní Service a URI cestu ke koncovým bodům. V konkrétním controlleru může být definována i obsluha pro jiný koncový bod, než jaké definuje RestController.

```
@RestController
@CrossOrigin(origins = {"https://www.mp.home-lab.rozekja.fun"})
@RequestMapping(path = "/api/age_categories")
public class AgeCategoryController extends
    cz.rozek.jan.cinema_town.controllers.RestController<AgeCategory,
    AgeCategoryService> {

}
```

*Celá třída AgeController*

AuthController je poté zodpovědný za obsluhu uživatelských požadavků volaných za účelem autentizace.

```
@org.springframework.web.bind.annotation.RestController // Díky
anotaci bude řída rozpoznána jako RestController
@CrossOrigin(origins = {"https://www.mp.home-lab.rozekja.fun"}) //
povolení zdroje dotazů na API (URL ze které přijde dotaz)
@RequestMapping(path = "/auth") // určení URI cesty k endpointům
public class AuthController {
```

*Anotace definující controller*

### 4.1.2 Modely

Třídy modelů definují strukturu dat a základní operace na nich. Například metodu, která zkontroluje, zda je instance validní.

#### 4.1.2.1 Primární Modely

Modely, které bych nazval jako primární reprezentují data, tak jak jsou uložena v databázi. Všechny tyto modely implementují rozhraní Entity. Toto rozhraní deklaruje metody get() a set() pro Id typu String a metodu validate() s návratovým typem void a možností vyvolat výjimku ValidationException (k vyvolání výjimky dojde pokud je metoda zavolána na nevalidní instanci).

Každý primární model těží ze tří anotací:

```
@Document("films") // určí, že jde o třídu, jejíž instance je možné  
uložit do databáze mongoDB a to to kolekce 'films'  
@Data // anotace poskytne metody get a set pro všechny vlastnosti  
třídy  
@NoArgsConstructor // zaručí třídě bezparametrický konstruktor  
public class Film implements Entity {...}
```

*Ukázka anotací primárního modelu*

#### 4.1.2.2 DTO (Data Transfer Object)

Objekt DTO slouží k zapouzdření dat pro jejich posílání mezi částmi aplikace (takže i většina primárních modelů funguje i jako DTO). Jednou takovou třídou je například třída TokenDeviceId, která slouží k obalení přihlašovacího JWT a JWT, který slouží pro udělení důvěry zařízení, ze kterého se uživatel přihlašuje.

### 4.1.3 Servisy

Tyto třídy implementují logiku práce s daty. Podobně jako u Controllerů je zde abstraktní třída, konkrétně CrudService, která obsahuje výchozí implementaci metod CRUD.

#### 4.1.3.1 CrudService

Tato třída deklaruje vlastnosti repository a authService s dostupností protected spolu s jejich injectody. Injectory jsou abstraktní metody, které musí každá třída, která bude tuto abstraktní rozšiřovat implementovat.

Kromě základních metod `readAll()`, `readById()`, `create()`, `update()` a `delete()` tato třída deklaruje a implementuje i metodu `verifyAccess`, tato metoda ověří, zda je dotazovaný oprávněný k provedení dané akce. Volání této metody je prováděno v každé z metod CRUD a mělo by k tomu docházet i při přepisování těchto metod v konkrétních třídách service. Aby bylo možné pracovat už na abstraktní vrstvě s konkrétními oprávněními bylo nutné deklarovat abstraktní metody pro získání každého typu oprávnění (jedno oprávnění pro jednu metodu CRUD).

#### **4.1.3.2 AuthService**

Třída implementuje logiku pro správu autentizace a autorizace. Při svém zavedení si vytvoří nebo načte páry klíčů pro asymetrické podepisování JWT, které vydává a ověřuje.

#### **4.1.3.3 PdfService**

Poskytuje metodu, pro vygenerování lístků (potvrzení platby) ve formátu `.pdf`.

#### **4.1.3.4 PaymentService**

Konstruktor slouží jako injector, která přidá instanci závislost na mapu, která pod typy plateb obsahuje metody pro zpracování platby. Díky tomuto je přidání dalšího způsobu platby pouze otázkou napsání třídy, která bude anotovaná anotací `Payment` a bude rozšiřovat funkcionální rozhraní `IPayment` s implementací metody `pay()`, která zpracuje platbu. Takto napsanou třídu není třeba nikde dál uvádět díky anotaci `Payment` bude automaticky obsažena v mapě dostupných plateb.

##### **4.1.3.4.1 EmailService**

Poskytuje metody pro načtení rozložení emailu a pro poslání emailu s možností přílohy.

#### **4.1.4 Repositáře**

Každé rozhraní rozšiřuje generické rozhraní `MongoRepository`. Díky tomu je přístup da databáze řešen frameworkem, který implementuje základní operace CRUD. Pro pokročilejší vyhledávání je zde použita anotace `@Query`.

```
@Query("{ 'halls.?0' : { $exists : true } }")
Cinema findByHall(String hallId);
```

*Použití `@Query` pro načtení kina, ve kterém je sál se zadaným id*



## 4.2 Princip komunikace

Komunikace mezi backendem a frontendem je založena na protokolu HTTP (Hypertext Transfer Protocol). Díky nasazení projektu za reverzní proxy se ke komunikaci používá zabezpečená verze HTTPS.

Veškerá data, která si aplikace vyměňují, jsou textové řetězce ve formátu JSON (JavaScript Object Notation). Tento formát dat umožňuje v podstatě jakoukoliv API komunikaci

Pro autorizaci v rámci aplikace slouží JWT (JSON WEB TOKENy). O jejich samotnou implementaci se stará jose4j.

```
<dependency>
  <groupId>org.bitbucket.b_c</groupId>
  <artifactId>jose4j</artifactId>
  <version>0.9.4</version>
</dependency>
```

*Závislost, která pracuje s JWT*

## 4.3 Koncové body a HTTP metody

Metody HTTP použité pro volání API odpovídají specifikaci. GET – načítá data, POST – předá data, PUT – edituje data, DELETE – odstraní dat. Odpovědi na požadavky mohou vracet stavové kódy:

- 200 (Ok) – když vše proběhne správně
- 202 (přijato) – když při přihlašování bude vyžadována druhá fáze ověření
- 400 (špatný požadavek) – když data nejsou v požadovaném formátu
- 401 (neautorizováno) – když je vyžadována autorizace
- 403 (přístup odepřen) – když uživatel nemá potřebná oprávnění
- 404 (nenalezeno) – když záznam nebyl nalezen
- 405 (metoda není povolena) – když koncový bod existuje, ale nic nezpracovává
- 500 (interní chyba serveru) – Nastala neočekávaná chyba na serveru

### 4.3.1 Controllery se standartním fungováním

Tyto controllery poskytují koncové body, které poskytuje abstraktní třída RestController a jejich chování neupravují.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/age_categories /api/users /api/genres /api/people	/	GET	získá všechny záznamy	200, 401, 403, 500
	/ {id}	GET	získá konkrátní záznam	200, 401, 403, 404, 500
	/	POST	vytvoří nový záznam	200, 400, 401, 403, 500
	/ {id}	PUT	edituje stávající záznam	200, 400, 401, 403, 404, 500
	/ {id}	DELETE	odstraní záznam	200, 401, 403, 404, 500

Obrázek 7 Tabulka s koncovými body standartních controllerů

### 4.3.2 Controllery s deaktivovanými koncovými body

Níže zmíněné controllery se od standartních liší tím, že dostupnost některých koncových bodů, které dědí z RestControlleru, nebyla žádoucí.

Získávání všech se sedadel nemá žádný význam a správa sedadel jednoho po druhém by bylo pro uživatele příliš nepohodlná. Správa sedadel je řešena při spravování sálů. Proto jsou níže uvedené metody nedostupné.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/seats	/	GET	metoda není povolena	405
	/ {id}	GET	získá konkrétní sedadlo	200, 401, 403, 404, 500
	/	POST	metoda není povolena	405
	/ {id}	PUT		
	/ {id}	DELETE		

Obrázek 8 Tabulka s koncovými body sedadel

Města jsou uchovávána pro menší redundanci dat. Jsou spravována při správě kin. Jejich samostatné spravování by činnost zbytečně komplikovalo. Oprávnění není možné přidávat a editovat ani mazat už jen jako ochrana proti administrátorům aplikace samotné.

Pokud by mělo dojít k nasazení aplikace nejlepší způsob editace těchto dat by byl přímo v databázi osobou k tomu pověřenou.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/cities /api/permissions	/	GET	získá všechny záznamy	200, 401, 403, 500
	/id	GET	získá konkrétní záznam	200, 401, 403, 404, 500
	/	POST	metoda není povolena	405
	/id	PUT		
	/id	DELETE		

Obrázek 9 Tabulka s koncovými body měst a oprávněních

Role jsou implementovány bez možnosti přidání nové nebo smazání staré. Aby bylo možné řešit přidávání a mazání muselo by dojít k ustanovení komplexní politiky rolí (například by bylo třeba řešit které role může spravovat, které role, nebo zda má role oprávnění přidělovat konkrétní oprávnění). V kontextu celé této aplikace se tato problematika zdá být zbytečná. Pokud by bylo nejhůř osoba za to zodpovědná by mohla vytvořit novou roli přímo v databázi.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/roles	/	GET	získá všechny role	200, 401, 403, 500
	/id	GET	získá konkrétní roli	200, 401, 403, 404, 500
	/	POST	metoda není povolena	405
	/id	PUT	edituje stávající roli	200, 400, 401, 403, 404, 500
	/id	DELETE	metoda není povolena	405

Obrázek 10 Tabulka s koncovými body rolí

### 4.3.3 Controllery s poupraveným chováním

Jedná se o controllery, které upravují nebo přidávají koncové body.

Controller pro kina přidává koncový bod, který vrací kino, ve kterém se nachází sál d daným id.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/cinemas	/	GET	získá všechna kina	200, 401, 403, 500
	/id	GET	získá konkrétní kino	200, 401, 403, 404, 500
	/by_hall/id	GET	získá kino podle id sálu	
	/	POST	vytvoří nové kino	200, 400, 401, 403, 500
	/id	PUT	edituje stávající kino	200, 400, 401, 403, 404, 500
	/id	DELETE	odstraní kino	200, 401, 403, 404, 500

Obrázek 11 Tabulka s koncovými body kin

U filmů je třeba s API získat filmy, které se mají použít pro galerii na úvodní stránce (aktuální trháky). Dále controller disponuje koncovým bodem pro nahrání obrázku k filmu

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/films	/	GET	získá všechny filmy	200, 401, 403, 500
	/block-busters	GET	získá filmy určené pro prezentaci	200, 401, 403, 404, 500
	/id	GET	získá konkrétní film	
	/	POST	vytvoří nový film	200, 400, 401, 403, 500
	/store-img	POST	nahrává obrázek na server	200, 400, 401, 403, 404, 500
	/id	PUT	edituje stávající film	
	/id	DELETE	smaže film	200, 401, 403, 404, 500

Obrázek 12 Tabulka s koncovými body filmů

Aby bylo možné zachovat integritu databáze není z ní možné odebrat ty sály, ve kterých už bylo naplánováno nějaké promítání. Uživatel na stránce vidí, které sály může odebrat a které ne. Pro zjištění toho, které sály je možné odebrat a které by se odebrat nepovedlo, poskytuje specifický koncový bod.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/halls	/	GET	získá všechny sály	200, 401, 403, 500
	/id	GET	získá konkrétní sál	200, 401, 403, 404, 500
	/unremovable/cinemaId	GET	získá sály konkrétního kina, které není možné odebrat	
	/	POST	přidá nový sál	200, 400, 401, 403, 500
	/id	PUT	edituje stávající sál	200, 400, 401, 403, 404, 500
	/id	DELETE	metoda není povolena	405

Obrázek 13 Tabulka s koncovými body sálů

Při získávání naplánovaných promítání je pro aplikaci důležité, zda promítání už proběhlo, nebo ne. Controller upravuje metodu pro získání všech záznamů tak, že vrací pouze budoucí promítání. Také umožňuje získat promítání, která budou probíhat podle id konkrétního filmu. Aby bylo možné prohlížení už proběhlých představení implementuje ještě koncový bod pro archivovaná promítání.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/projections	/	GET	získá všechny promítání, které ještě neproběhly	200, 401, 403, 404, 500
	/by/film/{filmId}	GET	získá konkrétní promítání, pro daný film, která ještě neproběhla	
	/archived	GET	získá všechny promítání, které proběhly	
	/id	GET	získá konkrétní promítání	200, 400, 401, 403, 500
	/	POST	vytvoří nové promítání	
	/id	PUT	edituje stávající promítání	200, 400, 401, 403, 404, 500
	/id	DELETE	smaže stávající promítání	200, 401, 403, 404, 500

Obrázek 14 Tabulka s koncovými body představení

Při vytváření rezervace dochází okamžitě k provedení platby, takže editace rezervace možná není. K rezervová jsou potřebná jiná data, než předpokládá abstraktní RestController

a proto je i standartní koncový bod pro vytvoření deaktivován a nahrazen jiným. Aby bylo možné zjistit, jaká místa už jsou na jakém promítání už rezervovaná poskytuje controller ještě koncový bod pro „cenzurované načtení“ rezervací pomocí id promítání (Cenzura znamená odebrání konkrétních typů lístků, uživatele a data, kdy rezervace proběhla).

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/api/reservations	/	GET	získá všechny rezervace toho uživatele, který provedl dotaz	200, 401, 403, 500
	/censored/{projectionId}		získá všechny rezervace na konkrétní promítání (v cenzurované podobě)	200, 404, 500
	/id		získá konkrétní rezervaci	200, 401, 403, 404, 500
	/reservate	POST	vytvoří novou rezervaci	200, 400, 401, 403, 500
	/		metoda není povolena	405
	/id	PUT		
	/id	DELETE	odebere rezervaci	200, 400, 401, 403, 404, 500

Obrázek 15 Tabulka s koncovými body rezervací

#### 4.3.4 AuthController

Koncové body tohoto controlleru neslouží pro vyměňování dat aplikace jako všechny ostatní, ale jsou zodpovědné za autentizaci uživatele.

Cesta ke kontroleru	Endpoint	HTTP metoda	Funcionalita	Možné stavy
/auth	/register	POST	registruje nového uživatele	200, 400, 403, 500
	/activate-account		aktivace neaktivovaného účtu	
	/reset-activation-code		vygeneruje nový aktivační kód	200, 403, 500
	/login		přihlásí uživatele	200, 202, 400, 403, 404, 500
	/second-verify		provede druhou fázi ověření	200, 403, 404, 500
	/logout		odhlásí přihlášeného uživatele	200, 500
	/change-pw		přihlášenému uživateli změni heslo	200, 400, 403, 404, 500
	/forgotten-password/reset-code		vygeneruje kód pro obnovu hesla	200, 403, 500
	/forgotten-password/		obnoví uživateli heslo	200, 400, 403, 500
	/token	GET	vrátí uživateli přístupový token	200, 403, 500

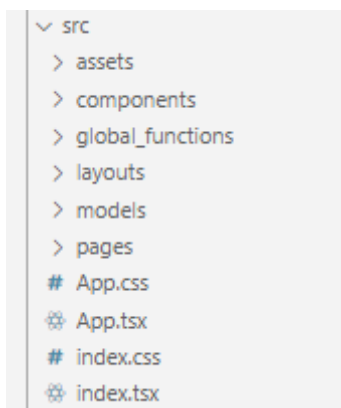
Obrázek 16 Tabulka s koncovými body pro autentizaci

## 5 Frontend projektu

Frontendová část je napsaná v jazyku JavaScript a běží v prostředí Node.js s použitím knihovny React s nadstavbou TypeScript.

### 5.1 Struktura kódu

Kód je strukturován do balíčků:



Obrázek 17 Struktura frontendové části

- assets – uložení obrázků
- components – komponenty, balíček se dělí na další balíčky. Každý vnořený balíček souvisí se stejně jmennou stránku
- global\_functions – zde jsou umístěny funkce volané na více místech aplikace. Např.: generické volání API
- layouts – obsahuje layouty
- models – balíček obsahuje, šablony pro datové struktury
- pages – balíček obsahuje komponenty, které vykreslují celé stránky

#### 5.1.1 Volání API

Volání API je zpracováno genericky v souboru ServerApi. Funkce pro volání API jsou asynchroní. Pro výměnu dat slouží tyto:

- **loadData()** – Slouží pro načtení dat z backendu. Generický parametr určí typ vrácených dat. Parametr modelEndpoint určí koncový bod, který se má použít pro volání API. Parametr ids má výchozí hodnotu nastavenou na prázdné pole a určuje,

jaké záznamy se mají načíst. Pokud dojde k volání funkce s parametrem `ids`, který se rovná prázdnému poli, předpokládá se, že je snaha načíst všechna data. Návrátová hodnota je vždycky pole záznamů.

- **storeData()** – Slouží pro uložení dat na backend. Generický parametr určuje typ záznamů, které vrátí volání API. Metoda pak vrací pole záznamů, které jsou typu `Entity`. Parametr `modelEndpoint` určí koncový bod, který se má použít pro volání API. V parametru `data`, jsou záznamy, které se mají nahrát na backend. To, zda se použije HTTP metoda `POST` nebo `PUT` se rozhodne podle vlastnosti `id`. Když je `id` `null` nebo `undefined`, tak, se volá `POST` jinak se volá `PUT`. Funkce vrací pole `Entity` (typu `Entity`), která dostane jako odpovědi na požadavky.
- **deleteData()** – Slouží pro smazání dat v backendu. Generický parametr určuje typ dat parametru `data`. Parametr `modelEndpoint` určí koncový bod, který se má použít pro volání API. Při úspěšném odebrání metoda vrací data, která dostala jako parametr.

Parametr `modelEndpoint` je typu `ModesEndpoints`. `ModesEndpoints` je výčtový typ (enum), který obsahuje dostupné koncové body. Tyto metody chyby nezpracovávají, ale vyhazují je výš (z důvodu debugování jsou dotazy na API v try catch blocích i přesto, že v bloku catch výjimku pošlou výš).

### 5.1.2 Modely

Modely jsou TypeScriptové třídy, které odpovídají modelům v backendové části. Aby bylo možné použít generické parametry v API metodách musejí modely rozšiřovat třídu `Entity`. Ta definuje parametr `id`. Modely, které nerozšiřují `Entity` jsou ty, které odpovídají DTO modelům (`ReservationDTO`, `TokenDeviceId`). V souborech s definicí modelu je i definována instance to usnadňuje vložení úvodních dat do formuláře.

### 5.1.3 Stránky

Stránky jsou, stejně jako jejich části, reactové komponenty. Jsou rozděleny do balíčků podle toho k čemu souží.

- `usersPages` – stránky pro standartní chování na webu (domovská stránka, stránky s filmy, stránka pro rezervování ...)

- management – zde jsou umístěny stránky pro zobrazení a editaci dat
- login – v tomto balíčku je nachází stránky pro přihlášení, registraci, ověření a pro změnu a obnovu hesla

## 5.2 Routování

Směrování zajišťuje komponenta `App.tsx` prostřednictvím `BrowserRouter`. Jednotlivé routy obsahují parametry:

- `path` – URL adresa stránky
- `element` – element, který se má na stránku vykreslit

Routy se mohou zanořovat do jiných. Obsah elementů ve vnořených routách se do nadřazených pomocí komponenty `<Outlet/>`

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<MainLayout />}>
      <Route index element={<Home />} />

      {/** Stránky pro přihlašování */}
      <Route path="/register" element={ (verifyAccess() ?
loginErr : <Register />) }/>
      <Route path="/login" element={ (verifyAccess() ? loginErr
: <Login />) }/>
      {/** Stránky pro rezervování */}
      <Route path="/film/:filmId" element={ <FilmDetail /> }/>
      <Route path="/my-reservation/:userId" element={
(verifyAccess() ? <MyReservations /> : accessDenite) } />
      <Route path="/management">
        <Route path="" element={ (verifyAccess("projection-
create") ? <Management /> : accessDenite) } />
```

*Ukázka části routovací komponenty*

Jak je vidět v ukázce, tak všechny stránky používají jeden základní layout, který je hodně obecný a specifikuje jen navigaci a footer. Elementy se jsou vyplňovány na základě ověření práv.



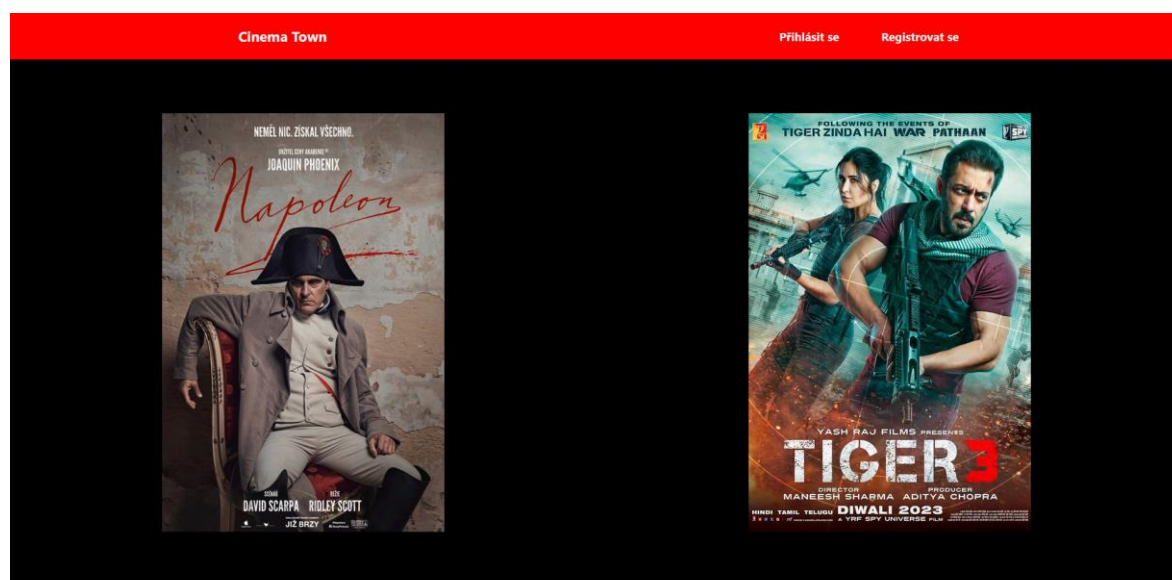
## 6 Popis jednotlivých částí aplikace

V této části se budu věnovat jednotlivým stránkám, které si může uživatel zobrazit.

### 6.1 Úvodní stránka / Program

Po otevření webové stránky uživatele překvapí rovnou stránka s programem, která je rozdělena do tří částí (navigaci nepočítám).

První částí tvoří automatická galerie obrázků filmů, které byly zvoleny jako propagační. Galerie v nezmenšené formě obsahuje dva filmy a ve zmenšené verzi jeden. Obrázky se automaticky mění po 10 sekundách. Nové filmy přijíždí z prava

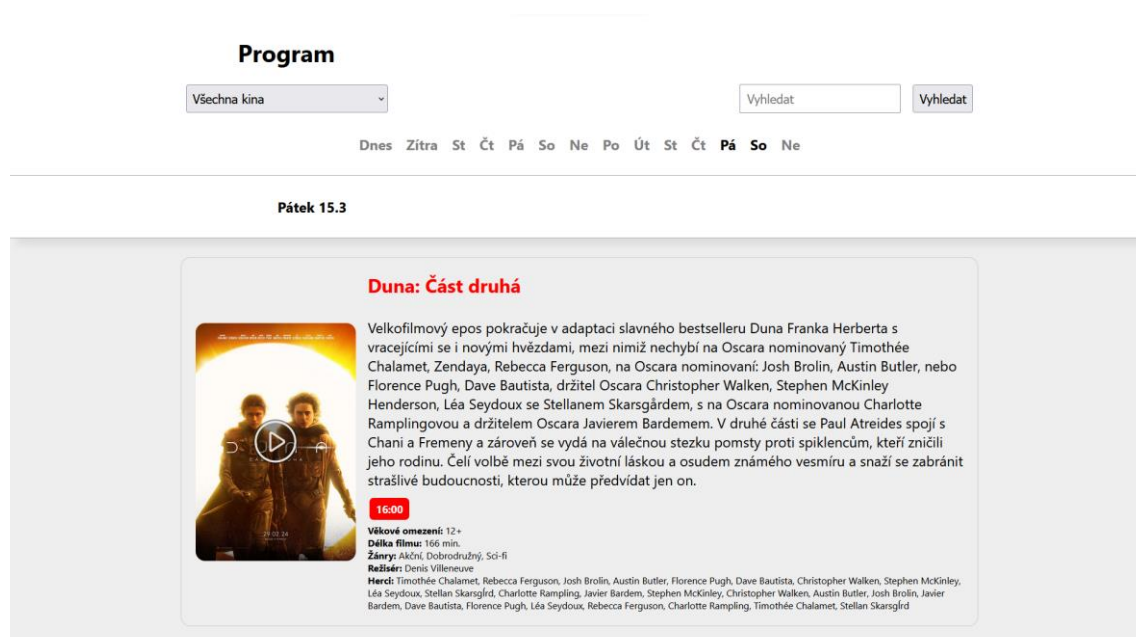


#### Program

Obrázek 18 Úvodní stránka 1. část

Druhou část tvoří filtr, ten umožňuje vybrat jedno kino a také použít vyhledávací pole, které vyhledává automaticky při psaní a porovnává podle: jména filmu a názvu každého žánru každého filmu. Pokud není vybráno žádné kino (počáteční stav), tak je jedno v jakém kině bude probíhat promítání. Vyhledávací okno nerozlišuje malá a velká písmena, myslím si, že je to tak lepší (pro uživatele pohodlnější). Poslední částí filtru je pak čtrnácti denní přehled. Po kliknutí na den je uživatel přesunut na sekci, kde začínají filmy, které se v daný den budou promítat

Poslední, třetí, částí úvodní stránky jsou pak samotná naplánovaná promítání. Ta jsou dělena podle dnů, kdy k nim má dojít, do sekcí. Sekce jsou uspořádané podle data seshora dolů. Promítání se skládá z obrázku filmu, nadpisu, popisu filmu a dalších informací, které by mohly budoucího návštěvníka zajímat (žánry, režisér, herci, doba trvání ...). Favikon uprostřed obrázku vyzívá uživatele, aby si přehrál trailer k filmu (trailer je umístěn na YouTube a do stránky jen vložen). Uživatel se může přes název filmu překliknout na detail filmu a odtud provést rezervaci. Červené tlačítko s časem pak uživateli v detailu po kliknutí na tlačítko zakoupit lístky vyplní promítání, ke kterému je vázáno.



Obrázek 19: Úvodní stránka část 2 a 3

## 6.2 Registrační a přihlašovací formuláře

Pro přihlášení a registraci jsem zvolil jednoduchý design s textem v levé části a formulářem v pravé části.

Protože jediné, co je třeba k tomu, aby uživateli došli vstupenky na email je jeho email, je registrační formulář velmi jednoduchý. Po registraci je třeba ověřit email (na email uživatel dostane kód, který mu tam byl zaslán). Uživatel má možnost si nechat vygenerovat kód znovu. Účet, který není aktivovaný se nepropíše do DB a pokud dojde k restartu serveru, tak budou neaktivovaní uživatelé zapomenuti.

Cinema Town

Přihlásit se

Registrovat se

**Získej možnost rezervování míst na své oblíbené filmy a nenech si tuto příležitost už nikdy utéct.**

Registruj se nyní.

Email

email@email.com

Heslo

heslo

Heslo znovu

heslo

Registrovat

Aktivovat účet

Obrázek 20 Stránka pro přihlašování

Pokud uživatel neaktivuje účet a pokusí se o přihlášení. Bude požadována aktivace, kterou aplikace vezme jako dvoufázové ověření. Na stránce pro přihlášení má uživatel možnost resetovat si heslo. Aby bylo možné resetovat heslo musí uživatel zadat email. Aplikace ho přesměruje na formulář, ve kterém vyplní nové heslo a zadá kód, který mu byl zaslán na email.

Cinema Town

Přihlásit se

Registrovat se

Obnovovací kód

Nové heslo

Nové heslo znovu

Zpět

Změnit

Obrázek 21 Formulář pro obnovu hesla

Aby nebylo možné spamovat emailovou schránku uživatelů jsou požadavky na změnu hesla časově limitované.

## 6.3 Detail filmu

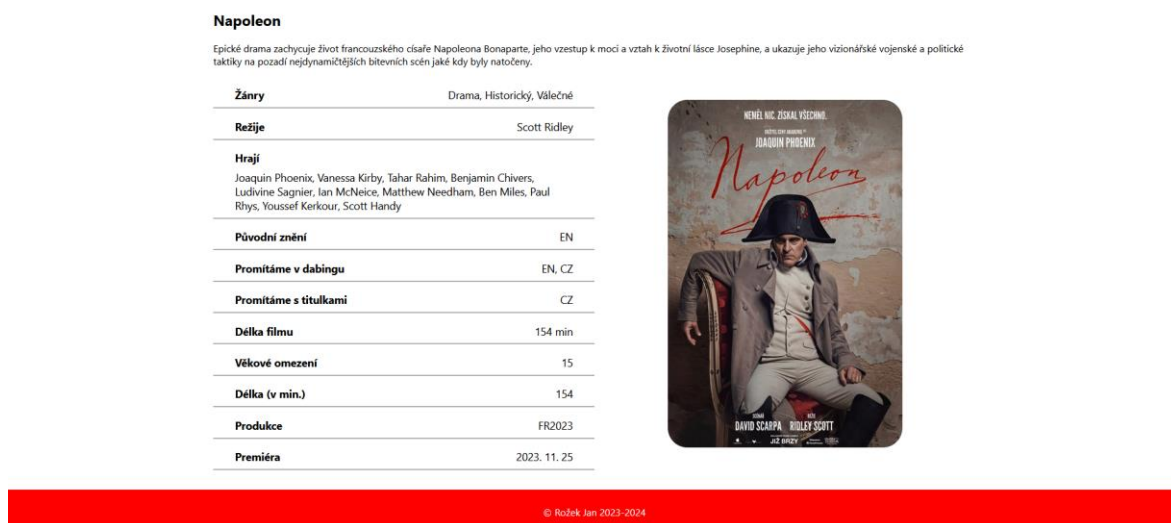
Tato stránka zobrazí uživateli všechna data o filmu spolu s trailerem na film a tlačítkem pro rezervaci. Celkem je stránka rozdělena do dvou bloků.

První blok tvoří trailer a tlačítko pro provedení rezervace. Pokud není naplánované promítání pro daný film a uživatel by se na něj dostal (url není hlídána pro tuto podmínku) bude tlačítko pro rezervování neaktivní.



Obrázek 22 První blok stránky s detailem filmu

Druhý blok obsahuje všechny informace o filmu a plakát. Popis filmu je nad ostatními informacemi, které jsou v tabulce. Vedle tabulky je pak umístěn plakát.



Obrázek 23 Druhý blok stránky s detailem filmu

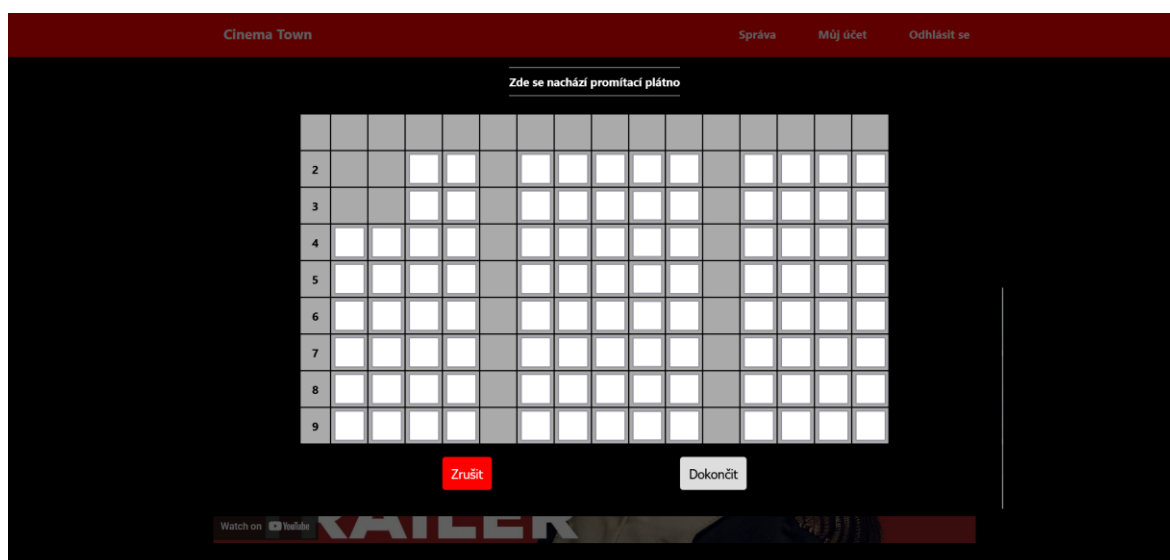
Po kliknutí na tlačítko rezervovat se zobrazí dialogové okno pro provedení rezervace (pokud uživatel není přihlášený je přesměrován na stránku pro rezervaci). Okno pro provedení rezervace obsahuje select pro výběr kina. Po vybrání kina se uživateli zobrazí termíny, kdy se bude konat ve vybraném kině promítání konkrétního filmu (pokud uživatel na detail přejde z úvodní stránky pomocí tlačítka s časem, tak se kino a termín vyplní automaticky).

The screenshot shows a reservation interface for the movie "Napoleon" on the Cinema Town website. The header includes the site name and navigation links. The main content area displays the movie title, a dropdown menu for selecting a cinema (currently set to "Ostrava, Divadelní, 78"), and a section for choosing a screening time. Two time slots are available: "3. 13. 20:00" and "3. 13. 19:30". Below this is a table for selecting ticket categories and quantities. The table has four columns: "Cenová kategorie", "Cena lístku", "Počet lístků", and "Cena celkem". The categories listed are "Dítě (do 15 let)", "Důchodce (65+)", "Dospělý", and "Student". Each category has a corresponding price and a quantity selector. At the bottom, there is a button labeled "Zde se nachází promítací plátno" and a "Watch on YouTube" link.

Cenová kategorie	Cena lístku	Počet lístků	Cena celkem
Dítě (do 15 let)	225 Kč	0	0 Kč
Důchodce (65+)	225 Kč	0	0 Kč
Dospělý	300 Kč	0	0 Kč
Student	225 Kč	0	0 Kč

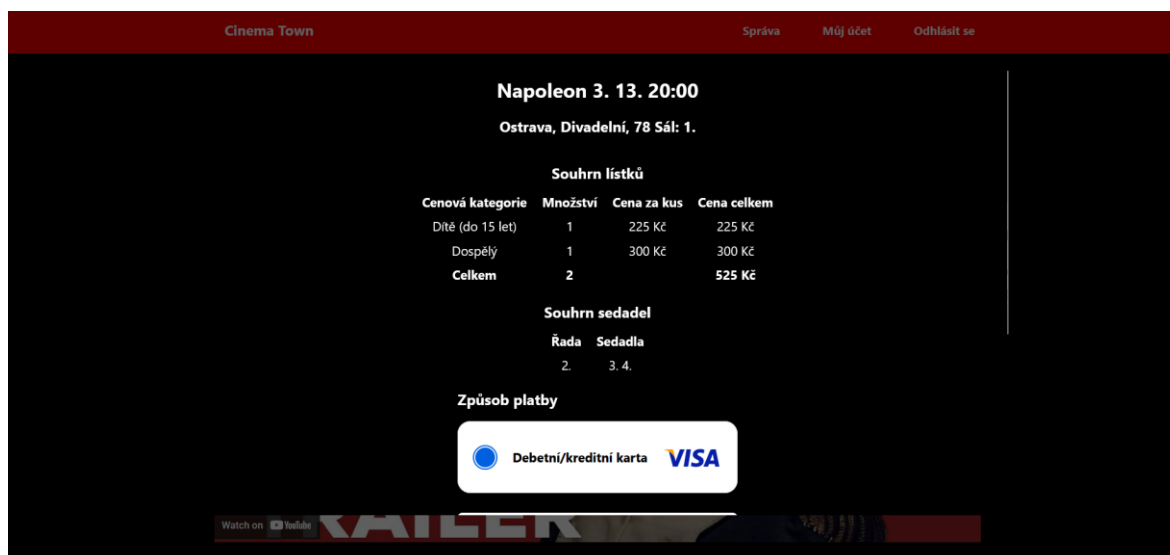
Obrázek 24 Hlavička dialogového okna pro rezervování

Po vybrání konkrétního termínu promítání se uživateli zobrazí pole s výběrem sedadel. Před tím, než uživatele může vybírat sedadla si musí zvolit jaké lístky a kolik jich bude chtít. Když už začne vybírat místa, tak není možné snížit počet lístků, tak, aby byl menší než počet. Když se celkový počet lístků shoduje s počtem vybraných sedadel, tak se tlačítko „dokončit“ stane aktivním a uživatel se může přesunout k dokončení rezervace.



Obrázek 25 Ukázka sálu, pro výběr sedadel

Po kliknutí na tlačítko pro dokončení se uživateli zobrazí další dialog, ve kterém uživatel zkontroluje rezervaci a provede platbu. Platby je v projektu umožněna jen pomocí Visy (způsob platby je napsán s jednoduchou rozšiřitelností) (protože využívám k ověření platby paywall Stripe a proto je možná jen karta 4242 4242 4242 4242 a libovolnou, validní platností karty a cvv/cvc). Po validním zadání platebních údajů bude moci uživatel dokončit platbu, tím, že klikne na tlačítko zaplatit. Pokud platba proběhne úspěšně bude uživatele přesměrován na stránku „moje rezervace“.



Obrázek 26 Přehled rezervace a zvolený způsob platby

## 6.4 Moje rezervace

Stránka „moje rezervace“ obsahuje v horní části informace o uživatelském účtu (název a zda je odběratel) spolu s informací, zda se tomu zařízení, přes které je uživatel přihlášený, důvěřuje.

Vedle těchto informací jsou tlačítka pro změnu hesla, přihlášení se k odběru a pokud se zařízení důvěřuje zobrazí se tlačítko pro odebrání důvěry pro toto zařízení.

Druhou část stránky tvoří filtr, který filtruje rezervace přihlášeného uživatele (filtrující podle: názvu filmu, datumu, času a názvu města). Rezervace jsou seskupeny do skupin podle dne promítání. Každá skupina má ve své hlavičce datum a počet rezervací na konkrétní datum. Pod hlavičkou jsou vypsány uživatelovy rezervace.

Cinema Town Správa Můj účet Odhlásit se

Ahoj, rozekja20@zaci.spse.cz Změnit heslo

Odběratel: Ne Přihlásit se k odběru

Důvěřovat tomuto zařízení: Ano Přestat důvěřovat

Filtruj

15.3.2024 Počet rezervací 1

Rezervace na film **Duna: Část druhá**

Multikino: Kamera 3, Plzeň

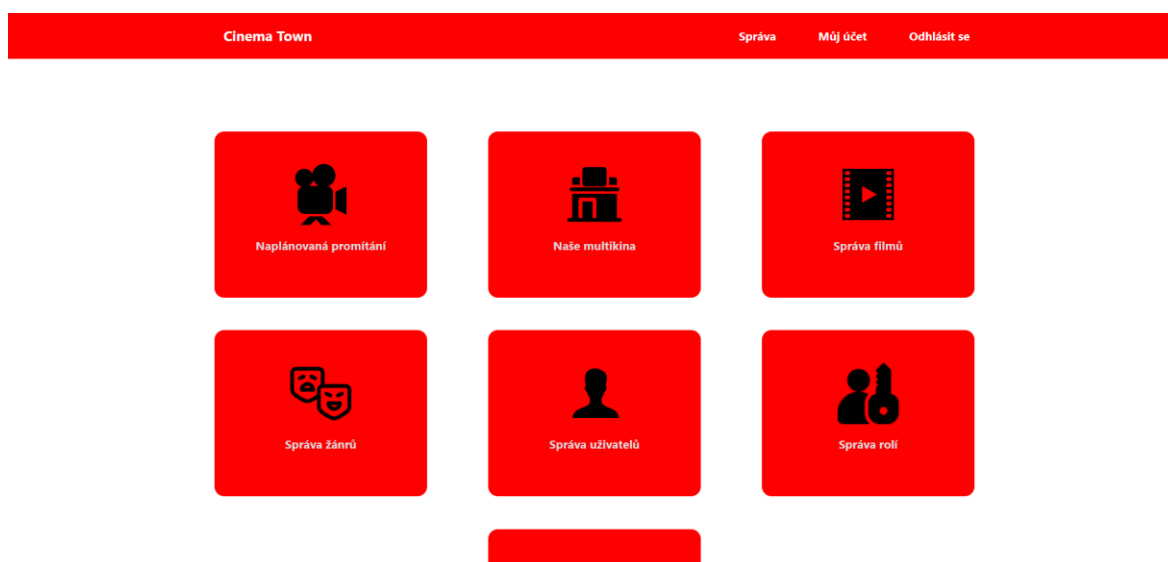
Konání představení: 15. 3. 16:00

Rezervace vytvořena: 7. 3. 09:44

Obrázek 27 Ukázka stránky Moje rezervace

## 6.5 Správa

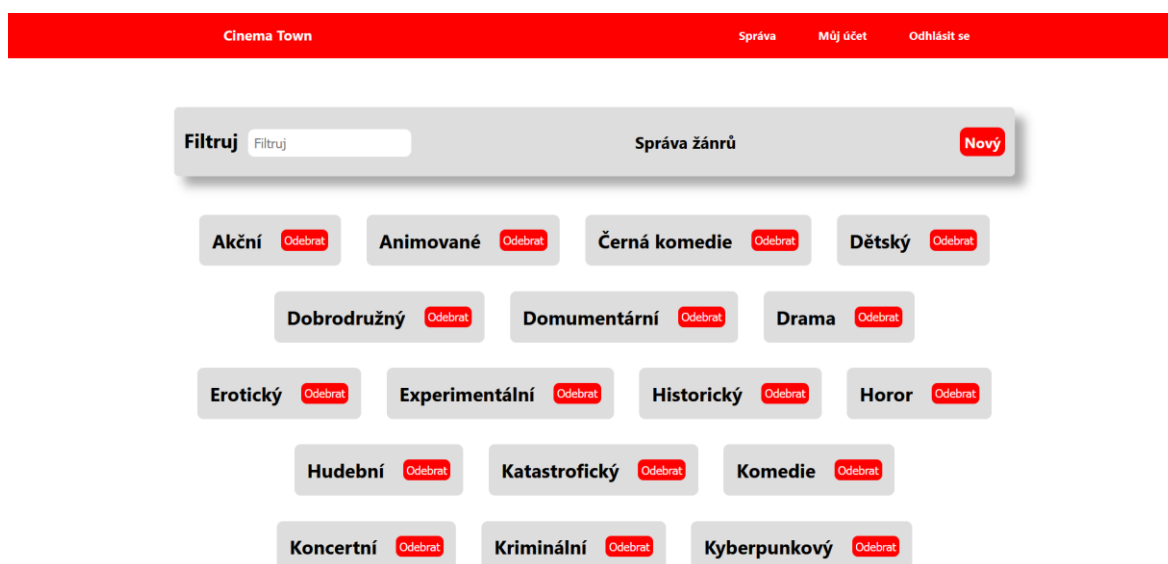
Stránka funguje jako rozcestí na další stránky s administrační funkcí. Má jednoduchý design skládající se z dlaždic. Dlaždice má ve svém středu favicon a pod ním text. Text i favicon sdělují administrátorovi to kam ho kliknutí na dlaždici nasměruje.



Obrázek 28 Ukázka rozcestí k administraci

### 6.5.1 Přehled žánrů a spol.

Stránky, které reprezentují data, která může administrátor upravovat mají jednotný a jednoduchý design. V hlavičce stránky je umístěn filtr, název dat, která jsou zobrazena a tlačítko na přidání nového záznamu. Pod hlavičkou pak jsou zobrazena data ve formě dlaždic. Na editaci určitých dat se administrátor dostane kliknutím na dlaždici. Tlačítko pro odebrání vyžaduje potvrzení před tím, než se pokusí data odebrat.




Obrázek 29 Ukázka prezentace dat



## 6.5.2 Editování a přidávání žánrů a spol

Stránky pro editaci a přidávání dat jsou tvořeny formulářem, který je vycentrovaný na střed. Po kliknutí na tlačítko „Zahodit změny“ je administrátor přesměrován na stránku se všemi daty. Tlačítka „Potvrdit změny“ propíše změny do databáze.



The screenshot shows a web application interface for Cinema Town. At the top, there is a red navigation bar with the text 'Cinema Town' on the left and three links: 'Správa', 'Můj účet', and 'Odhlásit se' on the right. Below the navigation bar, the main content area is white. In the center, there is a light gray rounded rectangle containing a form titled 'Kategorie: Student'. The form has two input fields: the first is labeled 'Název kategorie' and contains the text 'Student'; the second is labeled 'Cenový modifikátor' and contains the value '0,75'. At the bottom of the form, there are two red buttons: 'Zahodit změny' on the left and 'Potvrdit změny' on the right.

Obrázek 30 Ukázka editačního formuláře

## Závěr

Mým hlavním cílem bylo napsat jednoduchou aplikaci pro rezervace filmů a informování široké veřejnosti o budoucích promítání ve vlastních multikinech. Věřím, že se mi to podařilo. Aplikace by ve své současné podobě potřebovala vyřešit vracení peněz přes paywall Stripe, protože v tuto chvíli je implementována pouze platba při rezervaci. Dále by bylo před nasazením potřeba přidat na Stripu použitelné platební údaje (aplikace využívá demo verzi paywallu).

Při zpracovávání projektu jsem se hodně posunul ve využívání knihovny React, obzvláště v rovině psaní generického kódu, a osvojil si základní práci z TypeScriptem, se kterým jsem dříve nepřišel do styku. Při vytváření backendové části jsem vytvořil několik generických tříd, které mi usnadnily vývoj. Třídy implementovali základní logiku CRUD a vystavili ji pomocí protokolu HTTP do internetu. Moje další úsilí mimo tento projekt bude vést ke zdokonalení těchto tříd.

## Seznam přístupových údajů

URL adresa webu: [www.mp.home-lab.rozekja.fun/](http://www.mp.home-lab.rozekja.fun/)

Úroveň oprávnění	Přihlašovací jméno	Heslo
Administrátor	ucet1.cinema.town@gmail.com	J0w89jROWMHVzIDiriw4
Registrovaný uživatel	ucet2.cinema.town@seznam.cz	8uWeVRsncX3NsUmkX3xx

Heslo jsou stejná jak pro přihlášení do aplikace, tak do emailových schránek.

## Seznam použité literatury a zdrojů obrázků

Data týkající se filmů stejně jako obrázky k nim jsem převzal z webu: [cinemacity.cz](https://cinemacity.cz)

- [1] JavaScript. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2004. Dostupné z: <https://cs.wikipedia.org/wiki/JavaScript>. [cit. 2024-03-11].
- [2] ITNETWORK. Lekce 1 - Úvod do JavaScriptu. Online. HARTINGER, David. Itnetwork.cz. 2013. Dostupné z: Zdroj: <https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk>. [cit. 2024-03-11].
- [3] TypeScript: JavaScript With Syntax For Types. Online. C2012-2024. Dostupné z: <https://www.typescriptlang.org/>. [cit. 2024-03-11].
- [4] TypeScript. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2023. Dostupné z: <https://cs.wikipedia.org/wiki/TypeScript>. [cit. 2024-03-11].
- [5] React (software). Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: [https://en.wikipedia.org/wiki/React\\_\(software\)](https://en.wikipedia.org/wiki/React_(software)). [cit. 2024-03-11].
- [6] React. Online. C2024. Dostupné z: <https://react.dev/>. [cit. 2024-03-11].
- [7] Node.js. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: <https://cs.wikipedia.org/wiki/Node.js>. [cit. 2024-03-11].
- [8] Npm. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: <https://en.wikipedia.org/wiki/Npm>. [cit. 2024-03-11].
- [9] Java (programovací jazyk). Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: [https://cs.wikipedia.org/wiki/Java\\_\(programovac%C3%AD\\_jazyk\)](https://cs.wikipedia.org/wiki/Java_(programovac%C3%AD_jazyk)). [cit. 2024-03-11].
- [10] Java | Oracle. Online. C2024. Dostupné z: <https://www.java.com/en/>. [cit. 2024-03-11].
- [11] Spring | Home. Online. C2005-2024. Dostupné z: <https://spring.io/>. [cit. 2024-03-11].

- [12] Spring Boot. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: [https://en.wikipedia.org/wiki/Spring\\_Boot](https://en.wikipedia.org/wiki/Spring_Boot). [cit. 2024-03-11].
- [13] Apache Maven. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: [https://cs.wikipedia.org/wiki/Apache\\_Maven](https://cs.wikipedia.org/wiki/Apache_Maven). [cit. 2024-03-11].
- [14] Maven – Welcome to Apache Maven. Online. C2002-2024. Dostupné z: <https://maven.apache.org/>. [cit. 2024-03-11].
- [15] MongoDB: The Developer Data Platform | MongoDB. Online. C2024. Dostupné z: <https://www.mongodb.com/>. [cit. 2024-03-11].
- [16] MongoDB. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: <https://cs.wikipedia.org/wiki/MongoDB>. [cit. 2024-03-11].

## Seznam obrázků

Obrázek 1 CineStar (zdroj: vlastní) .....	11
Obrázek 2 Cinema City (zdroj: vlastní) .....	12
Obrázek 3 Kino Světozor (zdroj: vlastní) .....	13
Obrázek 4 Use Case diagram .....	15
Obrázek 5 Struktura databáze UML Class diagram.....	16
Obrázek 6 Struktura backendové části.....	21
Obrázek 7 Tabulka s koncovými body standartních controllerů .....	26
Obrázek 8 Tabulka s koncovými body sedadel .....	26
Obrázek 9 Tabulka s koncovými body měst a oprávněních .....	27
Obrázek 10 Tabulka s koncovými body rolí.....	27
Obrázek 11 Tabulka s koncovými body kin .....	27
Obrázek 12 Tabulka s koncovými body filmů.....	28
Obrázek 13 Tabulka s koncovými body sálů .....	28
Obrázek 14 Tabulka s koncovými body představeních .....	28
Obrázek 15 Tabilka s koncovými body rezervací.....	29
Obrázek 16 Tabulka s koncovými body pro autentizaci.....	29
Obrázek 17 Struktura frontendové části.....	30
Obrázek 18 Úvodní stránka 1. část .....	33
Obrázek 19: Úvodní stránka část 2 a 3 .....	34
Obrázek 20 Stránka pro přihlašování.....	35
Obrázek 21 Formulář pro obnovu hesla.....	35
Obrázek 22 První blok stránky s detailem filmu.....	36
Obrázek 23 Druhý blok stránky s detailem filmu .....	36
Obrázek 24 Hlavička dialogového okna pro rezervování .....	37
Obrázek 25 Ukázka sálu, pro výběr sedadel .....	38

Obrázek 26 Přehled rezervace a zvolený způsob platby .....	38
Obrázek 27 Ukázka stránky Moje rezervace .....	39
Obrázek 28 Ukázka rozcestí k administraci.....	40
Obrázek 29 Ukázka prezentace dat .....	40
Obrázek 30 Ukázka editačního formuláře .....	41

## **Přílohy**

Příloha 1 – Zdrojový kód aplikace na USB disku.

Příloha 2 – Tabulka se všemi koncovými body rovněž na USB disku.