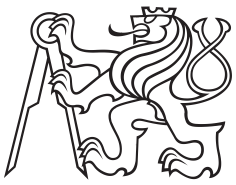
Bakalárská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

# Systém pro správu divadla

Ramir Azizov

Vedoucí: Ing. Božena Mannová, Ph.D.
Obor: Softwarové inženýrství a technologie
Zaměření: Enterprise systémy
Květen 2024

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Azizov** Jméno: **Ramir** Osobní číslo: **501315**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Enterprise systémy**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Systém pro správu divadla**

Název bakalářské práce anglicky:

**Theater management system**

Pokyny pro vypracování:

Cílem bakalářské práce je dosáhnout modernizace a efektivity v divadelním prostředí prostřednictvím návrhu a implementace divadelního správcovského systému. Aplikace umožní centralizovanou a dynamickou správu herců, představenía směn. Výsledný divadelní správcovský systém bude podporovat efektivní spolupráci v divadelním prostředí, umožní jednoduchou správu herců, techniků a dalších účastníků a zlepší celkovou organizaci a plánování představení.
Postup řešení:
1. Seznamte se s problematikou správy divadla..
2. Proveďte analýzu vám dostupných podobných existujících aplikací, proveďte jejich porovnání a vyhodnocení.
3. Na základě provedené analýzy navrhněte základní funkcionality navrhované aplikace.
4. Zvolte architekturu aplikace a vyberte nejvhodnější technologie pro implementaci. Výběr technologií zdůvodněte.
5. Aplikaci implementujte a otestujte včetně uživatelských testů.
6. Zhodnoťte výsledky a navrhněte případné další funkcionality nebo jiná zlepšení.

Seznam doporučené literatury:

[1] Theatron, Dostupné na https://www.theatron.eu/
[2] Komárek, Ing.Martin., Specifikace požadavků (přednáška). Dostupné na https://moodle.fel.cvut.cz/pluginfile.php/338210/mod_resource/content/4/SpecifikacePozadavku_2019_Prednaska3.pdf
[3] IBM, Rational Software Architect. Dostupné na https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

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: **16.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

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

Chci poděkovat všem přátelům, rodině a učitelům za podporu během bakalárské práci. Bez vaší víry a pomoci by to nebylo možné. Zvláštní díky paní Boženě Mannové za vedení a cenné rady.


# Prohlášení

Prohlašuji, že jsem samostatně vytvořil prezentovanou práci a že jsem v souladu s Metodickým pokynem ohledně dodržování etických principů při přípravě vysokoškolských závěrečných prací uváděl všechny použité informační zdroje.

V Praze dne 24.05.2024

# Abstrakt

Práce je zaměřena na vytvoření informačního systému pro správy divadla. V rámci této praci byla provedena specifikace požadavků a návrh implementace systému



**Klíčová slova:** divadlo, informační systém, Java, Spring Boot, mikroservesy, Docker, Junit, Mockito

**Vedoucí:** Ing. Božena Mannová, Ph.D.



# Abstract

The project is aimed at creating an information system for theater management. Within this work, requirements specification and system implementation design have been conducted.



**Keywords:** theater, information system, Java, Spring Boot, microservices, Docker, Junit, Mockito

# Obsah

# Obrázky

# Kapitola 1
# Úvod

Tato práce se věnuje návrhu a vytvoření uživatelsky přívětivé aplikace s cílem usnadnit provoz a organizaci divadla. Aplikace je koncipována pro interní potřeby divadelního provozu a nesoustředí se na aspekty financí či veřejné prezentace divadla.

## 1.1 Cíl prace

Cílem této práce je dosáhnout modernizace a efektivity v divadelním prostředí prostřednictvím návrhu a implementace divadelního správcovského systému. Aplikace umožní centralizovanou a dynamickou správu herců, představení a směn. Výsledný divadelní správcovský systém bude podporovat efektivní spolupráci v divadelním prostředí, umožní jednoduchou správu herců, techniků a dalších účastníků a zlepší celkovou organizaci a plánování představení.

## 1.2 Motivace

Moderní divadelní prostředí čelí mnoha výzvám při správě herců, představení a dalších aspektů produkce. Tradiční metody mohou být časově náročné a náchylné k chybám. Motivací pro návrh a implementaci divadelního správcovského systému je moje osobní zkušenost s podobným systémem od společnosti Shameless[1], který jsem používal jako uživatel. Tato zkušenost mě přesvědčila o potřebě modernizace v divadelním prostředí a o výhodách efektivního správcovského nástroje.

# Kapitola 2
# Analýza řešení

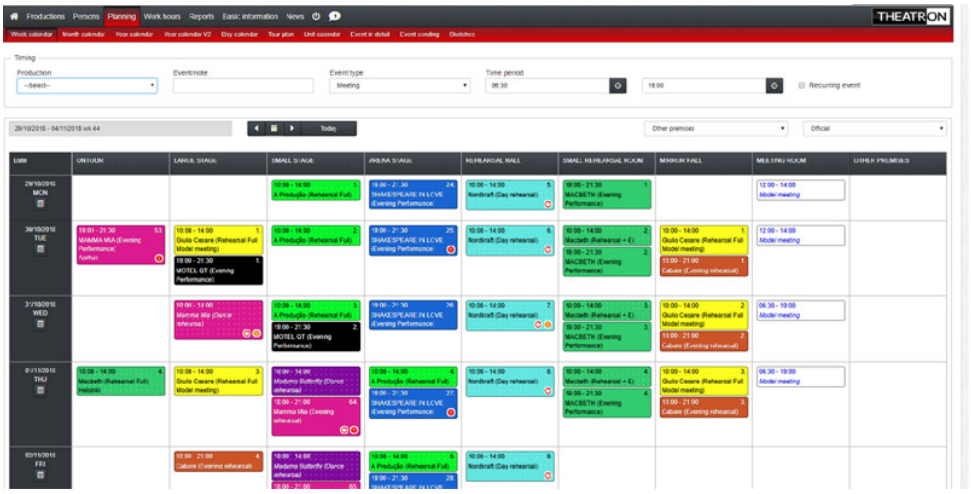
V této kapitole provedu detailní analýzu potřebnou k výběru optimálního řešení pro divadelní správcovský systém. Zaměřím se na identifikaci specifických požadavků divadelní produkce a následně zhodnotím možné řešení s důrazem na vytvoření vlastního systému.

## 2.1 Existující řešení

V této části bude popsaný jíž existující systémy na správy divadel. Byly vybrany dva takovych systemu: Theatron a Propared. Prozkoumám jejich klíčové vlastnosti, funkcionality a provedu srovnání s cílem poskytnout ucelený pohled na dostupné možnosti v oblasti správy divadelních aktivit.

### 2.1.1 Theatron

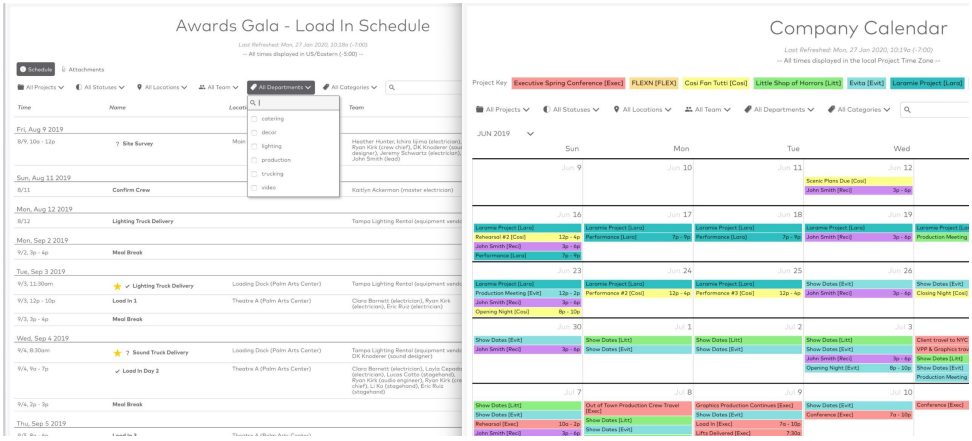
Theatron je komplexní systém pro efektivní správu divadelních produkcí, umožňující plánování místností, rolí, obsazení a sledování různorodých událostí. Zvládá organizaci uměleckého i technického personálu, včetně podrobného plánování vystoupení, zkoušek, schůzek, návštěv a dalších aktivit. S využitím centralizovaného plánovacího kalendáře usnadňuje sledování a správu různých aspektů divadelního provozu.[2]




**Obrázek 2.1:** Ukázka aplikace Theatron

### 2.1.2 Propared

Propared představuje výkonný nástroj pro plánování a správu divadelních produkčních operací. S jeho pomocí mohou divadelní organizace efektivně plánovat vystoupení, zkoušky, schůzky a další události. Aplikace nabízí přehledné prostředí pro plánování uměleckého a technického personálu, správu místností a událostí. S centralizovaným plánovacím kalendářem a flexibilními nástroji umožňuje Propared divadelnímu týmu snadnou koordinaci, organizaci a transparentní sledování všech aspektů produkčního procesu.[3]




**Obrázek 2.2:** Ukázka aplikace Propared

## 2.2 Sběr požadavků navrhovaného systému

Sběr požadavků je klíčovým krokem v procesu vývoje. Tento krok zajišťuje celkové porozumění potřebám uživatelů a dalším zainteresovaným stranám, což poskytuje pevný základ pro návrh a implementaci systému.[4]
Požadavky na systém jsou klasifikovány do dvou hlavních kategorií: funkční a nefunkční. Funkční požadavky popisují konkrétní operace a funkce, které by měl systém provádět, zatímco nefunkční požadavky se zaměřují na vlastnosti systému, jako je výkon, bezpečnost a dostupnost.[5]

### 2.2.1 Funkční požadavky

Funkční požadavky, označované také jako functional requirements (FR), jsou klíčovým prvkem definujícím všechny operace a schopnosti, které aplikace poskytne uživatelům.[5]

#### Admin

Funkční požadavky pro roli Admina jsou především orientovány na klíčové funkce, které bude tato role v systému vykonávat.

- **FR01 - Definice rolí:**
  Admin může vytvořit a definovat různé role, jako například Technik, Zvukař, Barman, Herec, atd.

- **FR02 - Vytvoření účtů pro uživatele:**
  Admin může vytvořit účty pro uživatele s přiřazením k jedné z rolí, například Technik, Zvukař, Barman, Herec.

- **FR03 - Odeslání odkazu pro registraci herců:**
  Admin může poslat odkazy pro registraci hercům s předdefinovanou rolí "herec"a usnadnit jim proces registrace.

- **FR04 - Editace role uživatelů:**
  Admin může editovat role již existujících uživatelů.

- **FR05 - Smazání uživatelského účtu:**
  Admin může smazat uživatelský účet.

- **FR06 - Vytvoření entit představení:**
  Admin může vytvořit a spravovat entitu představení, určit počet a role uživatelů potřebných k jeho realizaci.

- **FR07 - Editace představení:**
  Admin může editovat parametry již existujícího představení.

- **FR08 - Smazání představení:**
  Admin může smazat již existující představení.

- **FR09 - Nastavení kalendáře:**
  Admin může naplánovat představení do kalendáře.

#### Zaměstnanec divadla

Funkční požadavky pro roli zaměstnance divadla jsou navrženy s ohledem na činnosti, které bude tato role vykonávat v rámci systému

- **FR10 - Přihlášení do systému:**
  Zaměstnanec se může přihlásit do systému pomocí uživatelského jména a hesla.

- **FR11 - Odhlášení z systému:**
  Přihlášený zaměstnanec se může odhlásit ze systému.

- **FR12 - Zobrazení svého osobního účtu:**
  Přihlášený zaměstnanec se může zobrazit stránku svého osobního účtu.

- **FR13 - Editace svého osobního účtu:**
  Přihlášený zaměstnanec se může editovat parametry svého profilu.

- **FR14 - Přihlášení na směny:**
  Přihlášený zaměstnanec se může přihlásit na směnu.

- **FR15 - Odhlášení ze směny:**
  Přihlášený zaměstnanec se může odhlásit ze směny.

- **FR16 - Zobrazení přihlášené směny:**
  Přihlášený zaměstnanec si může zobrazit své přihlášené směny.

### 2.2.2 Nefunkční požadavky

Nefunkční požadavky určují kvalitativní charakteristiky a omezení systému, na rozdíl od funkčních požadavků, které se zaměřují na operace a schopnosti. Tato část dokumentace se věnuje aspektům, jako je výkon, bezpečnost, spolehlivost a další faktory ovlivňující uživatelský zážitek a celkovou efektivitu systému.[5]

- **NFR01 - Přístupnost prostřednictvím webového prohlížeče.**
  Systém bude poskytován jako webová aplikace, dostupná prostřednictvím webového prohlížeče. Uživatelé budou interagovat s aplikací, která bude provozována na serveru, přičemž se jedná o přístup z tzv. tenkého klienta, tedy prostřednictvím webového prohlížeče.

- **NFR02 - Přístup k funkcionalitám dle opravnění**
  Systém bude umožňovat přístup k určitým funkcím pouze uživatelům s odpovídajícími oprávněními. To zaručí, že pouze oprávnění uživatelé budou mít možnost využívat specifické funkce systému.

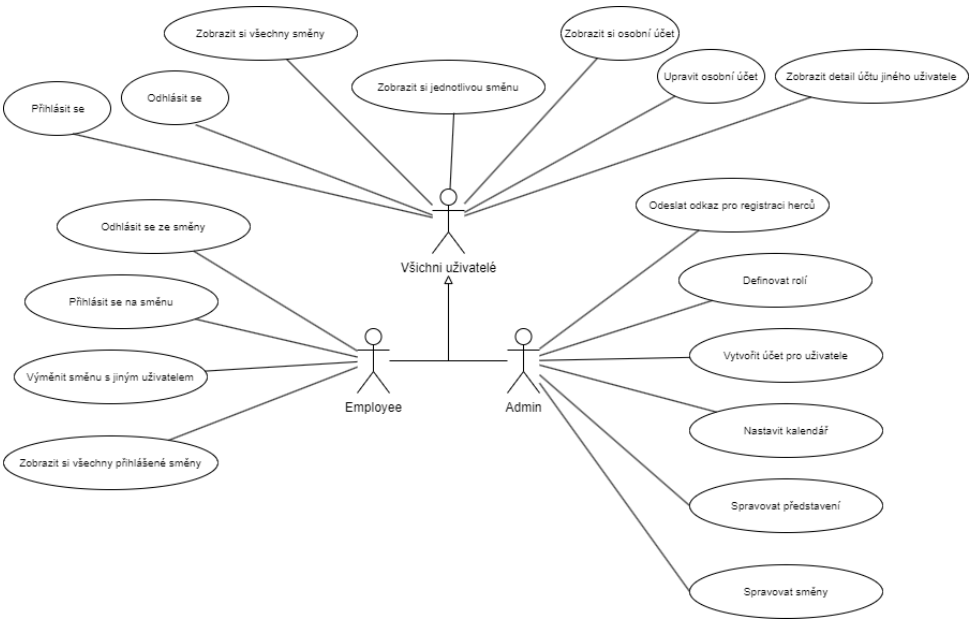
- **NFR03 - Výkonnost a odezva systému**
  Systém musí poskytovat rychlou odezvu a vysoký výkon, aby uživatelé mohli plynule interagovat s aplikací přes webový prohlížeč.

- **NFR04 - Škálovatelnost systému**
  Systém musí být schopen škálovat a efektivně zvládat nárůst uživatelů a dat bez výrazného snížení výkonu.

## 2.3 Případy užití

Případy užití jsou nástrojem pro modelování chování systému a slouží k zachycení požadavků na něj. Tyto případy popisují obecné funkce a rozsah systému, identifikují interakce mezi systémem a jeho uživateli (aktéry). Případy užití a aktéři popisují, jak systém reaguje na různé akce uživatelů, přičemž se zaměřují na vnější pohled na systém a nevěnují se interním operacím. Hlavním cílem je zachytit, co systém dělá a jak s ním uživatelé interagují, a to na vysoké úrovni abstrakce. [6]

### 2.3.1 Diagram případů užití




**Obrázek 2.3:** Use Case Diagram

### 2.3.2 Všichni uživatelé

Tyto požadavky se týkají jednotlivých funkcionalit, které mohou provádět všichni uživatelé systému.

**UC01 - Přihlásit se**
Uživatel se může přihlásit do systému pomocí uživatelského jména a hesla,

která mu pošle Admin.

**Scenář:**
1. Uživatel přistupuje ke stránce a systém mu zobrazí přihlašovací formulář.
2. Uživatel vyplní přihlašovací údaje (uživatelské jméno a heslo).
3. Pokud uživatel klikne na tlačítko "Přihlásit":
3.1. Systém provede validaci a porovná zadané údaje s údaji v databázi.
3.1.1. Pokud je validace v pořádku:
3.1.1.1. Systém přihlásí uživatele do systému.
3.1.1.2. Uživatele se přesměruje na hlavní stránku systému.
3.1.2. Pokud validace není v pořádku:
3.1.2.1. Systém zobrazí chybovou hlášku ohledně neplatných údajů.
3.1.2.2. Uživatel se vrátí na přihlašovací formulář a může znovu vyplnit údaje.

**UC02 - Odhlásit se**
Uživatel, který je přihlášený do systému, má možnost odhlásit se.

**Scenář:**
1. Uživatel klikne na tlačítko "Odhlásit":
2. Systém provede odhlášení uživatele ze systému.
3. Uživatele přesměruje na stránku s přihlašovacím formulářem.

**UC03 - Zobrazit si osobní účet**
Každý přihlášený uživatel má možnost zobrazit si stránku s informacemi o svém osobním účtu.

**Scenář:**
1. Uživatel klikne na tlačítko nebo odkaz pro zobrazení osobního účtu:
2. Systém načte a zobrazí informace o osobním účtu uživatele.
3. Uživatel může prohlížet a kontrolovat své osobní údaje, jako jsou jméno, příjmení, kontakt atd.

**UC04 - Upravit osobní účet**
Každý přihlášený uživatel má možnost upravit své osobní údaje na stránce s informacemi o osobním účtu.

**Scenář:**
1. Uživatel klikne na tlačítko nebo odkaz pro úpravu osobního účtu:
2. Systém načte formulář s aktuálními údaji o uživateli.
3. Uživatel může provést požadované změny v údajích, jako jsou jméno, příjmení, kontakt atd.
4. Pokud všechny změny jsou v pořádku:
4.1. Systém provede aktualizaci údajů v databázi a zobrazí potvrzení o provedených změnách.
4.2. Pokud nejsou:
4.2.1. Systém zobrazí chybovou hlášku ohledně neplatných údajů.
4.3.2. Uživatel se vrátí do bodu číslo 2.

**UC05 - Zobrazit detail účtu jiného uživatele**
Uživatel má možnost zobrazit detailní informace o účtu jiného uživatele v systému.

**Scenář:**
1. Uživatel klikne na tlačítko "Vyhledat uživatele"a zadá její uživatelské jméno.
2. Systém prověří existenci zvoleného uživatele v databázi.
2.1. Pokud uživatel existuje:
2.1.1.Systém načte a zobrazí detailní informace o účtu zvoleného uživatele.
2.2 Pokud uživatel neexistuje:
2.2.1. Systém informuje uživatele, že zvolený uživatel nebyl nalezen

**UC06 - Zobrazit si všechny směny**
Přihlášený uživatel má možnost zobrazit si přehled všech dostupných směn v systému.

**Scenář:**
1. Uživatel klikne na odkaz nebo tlačítko pro zobrazení všech směn:
2. Systém načte a zobrazí seznam všech aktuálních směn.

**UC07 - Zobrazit si kalendář**
Přihlášený uživatel má možnost zobrazit si kalendář s přehledem všech plánovaných představení v systému.

**Scenář:**
1.Uživatel klikne na odkaz nebo tlačítko pro zobrazení kalendáře:
2.Systém načte a zobrazí kalendář s označenými daty plánovaných představení na aktuální měsíc.

**UC08 - Zobrazit si jednotlivou směnu**
Přihlášený uživatel má možnost zobrazit si detaily jednotlivé směny v systému.

**Scenář:**
1. Pokud uživatel klikne na tlačítko pro zobrazení detailů jednotlivé směny:
2. Systém načte a zobrazí detaily vybrané směny.

### 2.3.3 Zaměstnanec divadla

Tyto požadavky se týkají specifických funkcionalit, které mohou provádět pouze zaměstnanci divadla.

**UC09 - Přihlásit se na směnu**
Zaměstnanec, který je přihlášený do systému, má možnost přihlásit se na konkrétní směnu.

**Scenář:**
1. Zaměstnanec klikne na odkaz nebo tlačítko pro přihlášení na směnu:
2. Systém zobrazí seznam dostupných směn, ze kterých si zaměstnanec může vybrat.
3. Zaměstnanec vybere konkrétní směnu, na kterou se chce přihlásit.
4. Systém ověří, zda jsou všechna potřebná místa na směně obsazena a zda odpovídá potřebná role zaměstnance s jeho aktuální rolí.
4.1. Pokud jsou všechna kritéria splněna:
4.1.1. Zaměstnanec je zapsán na vybranou směnu.
4.1.2 Systém aktualizuje seznam přihlášených zaměstnanců pro danou směnu.
4.2. Pokud některé kritérium není splněno:
4.2.1. Systém zobrazí chybovou hlášku informující zaměstnance o nemožnosti přihlášení na danou směnu.

**UC10 - Výměnit směnu s jiným uživatelem**
Zaměstnanec, který je přihlášený do systému, má možnost navrhnout výměnu směny s jiným uživatelem.

**Scenář:**
1. Zaměstnanec klikne na odkaz nebo tlačítko pro výměnu směny:
2. Systém zobrazí seznam jeho aktuálně přihlášených směn, které jsou možné pro výměnu.
2.1 Zaměstnanec vybere konkrétní směnu, kterou chce vyměnit.
3. Zaměstnanec vybere uživatele, se kterým chce provést výměnu směn.
3.1 Systém ověří, zda je vybraná směna k výměně a zda i druhý uživatel má směnu k výměně.
4.Pokud jsou všechna kritéria splněna:
4.1 Zaměstnanec navrhne výměnu směny s druhým uživatelem.
4.2 Systém odešle oznámení druhému uživateli s návrhem na výměnu směn.
5.Pokud některé kritérium není splněno:
5.1. Systém zobrazí chybovou hlášku informující zaměstnance o nemožnosti provedení výměny směn.

### 2.3.4 Admin

Tyto požadavky jsou zaměřeny na konkrétní funkcionality, které budou dostupné pouze administrátorovi. Admin bude mít pravomoci a možnosti, které mu umožní efektivně spravovat a řídit celý systém divadelního provozu.

**UC11 - Definovat rolí**
Správce divadla má možnost vytvořit a definovat různé role v systému.

**Scenář:**
1. Admin přistupuje k sekci definice rolí v systému.
2. Admin vybere možnost vytvoření nové role a zadá potřebné informace.
3. Systém potvrdí vytvoření nové role a zařadí ji mezi dostupné role v divadle.

**UC12 - Vytvořit účty pro uživatele**
Admin může vytvořit účty pro uživatele a přiřadit jim konkrétní role, jako jsou Technik, Zvukař, Barman, Herec apd.

**Scenář:**
1. Admin přistoupí ke sekci vytvoření účtů pro uživatele.
2. Admin zadá všechny potřebné informace, včetně přiřazení rolí (např. Technik, Zvukař, Barman, Herec atd.).
3. Systém provede validaci vstupních údajů.
3.1. Pokud jsou všechny údaje v pořádku:
3.1.1. Systém vytvoří nové účty pro uživatele.
3.2. Pokud validace není v pořádku:
3.2.1. Systém zobrazí chybovou hlášku informující admina o neplatných údajích.

**UC13 - Odeslat odkaz pro registraci herců**
Admin může poslat odkazy pro registraci hercům s předdefinovanou rolí "herec"a usnadnit jim proces registrace.

**Scenář:**
1. Správce divadla vybere možnost odeslat odkazy pro registraci hercům s předdefinovanou rolí "herec".
2. Systém generuje odkazy a automaticky je odesílá hercům.
3. Herci kliknou na odkazy a dokončí proces registrace s přiřazením role "herec".

**UC14 - Upravit role uživatelů**
Admin má pravomoc editovat role již existujících uživatelů v systému.

**Scenář:**
1. Admin vybere uživatele, kterému chce změnit role.
2. Admin klikne na tlačítko pro změnu role a vybere, kterou chce přiřadit.
3. Systém potvrdí provedenou změnu role uživatele.

**UC15 - Smazat uživatelský účet**
Admin může smazat uživatelský účet v případě potřeby.

**Scenář:**
1. Admin vyhledá uživatele, kterého chce smazat, a klikne na tlačítko pro zobrazení detailů účtu.

2. Admin klikne na tlačítko pro smazání uživatele.
3. Systém zobrazí adminovi okno s potvrzením, zda chce opravdu smazat uživatelský účet.
3.1. Pokud admin potvrdí smazání:
3.1.1. Systém odstraní uživatelský účet z databáze.
3.2. Pokud admin nepotvrdí smazání:
3.2.1. Systém přesměruje admina na stránku s detaily uživatele.

**UC16 - Vytvoření entit představení**
Admin má možnost vytvářet a spravovat entity představení, určit počet a role uživatelů potřebných k jeho realizaci.

**Scenář:**
1. Admin zobrazí všechna představení a klikne na tlačítko pro přidání nového představení.
2. Admin zadá potřebné informace o novém představení a určí počet a role uživatelů potřebných k jeho realizaci.
3. Systém provede validaci zadaných informací.
3.1. Pokud admin zadal všechno správně:
3.1.1. Systém potvrdí vytvoření entity představení.
3.2. Pokud admin špatně zadal parametry:
3.2.1. Systém zobrazí chybovou hlášku informující admina o nemožnosti provedení vytvoření představení.

**UC17 - Upravit představení**
Admin může editovat parametry již existujících představení.

**Scenář:**
1. Admin zobrazí všechna představení a vybere to, které chce upravit.
2. Admin klikne na tlačítko pro úpravu představení.
3. Admin provede potřebné úpravy informací.
3.1. Pokud všechno je zadáno správně:
3.1.1. Systém potvrdí provedení úprav představení.
3.2. Pokud admin zadal něco špatně:
3.2.1. Systém zobrazí chybovou hlášku a přesměruje admina na stránku s detaily představení.

**UC18 - Smazat představení**
Admin má možnost odstranit již existující představení z systému.

**Scenář:**
1. Admin zobrazí všechna představení a vybere to, které chce smazat.
2. Admin klikne na tlačítko pro smazání představení.
3. Systém zobrazí adminovi okno s potvrzením, zda chce opravdu smazat představení.
3.1. Pokud admin potvrdí smazání:
3.1.1. Systém provede odstranění představení z databáze.
3.1.2. Uživatel obdrží potvrzení o úspěšném smazání představení emailem.
3.2. Pokud admin odmítne smazat představení:
3.2.1. Systém přesměruje admina na stránku s detaily představení.

**UC19 - Nastavit kalendář**
Admin má pravomoc nastavit kalendář pro jednotlivá představení v divadle.

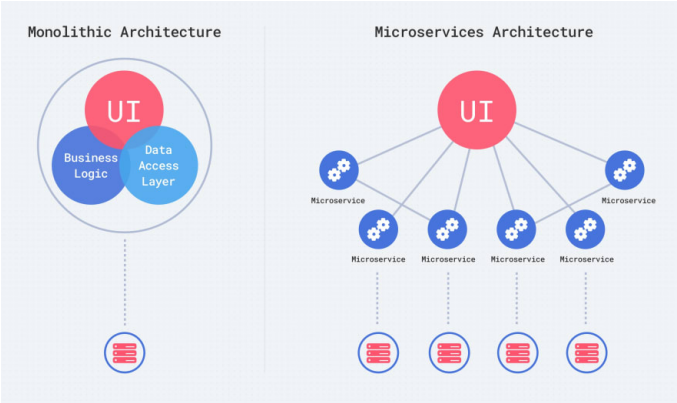
**Scenář:**
1. Admin přistoupí do sekce kalendáře a klikne na tlačítko "Nastavit kalendář".
2. Systém zobrazí formulář pro zadání informací o novém představení, včetně názvu, data, času a dalších relevantních údajů.
3. Admin vyplní formulář a potvrdí akci.
4. Systém ověří správnost zadaných údajů a aktualizuje kalendář s novým představením.
5. Adminovi se zobrazí notifikace o úspěšném naplánování představení.
6. Uživatel obdrží potvrzení o úspěšném naplánování představení do kalendáře divadla .

# Kapitola 3
# Návrh systému

V této kapitole jsou popsány návrh architektury systému, datový model a taky uživatelské rozhraní. Detailní analýza těchto klíčových prvků poskytne hlubší porozumění fungování celého systému a jeho schopností efektivně podporovat správu divadelních procesů.

## 3.1 Architektura systému

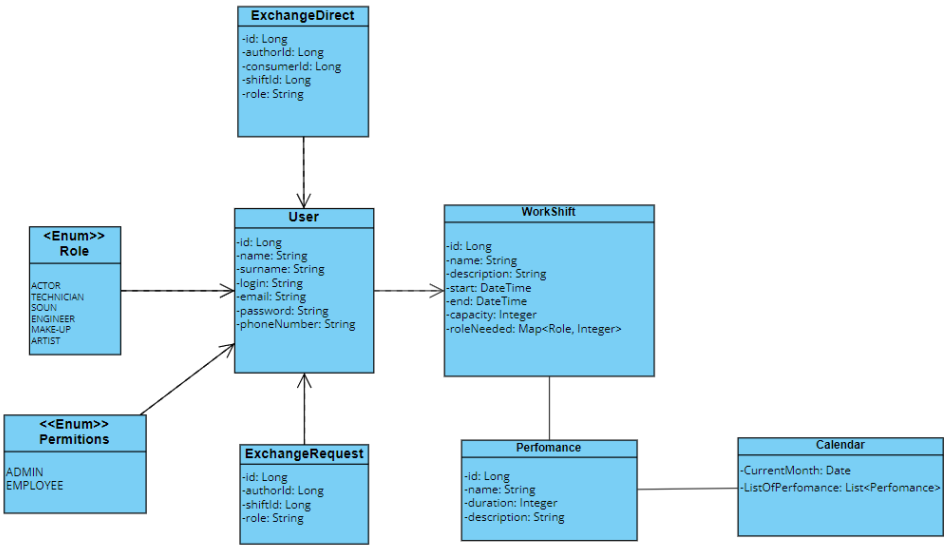
Vzhledem k povaze divadelního správcovského systému bych zvolil mikroservisy jako vhodnou architekturu pro projekt. Na rozdíl od monolitické architektury, která by spojovala všechny funkcionality do jednoho velkého celku, mikroservisy umožňují oddělenou správu a vývoj různých funkcionalit. Díky mikroservisům lze dosáhnout vyšší agility vývoje a škálovat jednotlivé služby podle potřeby. Tato architektura také umožňuje využívání různých technologií pro jednotlivé mikroslužby, což je výhodné při optimalizaci výkonu a údržby.[7]




**Obrázek 3.1:** Monolit architektura VS Mikroservisy.[8]

## 3.2 Datový model

Datový model systému je klíčovým prvkem při návrhu a implementaci, protože definuje strukturu a vztahy mezi různými datovými entitami.




**Obrázek 3.2:** Datový model

### 3.2.1 Entity

Entita je reprezentací konkrétního objektu nebo jeho části v reálném světě. Každá entita je charakterizována svým názvem a souborem atributů, které mají přiřazený specifický datový typ. Atributy popisují vlastnosti entity a obsahují informace, které jsou relevantní pro daný objekt nebo jeho část. Tímto způsobem je možné detailně popsat všechny prvky systému a usnadnit manipulaci s nimi v informačním systému.

**User**

- Reprezentuje uživatele systému.
- Může mít různé role, jako jsou Admin, Employee nebo Manager.
- Obsahuje informace jako jméno, příjmení, username, email, telefonní číslo

**Role**

- Reprezentuje jednotlive role uživatelů, které jsou nezbytné pro organizaci pracovních směn.

**Workshift**

- Reprezentuje konkrétní pracovní směnu v systému.
- Obsahuje informace jako název, popis, datum a čas začátku a konce, kapacitu a potřebné role.

**Performance**

- Reprezentuje samotné představení.
- Obsahuje název, delku konání a popis.

**ExchangeRequest**

- Reprezentuje objekt, který obsahuje ID autora a jednotlivé ID směny. Odpovídá za zpracování žádostí o výměnu směn, které mohou být zasílány všemi uživateli.

**ExchangeDirect**

- Reprezentuje objekt, který obsahuje ID autora, ID adresáta a jednotlivé ID směny. Odpovídá za přímou výměnu směn mezi konkrétními zaměstnanci na základě specifického požadavku uživatele.

**Permission**

- Reprezentuje objekt, který obsahuje role pro oprávnění, jako jsou "Admin"a "Employee". Slouží k řízení přístupu a práv uživatelů v systému na základě jejich rolí.

**Calendar**

- Poskytuje kalendářní funkce pro plánování a sledování pracovních směn, událostí a termínů.
- Může obsahovat informace o pracovních směnách, dovolené, svátcích a další.
- Obsahuje seznam všech naplánovaných představení.

## 3.3 Uživatelské rozhraní a prototypování

Uživatelské rozhraní je část systému, která umožňuje interakci mezi uživatelem a aplikací. To zahrnuje veškeré prvky, které uživatel vidí a s nimiž může pracovat, jako jsou obrazovky, dialogová okna, tlačítka, formuláře, navigační prvky atd. Cílem uživatelského rozhraní je poskytnout uživatelům intuitivní prostředí pro provádění operací a navigaci v aplikaci.[9]

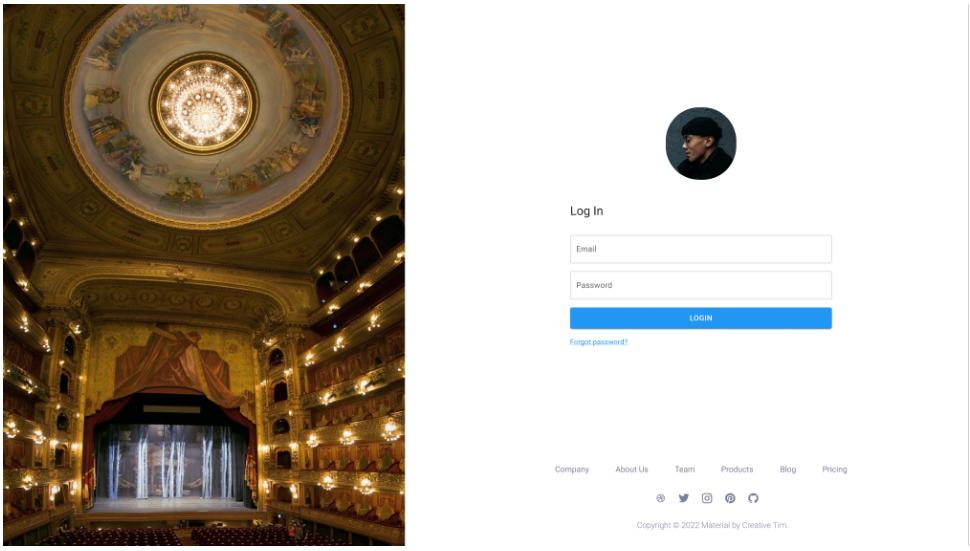
Pro tvorbu návrhů a prototypů jsem vybral nástroj Figma, protože ve většině svých projektů jsem jej používal. Figma je webový designový nástroj, který umožňuje vytváření interaktivních prototypů s různými průchody, stavby a animacemi. Na základě dohody s vedoucím projektu nebudu vytvářet kompletní frontend, ale zaměřím se na tvorbu prototypů.

Jako vzor jsem použil komponenty a prvky designového systému Material Design od společnosti Google, které jsou dostupné přímo v Figma prostřednictvím knihovny Material-UI[10]. Tato knihovna mi poskytla bohatou sadu předdefinovaných komponent a stylů v souladu s principy Material Design, což mi umožnilo efektivně vytvářet moderní a responzivní návrhy uživatelského rozhraní přímo v nástroji Figma.

### 3.3.1 Rozdělení obrazovek

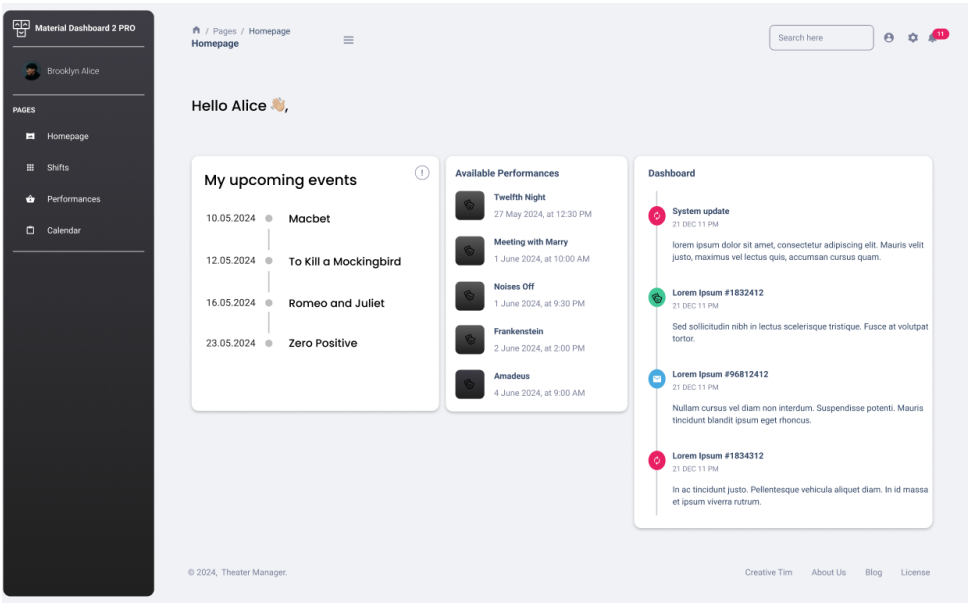
Rozdělení obrazovek vypadá nasledovně:

**Přihlášení**
Tato obrazovka umožňuje uživatelům přihlásit se do systému pomocí svého emailu a hesla.




**Obrázek 3.3:** Prototyp stránky přihlášení

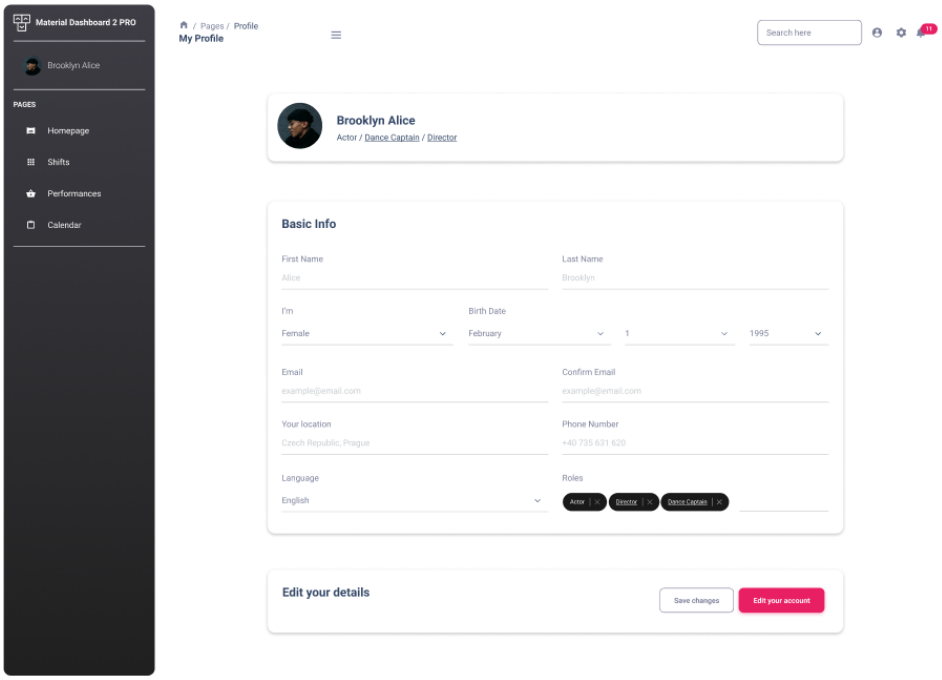
**Domovská stránka**
Domovská stránka slouží jako výchozí bod aplikace. Obsahuje přehledný a uživatelsky příjemný layout, který zobrazuje důležité informace a akce, například plánované směny, aktuální události nebo rychlé odkazy na často používané funkce.




**Obrázek 3.4:** Prototyp domovské stránky

### Profil uživatele

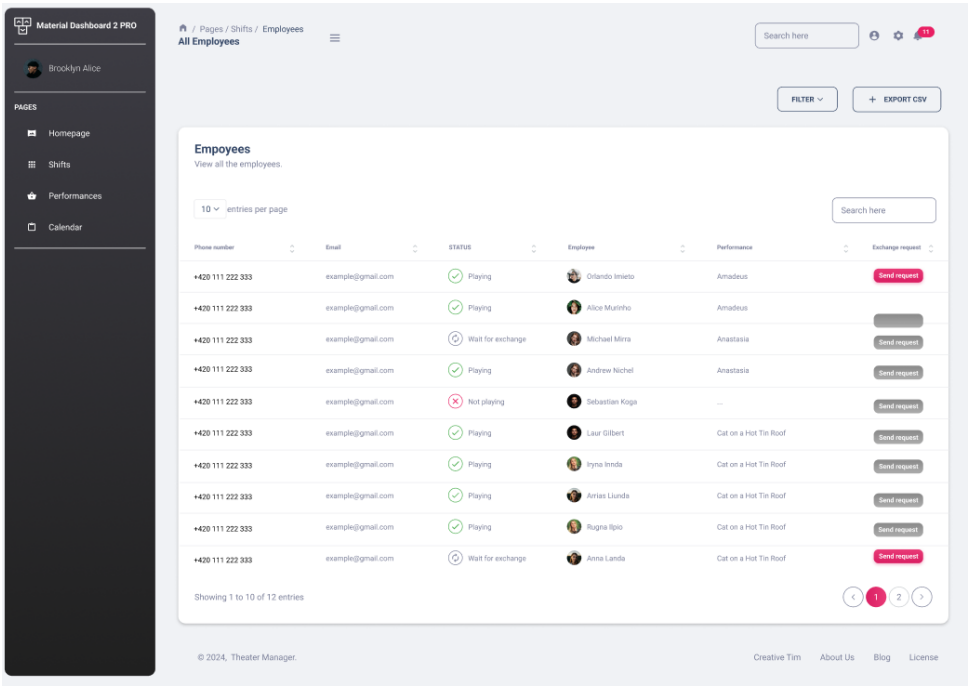
Tato obrazovka umožňuje uživatelům prohlížet a upravovat svůj profil. Zde mohou měnit svá osobní údaje, heslo nebo předvolby aplikace.




**Obrázek 3.5:** Prototyp stránky osobního profilu

### Seznam zaměstnanců

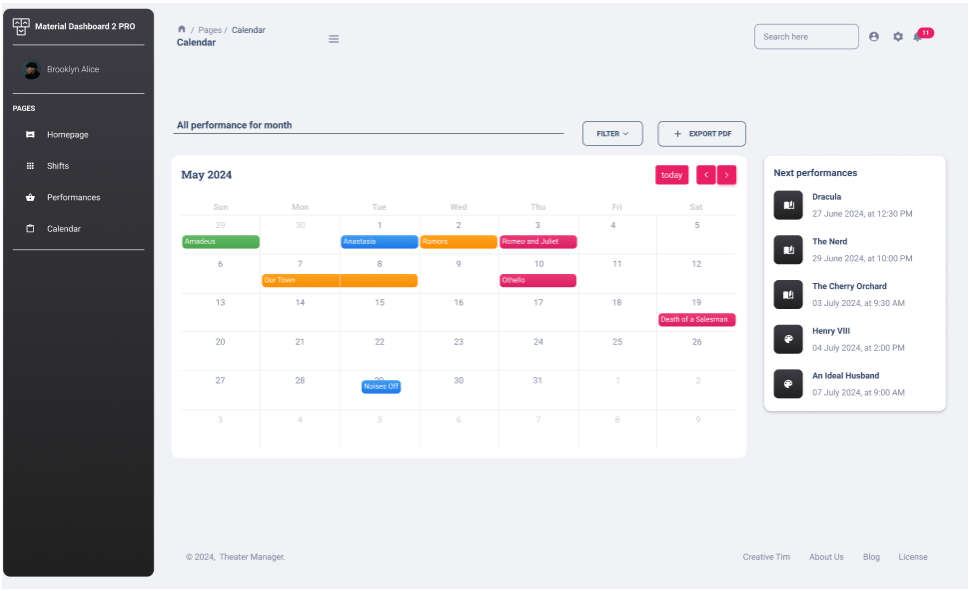
Na této obrazovce uživatelé mohou procházet seznam zaměstnanců a získat o nich podrobné informace, jako jsou jejich kontaktní údaje a přidělené role. Po vybrání konkrétního zaměstnance a zobrazení jeho profilu bude uživatelovi poskytnuta možnost poslat žádost o výměnu směny.




**Obrázek 3.6:** Prototyp stránky seznamu všech zaměstnanců

### Kalendář

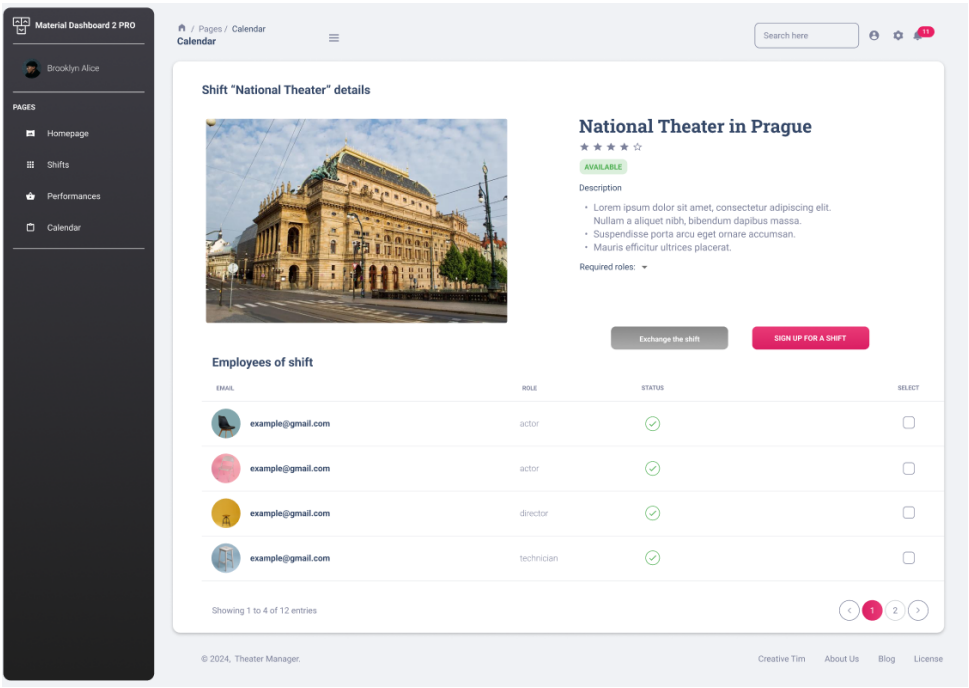
Tato obrazovka zobrazuje všechna představení divadla, která jsou naplánována. Uživatelé mohou procházet kalendář a získat přehled o všech událostech. Kalendář umožňuje filtraci, která zobrazí pouze představení, na kterých uživatel má naplánovanou směnu. Tímto způsobem uživatelé mohou snadno sledovat své pracovní povinnosti vzhledem k divadelnímu repertoáru.




**Obrázek 3.7:** Prototyp stránky kalendáře

### Jednotlivá směna

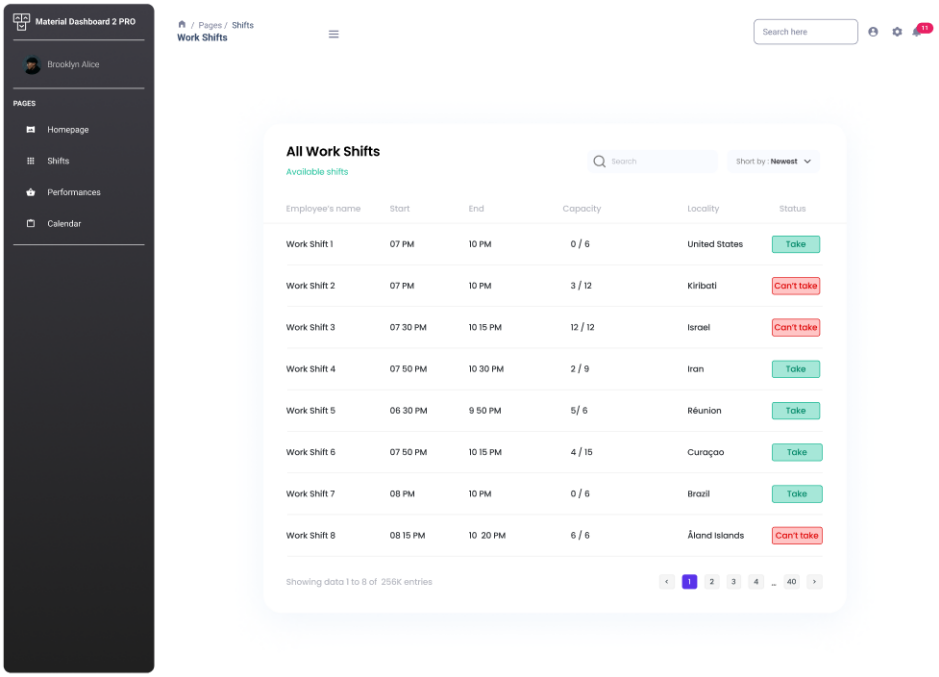
Tato obrazovka poskytuje podrobné informace o konkrétní směně. Uživatelé zde najdou informace jako datum a čas směny, popis práce, počet potřebných zaměstnanců a seznam již přidělených zaměstnanců. Tato obrazovka také umožňuje uživatelům žádat o změnu směny, pokud je to potřeba, a provádět další interakce související s danou směnou.




**Obrázek 3.8:** Prototyp stránky jednotlivé směny

### Seznam směn

Tato obrazovka zobrazuje uživateli seznam všech dostupných směn. Uživatelé mohou procházet seznam směn a získat podrobnosti o každé směně, včetně data, času, popisu a obsazení.




**Obrázek 3.9:** Prototyp stránky všech směn.

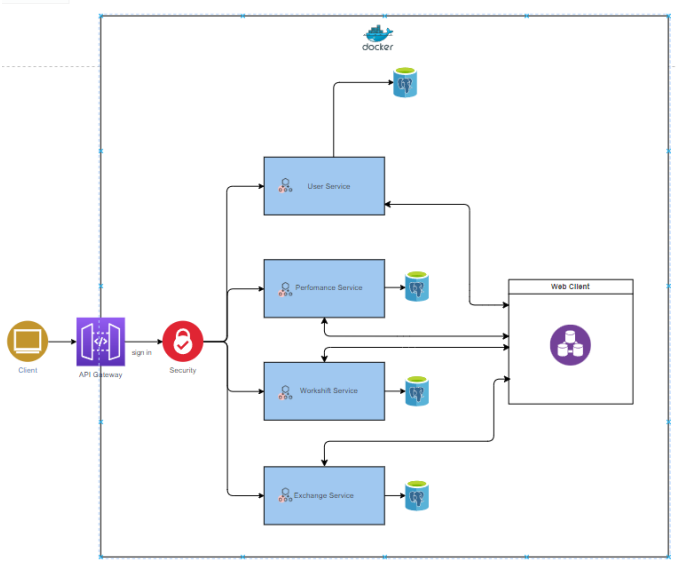
# Kapitola 4
# Implementace

Táto kapitola se zaměřena na proces implementace, který hraje klíčovou roli při realizaci každého projektu. Implementace představuje fázi, během které se koncepty, plány a design přeměňují z myšlenky na realitu. Tady bude uvedeno jake technologie jsem použil a zdůvodním proč.

## 4.1 Návrh backendu a technologie

Backend systému představuje klíčovou část, která je odpovědná za zpracování dat, implementaci logiky aplikace a interakci s databází. Je to serverová strana systému, která není přímo viditelná pro uživatele, ale poskytuje základní funkcionality a služby pro frontend.[11]

Výběr technologií pro vývoj systému je klíčovým rozhodnutím, které ovlivní výkonnost, flexibilitu a údržbu systému. Pro tento projekt byla zvolena kombinace programovacího jazyka Java a frameworku Spring Boot. Níže jsou uvedeny klíčové technologie, které mohou být využity pro implementaci různých částí divadelního systému.
Jak již bylo zmíněno, zvolenou architekturou pro tento projekt jsou mikroservisy. V této architektuře jsou požadavky od klienta zpracovávány pomocí API Gateway, který následně distribuuje tyto požadavky na příslušné mikroslužby.




**Obrázek 4.1:** Ukázka mikroservisní architektury

### 4.1.1 Java a Spring Boot

Jazyk Java a framework Spring Boot budou základem pro vývoj mikroslužeb v backendu. Poskytují robustní a efektivní prostředí pro vývoj, testování a nasazení.
Komunikace mezi mikroslužbami probíhá prostřednictvím Spring WebClient, což je neblokující, reaktivní klient pro provádění HTTP požadavků. Spring WebClient umožňuje efektivní a asynchronní volání mezi mikroslužbami, což zlepšuje celkový výkon a škálovatelnost systému.[12]
Pro správu a objevování služeb využívám Eureka[13], což je služba pro registraci a vyhledávání mikroservisů. Eureka zajišťuje, že jednotlivé mikroslužby mohou snadno najít a komunikovat mezi sebou, což je klíčové pro správnou funkci celého systému.

### 4.1.2 Maven

Maven bude používán pro správu a řízení projektu. Jeho funkcionalita zahrnuje stahování závislostí, kompilaci, testování a balení aplikace. Ačkoli existuje i jiný populární nástroj pro správu projektů, Gradle, rozhodl jsem se pro Maven kvůli jeho široké podpoře v komunitě, jednoduchosti konfigurace a rozsáhlým pluginům, které usnadňují práci na projektu. Navíc mi vyhovuje jeho syntaktická struktura, což zvyšuje efektivitu mé práce.

### 4.1.3 REST API

Komunikace mezi frontendem a backendem bude probíhat prostřednictvím RESTful API. Tento architektonický styl umožní jednoduchou integraci, jednotnou komunikaci a zajištění flexibility při vývoji aplikace.Ačkoli mám zkušenosti s gRPC, což je moderní framework pro vzdálené volání procedur s vysokým výkonem a efektivní komunikací, mé hlavní zkušenosti jsou s RESTful API. Rozhodl jsem se pro REST, protože mám s ním bohatší zkušenosti a je široce používaný a dobře podporovaný ve vývojářské komunitě.

### 4.1.4 Databáze

Každá mikroslužba bude má svou vlastní databázi PostgreSQL, což poskytuje široké možnosti pro správu a manipulaci se strukturovanými daty. PostgreSQL je známý svou vysokou úrovní konzistence a spolehlivosti, což je klíčové pro úspěšné fungování mikroservisní architektury. Díky své flexibilitě a škálovatelnosti je PostgreSQL ideální volbou pro ukládání dat v prostředí s různými mikroslužbami. Jelikož každá mikroslužba bude mít svou vlastní instanci databáze PostgreSQL, zajišťuje se izolace dat a minimalizuje se riziko konfliktů a výpadků systému.

### 4.1.5 Docker

V tomto projektu všechny mikroslužby a související aplikace budou nasazeny v kontejnerech pomocí Dockeru. To znamená, že celé aplikace bude spouštěna a provozována v kontejnerech, což zajišťuje konzistenci a jednoduchost při nasazování a správě aplikace.

### 4.1.6 Zabezpečení

Zabezpečení je zásadní pro každou moderní aplikaci, zejména pokud jde o správu citlivých dat a ochranu před neoprávněným přístupem. Pro zabezpečení své aplikace jsem zvolil Keycloak, který poskytuje robustní správu identit a přístupů.

# Kapitola 5
# Testování

## 5.1 Testování pomocí Postman

Pro testování jsem zvolil nástroj Postman. Tento nástroj jsem vybral, protože je jedním z nejpopulárnějších nástrojů pro testování API a mám s ním osobní zkušenosti z praxe. Postman nabízí uživatelsky přívětivé rozhraní, které umožňuje rychlé a efektivní vytváření, odesílání a správu HTTP požadavků. Díky svým bohatým funkcionalitám, jako jsou testovací skripty, kolekce a možnost automatizovaného testování, je Postman ideálním nástrojem pro testování jak jednoduchých, tak složitých API.[15]

### 5.1.1 Jak funguje testování pomocí Postman

Testování pomocí Postman probíhá ve více krocích:

- **Vytváření požadavků:** Postman umožňuje uživatelům vytvářet HTTP požadavky různých typů (GET, POST, PUT, DELETE atd.) a definovat jejich parametry, hlavičky a těla
- **Odesílání požadavků:** Po vytvoření požadavku může uživatel požadavek odeslat na určený API endpoint a okamžitě získat odpověď.
- **Analýza odpovědí:** Postman poskytuje detailní informace o odpovědi, včetně stavového kódu, hlaviček a těla odpovědi. Uživatelé mohou prohlížet odpovědi ve formátu JSON, XML nebo HTML.

Dalsí vyhoda Postmanu je v tom, že uživatelé mohou organizovat své požadavky do kolekcí, což usnadňuje správu a opakované testování API.

### 5.1.2 Vytváření testovacích scénářů

V Postmanu jsem vytvořil sadu testovacích scénářů pro jednotlivé API endpointy mé aplikace. Tady jsou popsany hlavní scénaři:

- **Přihlášení:** Testování autentizačního endpointu, ověření správnosti přihlašovacích údajů a získání autentizačního tokenu.
- **Profil uživatele:** Testování načítání a aktualizace uživatelských profilů, kontrola správnosti vrácených dat.
- **Seznam zaměstnanců:** Testování načítání seznamu zaměstnanců, včetně možnosti filtrování a zobrazení detailů jednotlivých zaměstnanců.
- **Kalendář:** Ověření správného zobrazení všech představení divadla, včetně možnosti zobrazení směn konkrétního uživatele.
- **Předání a odebírání směn:** Ověření, že uživatel může úspěšně převzit a odebrat směnu.
- **Vytváření a úprava představení:** Ověření správného ukládání všech potřebných informací o představení, jako jsou název, popis, datum a čas
- **Výměna směn:** Ověření, že zaměstnanci mohou úspěšně zahájit žádost o výměnu směny, druhý zaměstnanec může žádost přijmout nebo odmítnout, směny se správně aktualizují v systému

## 5.2 Unit testy

Unit testy jsou automatizované testy, které ověřují správnost jednotlivých částí softwarového kódu. Tyto jednotky jsou obvykle nejmenšími logickými komponentami aplikace, jako jsou funkce, metody nebo třídy. Cílem unit testů je zajistit, že každá část kódu funguje správně a nezávisle na ostatních částech systému.

Pro napsání unit testů jsem si vybral dvě populární knihovny: JUnit a Mockito. Kombinace těchto dvou knihoven mi umožňuje efektivně testovat jednotlivé jednotky kódu i jejich interakce s ostatními částmi systému:

### 5.2.1 JUnit

- JUnit testy testují jednotlivé jednotky kódu bez závislostí na dalších objektech nebo službách.

- JUnit poskytuje základní framework pro unit testování v Javě, který se dobře kombinuje s dalšími knihovnami a nástroji, což umožňuje rozsáhlejší a přizpůsobitelnější testovací strategie.
- JUnit testy jsou zpravidla rychlejší, protože nevyžadují inicializaci mock objektů a mají méně složitou logiku.[16]

### 5.2.2 Mockito

- Mockito umožňuje vytvářet mock objekty pro simulaci chování závislostí, což je užitečné pro testování interakcí mezi jednotlivými komponentami.
- Díky mockování lze izolovat testované jednotky od jejich závislostí, což umožňuje testovat jednotlivé části kódu v různých scénářích a podmínkách.
- Mockito je velmi flexibilní a poskytuje široké možnosti konfigurace mock objektů, což usnadňuje testování složitějších logik a závislostí.[17]

Kombinace JUnit a Mockito umožňuje efektivně pokrýt jak základní testy jednotlivých metod, tak i komplexnější testy interakcí mezi komponentami, čímž se zajišťuje vyšší kvalita a spolehlivost softwaru.

# Kapitola 6
# Záver

V závěru této bakalářské práce lze konstatovat, že proces implementace mikroslužeb představoval výzvu, avšak současně poskytl mnoho cenných zkušeností a poznatků. Byla to první zkušenost s vytvářením mikroslužeb, ačkoli náročná, přinesla mnoho přínosů. Během této práce jsem se setkal s různými technickými a organizačními výzvami, které jsem úspěšně překonal. Získal jsem hlubší porozumění architektuře mikroslužeb a správě kontejnerů, a to vše mi poskytlo cenné dovednosti a znalosti. Přestože bylo cestou mnoho obtíží, celkově hodnotím tuto zkušenost jako pozitivní a inspirativní. Mám za sebou mnoho nových poznatků a jsem připraven čelit budoucím výzvám s větší jistotou a znalostmi.

## 6.1 Budoucí stáv aplikace

V budoucím vývoji aplikace se zaměřím na implementaci nových funkcionalit a vylepšení uživatelského prostředí. Plánuji rozšířit možnosti výběru divadel, aby uživatelé měli možnost pracovat nejen pro jednotlivá divadla, ale i pro celé divadelní skupiny. Tímto krokem se otevřou nové perspektivy pro spolupráci a koordinaci mezi různými divadelními institucemi, což přinese efektivnější organizaci a plánování v divadelním průmyslu. Tato inovace umožní uživatelům lépe využít své dovednosti a zkušenosti v oblasti divadelní produkce a poskytne jim širší spektrum pracovních příležitostí.

### 6.1.1 Mobilní aplikace

Dalším důležitým krokem je implementace mobilní aplikace pro zaměstnance. Tato aplikace umožní zaměstnancům snadný přístup k potřebným informacím a funkcionalitám přímo z jejich mobilních zařízení. Mobilní aplikace bude

navržena tak, aby byla intuitivní a uživatelsky přívětivá, což umožní zaměstnancům rychle a efektivně provádět své úkoly i na cestách.

Mobilní aplikace bude integrována s existujícím systémem, což umožní synchronizaci dat v reálném čase. To zajistí, že zaměstnanci budou mít vždy aktuální informace, a umožní lepší plánování a rozhodování. Mobilní aplikace bude také podporovat notifikace, které zaměstnance upozorní na důležité události, změny v rozvrhu nebo nová oznámení.

V budoucnu lze zvažovat rozšíření aplikace o další funkcionality, jako je například možnost sledování výkonů zaměstnanců, správa úkolů a projektů, nebo integrace s dalšími nástroji používanými v divadelním prostředí. Tento krok přinese nejen technologický pokrok, ale také přispěje k modernizaci a zlepšení celkové uživatelské zkušenosti.

### 6.1.2 Synchronizace s Google Calendar

Následujícím zásadním vylepšením bude implementace synchronizace s Google Calendar. Tato funkce umožní zaměstnancům snadno integrovat svůj pracovní rozvrh s osobním kalendářem na svém mobilním zařízení, což zvyšuje pohodlí a zlepšuje organizaci času.

Synchronizace s Google Calendar poskytuje několik výhod. Umožňuje zaměstnancům mít všechny důležité informace na jednom místě, dostávat upozornění na nadcházející události a snadno spravovat své pracovní a osobní závazky. Díky této integraci mohou zaměstnanci lépe plánovat svůj čas a minimalizovat riziko konfliktů mezi pracovními a osobními aktivitami.

Mám s touto funkcionalitou osobní zkušenost. Ve svých studiích jsem importoval rozvrh předmětů z univerzitního systému STAG (Studijní agendy) do svého telefonu pomocí Google Calendar. Tato možnost mi velmi usnadnila organizaci času a pomohla mi efektivněji řídit své studijní a osobní aktivity.

# Literatura


[1] Shameless. [Cit.22.05.2025]. Dostupné na https://www.shameless.cz/

[2] Theatron. [Cit.12.01.2024]. Dostupné na https://www.theatron.eu/

[3] Propared. [Cit.12.01.2024]. Dostupné na https://www.propared.com/

[4] Komárek, Ing.Martin., Specifikace požadavků (přednáška). [Cit.12.01.2024]. Dostupné na https://moodle.fel.cvut.cz/pluginfile.php/338210/mod_resource/content/4/SpecifikacePozadavku_2019_Prednaska3.pdf

[5] Komárek, Ing.Martin., Analýza a Dokumentace Požadavků (přednáška). [Cit.12.01.2024]. Dostupné na https://moodle.fel.cvut.cz/pluginfile.php/338214/mod_resource/content/6/Analyza_A_Dokumentace_Formulace_Pozadavku_Prednaska7_Part2_LS2022.pdf

[6] IBM, Rational Software Architect. [Cit.21.02.2024]. Dostupné na https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

[7] IBM, Microservices. [Cit.25.02.2024]. dostupné na https://www.ibm.com/topics/microservices

[8] Microservices vs. Monolith Architecture. [Cit.13.05.2024]. Obrázek dostupen na https://dev.to/alex_barashkov/microservices-vs-monolith-architecture-4l1m

[9] User Interface (UI) Design. [Cit.15.5.2024]. Dostupné na https://www.geeksforgeeks.org/user-interface-ui/

[10] Material UI. [Cit.10.3.2024]. Dostupné na https://mui.com/

[11] Backend vs Frontend. [Cit.21.04.2024]. Dostupné na https://www.geeksforgeeks.org/frontend-vs-backend/

[12] Spring WebClient. [Cit.15.05.2024]. Dostupné na https://docs.spring.io/spring-framework/reference/web/webflux-webclient.html

[13] Introduction to Spring Cloud. Netflix – Eureka. [Cit.15.05.2024]. Dostupné na https://www.baeldung.com/spring-cloud-netflix-eureka

[14] How to Deploy Microservice Architecture in Docker. [Cit.11.04.2024]. Dostupné na https://middleware.io/blog/microservices-architecture-docker/

[15] What is Postman. [Cit.19.05.2024]. Dostupné na https://www.postman.com/home

[16] Junit Tutorial. [Cit.23.05.2024]. Dostupné na https://www.baeldung.com/junit

[17] Mockito - unit test framework. [Cit.23.05.2024]. Dostupné na https://www.itnetwork.cz/java/testovani/mockito-unit-test-framework

# Příloha A

# Obsah přiložených souborů

## A.1 Zdrojový kód

```
TheaterManager - Hlávní modul
├── userservice - Služba zodpovědná za správu uživatelů.
│   └── src
│       └── main - Hlavní adresář se zdrojovými kódy pro userservice.
│           ├── java
│           └── resources
│               └── application.propertioes - Soubor s vlastnostmi pro konfiguraci služby uživatelů.
├── performanceservice - Služba věnovaná správě představení.
│   ├── src
│   └── main - Hlavní adresář se zdrojovými kódy pro performanceservice.
│       ├── java
│       └── resources
│           └── application.propertioes - Soubor s vlastnostmi pro konfiguraci služby představení.
├── workshiftservice - Služba zodpovědná za správu pracovních směn.
│   ├── src
│   └── main - Hlavní adresář se zdrojovými kódy pro workshifrtservice.
│       ├── java
│       └── resources
│           └── application.propertioes - Soubor s vlastnostmi pro konfiguraci služby pracovních směn.
├── exchangeservice - Služba zajišťující výměnu pracovních směn mezi pracovníky.
│   ├── src
│   └── main - Hlavní adresář se zdrojovými kódy pro exchangeservice.
│       ├── java
│       └── resources
│           └── application.propertioes - Soubor s vlastnostmi pro konfiguraci výměnu směn.
├── discoveryservice - Služba umožňující ostatním službám dynamicky se najít a komunikovat mezi sebou (Eureka).
│   ├── src
│   └── main - Hlavní adresář se zdrojovými kódy pro Eureka.
│       ├── java
│       └── resources
│           └── application.propertioes - Soubor s vlastnostmi pro konfiguraci objevovací služby Eureka.
├── api-gateway - Služba  poskytující jednotný vstupní bod pro klienty.
│   ├── src
│   └── main - Hlavní adresář se zdrojovými kódy pro api-gateway.
│       ├── java
│       └── resources
│           └── application.propertioes - Soubor s vlastnostmi pro konfiguraci brány API.
├── keyclock - Adresář, který obsahuje soubory související s řízením přístupu a autentizací.
│   └── realms
│       └── realms-export.json - Soubor ve formátu JSON obsahující exportovanou konfiguraci.
└── docker-compose.yml - Tento soubor obsahuje konfiguraci Docker Compose, která definuje, jaké kontejnery a služby budou spuštěny.
```

**Obrázek A.1:** Prototyp stránky všech směn.

# Příloha B

# Uživatelská příručka

Pro spuštění aplikace je třeba postupovat následovně:

## B.1 Spuštění serveru Keycloak

- Přejděte do adresáře s Docker Compose konfigurací.
- Spusťte server Keycloak pomocí příkazu v terminálu.

```
docker-compose up -d keycloak
```

- Pokud používáte integrované vývojové prostředí (IDE) jako IntelliJ IDEA, lze server Keycloak spustit pomocí grafického rozhraní IDE. Je to možné pomocí tlačítka nebo volby v menu, která umožňuje spouštět Docker Compose služby přímo z IDE.

## B.2 Přihlášení do administračního rozhraní Keycloak

- Otevřete webový prohlížeč a přejděte na adresu localhost:8069.
- Přihlaste se jako administrátor s uživatelským jménem "admin"a heslem "admin".

## B.3 Zvolení realmu

- Po přihlášení vyberte realm "realm-theater-manager"ze seznamu realmů.

## B.4 Nastavení klienta

- Vyberte klienta "spring-cloud-client"ze seznamu klientů.
- Přejděte do záložky "Credentials".
- Vygenerujte a zaznamenejte si "Client Secret".

## B.5 Spuštění ostatních služeb z Docker Compose

- Po nastavení klienta spusťte všechny ostatní služby aplikace pomocí Docker Compose.

Po provedení těchto kroků by měla být aplikace úspěšně spuštěna a připravena k použití.

# Příloha C

## Seznam odkázů

**Repozitáře:**

- Github: https://github.com/azizoram/TheaterManager/tree/master
- Figma: https://www.figma.com/design/PiCtzXF9YDB5obBVoIHBdR/PrototypeBC?node-id=0-1&t=yCgyfLhhYFej51q7-0