ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

# Systém pro správu divadla

**Petr Jeřábek**
Otevřená informatika - Software

2021
Vedoucí práce: Ing. Božena Mannová, PhD.

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jeřábek** Jméno: **Petr** Osobní číslo: **483630**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Systém pro správu divadla**

Název bakalářské práce anglicky:

**Theater management system**

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci pro správu činnosti divadla. Seznamte se s problematikou současné praxe správy činnosti v divadle. Na základě vyhodnocení těchto poznatků specifikujte konkrétní požadavky na typ aplikace její funkcionality. Věnujte se problematice správy jednotlivých představení a jejich obsazení herci, techniky a dalšími zaměstnanci divadla. Uvažujte i nutné změny v důsledku mimořádných situací. Práce se nebude zabývat řízením financí divadla a prodejem lístků. Specifikujte konkrétní
funkční a nefunkční požadavky na systém a na jejich základě navrhněte aplikaci. Navržený program implementujte a otestujte.

Seznam doporučené literatury:

[1] Zlámalová, A. (2018). IS pro správu činnosti divadla, Bakalářská práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů
[2] Sling (webová aplikace pro plánování směn) [cit. 1.12.2020].
Dostupné z: https://app.getsling.com

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. Božena Mannová, Ph.D., kabinet výuky informatiky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.02.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2022**

| Ing. Božena Mannová, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

| Datum převzetí zadání | Podpis studenta |
|---|---|

# Poděkování / Prohlášení

Rád bych poděkoval Ing. Boženě Mannové, PhD. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20.5.2021

.....

Petr Jeřábek

# Abstrakt / Abstract

Práce je zaměřena na návrh a vývoj aplikace sloužící k usnadnění organizace a provozu divadla. V rámci práce byl proveden průzkum organizace v divadle a k tomu využitých technologií. Potom byl proveden návrh a specifikace požadavků na vyvíjenou aplikaci, která byla následně implementována. Aplikace splňuje stanovené požadavky, byla otestována a je připravena k použití.

**Klíčová slova:** divadlo, informační systém, Android, Java

The work is focused on design and development of an application, that simplifies theatre management. Within the framework of this thesis, research of theatre management and technologies used for it has been made. Subsequently requirement specification and design of the developed application has been done. The application was implemented according to requirements, tested and is ready to use.

**Keywords:** theatre, information system, Android, Java

**Title translation:** System for theatre management

# Obsah /

# Kapitola 1
# Úvod

Tato práce se zabývá návrhem a vytvořením uživatelsky přívětivé aplikace pro usnadnění provozu a organizace divadla. Aplikace bude sloužit pouze vnitřnímu provozu divadla, nikoli veřejnosti a nebude se zabývat financemi.

## 1.1 Motivace

Většina lidí pohybujících se v divadelních kruzích nemá blízko k digitálním technologiím, a proto je těžké v tomto odvětví prosazovat informační systémy. Chci tedy navrhnout a vytvořit systém, který bude této skupině uživatelů přívětivý a usnadní fungování divadla z pohledu plánování směn a bude udržovat všechny důležité informace jednom místě.

## 1.2 Cíl práce

Cílem této práce je navrhnout, implementovat a otestovat mobilní aplikaci platformy Android pro správu činnosti divadla. Aplikace má sloužit k vytváření směn, zobrazování směn uživatelům na základě jejich pracovní role a následné přihlašování na tyto směny, vytváření uživatelských účtů a jejich úprav a vytváření pracovních rolí pro uživatele.

# Kapitola 2
# Analýza řešení

V této kapitole se práce zabývá průzkumem fungování divadelního provozu, používaných technologií a existujících řešení. Na základě tohoto průzkumu pak byla vytvořena specifikace požadavků a případy užití.

## 2.1 Průzkum organizace v divadlech

Pro průzkum organizace a používaných systémů v divadlech jsem se spojil s paní Magdalenou Novotnou, tajemnicí divadla Na zábradlí, která mi vysvětlila, jak provoz v divadle funguje a jaké používají k organizaci prostředky a technologie. Pro plánování používají Google Tabulky, které ale slouží jen jako sdílený prostor, bez dalších užitečných funkcionalit. Důležité funkcionality, které by měl navrhovaný systém poskytovat jsou získávání odpracovaných hodin a počet herci odehraných představení, protože tyto informace musí získávat v současnosti ručně. Dále by bylo užitečné posílání notifikací na email i SMS zprávou, hlavně o změnách již přihlášených směn. Typické je i to, že každé představení, nebo i jiná akce v divadle potřebuje různé počty zaměstnanců různých rolí, například maskérů nebo kulisáků. Proto by také usnadnilo práci mít možnost nastavit si šablony pro různá představení, nebo jiné akce, s předem nastaveným počtem potřebných zaměstnanců, aby se to vždy nemuselo vypisovat ručně. Velkým problémem při organizaci uměleckého provozu je také, že mnoho herců funguje ve více divadlech a je velmi složité zjistit jejich časové možnosti a dle toho plánovat představení. Avšak zavedení systému, který by tuto problematiku řešil, by vyžadovalo zapojení většiny herců i divadel a bylo by to personálně, časově i finančně velmi náročné. Tím se tedy v tomto projektu nezabývám.

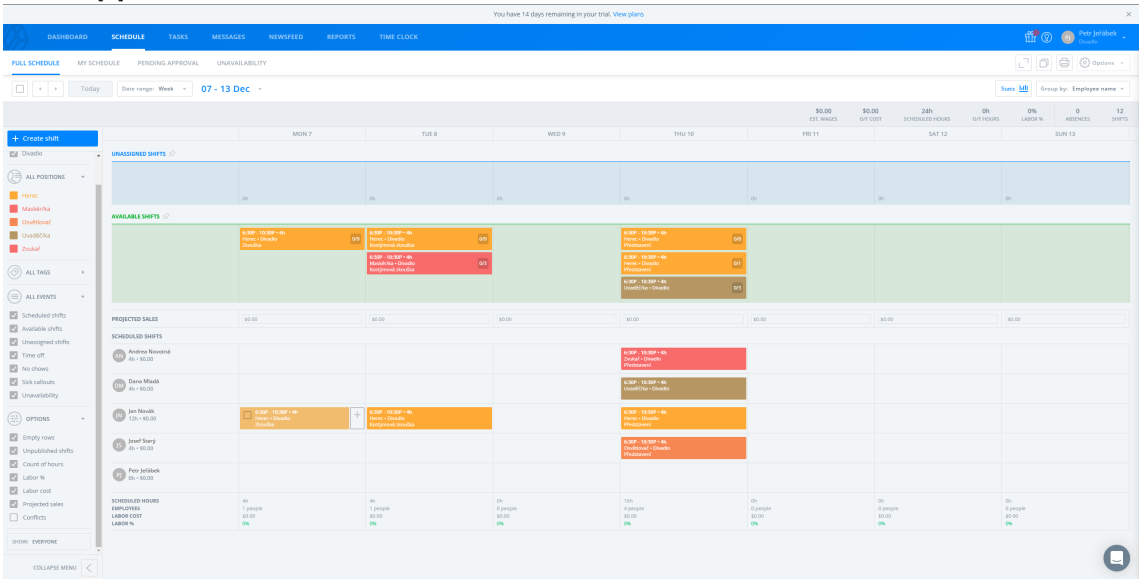
## 2.2 Průzkum existujících řešení

Na trhu existuje mnoho aplikací typu ERP (Enterprise Resource Planning), které slouží k organizaci a plánování podnikových procesů. Avšak tyto aplikace jsou pro toto použití příliš komplikované a drahé. Zmínil bych, že Národní divadlo používá informační systém Karat, což je komplexní ERP systém, ve kterém se spravuje většina procesů, ale většina divadel u nás je typicky mnohem menší a pro takové aplikace nemá užitek, ani dostatek prostředků. Existují i aplikace určené přímo pro organizaci divadla nebo obecně umělecky zaměřených akcí jako Theatron nebo Propared. Proto zde hodnotím aplikaci pro obecné plánování směn, která se velmi blíží tomu, co v tomto projektu navrhuji.

### 2.2.1 Sling

Webová aplikace pro plánování směn. Nabízí vytvoření uživatelských rolí a vytváření směn pro dané role uživatelů. Směny jo možné rovnou přiřazovat konkrétním uživatelům nebo je vystavit jako dostupné a mohou být přiřazeny později, či se na ně uživatelé

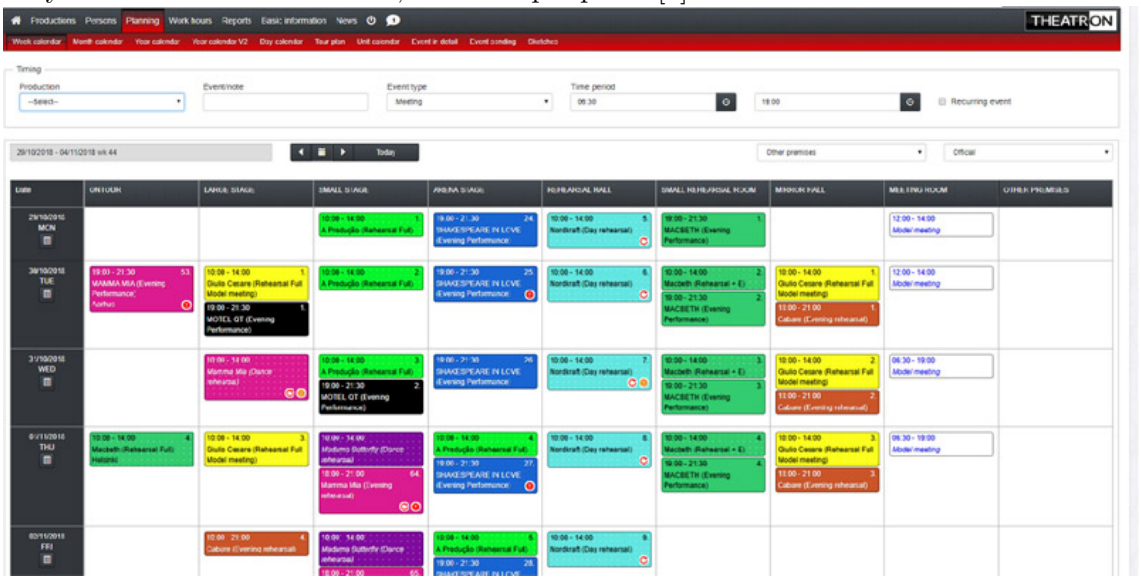
můžou hlásit. Systém neumožňuje vytvářet šablony s předem definovaným počtem potřebných uživatelských rolí a neumožňuje získávat počet odpracovaných směn nějakého typu. [1]



**Obrázek 2.1.** Ukázka webové aplikace Sling

### 2.2.2 Theatron

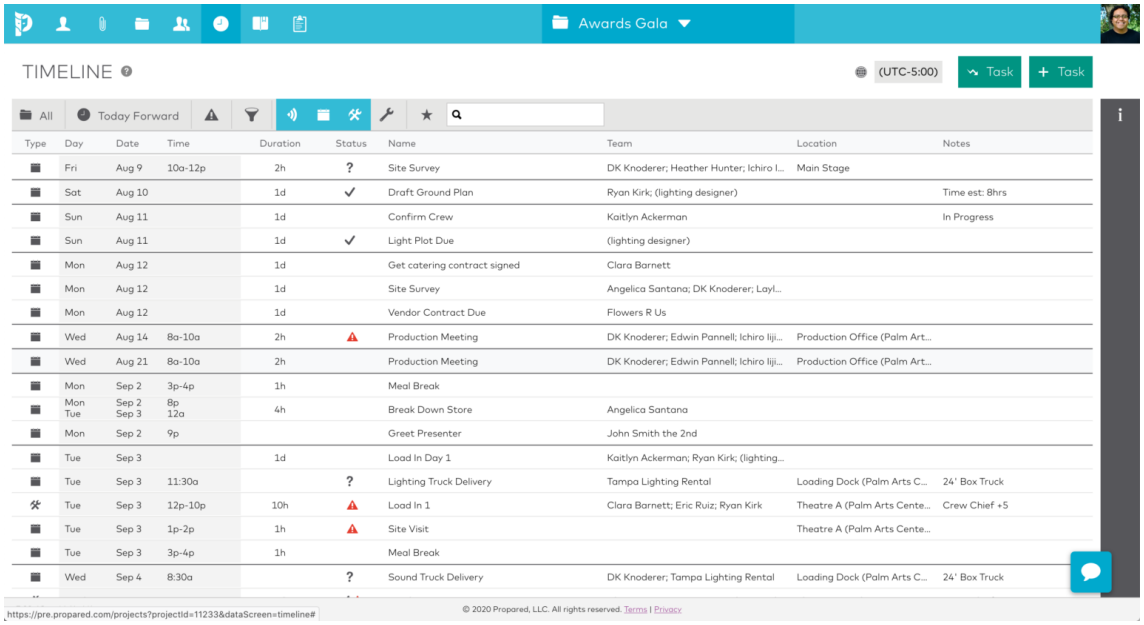
Theatron je software od finské společnosti Mansoft pro správu uměleckého, technického a administrativního personálu divadla. Zajišťuje přehlednost, sdílení a správu rozpisů směn a akcí. Jde o webovou aplikaci poskytovanou jako `Service solution`, tedy poskytovatel se stará o nasazení, údržbu i podporu. [2]



**Obrázek 2.2.** Ukázka aplikace Theatron

### 2.2.3 Propared

V případě aplikace Propared jde také o webovou aplikaci, která je přístupná i z mobilních zařízení. Nabízí mnoho funkcionalit, kromě plánování a přihlašování směn. [3] V některých případech jsou ikonky nepřehledné a menu se slovním popisem by bylo přehlednější. [4]




**Obrázek 2.3.** Ukázka aplikace Prepared

### 2.2.4 Srovnání existujících řešení

Všechny zmiňované aplikace jsou webové, tedy lze je používat bez nutnosti instalace a konfigurace. Theatron a Sling zobrazují směny a akce v kalendáři, což může být nevýhodou, při používání v menším divadle, kde neprobíhá tak velké množství směn. Kalendář pak bude poměrně prázdný a uživatel bude muset dlouho hledat, než najde nějakou směnu. Oproti tomu Propared používá seznam řazený podle data a nezobrazuje tedy prázdné dny, a tudíž se na jednu stránku vejde mnohem více informací. Zároveň použití seznamu je podobnější systému tabulek a seznamů používaných v divadle v současné době.

## 2.3 Specifikace požadavků navrhovaného systému

Na základě provedené analýzy byly specifikovány funkční a nefunkční, neboli kvalitativní, požadavky na implementaci aplikace. Požadavky reagují na potřeby organizace a popisují cíle a důvody realizace. Navrhovaná aplikace by měla být uživatelsky přívětivá a jednoduchá. Požadavky jsou specifikovány i s ohledem na současná řešení, aby přechod na novou aplikaci byl pro uživatele co nejjednodušší.

### 2.3.1 Funkční požadavky

Funkční požadavky specifikují všechny funkce, které bude aplikace nabízet. Požadavky jsou rozděleny do dvou sekcí podle uživatelských pravomocí.

**Admin**

Tyto požadavky se zaměřují na funkcionality aplikace, které bude používat správce nebo vedoucí pracovník.

**FR001 – Vytváření uživatelských účtů**

Jako admin potřebuji vytvářet uživatelské účty pro zaměstnance divadla.

**FR002 – Vytváření uživatelských rolí**

Jako admin potřebuji vytvářet uživatelské role, které budu přiřazovat jednotlivým uživatelským účtům.

**FR003 – Vytvoření směny**

Jako admin potřebuji vytvářet směny a u každé specifikovat počty uživatelských rolí, které jsou pro danou směnu potřeba. Nebo vybrat představení, ke kterému se tato směna vztahuje, a pro představení budou tyto počty definovány.

**FR004 – Editace směny**

Jako admin potřebuji možnost změnit parametry již vytvořené směny a notifikovat o tom dotčené uživatele.

**FR005 – Zrušení směny**

Jako admin potřebuji možnost odstranit vytvořené směny a notifikovat o tom dotčené uživatele.

**FR006 – Vytvoření šablony**

Jako správce potřebuji vytvářet šablony, u kterých budu kromě názvu a doby trvání definovat potřebné počty zaměstnanců různých rolí, abych při vytváření směn nemusel vždy ručně vyplňovat počty, ale systém měl výchozí hodnoty.

**FR007 – Editace šablony**

Jako admin potřebuji upravovat šablony, abych mohl měnit počty potřebných uživatelských rolí.

**FR008 – Přidělování směn**

Jako admin potřebuji přidělovat ostatním zaměstnancům směny na konkrétních představeních.

**FR009 – Odstranění uživatelského účtu**

Jako admin potřebuji odstraňovat uživatelské profily, abych zabránil přístupu do aplikace neplatnými uživateli.

**FR010 – Spočítání odpracovaných hodin**

Jako admin potřebuji pro všechny uživatele spočítat odpracované hodiny, podle odpracovaných směn za mnou vybrané období, abych podle toho mohl vyplácet peníze.

**FR011 – Zobrazení všech uživatelů**

Jako admin potřebuji zobrazit seznam všech uživatelů, abych měl přehled, jací uživatelé jsou v systému.

### Uživatel

Tyto požadavky se zaměřují na funkcionality aplikace, které bude používat běžný uživatel(zaměstnanec).

**FR011 – Přihlášení**

Jako uživatel se potřebuji přihlašovat do systému pomocí emailu a hesla.

**FR012 – Odhlášení**

Jako uživatel se potřebuji odhlásit ze systému po dokončení činnosti, abych zabránil přístupu do aplikace neoprávněnou osobou.

**FR013 – Zobrazení profilu uživatele**

Jako uživatel potřebuji zobrazovat detail profilu vybraného uživatele.

**FR014 – Editace uživatelského účtu**

Jako uživatel potřebuji možnost změnit údaje svého účtu (jméno, příjmení, kontaktní email, heslo).

**FR015 – Zobrazení přihlášených směn**

Jako uživatel si potřebuji zobrazit seznam směn, na které jsem přihlášen, abych věděl, kdy mám jít do práce.

**FR016 – Přihlášení na směnu**
Jako uživatel se potřebuji přihlašovat na vypsané směny, abych zajistil, že nepůjde někdo jiný.
**FR017 – Odhlášení ze směny**
Jako uživatel potřebuji možnost odhlásit ze směny, aby mě mohl někdo zastoupit, když nebudu schopen v daném termínu směnu odpracovat.
**FR018 – Přihlášení na směnu jako náhradník**
Jako uživatel potřebuji možnost přihlásit se na již zaplněnou směnu jako náhradník, abych byl automaticky na směnu přihlášen, když se z ní někdo odhlásí
**FR019 – Zobrazení všech směn**
Jako uživatel potřebuji zobrazit všechny vypsané směny ve formátu kalendáře, abych se mohl přihlásit na mně vyhovující.
**FR020 – Emailové/SMS/Push notifikace zaměstnancům**
Jako zaměstnanec potřebuji, aby mi chodili notifikace o přidělení, změně nebo zrušení směny.
**FR021 – Zobrazit historii směn**
Jako uživatel potřebuji zobrazit historii směn ve formě kalendáře abych mohl dohledat proběhlé směny.

### 2.3.2 Nefunkční požadavky

**NFR01 - Dostupnost přes mobilní aplikaci**
Systém bude dostupný přes mobilní aplikaci pro platformu Android.
**NFR02 - Uživatelská práva**
Uživatelé budou mít přístup k různým funkcionalitám na základě své role.

## 2.4 Případy užití

Případy užití vyjadřují, kdo bude jakým způsobem používat IT systém. Cílem modelování případů užití je určit hranice systému. Aktéři, neboli účastníci, jsou vně systému a přímo ho ovlivňují nebo používají. Jednotlivý účastník je většinou reálná osoba nebo častěji role. Případy užití definují, co bude systém umožňovat. Každý případ užití popisuje jednu systémem podporovanou aktivitu obvykle jednoho účastníka v jednu chvíli a je iniciován účastníkem. Následující případy užití jsou rozděleny do dvou skupin podle typu aktéra, který je iniciuje. [5]

### 2.4.1 Každý uživatel

Tyto případy užití může iniciovat každý uživatel systému.
**UC01 - Přihlásit se**
Uživatel se zadáním svého emailu a hesla přihlásí do aplikace.
Scénář:

1. Systém zobrazí přihlašovací formulář.
2. Uživatel vyplní své údaje.
3. IF Uživatel klikne na `Přihlásit`.
4. THEN Systém zvaliduje a porovná zadané údaje s údaji v DB.
   4.1. IF Validace v pořádku.
   4.2. THEN Systém přihlásí uživatele a přesměruje hlavní stránku.
   4.3. ELSE Systém vypíše chybovou hlášku a `GOTO 2`.

**UC02 - Odhlásit se**
Uživatel se odhlásí ze systému.
Scénář:

1. Uživatel klikne na `Odhlásit se`.
2. Systém odhlásí uživatele a přesměruje na přihlašovací obrazovku.

**UC03 - Zobrazit detail účtu**
Systém zobrazí stránku s detailem uživatelského účtu.
Scénář:

1. Systém zobrazí detail uživatelského účtu.

**UC04 - Upravit účet**
Uživatel změní údaje účtu.
Scénář:

1. Systém zobrazí formulář pro úpravu uživatelských údajů. předvyplněný údaji z DB, kromě hesla.
2. Uživatel změní vybrané údaje.
3. IF Uživatel klikne na `Uložit změny`.
   3.1. THEN Systém zvaliduje změněné údaje - email ve správném formátu a unikátní, jména bez zvláštních znaků, hesla se shodují a neobsahují zvláštní znaky, telefon je číslo ve správném formátu.
   3.2. IF Validace v pořádku
   3.3. THEN Systém uloží změny a přesměruje na detail účtu.
   3.4. ELSE GOTO 2.
4. IF Uživatel klikne na `Zrušit`.
5. THEN Systém zahodí změny a přesměruje na detail účtu.

**UC05 - Zobrazit všechny směny**
Systém zobrazí přehled vypsaných směn.
Scénář:

1. Uživatel klikne v menu na `Směny`.
2. Systém zobrazí přehled všech vypsaných směn.

**UC06 - Zobrazit přihlášené směny**
Systém zobrazí směny, na které je uživatel přihlášený.
Scénář:

1. Uživatel klikne na `Moje směny`.
2. Systém zobrazí přehled směn, na který je uživatel přihlášený.

**UC07 - Zobrazit detail směny**
Systém zobrazí detaily o vybrané směně.
Scénář:

1. Uživatel klikne v přehledu na vybranou směnu.
2. Systém zobrazí detaily o vybrané směně.

**UC08 - Přihlásit se na směnu**
Admin vytvoří nový uživatelský účet.
Scénář:

1. Uživatel vybere jakou ze svých pracovní rolí chce na směně obsadit.
2. Uživatel klikne na `Přihlásit se na směnu`.
3. Systém zkontroluje, že pozice ještě není obsazena a přihlásí uživatele na směnu.

#### UC09 - Odhlásit se ze směny

Admin vytvoří nový uživatelský účet.
Scénář:

1. Uživatel v detailu směny klikne na `Odhlásit ze směny`.
2. Systém zobrazí dialogové okno `Opravdu zrušit směnu`.
3. IF Uživatel klikne na `Ano`.
4. THEN Systém odhlásí uživatele ze směny.
5. IF Uživatel klikne na `Ne`.
6. THEN Systém zavře dialogové okno a neprovede žádnou změnu.

### 2.4.2 Administrátor

Tyto případy užití může iniciovat pouze uživatel s administrátorskými právy.

#### UC10 - Vytvořit uživatelskou roli

Admin vytvoří novou uživatelskou roli.
Scénář:

1. Systém zobrazí formulář pro vytvoření nové role.
2. Uživatel vyplní údaje.
3. IF Uživatel klikne na `Vytvořit`.
4. THEN Systém zvaliduje zadané údaje (název role musí být unikátní).
   4.1. IF Údaje jsou validní.
   4.2. THEN Systém vytvoří novou roli a přesměruje na hlavní stránku.
   4.3. ELSE Systém zvýrazní chybná pole a `GOTO 2`.
5. IF Uživatel klikne na `Zrušit`.
6. THEN Systém zahodí údaje a přesměruje na hlavní stránku.

#### UC11 - Vytvořit uživatelský účet

Admin vytvoří nový uživatelský účet.
Scénář:

1. Systém zobrazí formulář pro vytvoření uživatelského účtu.
2. Uživatel vyplní své údaje.
3. IF Uživatel klikne na `Vytvořit`.
4. THEN Systém zvaliduje zadané údaje (email musí být unikátní a ve správném formátu, jméno a příjmení nesmí obsahovat speciální znaky).
   4.1. IF údaje jsou validní.
   4.2. THEN Systém vytvoří nový účet a přesměruje na stránku detailu účtu.
   4.3. ELSE Systém označí chybně vyplněná pole a `GOTO 2`.
5. IF Uživatel klikne na `Zrušit`.
6. THEN Systém zobrazí dialogové okno `Opravdu zrušit registraci`.
7. IF Uživatel klikne na `Ano`.
8. THEN Systém zahodí údaje a přesměruje na domovskou stránku.
9. IF Uživatel klikne na `Ne`.
10. THEN `GOTO 2`.

#### UC12 - Zobrazit všechny uživatele

Systém zobrazí seznam všech uživatelů.
Scénář:

1. Uživatel klikne na `Seznam uživatelů`.
2. Systém zobrazí seznam všech uživatelů.

#### UC13 - Odstranit uživatelský účet

Admin odstraní uživatelský účet.
Scénář:

1. Uživatel v seznamu účtů klikne na `Odstranit`.
2. Systém zobrazí dialogové okno `Opravdu si přejete účet odstranit`.
3. IF Uživatel klikne na `Ano`.
4. THEN Systém smaže vybraný účet a zavře dialog.
5. IF Uživatel klikne na `Ne`.
6. THEN Systém neprovádí změnu a pouze zavře dialog.

#### UC14 - Vytvořit směnu

Admin vytvoří novou směnu.
Scénář:

1. Systém zobrazí formulář pro vytvoření směny.
2. Uživatel zvolí a případně upraví šablonu pro nějakou připravenou akci, nebo zadá hodnoty ručně.
3. IF Uživatel klikne na `Vytvořit`
4. THEN Systém vytvoří novou směnu.
5. IF Uživatel klikne na `Zrušit`
6. Systém zahodí změny.

#### UC15 - Upravit směnu

Admin upraví již vytvořenou směnu a systém o tom notifikuje dotčené uživatele.
Scénář:

1. Systém zobrazí formulář pro úpravu směny s předvyplněnými současnými údaji.
2. Uživatel změní vybrané údaje.
3. IF Uživatel klikne na `Ok`
4. THEN Systém zvaliduje údaje.
   4.1. IF Validace OK
   4.2. THEN Systém uloží úpravy a notifikuje dotčené uživatele o změnách.
   4.3. ELSE Systém zobrazí chybovou hlášku.
5. IF Uživatel klikne na `Zrušit`
6. THEN Systém zahodí změny a přesměruje na detail směny.

#### UC16 - Zrušit směnu

Admin zruší směnu a systém o tom notifikuje dotčené uživatele.
Scénář:

1. Uživatel klikne na `Zrušit směnu`.
2. Systém zobrazí dialog `Opravdu zrušit směnu`.
3. IF Uživatel klikne na `Ano`.
4. THEN Systém zruší směnu a notifikuje o tom dotčené uživatele.
5. ELSE Systém neprovede změny.

#### UC17 - Přiřadit směnu

Admin přiřadí ke směně vybraného uživatele.
Scénář:

1. Uživatel v detailu směny klikne na `Přiřadit zaměstnance`.
2. Systém zobrazí seznam uživatelů, filtrovatelný podle role a jména.
3. Uživatel vybere zaměstnance, které chce na směnu přiřadit.
4. IF Uživatel klikne na `Přiřadit`.
5. THEN Systém přiřadí vybrané uživatele na směnu a odešle jim o tom notifikace.
6. IF Uživatel klikne na `Zrušit`.
7. THEN Systém zavře seznam a neprovede žádnou změnu.

#### UC18 - Vytvořit šablonu akce

Admin vytvoří novou šablonu akce, podle které může později vytvářet směny.
Scénář:

1. Systém zobrazí formulář pro vytvoření nového templatu akce.
2. Uživatel vyplní typ akce, počty potřebných uživatelských rolí a dobu trvání.
3. IF Uživatel klikne na `Vytvořit`.
4. THEN Systém zvaliduje zadané údaje (doba trvání musí být celé kladné číslo, typ akce nesmí obsahovat zvláštní znaky)
5. IF Validace OK
6. THEN Systém uloží nový template akce.
7. ELSE Systém vypíše chybovou hlášku a označí chybně vyplněná pole. `GOTO 2`.

#### UC19 - Upravit šablonu akce

Admin upraví existující šablonu akce.
Scénář:

1. Systém zobrazí formulář pro úpravu templatu s předvyplněnými původními parametry.
2. Uživatel upraví požadované parametry.
3. IF Uživatel klikne na `Uložit změny`.
   3.1. THEN Systém zvaliduje zadané údaje (doba trvání musí být celé kladné číslo, typ akce nesmí obsahovat zvláštní znaky)
   3.2. IF Validace OK
   3.3. THEN Systém uloží změny
   3.4. ELSE Systém vypíše chybovou hlášku a označí chybně vyplněná pole. GOTO 2.
4. IF Uživatel klikne na `Zrušit`.
5. THEN Systém zahodí změny a zavře formulář.

#### UC20 - Zveřejnit směnu

Admin zveřejní vytvořenou směnu, aby se na ni mohli uživatelé přihlašovat.
Scénář:

1. Uživatel v detailu směny klikne na `Přidat do kalendáře`.
2. Systém přidá směnu do kalendáře a rozešle oznámení o nové směně všem uživatelům.

#### UC21 - Spočítat odpracované hodiny

Systém spočítá odpracované hodiny každého uživatele za vybrané období.
Scénář:

1. Systém zobrazí formulář pro zvolení období a příp. zaměstnance.
2. Uživatel zvolí časové období, za jaké chce spočítat odpracované hodiny a může vybrat všechny nebo pouze konkrétní zaměstnance, pro které chce hodiny spočítat.
3. IF Uživatel klikne na `Spočítat`
4. THEN Systém zobrazí tabulku zaměstnanců a jejich odpracovaných hodin za dané období.

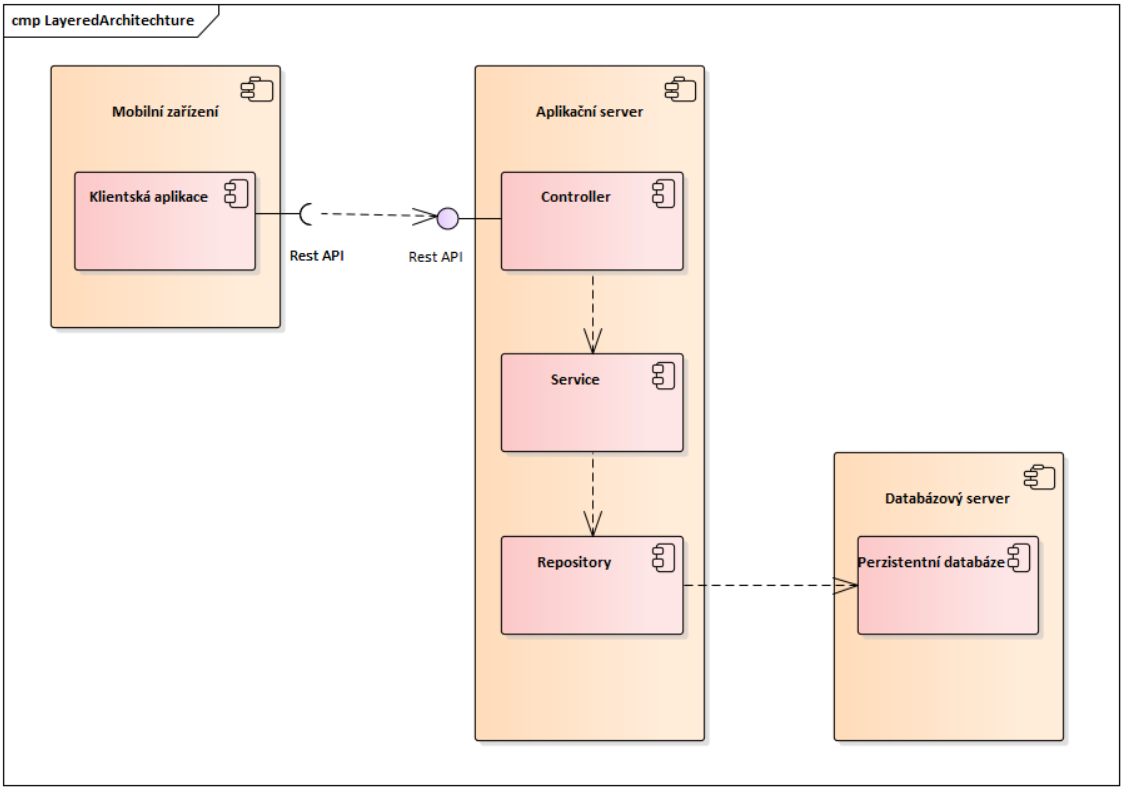
5. IF Uživatel klikne na `Zrušit`
6. THEN Systém přesměruje uživatele na předchozí stránku.

# Kapitola 3
# Návrh systému

V této kapitole je pospána zvolená architektura a datový model s popisem entit a jejich vzájemných vztahů. Dále je zde uveden návrh rozdělení obrazovek uživatelského rozhraní.

## 3.1 Architektura

Pro systém jsem vybral vrstevnatou architekturu, za využití poznatků z předmětů Softwarové inženýrství a Enterprise architektury. Princip této architektury spočívá v rozdělení odpovědnosti do více vrstev, kdy každá vrstva může provolávat vrstvu nižší. Každá vrstva se pak stará pouze o část logiky a tyto části aplikace je pak snazší měnit a upravovat. Rozdělení do vrstev souvisí i s fyzickým rozdělením, kdy každá část aplikace typicky běží na jiném stroji. Navrhovaná aplikace používá tři základní vrstvy: prezentační vrstva (klientská aplikace), aplikační vrstva (aplikační server) a perzistentní vrstva (databáze).




**Obrázek 3.1.** Vrstevnatá architektura aplikace

### 3.1.1 Prezentační vrstva

Prezentační vrstvu systému tvoří klientská mobilní aplikace pro platformu Android, kterou jsem zvolil, protože zařízení na této platformě jsou velmi běžná a dostupná.

Mobilní aplikace komunikuje se serverem pomocí REST API. Server požadavky posílané na jednotlivé endpointy zpracovává a propaguje do dalších vrstev, aby byly obslouženy. Před zpracováním požadavku server provádí autentizaci a autorizaci.

### 3.1.2 Aplikační vrstva

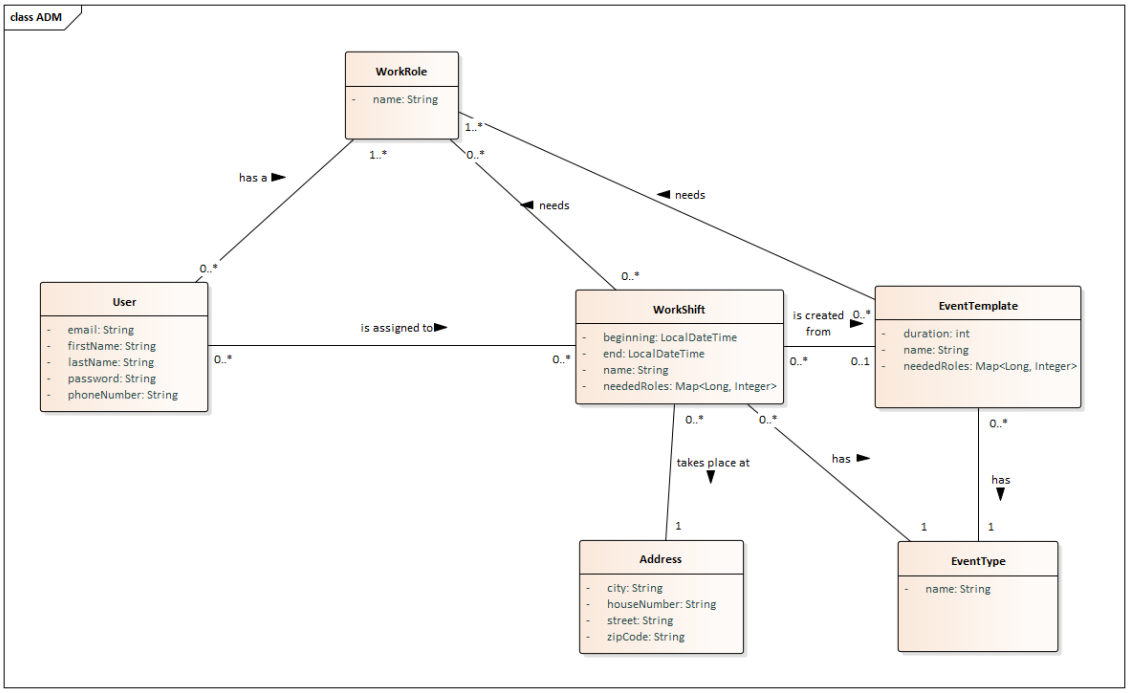
Do aplikační vrstvy patří samotný aplikační server, který zajišťuje rozhraní pro komunikaci s prezentační vrstvou, zpracovává požadavky od klientů a provádí většinu aplikační logiky. Server také zajišťuje autentizaci uživatelů a přístup k datům uloženým v perzistentní databázi.

### 3.1.3 Perzistentní vrstva

Perzistentní vrstva má na starosti ukládání dat, tvoří ji tedy databázový server s relační databází, který poskytuje rozhraní pro manipulaci s uloženými daty.

## 3.2 Datový model

Doménový model tříd popisuje jednotlivé entity a vztahy mezi nimi.




**Obrázek 3.2.** Doménový model tříd

### 3.2.1 Entity

Entita reprezentuje objekt, nebo jeho část, reálného světa. Každou entitu popisujeme názvem a sadou atributů s přiřazeným datovým typem.

**User - Uživatel**

Entita User reprezentuje uživatele systému (zaměstnance divadla).

**Atributy:**

- **email** - emailová adresa uživatele
- **firstName** - křestní jméno uživatele
- **lastName** - příjmení uživatele
- **password** - heslo k přihlášení
- **phoneNumber** - telefonní číslo uživatele

**WorkRole - Pracovní role**
Každý uživatel má libovolný počet pracovních rolí, na základě kterých mu jsou zobrazovány pracovní směny.
**Atributy:**

- **name** - název pracovní role

**WorkShift - Pracovní směna**
Tato entita reprezentuje jednu pracovní směnu, která má definované počty potřebných pracovníků různých rolí. Uživatelé se pak na tyto směny mohou hlásit, dle svých možností a pracovních rolí.
**Atributy:**

- **beginning** - datum a čas začátku směny
- **end** - datum a čas konce směny
- **name** - název směny
- **neededRoles** - požadované počty pracovníků dané role

**Address - Adresa**
Každá pracovní směna má přiřazenou adresu, na které se směna bude konat.
**Atributy:**

- **city** - město
- **houseNumber** - číslo popisné
- **street** - ulice
- **zipCode** - poštovní směrovací číslo (PSČ)

**EventTemplate - Šablona akce**
Šablona akce slouží k jednoduššímu vytváření opakujících se směn pro stejnou nebo podobnou událost.
**Atributy:**

- **duration** - doba trvání v minutách
- **name** - název události
- **neededRoles** - požadované počty pracovníků dané role

**EventType - Typ události**
Doplňková entita upřesňující typ události.
**Atributy:**

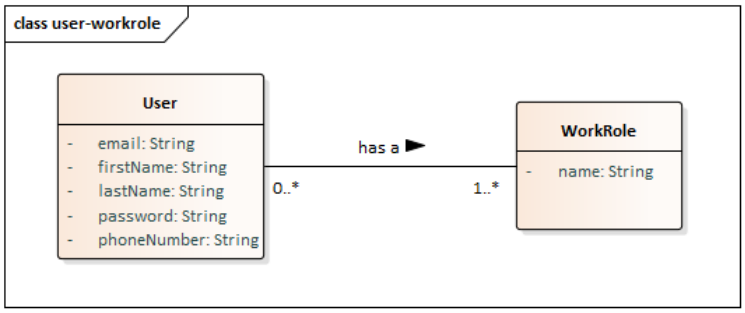
- **name** - název typu události

### 3.2.2 Vazby mezi entitami

Vazby mezi entitami představují logický vztah, upřesňující vzájemné závislosti.

**has a (User, WorkRole)**
Uživatel má libovolný počet pracovních rolí. Například může být zároveň uvaděč a

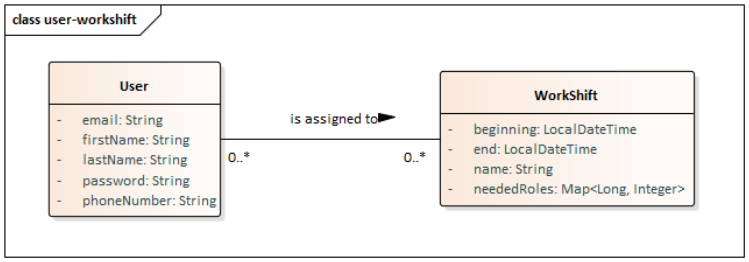
osvětlovač.




**Obrázek 3.3.** Vztah `has a` mezi uživatelem a pracovní rolí

### is assigned to (User, WorkShift)

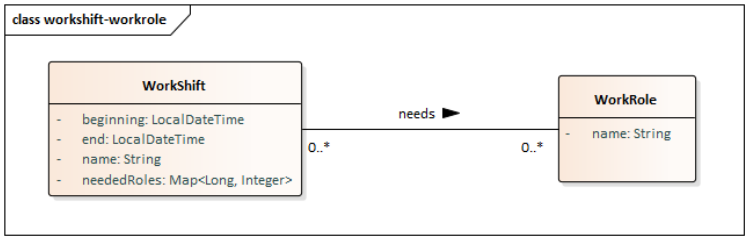
Uživatel je přihlášen nebo přiřazen na libovolný počet směn.




**Obrázek 3.4.** Vztah `is assigned to` mezi uživatelem a pracovní směnou

### needs (WorkShift, WorkRole)

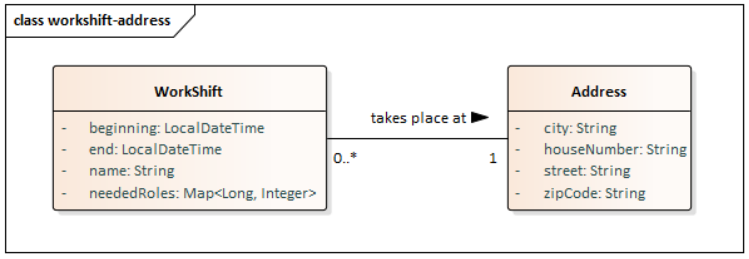
Pro směnu je třeba různý počet pracovních rolí v různých počtech.




**Obrázek 3.5.** Vztah `needs` mezi pracovní směnou a pracovní rolí.

### takes place at (WorkShift, Address)

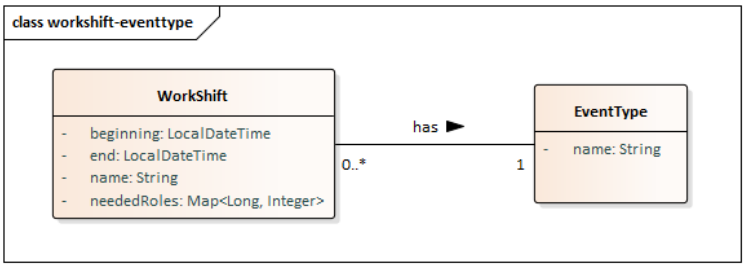
Směna se koná na nějaké adrese.




**Obrázek 3.6.** Vztah `takes place at` mezi pracovní směnou a adresou

### has (WorkShift, EventType)

Pracovní směna je nějakého typu, například představení nebo zkouška.




**Obrázek 3.7.** Vztah `has` mezi pracovní směnou a typem události

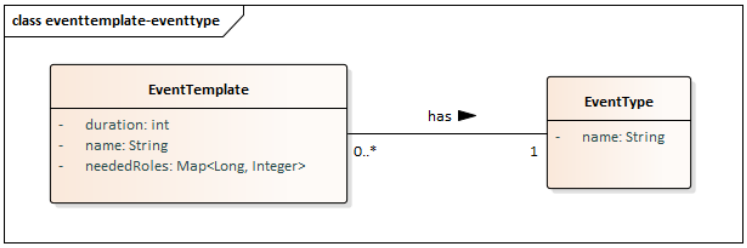
### is created from (WorkShift, EventTemplate)

Pracovní směna může být vytvořena z šablony.


**Obrázek 3.8.** Vztah `is created from` mezi pracovní směnou a šablonou události

### has (EventTemplate, EventType)

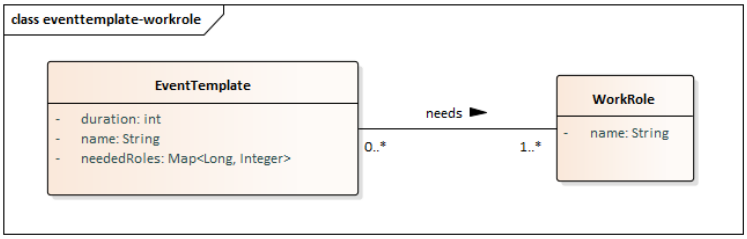
Událost, pro kterou je vytvořená šablona má nějaký typ.




**Obrázek 3.9.** Vztah `has` mezi šablonou události a typem události

### needs (EventTemplate, WorkRole)

Šablona události má předdefinované potřebné počty různých pracovních rolí.




**Obrázek 3.10.** Vztah `needs` mezi šablonou události a pracovní rolí

## 3.3 Uživatelské rozhraní

Uživatelské rozhraní aplikace musí být jednoduché a přehledné, aby se snadno používalo a uživatel se snadno zorientoval. Tlačítka by měla mít zejména slovní popis a jednotlivé

obrazovky zobrazují a nabízejí funkcionality, které spolu souvisí.
Základní rozdělení obrazovek:

- **Menu** - rozcestník zobrazený po přihlášení
- **Přihlášení** - přihlašovací formulář
- **Registrace** - registrační formulář
- **Přehled směn** - seznam dostupných směn a možnost přihlášení na směnu, případně vytváření a mazání směn
- **Přihlášené směny** - seznam uživatelem přihlášených směn a možnost odhlášení ze směny
- **Účet** - údaje o uživatelově účtu a možnost úprav
- **Šablony** - správa šablon pro vytváření směn
- **Statistiky** - počítání odpracovaných hodin a případné další funkcionality pro vedoucí zaměstnance

# Kapitola 4
# Implementace

V této kapitole jsou uvedeny použité technologie pro vytvoření aplikace a důvody pro jejich výběr.

## 4.1 Použité nástroje a technologie

Na základě návrhu a svých zkušeností jsem pro vývoj zvolil jazyk Java, jak pro serverovou tak klientskou část. Klientská část je mobilní aplikace pro platformu Android, pro jejíž vývoj jsem uvažoval ještě jazyk Kotlin, ale nakonec jsem od něj upustil, protože s ním nemám žádné zkušenosti. Java a Kotlin patří mezi nejpoužívanější jazyky pro vývoj Android aplikací, společně s C#, Python a C++. [6]

### 4.1.1 Java

Java je objektově orientovaný programovací jazyk. Zdrojový kód je psán do souborů `.java`. Ty jsou poté kompilovány do takzvaného `bytecode`, který je interpretován pomocí Java Virtual Machine (JVM) a je tedy nezávislý na platformě, k běhu je potřeba pouze JVM.

### 4.1.2 Spring Boot

Spring je framework pro programovací jazyk Java, který usnadňuje vývoj enterprise aplikací v Javě. Uplatňuje zejména návrhové vzory `Dependency Injection` a `Inversion of Control`. Spring Boot navíc jako nadstavba Springu obsahuje Tomcat server, který za nás vytvoří a nastaví. [7]

### 4.1.3 Maven

Maven je nástroj, který slouží ke správě, řízení a sestavování aplikací nad platformou Java. Při jeho využití nezávisí kompilace a sestavení programu na konkrétním IDE, protože potřebné informace jsou zapsány ve speciálních souborech `pom.xml` (POM = project object model). Tyto soubory se nachází v kořenovém adresáři projektu, kde lze spouštět příkazy mvn s parametry pro načtení odpovídajícího souboru a provedení zadané akce. Důležitou funkcí nástroje Maven je, že řeší závislosti, tedy není nutné ruční kopírování knihoven a jejich umisťování do classpath. [8]

### 4.1.4 Gradle

Gradle je automatizovaný nástroj pro sestavování, balíčkování, testování a nasazování aplikací podobně jako Maven. Je výchozím nástrojem pro sestavování Android aplikací vyvíjených v prostředí Android Studio.

### 4.1.5 PostgreSQL

PostgreSQL je open-source objektově-relační databázový systém, který se snaží přizpůsobovat standardům SQL. Funguje na všech hlavních operačních systémech a splňuje ACID vlastnosti transakčního zpracování. [9]

### 4.1.6 Android Studio

Android Studio je vývojové prostředí pro vytváření aplikací pro platformu Android. Použil jsem ho zejména kvůli integrovanému emulátoru, ve kterém lze snadno spouštět a manuálně testovat vyvíjenou aplikaci. A další výhodou tohoto prostředí je interaktivní grafické zobrazování `.xml` souborů, které se používají pro navrhování obrazovek.

## 4.2 Konfigurace

Všechna rozšíření a závislosti musí být definovaná v souboru `pom.xml` pro nástroj Maven v serverové aplikaci a `build.gradle` pro Gradle v mobilní aplikaci. Na základě těchto konfiguračních souborů se při buildu aplikace stáhnou, nebo načtou z lokálního úložiště, potřebné knihovny pluginy a frameworky.

### 4.2.1 Serverová konfigurace

Pro konfiguraci serverové aplikace jsou důležité konfigurační soubory frameworku Spring označené anotací `@Configuration`. Framework Spring pak vytváří Beany, neboli funkční objekty, podle konfigurací v tomto souboru anotovaných `@Bean`. Také scanuje ostatní soubory a pokud v nich narazí na některou z anotací `@Controller`, `@Service` nebo `@Repository` také z nich vytvoří Beany. Tímto je přesunuta odpovědnost za vznik vazeb na framework, neboli je využit princip Iversion of Controll (IoC) a vazby mezi jednotlivými objekty jsou volnější. Tyto vytvořené Beany pak framework vkládá do jiných tříd, kde jsou potřeba, tedy využívá principu Dependency Injection (DI). [10]
Konfigurační soubor `application.properties` slouží k definici různých vlastností aplikace. V tomto případě je zde definován port, na kterém server poslouchá a připojení k databázi.

```
#Port
server.port = 8080

#PostgreSQL
spring.jpa.database=postgresql
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/theatreDB
spring.datasource.username=postgres
spring.datasource.password=
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto=update
```

### 4.2.2 Konfigurace uživatelské aplikace

V mobilní aplikaci je důležitý soubor `AndroidManifest.xml`, kde se konfiguruje, která aktivita je spuštěna při spuštění aplikace.

## 4.3 Zabezpečení

O zabezpečení se stará Spring Security framework, který zajišťuje autentizaci a autorizaci. Konfigurační soubor je označen anotací `@EnableWebSecurity`, která povoluje zabezpečení. Tento soubor pak obsahuje třídu `WebSecurityConfigurerAdapter`, pro

kterou je důležitá metoda `configure(HttpSecurity http)`, ve které jsou pravidla pro zabezpečení.

```
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true, securedEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
...
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
        .antMatchers("/login", "/rest/user/register",
                "/rest/user/unique",
                "/rest/workRole")
                .permitAll()
        .anyRequest().authenticated()
        .and().exceptionHandling()
        .authenticationEntryPoint(
            new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED))
        .and().headers().frameOptions().sameOrigin()
        .and().authenticationProvider(authenticationProvider)
        .csrf().disable()
        .formLogin().successHandler(authenticationSuccessHandler)
        .failureHandler(authenticationFailureHandler)
        .loginProcessingUrl(SecurityConstants.SECURITY_CHECK_URI)
        .usernameParameter(SecurityConstants.USERNAME_PARAM)
        .passwordParameter(SecurityConstants.PASSWORD_PARAM)
        .and()
        .logout().invalidateHttpSession(true)
        .deleteCookies(COOKIES_TO_DESTROY)
        .logoutUrl(SecurityConstants.LOGOUT_URI)
        .logoutSuccessHandler(logoutSuccessHandler);
}
}
```

## 4.4 Perzistentní vrstva

Pro ukládání dat aplikace jsem zvolil objektově-relační databázi PostgreSQL, ke které přistupuje serverová aplikace pomocí JDBC ovladače. Třídy reprezentující objekty v databázi jsou anotovány `@Entity` a jsou mapovány pomocí frameworku Hibernate. Pro přístup k objektům mapovaným do relační databáze je použito JpaRepository, které spravuje framework Spring a nabízí základní CRUD operace. Výhodou použití tohoto rozhraní je snadná tvorba dalších operací nad daty, protože stačí pouze ve správném formátu zadat název a parametry požadované metody a framework ji vytvoří za nás. Naopak někdy tento přístup může být nevýhodný, protože programátor nemá kontrolu nad tím, jak je metoda implementována. V takovém případě je lepší použít Data Access Object (DAO), kde má programátor nad všemi metodami kontrolu. Pro posílání dat po síti je použit Data Transfer Object (DTO), který zajišťuje, aby přenášené objekty neměly cyklické odkazy a vynechává atributy, které mají být skryty.

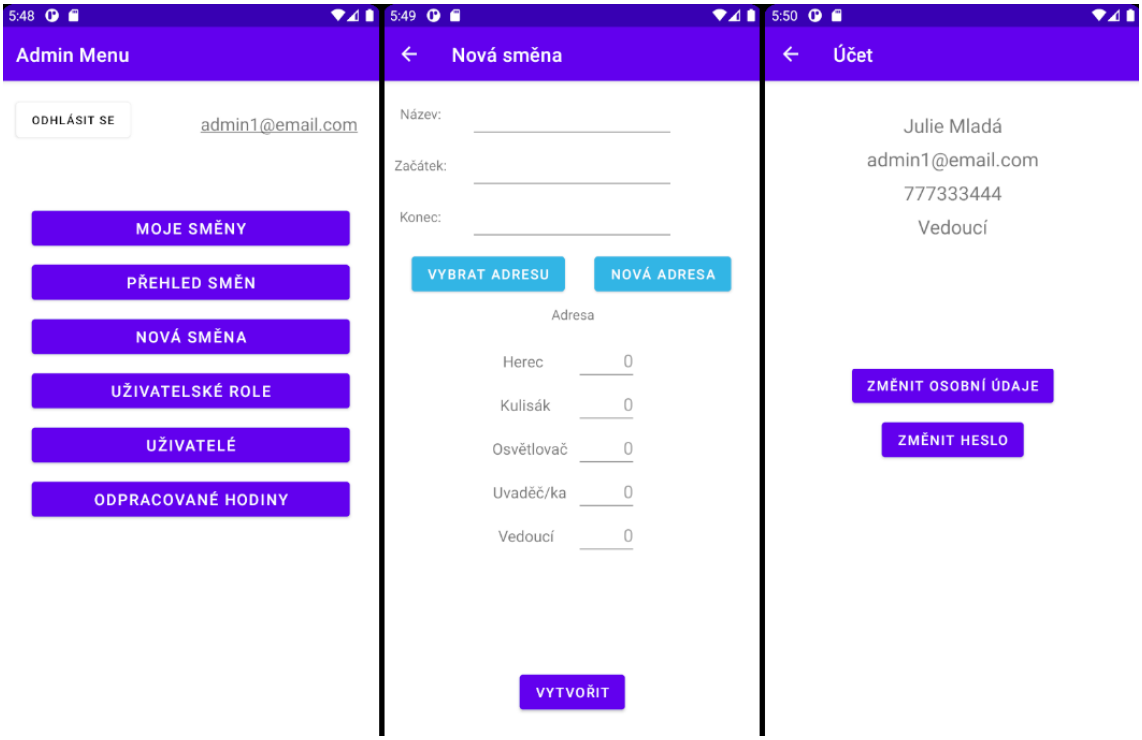
## 4.5 Aplikační vrstva

Aplikační vrstva se stará o aplikační logiku, přístup k perzistentní vrstvě a obsluhuje příchozí požadavky ze sítě. O většinu logiky a přístup k perzistentní vrstvě se starají třídy anotované `@Service`. Server obsluhuje HTTP uživateli posílané požadavky, které odchytávají jednotlivé kontrolery, tedy třídy anotované `@RestController`. V těchto třídách je pak pomocí anotací `@GetMapping` a `@PostMapping` zajištěno mapování na správné metody, které požadavek zpracují. Objekty posílané v těle požadavků a odpovědí jsou reprezentovány JSON objekty, které framework automaticky konvertuje.

## 4.6 Prezentační vrstva - Uživatelské rozhraní

Prezentační vrstvu zajišťuje mobilní aplikace pro platformu Android verze 5.0 a novější, která s aplikačním serverem komunikuje přes REST API posíláním HTTP požadavků. Nabízí uživateli funkcionality podle jeho práv a aby byla co nejplynulejší, všechny požadavky na aplikační server jsou zpracovávány asynchronně.

### 4.6.1 Ukázka grafického rozhraní aplikace




**Obrázek 4.1.** zleva: Administrátorkské menu, Formulář pro vytvoření směny, Uživatelský účet

# Kapitola 5
# Testování

## 5.1 Manuální testování

Aplikace byla manuálně testována pomocí vytvořeného uživatelského rozhraní a aplikací Postman. Postman je nástroj pro návrh a interakci s HTTP API a jedno z jeho použití je právě testování. Umožňuje vytváření a odesílání HTTP požadavků, které je možné uložit pro opakované použití. Tímto způsobem jsem testoval většinu požadavků, které server umí zpracovat a kontroloval odpovědi. Cílem tohoto testování bylo zjistit, zda server správně přijímá a případně i odmítá požadavky a jestli je v odpovědích obsaženo vše, co je potřeba. Manuální testování uživatelského rozhraní se týkalo především korektního zobrazování dat a případných chybových hlášek.

## 5.2 Unit testy

Unit testy slouží k odhalení chyb na úrovni kódu. Principem unit testu je srovnání výsledku volání metody nebo procedury s očekávaným výsledkem. K programu je tedy vytvořena další část kódu, která kontroluje funkční část kódu. Tyto testy je možné spouštět opakovaně po každé změně programu. K testování aplikace byl požit framework JUnit, který usnadňuje přípravu i spouštění testů. [11]
Příklad jednoduchého JUnit testu, který testuje přidávání pracovníka na směnu:

```
    @Before
    public void setUp()
    {
        workRole = new WorkRole();
        workRole.setName("TestRole");
        workRole.setId(2L);
        workRole2 = new WorkRole();
        workRole2.setName("TestRole2");
        workRole2.setId(3L);
        user = new User();
        user.setId(1L);
        workShift = new WorkShift();
        Map<Long, Integer> neededRoles = new HashMap<>();
        neededRoles.put(2L, 2);
        workShift.setNeededRoles(neededRoles);
    }

    @Test
    public void addWorkerToWorkShift_userWithNeededRole
    _addsUserToListAndUpdatesNeededRoles()
```

```
    {
        //prepare
        user.setWorkRoles(List.of(workRole));

        //act
        workShift.addWorkerWithGivenRole(user, workRole);

        //assert
        assertEquals(1, workShift.getNeededRoles().get(2L));
        assertTrue(workShift.getWorkers().contains(user));
    }
```

Anotace `@Before` označuje metodu, která bude zavolána před každým testem a `@Test` označuje testovací metodu.

## 5.3 Uživatelské testování

Testování se zúčastnili dva zástupci Divadla U Váňů z Poděbrad a jeden zástupce divadelního spolku Vojan Libice nad Cidlinou a každý dostal některou z následujících skupin úkolů. Úkoly se každý pokusil splnit a následně jsem s nimi prodiskutoval, jak se jim s aplikací pracovalo a k čemu měli připomínky.
Uživatelé dostali následující úkoly pro otestování aplikace jako administrátor:

### Admin 1

1. Přihlaste se do aplikace emailem `admin1@email.com` a heslem `admin1`.
2. Vytvořte novou uživatelskou roli `Úklid`.
3. Přidejte novou roli `Úklid` uživateli `Novotná Jana`.
4. Vytvořte novou směnu s následujícími parametry:
   a) Název: `Směna 1`.
   b) Začátek: `21.5.2021, 18:00`.
   c) Konec: `21.5.2021, 21:30`.
   d) Na nově vytvořené adrese:
      - Ulice: `Zkušební`.
      - Č.P.: `123`.
      - Město: `Praha`.
      - PSČ: `160 00`.
   e) Pro 3 herce, 1 kulisáka a 1 osvětlovače.
5. Najděte nově přidanou směnu v seznamu směn a přiřaďte na pozici herec uživatele `Jan Novák`.

### Admin 2

1. Přihlaste se do aplikace emailem `admin1@email.com` a heslem `admin1`.
2. Přidělte uživateli `Vladimír Svoboda` administrátorská práva.
3. Přiřaďte na směnu `Zkouška 1` do role `Kulisák` uživatele `Václav Kučera`.
4. Smažte směnu `Představení 4`.
5. Zobrazte si odpracované hodiny uživatelů v období 1.1.2021 – 30.5.2021.

A dále tyto úkoly pro testování jako obyčejný uživatel:

**Uživatel 1**

1. Vytvořte si uživatelský účet se svým emailem, heslem a telefonem a vyberte si uživatelskou roli `Herec`.
2. Zobrazte si svůj profil.
3. Přidejte mezi své pracovní role roli `Osvětlovač`.
4. Zobrazte si dostupné směny a přihlaste se na směnu `Představení 2`, která se koná 30.5.2021 od 9:00, jako `Osvětlovač`.
5. Odhlaste se z aplikace.

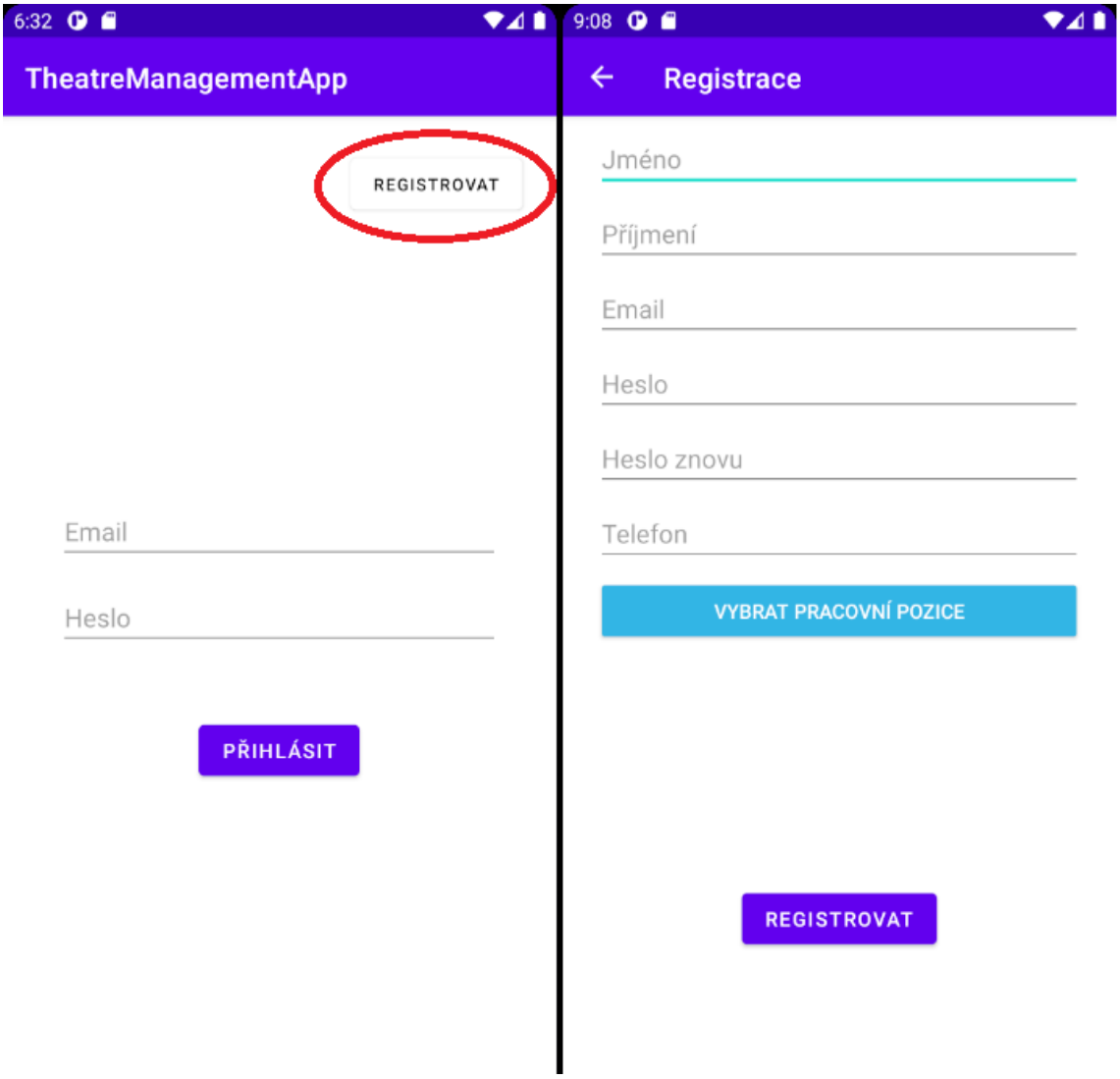
**Uživatel 2**

1. Přihlaste se do aplikace emailem `user1@email.com` a heslem `user1`.
2. Zobrazte si své přihlášené směny.
3. Zjistěte, na jaké adrese se koná směna, na kterou jste přihlášeni 21.6.2021 od 16:00, a jakou pozici zde zastáváte.
4. Odhlaste se ze směny z bodu 4.
5. Změňte heslo na `123456`.
6. Odhlaste se z aplikace a přihlaste novým heslem.

### 5.3.1 Výsledky testování

Mobilní aplikace běžela na fyzickém zařízení Samsung Galaxy S4 (GT-I9506) s verzí Androidu 5.0.1. Všichni zúčastnění zvládli práci s aplikací velmi dobře. Nejčastější problém byl při úkolu vytvoření novéhu uživatelského účtu, kdy začala většina vyplňováním přihlašovacího formuláře, než si všimli tlačítka `Registrace` pro zobrazení registračního formuláře (viz. Obrázek 5.1). Bylo by tedy vhodné přehodnotit design úvodní obrazovky. Na druhou stranu uživatel se typicky registruje jen jednou a proto tuto chybu nebude moci opakovat.

Jinak uživatelé hodnotili aplikaci jako přehlednou a intuitivní, avšak některé funkcionality by mohli být přístupné z více míst. Například přidělení uživatele na směnu je možné z detailu směny, ale ne z detailu uživatele. Poslední připomínkou bylo nepohodlí při složitějších administrativních aktivitách jako je vytváření nové směny, kdy by dali přednost práci na počítači. To je rozhodně jeden z primárních cílů dalšího vývoje zmíněný v závěrečné kapitole.




**Obrázek 5.1.** Přihlašovací formulář, Registrační formulář

# Kapitola 6
# Závěr

Cílem práce bylo navrhnout a implementovat aplikaci pro správu činnosti divadla. Byl proveden průzkum fungování divadla a na jeho základě byla vytvořena analýza, specifikující požadavky a případy užití. Podle provedené analýzy byl vytvořen návrh architektury aplikace a datový model. Serverová část aplikace byla implementována v jazyku Java za použití nástroje Maven a frameworku Spring. Serverová aplikace komunikuje s PostgreSQL databází pomocí JDBC ovladače. Jako grafické uživatelské rozhraní byla vytvořena aplikace pro platformu Android, implementovaná také v jazyce Java s použitím frameworku Gradle.
Analýza byla vytvořena nezávisle před implementací, a proto vytvořená aplikace nesplňuje úplně všechny specifikované požadavky, ale pouze ty nejdůležitější pro základní fungování aplikace.
Při vypracování jsem uplatnil znalosti z různých předmětů a také jsem využíval návody a dokumentace na internetu, zejména při tvorbě aplikace pro platformu Android, se kterou jsem neměl předchozí zkušenost. S výběrem použitých technologií jsem byl poměrně spokojený, protože jsou všechny často používané bylo velmi snadné najít potřebné návody a dokumentace a s jazykem Java mám ze svého studia největší zkušenosti.

## 6.1 Budoucí vývoj aplikace

Aby byla aplikace co nejlépe použitelná bylo by vhodné ji rozšířit o frontend pro iOS a webové prohlížeče. Pro příjemnější používání by měly být doimplementovány všechny požadavky a případně přidané další funkcionality jako je například filtrování. K usnadnění provozu divadla by také přispělo rozdělení pravomocí systému do více skupin, nebo přiřazování pravomocí konkrétním uživatelům. Z hlediska zabezpečení aplikace by mělo být přidáno šifrování komunikace po síti.

### 6.1.1 Push notifikace

Push notifikace jsou nejobvyklejší způsob, kterým dnes mobilní aplikace upozorňují uživatele. Zde jsou důležité zejména pro informace o změnách času konání směn, případně jejich zrušení.

### 6.1.2 Integrace s iCal

Aplikace by podporovala automatické přidávání naplánovaných směn do vybraného kalendáře podporující tento standard.

# Literatura


[1] *Sling* [online]. [vid. 21. 1. 2020]. Dostupné na `https://app.getsling.com/`.

[2] *Theatron.eu* [online]. [vid. 27. 4. 2021]. Dostupné na `https://www.theatron.eu/about`.

[3] *Propared* [online]. [vid. 27. 4. 2021]. Dostupné na `https://www.propared.com/`.

[4] Zlámalová, Anna. *IS pro správu činnosti divadla*. Karlovo nám. 13, 121 35 Praha 2: České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů, 2018. Bakalářská práce.

[5] Komárek, Ing. Martin. *Softwarové inženýrství, Modelování případů užití* [přednáška]. [vid. 27. 4. 2021]. Dostupné na `https://moodle.fel.cvut.cz/pluginfile.php/251079/course/section/46823/P%C5%99edn%C3%A1%C5%A1ka3_PripadyUzitiKomplet_SIN_ZS_2020.pptx`.

[6] *Top 5 Android app development languages 2020* [online]. [vid. 15. 5. 2021]. Dostupné na `https://www.appdevelopmentpros.com/blog/top-5-android-app-development-languages-2020`.

[7] *Itnetwork.cz* [online]. [vid. 7. 4. 2021]. Dostupné na `https://www.itnetwork.cz/java/spring-boot/uvod-do-spring-boot-frameworku-pro-javu`.

[8] Hordějčuk, Vojtěch. *Maven* [online]. [vid. 7. 4. 2021]. Dostupné na `http://voho.eu/wiki/maven/`.

[9] *PostgreSQL* [online]. [vid. 7. 4. 2021]. Dostupné na `https://www.postgresql.org/about/`.

[10] Kunčar, Petr. *Spring konfigurace* [online]. [vid. 28. 4. 2021]. Dostupné na `https://www.itnetwork.cz/java/spring-boot/spring-ioc-kontejner`.

[11] Ing. Miroslav Bureš, Ph.D. doc. *Testování software, Unit testy* [přednáška]. [vid. 27. 4. 2021]. Dostupné na `https://moodle.fel.cvut.cz/pluginfile.php/235111/mod_resource/content/1/TS1_prednaska_6_7_8.pdf`.

[12] Ledvinka, Ing. Martin. *EAR E-shop* [online]. [vid. 9. 3. 2021]. Dostupné na `https://gitlab.fel.cvut.cz/ear/b201-eshop`.

[13] Baeldung. *Spring security* [online]. [vid. 16. 3. 2021]. Dostupné na `https://www.baeldung.com/security-spring`.

[14] *Retrofit* [online]. [vid. 2. 3. 2021]. Dostupné na `https://square.github.io/retrofit/`.

[15] *Okhttp* [online]. [vid. 2. 3. 2021]. Dostupné na `https://square.github.io/okhttp`.

# Příloha A
## Obsah přiložených souborů

```
system-pro-divadla
|
+---src
|   +---main
|   |   +---java - kód serverové čáasti aplikace
|   |   |
|   |   \---resources - konfigurační soubory serverové části
|   |
|   \---test
|
\---pom.xml - konfigurační soubor pro Maven

TheatreManagementApp
    +---app
    |   |   build.gradle - konfigurační soubor pro Gradle
    |   |
    |   \---src
    |       +---main
    |       |   |   AndroidManifest.xml
    |       |   |
    |       |   +---java - kód mobilní klientské aplikace
    |       |   |
    |       |   \---res - .xml soubory s layouty obrazovek
    |       |
    |       \---test
```