Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Podniková sociální síť – back end

Bc. Denisa Plecháčková

Diplomová práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Denisa Plecháčková**
Osobní číslo:	**I16236**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Podniková sociální síť - back end**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce je vytvořit back-end část systému pro tvorbu podnikových sociálních sítí, přičemž bude kladen důraz na komponentově orientovanou architekturu celého systému. Podmínkou při tvorbě je důraz na bezpečnost API, vytvoření autentizačních i autorizačních mechanismů dle práv (rolí). Předpokladem je využití externích zdrojů pro přihlášení, není nutné budovat vlastní registrační systém a prvky s tím související.
Předpokladem je využití databáze MySQL v nejnovější dostupné verzi s JDBC, případně Spring. Pro zápis logické vrstvy aplikace je předpokladem využití jazyka Java v nejnovější verzi a zápis kódu s využitím návrhových vzorů. Nezbytnou součástí práce je správně vedená dokumentace zdrojových kódů.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **60**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. MySQL profesionálně: optimalizace pro vysoký výkon. Brno: Zoner Press, 2009. ISBN 978-80-7413-035-9.
2. BÖHMER, Marian. Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: Computer Press, 2012. ISBN 978-80-251-3338-5.
3. BURKE B. RESTful Java with JAX-RS 2.0. O'Reilly Media, 2013, 392 pp. ISBN 978-1449361341.

Vedoucí diplomové práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **30. října 2017**

Termín odevzdání diplomové práce: **18. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2017

Prohlašuji:

Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury. Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 1. 2017

Denisa Plecháčková

PODĚKOVÁNÍ

Děkuji svému vedoucímu diplomové práce, panu doc. Ing. Michaelu Bažantovi, Ph.D., který mi byl vždy nápomocen při psaní této práce. Děkuji i za cenné rady pana Ing. Petra Moravce. Nakonec chci také poděkovat svojí rodině, která mi byla velkou oporou po celou dobu studia.

ANOTACE

Práce ukazuje postup při vývoji podnikové sociální sítě podle specifických požadavků zadavatele. Součástí práce je i rešerše dostupných řešení na trhu a jejich popis. Následuje kapitola o zvolené metodice vývoje. Poté jsou v kapitolách probrány jednotlivé fáze – požadavky, analýza, návrh, implementace a testování. Aplikace je naprogramována v programovacím jazyku Java a s použitím frameworku Spring. Aplikace běží na aplikačním serveru Apache Tomcat a využívá databázi MySQL.

KLÍČOVÁ SLOVA

Podniková sociální síť, back end, Spring, Java, Unified Process, MySQL, Maven

TITLE

Enterprise social network – back end

ANNOTATION

The thesis shows how to develop an enterprise social network, according to the specific requirements of the contract owner. Part of this thesis is research of available solutions on the market and its descriptions. The following is a chapter on the chosen development methodology. Then, the chapters describe the individual phases - requirements, analysis, design, implementation and testing. Application is programmed in the Java programming language and with the usage of Spring framework. The application runs on the Apache Tomcat application server and uses the MySQL database.

KEYWORDS

Enterprise social network, back end, Spring, Java, Unified Process, MySQL, Maven

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

API	Application Programming Interface
CAS	Central Authentication Service
CRUD	Create, Read, Update, Delete
CSRF	Cross-site Request Forgery
DAO	Data Access Object
DBMS	DataBase Management System
DDL	Data Definition Language
ER	Entity Relation
ESN	Enterprise Social Network
HR	Human Resources
HTML	HyperText Markup Language
IBM	International Business Machines Corporation
ID	IDentifier
IETF	Internet Engineering Task Force
IT	Information Technology
JAAS	Java Authentication and Authorization Service
JDBC	Java DataBase Connectivity
JOSSO	Java Open Source Single Sign-On
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MVC	Model View Controller
POM	Project Object Model
REST	REpresentational State Transfer
RSG	Real Story Group
RSS	Rich Site Summary
RUP	Rational Unified Process
UML	Unified Modeling Language
UP	Unified Process
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

ÚVOD

Dnešní doba nabízí celou řadu možností, jak ve společnostech získávat, zpracovávat, třídit, ukládat a sdílet data. S vývojem technologií se tyto možnosti rozšiřují a umožňují rychleji uspokojit poptávku na trhu. S tím úzce souvisí zvyšující se nátlak na firemní prostředí, které se snaží fungovat v co nejrychlejším tempu. Důsledkem toho jsou rostoucí požadavky na vyšší rychlost komunikace, kde se e-mail stává příliš pomalou a těžkopádnou formou. Problémem se také stává společné datové úložiště ve firmě. Podniky jej nejčastěji řeší formou sdíleného disku, což je nepříliš vhodné řešení. Zaměstnanci často vytvářejí složité hierarchické adresářové struktury, které se později stávají nepřehlednými. V neposlední řádě dochází i ke zneužití diskového prostoru k ukládání osobních dat. Další problém v této oblasti nastává v případě odchodu zaměstnance z firmy. Kvalifikovaní odborníci si s odchodem odnášejí nejen své zkušenosti, ale i data, o která firma přijde.

Jedním z řešení výše popsaných problémů je podniková sociální síť. Podnikové sociální sítě spojují sociální sítě s firemním intranetem a umožňují tak rychlou a snadnou spolupráci mezi zaměstnanci. Jedním z hlavních cílů je zvýšení produktivity zaměstnanců. Té je dosaženo díky rychlé komunikaci, okamžitému sdílení souborů a vytvářením skupin pro jednotlivé projekty či týmy odborníků. Díky sdílení informací na firemní sociální síti zůstává důležité know-how ve firmě i po odchodu zaměstnance.

Tato práce se zabývá procesem vývoje podnikové sociální sítě, která má definované požadavky od zadavatele. Jejím cílem je funkční Java aplikace s použitím definovaných technologií. Název aplikace je Mraveniště.

Diplomová práce je rozdělena do osmi kapitol, přičemž v první kapitole je provedena rešerše dostupných řešení a jejich popis. Ve druhé kapitole je osvětlena metodika UP, která byla vybrána, jakožto nejlepší řešení pro vývoj aplikace. Samotný proces vývoje začíná kapitolou tři, jenž popisuje všechny požadavky na systém. Po prozkoumání požadavků následuje jejich analýza, ze které nakonec vzniká analytický model. Ten je základem pro model návrhový, popsaný v kapitole pět. Kromě návrhového modelu vznikají též datové modely, konkrétně se jedná o konceptuální, logický a fyzický model. Po dokončení všech potřebných modelů ve fázi návrhu, následuje implementační fáze. Implementaci se věnuje kapitola šest, kde je zmíněný Apache Maven a jeho součásti. Velkou část implementace tvoří framework Spring, v kombinaci se Spring Security. Požadavkem zadavatele byla implementace pomocí technologie JDBC, která

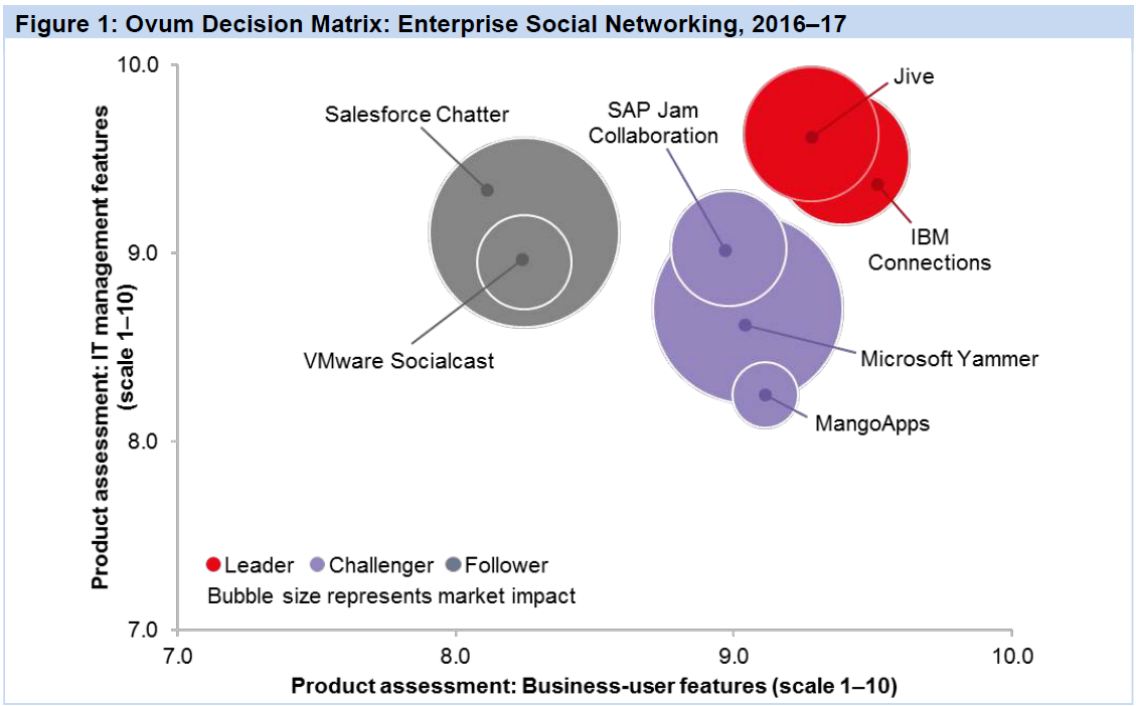
je v této části práce též popsána. Nakonec jsou zde popsány zdrojové kódy aplikace. Předposlední kapitola 7 nabízí pohled na fázi testování. Jsou zde uvedeny metodiky a způsoby testování a samozřejmě průběh testování aplikace. Na závěr je zhodnoceno celé řešení Mraveniště.

1 REŠERŠE

V této kapitole je uvedena rešerše dostupných řešení podnikových sociálních sítí. Nejpoužívanější sítě budou popsány podrobněji v první části rešerše. Ve druhé části budou krátce zhodnoceny i další produkty dostupné na trhu. Třetí část shrnuje důvody vzniku Mraveniště.

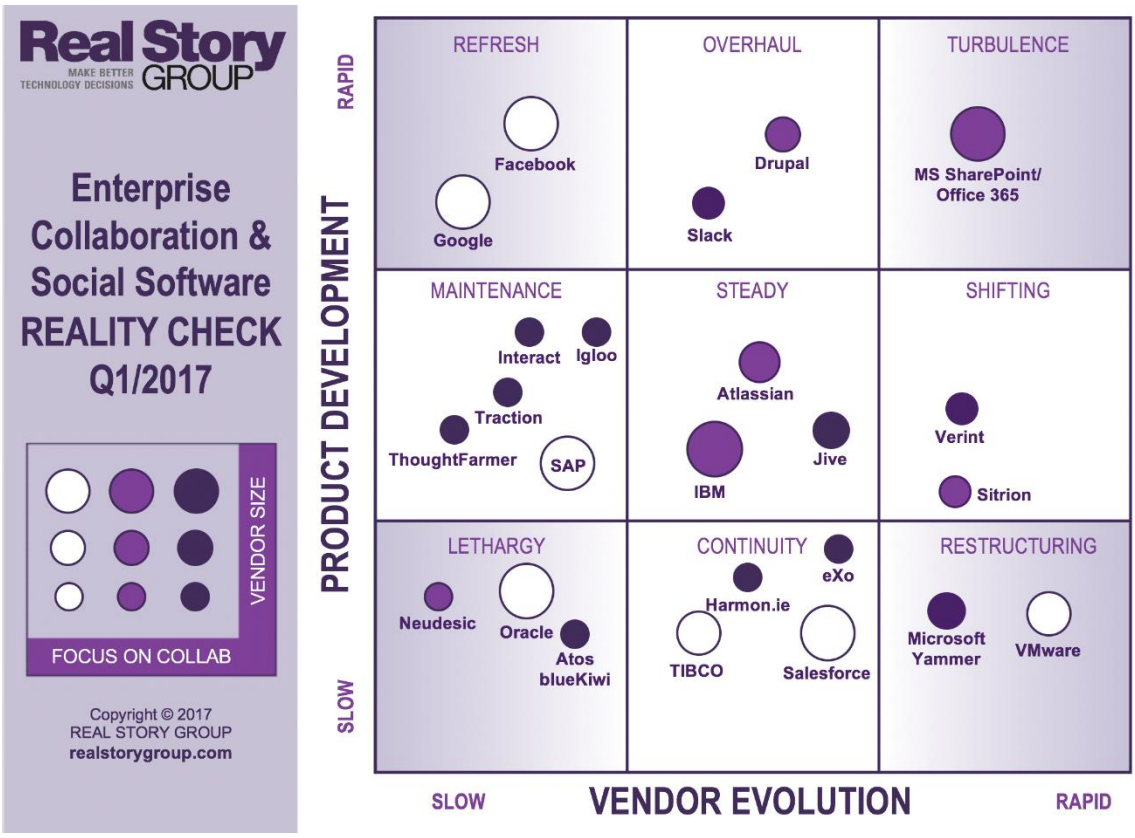
1.1 Nejpoužívanější ESN

Podle analytické společnosti Ovum patří mezi nejvýznamnější a nejpoužívanější produkty Jive, IBM Connections a Microsoft Yammer. Jako jejich nadějného následníka vidí Salesforce Chatter, viz obrázek 1. [4]



Obrázek 1: Graf analýzy podnikových sociálních sítí. Zdroj: [4]

Společnost Real Story Group provedla v roce 2017 analýzu, ve které uvedla, že sítě Jive, IBM Connections a Atlassian patří mezi ty nejstabilnější. Výše zmíněný Microsoft Yammer vidí jako pomalu se vyvíjecí produkt. MS SharePoint/Office 365 je dle RSG příliš rychle vyvíjen a z toho důvodu se stává nestabilním. RSG analyzovala samozřejmě i další sítě, viz obrázek 2. [5]



Obrázek 2: Analýza ESN od společnosti RSG. Zdroj: [5]

1.1.1 Microsoft Yammer

Společnost Yammer, Inc. odkoupila v roce 2012 firma Microsoft Corp. za 1,2 miliardy amerických dolarů. Již v té době byl Yammer používán více než 85 % firem ze seznamu Fortune 500. [2] „Yammer je platforma podnikové sociální sítě, jenž umožňuje rozvíjet komunikaci uvnitř firmy.“ [1] (překlad vlastní, citováno dle [2]) Tento software je velice podobný sociální síti Facebook. Yammer nabízí vytváření skupin (například podle jednotlivých oddělení), rychlý přístup k informacím, synchronizovaná sdílení, práci s firemními dokumenty a s tím související sdílené úložiště dokumentů [2]. Dále nabízí “klasické” funkce sociální sítě: real-time přijímání a odesílání zpráv (tzv. “instant messaging”), sdílení fotografií, notifikace, tagy a “nástěnku” s příspěvky. Na rozdíl od Facebooku, Yammer podporuje firemní fóra, hodnocení nápadů, videokonference aj. [3]. Zajímavostí je jistě sběr, analýza a následné vyhodnocení nasbíraných

[1] „Yammer is a business social networking platform that helps companies manage their internal communications.“ [2]

informací pomocí sady nástrojů generujících hlášení (reports) [2]. Yammer nabízí funkci "Connectors", díky které lze propojit síť Yammer s aplikacemi třetích stran, jako jsou např. Twitter, GitHub, Bing News,Trello, SharePoint a další [1]. K využívání Yammeru stačí pouze platný firemní email [2].

1.1.2 Jive-n

Jive-n je jeden z nejpřednějších nástrojů ESN, jenž nabízí pokročilé funkce a schopnosti vedoucí ke kvalitní spolupráci mezi zaměstnanci, členy týmu, vedením a dalšími stranami ve firemním prostředí. Jive-n je propojen s mobilními aplikacemi, díky kterým může být uživatel stále ve spojení s ostatními a získávat tak nejnovější informace. Je také velice flexibilní, co se týče spolupráce s desktopovými aplikacemi a systémy. Dalším důležitým kritériem je možnost přizpůsobit si Jive-n pro firemní účely a velmi snadná konfigurovatelnost. [6]

Uživatelé (zaměstnanci i osoby mimo podnikové prostředí) mají možnost se zapojovat do diskusí a anket, přispívat do sdílených dokumentů, vytvářet nápady a také o nich hlasovat. Položky (items) lze také označovat jako ukončené, oficiální, užitečné aj. Kromě přidávání příspěvků a videí lze mimo jiné vytvářet vlastní webové stránky, portály oddělení a proudy událostí (activity streams). Tyto proudy si lze upravovat, respektive odebírat pouze ty, které uživatele zajímají a vyfiltrovat nepotřebné. Stejně jako Yammer má i Jive-n vestavěnou funkci analyzování. Tato funkce umožňuje zjistit, které věci fungují či nefungují a které se případně dají vylepšit. Administrátoři mohou navíc zjišťovat trendy, denní aktivity, žebříčky atd. Uživatelé mohou získávat body, úrovně a odznaky za jejich zapojení a mohou udělovat odznaky svým kolegům. Jive-n má integrovanou spolupráci s mnoha aplikacemi, jako jsou například Outlook, GMail, Microsoft Office 365, Google Docs, Google Drive, SharePoint, Dropbox, Microsoft Skype for Business Online, Jabber, Google Hangouts. Jive-n spolupracuje také s jinými systémy: Salesforce, ServiceNow, Zendesk, Jira a další. [7] Dalšími funkcemi jsou inteligentní vyhledávání, IT "help desk", automatické odebírání novinek, služba exportu dat, profily, blogy, skupiny, diskuse a certifikované zabezpečení [6].

Co se týče ceny tohoto produktu, prodejní oddělení firmy vytváří cenové nabídky pro každého zákazníka zvlášť tak, aby vyhovovaly jeho potřebám. Dále společnost Jive nabízí také 30denní zkušební verzi. [7]

1.1.3 IBM Connections

"IBM® Connections ™ je platforma pro spolupráci, která integruje správu e-mailů, činností a úkolů, rychlé zasílání zpráv, sdílení souborů, úpravu sdílených dokumentů a další, do jednotného řešení." [2] (překlad vlastní, citováno dle [8])

IBM Connections nabízí vytváření pracovních prostorů pro spolupráci (collaborative workspace) na společném projektu. Týmy tak mohou spravovat úkoly týkající se jednoho cíle, komunikovat spolu, sdílet soubory a hlásit stav projektu. Každý uživatel vlastní svůj profil, tak jako u jiných ESN, který zahrnuje jeho role, dovednosti atp. Samozřejmostí je správa e-mailu, kalendář a rychlé zprávy. Spolupráci ve firmě podporují také wiki[3], blogy, mikro blogy, fóra a další, jenž se dají synchronizovat mezi osobními počítači, mobilními zařízeními a tablety. IBM Connections Docs je podobný produktu Google Drive; oba skýtají spolupráci na dokumentech rozdílného typu v reálném čase, s možností kontextového komentování. Email od IBM Connections – IBM Verse – vyžívá analýzu, aby vždy zobrazoval kritické zprávy, odstraňoval duplicity a snižoval množství pošty. To však není jediný druh analýzy, který IBM nabízí: existují technologie pro analýzu aktivity uživatele, a to za účelem organizace pracovního prostoru (zobrazuje prioritní informace). Podobně funguje i služba IBM Connections Orient Me, jež navrhuje, které osoby a obsah budou s největší pravděpodobností důležité pro uživatele. IBM Connections dále nabízí sadu IBM Connections Engagement Suite. Je to sada nástrojů, obsahující IBM Connections Engagement Center, IBM Connections Docs a IBM Connections. Ty společně vytváří prostor pro spolupráci uvnitř i vně organizace, dále personalizovaný digitální rozbočovač pracoviště a v neposlední řadě výše zmíněnou spolupráci na podnikových dokumentech. [8]

Zajímavostí je jistě doplněk na dodržování předpisů. IBM jej popisuje následovně: "Chraňte svoji značku a pověst a splňujte přitom firemní, právní nebo správní požadavky. IBM Connecti-

[2] „IBM® Connections™ is a collaboration platform that integrates email, activity and task management, instant messaging, file sharing, collaborative document editing and more into a unified solution." [8]

[3] Co je to wiki

ons Compliance poskytuje řešení pro vyhledávání shody, včetně jednotlivých pravidel pro kontrolu obsahu[4], "logování"[5] a archivaci sociálních a instant messaging zpráv. Prosazujte přijatelná pravidla použití definovaná na globální, skupinové nebo uživatelské úrovni a omezujte právní náklady prostřednictvím rychlého a jednoduchého přezkoumání a vyhledávání a se sledováním v téměř reálném čase a výstrahami."[6] (citováno dle [8])

V neposlední řadě též zmíním IBM PureApplication. Je to předem nakonfigurovaný vzor, který snižuje čas a náklady na instalaci a konfiguraci ESN [8].

Společnost IBM spolupracuje s obchodními partnery po celém světě, jedním z partnerů je i česko-belgická firma Whitesoft, která má pobočky v Praze, Liberci, Ústí nad Labem, Plzni a v belgických Antverpách [9][10]. Whitesoft nabízí svým zákazníkům zabudování rozličných rozšíření s mnoha funkcemi a upravením vzhledu, a to vše na míru [10].

1.1.4 Confluence

Confluence je úspěšný software od společnosti Atlassian vytvořený pro snadnější spolupráci a rychlé sdílení informací v podnikovém prostředí. Jeho základní funkcí je rozdělení pracovního prostoru po jednotlivých týmech, odděleních či projektech, čímž dochází k efektivnímu uspořádání informací, a tedy rychlejší práci na projektech. Confluence oproti jiným ESN nabízí verzování souborů a dokonce stránek, takže se uživatelé mohou vrátit k předchozím verzím a mít tak zálohu. Atlassian silně podporuje integraci s aplikacemi od jejich společnosti, například JIRA software. Podporuje ale i jiné, například Google Drive a Microsoft Office. Pro aplikace třetích stran nabízí řadu doplňků a pluginů; jejich kompletní seznam lze najít na těchto webových stránkách: https://marketplace.atlassian.com/search?product=confluence. Confluence dokáže omezovat přístup k obsahu na 3 úrovních – globální, prostorové a pro individuální stránky. Mimo jiné lze také vytvářet požadavky na produkt, poznámky ke schůzkám a plánovat

[4] Tato funkce umožňuje automatické vyhledávání obsahu s pomocí klíčových slov či frází a okamžitě na případný škodlivý obsah upozorní příslušného správce. [8]

[5] Logováním je zde myšleno zaznamenávání dat do souboru za účelem analýzy.

[6] "Protect your brand and reputation and meet corporate, legal or governance requirements. IBM Connections Compliance provides compliance discovery solutions, including granular policy controls for content inspection, logging and archiving of social and instant messaging communications. Enforce acceptable use polices defined at global, group or user levels and reduce legal costs through fast and simple content review and retrieval and near real-time monitoring and alerts." [8]

projekty. Lze se přihlašovat k odběru přes email, RSS[7], či si nechat odeslat stránku přímo do emailu. Dále zvládá typické vlastnosti ESN: komentáře, "like", označování uživatelů, blogy (pro jednotlivce i pro skupiny), diskuse, práci s firemními dokumenty. Pro usnadnění práce lze využít šablon pro vytváření dokumentů, stránek anebo si přečíst tipy pro vyhledávání (jak v obsahu, tak i v dokumentaci). Všechny desktopové aplikace jsou synchronizovány s Confluence cloud a s mobilními zařízeními iOS/Android. Při zakoupení desktopové verze produktu, zákazník získává 12 měsíční technickou podporu zdarma. [12]

1.2 Ostatní ESN řešení

Ovum řadí k dalším úspěšným podnikovým sítím například Salesforce Chatter a SAP [4]. RSG vidí naopak budoucnost ESN v Microsoftu, který uvedl na trh úspěšný produkt MS Sharepoint [5]. Adrienne Fichter uvádí v příspěvku s názvem "The tool jungle in the corporate world – which platform performs well?" krátký seznam úspěšných řešení. Mezi první řadí nástroj Slack. [14]

1.2.1 Salesforce Chatter

Salesforce Chatter se především zaměřuje na propojení komunit s byznys procesy firmy. Dává také důraz na zapojení zaměstnanců do konverzací, které odměňují body (skóre). Aby se zvýšila míra příspěvků, aktivní přispěvatelé jsou označováni pomocí různých odznaků (badges). Zaměstnanci mohou dávat vedení zpětnou vazbu pomocí témat, anket a dalších nástrojů. Stejně jako další ESN nabízí Salesforce Chatter možnosti konverzací, sdílení souborů či jiných dat, skupiny a práci na mobilním zařízení. [15]

1.2.2 SAP Jam Collaboration

SAP Jam je, na rozdíl od výše zmíněných sítí, provozován jako cloudové řešení. Má široký výběr pro práci se šablonami – pro školení zaměstnanců, koučování, sdílení informací apod. Dále nabízí správu marketingových kampaní, nástroje pro prodejní oddělení a nástroje pro

[7] "Formát pro publikování seznamu odkazů na aktualizované WWW stránky za účelem rychlého informování uživatele o novinkách a změnách." [12]

služby a podporu. Pro správu byznys procesů lze využít nástroje SAP SuccessFactors Employee Central/Foundation, AP SuccessFactors Learning, Content management systems. Obsahují profily uživatelů, nástroje pro školení zaměstnanců a integrované aplikace třetích stran (např. Microsoft SharePoint). [16]

1.2.3 Microsoft Sharepoint

Microsoft na svém webu uvádí, že jejich produkt MS Sharepoint již používá více než 200 000 organizací. Jako jiné podnikové sítě umožňuje i MS Sharepoint sdílení souborů a informací napříč celou firmou a usnadňuje spolupráci mezi všemi zaměstnanci. Dá se používat i na mobilních zařízeních a sdílet data jak uvnitř, tak i zvenku organizace. Díky rychlému vyhledávání lze snadno najít požadovaný obsah a pomocí vestavěných aplikací (SharePoint seznamy a knihovny, Microsoft Flow, PowerApps) je k dispozici možnost budovat byznys procesy. [17]

1.2.4 Slack

Slack umožňuje vytváření kanálů, které slouží k roztřídění konverzací do různých témat, s možností zapojit vybrané uživatele do těchto konverzací. Nabízí též videokonference ve dvou a více uživatelích, a to jak na stolním počítači, tak i na mobilním zařízení. Stejně jako Confluence od společnosti Atlassian, nabízí i Slack verzování a zálohování všech dokumentů, konverzací a sdílených souborů. Jsou tu také dostupné aplikace třetích stran, jako jsou například Google Drive, Salesforce, ZenDesk, Jira Cloud, Zoom a další. Oproti ostatním ESN je zde možnost vybudování vlastního API, které lze vytvořit podle srozumitelného návodu na webových stránkách společnosti. [18]

1.3 Závěr rešerše

Všechna výše zmíněná řešení jsou bezesporu cennými nástroji ve firemním prostředí. Bohužel ani jedno z nich nepokrývá všechny požadavky, které si zadavatel vytyčil. Mraveniště je proto jedinečné díky této kombinaci vlastností. Zároveň je zde otevřen prostor pro změny a rozšíření aplikace, které zadavatel rovněž plánuje. Aplikace je unikátní i z hlediska implementace, konkrétně výběrem technologií a jejich použitím.

2 ZVOLENÁ METODIKA VÝVOJE

Tato kapitola se věnuje zvolené metodice vývoje softwaru. Nejdříve je vysvětlen důvod volby metodiky a následně je popsána samotná metodika.

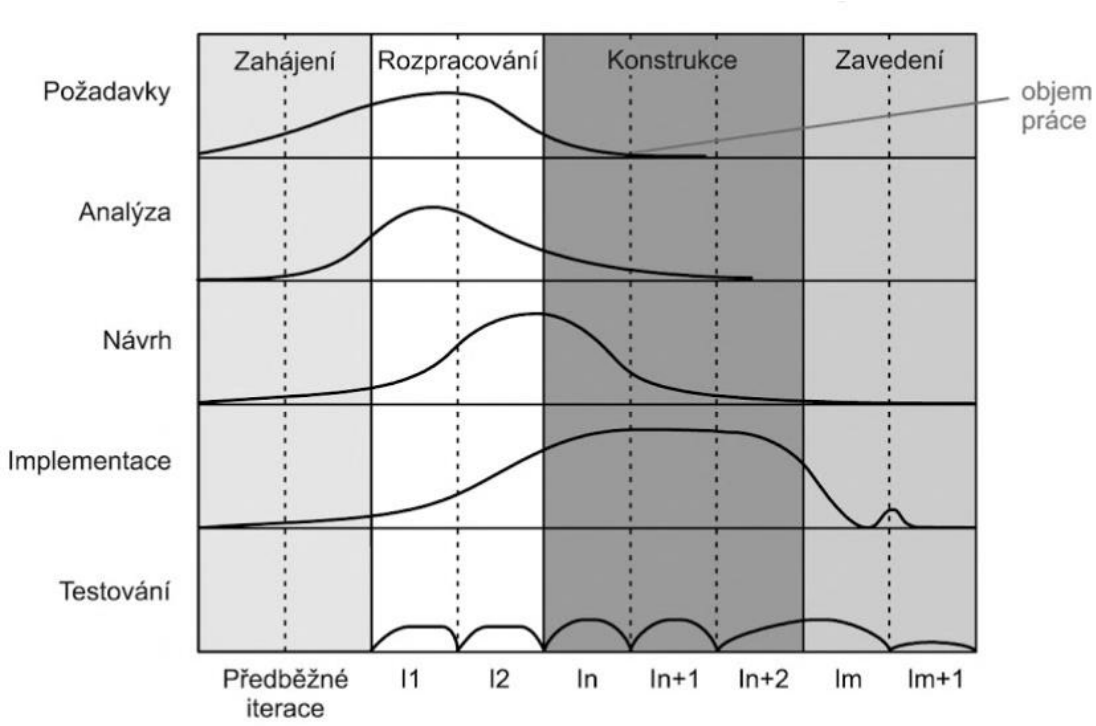
Pro vlastní vývoj systému byla zvolena metodika Unified Process (UP) z důvodu předchozích pozitivních zkušeností s touto metodikou a též s její rozšířeností.

Metodika UP je ověřená metoda vývoje, založená na iteracích pracovních postupů. V každé iteraci existuje těchto pět základních pracovních postupů:

- požadavky,
- analýza,
- návrh,
- implementace,
- testování. [19]

Tyto postupy budou podrobněji vysvětleny v následujících kapitolách.

Životní cyklus projektu je rozdělen na čtyři fáze: zahájení, rozpracování, konstrukce a zavedení, přičemž každá fáze končí milníkem a může obsahovat jednu či více iterací. Vše názorně zobrazuje obrázek číslo 3, který je převzatý z knihy UML 2 a unifikovaný proces vývoje aplikací. [19]



Obrázek 3: Fáze metodiky UP. Zdroj: [19]

Ve fázi zahájení by měly být specifikovány požadavky a provedena jejich analýza. Mohou být prováděny i návrhářské a implementační práce, a to v případě vytváření technického prototypu. Dále může být nadnesen podnikatelský záměr a označena kritická rizika. V této fázi však nedochází k žádnému testování. [19]

Fáze rozpracování zahrnuje především upřesnění požadavků a jejich případů užití, vylepšení odhadu rizik, tvorbu architektonického základu. Co se týče analýzy, je třeba stanovit cíle projektu. Po vytvoření spustitelného architektonického základu následuje jeho testování. [19]

Hlavními cíli ve fázi konstrukce je splnění všech požadavků analýzy i návrhu a vytvoření funkčního systému. Jinak řečeno, v této fázi se dokončuje analytický i návrhový model a je podle nich vytvořen počáteční systém, který je vzápětí testován. [19]

Poslední fáze, zavedení, má na starosti opravu chyb v systému (případně i v návrhovém modelu). Je tedy kladen důraz na pracovní postupy implementace a testování (beta-testy, přejímací testy). Dále se připravuje pracoviště pro zavedení nového systému, tvorba manuálů a dokumentace, konzultace s uživateli, a nakonec poslední revize. [19]

3 SPECIFIKACE POŽADAVKŮ

V této kapitole bude nejdříve vysvětlen termín požadavky a jejich rozdělení na funkční a nefunkční. V podkapitolách funkční a nefunkční požadavky budou v tabulkách uvedeny požadavky na Mraveniště, spolu s jejich popisem. Poté bude následovat kapitola modelování případů užití, v níž bude vysvětlen termín případ užití a uvedena praktická ukázka modelu. Modely případů užití a matice sledování požadavků byly vytvořeny v programu Enterprise Architect.

Specifikace softwarových požadavků se skládá z modelu požadavků a z modelu případů užití. Model požadavků obsahuje funkční a nefunkční požadavky. Požadavky jsou vyjádřením toho, co by měl systém dělat a jaké chování by měl systém poskytovat. Neměly by ale říkat, jak by to měl systém dělat. Jsou prvním krokem při tvorbě nového systému a neměly by být podceněny, protože kvůli tomu může dojít k selhání celého projektu. Požadavky lze získat různými způsoby a v této práci budou uvedeny jen některé z nich. První a základní metodou získávání požadavků je z dodaného dokumentu, ve kterém je zhruba popsáno, co by měl systém vykonávat za funkce, případně jaká jsou jeho omezení. Další technikou může být například konzultace s budoucími uživateli. Těm by měly být kladeny otázky, týkající se používání systému, avšak nesmí být zavádějící. Jejich odpovědi je třeba si zapsat a po skončení konzultace je zanalyzovat. Jako vhodný doplněk pro konzultace je dotazník, který se dá předat většímu množství osob. V něm lze klást otázky, které nás napadly během konzultací. Podle Arlowa a Neustadta patří mezi nejefektivnější způsob tzv. dílna požadavků. Jedná se o volnou diskusi o funkcionalitách systému, jejímiž účastníky by měly být experti v oboru, zainteresované osoby, sparingpartner[8] a inženýr požadavků. Všechny nápady se zapisují a jejich analýza probíhá až po skončení diskuse. [19]

Požadavky mají jednotný formát: <id> <systém> bude <funkce>, kde id je jedinečný identifikátor, systém nese název systému a funkce je požadovaná funkce systému [19].

3.1 Funkční požadavky

Funkční požadavek definuje určitou funkci systému; říká, co by měl systém dělat [19].

[8] Sparingpartner je osoba, která nám bude názorově oponovat. Doslovný překlad podle Ústavu pro jazyk český je „soupeř pro tréninkové utkání“.

Požadavky jsou pro přehlednost rozděleny do jednotlivých tabulek, které jsou rozřazené podle jejich funkce.

V tabulce 1 jsou umístěny požadavky, týkající se profilu uživatele. Uživatelem je zde myšlena osoba v rámci firmy, která je v této síti registrována. Každý uživatel bude mít pouze jeden profil, který bude svázaný s jeho účtem. Uživatel bude mít na svém profilu zobrazeny osobní údaje jako je: jméno, příjmení, bydliště, telefon. Dále budou následovat údaje týkající se firemního prostředí, například číslo oddělení a číslo kanceláře. Uživatel si bude moci přidat na svůj profil jeho profesní znalosti/dovednosti, které budou ostatní spolupracovníci schvalovat.

Tabulka 1: Funkční požadavky na profil uživatele

Profil uživatele	
001	Systém umožní vytvořit profil uživatele.
002	Systém umožní zobrazit v profilu uživatele jeho osobní údaje.
003	Systém umožní zobrazit znalosti uživatele.
004	Systém umožní získat reference na uživatele od jeho spolupracovníků.
005	Systém umožní zobrazit v profilu uživatele údaje týkající se firemního prostředí.

Tabulka číslo 2 obsahuje požadavky na úvodní stránku. Úvodní stránka bude první strana, kterou uživatel po přihlášení uvidí. Budou na ní zobrazeny příspěvky od jiných uživatelů, ale mohou tam být i příspěvky vlastní. Každý příspěvek si ponese textovou či jinou formu informace (například soubory či fotografie) a komentáře, které se tohoto příspěvku týkají. U příspěvku bude možné dávat „To se mi líbí“, jakožto kladné hodnocení daného příspěvku anebo naopak dávat „To se mi nelíbí“. Uživatel bude mít možnost zakládat skupiny. Zakladatel skupiny (i jeho členové) budou moci upravovat údaje o skupině a přidávat do ní příspěvky. Pouze zakladatel bude mít právo ji úplně smazat. Zaměstnanci budou moci využít chat pro rychlou komunikaci. Zároveň budou upozorňováni na nové události.

Tabulka 2: Funkční požadavky na úvodní stránku

Úvodní stránka	
006	Systém umožní zakládat skupiny.
007	Systém umožní spravovat skupiny.
008	Systém umožní mazat skupiny.
009	Systém umožní použít chat.
010	Systém umožní upozornění na nové události.

011	Systém umožní komentování příspěvků.
012	Systém umožní interakci mezi uživateli pomoci “Like”.
013	Systém umožní interakci mezi uživateli pomoci “Dislike”.

Tabulka 3 shrnuje požadavky, týkající se přidávání nápadů, stránek a položek. Uživatel bude moci přidávat nové nápady a spravovat stávající. Oprávnění uživatelé budou moci měnit stav nápadu (například zda nápad čeká na realizaci, či již byl realizován apod.). Každý uživatel bude moci také vytvářet jednoduché webové stránky.

Tabulka 3: Funkční požadavky na přidávání obsahu

Přidávání obsahu	
014	Systém umožní vytvářet jednoduché webové stránky.
015	Systém umožní přidávání souborů na webové stránky.
016	Systém umožní přidávat nové nápady.
017	Systém umožní zobrazení seznamu nápadů.
018	Systém umožní změnu stavu nápadu.

Tabulka 4 obsahuje požadavky na kontakty. Každý uživatel bude moct kontaktovat jiného uživatele systému, a to prostřednictvím chatu. Dále bude systém umožňovat filtrování kontaktů podle jednotlivých oddělení, jména a příjmení, pozice.

Tabulka 4: Funkční požadavky na kontakty

Kontakty	
019	Systém umožní kontaktování spolupracovníků.
020	Systém umožní filtrovat kontakty podle jednotlivých oddělení.
021	Systém umožní filtrovat kontakty podle osobních údajů (jméno, příjmení, pozice).

Tabulka 5 se zabývá požadavky na „Human Resources“ složku. Tyto požadavky se týkají uživatelů sítě, kteří budou mít roli HR. Těmto uživatelům bude umožněno spravovat firemní dokumenty jako jsou: formuláře, předpisy, pracovní smlouvy, výplatnice, kariérní a vzdělávací nabídky, informace o čerpání dovolené. U některých z nich bude možnost popisu pomocí WISIWYG editoru, v překladu doslova „Co vidíš, to dostaneš“. Tento editor umožňuje napsat webovou stránku bez znalosti HTML jazyka [20].

Tabulka 5: Funkční požadavky na HR složku

HR složka	
022	Systém umožní stažení formulářů a popis pomocí WYSIWYG editoru.
023	Systém umožní nahrání formulářů.
024	Systém umožní stažení předpisů a jejich popis pomocí WYSIWYG editoru.
025	Systém umožní nahrání předpisů.
026	Systém umožní stažení pracovní smlouvy a její popis pomocí WYSIWYG editoru.
027	Systém umožní nahrání pracovní smlouvy.
028	Systém umožní stažení výplatnice a její popis pomocí WYSIWYG editoru.
029	Systém umožní nahrání výplatnice.
030	Systém umožní zobrazit seznam nabídek pracovních příležitostí.
031	Systém umožní vytvořit seznam nabídek pracovních příležitostí.
032	Systém umožní zobrazit seznam nabídek vzdělávání.
033	Systém umožní vytvořit seznam nabídek vzdělávání.
034	Systém umožní zobrazit informace o čerpání dovolené.

Poslední tabulka s funkčními požadavky, tabulka 6, zahrnuje ostatní funkce ESN. Konkrétně se jedná o vyhledávání pomocí zadaného výrazu a vyhledávání podle tagů. Při vyhledávání podle výrazu vyhledá Mraveniště slovo v textu uvnitř stránky či příspěvku (záleží na oblasti vyhledávání). Při vyhledávání podle tagu Mraveniště nalezne požadované příspěvky, jež nesou daný tag. Tagy budou ve tvaru „#názevtagu #druhýTag“ atd.

A jak již jistě vyplynulo z předchozího textu, každý uživatel bude mít svou roli. Uživatelé budou mít více rolí, s možností mezi nimi přepínat. Každý uživatel se bude přihlašovat přes externí zdroj (Google Sign-In).

Tabulka 6: Funkční požadavky na ostatní funkce

Ostatní	
035	Systém umožní vyhledat a zobrazit zadané výrazy.
036	Systém umožní individuální používání systému na základě přidělené role.
037	Systém umožní přihlášení pomocí externích zdrojů.
038	Systém umožní uživatelům s rolí HR, Manažer či Admin přepínání mezi rolemi.

3.2 Nefunkční požadavky

Nefunkční požadavek omezuje funkce a chování systému [19].

V tabulce číslo 7 jsou sepsány nefunkční požadavky na podnikovou sociální síť. Ty zahrnují například: zvolený programovací jazyk (Java), databázi (MySQL) a omezují systém na webovou aplikaci. Webová aplikace je obvykle složitější aplikace, která běží na straně serveru a k jejímu používání je třeba zařízení s připojením na internet a internetový prohlížeč [21]. Systém bude používat tzv. Apache Maven, což je nástroj určený k sestavení projektu. Bližší informace lze nalézt v kapitole 6. Poslední požadavek říká, že data v databázi se nebudou mazat. Při požadavku uživatele na smazání dat, bude u těchto dat nastaven příznak smazání, ale data ve skutečnosti nebudou smazána. Uživatelům se již tato data nebudou dále zobrazovat.

Tabulka 7: Nefunkční požadavky

Nefunkční požadavky	
001	Systém bude používat databázi MySQL.
002	Systém bude používat Spring REST service.
003	Systém bude webová aplikace.
004	Systém nebude vícejazyčný (uživatelská část aplikace bude v českém jazyce).
005	Systém bude naprogramován v jazyce Java.
006	Systém bude využívat k sestavení projektu nástroj Apache Maven.
007	Systém nebude mazat data v databázi, jen je označí jako smazané.

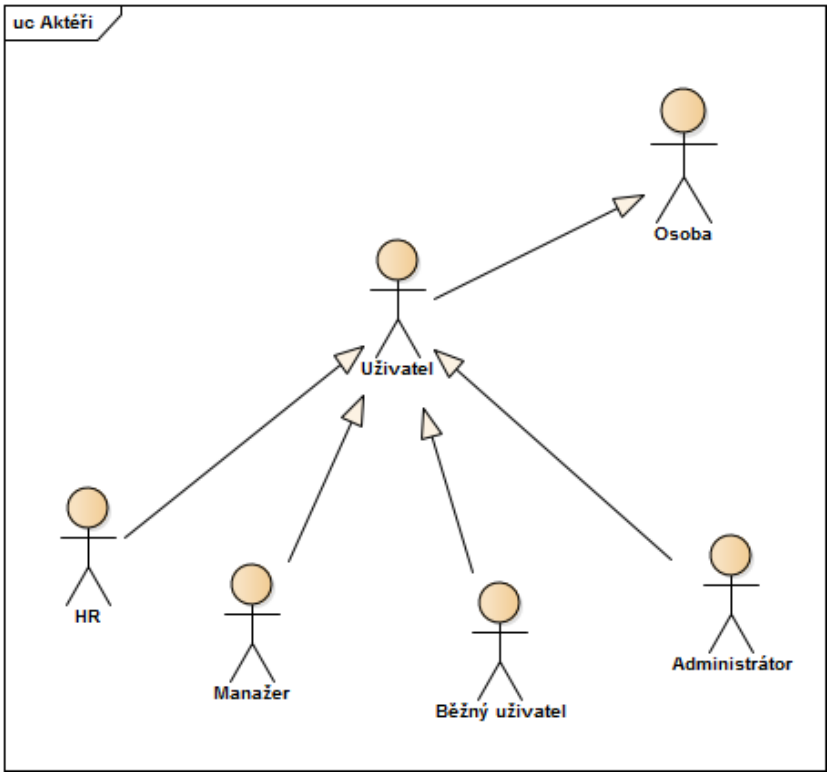
3.3 Modelování případů užití

Modelování případů užití je dalším prostředkem pro dokumentování požadavků. Skládá se z aktivit: nalezení hranic systému, vyhledání všech aktérů a nalezení případů užití. Hranice systému neboli subjekt je grafické znázornění hranic systému okolo případů užití. Jejím smyslem je vytyčení toho, co je a co není součástí systému. Tuto hranici systému definují aktéři, jež systém používají, a případy použití. Subjekt je znázorněn jako rámeček s názvem systému, aktéři jsou nakresleni mimo něj a případy užití jsou uvnitř subjektu. Aktér může být uživatel systému, jiný systém, či předmět s konkrétní rolí a přidělenými případy užití. Aktéři jsou v modelu případu užití zobrazeni jako figurky, které mají v popisku název role, jež zastávají. Arlow a Neustadt citují ve své knize *UML 2 a unifikovaný proces vývoje aplikací* definici případu užití, z knihy od J. Rumbaugha. Případ užití je „specifikace posloupnosti činností, včetně proměnných posloupností a chybových posloupností, které systém, podsystém nebo třída může vykonat pro-

střednictvím interakce s vnějšími (externími) aktéry“ [19]. Případy užití jsou v modelu zobrazeny jako elipsy s vepsanou činností aktéra uvnitř. Specifikace případu užití se obvykle skládá z následujících částí:

- název případu užití,
- jedinečný identifikátor,
- popis,
- aktéři,
- vstupní podmínky,
- hlavní scénář,
- výstupní podmínky
- a alternativní scénáře. [19]

Mezi aktéry Mraveniště patří: osoba, uživatel, administrátor, HR, manažer a běžný uživatel. Tyto aktéry a jejich vztahy lze vidět na obrázku níže (obr. 4). Mezi aktéry mohou vznikat vztahy, v našem případě se jedná o generalizaci (zobecnění).



Obrázek 4: Aktéři

Osoba je člověk, který ještě nebyl zaregistrovaný (či přihlášený) do systému. Uživatel je již registrovaná osoba, přihlášená do systému, s přidělenou rolí. HR je role uživatele, pod kterou lze nahrávat a editovat formuláře, předpisy, pracovní smlouvy, výplatní pásky, nabídky pro kariéru i vzdělávání. Tito uživatelé též mají právo registrovat do systému nové uživatele a přidělovat jim role. Manažer je další z rolí, díky níž může uživatel editovat úvodní stránku společnosti, „nápadník“ a „Wikislovník“. Také může měnit stavy dovolených zaměstnanců. Běžný uživatel má práva na stahování všech typů souborů od HR oddělení (pokud se nejedná o výplatnici či pracovní smlouvu – ty jsou, co se týče stahování, omezeny). Také může vytvářet a upravovat svůj profil a úvodní stranu, přidávat nové nápady. Na většinu obsahu Mraveniště má právo pro zobrazení. Kompletní přehled o právech uživatelů nabízí tabulka 8.

Tabulka 8: Práva uživatelů

Funkce	**Běžný uživatel**		**Manažer**		**HR**	
	Zobrazení	**Úprava**	**Zobrazení**	**Úprava**	**Zobrazení**	**Úprava**
Profil	Ano	Ano	Ano	Ne	Ano	Ne
Úvodní stránka	Ano	Ano	Ano	Ano	Ano	Ano
O společnosti	Ano	Ne	Ano	Ano	Ano	Ne

Nápadník	Ano	Ano	Ano	Ano	Ano	Ne
Wikislovník	Ano	Ne	Ano	Ano	Ano	Ano
Formuláře	Ano	Ne	Ano	Ne	Ano	Ano
Předpisy	Ano	Ne	Ano	Ne	Ano	Ano
Pracovní smlouvy	Ano	Ne	Ano	Ne	Ano	Ano
Výplatnice	Ano	Ne	Ne	Ne	Ano	Ano
Interní nábor	Ano	Ne	Ano	Ne	Ano	Ano
Vzdělávání	Ano	Ne	Ano	Ne	Ano	Ano
Dovolená	Ano	Ne	Ano	Ano	Ano	Ano
Skupiny	Ano	Ano	Ano	Ano	Ano	Ano
Zprávy	Ano	Ne	Ano	Ne	Ano	Ne
Příspěvky	Ano	Ano	Ano	Ano	Ano	Ano

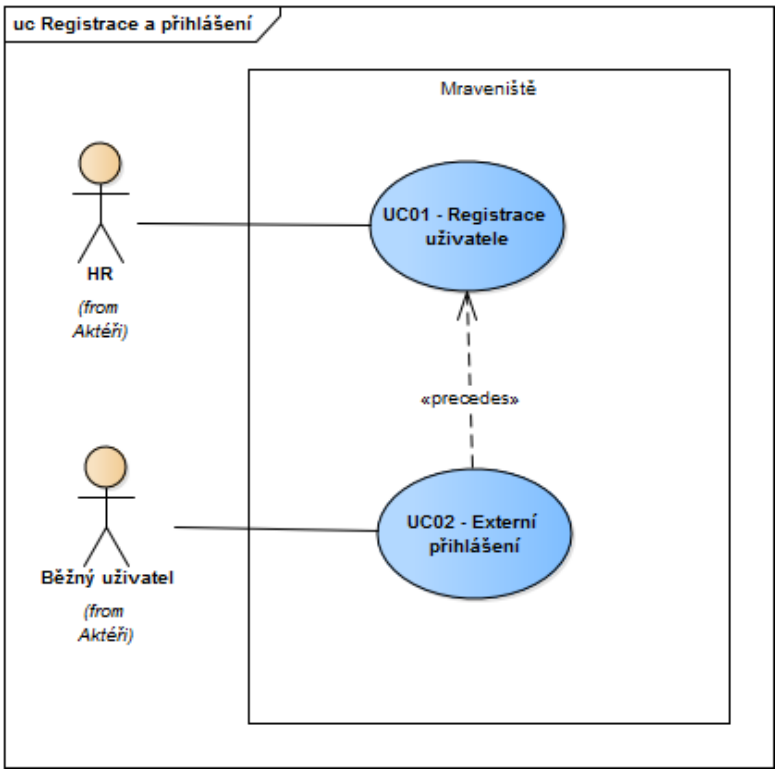
Mraveniště disponuje i rolí administrátora. Administrátor má na starosti správu rolí uživatelů, výčtových typů (typy souborů, stránek, nabídek), výčtových stavů (stavy dovolených a nápadů) a správu změn (upozornění pro uživatele a logování aktivit). Kromě toho má možnost zobrazit kompletní informace o skupinách a zprávách.

Tabulka 9: Práva administrátora

Funkce	**Administrátor**	
	Zobrazení	**Úprava**
Správa typů souborů	Ano	Ano
Správa typů stránek	Ano	Ano
Správa typů nabídek	Ano	Ano
Správa stavů nápadů	Ano	Ano
Správa stavů dovolených	Ano	Ano
Správa rolí	Ano	Ano
Zobrazení všech skupin	Ano	Ne
Zobrazení všech zpráv	Ano	Ne
Zobrazení všech změn	Ano	Ano

Ukázka modelu případu užití je zobrazena na obrázku 4. Jsou zde uvedeny dva případy užití, a to UC01 – Registrace uživatele a UC02 – Externí přihlášení. Aktéry jsou dva uživatelé, jeden

s rolí HR a druhý v roli běžného uživatele. Relace <<precedes>> znamená, že (úspěšnému) externímu přihlášení uživatele předchází jeho registrace.



Obrázek 5: Model případů užití – registrace a přihlášení

Uživatel v roli HR používá případ užití „Registrace uživatele“, jehož hlavní scénář se odehrává dle tabulky 10. V této tabulce je také uvedena vstupní podmínka – platná G-mailová adresa. Pokud by HR uživatel zadal neplatnou adresu, běžnému uživateli by se nezdařilo přihlášení do systému. Výstupní podmínkou je úspěšné vytvoření nového uživatele Mraveniště.

Tabulka 10: Případ užití: Registrace uživatele

Případ užití	Registrace uživatele
ID	1
Stručný popis	Systém zaregistruje nového uživatele.
Hlavní aktéři	Uživatel s rolí HR
Vedlejší aktéři	Žádní
Vstupní podmínky	Platná e-mailová adresa u společnosti Google.
Hlavní scénář	1. Uživatel vyšle požadavek na zobrazení formuláře pro registraci. 2. Systém zobrazí formulář pro registraci. 3. Uživatel vyplní údaje a odešle formulář.

	4. Systém zpracuje formulář. 5. Systém registruje nového uživatele.
Výstupní podmínky	1. Systém vytvořil účet pro nového uživatele.
Alternativní scénáře	Neplatné údaje

Pro přehlednost je alternativní scénář dokumentován samostatně, v tabulce číslo 11. Alternativní scénář obsahuje, stejně jako hlavní scénář, název, ID, popis, aktéry, vstupní/výstupní podmínky. Již neobsahuje hlavní scénář. Název se odvíjí od případu užití a jeho identifikátor je pouze navýšen o jedna.

Tabulka 11: Alternativní scénář: Neplatné údaje

Alternativní scénář	Registrace uživatele: Neplatné údaje
ID	2
Stručný popis	Systém informuje uživatele o zadání neplatných údajů.
Hlavní aktéři	Uživatel s rolí HR
Vedlejší aktéři	Žádní
Vstupní podmínky	Uživatel zadal špatné údaje.
Alternativní scénář	1. Systém informuje uživatele o zadání neplatných údajů.
Výstupní podmínky	Žádné

Při vytváření modelů případů užití byly nalezeny další požadavky na systém a doplněny do patřičných tabulek. Také byla upřesněna jednotlivá práva uživatelů.

3.4 Sledování požadavků

Ke sledování požadavků je potřeba mít specifikované požadavky a případy užití. Poté lze pomocí vybraného nástroje (CASE, DOORS, Rational Requisite Pro aj.) ověřit, že každý požadavek má svůj případ užití a naopak. S velikou pravděpodobností nebudou pouze vznikat vazby 1:1, ale vazby několikanásobné, tedy N: M. Jeden požadavek může být pokrytý více případy užití a jeden případ užití může pokrývat více požadavků, což je v pořádku. [19]

V programu Enterprise Architect byla vytvořena matice sledovatelnosti požadavků, díky níž bylo nalezeno několik dalších nepokrytých požadavků, které byly opět doplněny do tabulek. Tuto matici lze vygenerovat dvěma kliknutími v hlavním menu: *Tools → Relationship matrix*. Následně si lze zvolit konkrétní balíčky, které chcete porovnávat. Na obrázku číslo 6 je vidět

ukázka matice sledovatelnosti požadavků na profil uživatele. Tam, kde se požadavek a případ užití překrývají, je označené políčko.

Source \ Target	Profil uživatele::UC04 - Vytvoření profilu	Profil uživatele::UC05 - Editace profilu	Profil uživatele::UC06 - Smazání profilu	Profil uživatele::UC07 - Hodnocení znalostí	Profil uživatele::UC08 - Zobrazení profilu
Profil uživatele::001_Systém umožní vytvořit profil uživatele.	↑				
Profil uživatele::002_Systém umožní zobrazit v profilu uživatele jeho osobní údaje.					↑
Profil uživatele::003_Systém umožní zobrazit znalosti uživatele.				↑	
Profil uživatele::004_Systém umožní získat reference od svých spolupracovníků.				↑	
Profil uživatele::005_Systém umožní zobrazit v profilu uživatele údaje týkající se firemního prostředí.				↑	

Obrázek 6: Matice sledovatelnosti požadavků na profil

4 ANALÝZA

V této části práce bude vysvětlen pojem analýza a analytický model. V první podkapitole je popsána analytická třída spolu s metodami pro jejich vyhledávání. Následuje kapitola o vybrané metodě hledání analytických tříd, a to pomocí vyhledávání podstatných jmen a sloves. Na tuto kapitolu navazuje téma zabývající se samotným návrhem analytických tříd v rámci aplikace. Druhá kapitola se věnuje realizaci případů užití a jejich definici. V navazujících podkapitolách se lze setkat se sekvenčními diagramy a diagramy aktivit. U každé je uveden diagram, související s Mraveništěm.

Analýza je fáze metodiky UP, která probíhá ke konci fáze Zahájení a zaměřuje se na požadavky a fázi Rozpracování. Jejím hlavním cílem je vytvoření analytického modelu, jenž se zaměřuje na to, co systém dělá, ale nezáleží na tom, jak to dělá (tuto otázku řeší Návrh). Analytický model obsahuje artefakty, což jsou v tomto případě analytické třídy a realizace případů užití. Analýza UP zahrnuje následující aktivity:

- architektonickou analýzu,
- analýzu případu užití,
- analýzu třídy,
- a analýzu balíčku. [19]

4.1 Analytické třídy

„Analytické třídy:

- reprezentují křehkou abstrakci problémové domény,
- měly by mapovat pojmy skutečného světa (…).“ [19]

Analytická třída by měla obsahovat alespoň tyto části: název, atributy, operace, typ viditelnosti, stereotypy, označené hodnoty. Správně navržená třída by zároveň měla splňovat několik kritérií:

1. Je to abstrakce prvku z problémové domény.
2. Název třídy by měl odrážet účel dané třídy.
3. Odráží vlastnost problémové domény.
4. Je soudržná.

5. Obsahuje určitou množinu odpovědností.
6. Má minimum vazeb na ostatní třídy. [19]

Pro vyhledání analytických tříd neexistuje žádný standard, jen osvědčené postupy. Arlow a Neustadt [19] doporučují jako první metodu k vyhledání tříd tzv. analýzu podstatných jmen a sloves. Tato metoda bude podrobněji popsána níže, v kapitole 4.1.1. Analýza podstatných jmen a sloves.

Jako druhou metodu zmiňují analýzu pomocí štítků CRC. Jedná se o skupinovou metodu, kde je nejprve každý nápad (respektive předmět dané domény) zapsán na samolepící lísteček a přilepen na tabuli. Dále jsou těmto předmětům přiděleny a zapsány odpovědnosti. Potom jsou vyhledány spolupracující předměty. Nakonec analytici zhodnotí, které předměty budou třídy a které předměty budou jejich atributy. [19]

Poslední metodou je hledání analytických tříd pomocí stereotypů metodiky RUP. Tato metoda je vhodné doplnění předchozích metod. Spočívá v roztřídění tříd do tří typů (stereotypů) – boundary, control a entity. Jejich podrobnější popis viz obrázek 7. [19]

Stereotyp	Symbol	Sémantika
«boundary»		Třída, jež je prostředníkem mezi systémem a jeho prostředím.
«control»		Třída, která zapouzdřuje chování příznačné pro daný případ užití.
«entity»		Třída, jež se používá k modelování persistentních informací o něčem.

Obrázek 7: Stereotypy podle RUP. Zdroj: [19]

Podrobnější vysvětlení pojmů třída, objekt, spojení a dalších základních znalostí není cílem této práce. Jejich vysvětlení lze nalézt v literatuře viz [19].

4.1.1 Analýza podstatných jmen a sloves

Tato metoda je velice jednoduchá – je třeba vyhledat v textu podstatná jména a jejich spojení, která reprezentují budoucí třídy a jejich atributy. Dále se vyhledají slovesa a slovesné fráze, které značí budoucí operace a odpovědnosti tříd. Tato podstatná jména a slovesa jsou zanesena do tabulky (nebo nástroje CASE) a s pomocí porovnávání případů užití a požadavků jsou vytvořeny třídy, kterým jsou přiřazeny atributy a operace. [19]

Tato metoda byla aplikována na text poskytnutý zadavatelem práce, v němž jsou ve zkratce popsány požadavky na systém. V tabulce číslo 10 jsou uvedena nalezená podstatná jména a jejich spojení, přičemž každé podstatné jméno je určeno jakožto třída, anebo atribut třídy.

Tabulka 12: Analýza podstatných jmen

Podstatná jména a slovní spojení	**Analytická třída**	**Atribut analytické třídy**
Profil	Profile	-
Jméno uživatele	Profile	firstName
Příjmení uživatele	Profile	lastName
Telefon	Profile	mobileNumber
Číslo kanceláře	Profile	officeNumber
Fotografie	Profile	photo
Znalosti	Knowledge	-
Reference	Profile	List<Knowledge>
Zaměstnanec	User	-
Nadřízený	User	leader
Tag	Knowledge	-
Úvodní stránka	Page	-
Příspěvek	Post	-
Skupiny	Group	-
Chat	Message	-
Upozornění	Change	-
Stránky	Page	-
Nápadník	Idea	-
Kontakty	-	-
Oddělení	Department	-
Filtrování	-	-
Sloupce	-	-
Wikislovník	PageType	-
Hledání	-	-
HR	Role	-
Formuláře	FileType	-
Seznam souborů	File	-
Popis	File	fileDescription
Výplatnice	FileType	-
Předpisy	FileType	-
Pracovní smlouva	FileType	-
Interní nábor	OfferType	-

Seznam nabídek	Offer	-
Vzdělávání	OfferType	-
Dovolená	Holiday	-
Schvalování	HolidayState	-
Interakce	Change	-
Like	Like	likeOrDislike
Dislike	Like	likeOrDislike

Ve tabulce 11 jsou uvedená slovesa a z nich plynoucí operace. Pomocí analýzy podstatných jmen a sloves byla nalezena většina základních tříd.

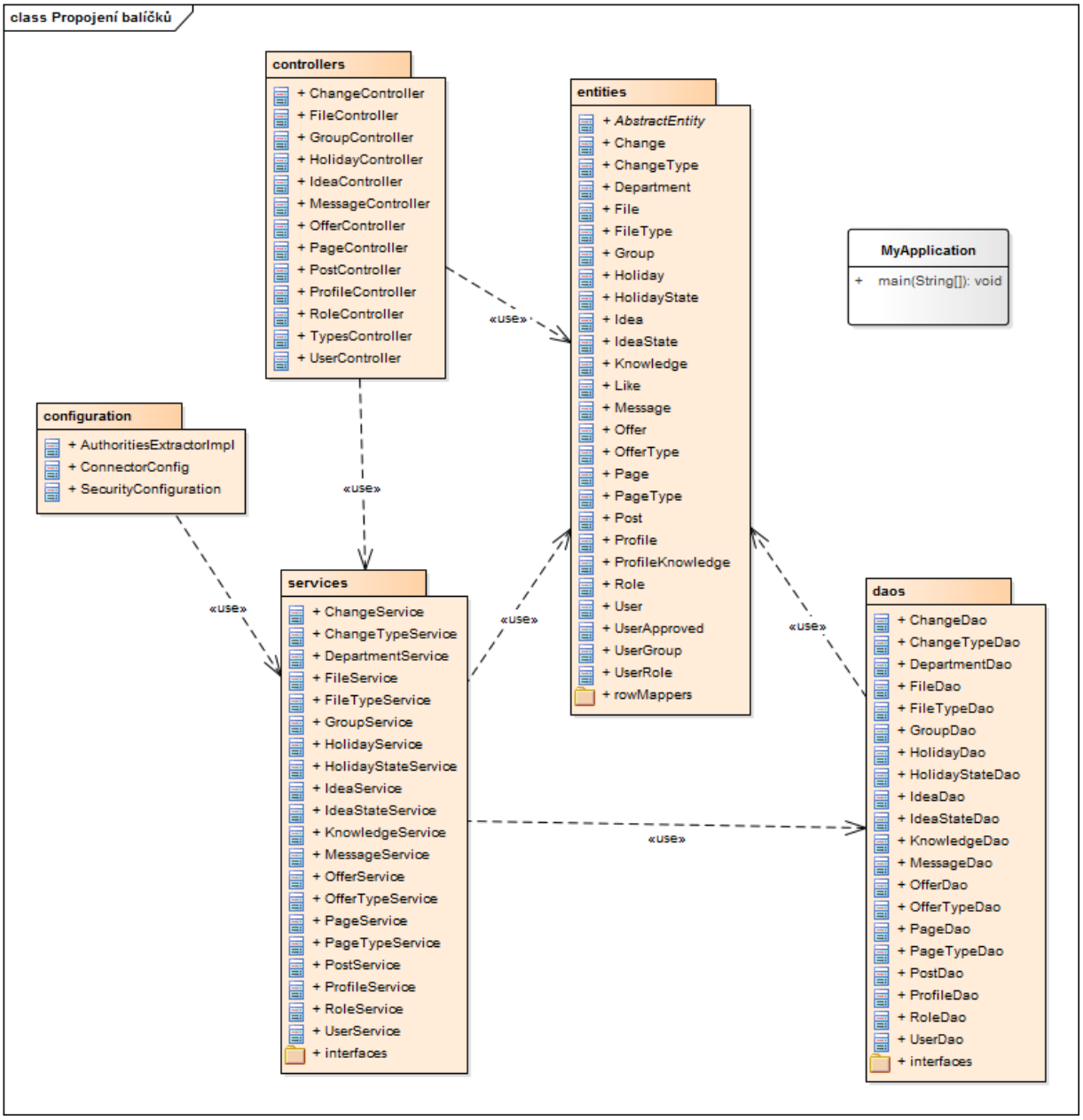
Tabulka 13: Analýza sloves

Slovesa a slovesná spojení	**Operace**
Vytvářet uživatele	createUser
Editovat uživatele	editUser
Mazat uživatele	deleteUser
Vytvořit profil	createProfile
Editovat profil	editProfile
Smazat profil	deleteProfile
Schvalovat znalosti	approveKnowledge
Zakládat skupiny	createGroup
Přidávat stránky	addPage
Přidávat příspěvky	addPost
Okomentovat příspěvek	addPost
Přidat nápad	createIdea
Zobrazit nápady	getAllIdeas
Filtrovat podle sloupců	searchBy
Vyhledávat	search
Zobrazovat kontakty	getUserById

4.1.2 Návrh analytických tříd

Za pomoci výše uvedené analýzy podstatných jmen a sloves byl vytvořen UML návrh analytických tříd. Některé třídy a jejich atributy byly logicky odvozeny, i když nebyly nalezeny v analýze podstatných jmen a sloves. Tento návrh byl modelován v programu Enterprise Architect.

Na obrázku číslo 8 jsou zobrazeny analytické balíčky a relace mezi nimi. Aplikace bude obsahovat 5 balíčků, z nichž některé obsahují další balíčky. Jeden ze základních balíčků (entities) obsahuje entitní třídy reprezentující datové objekty v databázi. Tyto objekty uchovávají samotná data, jako například informace o uživatelích, rolích, skupinách aj. Navíc je zde zařazena i abstraktní třída AbstractEntity, od které všechny entitní třídy dědí metody. Balíček entities obsahuje také balíček rowMappers, obsahující tzv. mapovače (z anglického mapper). Tyto mapovače řádků mapují databázové objekty na objektově orientované objekty. Třídy v tomto balíčku jsou využívány ve všech ostatních částech aplikace, jak napovídají relace typu <<use>>. Balíček daos seskupuje tzv. DAO třídy (Data Access Object), sloužící ke snadnému přístupu do databáze. Každá třída v balíčku implementuje rozhraní umístěné v podbalíčku */daos/interfaces*. Balíček services obsahuje třídy, které volají metody nad instancemi dao tříd. Stejně jako dao třídy implementují rozhraní, která jsou v balíčku */services/interfaces*. Balíček controllers obsahuje tzv. ovladače (controller). Jsou to Java třídy, které zajišťují zachycení požadavků od uživatele a jejich zpracování. Uvnitř jejich metod jsou využívány instance tříd z balíčku services. Poslední balíček, configuration, obsahuje třídy nutné pro nastavení aplikace. Bližší informace jsou k dispozici v kapitole Implementace.

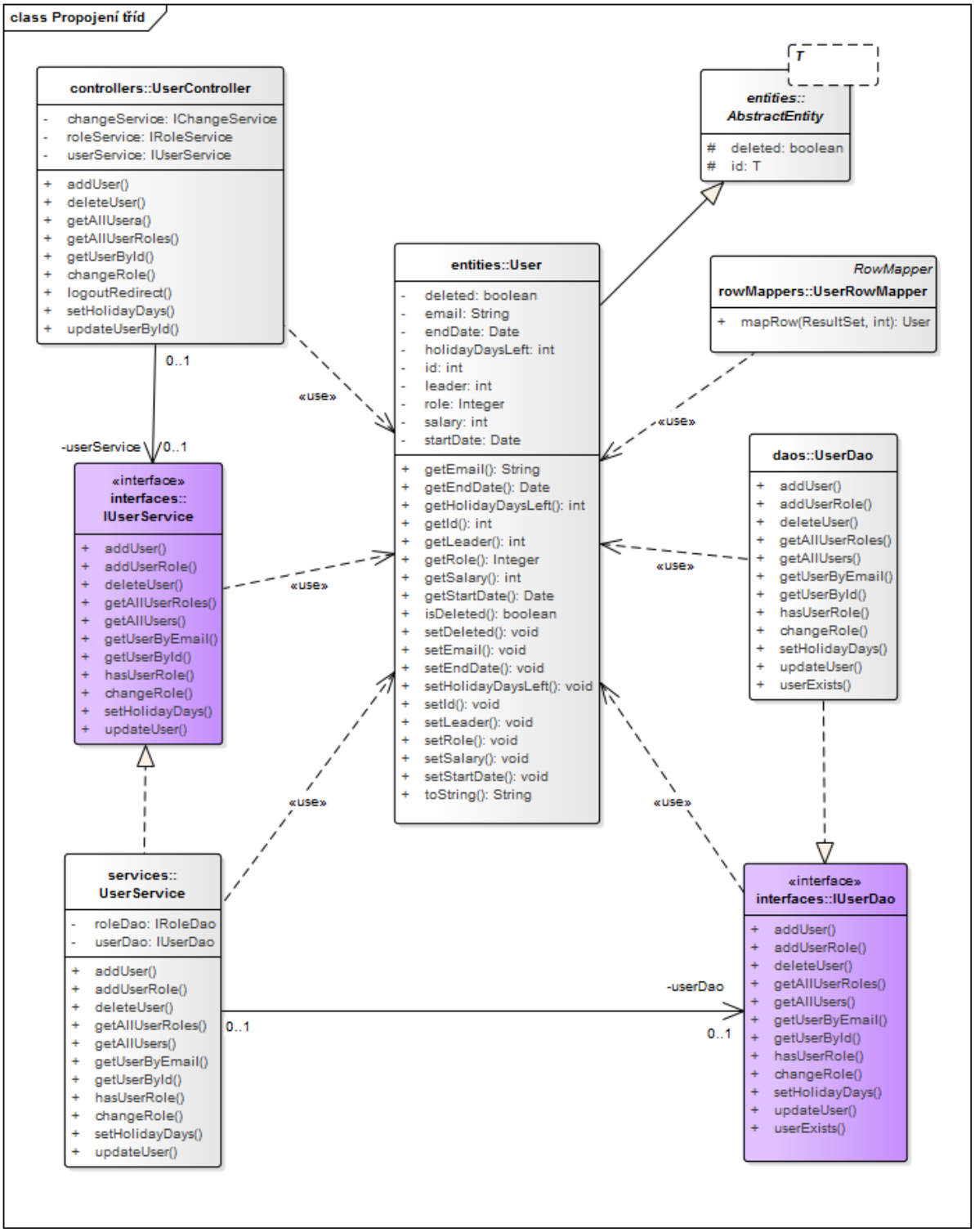


Obrázek 8: Diagram analytických balíčků

Pro lepší pochopení vztahu mezi balíčky i třídami byl vytvořen diagram tříd upřesňující tyto relace (viz obrázek 9). Tento diagram též částečně reprezentuje architekturu frameworku Spring (ukazuje pouze architekturu na straně „back endu“).

Diagram je založený na tzv. MVC modelu, což je zkratka pro angl. Model View Controller. Jedná se o separaci vrstev aplikace, které spolu komunikují a každá má určitá práva. Model je vrstva, která uchovává data a pracuje s nimi. View je vše, co uživatel v rámci aplikace vidí. Controller vše řídí a ovládá.

V tomto diagramu je modelem třída User obsahující informace o uživateli, které nebudou na jeho veřejném profilu. Zahrnuje atributy a jejich get/set metody. Třída Uživatel dědí od třídy AbstractEntity metody getId(), setId(), isDeleted() a setDeleted(). Tyto metody slouží k vrácení či nastavení atributu id a deleted (příznak smazání). Tato třída je používána ve všech třídáchtýkajících se uživatele (tedy všechny třidy na obrázku 9). Třída UserRowMapper má jedinou metodu mapRow(), zděděnou od svého předka RowMapper. V metodě jsou namapovány sloupečky z tabulky User s atributy třídy User. Důležitou třídou je UserDao, která pracuje s databází a využívá přitom UserRowMapper. UserDao implementuje rozhraní IUserDao, které používá tvz. CRUD metody. Název je zkratkou pro často využívané metody create, read, update a delete. Třída UserService obsahuje instanci rozhraní IUserDao a samo implementuje rozhraní IUserService, s téměř stejnými metodami. Poslední důležitou třídou je UserController, obsahující instanci IUserService. Voláním metod tohoto rozhraní zajišťuje obsloužení událostí vzniklých z akcí uživatele. Události jsou v tomto případě požadavky, odpovědi či jiné HTTP metody, které uživatel vyvolal přechodem na jinou URL adresu. Tento princip přístupu či architektura se nazývá REST služby.



Obrázek 9: Diagram analytických tříd

4.2 Realizace případů užití

Realizace případu užití ukazuje vzájemnou spolupráci mezi analytickými třídami, případy užití a aktéry, za účelem dosažení chování popsaném v daném případu užití. Případ užití je tímto

způsobem převeden na diagram tříd a interakcí. Diagramy interakce se dělí na dva typy: diagramy spolupráce a sekvenční diagramy. V těchto diagramech narazíme na pojmy interakce, čáry života a zprávy. Interakce jsou jednotky chování klasifikátoru, který poskytuje kontext a funkce pro jednotlivé interakce. Čára života reprezentuje účastníka (aktéra) interakce. Její syntaxe se může skládat z názvu, typu a selektoru (logické podmínky). Podle typu interakce a čar života rozlišujeme dvě formy: obecnou a konkrétní. Obecná forma je více abstraktní a zastupuje libovolné instance. Je také více používaná. Konkrétní forma znázorňuje interakce mezi jednotlivými instancemi. Zpráva je specifický typ komunikace mezi čarami života během interakce. Tato komunikace může být různého typu: volání operace (zprávy), tvorby nebo uvolnění instance (zprávy) a odeslání signálu. V případě odeslání zprávy čárou života se hovoří o aktivaci a tato aktivace může probíhat mezi jednotlivými čárami života – tento pohyb se označuje jako scénář. [19]

4.2.1 Sekvenční diagramy

„Sekvenční diagramy znázorňují interakce mezi čárami života jako časově uspořádanou posloupnost událostí.“ [19]

V názvu sekvenčního diagramu lze používat předponu „sd“, jakožto označení diagramu interakce. Časová osa plyne shora dolů, čáry života postupují vodorovně zleva doprava. Od každé čáry života vede svisle čerchovaná čára, která značí délku jejího života. [19]

Ukázka sekvenčního diagramu se nachází v příloze A. Sekvenční diagram znázorňuje vytvoření nové dovolené. To probíhá následovně: Uživatel s rolí HR se přihlásí pomocí svého účtu Google. Uživatel se přepne do dovolených a zvolí možnost vytvořit novou dovolenou. Systém zobrazí formulář pro vytvoření nové dovolené. Uživatel jej vyplní, odešle a systém novou dovolenou uloží na databáze. Také odešle zprávu vedoucímu uživatele, ve které bude oznámeno vytvoření dovolené a požádání o její schválení. Tento postup využívá v diagramu jednoho aktéra – uživatele v roli HR a několik analytických tříd. Tyto třídy jsou: HolidayController, HolidayService, HolidayDao, Message-Service a MessageDao. HolidayController zachycuje požadavky od uživatele a volá příslušné metody na instancích IHolidayService a IMessageService. Služba dále volá IHolidayDao (IMessageDao), který uloží objekt do databáze.

4.2.2 Diagramy aktivit

Diagramy aktivit slouží k modelování procesů a pracovních postupů jako aktivit za účelem modelování jejich chování. V UML 2 je nově sémantika těchto diagramů založená na tzv. Petriho sítích[9], které zajišťují větší přizpůsobivost a odlišnost od stavových automatů. Diagramy aktivit jsou hojně využívány díky své srozumitelnosti. [19]

Aktivity se skládají z uzlů, které jsou mezi sebou propojeny hranami. Uzly mohou být trojího druhu:

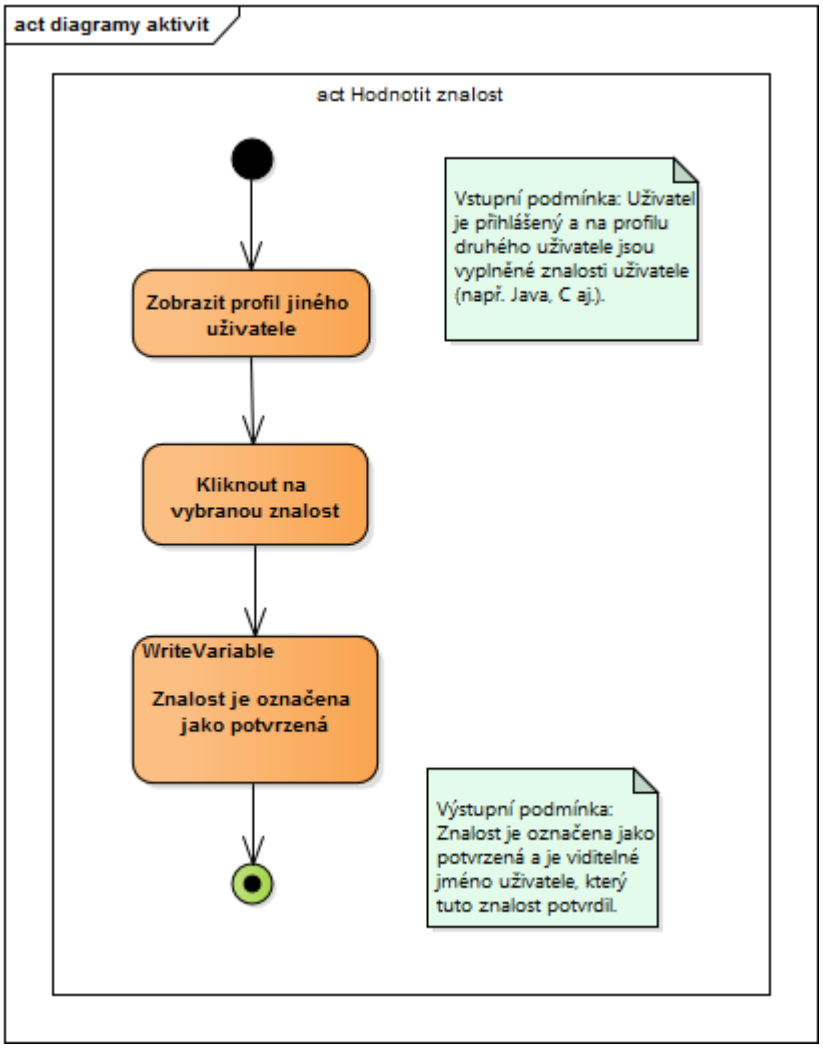
1. **Akční uzly** – jsou to samostatné nedělitelné jednotky.
2. **Řídící uzly** – řídí cestu uvnitř aktivity.
3. **Objektové uzly** – reprezentují jednotlivé objekty. [19]

Hrany se dělí na dva typy:

1. **Řídící hrany** – řídí postup uvnitř aktivity.
2. **Objektové hrany** – jsou to cesty objektů uvnitř aktivity. [19]

Na obrázku níže je ukázka jednoduchého diagramu aktivit. Začíná počátečním uzlem, který je značen jakožto černé kolečko. Pro tento počátek existuje vstupní podmínka, že uživatel musí být přihlášený a druhý uživatel musí mít vyplněnou alespoň jednu znalost (např. Java, C apod.) na svém profilu. Od počátečního uzlu vede řídící hrana do akčního uzlu. Zde se vykoná aktivita „Zobrazit profil jiného uživatele“. Přes další hranu se dostaneme do aktivity „Kliknout na vybranou znalost“ (jedná se o znalost výše zmíněnou ve vstupní podmínce). Uživatel zde vybere znalost, u které si myslí, že ji druhý uživatel plně ovládá a klikne na ni. Následuje hrana vedoucí do aktivity „Znalost je označena jako potvrzená“. Tato aktivita je typu WriteVariable, tedy mění hodnotu proměnné. Znalost se zde nastaví jako potvrzená a je u ní zobrazen uživatel, jenž tuto znalost potvrdil. Diagram aktivity je ukončen hranou, vedoucí do koncového uzlu, který je značen jako zelené kolečko s menším černým kolečkem uvnitř.

[9] Petriho sítě jsou formálním grafickým modelovacím jazykem [19].



Obrázek 10: Diagram aktivit

5 NÁVRH

Tato kapitola se věnuje definici pracovního postupu Návrh. V první podkapitole je probrána problematika návrhového modelu a návrhové třídy. Následuje samostatná kapitola o návrhovém modelu aplikace, kde je model popsán a vysvětlen. Druhá kapitola se zaměřuje na definici datového modelu. Ve dvou následujících podkapitolách jsou probrány pojmy logický a fyzický model, s popsanými modely aplikace.

Cílem analýzy bylo vytyčit hlavní funkce systému. „Smyslem návrhu je přesná specifikace způsobu, jak takové funkce implementovat“. Je třeba vytvořit model systému, který půjde implementovat. [19]

5.1 Návrhový model a návrhová třída

Návrhový model lze také chápat jako rozšíření analytického modelu.

„Návrhové modely se skládají z:

- návrhových podsystémů,
- návrhových tříd,
- rozhraní,
- návrhových realizací případů užití,
- diagramů nasazení.“ [19]

Návrhová třída je stavebním blokem návrhového modelu, který slouží jako základ pro implementaci zdrojového kódu. Třída by měla obsahovat atributy spolu s jejich specifikacemi (název, datový typ, viditelnost, případně výchozí hodnoty) a dále metody s názvem, parametry, viditelností a návratovým typem. Správně formulovaná návrhová třída splňuje tyto podmínky [19]:

1. Třída je úplná a dostačující. Úplnost znamená, že třída obsahuje všechny atributy a metody, které jsou od ní očekávány. Dostatečnost zajišťuje, že všechny metody třídy souvisí s účelem třídy a nejsou nijak překvapující.
2. Třída je jednoduchá. Její metody by měly poskytovat jednoduché a nedělitelné služby.
3. Návrhová třída je vysoce soudržná. Soudržnost třídy se vyznačuje menším počtem odpovědností, které však odpovídají účelu třídy. Tyto třídy jsou snadno srozumitelné, snadno udržovatelné a opakovatelně použitelné.

4. Třída neobsahuje těsné vazby. Třída by měla obsahovat co nejmenší množství vazeb a to tak, aby byla umožněna realizace dané třídy. [19]

5.1.1 Návrhový model systému

Návrhový model vychází z modelu analytického. Jsou v něm uvedeny přesnější informace jako například návratové typy a parametry metod. Z důvodu velké rozsáhlosti návrhového modelu Mraveniště je model umístěn v elektronické příloze. Hlavní princip vrstvené architektury byl již popsán ve fázi analýzy, a proto zde nebude znovu popisován. V příloze B je umístěn návrhový UML diagram, jenž ukazuje vzájemné propojení tříd (stejně jako ve fázi analýzy). Pomocí tohoto diagramu lze lépe pochopit návrhový model aplikace. Entitní třída User zůstává stejná jako v analytickém modelu. Zahrnuje informace o uživateli. Mezi entitní třídy patří i ty, jenž reprezentují vazební tabulky v databázi (například UserRole, UserGroup atp.). Ty většinou mají jen dva atributy datového typu int. Entitní třídy jsou využívány v rámci celé aplikace. A jak již bylo řečeno, dědí od předka AbstractEntity atributy ID a „deleted“. Mapovač UserRowMapper byl již popsán výše a jeho podrobnější popis lze najít v kapitole Implementace. DAO třída UserDao obsahuje metody pro práci s databází. Jedná se převážně o metody pro získání, vkládání, editaci a mazání dat. Parametry metod jsou většinou ID objektu datového typu Integer, ale mohou to být i textové vstupy od uživatele (například při vyhledávání podle tagů uživatel zadá „#mraveniště“ anebo vyhledá obyčejný výraz v textu „mraveniště“) nebo také vkládané objekty (například File, User, Holiday atd.) aj. Entitní třídy implementují svá rozhraní, například UserDao implementuje IUserDao, GroupDao implementuje IGroupDao a tak dále. Servisní třída UserService také implementuje své rozhraní, IUserService. Servisní třídy volají metody DAO tříd. IUserService a IRoleService jsou jako jediné servisní třídy použité i v nastavení bezpečnosti – jejich instance se nacházejí ve třídě AuthoritiesExtractorImpl, která přiděluje přihlášeným uživatelům role. Poslední vrstvou jsou ovladače, které přijímají vstupy od uživatele, respektive zachycují HTTP požadavky, a po zpracování poskytnou uživateli výstup. Metody uvnitř ovladačů často vracejí objekt ResponseEntity<T>, podrobněji popsaný v kapitole Implementace. Parametrem metod je často objekt OAuth2Authentication, jenž v sobě nese informace o přihlášeném uživateli, a umožňuje tak lépe ošetřit přístup k metodám.

5.2 Datový model

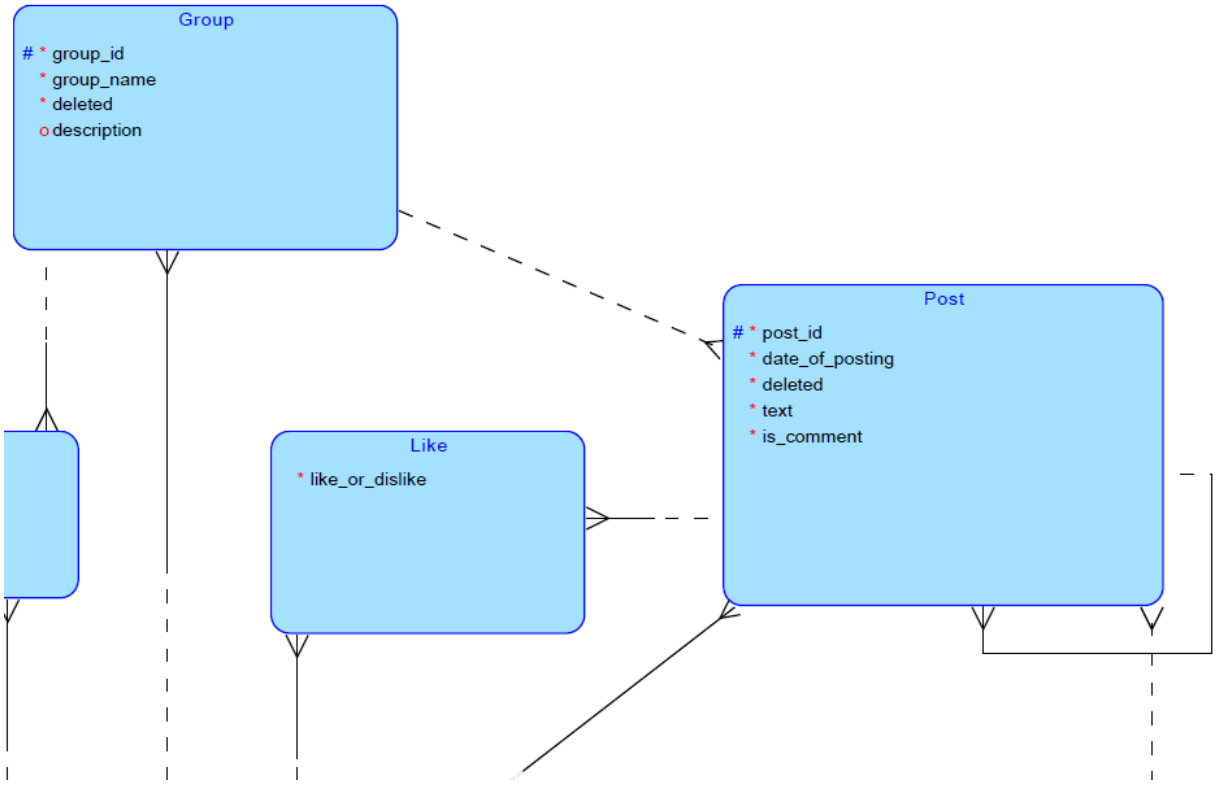
Metodiky datového modelování rozlišují tři základní modely při návrhu databáze: konceptuální, logický a fyzický. Konceptuální model zohledňuje entity a relace mezi nimi. Není vůbec závislý

na fyzické implementaci dat. Logický datový model také není závislý na konkrétním typu relační databáze. Obsahuje entity, relace mezi nimi a také formuje model dat. Fyzický datový model již, oproti logickému modelu, závisí na zvolené databázi, a proto je upraven pro konkrétní DBMS[10]. [23] [24]

5.2.1 Konceptuální model

Prvním krokem při vytváření konceptuálního návrhu databáze je vytvoření tzv. ER modelu. ER (Entity Relation) model zobrazuje entity a vztahy mezi nimi. [24] V modelu existují tři důležité pojmy: entita, atribut a vazba. Entita je datový objekt, který v sobě nese logicky seskupená data. Atribut je položka entity, tedy určitá vlastnost. Vazba vyjadřuje vztahy mezi entitami. [23] V tomto modelu nezáleží na typu relační databáze, ani na datových typech či integritních omezeních [24]. Z uvedených datových modelů je to nejvyšší forma abstrakce. Ukázka takového modelu se nachází níže, na obrázku 11. Tento model byl vytvořen v programu Oracle SQL Developer Data Modeler, ale existují i další vhodné programy. Například ERD Plus, E/R Studio, Microsoft Visio či výše používaný Enterprise Architect a mnoho dalších. V Data Modeleru je však jiné názvosloví, které může některé jedince zmást. Pro tvorbu konceptuálního modelu je třeba vytvořit Logical Model. Konceptuální model na obrázku 10 ukazuje relace mezi několika entitami. Dle značení čar mezi entitami lze rozeznat, o jaký typ relace se jedná a jaké nese vlastnosti. Například vazba mezi entitou Group a Post znamená, že se jedná o vazbu 1: N (tzv. kardinalita). Na obou stranách je vazba volitelná (tzv. parcialita) – pozná se podle přerušované čáry (plná čára značí povinnou vazbu). Vazby mohou být i rekurzivní, jak je vidět u entity Post. Každá entita obsahuje atributy, které jsou vlevo značeny symboly. Tyto symboly určují, zda se jedná o primární klíč (křížek), povinný atribut (hvězdička), či nepovinný atribut (kolečko). Celý model lze nalézt v elektronické příloze.

[10] DBMS (Database Management System) je softwarový systém, který uživateli umožňuje, mimo jiné, vytvářet databázi a manipulovat s ní.



Obrázek 11: Ukázka konceptuálního modelu

5.2.2 Logický model

Logický model vychází z modelu konceptuálního. Při jeho tvorbě se kontrolují tabulky s pomocí normalizace a kontrolují se také integritní omezení a podpora uživatelských transakcí. [24] Logický model lze jednodušše vytvořit v programu Oracle SQL Developer Data Modeler. Po vytvoření konceptuálního modelu (v programu je označen jako Logical Model), lze logický model vygenerovat přes kontextové menu. Pravým kliknutím na *Logical Model* zvolit možnost *Engineer to Relational Model* a zvolit si model, který chceme převést na logický. Po kliknutí na *Engineer* se vygeneruje nový logický model. V následujícím kroku je třeba určit datové typy atributů a jejich velikost. Pro přehlednost je vhodné přejmenovat cizí klíče v tabulkách, aby odpovídaly jejich účelu. Avšak jelikož Mraveniště používá MySQL databázi, byl logický model vytvořen v programu MySQL Workbench. Výsledný model je k dispozici v příloze C. V následujícím textu bude popis samotných tabulek, jejich účelu a relací mezi nimi.

Tabulka User je jednou ze stěžejních částí Mraveniště. Obsahuje základní informace o uživateli. Pro přihlášení uživatele bude použit email. Příznak deleted se nachází téměř ve všech tabulkách a značí, zda byl záznam v tabulce smazán uživatelem či ne. Pokud by byl záznam „smazán“,

bude pouze vyřazený, nebude doopravdy vymazaný (vychází z nefunkčního požadavku). Dalším důležitým atributem je role, který určuje aktuálně přidělenou roli uživatele. Tyto tři atributy jsou v tabulce povinné. Mezi nepovinné patří například datum nástupu a datum ukončení pracovního poměru (start_date, end_date), plat (salary), počet dní zbývají dovolené (holiday_days_left) a nadřízený (leader).

Tabulka Role obsahuje jednoznačný identifikátor role a její název. S touto tabulkou souvisí vazební tabulka User_role, která spojuje uživatele s rolemi (jeden uživatel může mít více rolí).

Holiday znázorňuje přidělenou dovolenou, která má začátek, konec a aktuální stav. Tento stav je cizí klíč tabulky Holiday_state, ve které budou uchovávány stavy dovolených (například schváleno, čeká na schválení, neschváleno). Každá dovolená si nese informaci o uživateli, jehož se dovolená týká.

Idea umožňuje vytvořit uživateli nápad, jenž je součástí tzv. „Nápadníku“. Každý nápad má své id, text, datum přidání nápadu, příznak smazání, text řešení a stav nápadu. Stav nápadu je cizí klíč tabulky Idea_state, ve které budou, podobně jako v Holiday_state, uchovávány stavy nápadů (nový, čeká na realizaci, vyřešen apod.).

Důležitou součástí sítě jsou skupiny, které jsou uloženy v tabulce Group. Každá skupina má své id, název, popis skupiny, příznak smazání a zakladatele skupiny. Zakladatel je opět cizím klíčem, v tomto případě tabulky User. Tabulka User_group je vazební tabulkou mezi User a Group a přiřazuje uživatele do skupiny. Uživatel může patřit do více skupin a zároveň skupina může mít více uživatelů.

Tabulka Post neboli příspěvek obsahuje atributy: id příspěvku, datum přidání příspěvku, příznak smazání, text. Příznak is_comment označuje, zda se jedná o komentář či o běžný příspěvek. Cizí klíč post_fk umožňuje vložit komentář k danému příspěvku. Dalšími cizími klíči jsou user a group. User pochopitelně značí uživatele, jenž příspěvek vložil. Group je skupina, ve které se objevil tento příspěvek. K příspěvku lze také vkládat soubory (file_fk). S příspěvky samozřejmě úzce souvisí „dávání like a dislike“ – jinak řečeno, zda se uživateli příspěvek líbí nebo nelíbí. K tomuto účelu slouží vazební tabulka Like, která přiřazuje uživateli příspěvek a zároveň pomocí příznaku like_or_dislike udává, zda se příspěvek uživateli líbil.

Tabulka Profile uchovává informace na profilu uživatele, jako je jeho jméno a příjmení, pracovní pozice, adresa, telefonní číslo a číslo kanceláře. Také obsahuje cizí klíč tabulky Department (oddělení), ve kterém uživatel pracuje, a fotografii uživatele. Každý uživatel má možnost přidat na profil seznam schopností (znalostí), které mohou ostatní uživatelé potvrzovat. Tyto

znalosti jsou uloženy v tabulce Knowledge, ve formě id a názvu znalosti. Vazební tabulka Profile_knowledge přiřazuje každému profilu sadu znalostí. Oproti ostatním vazebním tabulkám má tato tabulka svůj primární klíč, který je cizím klíčem v tabulce User_approved. User_approved představuje uživatele, kteří potvrdili znalost na profilu jiného uživatele. Příznak approved značí potvrzení či nepotvrzení dané znalosti.

Další částí aplikace je i chat, jenž reprezentuje tabulka Message. Její součástí je id zprávy, datum odeslání zprávy, příznak smazání, text zprávy. Cizími klíči tabulky User jsou zde příjemce a odesílatel (sender, recipient). Pomocí zpráv lze zasílat i soubory (file).

Do tabulky File jsou ukládány soubory různých typů. Typem je zde míněno, jakým způsobem je soubor vnímán v rámci aplikace. Tedy, zda se jedná o výplatnice zaměstnanců, předpisy, formuláře, pracovní smlouvy a podobně. Z důvodu odlišnosti přístupových práv k těmto souborům vznikla tabulka File_type, která obsahuje identifikátor a název typu souboru. Podle tohoto typu se dále odvíjí práva při nahrávání či stahování souboru. V tabulce File se nachází unikátní identifikátor, název souboru, nahraný soubor (uploaded_file) datového typu BLOB, cizí klíč uživatele, datum nahrání souboru a cesta k souboru. Dále může obsahovat také popis a velikost souboru.

Tabulka Page představuje stránky, které jsou editovatelné pomocí WYSIWYG editoru. Nesou údaje o názvu, textu, typu stránky, uživateli a příznaku smazání. Typ stránky (Page_type) určuje, zda se jedná o úvodní stránku, stránku o společnosti, „wikislovník“ či jiné stránky přidané HR oddělením. V případě přidání souboru na stránku, existuje tabulka Page_file, která spojuje soubory a stránky. Opět je to vazební tabulka, řešící vztah N: M.

Tabulka Offer obsahuje informace o pracovních, či vzdělávacích nabídkách. Z důvodu rozdílných přístupových práv, existuje tabulka Offer_type. Offer_type má své id a název typu nabídky. Samotná nabídka má také své id, typ nabídky (offer_type), název nabídky, datum vytvoření nabídky, text a uživatele, který nabídku vytvořil.

Poslední tabulky, Change a Change_type zaznamenávají změny v aplikaci a ukládají je. Buď jsou to změny typu audit (log) nebo upozornění pro uživatele (typy jsou uloženy v tabulce Change_type). Logují se všechny provedené změny jako je vytváření, editace a mazání. Také jsou logovány aktivity týkající se nahrávání a stahování souborů. Upozornění pro uživatele jsou komentáře u příspěvků, nové příspěvky ve skupinách a podobně. Každá změna má své id, text akce, datum změny, typ změny a uživatele, který změnu vyvolal.

5.2.3 Fyzický model

Fyzický návrh databáze již lze implementovat ve zvoleném DBMS. Při jeho tvorbě se analyzují transakce, na jejichž základě se vybírá organizace souborů a indexů. Dále se rozhoduje o implementaci uživatelských pohledů a návrhu bezpečnostních opatření. Nakonec je monitorován a laděn systém během provozu. [24]

Program MySQL Workbench umožňuje vygenerovat potřebný DDL skript, který vychází z logického modelu. Postup pro vygenerování je velmi snadný:

1. Kliknout na *File -> Export -> Forward Engineer to SQL CREATE Script*.
2. V dialogovém okně lze navolit možnosti exportu skriptu do souboru. Například vygenerování DROP a INSERT výrazů, přeskočení vytváření cizích klíčů aj.
3. V dalším kroku následuje kontrola objektů, které chceme vygenerovat. Tyto objekty si můžeme vyfiltrovat.
4. Po kliknutí na tlačítko *Next* se zobrazí vygenerovaný skript, jenž lze případně upravit. Kliknutím na *Finish* se skript uloží do souboru z kroku 1.

Vytvořený DDL skript je k dispozici v elektronické příloze.

6 IMPLEMENTACE

V této kapitole bude popsán pracovní postup implementace a následně budou popsány i implementační detaily systému. V první kapitole bude popsán nástroj Apache Maven a jeho části, včetně životního cyklu. Druhá kapitola je věnována frameworku Spring. V podkapitole se lze dočíst o Spring security. Třetí kapitola se zaměřuje na JDBC. Pátá kapitola se věnuje implementaci Mraveniště a jsou v ní popsány ukázky kódu.

V pracovním postupu implementace je hlavní činností vytvoření spustitelného kódu. K tomu je využíván návrhový model, ze kterého může být tento kód vygenerován. Kromě kódu může v této fázi vzniknout i implementační model, který je součástí návrhového modelu. [19]

Aplikace byla vyvíjena ve vývojovém prostředí IntelliJ IDEA. Ukázky kódů aplikace jsou bez dokumentačních komentářů.

6.1 Apache Maven

Apache Maven je nástroj, který umožňuje sestavení (build) projektu a jeho následnou správu, bez nutnosti znalostí těchto mechanismů. Sestavení projektu je založené na tzv. POM (Project Object Model) souboru a sadě zásuvných modulů (plugin). Kromě jiného, lze Maven také používat pro spouštění jednotkových testů a poskytování informací o vytvořeném projektu. [25] Následující podkapitoly byly čerpány z oficiálních webových stránek Apache Maven.

6.1.1 Životní cyklus sestavení projektu

Maven nabízí celkem tři různé životní cykly: default, clean a site. Cyklus default nabízí nasazení projektu, clean zajišťuje vyčištění projektového adresáře od vygenerovaných souborů vzniklých při sestavení projektu. Cyklus site vytvoří dokumentaci k projektu. [26]

Každý životní cyklus se skládá z jednotlivých fází, které jsou spouštěny sekvenčně. The Apache Software Foundation na svých stránkách uvádějí jako příklad životní cyklus default, který obsahuje následující fáze:

1. Validate – zvaliduje celý projekt.
2. Compile – dojde ke kompilaci zdrojového kódu.
3. Test – otestuje zkompilovaný kód pomocí frameworku na jednotkové testy.
4. Package – zkompilovaný kód zabalí do distribuovatelného formátu.

5. Verify – spustí integrační testy.
6. Install – nainstaluje balíček do lokálního repozitáře.
7. Deploy – zkopíruje finální balíček na vzdálený repozitář. [26]

Pro vykonání všech fází zadáme do příkazové řádky příkaz:

```
mvn deploy
```

V případě instalace na lokální úložiště (bez kopírování na vzdálený repozitář) napíšeme:

```
mvn install
```

Před vykonáním fáze install budou implicitně provedeny i všechny předchozí fáze; tedy validate, compile, test, package, verify a až pak install. Jinak řečeno, fáze nelze spouštět samostatně. [26]

Maven dále nabízí spouštění tzv. cíle pluginu (plugin goal), který není vázaný na výše vysvětlené fáze. Lze ho spustit samostatně (pokud není vázaný na některou fázi) anebo v kombinaci s fázemi cyklu. Například tento příkaz:

```
mvn clean dependency:copy-dependencies package
```

spustí nejprve fázi clean, poté provede cíl dependency:copy-dependencies a nakonec vykoná fázi package. [26]

6.1.2 Adresářová struktura

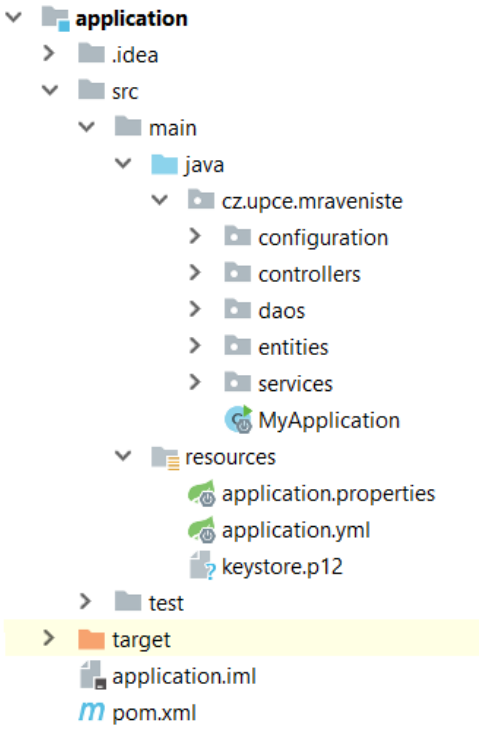
Maven projekt vytváří při výchozím nastavení adresářovou strukturu, která je detailněji popsána v tabulce číslo 12. The Apache Software Foundation doporučují uživatelům, aby dodržovali tuto konvenci. Při jejím dodržení dochází k rychlejší orientaci v projektu. Toto nastavení adresářové struktury lze změnit v projektovém deskriptoru (pom.xml), který se nachází na nejvyšší úrovni. [27]

Tabulka 14: Adresářová struktura Maven projektu. Zdroj: [27]

src/main/java	Aplikační/knihovní zdroje
src/main/resources	Aplikační/knihovní prostředky
src/main/filters	Zdrojové filtrovací soubory
src/main/webapp	Zdroje webové aplikace
src/test/java	Testovací zdroje
src/test/resources	Testovací prostředky

src/test/filters	Zdrojové filtrovací soubory
src/it	Integrační testy (primárně pro pluginy)
src/assembly	Deskriptor sestavení
src/site	Stránky
LICENSE.txt	Projektová licence
NOTICE.txt	Oznámení a přiřazení vyžadovaná knihovnami, které projekt využívá
README.txt	Projektové „přečti mě“

Adresářovou strukturu vyvíjeného projektu lze vidět na obrázku 12. Struktura neobsahuje všechny složky z oficiální dokumentace Maven, ale jen ty nutné. Zdrojové kódy aplikace jsou uloženy v adresáři *src/main/java/cz/upce/mraveniste*. V adresáři *src/main/resources* se nachází konfigurační soubor *application.properties*, který obsahuje nastavení pro spojení s databází aj. (viz kapitola Implementace Mraveniště). Také obsahuje certifikát a konfigurační soubor *application.yml*, popsaný níže. Ve složce target se nachází vygenerované soubory, vzniklé při sestavení projektu. Na nejvyšší úrovni je umístěn *pom.xml*, jehož podrobný popis lze nalézt v následující kapitole.



Obrázek 12: Adresářová struktura Mraveniště

6.1.3 Project Object Model (POM)

Project Object Model (dále POM) je soubor ve formátu XML, jenž obsahuje důležité informace o Maven projektu. POM obsahuje informace jako jsou konfigurační soubory, projektové závislosti, sledování závad, informace o organizaci a vývojářích. Také může ovlivnit životní cyklus sestavení projektu, viz kapitola výše: Životní cyklus sestavení projektu. [28]

Minimální konfigurace pom.xml obsahuje tyto tři povinné položky: groupId, artifactId, version. [28] Jejich popis naleznete níže, spolu s popisem ostatních částí POM souboru, jenž využívá Mraveniště. Část POM souboru vypadá takto:

```
…
<project …>
    <modelVersion>4.0.0</modelVersion>

<groupId>cz.upce.mraveniste</groupId>
<artifactId>application</artifactId>
<version>1.0-SNAPSHOT</version>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>LATEST</version>
</parent>
<properties>
    <java.version>1.8</java.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <version>LATEST</version>
    </dependency>

     …
</dependencies>
<build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Vysvětlení jednotlivých položek podle dokumentace Maven:

- *Project* – hierarchicky nejvyšší položka, povinná.
- *ModelVersion* – určuje verzi POM souboru, který projekt využívá. Je také povinná.

- *GroupId* – obsahuje unikátní identifikátor organizace či skupiny (většinou je to doménové jméno organizace).
- *ArtifactId* – obsahuje unikátní název artefaktu, vygenerovaného projektem.
- *Version* – značí verzi vygenerovaného artefaktu.
- *Parent* – obsahuje informace o rodiči, od kterého projekt bude dědit informace.
- *Properties* – umožňují změnit nastavení projektu.
- *Dependencies* – seznam závislostí nutných k sestavení projektu.
- *Build* – obsahuje sadu prvků, přičemž v aplikaci je využíván prvek zásuvného modulu (plugin). Moduly jsou spouštěny v průběhu sestavení projektu. [28]

Úplný seznam všech položek a jejich význam lze nalézt na adrese https://maven.apache.org/ref/3.5.4/maven-model/maven.html.

6.1.4 Profily

Profily nejsou v této aplikaci využívány, a proto zde budou jen krátce představeny.

Profily (profiles) umožňují individuální nastavení sestavení projektu. Jeden projekt lze poté spouštět s různými parametry. To lze využít například při práci v týmu (každý člen bude mít jinak nastavené parametry) nebo odděleních. Profily tímto způsobem umožňují přenositelnost. [29]

Existují čtyři typy profilů: pro projekt, pro uživatele, globální a deskriptor profilu. Profil pro projekt je definovaný v samotném *pom.xml* souboru. Profil pro uživatele je umístěn v souboru *%USER_HOME%/.m2/settings.xml*. Globální se nachází v konfiguraci Maven, a to konkrétně v *${maven.home}/conf/settings.xml*. Poslední typ profilu je v základním adresáři projektu, v souboru s názvem *profiles.xml*. [29]

Profil můžeme explicitně aktivovat zadáním možnosti *-P* do příkazové řádky [29]:

```
mvn groupId:artifactId:goal -P profile-1, profile-2
```

6.2 Spring Framework

Spring Framework je open-source řešení založené na platformě Java, které slouží k vývoji Java aplikací. Spring se skládá z modulů, které umožňují vlastní rozvržení aplikace, a tím pádem lze moduly omezit jen na ty, které jsou potřeba. [30]

6.2.1 Spring Boot

Spring Boot je řešení založené na frameworku Spring, které nabízí mnohem rychlejší vývoj aplikací. Není potřeba XML konfigurace ani generování kódu, je tedy vhodný pro vývojáře, které začínají s vyvíjením ve Spring frameworku. [30]

Mraveniště využívá Spring Boot v nejnovější verzi, aktuálně 2.0.3. Spouštěcí třída obsahuje anotaci @SpringBootApplication, která spustí autokonfiguraci, provede sken všech balíčků a umožní registraci dalších bean v kontextu, případně konfiguračních tříd [31]. Takto vypadá spouštěcí třída Mraveniště:

```
1       package cz.upce.mraveniste;
2
3       import org.springframework.boot.SpringApplication;
4       import org.springframework.boot.autoconfigure
5               .SpringBootApplication;
6
7       @SpringBootApplication
8       public class MyApplication {
9
10          public static void main(String[] args) {
11              SpringApplication.run(MyApplication.class, args);
12          }
13      }
```

Spring Boot vyžaduje instalaci Javy alespoň ve verzi 8, Spring Framework verze 5.0.8. a Maven verze 3.2 [31]. Dále je třeba přidat do *pom.xml* potřebné závislosti. Ty jsou vidět v ukázce POM souboru v kapitole s názvem Project Object Model (POM).

6.2.2 Anotace

Anotace ve Springu slouží k mnoha účelům:

- automatická konfigurace,
- injektování objektových závislostí,

- označení tříd a metod (například entita, služba, repozitář, bean, ovladač, konfigurační třída aj.)
- zachycení požadavků z webu
- usnadnění práce s třídami, které přistupují k datům
- a další. [32]

Anotace se zapisují ve tvaru *@NázevAnotace* před název třídy, objektu, či metody. Přehled Spring anotací nabízí například John Thompson na svých webových stránkách [32].

Aplikace využívá následující anotace:

- *@SpringBootApplication* – anotace spustí autokonfiguraci, provede sken všech balíčků a umožní registraci dalších bean v kontextu, případně konfiguračních tříd [31].
- *@Autowired* – implicitně injektuje objektovou závislost [32].
- *@Service* – třída označená touto anotací poskytuje určité služby, v našem případě byznys logiku [32].
- *@Transactional* – anotace umožňuje řízení transakcí za pomoci AOP „proxies“ a metadat [30].
- *@Repository* – označuje třídu, která přímo přistupuje do databáze [32]. V aplikaci to jsou všechny DAO třídy, jenž pracují s databází.
- *@Controller* – anotace označuje třídu jako ovladač (controller), který bude zachytávat požadavky od uživatele [32]. Po zachycení požadavku následuje akce, která je definovaná uvnitř metody s příslušnou HTTP metodou (GET, POST, DELETE...) a url. Například GET požadavek *https://localhost:8443/user/1* v aplikaci způsobí aktivaci UserControlleru, který má anotaci *@RequestMapping("/user").* Následně se zavolá metoda getUserById(…) s anotací *@GetMapping("/{id}")*, která ve výsledku vrátí uživatele s ID = 1.
- *@RequestMapping("/user")* – anotace slouží k zachycení požadavků a následném zavolání metod, které mají tento požadavek obsloužit (handler) [32].
- *@GetMapping("/all")* – mapuje HTTP GET požadavky na příslušné zachycující metody. Stejné jako bychom použili *@RequestMapping(method = RequestMethod.GET)*

[32]. Stejně tak se používají i metody PUT, POST a DELETE. Metod je samozřejmě mnohem více, avšak Mraveniště využívá jen tyto čtyři.

- *@PreAuthorize("hasAuthority('ROLE_ADMIN')")* – anotace je v Mraveništi používána u metod ovladačů a určuje, která role má právo tuto metodu využívat. V tomto případě by metodu mohli používat pouze uživatelé v roli administrátora. Anotace může obsahovat i metodu *hasAnyAuthority*, díky které lze zvýšit počet rolí, jež budou mít oprávnění.
- *@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")* – tato anotace se vyskytuje u několika atributů tříd, kde je třeba ručně zadávat datum (například u dovolených – od kdy do kdy). Zformátuje datum podle šablony (pattern) tak, aby odpovídalo formátu dat v databázi.
- *@Component* – při označení třídy touto anotací ji mechanismus pro skenování komponent přidá do aplikačního kontextu [32].
- *@Configuration* – anotace označuje třídu, jenž obsahuje konfiguraci aplikace.
- *@EnableOAuth2Sso* – umožňuje použití OAuth2 autentizace.
- *@EnableGlobalMethodSecurity(prePostEnabled = true)* – díky této anotaci je umožněno využití anotací *@PreAuthorize()* u jednotlivých metod.

6.2.3 Spring security

Spring security je framework určený k zabezpečení podnikových Java aplikací. Zaměřuje se především na autentizaci a autorizaci. Autentizace je proces ověření přístupu uživatele (zařízení, či jiného systému) do aplikace (například jestli má uživatel A s uživatelským jménem XXX a heslem YYY povolený přístup do systému). Autorizace je proces, navazující na autentizaci a zajišťující ověření práv (například jestli může uživatel A provést operaci X). [35]

Spring Security podporuje různé autentizační metody, z nichž většina je poskytována třetími stranami. Zde uvedu pouze některé z nich [35]:

- HTTP autentizace založená na IETF RFC standardu,
- LDAP (Lightweight Directory Access Protocol),
- OpenID,

- CAS (Jasig Central Authentication Service),
- JAAS (Java Authentication and Authorization Service),
- Kerberos,
- JOSSO (Java Open Source Single Sign-On),
- vlastní autentizační systémy.

Jak již bylo řečeno výše, Spring Security poskytuje také možnost autorizace. Tu je možné implementovat na tři způsoby: autorizace webových požadavků, autorizace při volání metod a autorizace přístupu k instancím objektů. [35]

Jedním z požadavků na systém byla možnost registrace a přihlášení pomocí externího mechanismu. Z důvodu rozšířeného využívání služeb od společnosti Google, byl zvolen framework OAuth 2.0 (autentizace pomocí Google Sign-in).

OAuth definuje 4 role, které se účastní při procesu autentizace. Jsou to: vlastník zdroje (resource owner), zdrojový server (resource server), klient a autorizační server (authorization server). Vlastník zdroje kontroluje zdroje, ke kterým přistupují klientské požadavky. Zdroje jsou umístěny na zdrojovém serveru. Autorizační server vydává tzv. přístupové „tokeny“, které obsahují různé přístupové atributy, díky nimž klient prokáže svou totožnost u zdrojového serveru. [36]

Postup při autentizaci je následující [36] [35]:

1. Klient vyšle požadavek na přístup ke zdrojům, který zachytí vlastník zdroje.
2. Vlastník zdroje odešle klientovi autorizační grant (authorization grant).
3. Klient odešle tento grant autorizačnímu serveru.
4. Autorizační server odešle zpět přístupový „token“.
5. Klient odešle přístupový „token“ zdrojovému serveru.
6. V případě validního „tokenu“ server poskytne klientovi přístup ke zdrojům.

Při nastavování OAuth 2.0 bylo postupováno podle oficiální dokumentace Spring Security, viz zdroj [35].

Prvním krokem při nastavení OAuth autentizace je registrace aplikace v Google API. Stačí se přihlásit na svůj Google účet a přejít na adresu *https://console.developers.google.com/apis/*, kde si v levém menu zvolíte možnost „*Credentials*“ a vytvoříte nový projekt s novým pověřením (credential) OAuth Client ID. Důležité je i nastavení adresy pro přesměrování, poté, co je klient

autentizován. Po vytvoření je vygenerováno klientské ID a tajemství (secret). V samotné aplikaci si poté vytvoříme soubor s názvem *application.yml* a umístíme ho do složky se zdroji. Při konfiguraci OAuth 2.0 bylo postupováno podle oficiálního návodu na stránkách https://spring.io/guides/tutorials/spring-boot-oauth2. Soubor application.yml byl podle tohoto návodu inspirován [37]:

```
security:
    oauth2:
        client:
            scope: email, openid, profile
            clientId: KLIENTSKÉ_ID
            clientSecret: KLIENTOVO_TAJEMSTVÍ
            accessTokenUri: https://accounts.google.com/o/oauth2/token
            userAuthorizationUri: https://accounts.google.com/o/oauth2/auth
            clientAuthenticationScheme: form
            preferTokenInfo: false
        resource:
            userInfoUri: https://www.googleapis.com/userinfo/v2/me
```

Za `KLIENTSKÉ_ID` a `KLIENTOVO_TAJEMSTVÍ` u doplníme údaje, které jsme vygenerovali v předchozím kroku. V návodu oficiální dokumentace však není zmíněný povinný parametr *scope*. v němž jsou přenášeny informace o uživateli (po úspěšné autentizaci). Může nabývat různých hodnot, v závislosti na použití aplikace. Jejich seznam lze najít na adrese *https://developers.google.com/identity/protocols/googlescopes*. Tato aplikace využívá přihlášení pomocí Google Sign-in, takže hodnoty parametru scope mohou být pouze tři: profile, email, openid. Profile zajišťuje vrácení základních informací o profilu uživatele, email zobrazí email uživatele a openid zajistí autentizaci pomocí OpenID Connect. [38] [39]

6.3 JDBC

Java Database Connectivity (JDBC) je průmyslový standard pro nezávislé připojení mezi programovacím jazykem Java a širokou škálou databází. Poskytuje rozhraní určené pro přístup k SQL databázím. [33] Lze přistupovat i k jiným formátům dat, jako jsou *.xlsx* a *.csv* soubory, případně k textovým souborům (*.txt*). Pro přístup k databázi existuje tzv. JDBC ovladač (JDBC driver), který překládá příkazy do nativních volání databáze. Každá databáze má svůj vlastní ovladač. [34]

Nastavení datových zdrojů aplikace je umístěno v souboru *src/main/resources/application.properties*:

```
1       spring.datasource.driver-class-name=com.mysql.jdbc.Driver
2       spring.datasource.url=jdbc:mysql://localhost:3311/dp
3       spring.datasource.username=****
4       spring.datasource.password=****
```

Údaj na řádku číslo jedna říká, že pro navázání připojení do databáze MySQL se použije JDBC ovladač (uvádí se zde plný název ovladače). U Spring Boot tento údaj není povinný; Spring Boot automaticky detekuje název ovladače [31]. Na druhém řádku lze vidět nastavení URL připojení do konkrétní databáze. Je uváděno ve formátu *{typ připojení}:{typ databáze}://{host}:{port}* a za ním může být uveden název schématu */{schéma}*. Třetí a čtvrtý řádek udávají přístupové údaje do databáze jakožto uživatelské jméno a heslo. Následují řádky, které nejsou povinné a pouze doplňují nastavení připojení:

```
5       spring.datasource.connectionProperties=useUnicode=true;
6                       characterEncoding=UTF-8;
7       spring.datasource.tomcat.max-wait=20000
8       spring.datasource.tomcat.max-active=50
9       spring.datasource.tomcat.max-idle=20
10      server.tomcat.max-http-header-size=52428800
```

Řádek 5 je nastavuje kódování na UTF-8. Další řádky konfigurují připojení k aplikačnímu serveru Apache Tomcat. Řádek číslo 7 omezuje dobu čekání na připojení na 20000 ms; po této době bude vyhozena výjimka. Řádek 8 deklaruje maximální počet aktivních spojení připojení ze stejného „bazénu“ (pool), v dané chvíli. Řádek 9 definuje maximální počet připojení, jenž bude v „bazénu“ uchováván. [31] Řádek 10 udává maximální velikost HTTP hlavičky v bajtech [45].

```
11      spring.servlet.multipart.max-file-size=5MB
12      spring.servlet.multipart.max-request-size=5MB
```

Řádky 11 a 12 řeší problematiku odesílání a stahování souborů. Nejprve je nastavena maximální velikost souboru a následně maximální velikost požadavku. Následující konfigurace se zabývá nastavením SSL:

```
13      server.port: 8443
14      server.security.require-ssl=true
15      server.ssl.key-store: classpath:keystore.p12
16      server.ssl.key-store-password: Mravenec
```

```
17      server.ssl.key-store-type=PKCS12
18      server.ssl.key-alias=tomcat
```

Řádek 13 definuje číslo portu, řádek 14 požaduje použití SSL. Poté je uvedena cesta k certifikátu a s ním související heslo. Dále je uveden typ certifikátu a jeho alias.

6.4 Implementace Mraveniště

V této kapitole budou probrány jednotlivé části aplikace spolu s ukázkami zdrojových kódů. Nejprve budou popsány entitní třídy, poté DAO třídy a na ně navazující servisní třídy. Nakonec budou vysvětleny ovladače a konfigurace aplikace.

Třída Loader v adresáři configuration slouží k načtení zaváděcích dat. V této třídě je před spuštěním potřeba upravit email uživatele a případně jeho roli. Před druhým spuštěním aplikace je nutné třídu zakomentovat. Uvnitř třídy Loader jsou vytvářeny instance entitních tříd, které jsou následně ukládány. Z tohoto důvodu třídy User, Role, ChangeType, FileType, OfferType, PageType, HolidayState, IdeaState a Department implementují rozhraní Serializable.

6.4.1 Entitní třídy

Entitní třídy již byly výše popsány v kapitole Návrh, proto zde uvedu pouze stručný popis. Všechny entity dědí od abstraktní třídy AbstractEntity, jenž obsahuje dva atributy: id a deleted. ID značí jednoznačný identifikátor v databázi a deleted je příznak smazání. Ve třídě jsou dále metody pro nastavení a vrácení těchto proměných. Entitní třídy obsahují atributy, metody pro nastavení (setter) a vrácení (getter) proměnných a přetíženou metodu toString() pro výpis objektu.

S entitními třídami souvisí mapovače, které mapují databázové tabulky na objekty v aplikaci. Ukázka mapovače pro entitní třídu Change:

```
1       public class ChangeRowMapper implements RowMapper<Change> {
2
3           @Override
4           public Change mapRow(ResultSet resultSet, int i) throws
5                                SQLException {
6               Change change = new Change();
7               change.setId(resultSet.getInt("change_id"));
8               …
9               change.setDeleted(resultSet.getBoolean("deleted"));
10              return change;
11          }
12      }
```

Mapovače implementují třídu RowMapper<T> a přetěžují metodu mapRow(ResultSet resultSet, int rowNum). Uvnitř metody je nejprve vytvořena instance entitní třídy, které jsou následně nastavovány hodnoty z ResultSet. Parametrem pro získání objektu z ResultSet je název sloupečku v tabulce (zde tabulka Change). Metoda nakonec vrátí objekt, naplněný hodnotami z databáze.

6.4.2 DAO třídy

DAO třídy umožňují práci s databází. Každá z DAO tříd v Mraveništi implementuje své rozhraní. Ukázka rozhraní IUserDao je k dispozici níže. Z důvodu rozsáhlosti textu zde nejsou uvedeny dokumentační komentáře.

```
1       public interface IUserDao {
2           List<User> getAllUsers();
3           User getUserById(int userId);
4           User getUserByEmail(String email);
5           User addUser(User user);
6           void updateUser(User user);
7           void deleteUser(int userId);
8           boolean userExists(String email);
9           boolean userExists(int id);
10          void setHolidayDays(Integer count, Integer userId);
11          boolean hasUserRole(Integer userId, Integer roleId);
12          void changeRole(Integer userId, Integer roleId);
13          void addUserRole(Integer userId, Integer roleId);
14          List<Role> getAllUserRoles(Integer userId);
15      }
```

Jak již bylo řečeno, toto rozhraní implementuje třída UserDao. Stejně jako ostatní DAO třídy, obsahuje instanci JdbcTemplate, díky které je umožněna práce s databází:

```
1       @Transactional
2       @Repository
3       public class UserDao implements IUserDao {
4
5           @Autowired
6           private JdbcTemplate jdbcTemplate;
```

Metody většinou začínají inicializací SQL dotazu datového typu String. Na místa, kam se budou dosazovat hodnoty, jsou vloženy otazníky. Následuje vytvoření instance mapovače (viz řádek deset). V případě GET požadavku na konkrétní objekt, je další část metody obalena do bloku try-catch, aby v případě prázdného výsledku nespadla aplikace. Pomocí instance jdbcTemplate

je zavolána metoda queryForObject, jejímiž parametry jsou: SQL dotaz z řádku osm, mapovač a nakonec hodnoty proměnných, které chceme dosadit. Zde se jedná o vrácení uživatele dle zadaného ID, takže dosazujeme proměnnou userId. Jelikož nechceme vrátit smazaného uživatele, nastavíme automaticky příznak deleted na false. JdbcTemplate se dotáže databáze a vrátí výsledek, jenž je následně vrácen metodou getUserById.

```
7       public User getUserById(int userId) {
8           String sql = "SELECT * FROM user WHERE user_id = ?
9                               AND deleted =? LIMIT 1";
10          RowMapper<User> rowMapper = new UserRowMapper();
11          try {
12              User user = jdbcTemplate.queryForObject(sql, rowMapper,
13                              userId, false);
14              return user;
15          } catch (EmptyResultDataAccessException e) {
16              return null;
17          }
18      }
```

JdbcTemplate nabízí i další metody pro práci s databází. V Mraveništi je často využívána metoda query, jenž dokáže z databáze vrátit více objektů. Další často používanou metodou je metoda update, která je přítomna u všech vkládacích, editovacích a mazacích metod. Metody pro vkládání do databáze se přesto liší. Tento odlišný přístup byl inspirován zdrojem [48]. MySQL nativně nevrací hodnotu ID vloženého objektu, a proto se vkládací metody liší od ostatních:

```
19      public User addUser(User user) {
20          KeyHolder holder = new GeneratedKeyHolder();
21
22          String sql = "INSERT INTO user (email, end_date, salary,
23                      holiday_days_left, start_date, leader_fk,
24                      current_role_fk) VALUES (?,?,?,?,?,?,?)";
25
26          jdbcTemplate.update(new PreparedStatementCreator() {
27              @Override
28              public PreparedStatement createPreparedStatement(Connection
29                      connection) throws SQLException {
30                  PreparedStatement ps = connection.prepareStatement(sql,
31                      Statement.RETURN_GENERATED_KEYS);
32                  ps.setString(1, user.getEmail());
33                  …
34                  return ps;
35              }
36          }, holder);
37
38          int userId = holder.getKey().intValue();
49          user.setId(userId);
40          addUserRole(userId, user.getRole());
41          return user; }
```

Nejprve je vytvořena instance KeyHolder, která v sobě bude udržovat hodnotu vygenerovaného ID. Následně je inicializován SQL dotaz, který je zpracován pomocí PreparedStatementCreator. Podobně jako v mapovači, jsou jednotlivé atributy třídy namapovány do sloupců v databázi (avšak obráceně). JdbcTemplate zpracuje PrepareStatementCreator a po zpracování nastaví vygenerované ID z databáze KeyHolderu. ID je poté nastaveno vytvořenému uživateli a uživatel je vrácen zpět.

6.4.3 Servisní třídy

Každá servisní třída implementuje své rozhraní, které je později použito v ovladači. Rozhraní servisních tříd jsou velmi podobná rozhraním DAO, a proto je zde nebudu uvádět. Samotné servisní třídy začínají anotací *@Service,* viz kapitola výše. Uvnitř jsou vytvořené instance IDao tříd a anotací *@Autowired*. Metody pro vrácení objektů není v této části nutné ošetřovat, a tak vypadají následovně:

```
public User getUserById(int userId) { return userDao.getUserById(userId); }
```

Naopak metody, které editují obsah databáze, jsou na této vrstvě ošetřeny:

```
1       public User addUser(User user, String role) {
2               if (userDao.userExists(user.getEmail())) {
3                   return null;
4               } else {
5                   Role roleReturned = roleDao.getRoleByName(role);
6                   if (roleReturned == null) {
7                       return null;
8                   } else {
9                       user.setRole(roleReturned.getId());
10                      return userDao.addUser(user);
11                  }
12              }
13      }
```

Například při vkládání nového uživatele musí být zajištěno, že zadaný email v databázi neexistuje (viz řádek dva). Role uživatele naopak existovat musí (viz řádky pět, šest). V případě splnění obou podmínek je uživateli nastavena role a uživatel je vložen do databáze pomocí výše popsané metody.

6.4.4 Ovladače (controllers)

Ovladače zachycují požadavky od uživatele pomocí anotací *@GetMapping* (případně Post, Put apod.). U některých metod následuje anotace *@PreAuthorize*, která zajišťuje další stupeň autorizace. Níže bude popsána metoda pro vrácení uživatele dle zadaného ID.

Na řádku 3 lze vidět, že metoda vrací ResponseEntity<User>. ResponseEntity vrátí daný objekt a také stav HTTP (HttpStatus, viz řádek šest). Parametrem je ID uživatele, které je součástí URL adresy. Tuto možnost zajišťuje anotace *@PathVariable("nazev_promenne_v_url")*. Kromě této je v Mraveništi také využívána anotace *@ModelAttribute("model")*, umožňující vložení objektu ve formě formuláře. Názvy atributů se ve formuláři musí shodovat s názvy atributů dané entity. Poslední anotací uvnitř parametrů metod je anotace *@RequestParam*, zajišťující povinnost odeslání parametru metody. Uvnitř getUserById je volána stejnojmenná metoda getUserById, která vrátí konkrétního uživatele a status OK (tedy číselně stav 200). Veškeré ovladače využívají instance servisních tříd s anotací *@Autowired*.

```
1       @GetMapping("/{id}")
2       @PreAuthorize("hasAnyAuthority('ROLE_HR', 'ROLE_MANAZER')")
3       public ResponseEntity<User> getUserById(@PathVariable("id") Integer
4                                               id) {
5           User user = userService.getUserById(id);
6           return new ResponseEntity<User>(user, HttpStatus.OK);
7       }
```

Další metoda, kterou bych zde ráda zmínila jakožto ukázku, je přidání nového uživatele. Na řádku 12 lze vidět často používaný parametr metod – OAuth2Authentication authentication. Tento objekt je předáván automaticky při přihlášení (autentizaci) uživatele. Obsahuje informace o uživateli, například email, id, jméno a příjmení, fotografii. Obsahuje ale i další informace, například seznam rolí a informace, zda byl uživatel autentizován aj. Tyto informace nejsou součástí Mraveniště, ale jsou to informace, jenž uživatel vyplnil na svém účtu Google. Pomocí metody na řádku 16 jsou z objektu authentication vráceny detaily o uživateli. O řádek níže je z detailů „vytažen“ email, pod kterým se uživatel přihlásil. Abychom však získali kompletní informace o uživateli Mraveniště, je třeba využít servisní třídu a získat tak uživatele dle zadaného emailu. Stejně tak je získána i zadaná role.

```
8       @PostMapping("add")
9       @PreAuthorize("hasAnyAuthority('ROLE_HR', 'ROLE_MANAZER')")
10      public ResponseEntity<String> addUser(@ModelAttribute("add_user")
```

```
11                              User user, @RequestParam String roleName,
12                              OAuth2Authentication authentication) {
13
14          LinkedHashMap<String, String> details =
15                      (LinkedHashMap<String, String>)
16                      authentication.getUserAuthentication().getDetails();
17
18          String email = details.get("email");
19          User user1 = userService.getUserByEmail(email);
20          Role r = roleService.getRoleByName(roleName);
21          String message = "";
```

Následně je ověřena existence vedoucího uživatele (řádek 22). V případě, že vedoucí nebyl registrován (neexistuje), je vrácena zpráva s chybovým stavem BAD_REQUEST (400).

```
22      User leader = userService.getUserById(user.getLeader());
23      if (leader == null) {
24          message = "Nadřízený musí být registrovaný!";
25          return new ResponseEntity<String>(message,
26                      HttpStatus.BAD_REQUEST);
27      }
```

Potom je nový uživatel vložen do databáze a zároveň je ověřeno, že se vložení do databáze zdařilo. V případě úspěchu, je zapsán záznam do tabulky Change (jakožto logování aktivity) a vrácen stav OK. Pokud se vložení do databáze nezdařilo, je vrácen chybový stav. HttpStatus může nabývat různých hodnot, v Mraveništi jsou však využívány pouze následující: OK (200 – úspěch), BAD_REQUEST (400 – chyba v požadavku), FORBIDDEN (403 – uživatel nemá oprávnění), NO_CONTENT (404 – nenalezeno).

```
28      User newUser = userService.addUser(user, roleName);
29      if (newUser != null) {
30          message = "Uživatel byl přidán.";
31          String action = "Uživatel " + user1.getId() + " přidal
32                              uživatele " + newUser.getId();
33          changeService.addChange(email, action, 1);
34          return new ResponseEntity<String>(message, HttpStatus.OK);
35      } else {
36          message = "Uživatel nebyl přidán!";
37          return new ResponseEntity<String>(message,
38                      HttpStatus.BAD_REQUEST);
39      }
40      }
```

6.4.5 Nastavení zabezpečení

Konfigurace byla nastavena podle dokumentace Spring Security [35].

Globální nastavení HTTP požadavků se nachází ve třídě SecurityConfiguration. Tato třída dědí od třídy WebSecurityConfigurerAdapter metodu configure(HttpSecurity http). Metoda požaduje od všech příchozích požadavků autentizaci (řádek deset) a následnou autorizaci na základě role uživatele (řádky 11 a 12). Požadavky, jejichž URL začíná na /login či /error nepotřebují žádnou autentizaci (řádek patnáct). Řádek 16 zajišťuje automatické odhlášení uživatele při zadání cesty /logout (výchozí nastavení frameworku Spring) a po úspěšném odhlášení jej přesměruje na stránku /login?logout (sedmnáctý řádek). Na řádku níže je zakázána CSRF ochrana, která v naší aplikaci není potřeba.

```
1       @Configuration
2       @EnableOAuth2Sso
3       @EnableGlobalMethodSecurity(prePostEnabled = true)
4       public class SecurityConfiguration extends
5       WebSecurityConfigurerAdapter {
6
7           @Override
8           protected void configure(HttpSecurity http) throws Exception {
9               http
10                      .authorizeRequests()
11                      .antMatchers("/role/**", "/type/**",
12                              "/change/**").hasAuthority("ROLE_ADMIN")
13                      .anyRequest().hasAnyAuthority("ROLE_UZIVATEL",
14                              "ROLE_HR", "ROLE_ADMIN", "ROLE_MANAZER")
15                      .antMatchers("/login**", "/error**").permitAll()
16                      .and().logout()
17                     .logoutSuccessUrl("/login?logout").permitAll();
18              http.csrf().disable();
19          }
20      }
```

Na řádku 12 je použita metoda hasAnyAuthority(…), které předchází nastavení autorit (authorities). V Mraveništi je to přiřazení rolí autentizovaným uživatelům. To zajišťuje třída AuthoritiesExtractorImpl, jenž imlementuje třídu AuthoritiesExtractor. Implementace této třídy byla inspirována podle B. Vulaje [46]. Metoda extractAuthorities(…) nejprve získá ze scope (viz application.properties) email, pod kterým se uživatel přihlásil. Následně je zavolána metoda rozhraní servisní třídy IUserService na získání uživatele podle emailu. Poté je použito rozhraní IRoleService, pomocí něhož je získána a vrácena role přihlášeného uživatele.

```
1       @Component
2       public class AuthoritiesExtractorImpl implements
3               AuthoritiesExtractor {
4           @Autowired
5           public IUserService userService;
6
```

```
7           @Autowired
8           public IRoleService roleService;
9
10          @Override
11          public List<GrantedAuthority> extractAuthorities(Map<String,
12                      Object> map) {
13              String username = (String) map.get("email");
14              User user=null;
15              try {
16                  user= userService.getUserByEmail(username);
17              } catch(Throwable e) {
18                  return Collections.<GrantedAuthority> emptyList();
19              }
20
21              if (user == null) {
22                  return Collections.<GrantedAuthority> emptyList();
23              }
24              Role r = roleService.getRoleById(user.getRole());
25              return AuthorityUtils.createAuthorityList(r.getRoleName());
26          }
27      }
```

Poslední třída týkající se nastavení je ConnectorConfiguration. Tato třída zajišťuje přesměrování z nezabezpečeného portu 8080 na zabezpečený portu 8443. SSL zabezpečení bylo vytvářeno pomocí návodů ze zdrojů [47] a [50]. Třída ConnectorConfiguration je čerpána ze zdroje [47]. Postup pro nastavení SSL s vygenerováním vlastního certifikátu byl následující:

1. Spustit příkazovou řádku pod právy Administrátora a přepnout se do složky Java JDK nejnovější verze, adresář bin (příkaz „cd cesta“). Zde je implicitně nainstalovaný program keytool, určený pro vytváření certifikátů. Dále je třeba zadat do příkazové řádky následující řádek:

   ```
   keytool -genkey -alias tomcat -storetype PKCS12 -keyalg RSA -keysize
   2048 -keystore keystore.p12 -validity 3650
   ```

2. Následně jsme vyzvání k zadání hesla a údajů o společnosti. Je vhodné si heslo zapamatovat. Údaje o společnosti apod. není nutné vyplňovat.

3. Výsledkem je vygenerovaný soubor „keystore.p12“, obsahující náš certifikát. Soubor je vygenerován do složky, odkud byl příkaz spuštěn. V našem případě je to složka ~/Java/jdk…/bin. Tento soubor je třeba nakopírovat do projektu, do složky Resources.

4. V projektu se dále nastaví soubor application.properties, viz kapitola JDBC. Poté je třeba zajistit přesměrování z HTTP (port 8080) na HTTPS (port 8443). K tomu slouží třída ConnectorConfiguration, viz výše. Tuto třídu zde z kapacitních důvodů nebudu uvádět. Tímto je nastavené SSL zabezpečení a další krok není povinný.

5. Klienti jsou nyní upozorňování na nebezpečnou stránku. Pro odstranění tohoto upozornění lze exportovat certifikát pro klienty, a to pomocí tohoto příkazu:

   ```
   keytool -export -keystore keystore.jks -alias tomcat -file mujCertifikat.crt
   ```

   Tento certifikát následně importujeme do prohlížeče.

Vlastní certifikát samozřejmě není vhodnou volbou do firemního prostředí. Z finančních důvodů nebyl koupen certifikát od certifikační autority. V aplikaci ovšem není problém vlastní certifikát nahradit jiným, který je pro firemní prostředí vhodnější.

7 TESTOVÁNÍ

Tato kapitola se věnuje vysvětlení základních pojmů v oblasti testování. V první podkapitole jsou popsány čtyři metodiky testování, jak je uvádí ve své knize Bureš a kolektiv. Ve druhé podkapitole jsou uvedeny způsoby testování softwaru. V poslední části je rozebrán ukázkový testovací scénář aplikace.

Testování softwaru lze dle Blacka a spol. definovat jako proces, skládající se z aktivit životního cyklu vývoje softwaru. Tyto aktivity mohou být statické, dynamické, mohou zahrnovat plánování, přípravy a hodnocení produktu. Jejich účelem je zjistit, jestli produkt splňuje zadané požadavky, zda správně funguje a také odhalit jeho chyby. [44]

7.1 Testovací metodiky

Metodika je souhrn principů, který může být aplikován na určité řešení situace. V našem případě se jedná o metodiky testování softwaru. Bureš a kolektiv zavádějí celkem čtyři metodiky. Jedná se o V-model, W-model, agilní přístup a DevOps. [43]

7.1.1 V-model

V-model je variací na jeden z často používaných modelů při vývoji systému – vodopádový model. Název „V-model“ vznikl díky jeho tvaru do písmene V. Bureš a spol. jej popisují následovně: „V-model tvoří několik dvojic aktivit (specifikace požadavků – uživatelské akceptační testy, hrubý návrh – systémové testy, detailní návrh – integrační testy, programování – jednotkové testy). Na levé straně se přitom nacházejí aktivity související se specifikací a návrhem, na pravé aktivity testování (…).“ [43] Z V-modelu vychází, že systém je třeba testovat během všech fází vývoje softwaru, a nejen pouze po dokončení vývoje. Časové okamžiky, během kterých je software testován, jsou nazývány úrovně testování. Nevýhodou V-modelu je jeho idealizace, jelikož realita bývá jiná. Úrovně testování většinou nejsou dodržovány a model bývá také chybně interpretován z hlediska časového nástupu testování (testuje se až výsledný produkt). [43]

7.1.2 W-model

W-model zdůrazňuje především včasné zapojení testerů do procesu vývoje. Testeři by se měli zapojit již při specifikování požadavků na systém a pokračovat v testování při každé fázi vývoje

až do uživatelských akceptačních testů (ty jsou krátce popsány níže v kapitole Způsoby testování softwaru). První „V“ v tomto modelu představuje fáze vývoje softwaru od specifikací požadavků až po instalaci hotového systému. Druhé „V“ přiřazuje k jednotlivým fázím vývoje softwaru určité fáze testování. Testování začíná testováním specifikací a končí uživatelskými akceptačními testy. Tyto dvě „V“ se překrývají a navzájem přiřazují ke každé fázi vývoje fázi testování. Dohromady tvoří písmeno „W“, podle kterého je model také pojmenován. Mimo tento model jsou vypsány další typy testů, které se neřadí ke konkrétní úrovni testování. V praxi se opět tento model velice těžko dodržuje. [43]

7.1.3 Agilní přístup

Základem agilního přístupu je důraz na kvalitu produktu, která pozitivně ovlivňuje vztahy mezi vývojáři a testery. Tato metodika není standardizována a v každém podniku je jinak prováděna. Bureš a kolektiv ji vidí spíše jako sdílení znalostí a zkušeností formou tzv. komunit testerů. Tento přístup nedodržuje tradiční metodiky vývoje ani standardizaci. [43]

7.1.4 DevOps

DevOps je metodika, která spojuje oddělení vývoje, provozu a testování. Rozšiřuje koncept agilní metodiky vývoje o oblast provozu systémů. Její název je složení slov **Dev**elopment (vývoj) a **Op**eration**s** (provoz). Metodika v sobě zahrnuje několik elementů, uvedených pod zkratkou CAMS (Culture, Automation, Measurement, Sharing). Tyto prvky znamenají změnu organizační struktury, automatizaci, průběžná měření a sdílení odpovědností a problémů. [43]

7.2 Způsoby testování softwaru

Patton ve své knize *Testování softwaru* uvádí dva způsoby testování softwaru, které jsou rozděleny podle různých kritérií.

Prvním kritériem je testerova znalost zdrojového kódu:

- **Černá skříňka** – tester zadává vstupy a dostane výstupy daného software, ale neví, jak vnitřně funguje.
- **Bílá skříňka** – tester má přístup ke zdrojovému kódu. Nevýhodou tohoto přístupu je tzv. testování na míru (tester přizpůsobí testy programu). [40]

Pattonovým dalším kritériem je spuštění softwaru:

- **Statické testování** – program se nemusí spouštět. Testují se například specifikace softwaru.
- **Dynamické testování** – program se testuje spuštěný; testuje se především funkčnost. [40]

Kitner toto rozdělení doplňuje o kritérium testování v různých částech vývoje softwaru. Dle fáze testování rozděluje testy na následující [41]:

- **Jednotkové testy** (Unit tests) – první testy píší programátoři a ověřují jimi funkčnost zdrojového kódu.
- **Testy modulů** – jsou podobné jednotkovým testům, ale mají větší rozsah (používají se na větších projektech). Ověřují funkčnost modulu nebo komponenty.
- **Integrační testy** – testují komunikaci mezi jednotlivými komponentami, případně mezi různými systémy, či mezi modulem a hardwarem a podobně [41] [42].
- **Funkční testy** – testy ověřují funkčnost části softwaru, nejedná se pouze o testy jednotlivých metod. Funkční testy se dále dělí na smoke testy, sanity testy, regresní testy a testy použitelnosti. Tento typ testování bude předveden na příkladu níže.
- **Systémové testy** – poslední testy na straně vývojového týmu, které ověřují celkovou funkčnost systému podle požadavků zákazníka.
- **Akceptační testy** – testují funkčnost systému na straně zákazníka. [41]

Jedním z dalších základních dělení testování je dle způsobu provádění testů. Testy lze realizovat **manuálně**, **automatizovaně** a **exploravitně**. Manuální testování je prováděno testerem, který se řídí testovacími případy. Automatizované testování využívá programovací jazyky (Java, Python aj.), díky kterým jsou vytvářeny programy, ověřující testovací scénáře. Ty lze opakovaně (i automaticky) spouštět, a to bez nutné přítomnosti testera. V dnešní době jsou k dispozici frameworky, které tuto práci usnadňují (z vlastní zkušenosti jmenuji framework Selenium). Explorativní testování vychází z manuálního testování, ale na rozdíl od něj, se tester neřídí podle testovacích případů. [41]

Důležitou součástí testování je také vedení dokumentace. Do této dokumentace lze zařadit tyto části:

- **Testovací plán** – popisuje metodiku testování, kvalitativní cíle, úkoly a další.

- **Seznam testovacích případů** – v každém testovacím případu jsou popsány testované části a kroky, podle kterých bude software otestován.
- **Záznamy o chybách** – popis problémů a chyb vzniklých při testování. Chyby mohou být uchovávány v databázi, či vedeny uvnitř specializovaného software na jejich evidenci (například Redmine).
- **Metriky, statistiky, souhrny** – umožňují přehlednější sledování postupu při testování. [40]

7.3 Funkční testy aplikace

Pro otestování funkčnosti aplikace byl využíván nástroj Restlet client, který umožňuje snadné odesílání a přijímání HTTP požadavků.

Tabulka pro testovací případ byla inspirována webovými stránkami http://testovanisoftwaru.cz/dokumentace-v-testovani/test-case/.

Tabulka 15 uvádí testovací případ číslo 1, který ověřuje funkčnost externího přihlášení uživatele.

Tabulka 15: Testovací případ „Externí přihlášení uživatele“

ID	1
Název testovacího případu	Externí přihlášení uživatele
Účel	Ověření funkčnosti externího přihlašování
Typ testu	Funkční
Čas	1 minuta
Počáteční podmínky	Registrovaný uživatel s rolí „ROLE_UZIVATEL“
Kroky	1. Uživatel vyšle požadavek na přístup na stránku, která je k dispozici pouze pro přihlášené uživatele s rolí „ROLE_UZIVATEL“. 2. Uživateli je zobrazen formulář Google Sign-in. 3. Uživatel zadá přihlašovací údaje a stiskne tlačítko přihlásit se. 4. Uživatel je přesměrován na stránku, kterou požadoval v kroku 1.
Očekávaný výsledek	1. Odeslaný požadavek na zobrazení stránky.

	2. Zobrazení formuláře pro přihlášení proběhne úspěšně. 3. Uživatel zadá správné údaje pro přihlášení. 4. Aplikace provede autorizaci uživatele a zobrazí požadovanou stránku.
Výsledek testu	Prošel
Poznámky	Žádné

Postup při realizaci testovacího případu se řídil kroky, uvedenými v tabulce 13. Uživatel nejprve zadal adresu, jenž vyžaduje autentizovaného uživatele. Tato práce se nezabývá front endem, a proto není zatím uživatelsky přívětivá – uživatel musí zadávat celou adresu URL. Například zobrazení všech nápadů ve firmě: „https://localhost:8443/idea/all“. Po stisknutí klávesy Enter je zobrazen přihlašovací formulář, ve kterém si uživatel vybere účet, který je registrovaný v aplikaci. Pokud by vybral jiný účet, jenž není registrován, aplikace mu přístup zamítne. Stejně tak mu zamítne přístup, pokud by neměl správná oprávnění (roli). Po přihlášení jsou uživateli zobrazeny všechny nápady ve formátu JSON: „`[{"id":1,"deleted":false,"ideaText":"Nový automat na bagety","solution":"Koupen nový automat.","stateId":3,"userId":6,"creationDate":"2018-06-23"},…]`“.

Dalším ukázkovým testem je verifikace vytvoření profilu uživatele, jehož testovací případ je v tabulce číslo 16.

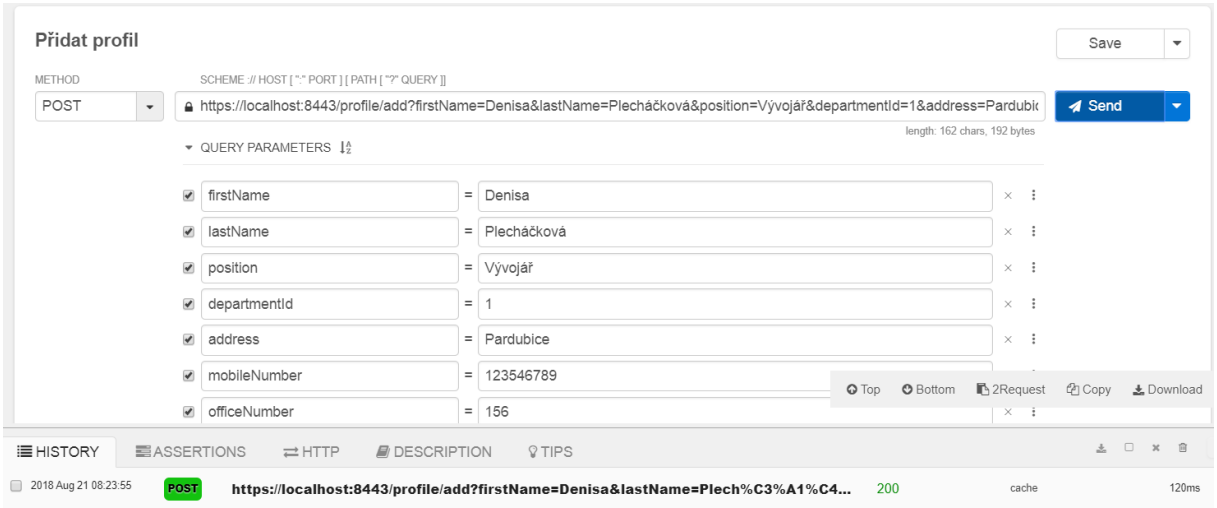
Tabulka 16: Testovací případ „Založení profilu uživatele“

ID	2
Název testovacího případu	Založení profilu uživatele
Účel	Ověření funkčnosti založení profilu
Typ testu	Funkční
Čas	5 minut
Počáteční podmínky	1. Uživatel je již registrovaný v systému. 2. Uživatel je přihlášený pod některou z rolí.
Kroky	1. Uživatel odešle požadavek na vytvoření profilu. 2. Uživatel vyplní údaje ve formuláři a odešle jej. 3. Systém vytvoří profil uživatele.
Očekávaný výsledek	1. Požadavek na vytvoření profilu je úspěšně odeslán.

	2. Je zobrazen formulář pro vytvoření profilu a po jeho vyplnění je úspěšně odeslán. 3. Systém zpracuje údaje a pokud neexistuje profil se stejnou emailovou adresou (uživatelem), je vytvořen nový. Jinak systém nahlásí chybu.
Výsledek testu	Prošel
Poznámky	Žádné

Při testovaní této části aplikace byl využit Restlet client, zmíněný výše. Nejprve proběhlo přihlášení uživatele a následně byly vyplněny údaje v Restlet client. Nástroj umožňuje ukládání různých typů požadavků a testovacích scénářů. Také je možné přidávat podmínky (assertions), které ale nebyly využívány, protože jsou zabudovány uvnitř aplikace. Rozšíření nabízí i tvorbu výrazů (expressions) a sady proměnných (environments). Mimo jiné lze s jeho pomocí testovat odesílání souborů, což bylo náležitě využito.

Po vyplnění údajů stačí kliknout na tlačítko „*Send*“, které odešle požadavek. Vzápětí je vidět výsledek požadavku v záložce „*History*“, viz obrázek 13. K dispozici jsou informace například o datu a čase, HTTP metodě, URL adrese, stavovém kódu, či době trvání v milisekundách. Pokud sjedete níže, je zobrazena hlavička a tělo požadavku. V našem případě proběhlo přidání profilu úspěšně, se stavovým kódem 200. Pro úplné ověření byl odeslán GET požadavek na vypsání všech profilů, kde byl poté vidět i námi vytvořený profil.



Obrázek 13: Restlet client

ZÁVĚR

Cílem této diplomové práce bylo vytvořit back end podnikové sociální sítě. Aplikace byla vytvořena podle zvolené metodiky Unified Process, jejíž fáze byly popsány v jednotlivých kapitolách.

Před zahájením vývoje aplikace byla provedena rešerše produktů, jenž jsou momentálně rozšířené na trhu. Tato rešerše se nachází v první kapitole a popisuje vybrané podnikové sociální sítě a jejich vlastnosti.

Ve druhé kapitole byla popsána zvolená metodika vývoje webové aplikace.

Vývoj aplikace odstartoval proces vytváření požadavků na aplikaci. Požadavky byly vytvářeny na základě dodaného dokumentu od zadavatele. Část požadavků vznikla po konzultaci se zadavatelem. Požadavky byly roztříděny do dvou skupin – na funkční a nefunkční, jejichž popis a úplný seznam je uveden v kapitole číslo tři. Část požadavků byla prodiskutována se zadavatelem, přičemž některé požadavky byly upraveny, doplněny nebo vznikly nové.

Na tuto kapitolu navazuje fáze analýzy, ve které byly vysvětleny některé základní pojmy. Pomocí metody Analýzy podstatných jmen a sloves byl vytvořen analytický model tříd. Další metodou analýzy byla realizace případů užití ve formě sekvenčních diagramů a diagramů aktivit. Pomocí analýzy podstatných jmen a sloves byla odhalena většina tříd a metod. Zbytek tříd, atributů a sloves byl logicky odvozen na základě požadavků na systém.

Čtvrtá kapitola byla věnována pracovnímu postupu návrh. Analytický model tříd z předchozí kapitoly dal vznik návrhovému modelu tříd, který byl podkladem pro samotnou implementaci aplikace. V této fázi také vznikly datové modely; konkrétně konceptuální, logický a fyzický model.

Po této fázi následovala implementace. Nejprve byl vytvořen projekt pomocí nástroje Apache Maven, s využitím vývojového prostředí IntelliJ Idea. Potom byly naprogramovány všechny třídy dle návrhového modelu a byla vytvořena databáze dle fyzického modelu. V průběhu implementace probíhalo i testování aplikace a zároveň bylo doplněno několik nových metod.

V šesté kapitole bylo popsáno testování aplikace. Kapitola se věnovala problematice testování a popisovala metodiky a kategorie testů. Nakonec byla uvedena praktická ukázka toho, jak probíhalo testování Mraveniště. Při odhalení chyby byla okamžitě provedena oprava a proběhlo opětovné testování.

Cílem práce bylo vytvoření funkční podnikové sociální sítě, která umožňuje přihlašovat uživatele, kteří již mají založený svůj Google e-mail, a přiřazovat jim role. Přihlášení uživatelé mohou vytvářet své profily, přidávat příspěvky, vytvářet nápady, zakládat skupiny a posílat si zprávy. Uživatelé dostávají upozornění na komentáře u jejich příspěvku či při přidání příspěvku do jejich skupiny. Nechybí ani možnost „dávat Like a Dislike“, či schvalovat znalosti na profilech ostatních uživatelů. Uživatelé s oprávněnými rolemi mohou přidávat samostatné stránky, s možností nahrávání a stahování souborů. Soubory je možné přidávat i do příspěvků a zpráv. Aplikace také podporuje správu dovolených. Administrátor Mraveniště má možnost spravovat role uživatelů, typy souborů, stránek a nabídek, stavy dovolených a nápadů. Také vidí všechny změny, které byly v Mraveništi provedeny (zaznamenávání aktivit uživatelů). Uživatelé s rolí „ROLE_HR“ mohou přidávat pracovní a vzdělávací nabídky a registrovat nové uživatele.

Výsledkem práce je funkční aplikace, která splňuje všechny výše zmíněné požadavky zadavatele. Aplikace dodržuje vrstvenou architekturu frameworku Spring. Aplikace je také zabezpečena pomocí protokolu SSL a její součástí jsou dokumentační komentáře. Cíle diplomové práce byly splněny. Nasazení aplikace bylo plánováno na zkušební verzi Google Cloud, ale bylo zjištěno, že využívá jiný aplikační server. V současné době se čeká na vyjádření zadavatele k alternativnímu hostingu. Poté bude řešení nasazeno a případně upraveno dle potřeby.

POUŽITÁ LITERATURA

[1] YAMMER TEAM. New in Yammer – building a more connected and engaged organization. *Microsoft* [online]. © Microsoft 2018, 2017 [cit. 2018-02-07]. Dostupné z: https://blogs.office.com/en-us/2017/05/16/new-in-yammer-building-a-more-connected-and-engaged-organization/.

[2] Yammer REVIEW: Collaboration Software. *FinancesOnline.com* [online]. Poland, 2017 [cit. 2018-02-07]. Dostupné z: https://reviews.financesonline.com/p/yammer/.

[3] REDMOND, Wash. Microsoft to Acquire Yammer. *Microsoft.com* [online]. San Francisco: © Microsoft 2018, 2012 [cit. 2018-02-07]. Dostupné z: https://news.microsoft.com/2012/06/25/microsoft-to-acquire-yammer/.

[4] Enterprise Social Network (ESN) Vendor Comparison: Research Report. In: *Dialogue consulting* [online]. Melbourne: © Dialogue Consulting, 2013 [cit. 2018-02-08]. Dostupné z: http://www.dialogueconsulting.com.au/documents/2013/05/report-enterprise-social-networking-platform-vendor-comparison.pdf.

[5] EDWARDS, Richard. Ovum Decision Matrix: Selecting an Enterprise Social Networking Product, 2016–17. In: *Jive* [online]. © Ovum, 2016, 2016 [cit. 2018-02-08]. Dostupné z: https://www.jivesoftware.com/pdf/analystreport/2016/11/ovum-analyst-report_selecting_esn-products.pdf.

[6] 2017 Enterprise Social-Collaboration Marketplace Analysis. In: *Real Story Group* [online]. Real Story Group, 2017, 1-Mar-2017 [cit. 2018-02-08]. Dostupné z: https://www.realstorygroup.com/Blog/3168-2017-Enterprise-SocialCollaboration-Marketplace-Analysis.

[7] Jive REVIEW: Social Collaboration Software. *FinancesOnline.com* [online]. Poland, 2017 [cit. 2018-02-10]. Dostupné z: https://reviews.financesonline.com/p/jive/.

[8] *Jive software* [online]. Portland, Oregon: Jive Software, 2018 [cit. 2018-02-13]. Dostupné z: https://www.jivesoftware.com.

[9] *IBM Connections* [online]. New York, United States: IBM Corporation [cit. 2018-02-13]. Dostupné z: https://www.ibm.com/us-en/marketplace/ibm-connections.

[10] VOIGTS, Richard Jan. Vycházejí IBM Connections V6.0. *ITBIZ* [online]. Nitemedia, 2017, 12. květen 2017, 1 [cit. 2018-02-15]. Dostupné z: http://www.itbiz.cz/clanky/vychazeji-ibm-connections-v6-0.

[11] *Whitesoft* [online]. Praha: Whitesoft, ©2014-2015 [cit. 2018-02-15]. Dostupné z: https://www.whitesoft.cz/.

[12] Confluence: Features & Functions. *Atlassian: Atlassian support* [online]. Sydney: Atlassian, ©2018, 2017 [cit. 2018-02-18]. Dostupné z: https://confluence.atlassian.com/confeval/confluence-evaluator-resources/confluence-features-functions.

[13] HAVLOVÁ, Jaroslava. RSS. In: *KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV)* [online]. Praha: Národní knihovna ČR, 2003- [cit. 2018-02-18]. Dostupné z: http://aleph.nkp.cz/F/?func=direct&doc_number=000014607&local_base=KTD.

[14] The tool jungle in the corporate world – which platform performs well?. *Swisscom* [online]. Švýcarsko: Swisscom, ©2018, 2017 [cit. 2018-03-03]. Dostupné z: https://www.swisscom.ch/en/business/enterprise/themen/work-smart/technologie-vergleich-workstream-collaboration.html.

[15] Connect, collaborate, and take action. *Salesforce* [online]. San Francisco (United States): Salesforce.com, ©2018, 2018 [cit. 2018-03-06]. Dostupné z: https://www.salesforce.com/products/chatter/features/.

[16] SAP Jam Collaboration: Cloud Collaboration Software and Tools. *SAP* [online]. Walldorf: SAP Global Corporate Affairs, 2018 [cit. 2018-03-06]. Dostupné z: https://www.sap.com/products/enterprise-social-collaboration.html.

[17] SharePoint 2016, Team Collaboration Software Tools. *Microsoft* [online]. Redmond (Washington): Microsoft, ©2018 [cit. 2018-03-07]. Dostupné z: https://products.office.com/en-us/sharepoint/collaboration.

[18] Features | Slack. *Slack* [online]. San Francisco: Slack, 2018 [cit. 2018-03-07]. Dostupné z: https://slack.com/features.

[19] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2*., aktualiz. a dopl. vyd. Brno: Computer Press, 2008. ISBN 978-80251-1503-9.

[20] JANOVSKÝ, Dušan. Editory HTML stránek. *Jak psát web* [online]. Praha, 2009 [cit. 2018-03-11]. Dostupné z: https://www.jakpsatweb.cz/editory.html.

[21] Webová aplikace (Web Application). *Management Mania* [online]. Wilmington (USA): ManagementMania.com, ©2011-2016, 2016 [cit. 2018-03-14]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application.

[22] FOWLER, Martin. *Destilované UML*. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

[23] KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně. 2*., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 80-251-1083-4.

[24] CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. *Mistrovství – databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.

[25] Maven – Introduction. *Apache Maven Project* [online]. Wakefield: The Apache Software Foundation, 2018 [cit. 2018-07-13]. Dostupné z: https://maven.apache.org/what-is-maven.html.

[26] Introduction to the Build Lifecycle. *Apache Maven Project* [online]. Wakefield: The Apache Software Foundation, 2018 [cit. 2018-07-13]. Dostupné z: https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html.

[27] Introduction to the Standard Directory Layout. *Apache Maven Project* [online]. Wakefield: The Apache Software Foundation, 2018 [cit. 2018-07-13]. Dostupné z: https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html.

[28] POM Reference. *Apache Maven Project* [online]. Wakefield: The Apache Software Foundation, 2018 [cit. 2018-07-13]. Dostupné z: https://maven.apache.org/pom.html.

[29] Introduction to Build Profiles. *Apache Maven Project* [online]. Wakefield: The Apache Software Foundation, 2018 [cit. 2018-07-13]. Dostupné z: https://maven.apache.org/guides/introduction/introduction-to-profiles.html.

[30] 1. Introduction to Spring Framework. *Spring* [online]. Pivotal Software, ©2018 [cit. 2018-07-14]. Dostupné z: https://docs.spring.io/spring/docs/5.0.x/spring-framework-reference/overview.html#overview.

[31] *Spring Boot Reference Guide: 2.0.3.RELEASE*. WEBB, Phillip a kolektiv. ©2012-2018. Dostupné také z: https://docs.spring.io/spring-boot/docs/current/reference/pdf/spring-boot-reference.pdf.

[32] THOMPSON, John. *Spring Framework annotations* [online]. Spring Framework Guru, ©2015 [cit. 2018-07-14]. Dostupné z: https://springframework.guru/about/.

[33] Java SE Technologies – Database. *Oracle* [online]. Argentina: Oracle, [2018] [cit. 2018-07-15]. Dostupné z: http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

[34] ŠEDA, Jan. Úvod do JDBC. *Interval.cz* [online]. Brno: ZONER software, 2003 [cit. 2018-07-15]. Dostupné z: https://www.interval.cz/clanky/uvod-do-jdbc/.

[35] ALEX, Ben. Spring Security Reference. Spring [online]. Pivotal Software, ©2004-2017 [cit. 2018-07-15]. Dostupné z: https://docs.spring.io/spring-security/site/docs/5.0.0.RELEASE/reference/htmlsingle/.

[36] The OAuth 2.0 Authorization Framework. HARDT, Dick a kolektiv. *IETF Tools* [online]. Internet Engineering Task Force (IETF), 2012 [cit. 2018-07-16]. Dostupné z: https://tools.ietf.org/html/rfc6749.

[37] Spring Boot and OAuth2. *Spring* [online]. Pivotal Software, ©2018 [cit. 2018-07-17]. Dostupné z: https://spring.io/guides/tutorials/spring-boot-oauth2.

[38] OAuth 2.0 for Server-side Web Apps. *Google Developers* [online]. 2017 [cit. 2018-07-18]. Dostupné z: https://developers.google.com/identity/protocols/OAuth2?csw=1.

[39] OAuth 2.0 Scopes for Google APIs. *Google Developpers* [online]. 2017 [cit. 2018-07-18]. Dostupné z: https://developers.google.com/identity/protocols/googlescopes.

[40] PATTON, Ron. *Testování softwaru: automatické i ruční testování, testování použitelnosti, lokalizace i kompatibility produktů nejen pro manažery softwarových projektů a testery, praktická cvičení na konci kapitol*. Praha: Computer Press, c2002. Programování. ISBN 80-7226-636-5.

[41] KITNER, Radek. Typy testování software. *KITNER* [online]. Modřice: Kitner, c2015 [cit. 2018-07-19]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/.

[42] HLAVA, Tomáš. Fáze a úrovně provádění testů. *Testování softwaru* [online]. Hlava, 2011 [cit. 2018-07-19]. Dostupné z: http://testovanisoftwaru.cz/metodika-testovani/druhy-typy-a-kategorie-testu/faze-testu/.

[43] BUREŠ, Miroslav, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ. *Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu*. Praha: Grada, 2016. Profesionál. ISBN 978-80-247-5594-6.

[44] BLACK, Rex, Erik VAN VEENENDAAL a Dorothy GRAHAM. *Foundations of software testing: ISTQB certification*. Third edition. United Kingdom: Cengage Learning EMEA, c2012. ISBN 9781408044056.

[45] Apache Tomcat Configuration Reference. *The Apache Software Foundation* [online]. Apache Software Foundation, ©1999-2012 [cit. 2018-08-14]. Dostupné z: https://tomcat.apache.org/tomcat-5.5-doc/config/http.html.

[46] VULAJ, Brandon. Mapping your Users and Roles with Spring Boot OAuth2. *Medium* [online]. Medium, 2017 [cit. 2018-08-14]. Dostupné z: https://medium.com/@bvulaj/mapping-your-users-and-roles-with-spring-boot-oauth2-a7ac3bbe8e7f.

[47] CHAKRABORTY, Sajal. Spring Boot SSL [https] Example. *HowToDoInJava* [online]. USA: HowToDoInjava.com, 2017 [cit. 2018-08-14]. Dostupné z: https://howtodoinjava.com/spring-boot/spring-boot-ssl-https-example/.

[48] DHIRAJ. *Fetch Auto Generated Primary Key Value After Insert* [online]. DevGlan, 2017 [cit. 2018-08-14]. Dostupné z: https://www.devglan.com/spring-jdbc/fetch-auto-generated-primary-key-value-after-insert-spring-jdbc.

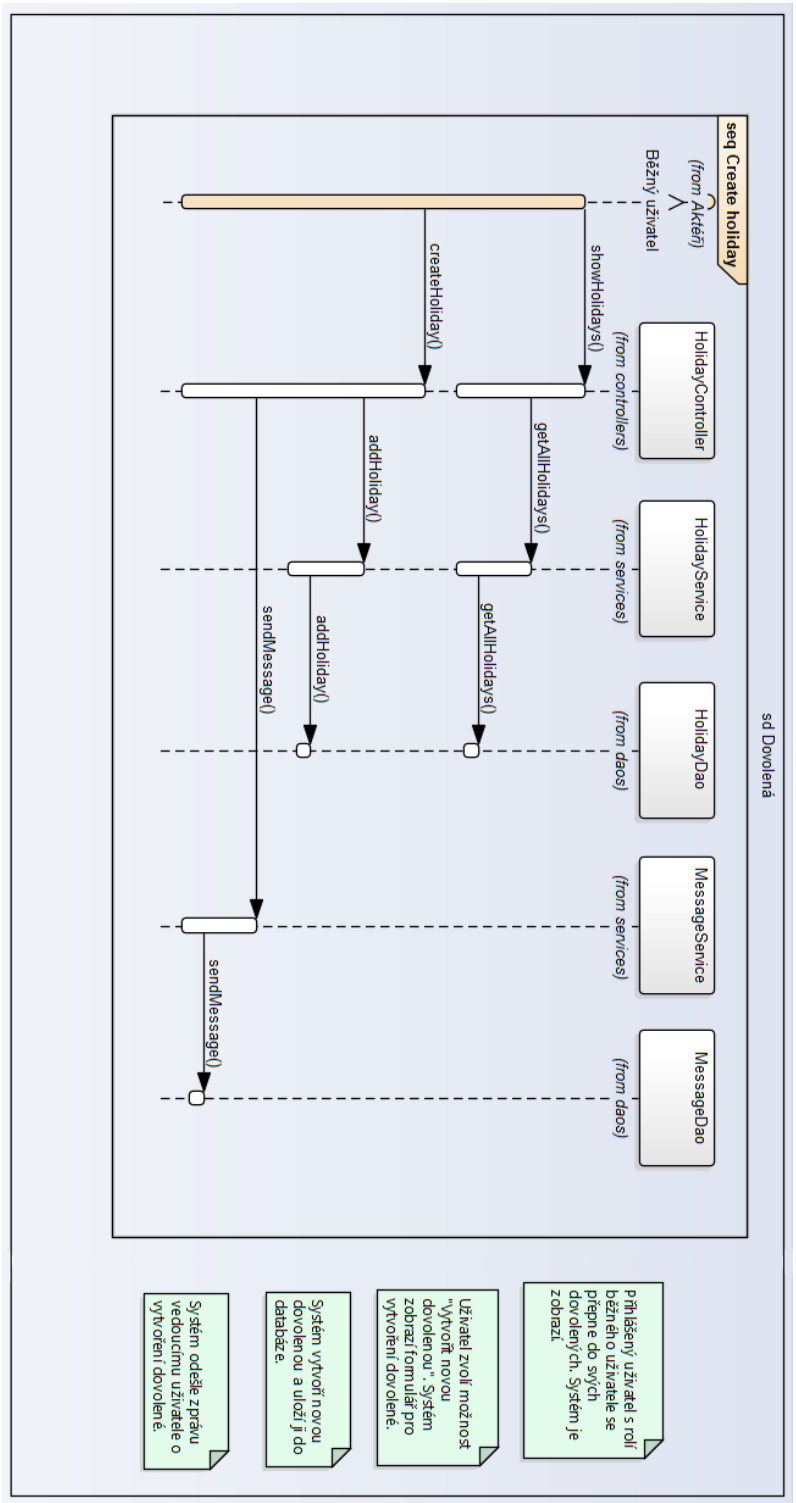
[49] HO, Clarence a Rob HARROP. *Pro Spring 3*. New York: Distributed to the Book trade worldwide by Springer Science+Business Media, c2012. Expert's voice in Spring. ISBN 978-1-4302-4107-2.

[50] VITALE, Thomas. How to enable HTTPS in a Spring Boot Java application. *Thomas Vitale* [online]. Dánsko: Thomas Vitale, 2017 [cit. 2018-08-14]. Dostupné z: https://www.thomasvitale.com/https-spring-boot-ssl-certificate/.

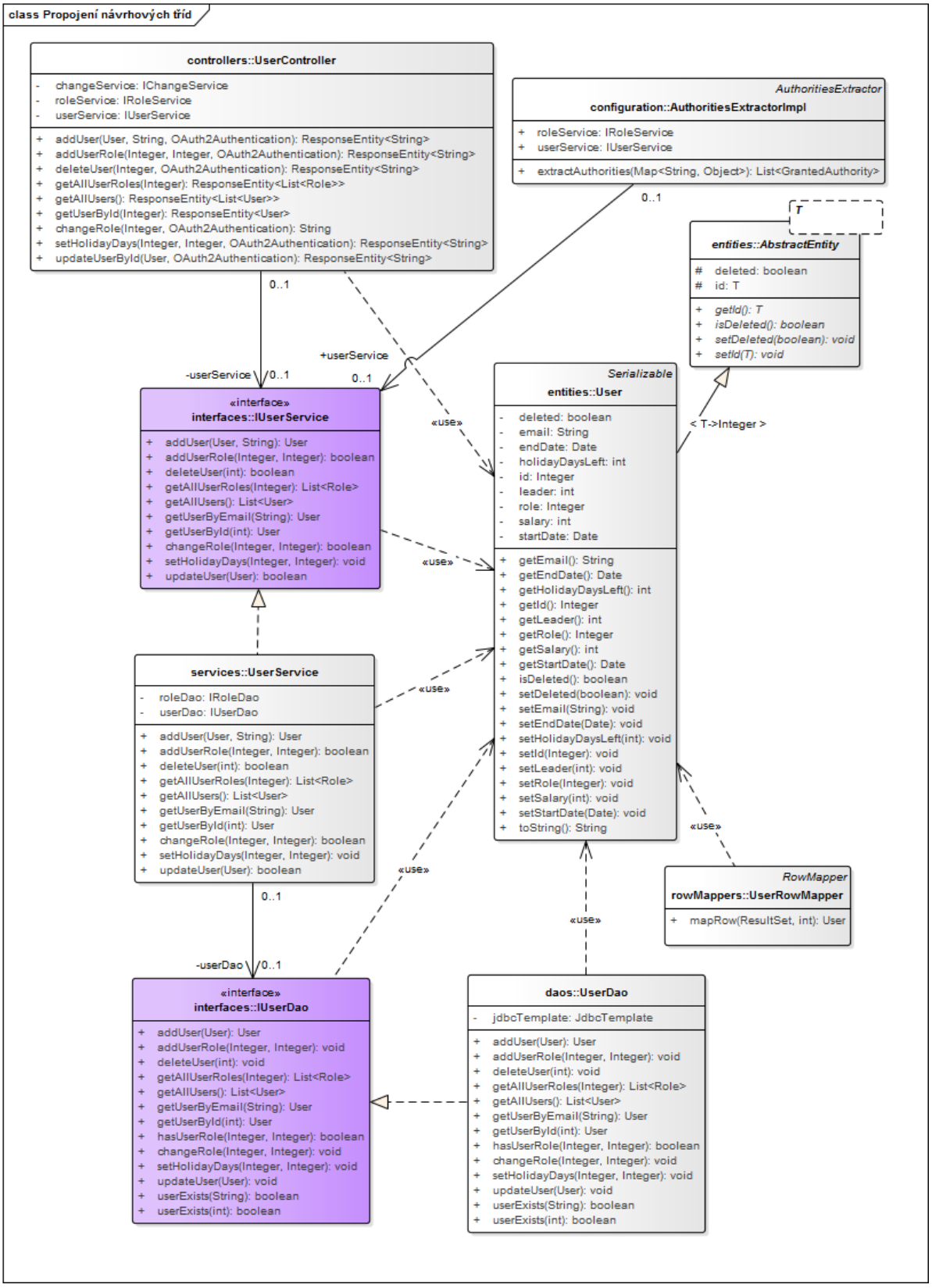
PŘÍLOHY

PŘÍLOHA A – SEKVENČNÍ DIAGRAM

Sekvenční diagram znázorňující vytvoření nové dovolené.



PŘÍLOHA B – DIAGRAM PROPOJENÍ NÁVRHOVÝCH TŘÍD



PŘÍLOHA C – LOGICKÝ MODEL