VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ŘÍZENÍ PROVOZU KINA

CINEMA MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETER NÁČIN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Náčin Peter**
Program: Informační technologie
Název: **Systém pro řízení provozu kina**
Cinema Management System
Kategorie: Informační systémy

Zadání:

1. Prostudujte současné webové technologie pro tvorbu aplikací s tlustým klientem.
2. Seznamte se s požadavky konkrétního kina na systém pro řízení provozu zahrnující přidělování služeb a rezervace.
3. Navrhněte architekturu systému a technické řešení.
4. Implementujte navržený systém pomocí vhodných technologií. Uvažujte použití systému jak na stolních, tak na mobilních zařízeních.
5. Proveďte testování vytvořeného systému.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až3

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 30. října 2018

Abstrakt

Cieľom tejto práce bakalárskej práce je analyzovať požiadavky, navrhnúť a implementovať webový informačný systém pre kino. Systém zabezpečuje administráciu, plánovanie pracovných smien a rezerváciu lístkov. Pri tvorbe bola použitá databázová technológia MySQL a framework Laravel s podporou PHP. Uživateľské rozhranie je vytvorené pomocou HTML a CSS.

Abstract

The main goal of the Bachelor Thesis is to analyze demands, design and implementation of a web information system for a cinema. The system deals with the administration, scheduling working shifts and ticket reservation for the cinema. During the development, database technology MySQL and Laravel framework were being used. The user interface is created with the support of HTML and CSS.

Klíčová slova

Informačný systém, Laravel framework, MySQL, PHP, HTML, CSS, Bootstrap.

Keywords

Information systém, Laravel framework, MySQL, PHP, HTML, CSS, Bootstrap.

Citace

NÁČIN, Peter. *Systém pro řízení provozu kina.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Systém pro řízení provozu kina

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta, Ph.d. a uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Náčin
1. května 2019

Poděkování

Touto cestou by som chcel podakovať vedúcemu práce Ing. Radkovi Burgetovi, Ph.D za poskytnutie odbornej pomoci.

Obsah

Kapitola 1

Úvod

Cieľom bakalárskej práce bolo vytvoriť informačný systém pre Cinemax Trenčín, ktorý by:

- umožňoval zamestnancom jednoduchý prístup k nahláseniu si pracovných smien,
- a pre zákazníka systém poskytuje informáciu o filmoch a rezerváciu sedadiel na filmy premietané v kine

Doposiaľ boli pre organizáciu pracovných smien využívané sociálne siete, kde sa informácie vyskytovali v neprehľadnej forme a hlavne bolo obtiažne informácie poskytnúť všetkým zamestnancom.

Navrhovaný informačný systém umožňuje prístup pre zamestnancov a poskytuje jednoduchú a prehľadnú prácu s dátami a zobrazenie pracovných smien v kalendári. V dnešnej dobe sú mobilné zariadenia často používanejšie ako stolové počítače a práve preto je systém navrhnutý a vytvorený ako webová aplikácia, aby podporoval jednoduchý prístup pre väčšinu zariadení a taktiež s pomocou Bootstrapu vytvára vhodný vzhľad aj pre mobilné zariadenia a rieši tak problém s responzivitou(rozlíšením). Práca obsahuje jednotlivé kapitoly, ktoré opisujú postup pri riešení mojej bakalárskej práce.

- Kapitola 2 popisuje súčasnú situáciu a požiadavky kina na informačný systém.
- Nasledujúca kapitola 3 je zameraná na výber a opis použitých technológií pri implementácii, ale taktiež obsahuje aj metodologický postup pri vytváraní práce. Technológie sú rozdelené do troch základných častí a to framework, web a databáza.
- V kapitole 4 je popísaný návrh z pohľadu relačnej databázy. Taktiež je tu návrh v podobe ER a Use Case diagramu.
- Kapitola 5 popisuje hlavné a najzaujímavejšie časti implementácie informačného systému. Takisto je v tejto kapitole opísaná implementácia užívateľského rozhrania a zabezpečenie responzivity.
- Predposledná kapitola 6 sa venuje testovaniu informačného systému a to hlavne z pohľadu výstupu údajov z databázy a responzivite pre mobilné zariadenia.
- Záverečná kapitola 7 zhrňuje jednotlivé kapitoly, celkový pohľad na prácu a taktiež obsahuje popis ďalších možných rozšírenií systému v budúcnosti.

Kapitola 2

Problematika

V nasledujúcej kapitole si objasníme súčasnú situáciu v Trenčianskom kine a spomenieme dôvody prečo by bolo vhodné informačný systém vytvoriť. V druhej časti si uvedieme ciele práce, ktoré by mali byť splnené.

2.1 Súčasný stav

Kino Cinemax Trenčín funguje v Trenčíne už od roku 2006. Spočiatku malo kino niekoľko stabilných zamestnancov, ktorí pracovali počas pracovného týždňa a víkend sa vypĺňal brigádnikmi. Pracovné smeny sa v tomto období dali pripravovať aj na celý mesiac dopredu. Informácie sa ale aj tak udržovali v papierovej forme, ktoré boli fyzicky uložené v kine.

Štruktúra zamestnancov sa ale rokmi menila a to hlavne takým smerom, že zamestnancov na trvalý pracovný pomer nahradili brigádnici. Dala sa tak príležitosť mladšej generácii. Z pôvodných zamestnancov ostalo len veľmi málo a to hlavne na manažérskej pozícii. S touto zmenou vznikli dva problémy.

Prvým problémom je, že brigádnici nemajú stabilnú pracovnú dobu a pracovné smeny sa musia vytvárať na každý týždeň.

Druhým problémom je komunikácia, pretože kino nikdy nemalo k dispozícii informačný systém, všetky administratívne záležitosti boli riešené osobne v kine. Aby sa vyriešil komunikačný problém tak sa časom prešlo na sociálne siete, ktoré ale veľmi situácii nepomohli z dôvodu, že informácie boli veľmi neprehľadné a nie každý je povinný mať účet na sociálnej sieti.

Takisto kino nedisponuje službou, ktorá by zákazníkov informovala filmoch (plánovaných a premietaných) a umožňovala by rezerváciu sedadiel na premietaný film bez potreby osobnej návštevy kina.

2.2 Cieľ práce

Cieľom práce je vytvoriť webový plánovací informačný systém, v ktorom budú mať nie len manažéri, ale takisto aj zamestnanci prehľad o pracovných smenách vo forme kalendára. Systém každému zamestnancovi poskytne možnosť si nahlásiť smenu do kalendára. Aby sa predišlo problému, že jeden zamestnanec si rezervuje pre neho najvýhodnejšie možné smeny, tak manažér bude mať poslednú možnosť smenu potvrdiť či pozmeniť, aby došlo k spravodlivému a rovnomernému rozdeleniu pracovných smien podľa aktuálnych potrieb

kina. Tento systém kinu výrazne uľahčí administratívu a taktiež umožní aj jednoduchý prístup pre zamestnancov kina.

Dôležitou častou práce je aj rezervačný systém sedadiel pre zákazníkov kina. Tento systém umožní zákazníkovi rezervovať si sedadlo na premietaný film. Rezerváciu bude potrebné potvrdiť a zaplatiť osobne pri návšteve kina najneskôr však pár minút pred začatím premietania filmu. V prípade nezaplatenia rezervácie a uplynutí stanoveného času pre rezerváciu miesto prepadne a uvoľní sa pre iných zákazníkov, ktorí čakajú fyzicky v kine. Čas, kedy rezervácia skončí bol stanovený na 10 minút pred začatím premietania filmu.

Keďže je systém navrhnutý ako webová aplikácia umožňuje prístup z rôznych prehliadačov ako Google Chrome, Mozilla Firefox, alebo Internet Explorer. Systém bude možné spustiť nie len v rôznych prehliadačoch na stolových počítačoch, ale takisto bude zabezpečené vhodné zobrazenie na mobilných zariadeniach z dôvodu jednoducho a rýchleho prístupu aj mimo stolových počítačov.

Kapitola 3

Technológie

Informačný systém je implementovaný pomocou frameworku Laravel a používa súčasné technológie v oblasti tvorby a vývoja webových informačných systémov. Tretia kapitola sa zaoberá opisom jednotlivých technológií použitých pri tvorbe systému. Podrobné popísaný je vybraný framework Laravel, ktorý je aj následne porovnaný s ďalšími. Taktiež sa táto často práce zaoberá programovacími jazykmi použitými pri tvorbe back-endu a front-endu.

3.1 Serverová časť

Moderné technológie umožňujú implementáciu webových aplikácii s podporou rôznych operačných systémov, aplikácií či programovacích jazykov. Nami vytvorený webový informačný systém je založený na frameworku Laravel, ktorý sa opiera primárne o programovací jazyk PHP.

3.1.1 Laravel

Prvotnou úlohou z pohľadu implementačnej časti bolo zvolenie si vhodného frameworku, ktorý bude podporovať moderné programovacie jazyky a technológie používané pri tvorbe webových aplikácií. Po prevedení základného výskumu a načerpaní rozhodujúcich informácií som sa rozhodol webový informačný systém implementovať pomocou frameworku **Laravel**[4] a to z dvoch hlavných dôvodov:

- **dostupnosť** - Laravel môžeme považovať za jeden z najrozšírenejších webových PHP frameworkov vo svete a tým pádom je zabezpečená dostupnosť informácií na vysokej úrovni. Kvalitne spracovaná dokumentácia bola prvým dôvodom prečo je práca vytvorená práve v tomto frameworku,
- **angličtina** - druhým dôvodom je jazyk frameworku. Laravel poskytuje široko spracovanú dokumentáciu vo viacerých smeroch, ktorá je zrozumiteľná, verejne prístupná a v prvom rade napísaná v angličtine a iných svetovo používaných jazykoch.

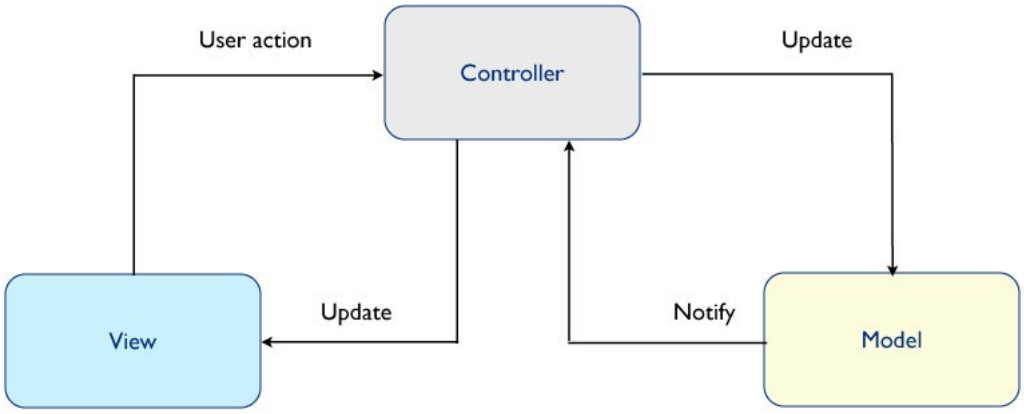
Laravel je voľne dostupný PHP framework, ktorého autorom je Taylor Otwell. Prvá verzia vznikla v roku 2012 s podporou MIT licencie (**MIT** licencia je slobodná softvérová licencia vytvorená Massachusettským technologickým inštitútom). Laravel ako framework slúži primárne na vývoj webových aplikácií a v súčasnosti podporuje multiplatformnosť tzn. že framework je spustiteľný aj na iných počítačových platformách, medzi najznámejšie patrí: Microsoft Windows, Linux či Mac OS X.

Väčšina novších frameworkov je založená na modele MVC, inak povedané **Model-View-Controller model**. Tento model môžeme vidieť na obrázku 3.1

V tejto štruktúre **model** reprezentuje informácie, s ktorými pracujeme. Popisuje štruktúru dát, zabezpečuje ich konzistenciu (celistvosť), ukladá a vyberá dáta v prípade potreby.

View časť zabezpečuje zobrazenie informácií pre užívateľa vo forme grafické užívateľského rozhrania, ktoré je primárne implementované pomocou HTML, CSS a JavaScriptu.

Controller vykonáva funkcie v pozadí, ktoré zabezpečujú zmenu dát v grafickom rozhraní po interakcii užívateľa s častou užívateľského rozhrania.



Obrázek 3.1: Návrhový MVC model [3]

Základná práca s Laravel frameworkom je pomocou konzoly, kde je potrebné použiť príkaz *php artisan*, za ktorým je nadviazaná konkrétna funkcia, ktorú voláme. Tento príkaz je kľúčovým príkazom, pretože je potrebné ho použiť pri každom príkaze súvisiacim s Laravel frameworkom. Pre lepšie porozumenie je vhodné si uviesť veľmi často používané príkazy:

- **php artisan serve** - najzákladnejší príkaz, pomocou ktoré je vytvorený lokálny server. Štandardne je to adresa 127.0.0.1:80, túto adresu je možná zmeniť v konfiguračnom súbore 5.1,
- **php artisan serve –host 0.0.0.0 –port 8000** - parametre –host 0.0.0.0 a –port 8000 sú prednastavené hodnoty. Pomocou parametra –host 0.0.0.0 je vytvorený server dostupný aj pre užívateľov rovnakej LAN siete. Štandardné nastavenie pre web prístup je http://192.168.x.y:port, kde 192.168.x.y je IPv4 adresa servera a port je TCP port, na ktorom server počúva. V tomto prípade je to port 8000. Použitie parametra host umožňuje rozšíriť škálu testovania pri ladení aplikácie či už pre iné platformy, ale aj pre mobilné, alebo ďalšie zariadenia,
- **php artisan make:controller NázovControlleru** - príkazy zahrňujúce *make* slúžia pre vytvárenie nových súborov na konkrétne použitie. V našom prípade to je Controller. Medzi štandardne zaužívané patrí aj *make:migration*, alebo *make:command*,
- **php artisan command:NázovCommandu** - pre použitie tohto príkazu musí programátor najprv vytvoriť požadovaný command a doplniť jeho funkcionalitu a následne je možné ho použiť pomocou konzoly.

Jazyk PHP sa používa najmä v Controller súboroch, ktoré majú na starosti prepojenie medzi front endom a back endom. *Front end* je prezentačná vrstva a *back end* je dátová prístupová vrstva. Tieto súbory môžu napríklad obsahovať aj SQL dopyty (SQL query) na získavanie informácií z databázy, ale taktiež štandardné funkcie vytvorené programátorom.

Navrhovaný informačný systém má mať formu webového informačného systému. Musíme teda riešiť aj výsledné zobrazovanie webovej stánky. Laravel ponúka riešenie v podobe **blade** súborov, ktoré slúžia ako zobrazovacie súbory. Blade súbory používajú zväčša HTML ako programovací jazyk doplnený PHP a JavaScriptom. Pre základné rozdelenie tried elementov (prvkov) HTML použijeme voľne dostupný Bootstrap, pomocou ktorého sa zabezpečí aj responzivita mobilných zariadení.

Spomenuté technológie podrobnejšie popíšem v nasledujúcich podkapitolách.

3.2 Ďalšie frameworky

Pre porovnanie technológií boli vybraté aj ďalšie frameworky, ktoré sa v súčasnosti používajú pri tvorbe webových programov.

3.2.1 Symfony

Tento framework vznikol v roku 2005 pod MIT licenciou a je voľne dostupným frameworkom. Tak ako aj Laravel, je zameraný na tvorbu webových aplikácií s podporou PHP programovacieho jazyka. Symfony, ako aj mnohé ďalšie používa MVC model, viď. 3.1.1.

Vďaka skorému vzniku, kvalitnému spracovaniu a širokým možnostiam, Symfony[6] patrí medzi najpoužívanejšie PHP frameworky v programátorskom svete.

3.2.2 Nette

Je vhodné spomenúť aj Nette[5], ktorý je voľne dostupným PHP frameworkom a bol vyvinutý českým programátorom Dávidom Grudlom. Nette patrí medzi frameworky založené na MVC modele. Podporuje PHP verziu 5.3.1 a vyššie. Nette je zameraný na vývoj webových aplikácií, medzi jeho hlavné výhody patrí napríklad: ladenie, eliminácia bezpečnostných rizík, smerovanie (routing) a mnohé ďalšie.

Keďže bol Nette vyvinutý v Českej republike za výhodu môžme považovať aj komunitu. Je viac ako zrejmé, že najviac nadšencov a programátor, ktorí využívajú tento framework sa nájde práve tu.

3.3 Webové technológie

V súčasnosti sa pri tvorbe webu využívajú technológie pre zobrazenie užívateľského rozhrania, ktorými sú HTML a CSS. Práve tieto dve technológie sú spoločne prepojené a použitie oboch je esenciálnou časťou vytvorenia kvalitného grafického užívateľského rozhrania. JavaScript zabezpečené veľmi kvalitné prepojenie už so spomínanými HTML a CSS. Funkcionalitu v pozadí zabezpečuje v našom prípadne PHP programovací jazyk. Spomenuté a ďalšie technológie si rozoberieme v nasledujúcej kapitole podrobnejšie.

3.3.1 HTML a CSS

HTML ako *Hypertext Markup Language*[12] a CSS ako *Cascading style sheets*[11], alebo po našom kaskádové štýly sú dve technológie, ktoré úzko súvisia jedna s druhou a sú základnými technológiami pri tvorbe webových stránok. Obe vznikli v priebehu 90-tých rokov. Pretože úlohou tejto práce je vytvoriť webový informačný systém, obe technológie sú jadrom práce.

HTML má rieši hlavne prezentáciu informácií napríklad: zobrazenie tabuliek, alebo odsekov. CSS zas rieši grafické zobrazenie, ako napríklad: zobrazenie inej farby pozadia, alebo písma a podobne.

Za rozšírenie týchto technológií môžeme považovať Bootstrap, o ktorom je viac popísané v kapitole 3.3.4.

3.3.2 JS

JS[13], alebo celým názvom JavaScript patrí medzi moderné technológie súvisiace s tvorbou webových stránok. Prvý krát sa objavil a následne sa vyvíjal v druhej polovici 90-tých rokov. V súčasnosti je veľmi rozšírený pre jeho flexibilitu a to hlavne preto, lebo zabezpečuje interaktívne funkcie grafického užívateľského rozhrania. Môže sa jednať napríklad o tlačidlá, či formuláre, ale takisto aj mnohé ďalšie prvky grafického rozhrania, ktorá vďaka JavaScriptu dokážu reagovať na zmenu vykonanú užívateľom a týmto spôsobom meniť celkové rozhranie, v ktorým užívateľ pracuje, napríklad v podobe zobrazenia ďalších informácií.

JavaScriptová činnosť je vykonávaná na strane klienta po načítaní stránky webovým prehliadačom a server je z tejto činnosti vynechaný. Taktiež nedisponuje funkciami, ktoré dokážu pracovať so súbormi a tým je ochránená bezpečnosť užívateľov.

3.3.3 PHP

PHP[7], alebo inak *Hypertext Preprocessor* je skriptovací jazyk, ktorý sa často používa na tvorbu webových aplikácií. Aplikácie tvorené pomocou PHP jazyka sú často vytvárané pomocou frameworku, tak ako je to aj v našom prípade. Celá logika programu a všetky logické funkcie sú naprogramované pomocou PHP jazyka.

O PHP môžeme povedať, že je serverovo orientovaný programovací jazyk, tým pádom komunikácia prebieha medzi serverom a klientom, ktorý je v našom prípade užívateľ. Tento prístup je výhodný z dôvodu, že všetky výpočty a prevod programovacieho kódu prebieha na serveri a užívateľ dostane už iba výslednú formu, ktorú mu prehliadač zobrazí.

Medzi najzaujímavejšie funkcie PHP jazyka patria metóda GET a POST. Obe metódy posielajú dáta ako súčasť URL. Rozdiel je v tom, že metóda GET posiela dáta tak, že ich zobrazí ako súčasť URL za otáznikom a musíme brať do úvahy aj maximálnu dĺžku reťazca, ktorá je napríklad pre Internet Expoloreri 1024 znakov. POST metóda naopak dáta v URL nezobrazí a tým pádom je vhodná pre poslanie informácií, napríklad osobné informácie z vyplnených formulárov. POST metóda je najvhodnejšia pri posielaní skrytých údajov, napríklad hesla.

3.3.4 Bootstrap

Bootstrap[1] vznikol v roku 2011 ako voľne dostupný framework na tvorbu front-endu. Obsahuje HTML a CSS podporu pri tvorbe foriem, tlačidiel, navigačných panelov a mnohých ďalších objektov. Bootstrap poskytuje už vytvorené súbory s kaskádovými štýlmi, ktoré sú

použité primárne pri tvorbe webovej responzivity a programátor používa už pred-pripravené triedy pre upravovanie daných elementov HTML kódu.

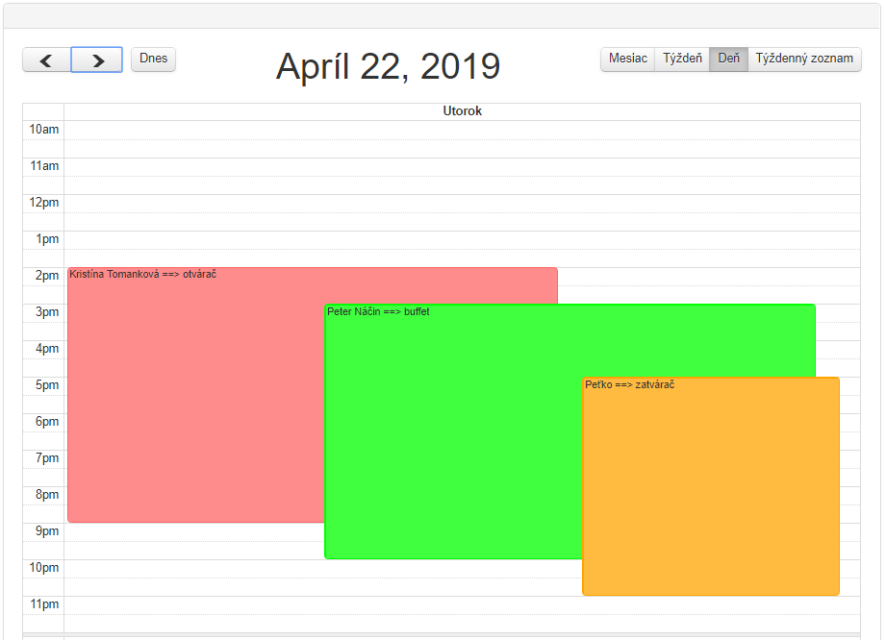
Základnou myšlienkou Bootstrapu je rozdelenie okna do 12 stĺpcov, s ktorými sa následne pracuje. Toto rozdelenie podporuje veľké, stredné, ale takisto aj malé rozlíšenia na monitoroch, počítačoch či mobilných zariadeniach. Klasické stolové počítače používajú médium, čiže stredné rozdelenie do stĺpcov.

3.3.5 FullCalendar

Pre prehľadné uchovanie a zobrazenie pracovných smien zamestnancov je najvhodnejšie využiť formát kalendára. Túto možnosť poskytuje skript FullCalendar, ktorý pracuje ako JavaScript [2]. FullCalendar slúži ako kalendár udalostí, ktorý sa v našom prípade dá využiť pre zobrazovanie každodenných pracovných smien. Skript pre jeho správnu funkčnosť požaduje časový interval a taktiež názov udalosti, pomocou týchto informácií dokáže pracovnú smenu zobraziť v kalendári.

FullCalendar umožňuje užívateľovi rôzne možnosti prehľadávania kalendára a to napr. zobrazenie vo forme dní, týždňov či mesiacov, kde sú uvedené pracovné smeny. Štandardne je kalendár zobrazený vo forme mesiacov, kde sú ku každému dňu priradené smeny, ktoré do tohoto dňa patria. Týmto spôsobom kalendár dokáže uchovávať aj históriu pracovných smien, ak sú informácie udržiavané v databáze.

Pri výbere zobrazenia na konkrétny deň sa pracovné smeny zoradia podľa počiatočného času. Príklad zobrazenia FullCalendar pre deň je možné vidieť na obrázku 3.2.



Obrázek 3.2: Zobrazenie dňa pomocou FullCallendar

3.4 Databáza

Poslednou využitou technológiou pri tvorbe informačného systému je databáza. Pre potreby riešenia mojej práce som zvolil MySQL databázu.

3.4.1 MySQL

MySQL databáza využíva SQL jazyk. Je to multiplatformový relačný databázový systém, ktorého silné stránky tvorí hlavne rýchly a jednoduchý prístup k dátam uloženým v databáze. Význam relačný v tomto prípade znamená, že dáta sú roztriedené do konkrétnych tabuliek, ktoré medzi sebou majú relácie, alebo inak povedané vzťahy.

Začiatky SQL jazyka sa spájajú s firmou IBM, ktorá začiatkom 70-tých rokov vytvorila projekt, ktorého hlavným cieľom bolo potvrdiť životaschopnosť relačného modelu s myšlienkou získať nové skúsenosti pri implementácii toho modelu databázy. Začiatkom 80-tých rokov IBM ohlásila prvú verziu nazvanú "SQL/data System", ktorá sa ešte vyvíjala niekoľko rokov. Túto éru môžme považovať za čas, v ktorom vznikol SQL jazyk. Informácie boli získané z knihy [9].

3.5 Vývoj a ladenie

Vytváranie a testovanie informačného systému bolo zabezpečené pomocou nástroja XAMPP, ktorý umožňuje vytvoriť lokálny webový server s nastavením WAMP. Skratka WAMP konkrétne odpovedá štvorici písmen:

- **W - Windows** - operačný systém Windows,
- **A - Apache** - Apache HTTP server,
- **M - MySQL** - databázový systém MySQL,
- **P - PHP** - skriptovací jazyk PHP.

Pre operačný systém Linux sa používa skratka LAMP. Konkrétny použitý nástroj pre prístup do databázy je PhpMyAdmin, ktorý podporuje jazyk PHP a tým pádom je podporovaný použitým frameworkom, ktorý sme si popísali v kapitole 3.1.1.

Kapitola 4

Metodológia a návrh

V poradí štvrtá kapitola je zameraná na popis vybraných metód pri tvorbe informačného systému a popis návrhu databázy. V prvom rade si objasníme metodológiu použitú pri písaní práce v podkapitole 4.1. Sú tu popísané jednotlivé časti návrhu ako **Use Case Diagram** v podkapitole 4.2 a taktiež **ERD** *Entity Relationship Diagram* v podkapitole 4.3. Posledná časť návrh, je popísaná z teoretického poňatia databáz 4.4. V podkapitole 4.4.1 je popísaná relačná databáza, ktorú sme sa v práci rozhodli použiť.

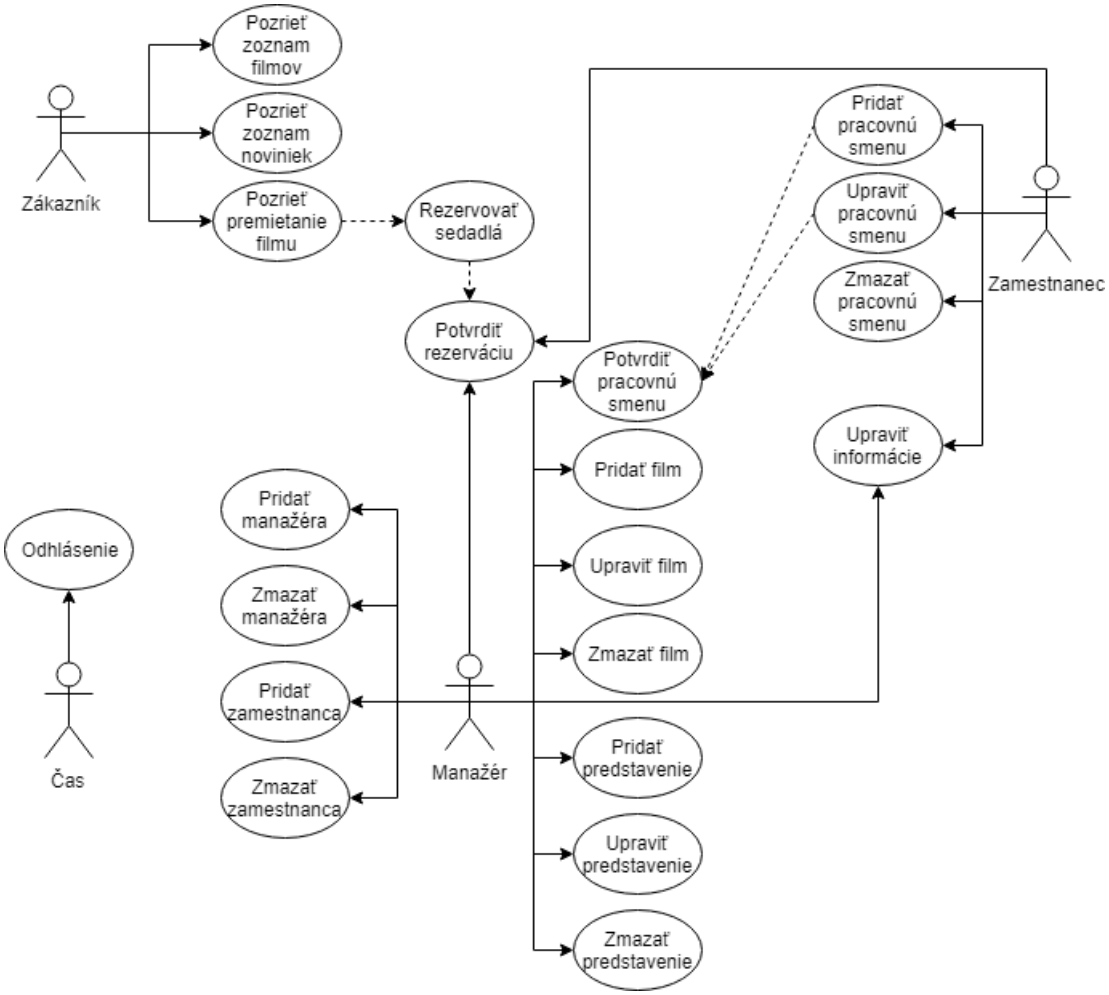
4.1 Metodológia

Táto podkapitola popisuje logický postup pri vytváraní informačného systému a popisuje jednotlivé fázy postupu.

- **Špecifikácia požiadaviek** - zistenie špecifických požiadaviek od zadávateľa (kina),
- **Analýza požiadaviek** - definovať si ciele a metódy, ako postupovať pri vytváraní práce a ako by mal vyzerať výsledok práce. Výsledok analýzy je popísaný v časti 2.2
- **Vytvorenie návrhu** - spracovanie zistených požiadaviek a následné vytvorenie diagramov, ktoré zobrazujú role užívateľov spojené s ich úlohami a dáta použité v systéme. Podrobný popis a diagramy sú v častiach 4.2 a 4.3
- **Implementácia systému** - hlavnou častou bolo implementovanie celého systému v súlade s vytvoreným návrhom
- **Testovanie a ladenie** - dôležitou etapou bolo testovanie a odladenie celého informačného systému, jeho častí, ale aj celku.
- **Zhodnotenie** - na záver prebehlo zhodnotenie celej práce a zhodnotenie splnenia počiatočných cieľov.

4.2 UseCase

Prvou časťou návrhu sa zaoberá Use Case diagram, alebo inak povedané Diagram prípadov použitia, ktorý je zobrazený pomocou UML jazyka. Tento diagram môžeme vidieť na obrázku 4.1, z ktorého zistíme, že sú tu vykreslené základné role užívateľov a s nimi spojené úlohy, ktoré im systém umožňuje vykonávať. Use Case diagram sa rozdeľuje na dve hlavné časti:



Obrázek 4.1: Use Case Diagram

- **prípad použitia** - akcie, alebo procesy, ktoré vykonávajú účastníci,
- **účastník** - role užívateľov, ktorí vstupujú do vzťahu s procesmi.

V diagrame prípadov použitia vidíme, že kino zaznamenáva 3 základné role užívateľov, ktorí pristupujú k informačnému systému a tými sú:

- **Zákazník**
- **Zamestnanec**
- **Manažér**

Postupne popíšeme jednotlivých účastníkov informačného systému, tak ako sú zobrazení v diagrame prípadov použitia a popíšeme ich možnosti interakcie so systémom.

Zákazník

Zákaznícka rola sa jediná v systéme pohybuje na úrovni bez prihlásenie sa užívateľa do systému. Poskytnuté sú nasledujúce úlohy, kde prvou je prezeranie filmov, ktoré kino

premieta a následne možnosť si rezervovať sedadlo na daný film pomocou informačného systému. Tento proces je vykonaný vo forme vybratia vyplnenia informácií, vybratia si sedadla a potvrdenia rezervácie. Druhou úlohou je možnosť pozerať v zozname noviniek, t.j. filmov, ktoré sa zatiaľ len plánujú premietať. Poslednou úlohou, ktorú rola zákazník má je prezerať zoznam filmov.

Zamestnanec

Druhou rolou sú zamestnanci, ktorí majú umožnené sa do systému prihlásiť a to za účelom nahlasovania pracovných smien. Poskytnutý im je aj celkový pohľad na smeny ostatných pracovníkov a taktiež majú možnosť základnej editácie dát, ale len v ich profile.

Manažér

Ako poslednej role a to manažérskej sú poskytnuté najväčšie právomoci a taktiež vlastná úroveň prístupu. Keďže manažér kina ako najvyššie postavený pracovník, ktorý má za úlohu viesť ostatných a takisto viesť celkovú administráciu musí mať prístup k najväčšej škále dát. Najvýznamnejším privilégiom je potvrdzovanie a celková možnosť upravovať pracovné smeny zamestnancom. Taktiež je manažérom poskytnutá možnosť vytvoriť účty novým pracovníkom. Dôležitou úlohou je aj správa filmov, rezervácií a zamestnaneckých účtov.

4.3 ERD

Druhou častou návrhu je Entity Relationship diagram, teda entitne vzťahový diagram, ktorý môžeme vidieť na obrázku 4.2. ER diagramy sa používajú pre abstraktné a konceptuálne (pojmové) zobrazenie dát. Jednoducho povedané ER diagram zobrazuje všetky dáta, ktoré sú v systéme používané a jednotlivé prepojenia medzi nimi. Tak ako aj Use Case diagram obsahuje dve základné časti, ktorými sú:

- **entity** - sú to objekty reálneho sveta, o ktorých chceme uchovať dáta v databáze,
- **vzťahy** - reprezentujú jednotlivé prepojenie medzi entitami.

Každá entita obsahuje kandidátne kľúče, ktorým odpovedajú jednotlivé informácie o entite. V prípade použitia výrazu tabuľky, atribút odpovedá riadku reprezentujúcemu dáta v danej tabuľke. Pre lepšie pochopenie je ale vhodné si vysvetliť čo pojem kandidátny kľúč znamená [14].

Kandidátny kľúč je atribút relácie, ak spĺňa dve podmienky, ktorými sú unikátnosť a minimálnosť atribútu. Unikátnosť rozumieme ako jednoznačnosť daného atribútu, teda stav, kedy neexistujú dve n-tice relácie s rovnakou hodnotou tohoto atribútu. Pod minimálnosťou rozumieme stav, že v prípade zloženého kandidátneho kľúču, nie je možné vypustiť žiadnu zložku tak, aby prestala byť splnená unikátnosť.

Špecifickou vlastnosťou každej entity je, že musí obsahovať tzv. primárny kľúč [14]. Primárny kľúč je jeden z kandidátnych kľúčov(nami vybraný), ostatné kandidátne kľúče sa nazývajú alternatívne, alebo sekundárne kľúče.

Najčastejšie používaným primárnym kľúčom je číselný identifikátor, ktorý je unikátny pre každú entitu. Tento identifikátor musí byť vždy pri vytvorení nového záznamu automaticky inkrementovaný, aby nedošlo k zhode a záznamy boli odlíšitelné. Pre bližšie objasnenie je vhodné si uviesť príklad. Ak by existovala tabuľka s názvom **človek** obsahujúca iba meno

a priezvisko, je potrebné aby táto tabuľka obsahovala identifikátor ako atribút pre rozlíšenie totožných záznamov. Týmto spôsobom by sme predišli zhode v prípade pridania dvoch totožných záznamov s krstným menom *Jožko* a priezviskom *Mrkvička*, pretože prvý by mal záznam s identifikátorom 1 a druhý s identifikátorom 2. Pomocou unikátneho atribútu by boli obidva záznamy odlíšitelné. Rovnakým princípom fungujú školské loginy, kde pre každého ďalšieho študenta je inkrementované číslo v logine v prípade zhody priezviska.

Navrhovaný ER diagram obsahuje 6 základných entít, s ktorými systém pracuje. V tejto časti si prejdeme jednu po druhej, aby sme si lepšie priblížili ich význam.

Zamestnanec

Prvou entitou v návrhu je entita zamestnanca. Pomocou obrázku 4.2 vidíme, že primárnym kľúčom zabezpečujúcim unikátnosť daného záznamu je číselný identifikátor zamestnanca. Entita obsahuje viacero základných atribútov ako meno, priezvisko, emailovú adresu, mobilné číslo a práva. V našom prípade ma atribút práv význam pre určenie či sa jedná o bežného zamestnanca, alebo o manažéra. Atribúty mobilné číslo aj emailová adresa slúžia ako kandidátne kľúče, ak predpokladáme, že neexistujú dvaja zamestnanci, ktoré používajú rovnaké mobilné číslo, alebo emailovú adresu.

Pracovná smena

Keďže sme sa rozhodli použiť technológiu FullCalendar, ktorá bola opísaná v kapitole 3.3.5, štruktúra entity pracovnej smeny má nasledovný formát.

Aby skript FullCalendar dokázal čítať dáta, je potrebné, aby dáta obsahovali:

- Názov. Tento sa v našom prípade skladá z mena zamestnanca a pozície, na ktorú sa hlási, prípadne je nahlásený.
- Ďalej je potrebné uchovať dátum, počiatočný a koncový čas pre vhodné zobrazenie pracovnej smeny v konkrétnom dni kalendára a zobrazenie časového intervalu pri možnosti rozkliknutia konkrétneho dňa. Dátum používa dátový typ *date* a čas dátový typ *time*.
- Ďalším je booleanovský atribút s názvom *potvrdená*. Tento atribút dátový typ *boolen* môže mať hodnoty 0, alebo 1. Reprezentuje rozdiel medzi nepotvrdenou pracovnou smenou, na ktorú sa zamestnanec sám hlási a potvrdenou smenou, ktorú potvrdil manažér zamestnancovi. Potvrdená smena je vždy chápaná ako pracovný záväzok.
- Ako posledný atribút je štandardne číselný identifikátor, ktorý je aj primárny kľúčom danej entity.

Film

Filmová entita je s entitou sedadla najrozsiahlejšia čo sa týka počtu atribútov. Po primárnom kľúči obsahuje hlavne atribúty s dátovým typom *varchar*, ktoré slúžia ako doplnkové informácie pre zákazníkov pri prehliadaní premietaných predstavení. Do tejto kategórie patrí: názov a žáner filmu, dĺžka trvania, jazyk, v ktorom je film premietaný a doplnkové informácie vo forme popisu. Taktiež sem patrí položka image, ktorá reprezentuje cestu k

obrázku, ktorý má byť zobrazený pri danom filme. Posledným atribútom je aktívnosť dátový typ *integer*, ktorá reprezentuje momentálnu premietanosť filmu a môže sa vyskytovať v jednej z troch hodnôt.

Hodnota 0 reprezentuje neaktívnosť filmu, inak povedané, že sú dáta o filme uchované v databáze, ale film sa nepremieta.

Ďalšou možnosťou je hodnota 1, ktorá odpovedá štandardnému filmu, ktorý je momentálne premietaný v kine.

Atribútová hodnota 2 priradí film medzi novinky, ktorým nie je možné priradiť premietanie, ale iba jednorázový dátum premiéry a to z dôvodu, že film je iba reklamovaný a kino sa v danom čase nezaoberá administráciou a premietaním tohoto filmu.

Premietanie

Tabuľka premietania slúži pre opakované premietanie daného filmu. Štandardne sa tu vyskytuje primárny kľúč vo forme identifikátora pre rozlíšenie daného premietania. Každé premietanie obsahuje dátum a čas, aby bolo možné zobraziť viacej premietaní jedného konkrétneho filmu.

Zaujímavým atribútom je atribút *film_id*, ktorý reprezentuje prepojenie dvoch tabuliek, kedy jeden film môže mať viacero premietaní v rôzny deň a čas bez potreby vytvárať jednotlivé záznamy v tabuľke filmu. Týmto spôsobom dokážeme priradiť unikátny identifikátor filmu každému jeho predstaveniu a tak získať informácie nie len o danom predstavení, ale pomocou spojenia tabuliek aj o danom filme. Tento atribút môžeme inými slovami popísať ako cudzí kľúč. Čo pojem cudzí kľúč znamená je popísané nižšie v kapitole 4.3.

Sedadlo

Entita sedadla je úzko prepojená s predchádzajúcou entitou premietania. Primárnym kľúčom je znova číselný auto-inkrementujúci sa identifikátor. Vytvorenie sedadla vznikne po pridaní konkrétneho premietania filmu, kde kľúčovým faktorom je aj výber sály pri vytváraní premietania filmu. Počet sedadiel odpovedá veľkosti danej sály. Keďže o sále nepotrebujeme uchovávať žiadne ďalšie informácie a jediným atribútom by bol názov, tak sme sa rozhodli pridať tento jeden atribút do entity sedadla. Každé sedadlo obsahuje atribút riadku a stĺpca, ktoré dátovým typom *integer* reprezentujú umiestnenie sedadla v danej sále pri premietaní filmu. Dôležitou častou je atribút a zároveň cudzí kľúč *premietanie_id*, ktorý slúži ako priradenie sedadla k existujúcemu premietaniu. V entite sedadlo sa nachádza aj atribút obsadené dátový typ *integer*, ktorý pri hodnote 0 reprezentuje prázdne sedadlo, pri hodnote 1 rezervované sedadlo a poslednou možnosťou je hodnota 2, ktorá zodpovedá už zaplatenému sedadlu. Množinu atribútov uzatvára *rezervácia_id*, ktorý poskytuje spojenie entity sedadla a entity rezervácia, kvôli možnosti získania informácií, kto dané sedadlo rezervoval.

Rezervácia

Záverečnou entitou je rezervácia, ktorá vzniká až po vykonaní rezervácie daného sedadla a je prepojená s entitou sedadla pomocou cudzieho kľúču. Dôvodom tohoto spojenia je možnosť spätne zistiť, ktoré sedadlo je kým rezervované. Nachádzajú sa tu atribúty ako emailová adresa, dátový typ *varchar*, ktorá slúži na overenie platnosti a vyhľadanie danej rezervácie v systéme. Ďalej je to atribút zaplatenia, dátový typ *boolen*, ktorý reprezentuje,

či už bola rezervácia fyzicky zaplatená na pokladni v kine. Môže mať len hodnoty 0, alebo 1. V prípade zaplatenia, alebo prepadnutia rezervácie nie je nutné záznam úplne zmazať. Ako náhrada bol vytvorený atribút expirácie, dátový typ *boolen*, ktorý má na starosti rozlíšiť, ktoré rezervácie prepadli a naopak. Využitie má hlavne v prípade vytvárania štatistiky, ale aj pre vyhľadanie emailových adries zákazníkov, ktorí majú tendenciu obsadiť miesto ale veľmi často ho nechajú aj prepadnúť. Posledný atribútom je časová značka vytvorenia rezervácia zákazníkom, ktorá má dátový typ *datetime* a je vhodná pre rýchlejšie vyhľadanie konkrétnej rezervácie v systéme.

Pojem cudzí kľúč

Pojem cudzí kľúč sme už niekoľko krát spomenuli a je vhodné si ho objasniť a vysvetliť, čo v skutočnosti znamená [14].

Cudzí kľúč relácie R2, je platný, ak spĺňuje tieto dve vlastnosti. Každá hodnota je buď zadaná, alebo nezadaná. Musí existovať relácie R1, s kandidátnym kľúčom takým, že zadaná hodnota cudzieho kľúča je zhodná s hodnotou kandidátneho kľúča nejakej n-tice relácie R1.

Štandardne cudzie kľúče spájajú tabuľky medzi sebou pomocou zhodnej hodnoty atribútov a taktiež sú cudzie kľúče dôležitou súčasťou našej práce. Popíšeme si význam a dôvody jednotlivých cudzích kľúčov medzi tabuľkami ER diagramu. Práca obsahuje celkovo 3 cudzie kľúče.

Premietanie -> Film

Prvý cudzí kľúč sa vyskytuje v tabuľke premietanie a zabezpečuje spojenie medzi premietaním a filmom. Toto spojenie poskytuje jednoznačné určenie, ktoré predstavenie patrí ktorému filmu. Toto spojenie je prezentované v podobe atribútu v tabuľke predstavenia, ktorý obsahuje rovnakú hodnotu ako identifikačné číslo filmu. Hlavným dôvodom vytvorenia vlastnej tabuľky premietanie s cudzím kľúčom na tabuľku filmu je zjednodušenie dát v prípadne opakovaného premietania filmu a vyhnutie sa vzťahu 1:1, kde by bolo potrebné vytvárať nový záznam filmu v prípade opakovaného premietania.

Sedadlo -> Premietanie

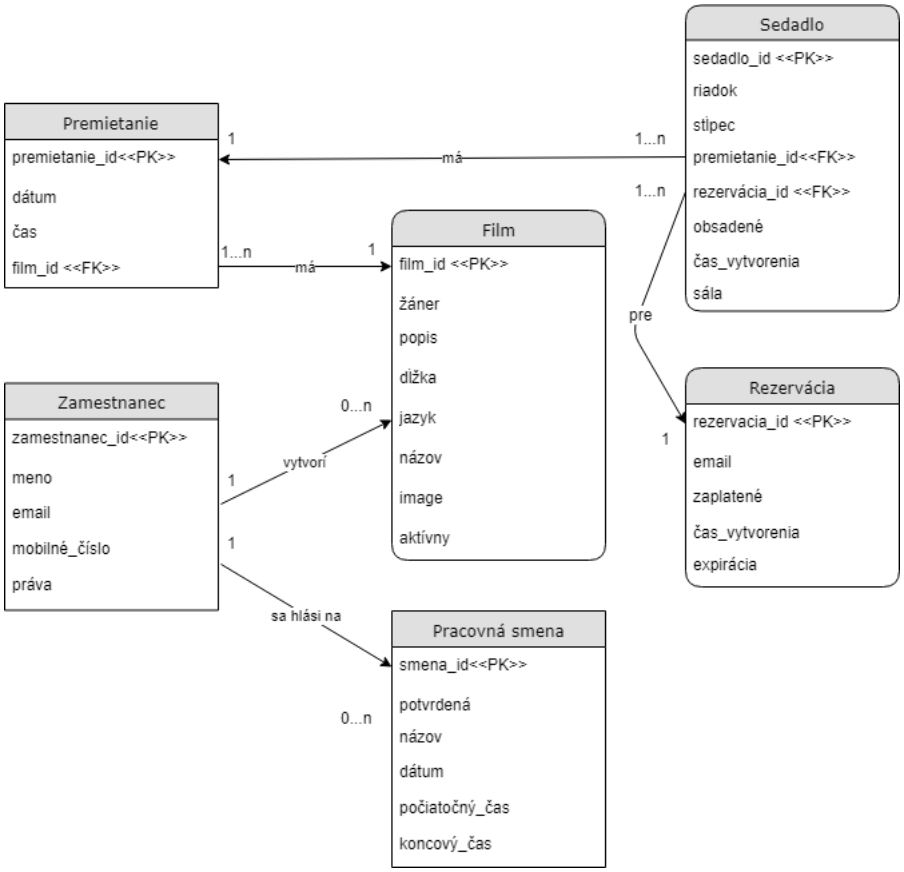
Tabuľka sedadla obsahuje dva cudzie kľúče, kde jeden z nich odkazuje na tabuľku premietanie. Významom tohto kľúča je prepojenie sedadla s premietaním a týmto spôsobom zabezpečenie celkovej štruktúry od filmu cez premietanie až na jednotlivé sedadlo.

Pre lepšie porozumenie si môžeme uviesť príklad. Predstavme si, že sú nám známe iba dáta o jednom konkrétnom sedadle s identifikačným číslom 12345 a máme za úlohu zistiť, na premietaní ktorého filmu sa toto sedadlo nachádza. Pozrieme sa na dáta tohto konkrétneho sedadla a zistíme, že obsahuje rôzne dáta v podobe umiestenia rady a stĺpca, taktiež zistíme sálu a či je to sedadlo obsadené. Pre nás najdôležitejšou informáciou, ktorá nás posunie ďalej je atribút *premietanie_id* s hodnotou 55, ktorý ako cudzí kľúč poslúži na prechod do tabuľky premietania. V databáze sa nájde záznam premietania s identifikátorom 55, ktorý zodpovedá dátumu a času, ale v daný moment ešte neznámemu filmu. Po použití atribútu *film_id*, kde hodnota atribútu zodpovedá hodnote 1 zistíme, že sa jedná o prvý vytvorený film v databáze, ktorého záznam v databáze poskytuje aj doplnkové informácie vo forme popisu, žánru, atď. Týmto spôsobom získavame informácie o celej štruktúre, ktorá

je prepojená pomocou cudzích kľúčov.

Sedadlo -> Rezervácia

Druhým cudzím kľúčom tabuľky sedadlo je cudzí kľúč, ktorý spája tabuľku sedadla a rezervácie. Slúži primárne rezervačnému systému sedadiel ako výstup pre pokladňu vo forme overenia platnosti rezervácií. Atribút *rezervácia_id* smeruje do tabuľky rezervácie, ktorá poskytuje informácie o zákazníkovi, ktorý rezerváciu vykonal. Taktiež je možné opačné dohľadanie informácii a to napríklad v podobe zistenia emailovej adresy rezervácie, kde získame unikátne identifikačné číslo. Následne pomocou identifikačného čísla rezervácie dokážeme cez cudzí kľúč získať všetky sedadlá, ktoré do rezervácie patria.



Obrázek 4.2: Entity Relationship Diagram

4.4 Model databázy

V tejto podkapitole si vysvetlíme a priblížime model databázy. Popísaný je konkrétne model relačnej databázy, ktorý je použitý v práci. Informácie k danej kapitole boli čerpané z knihy [8] a taktiež bola použití kniha [10].

4.4.1 Relačná databáza

Relačný model databázy vznikol v 70-tých rokoch minulého storočia. V dnešnej dobe sa považuje za najrozšírenejší model používaný pri tvorbe a správe databáz. Zakladateľom relačnej databázy je Dr. Edgar F. Codd, ktorý pracoval ako výskumník pre firmu IBM. Model je založený na dvoch matematických disciplínach - predikátovej logike a teórii množín.

Relačný model prezentuje a ukladá dáta vo vzťahoch, ktoré môžeme inak nazvať aj relácie, alebo tabuľky. Každý vzťah sa skladá z usporiadaných n-tíc, alebo záznamov a taktiež obsahuje atribúty, alebo polia. Poradie záznamov nehrá v databáze žiadne rolu, pretože každý záznam obsahuje unikátnu hodnotu, inak povedané primárny kľúč, ktorý bol už opísaný v kapitole 4.3. Relačná databáza umožňuje užívateľovi nepoznať fyzické umiestnenie dát, pokiaľ chce s dátami pracovať. Táto vlastnosť relačnej databázy je kľúčová.

Mimo vzťahov 1:1 a 1:N, relačná databáza umožnila používanie vzťahu M:N. Vzťah medzi tabuľkami je zabezpečený pomocou totožných hodnôt v zdielaných poliach. Tieto vzťahy môžme pomenovať aj pomocou pojmu cudzí kľúč, ktorý bol taktiež vysvetlený v kapitole 4.3. Ak má užívateľ znalosť o celkovej štruktúre konkrétnej databázy, ktorá je založená na relačnom modele môže k dátam pristupovať rôznymi spôsobmi.

Získavania dát prebieha pomocou jazyka SQL(structured query language, čiže štrukturovaný dopytovací jazyk). Tento jazyk zodpovedá štandardnému jazyku, ktorý zabezpečuje celkovú prácu s databázou vo forme vytvorenia následnej modifikácie a získavania dát.

Pre lepšie priblíženie ako SQL jazyk pracuje si uvedieme jednoduchý príklad, ako by mohol SQL dopyt vyzerať.

```
SELECT meno, priezvisko
FROM clovek
WHERE vek="25"
```

Základné tri časti SQL dopytu sú zobrazené na príklade vyššie. Prvou časťou je SELECT, ktorý má na starosti vybrať dáta, ktoré hľadáme v konkrétnej tabuľke. Klauzula FROM poukazuje na tabuľku, z ktorej majú byť dáta vybraté. Poslednou častou dopytu je WHERE klauzula, ktorá určuje podmienku a udáva, ktorý záznam z databázy vybrať. Častou používanou možnosťou je aj ORDER BY, pomocou ktorej je možné vybrané dáta zobraziť v zostupnom, alebo vzostupnom poradí.

Ak predpokladáme, že existuje tabuľka s názvom clovek, ktorá obsahuje atribúty meno, priezvisko a vek. SQL dopyt vyberie z databázy informácie vo forme mena a priezvisko zo záznamov, kde vek obsahuje hodnotu 25.

Relačná databáza poskytuje niekoľko výhod oproti prechádzajúcim štruktúram databáz, medzi tieto výhody patrí:

- **zabudovaná viacúrovňová integrita** - na úrovni tabuliek overuje neduplicitu záznamov a vyhľadáva chýbajúce hodnoty primárneho kľúča a na úrovni vzťahov zabezpečuje platnosť vzťahov medzi tabuľkami,
- **logická a fyzická nezávislosť dát na databázovej aplikácii** - zmeny logického návrhu a taktiež zmena databázového softwaru neovplyvnia aplikáciu,
- **garantovaná konzistencia a presnosť dát** - dáta sú presné a je garantovaná ich intergita,

- **jednoduché získavanie dát** - pomocou SQL dopytov je možné získať dáta zo spojených tabuliek mnohými spôsobmi.

Nevýhodou relačnej databázy v minulosti bolo hlavne obmedzenie úložného priestoru. Tento problém sa ale vyriešil začiatkom 90-tých rokov, kedy prebehol veľký pokrok v hardwarovej aj softwarovej oblasti a množstvo úložného priestoru sa rapídne zväčšoval.

Správu databáz zabezpečuje tzv. SRBD, celým názvom Systém riadenia bázy dát(RDBMS - relational database management system) je program, ktorý sa stará o celkovú správu databázy vo forme vytvorenia a modifikovania. Kvalita tohto systému odpovedá zhode s relačným modelom. Pre umožniteľný prístup pre viacerých užívateľov bol behom 80-tých a 90-tých rokov vyvíjaný SRBD klient/server model. Tento typ umožňuje dáta uložiť na stroji, ktorý považujeme ako databázový server. Naopak užívateľov, ktorí s dátami pracujú a pristupujú na databázový server môžeme nazvať databázoví klienti.

Kapitola 5

Implementácia

Kapitola 5 opisuje priebeh celej implementácie, kde sú vybrané najhlavnejšie a najzaujímavejšie časti. Prvá podkapitola 5.1 sa sústredí na opis adresárovej štruktúry, ktorý ponúka framework Laravel. Nasledujú hlavné časti implementácie, ktoré sú rozdelené na 8 častí: Zobrazenie filmov 5.2.1, Rezervácia sedadiel 5.2.2, QR kód 5.2.3, Overenie QR kódu 5.2.4, Užívateľské úrovne 5.2.5, Nahlasovanie na pracovnú smenu 5.2.6, Administrácia 5.2.7 a Databáza, práca s údajmi 5.2.8. Na konci kapitoly si podrobnejšie rozoberieme aj užívateľské rozhranie.

5.1 Adresárová štruktúra

Laravel framework ponúka už pred-pripravenú adresárovú štruktúru pri vytvorení projektu. Štruktúra je navrhnutá čo najprehľadnejšie a to tak, že jednotlivé typy súborov sú uložené v spoločnom adresári pre jednoduchý prístup programátora.

Koreňový adresár obsahuje niekoľko zložiek viď. obrázok 5.1, ale aj základné konfiguračné súbory, ktorými sú napríklad .env a composer.json.

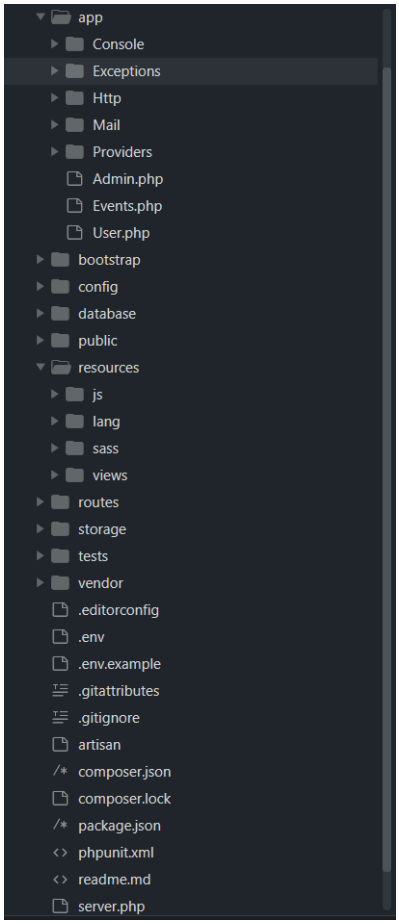
app

Zložka app obsahuje súbory so zdrojovými kódmi. Štruktúra Laravelu rozdeľuje túto zložku ďalej, kde jednou z častí našej práce je zložka **Console**, ktorá ponúka uloženie vlastne implementovaných príkazov, ktoré môžu byť po implementácii použité pomocou konzoly. Nasleduje zložka **Http**, ktorá obsahuje všetky controllery a tým pádom obsahuje najväčšiu časť zdrojových kódov. V našom prípade je logicky rozdelená do troch častí a to:

- **Admin** - obsahuje všetky funkcie pre manažérsku úroveň pripojenia,
- **Auth** - nachádzajú sa tu všetky controllery, ktoré pracujú s prihlasovaním, napr. prihlásenie, registrácia, overenie hesla a ďalšie,
- **Web** - v posledná časť obsahuje controllery s užívateľskými funkciami napr. zobrazenie sály premietania, rezervovanie sedadiel a ďalšie.

Dôležité je spomenúť aj zložku **Middleware**, ktorá sa stará o zabezpečenie úrovní prihlasovania a znemožnenia prístupu na stránky, na ktoré daný užívateľ nemá prístup.

Z dôvodu logickej štruktúry zložka app, obsahuje zložku **Mail**, v ktorej sa nachádza súbor pre prácu s mailom a príprava pre jeho nasledujúce odoslanie na zákaznícku emailovú



Obrázek 5.1: Adresárová štruktúra Laravelu

adresu v prípade úspešnej rezervácie sedadla na premietaný film.

config

Súbory uložené v zložke config sa starajú o celkovú konfiguráciu programu. Pre zaujímavosť vyberiem a priblížim niektoré z nich.

- **app.php** - sa skladá zo základnej konfigurácie napr. ako názov projektu, jazyk, časovú zónu, URL pre localhost atď.
- **database.php** - zahŕňa základné údaje pre nastavenia a pripojenie databázy,
- **mail.php** - pozostáva z nastavení pre posielanie mailu,
- **session.php** - obsahuje informácie o spojení, napr. dĺžku životnosti spojenia.

Takisto sa tu nachádzajú napríklad súbory zabezpečujúce: uchovanie histórie prihlasovania, hashovanie a mnohé ďalšie.

database

V zložke database je hlavnou častou zložka **migrations**, ktorá má v sebe tzv. migrácie. Migrácia slúži na vytvorenie, prípadné zmazanie či upravenie celých tabuliek, ale aj pridanie či odobratie konkrétnych stĺpcov tabuliek z databázy. Pred migrovaním súborov pomocou príkazu *make:migration*, je potrebné nastaviť každej vytváranej tabuľke aj dátový typ, kde medzi najbežnejšie patrí string, integer, boolean, date či time.

public

Zložka public ukrýva v sebe súbory, ktoré primárne súvisia s grafickým rozhraním. Hovoríme tu o css súboroch, ktoré pracujú s view súbormi a starajú sa o grafické zobrazenie. Tiež tu patrí zložka images, v ktorej sa nachádzajú obrázky použité v grafickom rozhraní. Je potrebné spomenúť aj zložku js, do ktorej sa ukladajú javascriptové súbory.

Najzaujímavejšou častou je ale zložka plugins, ktorá môže obsahovať rôzne pluginy, o ktoré je práca rozšírená. Práca obsahuje jeden použitý plugin a to je FullCalendar viď. 3.3.5. Všetky súbory súvisiace s týmto pluginom sú uložené práve tu.

resources

Nachádza sa tu niekoľko zložiek, napríklad zložka lang, ktorá má nastarosti jazyk vo frameworku. Štandardne je to angličtina, ale je možné si doplnkovo stiahnuť aj ďalšie jazyky a vyvíjať programy aj v nich.

Najpodstatnejšou častou Resourse zložky sú view súbory. Keďže framework Laravel pracuje na modele MVC 3.1.1, view súbory sú jadrom programu. Laravel view súbory majú špeciálnu koncovku *súbor.blade.php*. Pri ďalšej práci s týmito súbormi nie je potrebné uvádzať koncovku, ale iba názov súboru bez koncovky.

Aj keď view súbory majú koncovku obsahujúcu php, tak primárnym programovacím jazykom je HTML. PHP jazyk môže byť takisto použitý vo view súboroch, kde framework Laravel podporuje znak @ ako začiatok php funkcií. Jednoduchým príkladom môže byť použitie *if* podmienky.

```
@if(podmienka)
   <h1> Prvá možnosť</h1>
@else
   <h1> Druhá možnosť</h1>
@endif
```

Pre štandardné značenie začiatku PHP kódu v HTML súbore používame *<?php*, kde Laravel poskytuje substitúciu a to vo forme *@php* pre začatie a *@endphp* pre uzavretie PHP kódu.

routes

Medzi najdôležitejší súbor celej adresárovej štruktúry patrí *web.php*. Tento súbor zabezpečuje routovaciu funkciu celého frameworku. Do tohto súboru sa zapisujú všetky url cesty, ktoré sú spojené s jednotlivými controllermi, ktoré zabezpečujú funkcionalitu. Takto je logicky prepojená štruktúra, ktorá je rozdelená medzi view a controller súbory. Pre zau-

jímavosť si ukážeme ako daná funkcia vo web.php súbore vyzerá.

Route::get('/url','NázovControlleru@FunkciaControlleru');

Route popisuje, že sa jedná o routovaciu funkciu. Nasleduje metóda GET alebo POST, názov konkrétneho controlleru, v ktorom sa funkcia nachádza a znak @ spája controller s danou funkciou v ňom. Takýmto spôsobom sa postaráme o to, aby funkcie s podobným zameraním boli štrukturované v jednom controlleri. Napríklad MovieController môže obsahovať všetky funkcie, ktoré sa týkajú spracovania filmov.
Konkrétne použitie môže byť nasledovné:

Route::get('/movies','MovieController@index');

Pri zadaní URL */movies* sa zavolá MovieController. Controller nájde funkciu *index*, ktorá má na starosti vybrať požadované dáta z databázy a zobraziť view súbor *movies.blade.php*, kde pošle dáta na view, aby boli správne zobrazené.

storage

Nachádzajú sa tu súbory z predchádzajúcich spojení so serverom, prihlásenia užívateľov a celková činnosť, ktorá prebehla na serveri.

tests

Slúži pre uskladnenie súborov, ktoré majú testovací charakter.

vendor

Vendor ako najväčšia zložka obsahuje väčšinu súborov, ktoré Laravel framework používa. Nachádzajú sa tu stovky súborov, ktoré používa framework ako knižnice, prípadne iné súbory určené na správne fungovanie frameworku. Programátor len zriedkavo zasahuje do týchto súborov.

5.2 Hlavné časti

Implementačná časť je rozdelená na 2 časti:

- **Filmy, rezervácia lístkov a QR kód** - časť primárne vyhradená pre zákazníkov kina. Tu sa prezentujú informácie o filmoch a je tu poskytnutá možnosť rezervácie sedadiel na konkrétne premietanie filmu.
- **Zamestnanci a pracovné smeny** - naopak druhá časť implementácie je vyhradená len pre zamestnancov a používa view súbory pre zobrazenie pracovných informácií.

Väčšia časť programovania prebehla pomocou PHP jazyka.

5.2.1 Zobrazenie filmov

Väčšiu skupinu implementovaných súborov tejto časti tvoria view súbory a tým pádom je použitý HTML a CSS jazyk, keďže sa jedná o užívateľské rozhranie, hovoríme tu hlavne o

zobrazení filmov či premietacej sály so sedadlami. Občas sa vyskytnú aj prvky PHP, ktoré dopĺňajú HTML a CSS a to hlavne vo forme výberu dát z databázy. Taktiež je použitý v niektorých prípadoch JavaScript, ktorý zabezpečuje interaktívny prístup užívateľa.

Pri prvom zobrazení sa zobrazí web dostupný pre zákazníkov a je dostupný aj širokej verejnosti. Jedná sa konkrétne o zobrazenie dostupných filmových predstavení. Zákazník má možnosť vidieť premietané filmy v daný deň a taktiež má možnosť si vybrať konkrétny deň najbližší týždeň dopredu.

Pri každom premietaní v zozname sú zobrazené informácie o filme a tiež čas. Tento slúži aj na presmerovanie na konkrétne premietanie filmu. Čas daného premietania je kontrolovaný a porovnávaný so skutočným časom. V čase 10 minút pred začiatkom daného premietania sa sála v aplikácii uzamkne a nie je možné sedadlá ďalej rezervovať pomocou online rezervačného systému. Súčasne 10 minút pred začiatkom sa zrušia všetky rezervované sedadlá, ktoré neboli potvrdené a zaplatené fyzicky v kine na pokladni a to z dôvodu, aby nevznikol problém, že by sa našli zákazníci, ktorí si rezervujú sedadlá a do kina neprídu. Týmto spôsobom kino zabezpečí, že čakajúci zákazníci v kine budú mať možnosť si zakúpiť sedadlá, ktoré sa uvoľnia.

Z pohľadu administrácie filmov je poskytnutá zákazníkovi možnosť si pozrieť reklamované filmy, ktoré kino plánuje premietať v budúcnosti a to v sekcii *Novinky*. Najvyššie postavený pracovník konkrétneho kina, v našom prípade je to manažér, má možnosť spravovať filmy a to vo forme pridávania filmu ako takého, ale aj pridávania predstavenia pre každý jeden film. Okrem pridávania filmov existuje aj funkcia na upravovania či zmazanie filmov a predstavení v prípade potreby.

Implementovaná je funkcia pre filmy a predstavenia pomocou ktorej je možné filmy, alebo predstavenia vyhľadať podľa identifikačného čísla, názvu alebo dátumu.

5.2.2 Rezervácia sedadiel

Implementácie časti spojenej s rezerváciou sedadiel obsahuje aj použitie JavaScriptu, spojeného s HTML a CSS.

Po presmerovaní na predstavenie, je zákazníkovi zobrazená konkrétna sála, v ktorej sa film premieta a sú vykreslené všetky sedadlá. Sedadlá sú rozdelené do 4 tried:

- **Voľné sedadlá** - sú zobrazené bielou farbou a reprezentujú sedadlá, ktoré je možné si rezervovať pomocou rezervačného systému,
- **Aktuálne vybraté sedadlá** - v prípade vybratia voľného sedadla je farba z bielej zmenená na zelenú, ktorá reprezentuje aktuálne vybraté sedadlá, o ktoré má zákazník záujem
- **Rezervované sedadlá** - charakterizuje ich červená farba a symbolizujú rezervované sedadlá,
- **Kúpené/Zaplatené sedadlá** - reprezentujú sedadlá, ktoré boli potvrdené a zaplatené na pokladni a sú zobrazené oranžovou farbou,

Pre vykonanie úspešnej rezervácie musí zákazník postupovať nasledovne:

- **Vyplnenie informácií** - Pomocou JavaScriptu je uzamknutý výber sedadiel, pokiaľ zákazník nevyplní emailovú adresu, na ktorú mu budú neskôr odoslané informácie o rezervácii. Je potrebné vyplniť aj počet sedadiel, o ktoré má zákazník záujem. Po vyplnení základných údajov sú mu sprístupené sedadlá a môže začať výber.

- **Výber sedadiel** - Po sprístupnení sedadiel má zákazník možnosť kliknúť iba na voľné sedadlá zobrazené bielou farbou. Ostatné sedadlá sú uzavreté pomocou JavaScriptu, aby nebolo možné rezervovať sedadlá už rezervované iným zákazníkom. Po vybratí zvoleného počtu sedadiel, sa pomocou JavaScriptu zobrazí možnosť potvrdiť rezerváciu. Až po potvrdení je zákazník presmerovaný na ďalšiu stránku, kde sa zobrazia informácie, či už o filme samotnom, ale aj o vybraných sedadlách a zákazník má poslednú možnosť zmeniť emailovú adresu, na ktorú bude odoslaný informačný mail aj s QR kódom, ktorý slúži pre potvrdenie rezervácie v kine na pokladni.
- **Potvrdenie sedadiel** - Zákazník je po úspešnej rezervácii presmerovaný naspäť na výber miest, kde obdrží informačnú správu o tom, že mu bol na zadanú emailovú adresu odoslaný mail so základnými údajmi o filme a jeho rezervácii. K mailu je priložený aj QR kód.

Overenie dat pri spracovaní rezervácie prebieha v nasledovných fázach:

- **Overenie počtu vybratých sedadiel** - Overenie počtu sedadiel je zabezpečené formulárom HTML, tak aby vybraný počet nepresiahol viac ako 6. Pri odoslaní údajov je v controlleri overená emailová adresa aby zodpovedala formátu štandardnej emailovej adresy.
- **Prázdnosť sedadla** - Zaujímavým segmentom tejto časti je overenie prázdnosti sedadla. Overenie prebieha až pri odoslaní údajov, kde na začiatku controlleru sa overí najprv platnosť emailovej adresy a nasleduje preverenie sedadiel, či už nie sú obsadené. Každé sedadlo má v databáze príznak *obsadené*, kde sa môžu vyskytovať hodnoty 0, alebo 1.
- **Zhoda sedadiel** - Ak nastane situácia, že by dvaja, prípadne viacerí zákazníci po vyplnení formulára označili tie isté sedadlá, alebo aspoň jedno rovnaké sedadlo, dostanú sa k poslednému potvrdeniu rezervácie. Po potvrdení vždy controller na začiatku preverí pre každé sedadlo, či náhodou nie je už obsadené niekým iným. V prípade, že nie je voľné zákazník dostane chybovú hlášku a je presmerovaný na opakovaný výber sedadiel, kde sa mu už sedadlo zobrazí ako rezervované niekým iným. Týmto je zabezpečené, že každé sedadlo môže mať len jednu rezerváciu pre konkrétne premietanie.

  V druhom prípade, ak je rezervácia úspešná, vybrané sedadlá sa nájdu v databáze a pre každé sa nastaví príznak obsadenosti na hodnotu 1. Je vytvorený samostatný záznam v rezerváciách, kde identifikačné číslo vytvorenej rezervácie je priradené sedadlám, ktoré boli rezervované. Každá rezervácia obsahuje navyše aj emailovú adresu zadanú pri vyplnení údajov a tiež čas vytvorenia rezervácie pre vyhľadanie potrebných záznamov.

5.2.3 QR kód

Dôležitou častou rezervačnej časti je vytvorenie QR kódu, ktorý je vygenerovaný pomocou údajov o filme a je tiež zložený aj z unikátnych údajov, ku ktorým nemá zákazník prístup. Unikátne údaje v QR kóde slúžia pre bezpečnosť kina v prípade, že by sa našiel niekto, kto by si vytvoril vlastný QR kód a potom sa týmto prezentoval v kine a snažil sa získať miesta, ktoré mu nepatria.

Technológia vytvárajúca QR kód je zahrnutá v Laravel frameworku. Je potrebné ale doplniť framework o *simple-qrcode* knižnicu, ktorá obsahuje všetky potrebné funkcie pre prácu s QR kódom. Vytvorenie jednoduchého QR kódu môže vyzerať následovne:

```
QrCode::size(500)->format('png')
->generate('Hello World!',public_path('images\qrcode.png'));
```

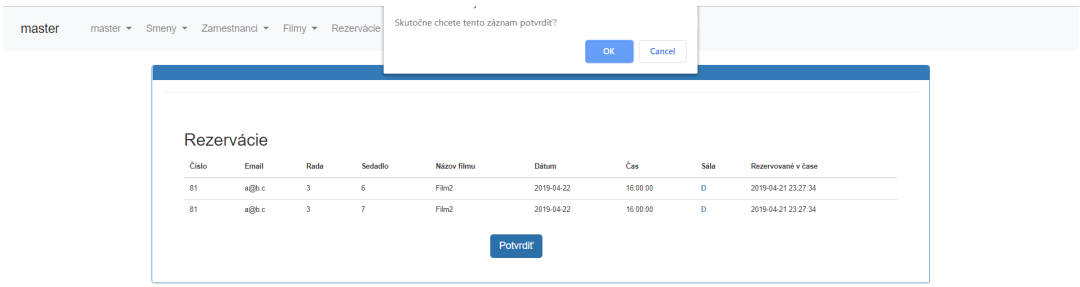
Ako môžeme vidieť, funkcia na generovanie QR kódu obsahuje niekoľko častí. *Size* je nepovinnou častou, ktorá slúži na určenie veľkosti obrázku a *format* určuje formát v ktorom chceme obrázok získať. Posledným a najdôležitejší segmentom je *generate*, ktorý sa podieľa na vytvorení QR kódu ako takého. Do tejto funkcie je potrebné vložiť dva parametre, kde prvým sú požadované dáta, ktoré má funkcia spracovať. Druhým parametrom je cesta, kde má byť vytvorený QR kód uložený.

Vytvorený QR kód je použitý ako príloha, ktorá je odoslaná s mailom na adresu zákazníka, ktorý si sedadlo rezervoval.

5.2.4 Overenie QR kódu

Informačný systém taktiež poskytuje výstup všetkých rezervácií pre pokladňu vo forme zoznamu. K tomuto zoznamu má prístup manažér, ktorý ho posunie pracovníkovi v pokladni, aby bolo možné finálne overenie. Toto prebieha vo forme zistenie, či aj v databáze rezervácií existuje konkrétny záznam rezervácie s daným identifikačným číslom, emailovou adresou a sedadlami na dané premietanie filmu. V tomto prípade je identifikačné číslo kľúčovým faktorom finálneho overenia.

Zákazník príde osobne s QR kódom na pokladňu, kde prebehne skenovanie QR kódu. Získané informácie o danej rezervácii, ktoré QR kód obsahuje sú následne použité pre vyhľadanie daného záznamu v databáze podľa identifikačného čísla. Po nájdení hľadaného záznamu prebehne zobrazenie údajov, ktoré môžme vidieť na obrázku 5.2. Zamestnanec kina pracujúci v pokladni následne porovná údaje QR kódu a záznamu. Ak sa QR kód aj záznam z databázy zhoduje, je možné prejsť k zaplateniu a potvrdeniu rezervácie.



Obrázek 5.2: Zobrazenie rezervácie

Ak je všetko v poriadku a rezervácia bola osobne zaplatená, je potrebné aby bola potvrdená zamestnancov pracujúcim v pokladni. Potvrdením rezervácie sa pristúpi do databázy a zmení sa príznak *zaplatené*, ktorý spôsobí, že sedadlá zmenia farbu na oranžovú. Pri následnom zobrazení daného premietania filmu, sú sedadlá označené ako kúpené a nie je možné k nim ďalej pristupovať. Zákazník následne dostane potvrdenie o platbe vo forme bločku, ktorý slúži aj ako lístok na daný film. Potvrdenie a zaplatenie sa musí vykonať najneskôr 10

minút pred začiatkom filmu, aby nebola rezervácia automaticky zrušená a sedadlá uvoľnené pre iných zákazníkov.

5.2.5 Užívateľské úrovne

Druhou a dôležitou častou implementácie bolo vytvoriť plánovací systém pre zamestnancov. Tento systém by sa nezaobišiel bez implementácie bezpečnostných úrovní pre zamestnancov a manažérov kina.

Základná myšlienka spočíta v rozdelení užívateľov na 3 úrovne:

- **Zákazník** - sa pripája na server bez autorizácie ako *hosť* a pristupuje iba k filmom a rezervácii lístkov,
- **Zamestnanec** - má možnosť sa prihlásiť do zamestnaneckého rozhrania po vytvorení účtu manažérom, pristupuje k plánovaciemu kalendáru smien,
- **Manažér** - prihlási sa podobne ako zamestnanec, aj keď jeho právomoci sú oveľa väčšie. Má posledné právo pri potvrdení, či pozmenení pracovnej smeny zamestnancov, má možnosť pridávať či upravovať filmy a následne k týmto filmom pridávať konkrétne predstavenia. Taktiež je manažérovi zverená administratíva užívateľských účtov.

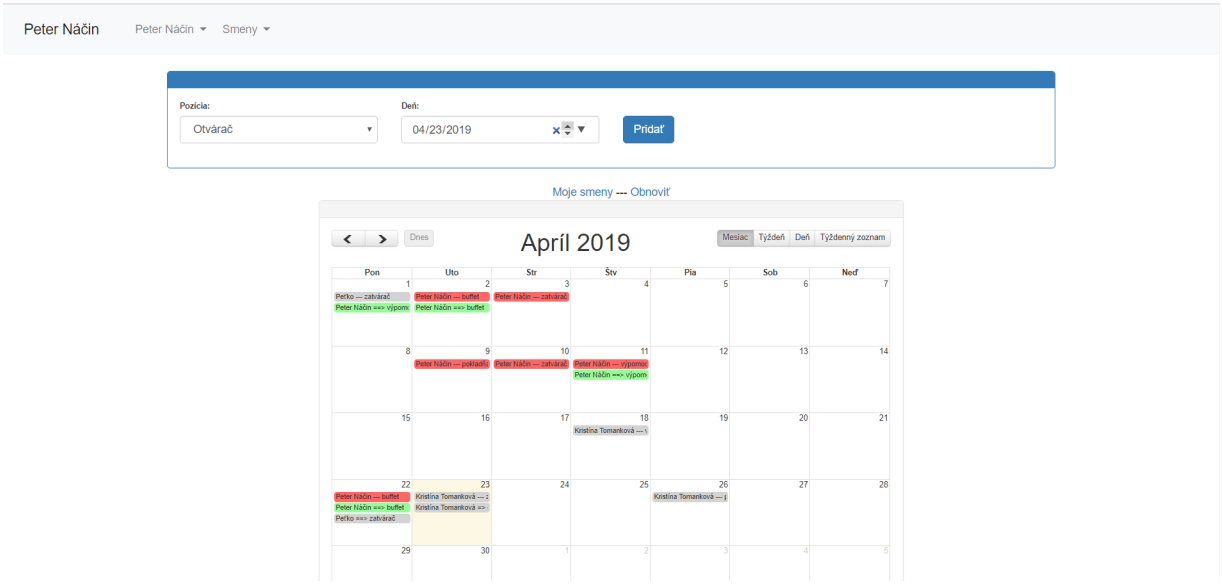
5.2.6 Nahlásenie na pracovnú smenu

Táto sekcia sa zaoberá problematikou nahlasovania a potvrdzovania pracovných smien. Na začiatku je potrebné spomenúť, že zamestnanci majú možnosť si nahlásiť pracovnú smenu, ale manažér rozhoduje o tom, či smenu potvrdí, alebo nie. Je vhodné spomenúť, že potvrdená pracovná smena reprezentuje pracovný záväzok zamestnanca, ktorý je povinný sa v daný deň do práce dostaviť.

Zamestnanec je po prihlásení presmerovaný na základnú stránku s údajmi o jeho profile, ktoré má možnosť upraviť a taktiež má možnosť zmeny hesla. Zaujímavejšou častou je ale kalendár, nad ktorým sa nachádza formulár, kde je možnosť si vybrať pozíciu a deň, na ktorý sa zamestnanec chce nahlásiť. Po potvrdení sa smena zapíše do databázy s príznakom *potvrdené* s hodnotou 0. Kalendár znova obnoví zobrazenie smien a novo pridaná smena sa objaví v konkrétnom dni kalendára s príslušným menom zamestnanca, ktorý ju nahlásil a s príslušnou pozíciou, ktorú si vybral. Zamestnanec túto smenu vidí s červeným pozadím, aby bolo jednoducho odlíšiteľné, že smena ešte nie je potvrdená. V opačnom prípade, ak je smena potvrdená pozadie, je zelené. Ostatné smeny, ktoré sa netýkajú konkrétneho zamestnanca, ktorý je prihlásený sú zobrazené šedou farbou. Taktiež bola implementovaná možnosť zobraziť len svoje smeny. Týmto filtrom je umožnené veľmi prehľadné prechádzanie v kalendári.

Veľmi dôležitou častou je manažérsky prístup ku kalendáru, ktorý poskytuje potvrdenie konkrétnej smeny na daný deň. Hlavným rozdielom zobrazovania smien v kalendári je pre prístup manažéra, ktorý nepotrebuje vedieť kto sa na daný hlási, ale na akú pozíciu sa hlási. Z tohto dôvodu má manažér možnosť filtrovať kalendár medzi nahlásenými smenami a potvrdenými smenami. Každá pozícia má priradenú charakteristickú farbu a nad kalendárom je malá legenda, ktorá zobrazuje všetky pozície spojené s farbou. Potvrdenie prebieha veľmi podobne ako nahlasovanie a to pomocou formulára, ktorý požaduje vyplnenie mena zamestnanca, pozíciu, deň a čas pracovnej smeny. Po vytvorení pracovnej smeny sa v databáze vytvorí pracovná smena s príznakom *potvrdené* na hodnote 1.

Obom manažérovi, aj zamestnancovi je poskytnutý zoznam všetkých smien, kde v prípade zamestnanca je možné meniť, prípadne zmazať smeny na ktoré sa nahlásil a manažér má možnosť editovať a zmazať smeny, ktoré sú potvrdené manažérom a reprezentujú skutočnú pracovnú smenu.



Obrázek 5.3: Zamestnanecký kalendár pracovných smien

5.2.7 Administrácia

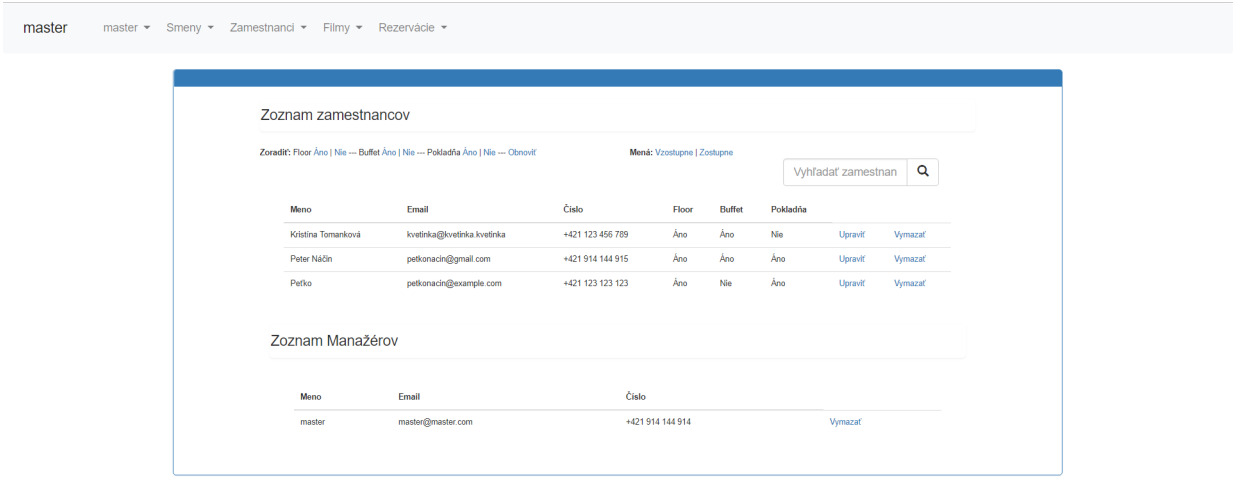
Keďže systém obsahuje mnoho rôznych informácií manažérska úroveň prístupu má v sebe implementovaný aj prístup ku všetkým potrebným údajom. Ako už bolo spomenuté v iných kapitolách manažér má prístup a má na starosti administráciu filmov ako takých, ale aj predstavení, ktoré súvisia s jednotlivými filmami. Administrácia sa netýka len filmov, ale podstatnou častou je aj administrácia pracovných smien a zamestnaneckých účtov. Pre všetky 3 možnosti filmy, smeny a zamestnanci je manažérovi poskytnutý zoznam všetkých informácií s podporou stránkovia. Každá jedna variácia zoznamu má implementovanú možnosť vyhľadávania v danom zozname s podporou stránkovia.

Ako doplnok sú pridané jednoduché filtre pre rýchle zobrazenie zamestnancov podľa ich schopnosti vykonávať určitú pozíciu vo forme áno/nie, alebo filter v podobe zostupného alebo vzostupného zoradenia informácií, či už podľa abecedy v prípade mien zamestnancov, alebo podľa dátumu v prípade filmov a pracovných smien.

5.2.8 Databáza, práca s údajmi

V tejto časti si rozoberieme, kde a ako prebieha vyberanie údajov z databázy. Následne sa pozrieme na to, kde a ako sú tieto dáta zobrazené. Vybratie a spracovanie dát typicky prebieha v controller súboroch pomocou PHP jazyka.

Výber dát prebieha pomocou SQL(Structured Query Language) dopytov. Absolútnym štandardom sú SQL select dopyty, ktoré vyberú dáta z tabuľky. Keďže v skutočnom svete sú jednotlivé objekty reálneho sveta úzko prepojené. Navrhnutá databáza má jednotlivé entity prepojené a to pomocou cudzích kľúčov. Aby bolo možné informácie z jednotlivých tabu-



Obrázek 5.4: Zobrazenie administratívnej časti zamestnancov

liek vybrať a spojiť, je potrebné spojiť aj jednotlivé tabuľky databázy. Spojenie prebieha pomocou funkcie *->join*, kde je potrebné porovnať hodnotu cudzieho kľúča, aby sa našiel záznam, ktorý hľadáme. Laravel framework ponúka použiť mierne pozmenenú syntax SQL dopytov ako štandardný SQL jazyk. Predstavme si, že potrebujeme získať dáta o všetkých predstaveniach, ktorých názov filmu je "TestovacíFilm". Dopyt spájajúci dve tabuľky môže vyzerať nasledovne:

```
$movies = DB::table('movies')
  ->where('movies.title','=','TestovacíFilm')
  ->join('show','show.movie_id','=', 'movies.id')
  ->select('movies.title','show.date','show.time','show.id')
  ->get();
```

Pri tvorbe dopytu si vyberieme premennú, do ktorej vybrané dáta uložíme, v našom prípade to je premenná *$movies*. Následne vyberieme tabuľku, ku ktorej chceme pristupovať, teda tabuľka **movies**. Keďže sme si určili vybrať predstavenia konkrétneho filmu, je potrebné určiť, o ktorý film sa jedná. Tu vchádza do dopytu časť *->where()*, ktorá má 3 parametre. Ako prvý je stĺpec tabuľky, v ktorom budeme vykonávať prehľadávania. Druhým parameter reprezentuje podmienku a posledný tretí určuje hodnotu, ktorú hľadáme. Predpokladajme, že identifikačné číslo záznamu, ktorý bol nájdený s názvom filmu "TestovacíFilm"je 123.

Pretože potrebujeme pristupovať aj k údajom z inej tabuľky použijeme funkciu *->join()*, ktorá má 4 parametre. Prvý parameter reprezentuje názov druhej tabuľky, ktorá má byť spojená s prvou, v našom prípade **show**. Ďalšie parametre reprezentujú cudzí kľúč tabuľky **show**, ktorý je porovnaný s primárnym kľúčom tabuľky **movies**.

Jednoduchšie môžme povedať, že dopyt nájde všetky záznamy z tabuľky **show**, ktoré spájajú predstavenia s filmom pomocou hodnoty 123 z tabuľky **movies**.

Výberová časť dopytu (select) má na starosti vybrať údaje tabuliek, s ktorými budeme ďalej pracovať. *Select* ponúka možnosť zvoliť iba konkrétne stĺpce tabuliek, tak ako je uve-

dené v príklade. Spojenie tabuľky a konkrétneho stĺpca v tabuľke prebieha pomocou znaku bodky a k dátam pristupujeme podľa jednotlivých názvov stĺpov tabuľky. Predpokladajme, že selectom:

->select('movies.title'),

vyberieme názvy filmov, ktoré potrebujeme. Pre posielanie dát na view je potrebné použiť foreach cyklus

foreach($movies as $movie)

a následne môžme zobraziť dáta pomocou $movie->title. Musíme si ale dávať pozor, ak spájame viacej tabuliek. Môže sa stať, že niektoré riadky majú rovnaký názov napráklad ID ako identifikačné číslo. Ak by sme potrebovali získať identifikačné číslo filmu, ale aj predstavenia, je vhodné priradiť každému názov pomocou, *as* a nový názov. Napríklad:

->select('movies.id as movie_id','show.id as show_id').

Týmto spôsobom vytvoríme dve premenné namiesto jednej a zabezpečíme tak konzistenciu dát. V prípade, že sa select rozhodneme nepoužiť, automaticky sa vyberú všetky dáta, ktoré záznam obsahuje. Tu však môže nastať presne tento problém s rovnakými názvami riadkov rozdielnych tabuliek.

Poslednou časťou je funkcia *->get*, ktorá symbolizuje výber dát. Alternatívou, ktorá je veľmi vhodná pri väčšom počte dát je *->paginate(10)*, ktorá poskytuje stránkovie záznamov, kde jediný parameter funkcie v našom prípade 10 udáva ich počet a určí, ktoré budú zobrazené na jednej stránke. V našej práci je nastavené stránkovia na 15 záznamov. Do view súboru je ešte potrebné pridať zobrazenie stránkovania, aby mal užívateľ možnosť prejsť na ďalšiu stránku záznamov. Jedná sa o funkciu *render()*, ktorú je potrebné spojiť s dátami určenými na stránkovanie. Časť kódu môže vyzerať následovne:

$movies->render().

Aby mohli byť dáta zobrazené musíme ich poslať na view súbor. Laravel ponúka viac možností, ako posunúť vybrané dáta z contolleru na view. Štandardne sa používa používa funkcia *->with('data',$data)*, kde prvý parameter reprezentuje pomenovanie posielaných dát, ku ktorým bude pomocou tohto pomenovania pristupované vo view súbore. Druhý parameter obsahuje premennú vytvorenú v contoller súbore, ktorá obsahuje dáta, ktoré chceme poslať.

Funkcia *->with('data',$data)* má možnosť byť modifikovaná a to vo forme pridania názvu dát priamo do názvu funkcie. Existujú aj ďalšie možnosti na posielanie dát a to napríklad funkcia *compact($data*, alebo použitím druhého parametra.

Pre lepšie pochopenie si uvedieme ku každej možnosti príklad.

return view('index')->withName('Meno Priezvisko');
return view('index')->with('data',$data);
return view('index', compact('data'));
return view('index', ["dáta->$data]);

Po odoslaní dát na view súbor je potrebné ich zobraziť. Prvý uvedený príklad ako modifikovaná funkcia *->withName('Meno Priezvisko')* obsahuje iba jediný reťazec a preto je možné k nej vo view súbore pristúpiť priamo pomocou premennej $name.

Naopak všetky ostatné spôsoby posielania majú rovnakú funkcionalitu a rovnaký výsledok preposlania dát, ale predpokladajú, že posielané dáta sú vo forme poľa. Preposlané pole môže obsahovať viac, jednu, alebo aj žiadnu položku. V prípade prázdneho poľa sa jednoducho nezobrazia žiadne informácie. Následne je potrebné načítať všetky údaje z poľa, kde do procesu prichádza PHP jazyk, ktorý poskytuje *foreach* cyklus. Tento cyklus autonómne zistí počet prvkov v poli dát, ktoré boli preposlané z controller súboru. Týmto spôsobom pre každý jeden prvok poľa dokážeme zobraziť preposlané dáta.

Predpokladajme, že pomocou SQL dopytu chceme získať údaje o filmoch, ktoré obsahujú len identifikačné číslo ako ID a názov filmu ako title. Získané dáta uložíme do premennej *$movies* a view súbor bude mať názov *index.blade.php*. Preposlanie dát a následné zobrazenie môže vyzerať napríklad takto:

return view('index')->with('movies',$movies);

```
@foreach($movies as $movie)
    <div>$movie->id</div>
    <div>$movie->title</div>
@endforeach
```

Prvá časť odpovedá preposlaniu dát na view pomocou controlleru, pričom v druhej časti vidíme zobrazenie dát. Ako môžeme vidieť na príklade, zobrazenie dát je možné vsadiť priamo do HTML kódu a zabezpečiť tak vhodné zobrazenie, ktoré je následne načítané webovým prehliadačom.

5.3 Uživateľské rozhranie

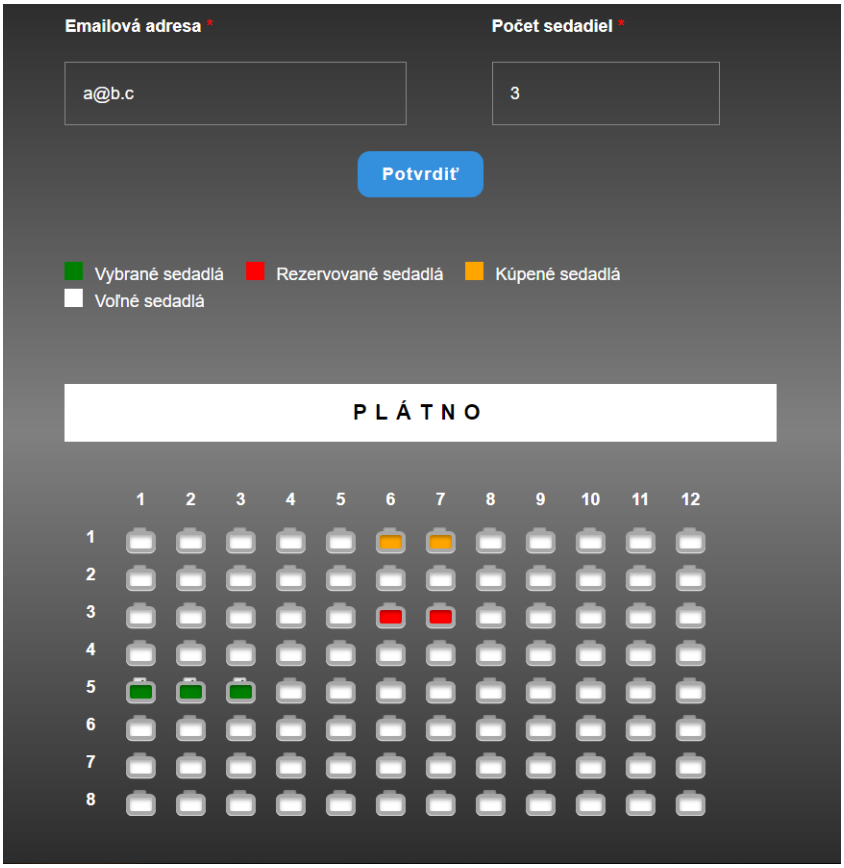
Uživateľské rozhranie je vytvorené pomocou HTML a CSS jazyka, kde CSS súbory sa nachádzajú v zložke *css* a HTML súbory sú štandardné view súbory v zložke *view*.

Zákaznícka časť uživateľského rozhrania tvorí len časť práce. Ide primárne o vykreslenie zoznamu premietaných filmov, z ktorých si zákazník má možnosť vybrať film, o ktorý má záujem. Zaujímavejšou častou je ale vykreslenie sály, kde sú potrebné pokročilejšie znalosti z oblasti CSS jazyka. Ako ukážku si uvedieme uživateľské rozhranie pri zobrazení sály konkrétneho premietania viď. obrázok 5.5.

Zamestnanecké užívateľské rozhranie sa skladá z najväčšej časti z kalendára, ktorý je sprostredkovaný JavaScriptom FullCalendar 3.3.5. Naopak menšiu časť tvorí zoznam smien a úprava profilových informácií. Manažérskemu prístupu sú navyše priradené časti zobrazujúce ostatné administratívne funkcie.

Keďže očakávame prístup do systému aj pomocou mobilných zariadení, systém bol doladený pomocou Bootstrapu 3.3.4, aby bola zabezpečená responzívnosť či už pre zákazníkov, ale aj pre zamestnancov a manažérov v prípade prístupu cez mobilné zariadenie.

Väčšie HTML tabuľky a zobrazenie všetkých sedadiel sály, ktoré by mohlo mať problematické načítanie na menších displejoch je vyriešené hlavne pomocou *overflow* možnosti, ktorá poskytuje vytvorenie *scrollovacieho okna* a tak umožní zobraziť vhodne časti, ktoré by inak presiahli veľkosť displeja mobilného telefónu.



Obrázek 5.5: Zobrazenie sály

Taktiež veľkú rolu v zabezpečovaní responzivity hrá Bootstrap vytvorením 12 stĺpcov aj pre menšie displeje a to pomocou *col-veľkosť displeja-počet stĺpcov*. Konkrétne toto nastavenie môžme aplikovať na štandardnú veľkosť monitora, ktorá má parameter veľkosti displeja stredný, čiže médium. Takto napríklad vznikne trieda *col-md-12*, ktorá zabezpečí rozloženie daného obsahu na celú stránku na stredne veľkom monitore. Pomocou zámeny parametru *md* za parameter *xs* dosiahneme to, že sa aj najmenší displej prispôsobí rozloženiu obsahu. Aj keď štandardne *xs* nastavenia odpovedá 1.5 násobku *md* nastaveniu, nie je možné toto brať ako pravidlo a bolo potrebné stĺpce otestovať a vhodne prispôsobiť pre každú stránku.

Kapitola 6

Testovanie a ladenie

Testovanie považujeme za veľmi dôležitú súčasť vývoja aplikácií a môže prebiehať rôznymi formami. Hlavnou podstatou testovania je overiť správnosť výstupných informácií, správny priebeh jednotlivých funkcií, ale aj overenie interakcie medzi jednotlivými časťami a zabezpečenie tak celkovej funkčnosti programu.

6.1 Priebeh testovania

Testovanie prebiehalo počas celého vytvárania informačného systému, kde si rozdelíme formy testovania do 3 rôznych kategórií:

- **Správnosť výstupov**
- **Grafické rozhranie**
- **Responzivita**

Správny výstup informácií bol testovaný počas celého priebehu implementácie. Po vytvorení každej funkcie prebehlo zobrazenie informácií pre overenie správnosti výstupu a taktiež následne po dokončení logického celku prebehlo testovacie danej časti. V prípade objavenia chyby, ma Laravel debugger informoval o danej chybe, kde sa nachádza a o aký typ chyby sa jedná. Po dokončení celého informačného systému prebehlo otestovanie celkovej funkcionality od administratívy až po rezerváciu sedadiel. Jedným z testov bolo aj vymazanie všetkých záznamov databázy a následné postupné pridávanie pomocou implementovaných funkcií v systéme.

Grafické rozhranie bolo taktiež testované počas vývoja programu. Rozhranie pre užívateľov je pojaté hlavne v smere jednoduchosti a intuitívnosti, ktoré dáva užívateľovi možnosť prechádzať systém bez vážnych problémov. Testovanie motívu grafického designu bolo pojaté rozdelením do zákazníckeho prostredia a do prostredia administrácie. Obidve prostredia sú rozdielne, ale v danom prostredí sa grafický motív zhoduje.

V poslednej časti testovania, ktorá má charakter grafického rozhrania sme sa zamerali na responzívnosť mobilných zariadení. Responzívnosť bola v prvom rade otestovaná pomocou debuggera na prehliadači Google Chrome, ktorý poskytuje ukážku, ako by stránka vyzerala na rôznych zariadeniach. Po prispôsobení grafického rozhrania s podporou debuggeru na rôzne mobilné zariadenia, bola pomocou lokálneho servera otestovaná responzivita na skutočnom Iphone SE/5, ktorý vykazoval rovnaké výsledky ako debugger prehliadaču Google Chrome.

Po potvrdení funkčnosti bolo naplánované stretnutie s dvomi manažérmi kina, ktorí mali aplikáciu k dispozícii a mohli si vyskúšať jednotlivé funkcie na lokálnom serveri. Jednalo sa hlavne o administratívne funkcie, ktoré majú manažéri v dispozícii.

6.2 Nasadenie systému

Nasadenie informačného systému je naplánované počas leta 2019. Prebiehajú rokovania s vedením kina a to nie len v Trenčíne, ktorý bol zadávateľom projektu, ale aj predstavenstvom v Bratislave, pretože firma má prevádzky po celom Slovensku. Taktiež prebieha komunikácia s technickými pracovníkmi zabezpečujúcimi správu kina a technického prostredia (hw, sw), na ktoré by pomohli systém nasadiť. Nasadenie systému zahŕňa:

- definovanie a prípadne doplnenie technického prostredia,
  - do ktorého bude informačný systém aplikovaný, resp. doplnený
  - správu technického prostredia a aplikácie
- administratívnu časť
  - riešiť problém s pridávaním pracovných smien
  - rezervačný systém lístkov by bolo potrebné v budúcnosti spojiť s pokladňou nachádzajúcou sa fyzicky v kine.

Kapitola 7

Záver

Základnou myšlienkou a cieľom tejto práce bolo vytvoriť webový informačný systém pre Trenčianske kino Cinemax. Primárnou funkciou systému by bolo plánovanie pracovných smien pre zamestnancov a celková administratíva v kine. Administratívna časť by obsahovala správu zamestnancov, pracovných smien, ale aj filmových predstavení pri každodennej činnosti kina. Vytvorený systém by mal poskytovať zákazníkom aj rezerváciu lístkov na premietané filmy, kde jednou z častí administrácie by bola aj správa rezervácií a následné overenie platnosti rezervácie.

Pred samotnou implementáciou systému bolo potrebné zistiť v akom stave sa nachádza momentálna situácia v kine. Po oboznámení sa so súčastným stavom a stanovení požiadavkov od vedenia kina, som sa zoznámil s používanými technológiami pri tvorbe webových informačných systémov. Použitie zaujímavých technológií zahrňovali použitie frameworku, ktorým v našom prípade bol Laravel. V tomto frameworku bol následne celý webový informačný systém implementovaný. Ďalším bodom bolo vytvorenie návrhu, ktoré bolo uskutočnené vo forme Use Case a ER diagramu. Tento návrh smeroval k široko používanej relačnej databáze MySQL, ktorá bola následne vysvetlená v technickej správe. Veľkou častou bola samotná implementácia, kde po vytvorení systému boli jednotlivo popísané časti implementácie. Vysvetlená bola v prvom rade štruktúra adresárov vo frameworku Laravel. Nasledoval popis jednotlivých časti informačného systému ako napríklad: rezervácia lístkov, nahlasovanie na pracovnú smenu. ale aj práca s databázou. Opis práce bol zakončený testovaním jednotlivých častí systému.

Záverom konštatujem, že počiatočné podmienky a návrh práce boli splnené. Výsledkom práce je vytvorený funkčný webový informačný systém, ktorý spája administratívnu časť a informačnú časť. Systém zjednodušuje pracu pre pracovníkov Trenčianskeho kina. Zároveň zabezpečuje správu zamestnancov, pracovných smien pre zamestnancov a celkovú prácu s filmami premietanými každodenne v kine.

7.1 Možné rozšírenia

V prípade zavedenia systému a plného využívania zamestnancami, by možným rozšírením bolo pridanie notifikácií pri potvrdení, či zmene pracovnej smeny, aby mohol byť zamestnanec informovaný, aj bez neustáleho sledovania systému. Kedže doposiaľ komunikácia funguje vo forme sociálnych sietí, zaujímavou myšlienkou, ktorá by smerovala k zlepšeniu komunikácie, by mohlo byť pridanie možnosti posielať správy medzi pracovníkmi pomocou informačného systému, alebo vytvorenie četovacej miestnosti, kde by mohla prebiehať celková

komunikácia ohľadom plánovania pracovných smien, alebo organizácie administratívnych záležitostí.

Kedže práca obsahovala aj časť, ktorá poskytovala rezerváciu lístkov, najpodstatnejším rozšírením by mohlo byť prepojenie sytému s fyzickým predajom lístkov v pokladni kina. Pre splnenie tohto rozšírenia, je potrebné prepojiť už existujúcu databázu kina a databázu použitú v práci informačného systému. Touto cestou by bolo taktiež vhodné vytvoriť grafické rozhranie pre zamestnancov pracujúcich v pokladni, ktorí zabezpečujú predaj lístkov.

Literatura

[1] *Bootstrap · The most popular HTML, CSS, and JS library in the world.* [Online; navštíveno 15.03.2019].
URL https://getbootstrap.com/

[2] *FullCalendar documentation.* [Online; navštíveno 12.03.2019].
URL https://fullcalendar.io/

[3] *LARAVEL - INVENTORY APP.* [Online; navštíveno 21.03.2019].
URL https://www.rolandbecsi.com/blog/inventory-app-tutorial-in-laravel

[4] *Laravel - The PHP Framework For Web Artisans.* [Online; navštíveno 27.02.2019].
URL https://laravel.com/

[5] *Nette Framework: Rychlý a pohodlný vývoj webových aplikací v PHP.* [Online; navštíveno 27.02.2019].
URL https://nette.org/cs/

[6] *PHP Tutorial.* [Online; navštíveno 27.02.2019].
URL https://www.php.net/

[7] *Symfony, High Performance PHP Framework for Web Development.* [Online; navštíveno 05.03.2019].
URL https://symfony.com/

[8] Hernandez, M. J.: *Návrh databází.* Grada Publishing, a.s., 2006, ISBN 80-247-0900-7.

[9] Hernandez, M. J.; Viescas, J. L.: *Myslíme v jazyku SQL tvorba dotazů.* Grada Publishing, a.s., 2004, ISBN 80-247-0899-X.

[10] Oppel, A. J.: *Databáze bez předchozích znalostí.* Computer Press, a.s., 2006, ISBN 80-251-1199-7.

[11] w3schools.com: *CSS Tutorial.* [Online; navštíveno 15.03.2019].
URL https://www.w3schools.com/css/

[12] w3schools.com: *HTML Tutorial.* [Online; navštíveno 15.03.2019].
URL https://www.w3schools.com/html/

[13] w3schools.com: *JavaScript Tutorial.* [Online; navštíveno 15.03.2019].
URL https://www.w3schools.com/js/

[14] Zendulka, J.; Rudolfová, I.: *Databázové systémy. IDS. Studijní opora.* FIT VUT v Brně, 2006(rev. 2018).

Příloha A

Obsah priloženého pamäťového média - CD

- **db/** - skript pre vytvorenie databázy
- **doc/** - dokumentácia
- **src/** - zdrojové kódy