České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

IS pro správu činnosti divadla

Anna Zlámalová

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

Softwarové inženýrství a technologie,
Bakalářský

2018

zadání

Poděkování

Tímto bych ráda poděkovala své vedoucí bakalářské práce Ing. Boženě Mannové, Ph.D. za rady a konzultace, na kterých jsme s ostatními studenty mohli sdílet postřehy při psaní práce a získávat zpětnou vazbu. Dále také své rodině a příteli za jejich finanční a psychickou podporu během studia.

Prohlášení

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

V Praze dne 24. 5. 2018 .....

Abstrakt

Práce je zaměřena na vývoj webových aplikací sloužících k řízení vnitřní umělecké organizace divadel. V rámci této práce byl proveden i průzkum fungování organizace v několika divadlech. Poté byla provedena specifikace, návrh a implementace systému. Realizovaný systém splňuje funkční poažadavky a je uživatelsky přívětivý. Výsledná aplikace byla otestována a je připravena na využití v několika divadlech.

Abstract

The work is focused on the development of web applications used to manage the internal artistic organization of theatres. In the framework of this work, a survey of the functioning of the organization was carried out in several theatres. Consequently, the specifications, design and implementation of the system were implemented. The system fulfills functional requirements and is user-friendly. The resulting application has been tested and is ready to be used in several theaters.

Obsah

Seznam příloh

Seznam obrázků

Úvod

Motivace

Motivací pro výběr tohoto tématu byly především osobní zkušenosti z pracovního prostředí divadla, konkrétně divadla SEMAFOR, kde jsme se občas potýkali s problémy s organizací směn. Napadlo mě vytvořit uživatelský portál, který by tuto organizaci usnadnil a zároveň zpřehlednil důležité informace, týkající se každodenního chodu divadla.

Popis problému

Problém, kterým se v této práci zabývám, se týká zpřehlednění umělecké organizace v divadlech. Umělecká organizace zahrnuje především řízení představení a umělců, kteří se v divadle angažují. Pojem Umělecká organizace používám právě proto, že se nebudu zabývat organizací finanční. Problém umělecké organizace je především v tom, že informace týkající se směn zaměstnanců a zaměstnanců samotných jsou roztroušené na více místech a jejich dohledání je zbytečně složité.

Cíl práce

Cílem práce je navrhnout, implementovat a otestovat webovou aplikaci pro správu umělecké činnosti divadel. Aplikace má sloužit ke správě současných směn, vytváření směn nových, tzv. fermanů, k zobrazení směn uživatelům na základě jejich přístupových práv, úpravu profilů zaměstnanců, vyhledávání infomací o směnách a zaměstnancích. Práce se nebude zabývat řízením financí divadla a řízením prodeje lístků.

Stručný popis systému

V systému umělecké organizace je udržován aktuální seznam představení, herců, kteří hrají v daných představeních a dalších zaměstnanců jako jsou technici, zvukaři, osvětlovači, uvaděčky, kostymérky či zaměstnanci kanceláře. Jednotliví zaměstnanci budou mít do vytvořené aplikace přístup na základě různých uživatelských rolí, shodných s jejich funkcemi v divadle. Pokud k tomu mají pravomoc, mohou v systému přidávat, upravovat či mazat směny, upravovat svůj profil, případně profil někoho jiného, tvořit fermany, vyhledávat potřebné informace a tyto informace pak zobrazit.

Obsah základních kapitol

Práce se skládá celkem z 5 kapitol, přičemž první čtyři kapitoly popisují jednotlivé fáze vývoje software.

- **Kapitola 1** se zabývá **analýzou** řešené problematiky. A to průzkumem existujících řešení organizace v divadlech a jejich porovnáním. Na základě průzkumu jsou vytvořeny funkční a nefunkční požadavky a požadavky na zavedení řešení, zanalyzovány uživatelské role v systému a jejich případy užití.
- **Kapitola 2** se věnuje **návrhu** architektury systému a jeho komponent, diagramům spojených s návrhem software a návrhu prototypu uživatelského rozhraní.
- **Kapitola 3** popisuje samotnou **implementaci** uživatelského portálu a konkrétní zvolené technologie pro vývoj.
- **Kapitola 4** pak pojednává o **testování** vytvořené aplikace.
- **Kapitola 5** je samotným **závěrem** práce, ve kterém jsou shrnuty výsledky práce, ohodnoceno splnění cíle práce a je uveden další postup vývoje aplikace.

Kapitola 1

Analýza řešení

Byl proveden průzkum v několika divadlech a průzkum dostupných systémů. Na základě výsledků byla vytvořena specifikace požadavků na aplikaci, byly určeny uživatelské role v systému a jednotlivé případy užití a jejich scénáře.

1.1 Průzkum organizace v divadlech

V rámci průzkumu existujících řešení byl proveden i průzkum toho, jak v některých divadlech organizace funguje. Domluvila jsem schůzky se zástupci celkem tří divadel, a to Divadla SEMAFOR, ABC a Rokoko a divadla Comica Economica. Průzkum probíhal formou rozhovoru, kdy každý zástupce odpovídal na otázky (viz. Příloha A) a popisoval, jak organizace v divadle funguje.

1.1.1 Divadlo SEMAFOR

Divadlo SEMAFOR je středně velké divadlo umístěné v Praze v Dejvicích a proslavilo se hlavně díky Jiřímu Suchému a Jitce Molavcové, kteří spolupracují již od roku 1970. Co se organizace týče, je divadlo rozdělené do více částí podle funkcí, které jednotliví zaměstnanci zastávají. Mezi tyto funkce patří herci, muzikanti, technici, osvětlovači, zvukaři, kostymérky, uvaděčky, barmanky a kancelářská administrativa.

Herci a muzikanti

Tyto dvě funkce se liší pouze v názvu, ale jsou téměř totožné, protože každý herec v divadle SEMAFOR je muzikant a každý muzikant je herec. Zároveň jsou to velmi podstatné funkce, od kterých se odvíjí určování tzv. fermanů, neboli rozpisů, na další měsíc. Tvorbu fermanů má na starost jedna osoba, která se s herci/muzikanty domlouvá přes e-mail, kdy mohou a kdy naopak nemohou hrát. Na základě těchto informací sestaví rozpis na další měsíc, podle kterého si pak rozdělují směny i ostatní zaměstnanci.

Technici, osvětlovači, zvukaři, barmanky a kostymérky

Každá z těchto částí má nejvýše 2 až 4 tzv. alternace, neboli zastoupení. Přičemž se v každé části domlouvají zaměstnanci mezi sebou a směny si rozdělí. To pak nahlásí v kanceláři, kde tyto informace zapíší do excel tabulek.

Uvaděčky

Princip tvorby rozpisů uvaděček je téměř podobný jako výše zmíněný, liší se však v tom, že je zde kolem 16 alternací a jedna hlavní uvaděčka. Ta dohlíží na to, aby byly všechny směny obsazené a aby uvaděčky na představení dorazily. Počet uvaděček na směnu se liší podle představení. Jednotlivé směny jsou variabilní, tj. se může stát, že si uvaděčky směny vymění i po zapsání do rozpisu.

Kancelář

Lidé v kanceláři jsou v divadle i během dne a starají se o veškerou administrativu. Pokud si chtějí zjistit, kdo kdy má jakou směnu, musí informaci najít v excelovských tabulkách nebo se jít zeptat daného člověka.

Představení se pak dělí na 3 typy, kterými jsou regulární představení, zájezdy, což jsou semaforská představení, ale hraná v jiných divadlech, a dále pronájmy jiným divadlům.

1.1.2 Divadlo ABC a Rokoko

Divadlo ABC a Rokoko se řadí mezi činoherní divadla a od roku 2005 působí jako společný subjekt s jedním souborem herců působícím na dvou scénách. Každodenní chod divadla zajišťují tajemníci a režiséři, kteří organizují herce v době zkoušení nových představení. Nová představení se zkouší vždy čtyřikrát do roka na premiéry v listopadu, březnu a červnu.

Fermany se dělí na měsíční, týdenní a denní. Přičemž měsíční musí být souboru rozeslán 2 měsíce předem, nejpozději do 15. dne měsíce, například do 15.11. musí být zaslán ferman na leden. Týdenní rozpis je pak posílán vždy ve čtvrtek do 14:00 a denní do 12:00 předcházejícího dne. Do fermanů se zapisují jak jednotlivá představení, která se ten den hrají, tak i například zájezdová představení, pronájmy, zkoušky na nová představení, tiskové konference, veřejná vystoupení, focení, apod.

S herci všechny záležitosti vyřizuje tajemník, který na základě domluvy s herci sestaví fermany. Ty pak rozešle ostatním zaměstnancům e-mailem v podobě excelovské tabulky. Všichni zaměstnanci, jako uvaděči, kostymérky, maskérky, zvukaři, osvětlovači nebo technici mají každý své oddělení a vedoucího, který s nimi domlouvá směny.

Herci

Herecké obsazení se dělí na 3 skupiny podle závazku vůči divadlům:

1. Herci, kteří mají **stálé angažmá** – Ti musí být vždycky k dispozici a přispůsobit se rozpisům. Také nemohou hostovat v ostatních divadlech.
2. Herci **s prioritou** – Také se musí přizpůsobovat rozpisům, ale na rozdíl od herců z 1. skupiny mohou hostovat i v jiných divadlech. Divadlo ABC a Rokoko však mají přednost před ostatními divadly, která se musí domluvit s tajemnicí, zda mohou herci v jiných divadlech hrát.
3. **Hosté** – Co se tvorby rozpisů týče, jsou hosté ti nejdůležitější a domlouvají se jako první. Na základě toho, kdy mohou a kdy naopak nemohou se totiž určují termíny představení, kterým se pak ostatní zaměstnanci musí přizpůsobit.

Herci vždy 14 dní před premiérou nesmí hostovat v jiných divadlech a musí chodit na zkoušky. Ty se dělí na hlavní zkoušky, generální zkoušky, technické svícené zkoušky a kostýmové zkoušky.

Režisér

Režisér určuje termíny zkoušek, kterým se všichni zaměstnanci musí přizpůsobit.

Tajemník

Přes tajemníka jde veškerá domluva, která se týká tvorby fermanů. Domlouvá se se všemi jak ohledně termínů zkoušek, které mu předá režisér, tak i s hosty, kteří mu nahlásí termíny, kdy nemohou hrát. Při tvorbě fermanů musí brát také v potaz, že má přednost divadlo ABC, které má větší kapacitu sálu.

Uvaděči

Domluva uvaděčů se v obou divadlech liší. V divadle Rokoko se směny domlouvají na měsíc dopředu a v divadle ABC vždy na týden dopředu přes Google dokument.

1.1.3 Divadlo Comica Economica

Divadlo Comica Economica je amatérské divadlo, které založil vysokoškolský pedagog Doc. RNDr. Bohumír Štědroň, CSc v roce 2010. Divadlo má vlastního organizátora a PR. Je složeno převážně ze studentů a dobrovolníků, kteří se scházejí každý týden v učebně, aby zkoušeli nová představení. Představení hrají dvakrát až třikrát do roka v jiných divadlech nebo na školách. Domluva mezi členy spolku probíhá přes e-mail. Rozpisy a aktuální alternace a herecké obsazení či aktuální představení nejsou nikde uvedeny.

1.1.4 Výsledek průzkumu v divadlech

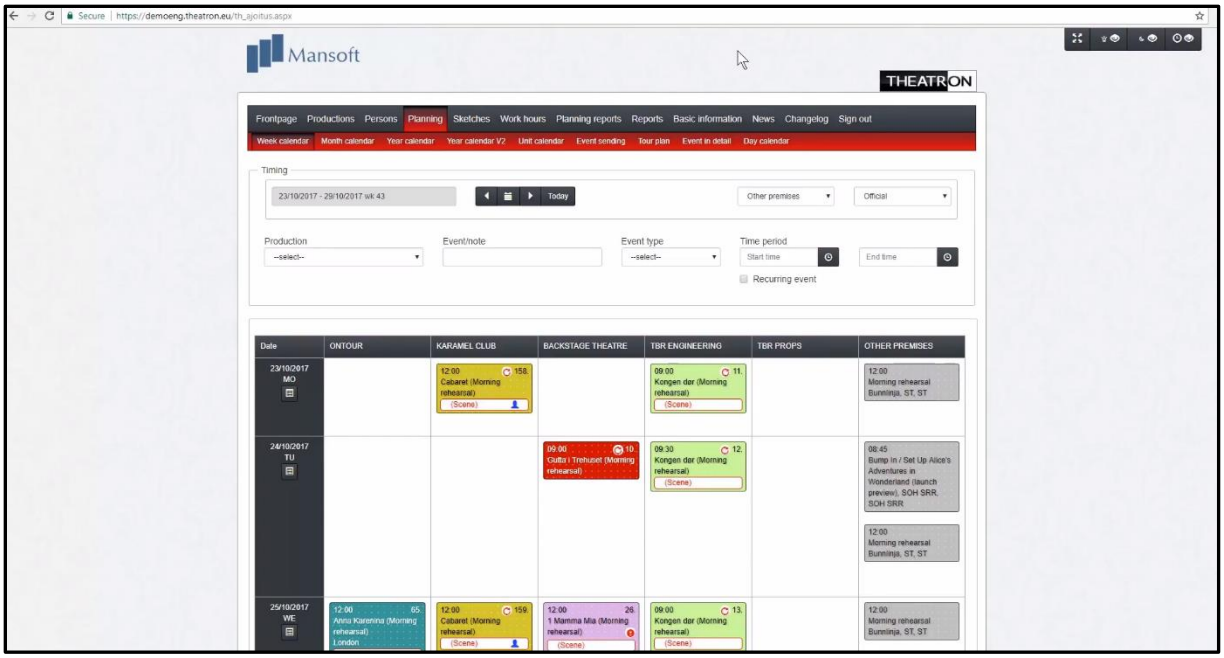
Po rozhovorech se všemi zástupci divadel jsem došla k závěru, že nikdo z nich nepoužívá žádnou aplikaci pro celkovou uměleckou organizaci a že většině by taková aplikace usnadnila práci. Všichni pro správu směn a zaměstnanců používají hlavně google tabulky, ve kterých je náročné udržet aktuální data. Pro větší divadla pak existuje mnoho tabulek a po čase může být vyhledávání informací v nich velmi zdlouhavé. Jako funkcionality v aplikaci by uvítali hlavně správu zaměstnanců a představení, a dále také správu směn či fermanů. Jediný problém, který by mohl podle členů divadel nastat při nasazení aplikace je přesvědčení členů divadla o využívání této aplikace. Všichni jsou po řadu let zvyklí na určité fungování organizace a zásah takovéto aplikace by mohl v některých případech ohrozit každodenní běh divadla, kde je občas už teď chaos na denním pořádku.

1.2 Průzkum existujících systémů

Průzkum existujících řešení zahrnuje průzkum existujících IT systémů. V této části jsem porovnala systémy Theatron, Propared a EventPro. Tyto aplikace jsem vybrala proto, že jsou specializované na chod divadla a jejich dema jsou přístupná na internetu. Cílem průzkumu bylo zhodnotit klady a zápory těchto aplikací a vybrat funkce, které jsou typické pro systémy tohoto typu.

1.2.1 Theatron

Theatron je software specializovaný na plánování produkce divadel. Konkrétně zajišťuje přehlednost informací v rámci vnitřního fungování divadla, jejich sdílení, správu rozpisů a responzivitu aplikace. Jedná se o webovou aplikaci, u které není potřeba instalovat nebo nastavovat server.



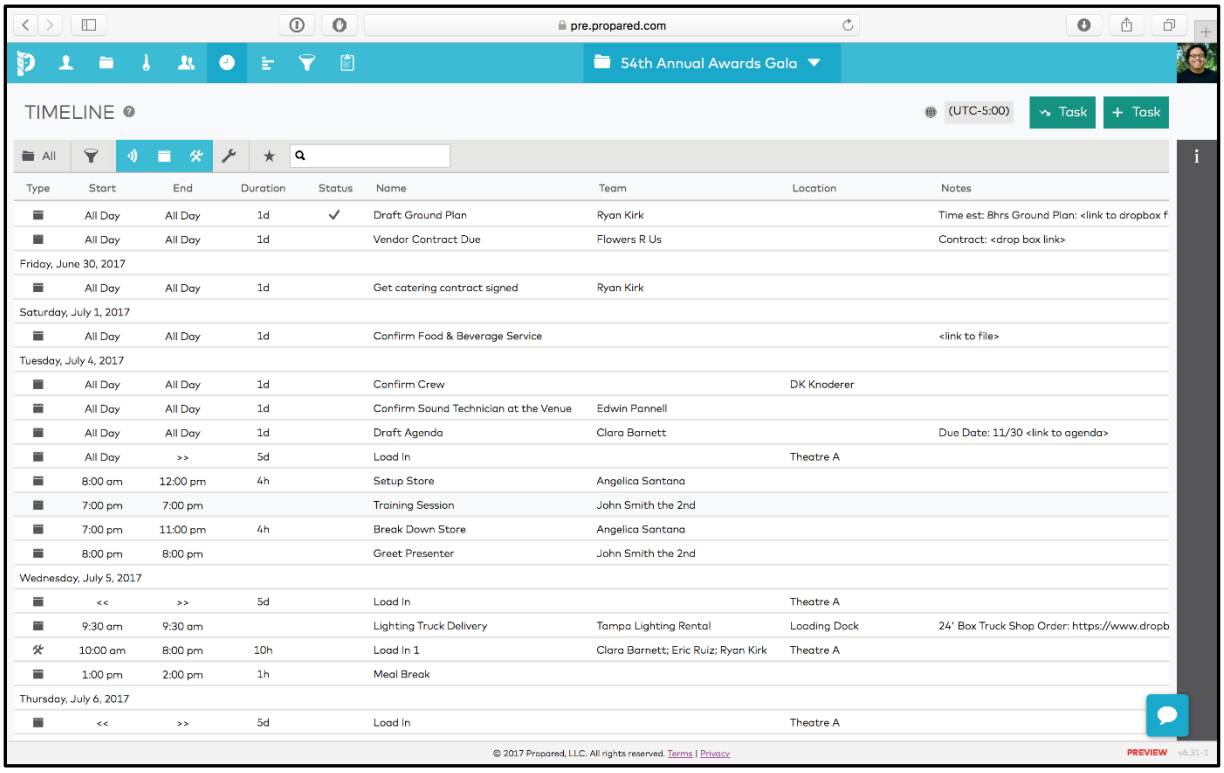
Obrázek 1 - Ukázka systému Theatron

Jako jeho výhodu bych uvedla především správu směn ve formě kalendáře. Směny jsou barevně rozlišitelné podle typu a obsahují pouze základní informace, což dělá kalendář velmi přehledným. Po kliknutí na směnu se pak objeví detailní informace a uživatel může informace rovnou upravit. Vyhledávání mezi směnami je velmi snadné, především díky filtru směn podle typu či data. Další nespornou výhodou je možnosti tisku směn přímo z aplikace. Aplikace nabízí také intranet pro zaměstnance, který má omezené funkcionality a slouží především k zobrazování informaci a k jejich rychlému dohledání. Aplikace je dostupná i pro mobilní zařízení a tablety.

Nevýhodou je především jistá nepřehlednost ve funkcionalitách spravovacího režimu. Je zde příliš mnoho funkcionalit a záložek, které musí uživatel nejprve proklikat, aby se v nich začal orientovat. Rozdělení tedy není příliš intuitivní. Další nevýhodou je design, který je celkem zastaralý a nedodržuje základní grafické konvence.

1.2.2 Propared

Propared je webová aplikace pro organizování rozpisů a logistiky pro představení, ve které je možné mimo zkoušky a představení vytvářet i to-do listy či tzv. run-of-show v rámci představení.



Obrázek 2 - Ukázka systému Propared

Údaje o směnách jsou zobrazeny ve formě seznamu, který má velmi propracovaný filtr pro vyhledávání. Velkou výhodou je i posílání informací o směnách uživatelům přímo z aplikace. Aplikace je přístupná i z mobilních telefonů i tabletů.

1.2.3 EventPro

Jedním z produktů EventPro je i řídící software specializovaný pro divadla. Zajišťuje funkcionality pro řízení událostí divadla jako např. rozvrh místností, správu účastníků, organizaci perzonálu nebo správu financí.

1.2.4 Porovnání existujících systémů

Uživatelsky nejpřívětivější je aplikace Propared. Design je relativně moderní a ikonky jsou víceméně intuitivní. V některých případech jsou však ikonky spíše na škodu a slovní popis by v těchto případech byl lepší. Theatron sice má slovní popis záložek, ale je jich příliš mnoho a chvíli potrvá, než se v nich uživatel zorientuje. Na rozdíl od Propared, zobrazuje směny barevně odlišené ve formě kalendáře, což mi přijde mnohem lepší než seznam. Obě tyto aplikace jsou na rozdíl od EventPro přímo vyvíjené pro divadla, tudíž jsou více přizpůsobené pro každodenní fungování divadla.

1.3 Specifikace požadavků

Na základě provedené analýzy byly specifikovány požadavky a to požadavky funkční, požadavky nefunkční, neboli kvalitativní a požadavky na zavedení řešení. [1]

Neformálně se jednotlivé požadavky značí buď jako „Must have" a zahrnují funkce, které v systému musí být, anebo „Nice to have", které jsou pro systém spíše doplňkové a nemají vliv na datovou strukturu systému. Pro tuto prioritizaci požadavků budu používat klíčová slova MUST a SHOULD. [2]

1.3.1 Funkční požadavky

Funkční požadavky definují všechny funkce a možnosti, které bude systém uživatelům umožňovat.

Základní uživatelské akce

Tyto požadavky se zaměřují na základní funkčnosti aplikace, které ovšem může provádět pouze registrovaný uživatel.

FR01 – Přihlásit uživatele [MUST]

Uživatelé se budou přihlašovat do systému pomocí e-mailu a hesla.

FR02 – Odhlásit uživatele [MUST]

Přihlášení uživatelé se budou moct odhlásit ze systému.

FR03 – Zobrazit vlastní profil [MUST]

Uživatel má možnost zobrazit a upravovat svůj vlastní profil a oprávnění uživatelé i profily ostatních uživatelů.

FR04 – Upravit profil uživatele [MUST]

Uživatel má možnost upravit svůj vlastní profil a oprávnění uživatelé i profily ostatních uživatelů.

FR05 – Přidat/odebrat uživatele [MUST]

Oprávnění uživatelé mohou přidávat nové uživatele a odebírat ty, kteří už v divadle nepracují.

FR06 – Zobrazit profil jiného uživatele [MUST]

Všichni uživatelé mohou zobrazovat profily ostatních uživatelů pouze se základními informacemi (např. jméno, příjmení, role). Oprávnění uživatelé mohou zobrazit i ostatní informace (např. adresu, e-mail, telefon).

FR07 – Zobrazit seznam uživatelů [MUST]

Všichni uživatelé mohou zobrazit seznam všech uživatelů a z tohoto seznamu se dostat na jejich profil.

FR08 – Vyhledat uživatele [SHOULD]

Všichni uživatelé mohou vyhledávat ostatní uživatele v seznamu všech uživatelů.

Směny

FR09 – Zobrazit všechny směny ve formě kalendáře [MUST]

Uživatel bude moct na úvodní stránce zobrazit směny ve formě kalendáře buď na aktuální den, týden nebo měsíc. Dále je možné vidět i směny na měsíc dopředu.

FR10 – Zobrazit vlastní směny [MUST]

Uživatel si může zobrazit pouze vlastní směny na aktuální, případně na další měsíc.

FR11 – Zobrazit směnu jiného uživatele [SHOULD]

Uživatel si může zobrazit směny ostatních uživatelů.

FR012 – Přidat/odebrat/upravit směnu [MUST]

Oprávnění uživatelé mohou v systému vytvářet nové směny, upravovat stávající či mazat ty staré. Směnám mohou přiřadit typ (premiéra, veřejná generálka, zkouška či zájezd).

FR13 – Prohlížet historii směn [SHOULD]

Uživatel může v systému dohledat i směny z minulých měsíců.

FR14 – Sčítat směny [SHOULD]

Na konci každého měsíce se provádí sčítání směn a jejich následné proplácení. Systém tedy poskytne funkci, která tyto směny automaticky spočte.

Představení

FR15 – Zobrazit informace o představení [MUST]

Všichni uživatelé mají přístup k zobrazení informací o představeních.

FR16 – Přidat/odebrat/upravit představení [MUST]

Oprávnění uživatelé mohou přidat, upravit nebo odstranit představení.

FR17 – Přidat/odebrat/upravit role [MUST]

Oprávnění uživatelé mohou přidat, upravit nebo odstranit role a alternace v představení.

FR18 – Vyhledat informace o představení [MUST]

Všichni uživatelé mají přístup k zobrazení informací o představeních.

Nástěnka

FR19 – Zobrazit vzkazy na nástěnce [SHOULD]

Uživatel může zobrazit všechny vzkazy na nástěnce.

FR20 – Přidat/odebrat/upravit vlastní vzkaz [SHOULD]

Všichni uživatelé mohou přidat, upravit nebo odebrat vlastní vzkaz na nástěnce.

FR21 – Přidat/odebrat/upravit vzkazy ostatních uživatelů [SHOULD]

Oprávnění uživatelé mohou přidat, upravit nebo odebrat i vzkazy jiných uživatelů.

1.3.2 Nefunkční požadavky

Nefunkční požadavky, neboli kvalitativní, jsou jakési doplňky k funkčním požadavkům. Týkají se například výkonu, délky odezvy, dostupnosti, spolehlivosti či bezpečnosti. [1]

NFR01 – Dostupnost přes webový prohlížeč [MUST]

Systém jako webová aplikace bude dostupný přes webový prohlížeč, tzv. tenkého klienta, ze serveru.

NFR02 – Responzivita [MUST]

Aplikace bude responzivní.

NFR03 – Zabezpečení systému [MUST]

Systém bude zabezpečený a všechny chyby budou správně odchyceny, tj. nedostanou se k uživateli.

NFR04 – Zajištění práv uživatelů [MUST]

K určitým funkcionalitám systému budou mít přístup pouze oprávnění uživatelé.

NFR05 – Vnitřní komunikace [MUST]

Komunikace mezi frontendem a backendem probíhá pomocí RESTful API.

NFR06 – Technologie [MUST]

Aplikace je psána v jazyce Java (Enterprise Edition), JavaScript s využitím HTML, CSS a technologií PosgreSQL, JPA, EclipseLink, Maven, Spring, React, Bootstrap a Node.js.

1.3.3 Požadavky na zavedení řešení

Požadavky na zavedení řešení jsou „dočasné“ požadavky, které po zavedení řešení ztrácejí smysl. Typicky jsou to požadavky na tvorbu uživatelské či administrátorské dokumentace, školení klíčových a ostatních uživatelů, požadavky na migraci dat, přípravu náhradních řešení, apod. [3]

TR01 – Vytvoření uživatelské příručky [SHOULD]

Je třeba vytvořit uživatelskou příručku, která bude obsahovat návod, jak s aplikací pracovat.

TR02 – Příprava náhradního řešení [SHOULD]

Do uživatelské příručky budou přidány postupy, jak se zachovat při neočekávaném chování aplikace.

TR03 – Zaškolení klíčových uživatelů [MUST]

Bude nutné vysvětlit klíčovým uživatelům, jak mají v aplikaci pracovat, aby mohli vykonávat svou práci.

TR04 – Uložení dat do databáze aplikace [MUST]

Před uvedením aplikace do chodu budou hromadně přidaná data do databáze. Tato data se týkají informací o představeních, zaměstnancích, apod.

TR05 – Instalace aplikace na server [MUST]

Aplikace musí být instalována na server divadla, aby byla přístupná všem uživatelům.

1.4 Případy užití

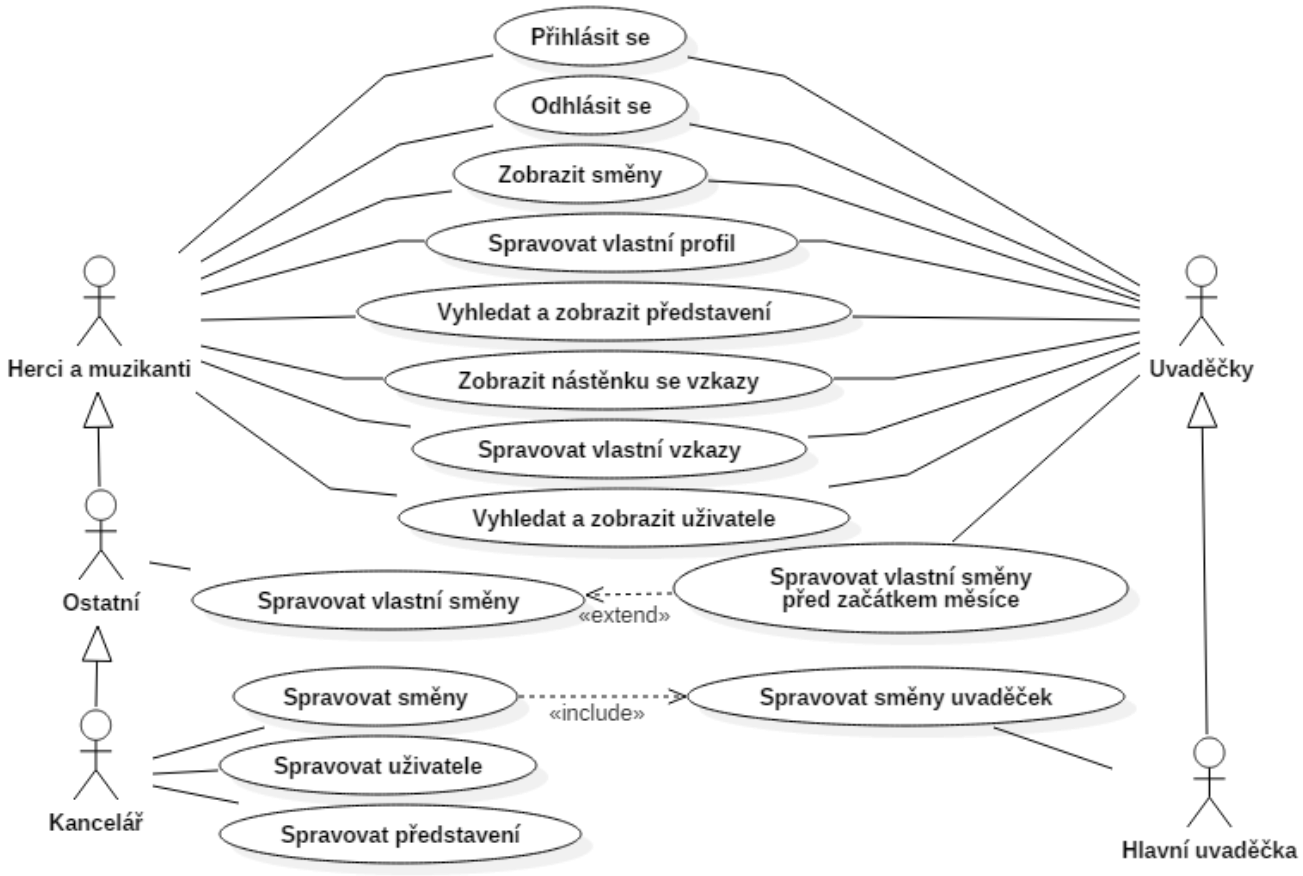
Případ užití (anglicky use case, zkráceně UC) popisuje chování systému z pohledu uživatele, nazývaného primární aktér. Tento aktér vyvolává v rámci případu užití interakci se systémem za účelem dosažení určitého cíle. V závislosti na požadavcích a podmínkách u původního požadavku se mohou objevit odlišné způsoby chování, neboli scénáře. Případy užití sdružují tyto rozdílné scénáře dohromady v jeden celek. [5]

1.4.1 Definice uživatelských rolí

Všichni uživatelé musí být do systému přihlášeni.

- **Administrátor** – Má přístup ke všem funkcionalitám systému.
- **Kancelář** – Zaměstnanci kanceláře mají přístup ke všem informacím v systému, které mohou měnit.
- **Herci, muzikanti** – Mohou zobrazit jejich vlastní směny a informace o ostatních hercích či muzikantech. Zde však budou i nějaké výjimky.
- **Hlavní uvaděčka** – Může upravovat směny ostatních uvaděček kdykoliv.
- **Uvaděčky** – Mohou upravovat směny pouze před začátkem měsíce.
- **Zvukaři, osvětlovači, technici, kostymérky** – Mohou upravovat vlastní směny kdykoliv.

1.4.2 Diagram případů užití



Obrázek 3 - Diagram případů užití

1.4.3 Jednotlivé případy užití

Jednotlivé případy užití jsou pro zjednodušení rozděleny do dvou skupin:

- Všichni uživatelé, kde jsou případy užití společné pro všechny uživatele a
- Oprávnění uživatelé, která obsahuje případy užití pouze pro uživatele, kteří mají k akci oprávnění.

Každý případ užití obsahuje jednoduchý popis a jeho hlavní, případně alternativní nebo chybový scénář užití.

Všichni uživatelé

UC01 – Přihlásit se

Uživatel zadá do přihlašovacího formuláře e-mailovou adresu a heslo a potvrdí odeslání. Systém uživatele vyhledá v databázi a ověří, zda uživatel opravdu existuje a pokud ano, zda zadal správné údaje.

Scénář:

1. Uživatel otevře úvodní stránku aplikace.
2. Systém zobrazí přihlašovací formulář.
3. Uživatel do něj zadá e-mailovou adresu a heslo a klikne na tlačítko „Přihlásit se“.
4. Systém vyhledá uživatele v databázi a pokud uživatel zadal správné údaje, systém ho automaticky přesměruje na hlavní stránku aplikace.

Chybový scénář:

Systém daného uživatele v databázi nenajde nebo zjistí, že uživatel zadal chybné heslo. V obou případech uživateli zobrazí chybovou hlášku, případně mu nabídne možnost heslo změnit.

UC02 – Odhlásit se

Přihlášený uživatel klikne na tlačítko „Odhlásit se“ v pravém horním rohu. Systém ho automaticky přesměruje na úvodní stránku s přihlašovacím formulářem.

UC03 – Zobrazit kalendář

Na hlavní stránce systém zobrazí kalendář, do kterého vypíše všechny směny. Výpis směn je buď na aktuální den, týden nebo měsíc, případně na následující měsíc.

Scénář:

1. Uživatel otevře hlavní stránku s kalendářem, na kterou je přesměrován systémem buď po přihlášení nebo po kliknutí na záložku „Kalendář“ v navigaci.
2. Systém vyhledá v databázi všechny směny odpovídající danému kritériu. Výchozí možnost je zobrazit směny na aktuální den. Uživatel si může vybrat i aktuální týden, aktuální měsíc či následující měsíc.

UC04 – Zobrazit rozpis vlastních směn

Uživatel může zobrazit vlastní směny a to na aktuální i následující měsíc.

Scénář:

1. Uživatel otevře stránku s rozpisy po kliknutí na záložku „Směny“ v navigaci.
2. Systém vyhledá v databázi směny uživatele odpovídající danému kritériu.

UC05 – Zobrazit rozpis směn jiného uživatele

Uživatel může zobrazit směny jiného uživatele a to na aktuální i následující měsíc.

Scénář:

1. Uživatel otevře stránku s rozpisy po kliknutí na záložku „Směny“ v navigaci.

2. Uživatel zadá jméno či příjmení hledaného uživatele do kolonky pro vyhledávání a klikne na tlačítko „Zobrazit směny“.
3. Systém vyhledá v databázi směny daného uživatele a vypíše je uživateli.

UC06 – Zobrazit vlastní profil

Uživatel může zobrazit vlastní profil se všemi informacemi.

Scénář:

1. Uživatel otevře profil po kliknutí na záložku „Profil“ v navigaci.
2. Systém vyhledá a zobrazí všechny informace o uživateli.

UC07 – Zobrazit profil jiného uživatele

Uživatel může zobrazit profil jiného uživatele se všemi informacemi.

Scénář:

1. Uživatel otevře profil po kliknutí na jméno uživatele v seznamu uživatelů.
2. Systém vyhledá a zobrazí takové informace o uživateli, na které má oprávnění.

UC08 – Upravit vlastní profil

Uživatel může upravit vlastní profil.

Scénář:

1. Include <Zobrazit vlastní profil>.
2. Uživatel klikne na tlačítko „Upravit“.
3. Systém zobrazí uživateli předvyplněné kolonky s údaji.
4. Uživatel upraví údaje a klikne na tlačítko „Potvrdit úpravy“.
5. Systém zobrazí uživateli profil se změněnými údaji.

Chybový scénář:

Pokud některá z kolonek bude prázdná, systém zabrání uložení do databáze a požádá uživatele o doplnění informací.

UC09 – Vyhledat uživatele

Uživatel může vyhledat jiného uživatele.

Scénář:

1. Uživatel otevře seznam uživatelů kliknutím na záložku „Zaměstnanci“ v navigaci.
2. Systém vyhledá a zobrazí všechny uživatele.
3. Uživatel zadá jméno a/nebo příjmení hledaného uživatele do kolonky pro vyhledávání a klikne na tlačítko „Vyhledat“.
4. Systém vyhledá a zobrazí uživatele odpovídajícího kritériím.
5. Include <Zobrazit profil jiného uživatele>.

UC10 – Zobrazit představení

Uživatel může zobrazit informace o představení.

Scénář:

1. Uživatel si ze seznamu vybere představení a klikne na jeho název.
2. Systém uživatele přesměruje na stránku s údaji o představení.

UC11 – Vyhledat představení

Uživatel může vyhledat představení.

Scénář:

1. Uživatel otevře stránku se seznamem všech představení kliknutím na záložku „Představení“ v navigaci.
2. Systém vyhledá a zobrazí seznam všech představení.
3. Uživatel do kolonky pro vyhledávání zadá název představení a klikne na tlačítko „Vyhledat“.
4. Systém vyhledá a vypíše do seznamu odpovídající představení.
5. Include <Zobrazit představení>.

UC12 – Zobrazit nástěnku se vzkazy

Uživatel může nástěnku se vzkazy.

Scénář:

1. Uživatel otevře nástěnku kliknutím na záložku „Nástěnka“ v navigaci.
2. Systém vyhledá a zobrazí všechny vzkazy s ikonkami pro úpravu a smazání.

UC13 – Přidat vzkaz na nástěnku

Uživatel může přidávat příspěvky na nástěnku.

Scénář:

1. Include <Zobrazit nástěnku se vzkazy>.
2. Uživatel klikne na „Přidat vzkaz“.
3. Systém uživatele přesměruje na formulář pro přidání nového vzkazu.
4. Uživatel vyplní všechny potřebné údaje a klikne na „Uložit vzkaz“.
5. Systém vzkaz uloží do databáze a přesměruje uživatele na stránku se vzkazy.

Chybový scénář:

Uživatel při vyplňování formuláře nevyplní kolonku „Obsah“. Systém vzkaz do databáze neuloží a uživatele vyzve k vyplnění kolonky.

UC14 – Upravit vlastní vzkaz na nástěnce

Uživatel může upravit svůj vzkaz na nástěnce.

Scénář:

1. Include <Zobrazit nástěnku se vzkazy>.
2. Uživatel klikne na ikonku pro úpravu.
3. Systém uživatele přesměruje na předvyplněný formulář.
4. Uživatel upraví údaje a klikne na „Uložit vzkaz“.
5. Systém změny uloží do databáze a přesměruje uživatele na nástěnku se vzkazy.

UC15 – Smazat vlastní vzkaz na nástěnce

Uživatel může smazat svůj vzkaz na nástěnce.

Scénář:

1. Include <Zobrazit nástěnku se vzkazy>.

2. Uživatel klikne na ikonku pro smazání.
3. Systém uživateli otevře okno pro potvzení smazání.
4. Uživatel potvrdí akci kliknutím na „Smazat".
5. Systém vzkaz smaže z databáze a přesměruje uživatele na nástěnku se vzkazy.

Oprávnění uživatelé

UC16 – Upravit profil jiného uživatele

Uživatel může upravit profil jiného uživatele.

Scénář:

1. Include <Zobrazit profil jiného uživatele>.
2. Uživatel klikne na tlačítko „Upravit".
3. Systém zobrazí uživateli předvyplněné kolonky s údaji.
4. Uživatel upraví údaje a klikne na tlačítko „Potvrdit úpravy".
5. Systém zobrazí uživateli profil se změněnými údaji.

Chybový scénář:

Pokud některá z kolonek bude prázdná, systém zabrání uložení do databáze a požádá uživatele o doplnění informací.

UC17 – Přidat představení

Uživatel může přidat nové představení.

Scénář:

1. Uživatel klikne na tlačítko „Přidat představení" buď hlavní stránce nebo v záložce „Představení".
2. Systém uživatele přesměruje na formulář pro přidání představení.
3. Uživatel vyplní potřebné údaje a klikne na tlačítko „Uložit představení".
4. Systém údaje uloží do databáze a přesměruje uživatele na formulář pro přidání rolí k představení.
5. Include <Přidat role k představení>.

UC18 – Přidat role k představení

Uživatel může přidat role k představení.

Scénář:

1. Uživatel vyplní název role.
2. Ke každé roli přidá uživatele kliknutím na tlačítko „Přidat alternaci".
3. Klikne na tlačítko „Uložit role".
4. Systém údaje uloží do databáze a uživatele přesměruje na stránku s detaily o představení.

UC19 – Upravit/smazat představení a role

Uživatel může upravit/smazat představení.

Scénář:

1. Include <Zobrazit představení>.
2. Pokud uživatel klikne na „Upravit", systém mu otevře formulář s předvyplněnými políčky a uživatel pak upraví údaje. Systém údaje uloží do databáze.
3. Pokud uživatel klikne na „Smazat", systém ho požádá o potvrzení akce. Uživatel akci potvrdí a systém údaje z databáze smaže.
4. Systém přesměruje uživatele na stránku s údaji o představení.

UC21 – Přidat směnu

Uživatel může přidat směnu.

Scénář:

1. Pokud chce uživatel přidat zkoušku, otevře stránku kliknutím na "Zkoušky" v navigaci a klikne na „Přidat zkoušku".
2. Pokud chce uživatel přidat představení, otevře stránku kliknutím na "Směny" v navigaci a klikne na „Přidat". Vybere buď představení nebo pronájem.
3. Systém ho přesměruje na přizpůsobený formulář výběru uživatele.
4. Uživatel vyplní potřebné údaje a odešle formulář kliknutím na „Uložit směnu".
5. Systém uživatele přesměruje na stránku s detaily směny.

UC22 – Upravit/smazat směnu

Uživatel může zobrazit profil jiného uživatele se všemi informacemi.

Scénář:

1. Include <Zobrazit kalendář>.
2. Pokud uživatel klikne na „Upravit", systém mu otevře seznam se všemi směnami, uživatel vybere směnu pak upraví údaje. Systém údaje uloží do databáze.
3. Pokud uživatel klikne na „Smazat", systém mu otevře seznam se všemi směnami, uživatel vybere směnu. Systém ho požádá o potvrzení akce. Uživatel akci potvrdí a systém údaje z databáze smaže.
4. Systém přesměruje uživatele na stránku s kalendářem .

UC23 – Přidat uživatele

Uživatel může zobrazit profil jiného uživatele se všemi informacemi.

Scénář:

1. Uživatel klikne na tlačítko „Přidat uživatele" v záložce „Zaměstnanci".
2. Systém uživatele přesměruje na formulář pro přidání uživatele.
3. Uživatel vyplní potřebné údaje a klikne na „Uložit".
4. Systém data uloží do databáze a přesměruje uživatele na profil přidaného.

UC22 – Upravit/smazat uživatele

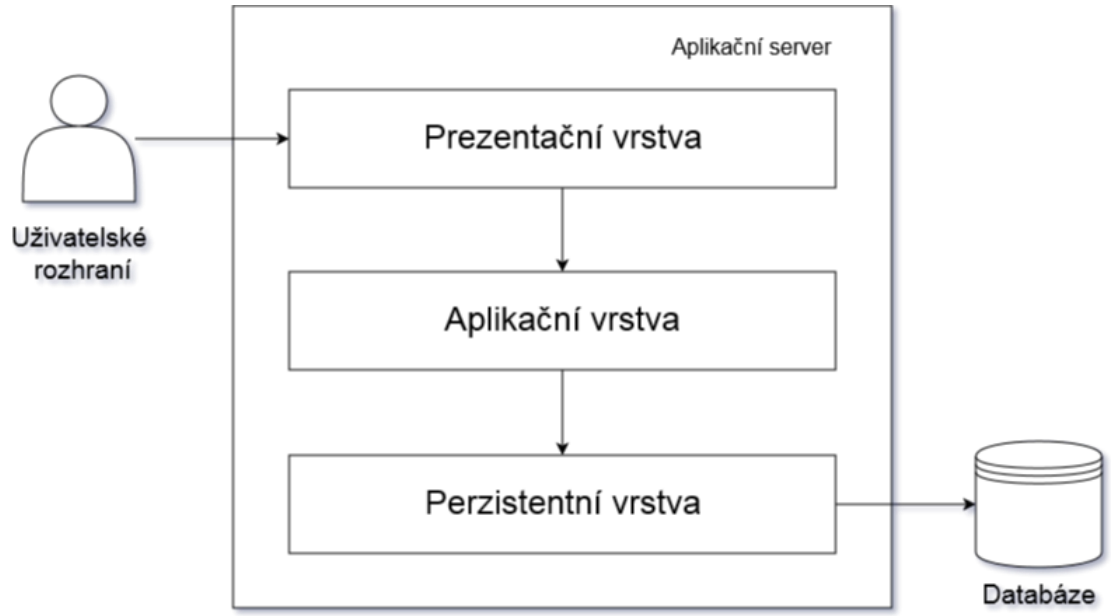
Uživatel může upravit/smazat uživatele po kliknutí na příslušné tlačítko na profilu uživatele.

Návrh systému

V této kapitole je popsána architektura systému a uveden datový model s popisem jeho entit a vztahů mezi nimi. Nakonec bylo navrženo grafické rozhraní a bylo uvedeno rozdělení jednotlivých stránek v navigaci podle jejich obsahu.

2.1 Architektura systému

Pro vývoj aplikace použijeme architekturu vícevrstvou (viz. Obrázek 4).



Obrázek 4 - Vícevrstvá architektura aplikace

Prezentační vrstva

Uživatel odešle požadavek přes uživatelské rozhraní aplikačnímu serveru. Na základě požadavku aplikace přistoupí ke kontrolerům na úrovni prezentační vrstvy, které dále přistupují ke službám (services) na úrovni vrstvy aplikační. Před zavoláním služeb dochází také k autentizaci a autorizaci uživatele.

Aplikační vrstva

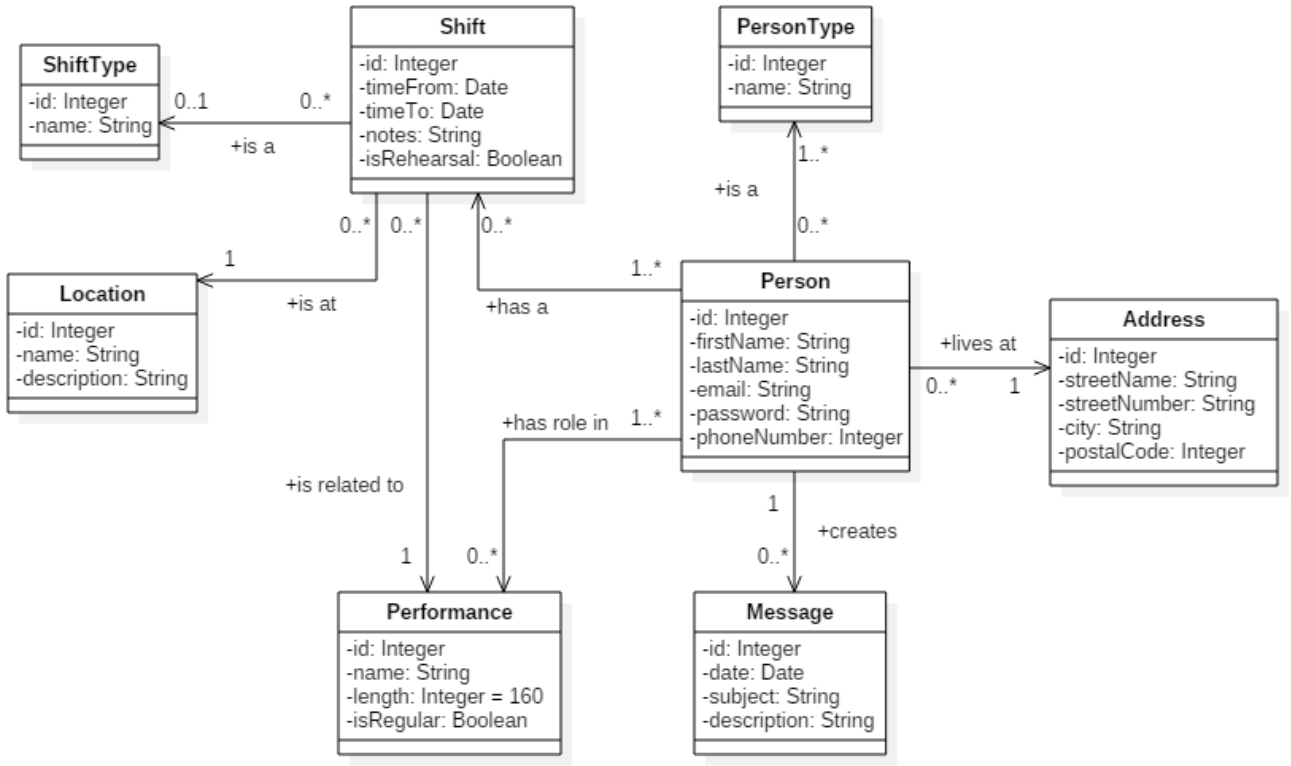
Služby zajišťují přes rozhraní přístup k datům perzistentní vrstvy a většinu aplikační logiky. Zároveň zde dochází například k ověřování hesla či existence uživatele.

Perzistentní vrstva

Tato vrstva nabízí přístup k úložišti dat přes tzv. DAO (Data Access Object), typicky za využití nějaké relační databáze (RDBMS) či objektově relační databáze (ORDBMS) za účelem zapouzdření veškerého přístupu ke zdroji dat.

2.2 Datový model

Pro převod datového modelu na logické schéma databáze byl vytvořen entitně-relační diagram (viz. Příloha B). Jednotlivé entity a vztahy mezi nimi popisuje doménový model tříd (viz. Obrázek 5).



Obrázek 5 - Doménový model tříd

2.2.1 Entity

Entita reprezentuje určitou část objektů reálného světa. Jedná se jak o objekty fyzické (například člověk) nebo abstraktní (například kategorie). Každá entita je popsána svým názvem a sadou atributů. Každý z těchto atributů pak má přiřazen datový typ. [5]

Person – Osoba

Entita Person reprezentuje zaměstnance divadla.

Atributy:

- **firstName**– křestní jméno zaměstnance,
- **lastName** – příjmení zaměstnance,
- **email** – e-mail zaměstnance,
- **password** – heslo k přihlášení zaměstnance,
- **phoneNumber** - telefonní číslo zaměstnance.

PersonType – Typ osoby

Každý zaměstnanec má svůj typ, tj. zaměření podle zaměstnání. Například herec/čka, muzikat/ka, technik, kostymérka, uvaděčka, atd.

Atributy:

- **name** – název typu osoby.

Address - Adresa

Každý zaměstnanec má přiřazenou adresu bydliště.

Atributy:

- **streetName** - název ulice, kde zaměstnanec bydlí,
- **streetNumber** – číslo popisné,
- **city** – název města,
- **postCode** – poštovní směrovací číslo města.

Message – Vzkaz na nástěnce

Každý zaměstnanec může přidávat vzkazy na společnou nástěnku.

Atributy:

- **date** - datum přidání vzkazu na nástěnku,
- **subject** – předmět vzkazu (nepovinný),
- **description** - obsah vzkazu.

Shift – Směna

Každému zaměstnanci může být přiřazena směna. Směny jsou vypisovány každý měsíc.

Atributy:

- **timeFrom** – čas, od kdy směna začíná,
- **timeTo** (do) – čas, kdy směna končí. Tento údaj je nepovinný, pokud se jedná o regularní představení, u kterého je čas konce přepočítán podle délky představení.

ShiftType – Typ směny

Směny rozlišujeme podle typu, ale pouze u regulárních představení. Typy jsou: premiéra, veřejná generálka, zkouška nebo zájezd.

Atributy:

- **name** - název typu směny.

Location – Místo konání

Každé představení (resp. směna) má své umístění. Výchozí umístění je v prostorách divadla SEMAFOR. Pokud se jedná o zájezd, umístění je možné změnit.

Atributy:

- **name** - název místa, kde se představení odehrává,
- **description** - popis místa.

Performance – Představení

Tato entita reprezentuje představení.

Atributy:

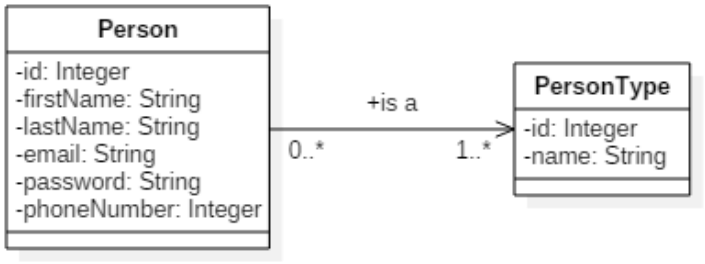
- **name** – název představení,

- **length** – délka představení (výchozí je 160 minut),
- **description** – popis představení (nepovinný),
- **isRegular** – určuje, zda se jedná o představení divadla SEMAFOR či pronájem.

2.2.2 Vazby mezi entitami

Vazba mezi entitami představuje logický vztah mezi entitami. Na vazbu můžeme pohlížet jako na dvě vazby v opačných směrech. V tomto smyslu se hovoří o takzvaných rolích, které představují pohled na danou vazbu ve směru od jedné entity k druhé. Ke každé roli přiřazujeme tzv. kardinalitu. Ta představuje omezení v počtu instancí druhé entity, které mají vztah s jakoukoliv instancí první entity. [5]

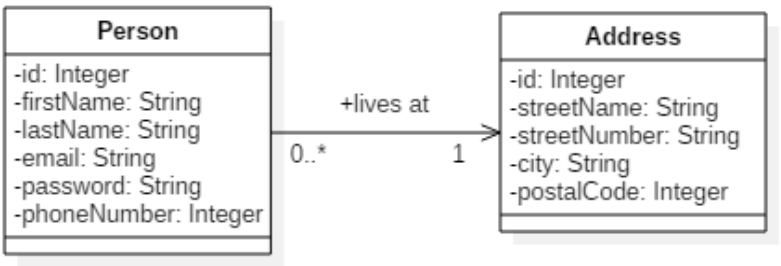
Is a (Person, PersonType)



Obrázek 6 - Vztah „Is a" mezi osobou a typem

Zaměstnanec má minimálně jedno zaměření, tj. může být například herečka a zároveň uvaděčka.

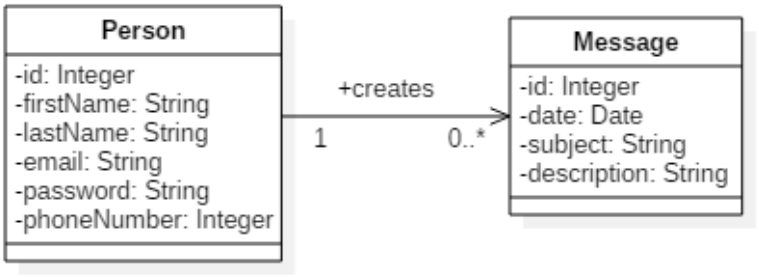
Lives at (Person, Address)



Obrázek 7 - Vztah „Lives at" mezi osobou a adresou

Každému zaměstnanci přísluší pouze jedna adresa bydliště.

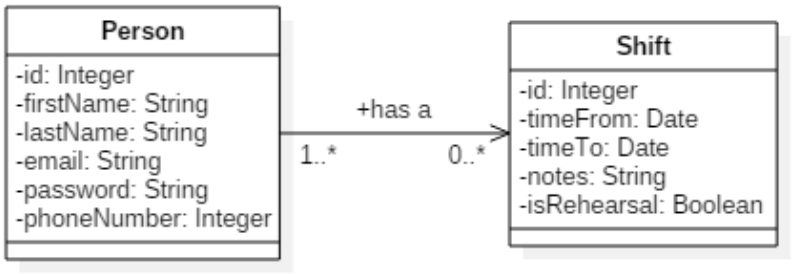
Creates (Person, Message)



Obrázek 8 - Vztah „Creates“ mezi osobou a vzkazem

Každý zaměstnanec může a nemusí vytvářet vzkazy na zdi.

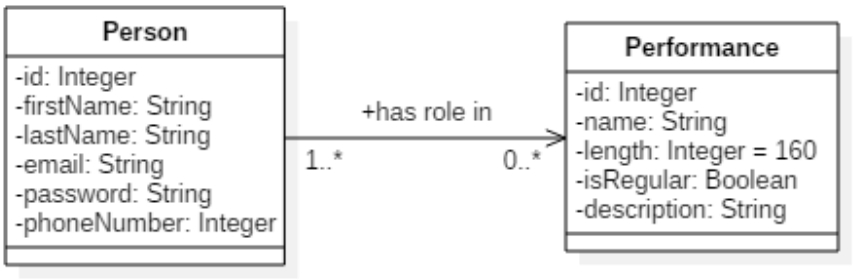
Has a (Person, Shift)



Obrázek 9 - Vztah „Has a“ mezi osobou a směnou

Každému zaměstnanci se přiřazují směny. Může se stát, že daný měsíc nemá směnu žádnou.

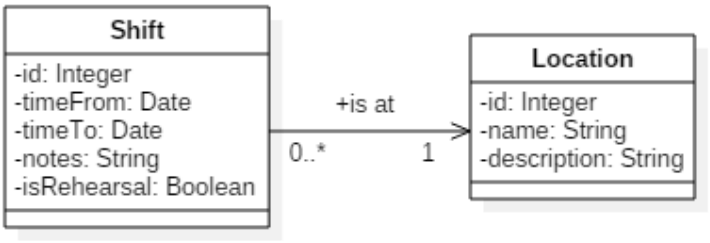
Has role in (Person, Performance)



Obrázek 10 - Vztah „Has role in“ mezi osobou a představením

Každý zaměstnanec může mít v představení nějakou roli.

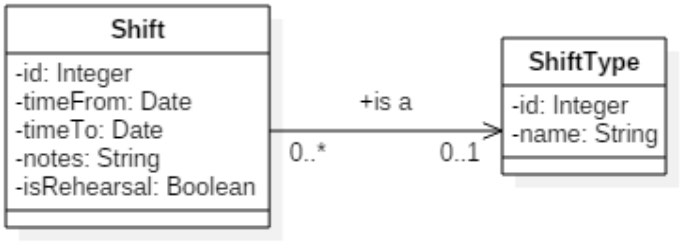
Is at (Shift, Location)



Obrázek 11 - Vztah „Has role in“ mezi osobou a představením

Každé představení má své umístění.

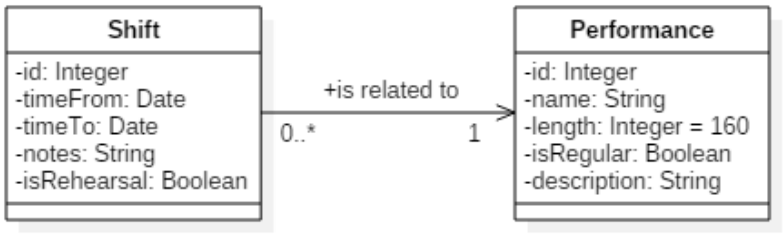
Is a (Shift, ShiftType)



Obrázek 12 - Vztah „Is a“ mezi směnou a jejím typem

Pokud se jedná o semaforské představení, můžeme představení/směně přiřadit speciální typ (premiéra, veřejná generálka, zájezd).

Is related to (Shift, Performance)

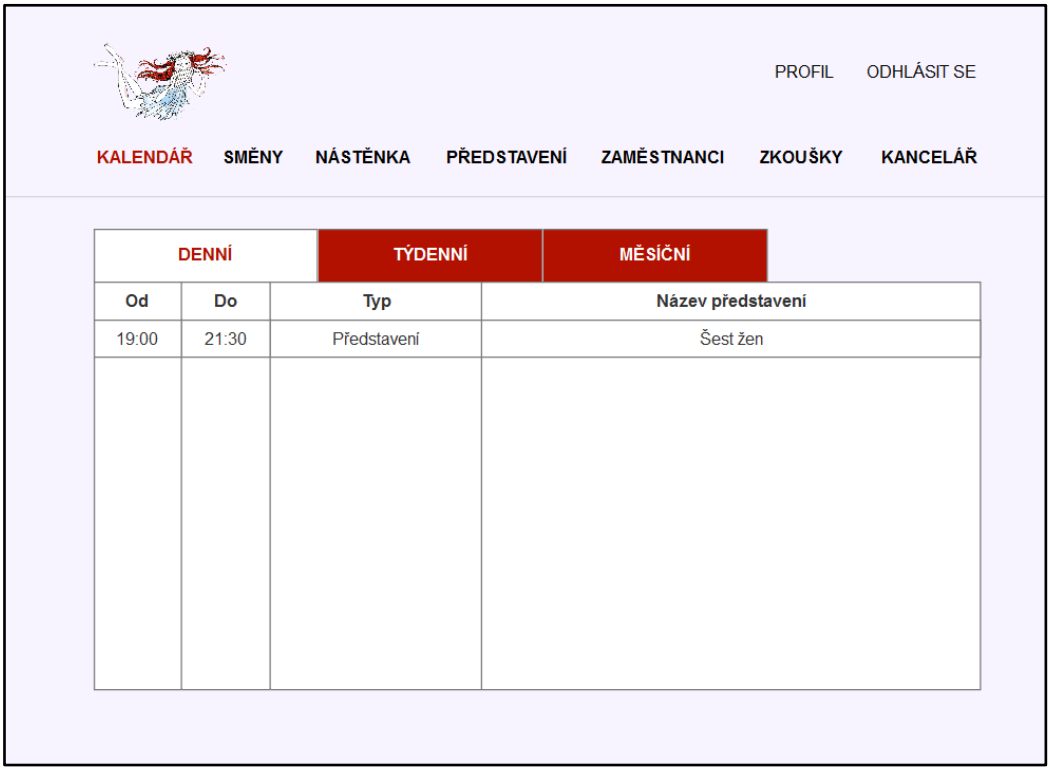


Obrázek 13 - Vztah „Is related to“ mezi směnou a představením

Každá směna je přiřazena pouze k jednomu představení.

2.3 Návrh grafického rozhraní

Při návrhu vzhledu portálu je dbáno na jednoduchost a funkčnost. Byl zvolen jednoduchý vzhled a přehledné menu v horní části stránky (viz. Obrázek 14).



Obrázek 14 – Návrh hlavní stránky aplikace

V rámci návrhu designu proběhl i výběr barev pro aplikaci (viz. Obrázek 15). Výběr správných barev je velmi důležitý, protože každá barva probouzí určitou emoci a symbolizuje různé věci. Jako hlavní barvu jsem vybrala tmavší odstín červené v kombinaci s blankytnou a odstíny šedé. Do prostředí divadla se červená hodí, jelikož je výrazem životní síly, aktivity či potěšení z činnosti a probouzí tvůrčí činnost. Může také vyvolat negativní emoce, jako je například vztek či rozzuřenost. Ale i to k divadlu patří. Na druhou stranu jsem vybrala blankytnou modř, která naopak značí klid a důvěru společně s neutrální šedou.

Obrázek 15 - Barevné schéma

2.3.1 Rozdělení obrazovek

Navigace obsahuje následující obrazovky:

Kalendář

- Denní rozpis
- Týdenní rozpis
- Měsíční rozpis
- Formulář pro přidání zvláštní akce

Směny

- Rozpis směn (vlastních i cizích)
- Formulář pro přidání/změnu směny podle typu

Nástěnka

- Dashboard příspěvků
- Formulář pro přidání/úpravu příspěvku

Představení

- Seznam představení
- Formulář pro přidání/úpravu představení
- Detail představení

Zaměstnanci

- Seznam zaměstnanců
- Vyhledávání zaměstnanců
- Přidání/úprava zaměstnance
- Zobrazení profilu zaměstnance

Zkoušky

- Rozpis vypsaných zkoušek
- Přidání/úprava zkoušky
- Zobrazení detailu zkoušky

Kancelář

- Seznam zaměstnanců kanceláře a jejich kontakty

Přihlašovací stránka

- Formulář pro přihlášení uživatele

Kapitola 3

Implementace

V této kapitole jsou vyjmenovány a popsány použité technologie pro vývoj aplikace. Následně jsou uvedeny příklady realizace jednotlivých vrstev architektury, konfigurace, zabezpečení aplikace a princip vývoje uživatelského rozhraní.

3.1 Použité nástroje a technologie

Na základě předchozího návrhu jsem zvolila následující technologie. Pro vývoj backendu jsem vybrala jazyk Java, protože s ním mám více zkušeností než s jazykem PHP. Pro vývoj frontendu jsem vybrala JavaScript a rozhodovala se mezi využitím knihovny React nebo frameworku Angular. Nakonec jsem zvolila React kvůli tomu, že jsem se chtěla naučit pracovat s Redux.

3.1.1 Java 8

Java se řadí mezi objektově orientované programovací jazyky. Zdrojový kód je nejprve zapsán do souborů končící na `.java` a poté jsou tyto soubory kompilovány do `.class` přes `javac` kompilátor. Tyto soubory pak obsahují tzv. bytecode, což je jazyk Java Virtual Machine (JVM). Příkaz `java` poté spustí celou aplikaci. Tento formát je nezávislý na architektuře počítače a může tedy běžet na libovolném zařízení, které má JVM [6]. Javu tvoří platformy jako Java ME (Java Mobile Edition) pro mobilní aplikace, Java SE (Java Standard Edition) pro desktopové aplikace či Java EE (Java Enterprise Edition) pro rozsáhlé distribuované systémy.

Java 8 s sebou přináší různé další funkčnosti, jako například Lambda výrazy, Date a Time API, Nashorn JavaScript engine, atd. [7].

3.1.2 Java EE 8

Java EE je součástí platformy Java rozšiřující Java SE o specifikace pro distribuované systémy a webové služby. Java EE API zahrnují technologie jako například Enterprise JavaBeans (EJB), JavaServer Pages (JSP), Java Server Faces (JSF), Servlet, Java Persistence API (JPA), atd. [8].

3.1.3 Maven

Apache Maven je systém pro správu projektů v jazyce Java [9]. Maven spravuje build projektu a jejich závislosti (dependencies) a dynamicky stahuje Java knihovny a Maven pluginy z repozitářů. Ty pak ukládá do lokální cache paměti. Pluginy zajišťují například kompilaci projektů do výsledných balíčků (jar, war), nasazení na servery či testování. Veškeré konfigurace a závislosti projektu poskytuje POM (Project Object Model) v souboru `pom.xml`. V tomto souboru jsou informace o projektu, jeho podprojektech či samotné závislosti.

3.1.4 Tomcat

Apache Tomcat, označován jako Tomcat Server implementuje JavaEE specifikace, jako Java Servrlet, JavaServer Pages, Java Expression Language a WebSocket technologie [10].

3.1.5 PostgreSQL

PostgreSQL je objektově-relační databázový systém (ORDBMS). Běží na všech hlavních operačních systémech a například pro Javu poskytuje rozhraní JDBC [11].

3.1.6 Spring

Spring je framework pro platformu Java. Obsahuje také rozšíření pro webové aplikace stavěné na Java EE. Stal se populární především kvůli doplnění či nahrazení Enterprise JavaBean (EJB).

3.1.7 Node.js

Node.js je platformně nezávislé JavaScriptové prostředí, které umožňuje spouštět kód psaný v JavaScript na straně serveru. Skripty spouští k vytvoření dynamických webových stránek ještě předtím, než je stránka odeslána webovému prohlížeči uživatele [12].

3.1.8 ReactJS

React je JavaScript knihovna určena pro vývoj uživatelských rozhraní. React nabízí elementy, které reprezentují standardní HTML elementy (např. div, img) nebo vývojáři umožňuje vytvářet vlastní. Renderováním HTML elementů pomocí JavaScript vytvoří React virtuální reprezentaci HTML v paměti, neboli Virtual DOM (Document Object Model), což znamená, že React nejprve renderuje HTML strom virtuálně a pak pokaždé při změně stavu (state) místo přepsání celého stromu, React přepíše pouze odlišnosti mezi starým a nově vytvořeným stromem. U React rozlišujeme vlastnosti (props) a stavy (states). Vlastnosti komponenta obdrží jako parametr k renderování a stavy udržuje a předává svým potomkům jako vlastnosti [13]. Ke snadnější manipulaci se stavy slouží kontejner Redux nebo architektura Flux.

3.2 Konfigurace

V prvé řadě je potřeba ve výše zmíněném souboru `pom.xml` buildovacího nástroje Maven definovat všechna rozšíření. Hlavní komponenta tohoto souboru je `<project>`, která obsahuje mimo jiné například `<groupId>`, `<artifactId>` nebo verze rozšíření v `<properties>`. Hlavní jsou však tzv. závislosti definované v `<dependencies>`, jejichž struktura je následující:

```
@Configuration
@ComponentScan(basePackages = "upd.service")
public class ServiceConfig {

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Stejně důležitá je také komponenta `<build>`, která zahrnuje i `<plugins>`, definující jednotlivé pluginy.

Všechny tyto pluginy, knihovny a frameworky se pak při build stáhnou, pokud však nejsou uloženy lokálně.

Konfigurační soubory frameworku Spring jsou označeny anotací `@Configuration` označující třídy, které deklarují jednu nebo více `@Bean` metod a lze je použít jako zdroj tzv. bean definitions. Bean je objekt řízený Spring kontejnerem a je vytvořen metadaty, které kontejneru poskytujeme. Anotací `@Bean` oznamujeme, že metoda vrátí objekt registrovaný jako bean v rámci Spring application context.

Konfigurační třídě můžeme přidat anotaci `@Import` umožňující načtení bean definitions z jiného konfiguračního souboru. Dále můžeme použít volitelnou anotaci `@ComponentScan(package_name)`, která v rámci tohoto package oskenuje všechny třídy a vytvoří beans v rámci `@Component` anotace [14]. Příklad konfigurační třídy:

```
<dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${org.springframework.version}</version>
</dependency>
```

Podstatné je také zajistit závislosti bean, které v našem případě umožní Servlet třídou `AbstractAnnotationConfigDispatcherServletInitializer`. Ta v metodě `getRootConfigClasses()` vrací konfigurační třídy importované námi vytvořenou třídu `AppConfig.java` s anotací `@EnableMBeanExport`.

3.3 Uživatelské rozhraní

Uživatelské rozhraní je implementováno v jazyce JavaScript s využitím knihovny ReactJS a Reflux. Obdobně jako u POM, moduly, které chceme požít, jsou vypsány v souboru `package.json`. Tyto moduly pak nainstalujeme a spravujeme díky Node.js a správci modulů NPM. Spuštěním příkazu `npm install` stáhneme všechny Node závislosti. Dále v našem případě příkazem `npm start` zahájíme kompilování kódu, čímž vytvoříme bundle, který je po kompilaci spuštěn aplikačním serverem.

Aplikace obsahuje pouze jedinou HTML stránku `index.html` s elementem, do kterého se renderuje dynamický obsah vytvářený React knihovnou. K renderování slouží metoda `ReactDOM.render()`, jejíž první parametr je komponenta, kterou má vykreslit a druhý je element, do kterého má obsah vepsat. V našem případě je první argument komponenta `<BrowserRouter>` s potomkem `<Layout/>`. Tato komponenta zajišťuje přesměrování stránek společně s kombinací komponent `<Link>`, `<Switch>` a `<Route>`.

Komunikace mezi uživatelem (klientem) a samotnou aplikací (serverem) probíhá přes protokol HTTP s použitím AJAX API knihovny Axios.

Obrázek 16 - Ukázka uživatelského rozhraní aplikace

3.4 Zabezpečení

Zabezpečení aplikace zajišťuje Spring Security, který poskytuje komplexní zabezpečení služeb. Konfigurační třída je označena anotací `@EnableWebSecurity` pro povolení podpory zabezpečení webových aplikací a k poskytnutí integrace Spring MVC. Třída `WebSecurityConfigurerAdapter` má metodu `configure(HttpSecurity)` definující ta URL, která mají být zabezpečena a která nikoliv [15]. Zjednodušená verze metody:

```
@Override
protected void configure(HttpSecurity http) {
   http
    .authorizeRequests().anyRequest().authenticated().and()
    .formLogin()
     .loginProcessingUrl(„/j_spring_security_check“)
     .usernameParameter(„email“).passwordParameter(„password“).and()
    .logout().logoutUrl(„/j_spring_security_logout“);
}
```

3.5 Prezentační vrstva

Na úrovni prezentační vrstvy jsou HTTP požadavky posílané uživatelem přes uživatelské rozhraní odchyceny kontrolory označené anotací `@RestController` a dále zpracovány. Anotace `@RequestMapping` pak zajišťuje mapování požadavku na správný kontroler. Příklad pro metodu GET:

```
@RequestMapping(method = RequestMethod.GET)
public List<Person> getAll() {
      return personService.findAll();
}
```

Služba, v tomto případě instance služby `PersonService`, je označena jako `@Autowired` kvůli dependency injection u bean. Příklad pro metodu POST:

```
@Service
public class PersonService {

  @Autowired
  private PersonDao personDao;

  public void persist(Person person) {
      personDao.persist(person);
  }
}
```

Anotace `@RequestBody` u parametru funkce slouží k přístupu k tělu požadavku. Obsah těla je převeden na deklarovaný argument díky metodě `HttpMessageConverter`. Jako anotaci k parametru můžeme použít i například `@PathVariable` nebo `@RequestParam` pro získání informací z URL.

3.6 Aplikační vrstva

Aplikační vrstva zprostředkovává služby umožňující přístup k perzistentní vrstvě. Třídy jsou označeny anotací `@Service`, která má stejnou funkci jako `@Component`. Příkladem služby je již výše zmíněná třída `PersonService`:

```
@RequestMapping(method = RequestMethod.POST)
public ResponseEntity<Void> create(@RequestBody Person person) {
      personService.persist(person);
      final HttpHeaders headers = RestUtils
         .createLocationHeaderFromCurrentUri("/{email}", person);
      return new ResponseEntity<>(headers, HttpStatus.CREATED);
}
```

3.7 Perzistentní vrstva

K přístupu do databáze slouží perzistentní vrstva obsahující DAO a model samotné DB pro manipulaci s entitami jako s objekty. DAO poskytuje veškeré operace pro správu dat uložených v DB. Třídy se označují anotací `@Repository` a obdobně jako `@Service`, mají stejnou funkci jako `@Component`. Příklad třídy reprezentující perzistentní vrstvu:

```
@Repository
public class PersonDao {
  @PersistenceContext
  EntityManager em;

  @Transactional
  public void persist(Person person) {
      try {
        em.persist(person);
      } catch (RuntimeException e) {...}
  }
}
```

`EntityManager` je rozhraní interagující s `@PersistenceContext`. Persistence context je množina unikátních instancí entit spravující jejich životní cykly. `EntityManager` API vytváří a odstraňuje tyto instance a dále zajišťuje vykonávání příkazů nad entitami. Anotace `@Transactional` jako taková definuje databázovou transakci, která je poté vykonána v rámci persistence context.

Testování

V této kapitole je popsán postup a způsoby testování, zejména tedy testování manuální v aplikaci Postman a přes UI, systémové JUnit testy a testování prototypu s uživatelem.

4.1 Manuální testování

Prototyp aplikace byl manuálně otestován v aplikaci Postman a některé funkcionality i přes uživatelské rozhraní. Aplikace Postman umožňuje odesílání požadavků na server aplikace a odpovědi zobrazuje ve formátu JSON. Nejprve byla do databáze vložena data obsahující základní informace, jako například seznam zaměstnanců, představení, role zaměstnanců, druhy představení či směny. Poté jsem posílala GET požadavky na server aplikace a u přijatých JSON struktur kontrolovala, zda obsahují všechna potřebná data. Dále jsem odeslala pár POST požadavků, abych zjistila, zda jsou data správně vkládána do databáze.

Po otestování backendu následovalo manuální testování uživatelského rozhraní. Především tedy zda se uživateli zobrazují aktuální informace, zda jsou případné chyby odchyceny a jestli fungují správně všechny formuláře v prototypu.

4.2 Unit testy

Tyto testy slouží k testování menších částí zdrojového kódu, zejména tedy jednotlivých metod. Ověřují, že se metody chovají správně i během mimořádných situací, například pokud na vstupu metoda dostane jako argument null. K jednotkovému testování naší aplikace jsem použila framework JUnit. Příklad jednoduchého testu v JUnit:

```
@Rule
public ExpectedException thrown = ExpectedException.none();

private Person person;

@Before
public void setUp() { this.person = new Person(); }

@Test
public void nameEqualsReturnsFalseForNullOtherPerson() {
      user.setName("Jmeno");
      user.setSurname("Prijmeni");
      assertFalse(person.nameEquals(null));
}
```

Tento způsob testování je důležitý především pro oddělení funkcionalit od testů a pro automatizaci testů. Metoda s anotací `@Before` se spustí před každým testem v této třídě.

Proto v ní vytváříme instanci testované třídy. Samotné testovací metody jsou označené anotací `@Test`. S rozhraním `@Rule` můžeme také měnit chování každé testovací metody. Třída `ExpectedException` umožňuje ověřit, zda testovaná metoda vyhodí konkrétní výjimku. Po definování očekávané výjimky bude test úspěšný, pouze pokud je vyhozena tato výjimka a neúspěšný, pokud bude vyhozena jiná či žádná výjimka.

4.3 Uživatelské testování

Uživatelského testování se zúčastnili celkem tři zástupci divadla SEMAFOR. Uživatelé dostali k dispozici prototyp aplikace a měli splnit následující úkoly:

1. Nalezněte rozpis všech směn.
2. Přidejte novou směnu.
3. Přidejte nové představení.
4. Přidejte nového zaměstnance.
5. Přidejte příspěvek na nástěnku.
6. Zobrazte svůj profil.
7. Zobrazte profil jiného zaměstnance.
8. Zobrazte informace o představení.
9. Zobrazte vlastní směny.

Po splnění úkolů jsem se uživatelů dotazovala, jak byli s používáním aplikace spokojeni a co by naopak změnili či přidali.

4.3.1 Výsledky testování

Uživatelské testování proběhlo v kanceláři divadla, kde všichni přítomní testovali jako skupina a každý řekl ke každému úkolu nějaké připomínky. Aplikace běžela na lokálním serveru a byla testována v prohlížeči Google Chrome 66.
plnění jednotlivých úkolů probíhalo následovně:

1. Uživatel nalezl směny hned na úvodní stránce Kalendář. Pro jistotu se podíval i do záložky Směny, kde nalezl rozpis svých směn.

   Připomínka: Uživatel si přeje filtrování v kalendáři.

2. Na stránce Kalendář klikl na „Přidat směnu“, vyplnil formulář a záznam uložil.

   Připomínka: Uživatel by chtěl při přidávání nové směny rovnou směně přiřazovat i uživatele a jakou funkci v rámci této směny uživatel má. Tato funkcionalita je nyní oddělena v sekci Směny a to uživateli nepřijde intuitivní. Také chce datum a čas jako rozdělená políčka.

3. Uživatel našel v sekci Představení tlačítko „Přidat představení“.

   Připomínka: Uživateli chybí přidávání rolí k představení.

4. Uživatel našel v sekci Zaměstnanci tlačítko „Přidat zaměstnance“.

   Připomínka: Uživateli chybí přidávání funkcí uživatele a rolí k představení.

5. Uživatel našel v sekci Nástěnka tlačítko „+“, vyplnil formulář a záznam uložil.
6. Uživatel zobrazil svůj profil kliknutím na „PROFIL“.
7. Uživatel v sekci Zaměstnanci klikl na „Detaily“ a profil se mu zobrazil.
8. Uživatel v sekci Představení klikl na „Detaily“ a informance o představení se mu zobrazily.
9. Uživatel nalezl své směny v záložce Směny, kde mu aplikace všechny jeho směny při načtení zobrazila.

Kapitola 5

Závěr

V rámci práce byl po předchozí analýze implementován a otestován informační systém pro organizaci divadel. Nejprve byly definovány požadavky a případy užití a byl vytvořen datový model aplikace, na základě kterého bylo vytvořeno schéma databáze. Serverová část prototypu aplikace byla implementována v jazyce Java za použití Maven a Spring. Komunikace s PostgreSQL databází probíhá přes rozhraní JDBC a JPA. Frontend aplikace byl implementován v JavaScript s využitím Node.js a knihovny React/Redux. Aplikace byla vyvíjena v IntelliJ IDEA a spuštěna na aplikačním serveru Tomcat.

Analýzu jsem vytvořila bez ohledu na to, co jsem se rozhodla v rámci práce později implementovat. Nakonec jsem se rozhodla pro implementaci základních funkčností, jako je přihlašování, kalendář (bez filtrování směn), správa směn, představení, zaměstnanců a nástěnky. Při dalším vývoji je tedy možno analýzu využít.

V průběhu práce jsem uplatnila znalosti z předmětů a zároveň jsem některé informace a návody vyhledávala na internetu. Především informace týkající se vývoje v Redux, se kterým jsem neměla žádné zkušenosti. S výběrem Reactu jako frontendové knihovny a využitím Redux jsem velmi spokojená, jelikož usnadňují manipulaci s DOM a generování dynamických stránek jako komponent.

Naopak s výběrem jazyku Java a JPA na straně serveru spokojená nejsem a to především kvůli mapování Java tříd jako entit na entity v databázi, kdy jsem měla problém s obousměrnou vazvou many-to-many s přidaným atributem ve vazební tabulce. Pozitivum však byl framework Spring Security, který zajišťoval veškeré zabezpečení aplikace.

5.1 Budoucí vývoj aplikace

V nejbližší době budou dokončeny zbylé funkčnosti uvedené v požadavcích, jako jsou například přístupy uživatelů do aplikace na základě oprávnění, responzivní vzhled, odhlašování, přidávání rolí k představením, úpravu vzkazů na nástěnce a přístup k zobrazení obsazení směn na aktuální den v kalendáři.

Ke stávajícím požadavkům bych dále chtěla připojit i možnost změny hesla uživatele či vyhledávací políčko v záhlaví webu, které by uživateli umožnilo fulltextové vyhledávání.

5.1.1 Automatické generování směn

Pro usnadnění plánování směn bude vytvořena funkce, která by na základě směn z předchozích měsíců a datumů, kdy herci/muzikanti mohou či nemohou hrát, vygenerovala seznam všech možných kombinací směn na příští měsíc.

Data do systému budou přímo zadávat herci/muzikanti a tajemník zadá kolikrát se bude dané představení v měsíci hrát.

5.1.2 Export/import směn z Excelu

Kromě ručního vyplňování formulářů v systému bude tajemníkovi umožněno importovat seznam směn vytvořený v Excelu. Zároveň bude mít možnost směny exportovat ze systému do Excel tabulky či přímo vytisknout.

5.1.3 Propojení s Google Calendar

Pokud uživatel do systému zadá Google e-mail, budou mu automaticky všechny jeho směny přidávány do Google kalendáře.

5.1.4 PWA (Progressive Web Application)

Progressive Web Application je označení pro "rozšíření" klasických webových aplikací o funkčnosti umožňující zacházet s webovou aplikací jako s nativní. Jelikož většina uživatelů bude přistupovat do systému z webových prohlížečů v mobilu, bude mu umožněno uložit aplikaci na obrazovku telefonu a po kliknutí na ikonku otevřít jako klasickou mobilní aplikaci. Také bude možné aplikaci otevřít a zobrazit naposledy načtená data i bez připojení k internetu. Pokud by uživatel do systému nějaká data přidal, byla by uložena do cache paměti a po připojení k síti uložena do databáze.

Literatura

[1] GOMAA, Hassan. *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge [U.K.]: Cambridge University Press, 2011. ISBN 978-0-521-76414-8 [Cit. 15.10.2017].

[2] RADNER, S. *Key words for use in RFCs to Indicate Requirement Levels* [online]. [Cit.15.10.2017] Dostupné z: https://www.ietf.org/rfc/rfc2119.txt

[3] KOMÁREK, Martin. *Analýza a dokumentace požadavků* [přednáška]. Praha, ČVUT Fakulta elektrotechnická, březen 2016. In: *moodle.fel.cvut.cz* [online]. [Cit.18.10.2017]. Přednáška dostupná z: https://moodle.fel.cvut.cz/pluginfile.php/42038/mod_resource/content/3/ANALYZA_A DOKUMENTACE POZADAVKU_3PREDNASKA.pdf

[4] COCKBURN, Alistair. *Use Cases: jak efektivně modelovat aplikace*. Brno: Computer Press, 2005. ISBN 80-251-0721-3 [Cit. 18.10.2017].

[5] HAJN. *Úvod do datového modelování* [online]. [Cit. 21.11.2017]. Přednáška dostupná z: www.fi.muni.cz/~hajn/vyuka/DB%20systemy%20a%20aplikace/%davod_datov%e9_modelov%e1n%ed.doc

[6] ORACLE, *About the Java Technology* [online]. [Cit.11.1.2018] Dostupné z: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

[7] ORACLE, *Java 8 informations* [online]. [Cit. 7.1.2018] Dostupné z: https://www.java.com/en/download/faq/java8.xml

[8] ORACLE, *Overview of Enterprise Applications* [online]. [Cit.11.1.2018] Dostupné z: https://docs.oracle.com/javaee/7/firstcup/java-ee001.htm

[9] SONATYPE. *Maven: The Definitive Guide*. Sebastopol, USA: O'Reilly Media, 2008. [Cit. 7.1.2018]

[10] APACHE, *Apache Tomcat* [online]. [Cit.7.1.2018] Dostupné z: http://tomcat.apache.org/index.html

[11] PostreSQL [online]. [Cit.7.1.2018] Dostupné z: https://www.postgresql.org/about

[12] MDN web docs, *Node.js* [online]. [Cit.11.1.2018] Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side

[13] LEDVINKA, Martin. *React* [přednáška]. Praha, ČVUT Fakulta elektrotechnická, 2016. In: *cw.fel.cvut.cz* [online]. [Cit. 7.1.2018]. Přednáška dostupná z: http://cw.fel.cvut.cz/wiki/_media/courses/b6b33ear/lecture-07-react-s.pdf

[14] Omar El Gabry, *A Head Start – Beans Configuration (Part 2)* [online][12.9.2017]. [Cit.12.2.2018] Dostupné z: https://medium.com/omarelgabrys-blog/spring-a-head-start-beans-configuration-part-2-4a8c239b070a

[15] Spring, *Securing a Web Application* [online]. [Cit.12.2.2018] Dostupné z: https://spring.io/guides/gs/securing-web/

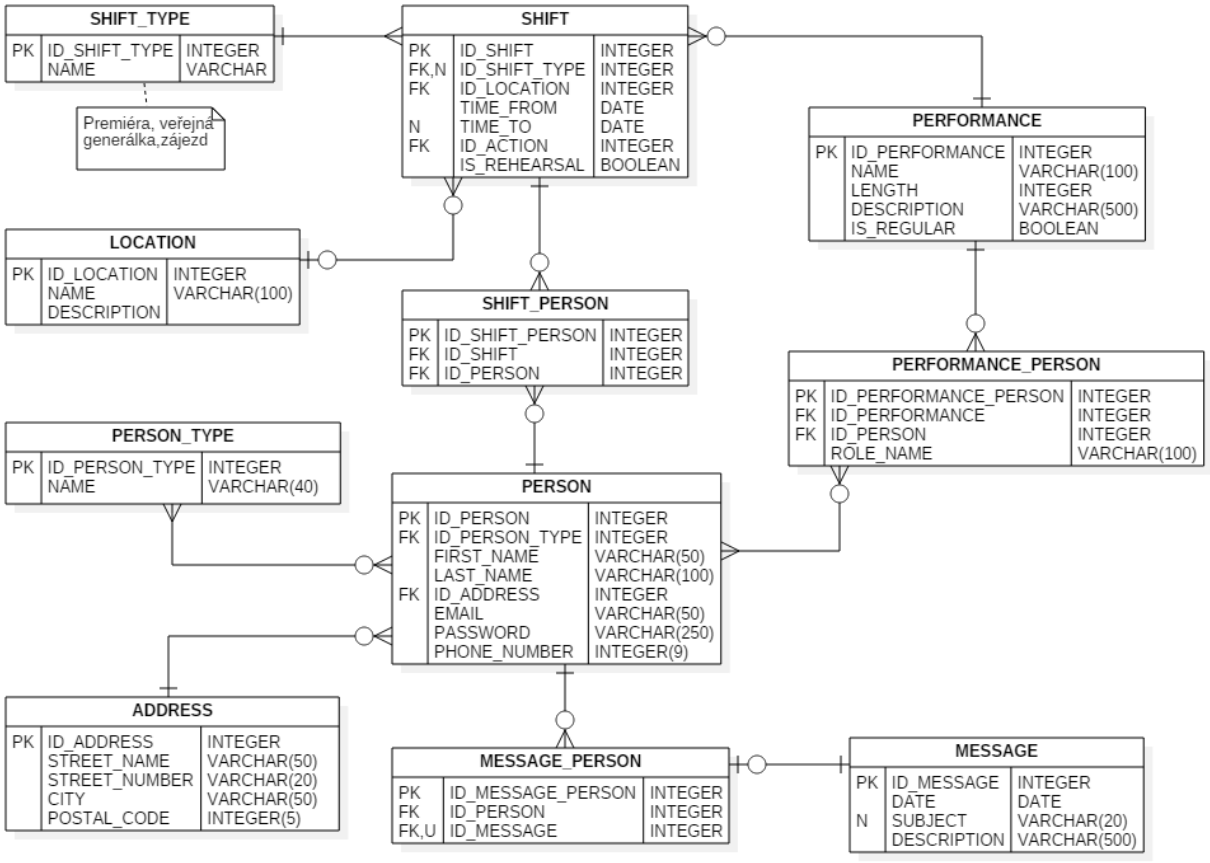
Otázky pro průzkum v divadlech

Níže jsou vypsány obecné otázky kladené všem zástupcům divadel. Okruhy jsou pouze přibližné, jelikož byly jednotlivé otázky přizpůsobeny osobám, se kterými byl rozhovor veden, případně byly nějaké otázky přidány. Otázky jsou následující:

1. Jaká je Vaše funkce v divadle?
2. Jak probíhá Váš běžný den v divadle?
3. Jak probíhá komunikace mezi Vámi a ostatními zaměstnanci?
4. Jak mezi sebou domlouváte a sdílíte směny?
5. Používáte nějaký nástroj pro usnadnění umělecké organizace?
6. Pokud ano, jaké funkcionality Vám nejvíce usnadňují práci?
7. Pokud ne, myslíte, že by Vám takový nástroj usnadnil práci?
8. Jaké funkcionality byste si představoval/a u takového nástroje?
9. Jaké jsou výhody a nevýhody současné organizace?
10. Co byste případně chtěla na současné organizaci změnit?

Entitně-relační diagram

Na obrázku níže (Obrázek 17) je entitně-relační diagram znázorňující entity a vztahy mezi nimi. Na základě tohoto diagramu bylo vytvořeno schéma databáze.



Obrázek 17 – Příloha A: ER diagram

Instalační příručka

Spuštění zkopírováním war souboru na server Tomcat

1. Nainstalujte server Tomcat.
2. Zkopírujte `upd-1.0.0.war` z CD do adresáře *webapps* serveru Tomcat.
3. Spusťte server Tomcat skriptem `startup.sh` (nebo `startup.bat`) v *bin* adresáři serveru Tomcat.
4. Otevřete v prohlížeči `http://localhost:8080/upd-1.0.0`.
5. Do přihlašivacího formuláře zadejte User: test@test.cz, Password: test.
6. Klikněte na Login a poté budete přesměrováni na hlavní stránku aplikace.

Zkompilování zdrojových souborů

Požadavky na Software

- Java 8
- Apache Maven 3
- NodeJS 6
- npm
- PostgreSQL 9.5
- Apache Tomcat 8

Nastavení aplikace

```
start `psql`
`CREATE USER upd WITH PASSWORD 'upd';`
`CREATE DATABASE upd WITH OWNER upd;`
```

PostgreSQL poběží na portu **5432** a Apache Tomcat na portu **8080**.

1. Vytvořte databázi pojmenovanou upd vlastněnou uživatelem *upd* s heslem *upd*.
2. V terminálu otevřete adresář *src/main/webapp* a spusťte příkaz `npm install`. Tento skript stáhne všechny potřebné JS závislosti.
3. V tom samém adresáři spusťte příkaz `npm run build`.
4. Vraťte se zpět do hlavního adresáře a spusťte příkaz `mvn clean package`. Tento skript stáhne všechny potřebné Java závislosti a vytvoří war soubor.
5. Zkopírujte soubor `upd.war` ze složky target do adresáře *webapps* serveru Tomcat.
6. Spusťte server Tomcat skriptem `startup.sh` (nebo `startup.bat`) v *bin* adresáři serveru Tomcat.
7. Otevřete v prohlížeči `http://localhost:8080/upd`.
8. Do přihlašivacího formuláře zadejte User: test@test.cz, Password: test.
9. Klikněte na Login a poté budete přesměrováni na hlavní stránku aplikace.

Obsah přiloženého CD

Součástí bakalářské práce je i CD s následujícím obsahem:

```
Projekt
|
|_ _ src
|     |
|     |_ _ main
|     |      |
|     |      |_ _ java – serverová část aplikace
|     |      |
|     |      |_ _ resources – konfigurační soubory
|     |      |
|     |      |_ _ webapp – uživatelská část aplikace
|     |
|     |_ _ test
|
|_ _ pom.xml
|
|
upd-1.0.0.war – zkompilovaná verze aplikace
|
|
BP_zlamalova_anna_2018.pdf – elektronická verze práce ve formátu PDF
```