České vysoké učení technické v Praze
Fakulta elektrotechnická

Softwarové inženýrství a technologie

Kolaborační modul pro Freeplane

Collaboration plugin for Freeplane

BAKALÁŘSKÁ PRÁCE

Vypracovala: Elina Smirnova
Vedoucí práce: Mgr. Miroslav Blaško, Ph.D.
Rok: 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Smirnova** Jméno: **Elina** Osobní číslo: **483572**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Kolaborační modul pro Freeplane

Název bakalářské práce anglicky:

Collaboration plugin for Freeplane

Pokyny pro vypracování:

Freeplane je desktopová aplikace pro tvorbu myšlenkových map [1]. Cílem práce je rozšíření aplikace o podporu kolaborace nad myšlenkovými mapami. Rozšíření bude realizováno pomocí pluginu do aplikace Freeplane a samostatného serveru se kterým bude plugin komunikovat pomocí REST API. Analýza řešení by měla zahrnout analýzu provedenú Freeplane komunitou [2] i řešení založené na Sémantických technologiích [3,4].
Instrukce:
1) přezkoumente a popište existující nástroje pro tvorbu myšlenkových map s podporou kolaborace
2) definujte požadavky pro kolaboraci a uživatelské scénáře pro práci s nástrojem
3) navrhněte a implementujte rozšíření aplikace
4) otestujte použitelnost aplikace na definovaných scénářích minimálně na 3 uživatelích
5) porovnejte implementované řešení s existujícími nástroji

Seznam doporučené literatury:

[1] Free mind mapping and knowledge management software. (https://www.freeplane.org)
[2] Collaboration server for Freeplane. (https://sourceforge.net/p/freeplane/wiki/Collaboration%20server%20for%20Freeplane).
[3] Křemen, P., Mička, P., Blaško, M., Šmíd, M. (2012, September). Ontology-driven mindmapping. In Proceedings of the 8th International Conference on Semantic Systems (pp. 125-132).
[4] Salai, Dominik. "Myšlenkové mapy s použitím sémantických technologií." (2014). (https://dspace.cvut.cz/handle/10467/23352)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. Miroslav Blaško, Ph.D., skupina znalostních softwarových systémů

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

Datum zadání bakalářské práce: **11.02.2021** Termín odevzdání bakalářské práce: **04.01.2022**

Platnost zadání bakalářské práce: **30.09.2022**

Mgr. Miroslav Blaško, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studentky

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracovala samostatně a použila jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Elina Smirnova

Poděkování

Chtěla bych poděkovat vedoucímu své bakalářské práce Mgr. Miroslavu Blaškovi, Ph.D. za vedení mé práce, za jeho čas, cenné poznámky a připomínky.

Elina Smirnova

Název práce:
Kolaborační modul pro Freeplane

Autor: Elina Smirnova

Studijní program: Softwarové inženýrství a technologie
Druh práce: Bakalářská práce

Vedoucí práce: Mgr. Miroslav Blaško, Ph.D.
České vysoké učení technické v Praze,
Fakulta elektrotechnická,
Skupina znalostních softwarových systémů

Abstrakt: Tato bakalářská práce se zabývá vývojem kolaboračního modulu do aplikace Freeplane, což je multiplatformní open-source nástroj pro tvorbu myšlenkových map napsaný v Javě. Text provází čtenáře od popisu aplikace Freeplane přes analýzu existujících nástrojů až po návrh vlastního řešení. Výstupem je funkční kolaborační modul, který se skládá z pluginu integrovaného do desktopové aplikace Freeplane, samostatného serveru a uživatelského rozhraní vytvořeného pomocí JavaScriptové knihovny React. Serverová část je implementována v jazyce Java za použití frameworku Spring Boot.

Klíčová slova: kolaborace, module, Freeplane, Java,
myšlenková mapa, myšlenkové mapování,
Spring, Spring Boot, React

Title:
Collaboration plugin for Freeplane

Author: Elina Smirnova

Abstract: This bachelor thesis addresses analysis, design and implementation of the collaboration module for the Freeplane application, which is a multiplatform open-source tool written in Java for mind mapping. The text leads a reader from description of the Freeplane application through analysis of existing tools to the own design solution. The output of the work is the functional collaboration plugin, which consists of the plugin integrated to the desktop Freeplane application, separated server and user interface developed with the use of the JavaScript library React. The server side of the system is based on Java using the Spring Boot framework.

Key words: collaboration, plugin, Freeplane, Java,
mind map, mind mapping, Spring,
Spring Boot, React

Obsah

Seznam použitých zkratek

API Application Programming Interface

DOM Document Object Model

EPL Eclipse Public License

GNU GPL GNU General Public License

GNU LGPL GNU Lesser General Public License

HTML HyperText Markup Language

HTTP HyperText Transfer Protokol

JDK Java Development Kit

JPA Java Persistance API

JPEG Joint Photographic Experts Group

JSON JavaScript Object Notation

OPML Outline Processor Markup Language

PDF Portable Document Format

PNG Portable Network Graphics

REST Representational State Transfer

STOMP Simple (nebo Streaming) Text Oriented Message Protocol

SVG Scalable Vector Graphics

TCP Transmission Control Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

XHR XMLHttpRequest

XML eXtensible Markup Language

Seznam obrázků

Úvod

Myšlenkové mapy jsou vizualizačním nástrojem sloužícím ke sběru, organizaci a vizualizaci myšlenek, nápadů či dalších informací. Myšlenkové mapy lze tvořit nejen ručně pomocí tužky a papíru, ale i na počítači. Aktuálně je k dispozici mnoho aplikací umožňujících tvorbu myšlenkových map a jejich další zpracování. Mezi takové aplikace patří Freeplane. Jedná se o multiplatformní opensource nástroj licencovaný pod GNU GPL[1]. Freeplane byl odvozen z programu FreeMind a je taktéž napsaný v Javě. Freeplane má velké množství funkcí, avšak není možné tvořit myšlenkovou mapu spolu s jinými uživateli (tzv. kolaborace). Mnoho uživatelů této aplikace se v diskusním fóru obrací na tým vývojářů s požadavkem vytvořit možnost spolupráce na jedné mapě. V roce 2017 tak z toho vznikla samostatná wiki stránka, kde vedoucí vývojáři zveřejnili námět architektury kolaboračního modulu pro Freeplane, ale vývoj na tomto modulu byl pozastaven z kapacitních důvodů. Tak vznikla myšlenka obohatit tento nástroj o kolaborační plugin.

Cílem této bakalářské práce je zanalyzovat existující nástroje pro tvorbu myšlenkových map s podporou kolaborace, definovat požadavky pro kolaboraci a uživatelské scénáře pro práci s nástrojem, navrhnout vlastní řešení a naimplementovat toto rozšíření do aplikace Freeplane.

V kapitole 1 se věnuji myšlenkovým mapám obecně, definicím pojmů a uvedením do relevantních technologií, na které budu odkazovat v bakalářské práci. Dále v kapitole 2 popíši aplikaci Freeplane včetně původního návrhu kolaboračního serveru. V další kapitole se zabývám analýzou existujících nástrojů s podporou kolaborace. V kapitole 4 přistoupím k samotnému návrhu kolaboračního modulu, kde na začátku popíši funkční i nefunkční požadavky na aplikaci, stejně jako jednotlivé případy užití. Dále v této kapitole pokračuji navržením architektury a analýzou alternativního řešení založeného na sémantických technologiích. V kapitole 5 popíši detailně implementaci samotnou. Na závěr v kapitole 6 rozebírám způsoby testování kolaboračního modulu a porovnávám implementované řešení s existujícími nástroji. Závěr věnuji výsledkům a vyhodnocení celé práce.

[1] GNU General Public License

Kapitola 1

Uvedení do tématu

V úvodu této kapitoly popíši myšlenkové mapy a definuji základní pojmy, které budou použity v rámci bakalářské práce. Následně popíši relevantní technologie pro komunikaci se serverem, které se budou vyskytovat v dalších kapitolách.

1.1 Myšlenkové mapy

Myšlenkové mapy přivedl na svět v 60. letech 20. století Tony Buzan. Definice, kterou uvádějí v knize [1] Tony Buzan a Barry Buzan, zní takto: "Myšlenková mapa je vizuální nástroj pro holistické, tedy celistvé myšlení, které podporuje všechny funkce mozku – především paměť, kreativitu, učení a veškeré přemýšlení."

1.2 Definice pojmů

Komplexní terminologie v oblasti myšlenkových map není pevně zavedená. Pro další orientaci je nutné definovat základní pojmy, na které se budu v práci dále odkazovat.

Pojmy související s obecnými prvky mapy, které byly převzaté z [2]:

- *uzel* (node) – jedna myšlenka nebo téma v rámci myšlenkové mapy;
- *větev* (branch) – soubor uzlů a jejich vzájemných vztahů; začíná v libovolném uzlu myšlenkové mapy (tj. v „počátečním uzlu větve") a zahrnuje celé rozvětvení pod ním (všechny uzly a jejich vztahy), „hlavní větev" začíná v uzlu o jednu úroveň pod kořenovým uzlem;
- *úroveň* (level) – stupeň zanoření vzhledem ke kořenovému uzlu;
- *spojovací linie* (line, linie, vztah) – vztah mezi dvěma uzly, který vyjadřuje buď hierarchii v rámci myšlenkové mapy, nebo vzájemnou souvislost dvou uzlů, které spolu nesousedí v rámci hierarchie (v tomto případě také „zvláštní vztah");
- *kořenový uzel* (root, central node, centrální uzel) – centrální myšlenka mapy, nemá rodičovský uzel;
- *samostatný uzel* (floating node, plovoucí uzel) – uzel, který nemá rodičovský uzel, ale zároveň není centrální myšlenkou mapy.

Pojmy relativní vzhledem k vybranému uzlu, které byly převzaté z [2]:

- *aktuální uzel* – vybraný uzel
- *rodičovský uzel* (parent, rodič, nadřazený uzel) – uzel o jednu úroveň nad aktuálním uzlem;
- *podřízený uzel* (child, potomek, přímý podřízený uzel) – uzel o jednu úroveň pod aktuálním uzlem;
- *sourozenecký uzel* (sibling) – uzel se stejným rodičem;
- *podřízená větev* (child branch) - větev s počátkem v aktuálním uzlu – tj. soubor aktuálního uzlu – všech jeho potomků a jejich podřízených větví (včetně vzájemných vztahů).

1.3 Relevantní technologie

1.3.1 AJAX

Zkratka AJAX znamená asynchronní JavaScript a XML[1]. Ajax umožňuje asynchronně měnit obsah stránek bez nutnosti znovunačítání celého HTML[2] kódu. Základním stavebním kamenem technologie AJAX je *XMLHttpRequest* (XHR). XHR je API[3], využívá skriptovací jazyky k přenosu XML dat mezi webovým prohlížečem a webovým serverem pomocí HTTP[4] požadavků. Spuštěním události v prohlížeči (například kliknutí na tlačítko) je JavaScriptem vytvořen objekt XHR, který následně odešle požadavek na webový server. Tento požadavek bude zpracován serverem a vrátí zpět odpověď, ta je poté předána opět JavaScriptu v prohlížeči a ten provede požadovanou akci pro webovou stránku [3].

1.3.2 WebSocket

WebSocket je protokol, který umožňuje obousměrnou komunikaci mezi klientem a vzdáleným serverem prostřednictvím jednoho TCP[5] spojení. Protokol WebSocket byl primárně vytvořený pro implementaci ve webových prohlížečích a webových serverech. V současné době protokol může být implementovaný mezi jakoukoli klientskou a serverovou aplikací. V tradičním modelu požadavek-odpověď použitém v protokolu HTTP klient odesílá požadavky na server a server mu odpovídá. Výměna zpráv je vždy iniciována klientem. Server nemůže odeslat žádná data, aniž by o tom klient nejprve požádal (jako v AJAX). WebSocket řeší tato omezení poskytnutím plně duplixního komunikačního kanálu mezi klientem a serverem. Ve WebSocket aplikaci má server koncový bod ve formě adresy URI[6], který klient využívá pro připojení k serveru. Existuje trvalé spojení mezi klientem a serverem a dvě strany mohou začít odesílat data kdykoli. Klienti se obvykle připojují pouze k jednomu serveru a server přijímá požadavky od více klientů. Protokol se skládá ze dvou částí:

[1] eXtensible Markup Language
[2] HyperText Markup Language
[3] Application Programming Interface
[4] HyperText Transfer Protokol
[5] Transmission Control Protocol
[6] Uniform Resource Identifier

„handshake“ a „data transfer“. Klient zahájí „handshake“ odesláním požadavku na koncový bod pomocí URI [4].

Kapitola 2

Aplikace Freeplane

2.1 Popis

Freeplane je multiplatformní open-source nástroj licencovaný pod GNU GPL. Freeplane byl odvozen z programu FreeMind a je taktéž napsaný v Javě. Ačkoli má Freeplane velké množství funkcí, není možné tvořit myšlenkovou mapu spolu s jinými uživateli.

2.2 Použité technologie

V projektu se používá JDK[1] 1.8. Grafické uživatelské rozhraní bylo vytvořeno pomocí knihovny Swing. K buildování projektu se používá Gradle.

2.3 Původní návrh kolaboračního serveru

V rámci analýzy problému jsem prozkoumala stránky Freeplane. Nalezla jsem několik zdrojů popisujících budoucí stav kolaboračního serveru.

Podle diskuze nad designem na této stránce[2] by kolaborační server měl mít architekturu klient-server. Každá instance Freeplane může vystupovat v roli serveru, což umožňuje dvěma a víc instancím aplikace připojit se prostřednictvím sítě a spolupracovat na jedné mapě. První instance, která nastartuje sdílení mapy, by vystupovala jako server a ostatní jako klienti. Každá desktopová instance Freeplane by vždy měla mít konzistentní obraz mapy, tzn. po odpojení na mapě lze pracovat samostatně. Vývojáři aplikace počítají s tím, že relevantní aplikace bude mít dedikovaný server, který by obsluhoval všechny mind mapy uživatelů. Všechny změny první instance Freemind, které se začnou sdílet, či serveru by měly být návratné díky „undo“ mechanismu. Tento mechanismus třídí uživatelské interakce do menších transakcí. Undo transakce jsou přenášeny do serveru a následně server distribuuje každou transakci ke všem připojeným klientům. Při připojení klient synchronizuje svůj čas s časem serveru. Transakce jsou vždy doplněny timestampem. Server aplikuje jednotlivé transakce v pořadí určeném timestampem, nikoli podle pořadí příchodu transakce. Pokud by dvě nebo více transakcí byly v konfliktu, tak se aplikuje první transakce, ostatní transakce budou odvolány.

[1] Java Development Kit

[2] dostupné z `https://www.freeplane.org/wiki/index.php/Talk:Collaboration_design` k 30.12.2021

Uzamčení nodů či větví mapy může být součástí protokolu, což zamezí konfliktům.

Kontent mapy může být distribuován jak v textovém formátu, tak i binárním formátu.

Protože všechny informace ze stránek Freeplane, které jsem nalezla, byly zroku 2012–2013, rozhodla jsem se kontaktovat hlavního vývojáře Freeplane Dimitryho Polivaeva a zjistit, v jakém je momentálně stavu kolaborační plugin a jestli Dimitry plánuje ve vývoji pokračovat. Na základě mé diskuze s Dimitrym Povivaevem jsem se dozvěděla několik pro tuto práci užitečných informací. Nejdřív popíši původní návrhserveru a potom důvod, proč byla práce na vývoji serveru zastavena.

Skrz server by bylo možné synchronizovat myšlenkové mapy a mít přístup k mapám z různých míst. Hlavní myšlenkou bylo poskytnout uživatelům možnost pracovat paralelně a současně. Výše zmíněný „undo“ mechanismus by vyřízoval případné konflikty ve verzích na serveru. Probíhala by výměna příkazů mezi klientem a serverem v souborech JSON[3]. V JSON souborech se přenáší mapa ve formátu XML. Podle Dimitryho je XML serializace a deserializace je naimplementována. Co se týče updatu mapy a následného odeslání těchto změn na server, tak bylo navrženo, že se budou posílat jenom ty části, kterých se dotkly změny (tzn. neposílá se celá myšlenková mapa). Pak by jednotlivé části mapy byly uloženy do databáze a daly by se spolu kombinovat a vytahovat z databáze. Také se plánovalo napojit na server budoucí mobilní aplikace.

Vývoj kolaboračního serveru byl zastaven z několika důvodů:

- nedostatek kapacit a času;
- ztráta motivace;
- neochota udržovat a administrovat server;
- nedostatek financí na provoz.

2.4 Varianty kolaborace

V této kapitole jsem rozebrala případné varianty kolaborace ve Freeplane.

2.4.1 Kolaborace bez žádné implementace

Na tuto možnost kolaborace bez jakéhokoli zásahu do aplikace mě nasměroval Dimitry Polivaev. Tato myšlenka spočívá v tom, že 2 lidi mohou spolu kolaborovat na mapě prostřednictvím cizí aplikace umožňující zpřístupnění vzdálené plochy. Jedenčlověk si pořídí aplikaci TeamViewer a vzdáleně se připojí ke vzdálené ploše počítače druhého člověka. Avšak tento způsob kolaborace způsobuje problémy. Protože se změny provádí z jednoho počítače, není možnost pracovat na mapě paralelně. Jeden člověk bude muset vyčkat na druhého a pak provést své změny. Také není vyloučeno, že tento přenos může být pomalý.

[3]JavaScript Object Notation

2.4.2 Využití a integrování existujícího řešení od Mindmeister

Z kapitoly 3.1 při analýze aplikace Mindmeister vyplynula ještě jedna možnost integrace hotového řešení pro kolaborační modul. Jedná se o využití existujícího REST[4] rozhraní aplikace Mindmeister a provolání již naimplementovaných endpointů z desktopové aplikace Freeplane. Tudíž odpadá nutnost implementace vlastního kolaboračního serveru. Na druhou stranu, je velice vysoké riziko, že se to hotové řešení nebude hodit pro účely kolaboračního modulu pro Freeplane a zbytečně bude ztracen čas.

2.4.3 Vlastní řešení

Ani jedna z výše rozepsaných variant se nebude hodit pro Freeplane. Popis vlastního řešení bude následovat v kapitole 4.

[4]Representational State Transfer

Kapitola 3

Analýza existujících nástrojů

Užitečnost myšlenkových map lze prokázat množstvím nástrojů dostupných na internetu. Kupříkladu XMind, Compendium, WiseMapping, FreeMind, MindMeister, Semantik, Coggle, ClickUp, Ayoa, Miro, MindMaster, Mindmup.

V rámci analýzy jsem realizovala detailnější výzkum některých z výše uvedených nástrojů. Svůj průzkum existujících aplikací jsem provedla v rámci semestrální práce v rozmezí 29.11.2020 – 31.12.2020 pomocí vyhledávání na google.com. Jelikož mě zajímaly open-source nástroje, které podporují kolaboraci, do vyhledávače jsem zadala klíčová slovní spojeníjako „mind mapping software for collaboration“, „open source software for mind mapping“. Také jsem se opírala o aktuálnost aplikací při výběru nástrojů pro detailní analýzu. K průzkumu jsem si následně vybrala následující nástroje uvedené v tomto článku [5]:

- Mindmeister;
- WiseMapping;
- XMind;
- FreeMind.

Mindmeister, WiseMapping a XMind se neustále vyvíjí a zdokonalují se. Do analýzy jsem musela zařadit také aplikaci FreeMind, jelikož FreeMind je předchůdcem Freeplane.

Při analýze mnou vybraných nástrojů byl kladen důraz na další body, které budou detailně rozepsány v této kapitole:

- mají-li import/export map z Freeplane;
- jaký je způsob ukládání map;
- jak vypadá operace ukládání map v REST API;
- jakým způsobem se kolaboruje na myšlenkové mapě;
- podporuje-li API různé verze myšlenkové mapy;

Jako doplnění k první otázce je nutné také uvést, že většina nástrojů podporuje import map z externích nástrojů. Freeplane je následníkem aplikace FreeMind a jednou z možností exportu map ve Freeplane je export ve formátu „.mm“. Tudíž můžeme prohlásit, že nástroje, které podporují možnost importu z FreeMind, podporují i import z Freeplane.

3.1 Mindmeister

Jedná se o webovou aplikaci, kterou lze použít pro brainstorming, plánování projektu a psaní poznámek. Je kompatibilní s operačními systémy Windows, Mac OS a Linux. Všechny mapy jsou bezpečně uložené v cloudu. Velkou výhodou tohoto nástroje je možnost upravovat mapu více uživatelům současně. Pro sdílení mapy stačí odeslat pozvánku ke kolaboraci prostřednictvím e-mailu nebo poslat odkaz na mapu. Mindmeister má i mobilní aplikace umožňující redigovat a synchronizovat myšlenkové mapy přímo z mobilu [6]. Mimo jiné Mindmeister podporuje režim offline, což umožňuje uživateli vytvářet myšlenkové mapy i bez přístupu k internetu. Mezi plusy aplikace také patří mnohofunkčnost a uživatelsky přívětivý interface. Mindmeister dovoluje vytváření map za co nejméně času. Toto je zásluhou jednoduchého a poměrně intuitivního ovládání. Intelektuální systém tvoření uzlů pomáhá uživateli s automatickým rozmístěním nových uzlů a případným posunem existujících uzlů. Nové uzly se například přidávají do mapy symetricky.

Mindmeister podporuje import a export map z takových nástrojů jako MindManager, XMind, FreeMind (tím pádem i Freeplane) a import map v podobě plain textu.

Základní neplacená verze umožňuje uživateli vytvořit maximálně 3 myšlenkové mapy a samozřejmě je sdílet a spolupracovat s ostatními. Pokud by to uživateli nestačilo a chtěl by pracovat s Mindmeisterem na vyšší úrovni, Mindmeister nabízí několik placených balíčků.

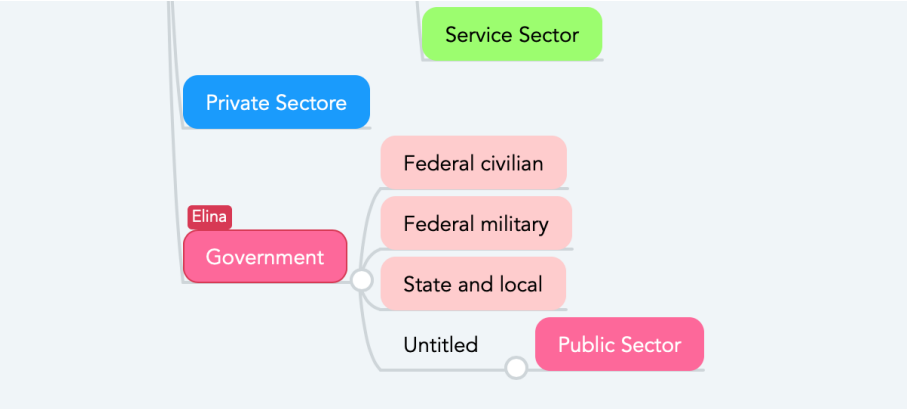
Kolaborace probíhá následujícím způsobem: po otevření odkazu se zobrazí plocha s mapou a tool barem, dole se objeví číslo, kolik uživatelů kolaboruje na mapěv reálném čase. Po rozkliknutí toho čísla je vidět celý seznam uživatelů, kteří spolupracují na mapě. Uživateli, který nasdílel mapu (dále souhrnně jen „administrátor“), se zobrazí hláška, že byl aktivován „brainstorming mode“ (tzv. režim kolaborace). Jako administrátor máte možnost přiřadit jednotlivým uživatelům práva, konkrétně právo prohlížení nebo právo úpravy na úrovni celé mind mapy. V reálném čase vidíte úpravy ostatních. Uzel, na kterém momentálně probíhají úpravy, bude označen jménem uživatele, který změny provádí (viz obrázek 3.1). Tento uzel se nezamykáa dá se s ním případně hýbat.

Změny provedené jinými uživateli se replikují s nevýznamným zpožděním. Pokud spolupracujete s jiným uživatelem na stejném uzlu, aplikuje se poslední verze tohoto uzlu. Tedy Mindmeister nepodporuje případné řešení konfliktu dvou verzí stejného uzlu. Najetím myši na objekt je možné zjistit čas poslední úpravy a jméno autora té změny. Mindmeister podporuje i historie všech úprav jednotlivých objektů. Máte možnost dohledat detaily úprav jednotlivých uzlů, datum a čas provedení úpravy, autora úpravy. Ale není možné rekonstruovat (revert) minulé úpravy.

Nahlédnutím do konzole v prohlížeči jsem zjistila, že webová aplikace využívá technologii XHR, která je detailně popsána v kapitole 1.3.1. Dohledáním requestů v konzoli je vidět, že se změny neustále načítají ze serveru pomocí POST metody „poll“. Při provádění jakýchkoliv změn (posun uzlu, přejmenování uzlu apod.) se volá metoda „do“. Metoda „do“ vyžaduje takové parametry jako „root“, „rev“, „request_id“, „protocolVersion"a „data“ - pole provedených změn. Reprezentativní příklad dat je možné nalézt v příloze B.1.

Pro vývojáře Mindmeister poskytuje otevřené API [7]. API mají dostupné ve dvou verzích, aktuálně se vyvíjí druhá verze API. Mindmeister API v2 umožňuje aplikacím třetích strán načítat data pro uživatele. Konkrétně jde o HTTP REST

API a nabízí zdroje uživatelů, map a souborů. Autorizace probíhá za pomocí OAuth 2.0



Obrázek 3.1: Ukázka kolaborace na uzlu v Mindmeister [vytvořeno autorem]

3.2 WiseMapping

WiseMapping je webový mapovací nástroj, který využívá HTML5 a SVG[1]. WiseMapping fungují na základě open source přístupu. Jedná se o jednoduchý nástroj sloužící k rychlému třídění myšlenek. WiseMapping nabízí pouze omezené kreativní možnosti, téměř nelze měnit barvy ani upravovat rámečky jednotlivých větví myšlenkových map. K mapám lze připojovat obrázky, konvertovat mezi různými formáty ke stažení. WiseMapping je dostupný v bezplatné i placené verzi [8]. WiseMapping nabízí možnost sdílet mapy a pozvat ostatní k prohlížení či kolaboraci. Nástroj dokáže naimportovat mapy z FreeMind 1.1, exportovat mapy ve formátu PDF[2], SVG, JPEG[3] a FreeMind.

Chcete-li pozvat lidi ke kolaboraci, vyberte možnost „Share“ a vyplňte e-mail. Máte možnost vybrat, jestli uživatel bude moci pouze prohlížet mapu, anebo ji i upravovat. Kolaborovat může jen přihlášený uživatel, tudíž pokud uživatel, který dostal pozvánku ke kolaboraci, nemá účet, musí se nejprve zaregistrovat a následně se přihlásit.

Upravovat mapu v reálném čase může jen jeden uživatel. Tedy po otevření odkazu z pozvánky, ostatní uživatelé dostanou notifikaci se jménem uživatele provádějícím úpravy a budou mít mapu ve stavu „read-only“. WiseMapping nepodporuje možnost sledování úprav mapy v reálném čase. Mapa se odemyká poté, co uživatel ji explicitně uloží tlačítkem „Save“ a odejde z té mapy. Tato synchronizace potrvá nějaký čas.

WiseMapping podporuje verzování map. Po rozkliknutí „History“ se uživateli objeví seznam verzí, konkrétně seznam uživatelů, kteří změny prováděli, a čas úpravy. Každou verzi je možné prohlédnout, anebo vrátit zpět.

Co se týče technické části, aplikace je postavená na REST API. Jelikož WiseMapping je open-source aplikace [9], měla jsem možnost detailně prozkoumat, jakým způsobem se ukládají mapy.

[1] Scalable Vector Graphics
[2] Portable Document Format
[3] Joint Photographic Experts Group

Jakýkoliv update mapy probíhá tak, že se nejdřív načte mapa dle id mapy a uživatele, který tyto změny prováděl. Pak se načtou properties. Následujícím krokem se zkontroluje, jestli mapa může být updatována, respektive jestli je uzamčena. Pak se updatnou properties pro kolaboraci. Následně se zkontroluje kontent mapy a celá mapa se uloží na server. Po nahlédnutí do databáze jsem zjistila, že se do relační databáze PostgreSQL mapa ukládá ve formátu BYTEA[4]. Do tabulky zvlášť se ukládá historie úprav.

3.3 XMind

XMind je další aplikace pro vytváření myšlenkových map k běžnému a profesionálnímu použití. Jedná se o open-source projekt, který je licencován EPL[5] a GNU LGPL[6]. Tento mapovací nástroj dokáže zvizualizovat myšlenky různými kreativními způsoby. Podporuje klasické myšlenkové mapy, diagram rybí kosti, stromové diagramy, „organization chart“, logické grafy a dokonce i tabulky. Často se používá pro management znalostí, zápisy z jednání, správu úkolů a GTD [7]. XMind je dostupný buď jako webová aplikace online, anebo v desktopové a mobilní verzi. [12].

Aplikace má poměrně dostačující exportní možnosti. Aplikace dokáže exportovat mapu v různých formátech – PDF, PNG[8], SVG, TextBundle, Markdown, Word, Excel, OPML[9] a také existuje možnost sdílení mapy do různých sociálních sítí.

XMind nepodporuje kolaboraci, ale umožňuje sdílení mapy v sociálních sítích. Také je možné získat odkaz na mapu, a to jak veřejný, tak i privátní.

Co se týče API aplikace, existuje několik metod pro ukládání myšlenkové mapy, které jsou reprezentovány v příloze s komentáři. Vždy se ukládá celý soubor (tzv. workbook) a je průběžně ukládán do dočasné paměti.

Workbook je ve skutečnosti standardní archiv ZIP [10], který se skládá z několika XML souborů, příloh ve formě obrázků a náhledů mapy ve formátu PNG a JPEG. Zaměřím se na soubor content.xml z archivu ZIP, který obsahuje hierarchicky uspořádaná data o mapě. Hlavními elementy souboru je „sheet“ a „topic“ (viz. obrázek 3.2).

„Xmap-content“ je kořenový element tohoto souboru. Jeho vnořeným elementem je „sheet“. Element „sheet“ reprezentuje myšlenkovou mapu s kořenovým „topic“, který představuje téma v myšlenkové mapě. Element „topic“ může mít potomky, ale musí mít právě jeden rodičovský element. Do „topic“ se vkládají další elementy jako „title“ (název tématu), „image“ (obrázek tématu), „notes“ (poznámky k tématu), „position“ (pozice tématu), „children“ (kontejner s podtématy) atd.

[4]BYTEA je datový typ, který umožňuje ukládání binárních řetězců v PostgreSQL. [10]

[5]Eclipse Public License

[6]GNU Lesser General Public License

[7]GTD neboli „Getting Things Done“ je metoda v oblasti osobní produktivity, která předefinuje, jak přistupujete ke svému životu a práci. [11]

[8]Portable Network Graphics

[9]Outline Processor Markup Language

[10]ZIP je souborový formát pro kompresi a archivaci dat.

```
<xmap-content xmlns="urn:xmind:xmap:xmlns:content:2.0" version="2.0">
    <sheet id="sheet1">
        <topic id="topic1">
            <title>Topic 1</title>
            <children>
                <topics type="attached">
                    <topic id="topic2">
                        <title>Topic 2</title>
                    </topic>
                </topics>
            </children>
        </topic>
        <relationships>
            <relationship id="relationship1" end1="topic1" end2="topic2">
            </relationship>
        </relationships>
    </sheet>
</xmap-content>
```

Obrázek 3.2: Ukázka souboru content.xml

3.4 FreeMind

FreeMind je open-source mapovací nástroj, licencovaný pod *GNU General Public License* (GNU GPL) a napsaný v Javě. Díky jednoduchému a intuitivnímu rozhraní, je aplikace snadno použitelná. FreeMind je předchůdcem aplikace Freeplane. Freemind má zajímavou sadu užitečných funkcí a vlastností.

Od roku 2016 je vývoj aplikace pozastaven. Kolaborace funguje na základě Jabber server collaboration source, jak je uvedeno v postupu komunikace (postup komunikace). Počítače by se měly nacházet na stejném hostu. Host, port, jméno uživatele a heslo je možné upravit v preferencích aplikace. Pro sdílení mapy je potřeba vyplnit heslo a port. Pro připojení k mapě je nutné vyplnit heslo, host a port. Vyzkoušet kolaboraci ve FreeMind se mi nepovedlo, selhalo to na připojení ke druhému počítači.

3.5 Shrnutí

Do sumarizační tabulky jsem zahrnula aplikace, které podporují kolaboraci. Tato analýza napomůže při dalším rozhodování a bude se z ní vycházet při návrhu kolaboračního pluginu.

	Mindmeister	WiseMapping	FreeMind
Obecná kritéria			
Operační systém			
Windows	✓	✓	✓
Mac OS X	✓	✓	✓
Linux	✓	✓	✓
Úložiště			
Cloudové	✓	✓	
Lokální			✓
Autorizace			
Ano	✓	✓	
Ne			✓
Undo/Redo			
Ano	✓	✓	✓
Ne			
Kritéria pro kolaboraci			
Sledování úprav real-time			
Ano	✓		
Ne		✓	
Přidělení práv na mapu			
Ano	✓	✓	
Ne			✓
Způsoby sdílení mapy			
Odkaz	✓	✓	
Odeslání pozvánky e-mailem	✓	✓	
Připojení k mapě přes port			✓
Paralelní úprava mapy			
Ano	✓		
Ne			
Uzamčení mapy na dobu úprav		✓	
Uzamčení části mapy			
Verzování			
Ano Ne	✓	✓	
Rollback změn			
Ano		✓	
Ne	✓		
Rezoluce konfliktů			
Ano			
Ne	✓	✓	

Tabulka 3.1: Sumarizační tabulka nástrojů, které podporují kolaboraci

Kapitola 4

Návrh řešení

Ve fázi návrhu řešení je nutné specifikovat, jaké požadavky musí budoucí kolaborační plugin splňovat a jaké vlastnosti má mít. Dále je potřeba navrhnout architekturu kolaboračního modulu.

4.1 Analýza požadavků

V této kapitole jsou popsány funkční a nefunkční požadavky na kolaborační modul. Každý z těchto požadavků je označen identifikačním číslem, má název a popis. Dále je u každého požadavku určena priorita pomocí metody MoSCoW (must have, should have, could have, won't have). Požadavky se rozpadají na vícero případů užití, které následně budou rozepsané v kapitole 4.2.

4.1.1 Funkční požadavky

FP1 Registrace

Plugin bude umožňovat uživatelům registraci.

FP1.1 Registrace e-mailem a heslem (must have)

Plugin umožní nepřihlášeným uživatelům zaregistrovat se do aplikace pomocí e-mailu a hesla. Každému registrovanému uživateli bude na serveru zřízen účet, který bude uchovávat jeho registrační údaje v databázi.

FP1.2 SSO registrace (could have)

Plugin bude umožňovat uživatelům zaregistrovat se prostřednictvím technologie SSO služeb třetích stran. Plugin nebude mít přístup k přihlašovacím údajům, a tedy heslo uživatele se nebude ukládat v databázi pluginu.

FP1.3 Potvrzení registrace (could have)

Plugin umožní odesílání e-mailů s potvrzovacím odkazem po registraci na e-mail, který byl uveden uživatalem při registraci v aplikaci.

FP2 Přihlášení

Plugin umožní uživateli přihlásit se do aplikace pro přístup ke jeho údajům uloženým na serveru, kolaboraci a ukládání map.

FP2.1 Přihlášení e-mailem a heslem (must have)

Plugin umožní zaregistrovaným uživatelům přihlásit se do aplikace vyplněním přihlašovacího formuláře, konkrétně vyplněním e-mailu a hesla.Plugin porovná e-mail a heslo zadané uživatelem s uživatelskými údaji uloženými v databázi a provede přihlášení.

FP2.2 Přihlášení pomocí SSO (could have)

Plugin bude umožňovat uživatelům přihlásit se pomocí technologie SSO, která ověřuje identitu uživatele pouze jednou a všem ostatním systémům, které jsou součástí SSO, zaručí, že je správná.

FP2.3 Automatické přihlášení (should have)

Plugin umožní automatické přihlášení (tzv. autologin) po spuštění aplikace. Pokud uživatel při prvním přihlášení do aplikace zaškrtne políčko „remember me", pak si server zapamatuje přihlašovací údaje uživatele a příště provede přihlášení automaticky.

FP3 Role uživatelů na úrovni aplikace (must have)

Plugin umožní rozlišovat role uživatelů pro přístup k aplikaci:

- Administrátor – uživatel s touto rolí má k dispozici přehled všech uživatelů s možností zakládat nové účty a mazat existující. Také má přístup k přehledu všech map, kde může k libovolné mapě přiřadit jednotlivé uživatele. Tím se vytvoří skupina uživatelů, která bude mít přístup k prohlížení či editaci mapy. Kromě přiřazení uživatelů má možnost odebrat jednotlivé uživatele z mind map nebo upravit jejich práva na úrovni mind mapy;
- User – tento typ uživatele má přístup k vlastním a s ním sdíleným mind mapám. Vlastní mind mapy může sdílet s dalšími uživateli prostřednictvím unikátního odkazu a mazat je.

FP4 Sdílení mind mapy

Plugin umožní uživatelům sdílet mind mapy pro ostatní uživatele.

FP4.1 Sdílení mind mapy odkazem (must have)

Plugin umožní sdílení mind mapy zasláním odkazu ke kolaboraci. Uživateli, který bude chtít přizvat dalšího ke kolaboraci, se vygeneruje odkaz namind mapu. Přes zaslaný odkaz se přizvaný uživatel bude moci připojit ke kolaboraci.

FP4.2 Sdílení mind mapy zasláním pozvánky (could have)

Plugin umožní sdílení mind mapy zasláním pozvánky prostřednictvím e-mailu. Uživatel, který bude chtít přizvat dalšího ke kolaboraci, vyplní jeho e-mailovou adresu. Pozvanému uživateli přijde e-mailová pozvánka s odkazem. Přes uvedený odkaz v e-mailu se bude moci připojit ke kolaboraci.

FP5 Přístup k mind mapě

FP5.1 Role uživatelů na úrovni mind mapy (must have)

Plugin umožní definovat role uživatelů pro jednotlivé mind mapy, což znamená omezení přístupu k mind mapě pomocí následujících rolí:

- Owner – ten, kdo první založil mapu a nasdílel ji pro ostatní uživatele. Jediný typ uživatele, který může smazat mind mapu. Spolupracovníci, kterým byla tato mapa nasdílena, ztratí přístup k smazané mind mapě;
- Reader – uživatel, kterému byla sdílena mapa s právem prohlížení. Nemá právo sdílet mapu dalším uživatelům;
- Editor – uživatel, kterému byla sdílena mapa s právem úprav. Nemá právo sdílet mapu dalším uživatelům.

FP5.2 Způsoby přístupů k mind mapě (must have)

Plugin umožní rozlišovat způsoby přístupů k mind mapě. Přístup k mind mapě bude mít uživatel, který splňuje alespoň jeden z následujících požadavků:

- první založil mind mapu na serveru, tudíž nabývá vlastnického práva pro další správu mind mapy. Mapa se autorovi zobrazí v seznamu vlastních mind map;
- uživatel dostal pozvánku či odkaz ke kolaboraci a připojil se ke kolaboraci. Následně se mapa zobrazí v seznamu se sdílenými mapami.

FP5.3 Přidělení práv na úpravu mind map (should have)

Plugin umožní přidělit právo prohlížení nebo právo úpravy na úrovni celé mind mapy při odeslání pozvánky nebo generování odkazu ke kolaboraci.

FP6 Kolaborace na mind mapě

Plugin umožní kolaborovat s ostatními uživateli na jedné mapě v reálném čase.

FP6.1 Uzamčení celé mind mapy během kolaborace (must have)

Plugin umožní uzamknout celou mind mapu po dobu úprav. To znamená, že další uživatelé, kteří budou chtít kolaborovat, budou informováni o tom, že nebudou moct upravovat mapu a budou tuto mapu mít k dispozici pouze v režimu čtení.

FP6.2 Uzamčení uzlů mind mapy během kolaborace (could have)

Plugin umožní zamykání uzlů po dobu úprav. To znamená, že další uživatelé, kteří budou chtít kolaborovat, budou moct upravovat mapu současně, akorát není možné kolaborovat na jednom elementu mind mapy paralelně.

FP6.3 Sledování úprav real-time (could have)

Plugin umožní sledovat úpravy ostatních uživatelů v reálném čase. Uzel, na kterém momentálně probíhají úpravy, bude označen jménem uživatele, který změny provádí.

FP6.4 Paralelní úprava mind mapy (could have)

Plugin umožní více uživatelům kolaborovat na jedné mind mapě paralelně, což znamená, že se nebudou zamykat jednotlivé části mind mapya dá se spolupracovat na jednom elementu mapy současně. Z toho plyne nutnost mít mechanismus řešení případných kolizí mezi verzemi.

FP7 Verzování mapy (could have)

Plugin umožní uchovávat historii změn mind map. Po dokončení úprav na mind mapě uživatel může uložit pojmenovanou verzi mind mapy na server. Pak se uživatel bude moct vrátit k jednotlivým verzím mind mapy a prohlédnout provedené změny.

FP8 Rollback verze mind mapy (could have)

Plugin umožní provádět operaci vrácení zpět (tzv. rollback) pro jednotlivé verze mind map z historie. Novější verze mind mapy se přepíšou na vybranou starší verzi.

FP9 Týmy (could have)

Plugin umožní uživatelům tvořit týmy pro společnou tvorbu mind map. Uživatel, který založí tým, se automaticky stane administrátorem týmu, tj. budemít nejvyšší práva.

FP10 Cloudové úložiště (could have)

Kolaborační plugin umožní uživateli ukládat mapu na své cloudové úložiště, ke kterému pak bude moct přistupovat i z jiných zařízení.

FP11 Podpora Google Drive (could have)

Plugin bude rozšířen o podporu Google Drive a umožní uživateli ukládat mapu přímo na svém Google disku.

4.1.2 Nefunkční požadavky

NFP1 Hashování hesel (must have)

Plugin bude ukládat do databáze osolené a zahashované heslo.

NFP2 Komunikace pluginu a serveru pomocí REST API (must have)

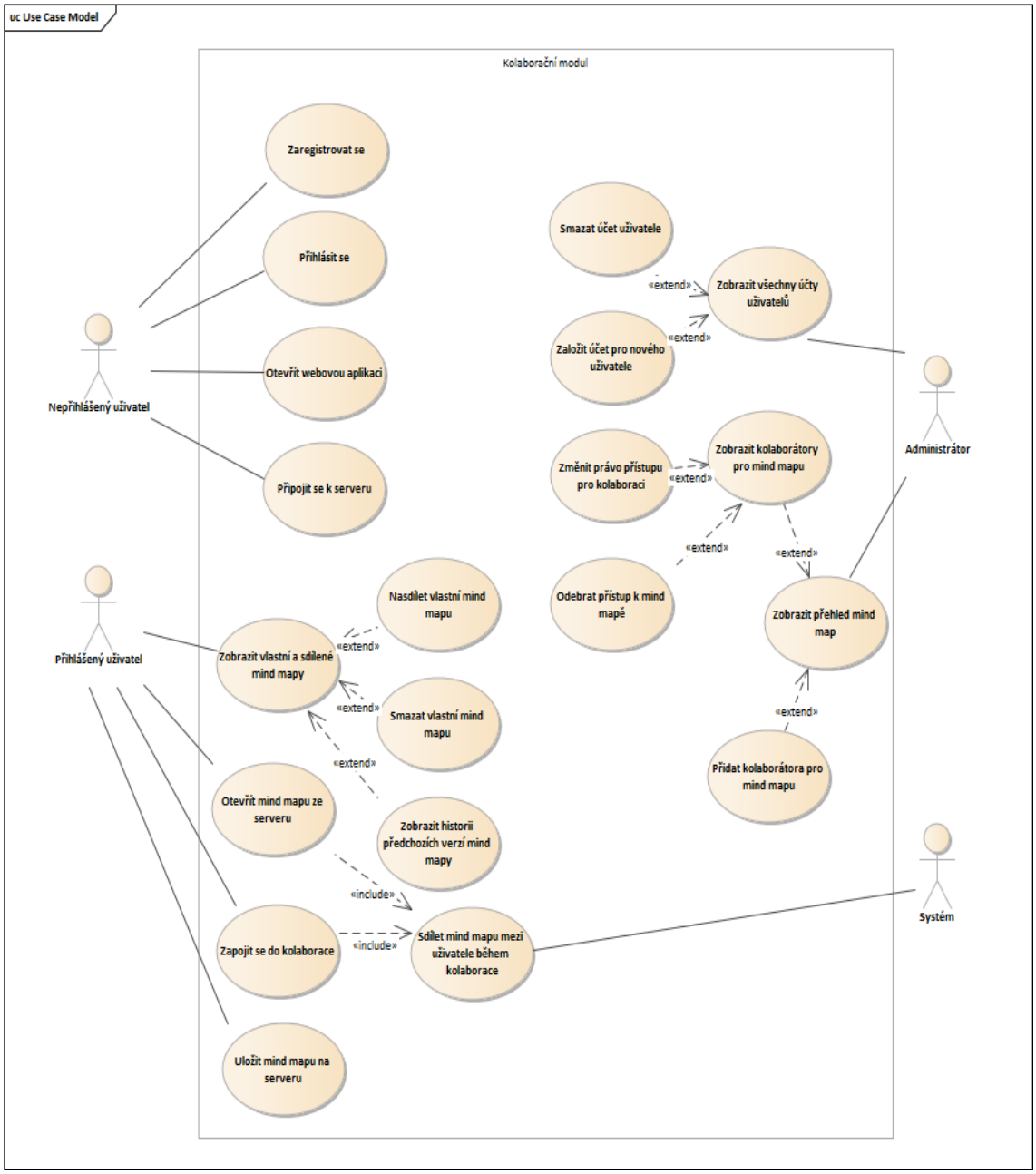
Plugin umožní komunikaci se serverem pomocí REST rozhraní.

4.2 Případy užití

Případy užití ukazují určité funkce kolaboračního pluginu, které mohou provádět konkrétní typy uživatelů. Ke každému případu užití je uveden scénář, kde jsou zachyceny jednotlivé kroky interakce uživatele pro dosažení jednoho uživatelského cíle.

4.2.1 Diagram případů užití

Diagram případů užití na obrázku 4.1 byl vypracován v programu Enterprise Architect [1].



Obrázek 4.1: Diagram případů užití [vytvořeno autorem]

[1] Enterprise Architect od společnosti Sparx Systems je vizualizační nástroj pro vytváření UML modelů, který pokrývá celý životní cyklus tvorby software.

4.2.2 Scénáře pro případy užití

- **Zaregistrovat se**

  Vstupní podmínky: uživatel má otevřenou webovou aplikaci kolaboračního modulu Freeplane.

  1. Webová aplikace zobrazí registrační formulář.
  2. Nepřihlášený uživatel vyplní svůj e-mail a heslo dvakrát.
  3. Aplikace provede validaci vyplněných polí.
  4. IF validace polí proběhla úspěšně a uživatel se shodným e-mailem neexistuje v databázi,
     THEN registrace je považována za úspěšnou a aplikace přesměruje uživatele na přihlašovací stránku,
     ELSE aplikace zobrazí chybovou hlášku.

- **Přihlásit se**

  Vstupní podmínky: uživatel má otevřenou webovou aplikaci kolaboračního modulu Freeplane.

  1. Webová aplikace zobrazí přihlašovací formulář.
  2. Nepřihlášený uživatel vyplní svůj e-mail a heslo.
  3. Aplikace provede validaci vyplněných polí.
  4. IF validace polí proběhla úspěšně a zároveň přístupové údaje byly zadány správně
     THEN přihlášení je považováno za úspěšné a aplikace přesměruje uživatele na uvítací stránku,
     ELSE aplikace zobrazí chybovou hlášku.

- **Otevřít webovou aplikaci**

  Vstupní podmínky: uživatel má otevřenou desktopovou aplikaci Freeplane.

  1. V desktopové aplikaci Freeplane nepřihlášený/přihlášený uživatel v horním menu zmáčkne záložku *Server -> Open mind maps management*.
  2. Aplikace přesměruje uživatele na stránku kolaboračního pluginu v prohlížeči.

     2a Aplikace přesměruje nepřihlášeného uživatele na uvítací stránku.

     2b Aplikace přesměruje přihlášeného uživatele na domovskou stránku.

- **Připojit se k serveru**

  Vstupní podmínky: uživatel má otevřenou desktopovou aplikaci Freeplane.

  1. V desktopové aplikaci Freeplane nepřihlášený/přihlášený uživatel v horním menu zmáčkne záložku *Server -> Connect*.
  2. Aplikace zobrazí přihlašovací okno.
  3. Uživatel vyplní e-mail a heslo.

4. IF přístupové údaje zadány správně,
   THEN připojení k serveru proběhlo úspěšně,
   ELSE aplikace zobrazí chybovou hlášku.

- **Zobrazit všechny účty uživatelů**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Administrátor zmáčkne záložku *Users* v menu.
  2. Aplikace načte seznam všech uživatelů z databáze a zobrazí je na samostatné stránce.

- **Založit účet pro nového uživatele**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce se seznamem všech uživatelských účtů.
  2. Administrátor zmačkne tlačítko *Create a new user*.
  3. Aplikace zobrazí formulář na nové stránce.
  4. Administrátor vyplní jméno, příjmení, e-mail a heslo pro nového uživatele.
  5. Aplikace provede validaci vyplněných polí.
  6. IF validace proběhla úspěšně a uživatel se shodným e-mailem neexistuje v databázi,
     THEN aplikace uloží nového uživatele do databáze,
     ELSE aplikace zobrazí chybovou hlášku.

- **Smazat účet uživatele**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce se seznamem všech uživatelských účtů.
  2. U vybraného uživatele administrátor zmáčkne ikonu koše pro smazání.
  3. Aplikace smaže uživatele z databáze a odstraní ho ze seznamu uživatelů na stránce.

- **Zobrazit přehled mind map**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Administrátor zmáčkne záložku *Mindmaps* v menu.
  2. Aplikace načte seznam všech mind map z databáze a zobrazí jejich přehled na samostatné stránce.

- **Zobrazit kolaborátory pro mind mapu**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce s přehledem mind map.

2. Administrátor rozklikne vybraný řádek s názvem mind mapy.
3. Aplikace načte kolaborátory z databáze a zobrazí jejich seznam pod řádkem s vybranou mind mapou na stránce.

- **Přidat kolaborátora pro mind mapu**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce s přehledem mind map.
  2. Administrátor zmáčkne tlačítko *Add a collaborator*.
  3. Aplikace zobrazí formulář na nové stránce.
  4. Administrátor vybere mind mapu a roli z rozbalovacího seznamu pro kolaboraci, vyplní e-mail uživatele.
  5. Aplikace provede validaci pole pro e-mail.
  6. IF validace proběhla úspěšně a v databázi již neexistuje kolaborátor se shodným e-mailem pro vybranou mind mapu,
     THEN aplikace uloží nového kolaborátora do databáze a přesměruje uživatele na stránku s přehledem všech mind map,
     ELSE aplikace zobrazí chybovou hlášku.

- **Odebrat přístup k mind mapě**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce s rozkliknutým seznamem kolaborátorů pro vybranou mind mapu.
  2. Na řádku s vybraným kolaborátorem administrátor zmáčkne ikonu koše pro smazání.
  3. Aplikace odebere uživatele ze seznamu kolaborátorů v databázi a vyvolá uživatele k obnovení stránky.
  4. Po obnovení stránky kolaborator zmizí ze seznamu.

- **Změnit právo přístupu pro kolaboraci**

  Vstupní podmínky: administrátor je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce s rozkliknutým seznamem kolaborátorů pro vybranou mind mapu.
  2. Na řádku s vybraným kolaborátorem administrátor zadá novou roli pro kolaboraci.
  3. Aplikace provede validaci vstupu.
  4. IF validace role proběhla úspěšně,
     THEN aplikace změní právo přístupu pro kolaboraci v databázi a vyvolá uživatele k obnovení stránky,
     ELSE aplikace zobrazí hlášku s povoleným seznamem rolí pro kolaboraci.

5. Po obnovení stránky kolaborátor bude mít aktualizované právo přístupu pro kolaboraci.

- **Zobrazit vlastní a sdílené mind mapy** Vstupní podmínky: uživatel je přihlášen do webové aplikace kolaboračního pluginu.

  1. Uživatel zmáčkne záložku *Mindmaps* v menu.
  2. Aplikace načte a zobrazí vlastní a sdílené s uživatelem mind mapy na samostatné stránce.

- **Nasdílet vlastní mind mapu**

  Vstupní podmínky: uživatel je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce se seznamem vlastních a sdílených mind map.
  2. Uživatel zmáčkne tlačítko Share na řádku s mind mapou ze seznamu *Your mindmaps*.
  3. Aplikace zobrazí modální okno pro generování odkazu pro kolaboraci.
  4. Uživatel vyplní e-mail a vybere role z rozbalovacího seznamu.
  5. Uživatel zmáčkne tlačítko *Generate*.
  6. Aplikace provede validaci pole pro e-mail.
  7. IF validace proběhla úspěšně,
     THEN aplikace vygeneruje a zobrazí odkaz pro kolaboraci,
     ELSE aplikace zobrazí chybovou hlášku.

- **Smazat vlastní mind mapu**

  Vstupní podmínky: uživatel je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce se seznamem vlastních a sdílených mind map.
  2. Uživatel zmáčkne *Delete* na řádku s mind mapou ze seznamu *Your mindmaps*.
  3. Aplikace smaže mind mapu ze serveru a odebere ji ze seznamu vlastních mind map.

- **Zobrazit historii předchozích verzí mind mapy**

  Vstupní podmínky: uživatel je přihlášen do webové aplikace kolaboračního pluginu.

  1. Scénář začíná na stránce se seznamem vlastních a sdílených mind map.
  2. Uživatel klikne na ikonu historie na vybraném řádku mind mapy.
  3. Aplikace načte a zobrazí seznam předchozích verzí mind mapy na samostatné stránce.

- **Otevřít mind mapu ze serveru**

  Vstupní podmínky: uživatel je připojen k vzdálenému serveru v desktopové aplikaci Freeplane.

1. V horním menu desktopové aplikace Freeplane přihlášený uživatel zmáčkne záložku *Server -> Open mind map from server*.
2. Plugin načte a zobrazí okno se seznamem vlastních mind map uživatele.
3. Uživatel kliknutím vybere mind mapu ze seznamu a zmáčkne tlačítko *Open*.
4. Plugin načte mind mapu z databáze a zobrazí ji v desktopové aplikaci.
5. «include» Sdílet mind mapu mezi uživatele během kolaborace.

- **Uložit mind mapu na serveru** Vstupní podmínky: uživatel je připojen k vzdálenému serveru v desktopové aplikaci Freeplane a již má otevřenou mind mapu.

  1. V horním menu desktopové aplikace Freeplane přihlášený uživatel zmáčkne záložku *Server -> Save mind map on server*.
  2. Plugin uloží mind mapu do databáze.

- **Zapojit se do kolaborace**

  Vstupní podmínky: uživatel je připojen k vzdálenému serveru v desktopové aplikaci Freeplane

  1. V horním menu desktopové aplikace Freeplane přihlášený uživatel zmáčkne záložku *Server -> Join*.
  2. Plugin zobrazí okno s polem pro vložení odkazu pro kolaboraci.
  3. Uživatel vyplní odkaz pro kolaboraci a zmáčkne tlačítko *Join*.
  4. Plugin provede validaci odkazu pro kolaboraci.
  5. IF validace proběhla úspěšně,
     THEN «include» Sdílet mind mapu mezi uživatele během kolaborace,
     ELSE aplikace zobrazí chybovou hlášku.

- **Sdílet mind mapu mezi uživatele během kolaborace**

  1. IF mind mapa není uzamčená,
     THEN Systém začne sdílení mind mapy pro ostatní kolaborátory,
     ELSE aplikace upozorní uživatele, že mind mapa je zamčená a není povoleno ji upravovat.

4.3 Architektura kolaboračního serveru

4.3.1 Monolitická architektura

Monolitická architektura je aplikace s jediným zdrojovým kódem a je sestavená z více modulů. Moduly jsou rozděleny buď podle byznys požadavků, anebo podle technických požadavků. Monolitická architektura se považuje za tradiční způsob vývoje aplikací. Taková architektura obvykle zahrnuje uživatelské rozhraní na straně klienta, server a databázi. Všechny funkce v monolitické aplikaci jsou spravovány a obsluhovány na jednom místě.

Aplikace s monolitickou architekturou se často vytváří s N-vrstvou strukturou, kde N je počet vrstev, které architektura vyžaduje. Klasickým příkladem je použití

třívrstvé architektury, kde jedna úroveň odpovídá za interakci s uživatelem, druhá za zpracování obchodní logiky a třetí zajišťuje přístup k datům v databázi.

Výhody:

- Jednoduchá na vývoj a monitorování;
- Jednoduchá na debugování a testování;
- Jednoduchá na nasazení.

Nevýhody:

- Čím větší aplikace, tím je složitější na pochopení;
- Škálovatelnost celé aplikace;
- Hůře udržitelný kód při delším vývoji.

4.3.2 Mikroservisní architektura

Myšlenkou mikroservisní architektury je rozdělení aplikace na malé, samostatně běžící komponenty (mikroservisy), které spolu při správném návrhu obvykle přímo nekomunikují. Klade se důraz na jejich maximální nezávislost. Mikroservisy poskytují své služby API gatewayi. Ta většinou vytváří centrální bod architektury a jednotné rozhraní mezi klientem a distribuovaným systémem – přijímá požadavky od klienta, přeposílá je do příslušných mikroservisů a také naopak skládá jejich odpovědi a posílá zpět klientovi. Hlavní výhodou tohoto přístupu je škálovatelnost [13]. Jednotlivé mikroservisy je možné spustit vícekrát a tím umožnují zvládat větší zátěže [14].

Výhody:

- Snáze pochopitelná;
- Každá servisa může využívat různé technologie;
- Snazší škálovatelnost po servisech;
- Autonomie komponent umožňuje paralelní vývoj v týmech;
- Nezávislé nasazování;
- Přepoužitelnost modulů.

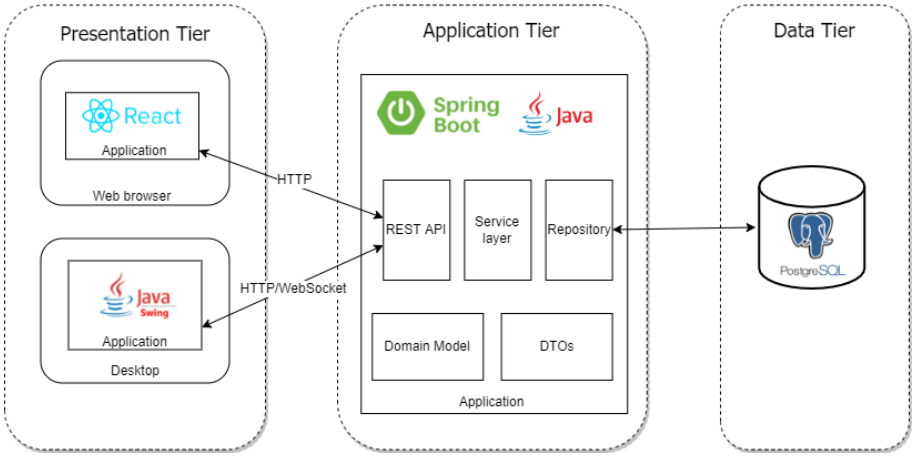
Nevýhody:

- Obtížnost návrhu;
- Vyžaduje více zdrojů;
- Náročné testování, konkrétně testování interakce mezi jednotlivými servisy;
- Složitější nasazování.

4.3.3 Výběr architektury

Pro účely kolaboračního pluginu jsem se rozhodla pro monolitickou architekturu s třívrstvou strukturou, jelikož není potřeba škálovat aplikaci. Kolaborační plugin se bude skládat z následujících vrstev, které jsou vidět na obrázku 4.2:

- Prezentační – odpovídá za grafické uživatelské rozhraní. V případě kolaboračního modulu bude uživatel interagovat jak s desktopovou aplikací Freeplane s grafickým rozhraním ve Swing, tak i s webovou React aplikací, která bude hlavně sloužit ke správě mind map a kolaborátorů;
- Aplikační – na této úrovni je uložena veškerá logika a funkčnost kolaboračního modulu. Prostřednictvím aplikační vrstvy probíhá komunikace mezi prezentační a datovou vrstvou. Komunikace mezi klientem a serverem bude probíhat pomocí REST rozhraní poskytované Spring Boot serverem, který bude detailně popsán v kapitole 5.2.1.
- Datová – úkolem této vrstvy je zpřístupnění dat z databáze. V kolaboračním serveru bude používána relační databáze PostgreSQL.



Obrázek 4.2: Architektura kolaboračního pluginu [vytvořeno autorem]

4.4 Analýza řešení založeného na sémantických technologiích

V této kapitole provedu analýzu sémantických technologií a pokusím se vyzkoumat jejich využití pro myšlenkové mapování. Dále na základě analýzy navrhnu alternativní řešení kolaboračního modulu, které bude založeno na sémantických technologiích.

4.4.1 Ontologie a sémantický web

Ontologie je explicitní a formalizovaný popis určité problematiky. „*Ontologie definuje soubor jedinců ve významu popisných prvků, pomocí kterých lze modelovat znalosti určitého oboru nebo problematiky. Kromě jedinců (reprezentantů) náleží obecně mezi popisné prvky třídy (soubory), atributy (vlastnosti) a vazby (vztahy mezi členy třídy).*“ [15]

S budováním ontologií souvisí sémantický web, s jehož přišel v roce 1998 Tim Berners-Lee s cílem rozšířit tehdy aktuální podobu webu o strukturované prvky, které pomohou strojům porozumět významu obsahu stránek. [16] K porozumění těmto strukturovaným prvkům slouží nejčastěji právě ontologie. Jejich schopnost zpracovávat a nakládat s daty je pro fungování sémantického webu velmi důležitá.

Při výzkumu ontologií a sémantického webu bylo vytvořeno a standardizováno více ontologických jazyků určených pro web. Mezi nejvýznamnější patří **OWL** (Web Ontology Language) vytvořený W3C [2]. Tento jazyk poskytuje formalizovaný sémantický popis dat založený na objektovém rozšíření jazyka XML pod názvem **RDF** (Resource Description Framework). RDF je postaven na specifikaci metadatového modelu. Tyto standardy se uplatnily i v dalších vědeckých i komerčních projektech.

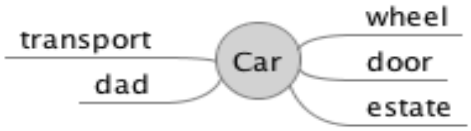
4.4.2 Využití sémantických technologií při myšlenkovém mapování

Jedním z vědeckých projektů, kde se uplatnily výše popsané technologie, je MONDIS, projekt vytvořený Katedrou kybernetiky Fakulty elektrotechnické ČVUT v Praze ve spolupráci s Ústavem teoretické a aplikované mechaniky Akademie věd České republiky. „*Projekt MONDIS je určen k podpoře profesionálních pracovníků zabývajících se stavebně-technickým stavem památkových objektů i návrhy na odpovídající opatření či stavební zásahy. K tomuto dochází s využitím znalostního systému určeného pro dokumentaci a analýzu poruch a poškození památkových objektů. Snahou je přitom překonat omezení starších databází a využít moderní sémantické technologie, které jsou schopné nezbytné informace o poškození a poruchách mnohem lépe vkládat, uspořádávat a zpracovávat.*“ [15]

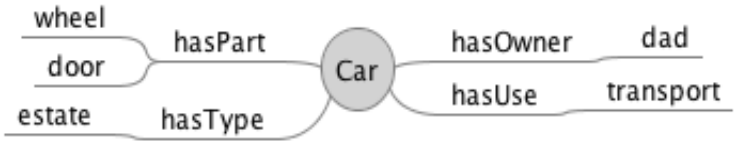
Součástí tohoto projektu je nástroj pro myšlenkové mapování řízený ontologiemi sémantického webu. V daném případě sémantický web umožňuje přiřadit uzlům myšlenkové mapy význam a vytvářet myšlenkové mapy pomocí navržení ontologických konceptů [17]. Nástroj je postaven na OWL ontologiích, které umožňují zachytit doménové znalosti přesně a důsledně. Avšak zavedení ontologie do struktury myšlenkových map má i své zápory. V diplomové práci [18], která se věnuje myšlenkovému mapování řízené ontologie a nástroji Ontomind, autor uvádí, že restrukturalizace myšlenkové mapy řízené ontologií je možná pouze na úrovni, kterou umožňuje ontologie. V Ontomindu ale vůbec není možné pochybovat s větví mapy. Na rozdíl od rychlého a jednoduchého procesu tvorby klasické myšlenkové mapy proces tvorby myšlenkové mapy řízené ontologií může být složitější a zabere víc času kvůli popisu ontologií. Občas to může vést i ke zbytečnému počtu uzlů. Nicméně normativní popis konkrétní domény popsané ontologií pomáhá lépe porozumět informacím, protože slovní zásoba a význam každého pojmu nebo slova použitého v myšlenkové mapě specificky popisuje ontologie a tím se odstraňuje případná nejednoznačnost. Rozdíl klasické myšlenkové mapy a myšlenkové mapy řízené ontologiemi je vidět na obrázcích 4.3 a 4.4. Myšlenková mapa zobrazená na obrázku 4.3 je rozhodně méně podrobná než myšlenková mapa na obrázku 4.4 a je patrný rozdíl v počtu použitých uzlů. Z toho plyne, že při tvorbě myšlenkových map řízené ontologiemi narážíme na problém verbalizmu.

Aplikace používá takzvanou komponentu AutocompleteTree, která je užitečná pro tvorbu myšlenkových map podle ontologických pojmů a vztahů. Tato kompo-

[2] W3C – World Wide Web Consortium, dostupné na `https://www.w3.org/` k 30.12.2021.



Obrázek 4.3: Ukázka klasické myšlenkové mapy [18]



Obrázek 4.4: Ukázka myšlenkové mapy řízené ontologiemi [18]

nenta slouží jako centrální zástupný bod přiřazování konceptů nebo vztahů pro nějaký určitý uzel mapy. Komponenta AutocompleteTree je velmi chytrá, protože čte informace z nadřazeného uzlu a poskytuje tak uživatelům pouze pojmy nebo vztahy povolené pro konkrétní uzel podle základní ontologie.

Ontomind je skvělým nástrojem pro tvorbu myšlenkových map řízenou ontologiemi, ale pro vytváření uzlů neontologických myšlenkových map není ovládání Ontomind tak snadné a uživatelsky přívětivé.

4.4.3 Integrace sémantických technologií do kolaboračního modulu

Alternativní řešení kolaboračního pluginu spočívá v tom, že se na datové vrstvě kromě relační databáze bude používat databáze triple store a do ní se budou ukládat myšlenkové mapy ve formátu RDF. K ukládání myšlenkových by bylo potřeba vytvořit OWL glosář, který by obsahoval deklarace vztahů pro uzly myšlenkové mapy.

Kapitola 5

Implementace

V rámci této kapitoly popíši detailněji použité technologie a architekturu kolaboračního modulu. Také se vrátím k funkčním a nefunkčním požadavkům z kapitoly 4.1 a zaměřím se na popisu jejich realizace.

5.1 Popis použitých technologií

Už na začátku bylo jasné, že kolaborační modul budu vyvíjet v programovacím jazyce Java, jelikož samotná aplikace Freeplane je napsána v Javě. Dalším důvodem pro výběr tohoto programovacího jazyka je to, že s Javou mám největší zkušenosti ze školy a z práce. Jakožto knihovnu pro serverovou část kolaboračního modulu jsem si vybrala Spring Boot, což je nadstavba pro Spring Framework [19]. Spring Boot usnadňuje proces vývoje tím, že již v sobě obsahuje přednastavení a eliminuje potřebu vytvářet boilerplate konfigurace. Dále pro ukládání mind map, kolaborátorů a uživatelů využívám relační databázi PostgreSQL.

V rámci vývoje uživatelského rozhraní jsem se rozhodla, že pro správu uživatelů, přehled myšlenkových map a kolaboraátorů vytvořím webovou aplikaci v React. Na stránky webové aplikace lze přejít z kolaboračního pluginu vytvořeného jako nadstavba pro desktopovou aplikaci Freeplane.

5.1.1 Backendová část

Spring

Spring (také **Spring Framework**) je populární javovská knihovna určená pro vývoj enterprise aplikace.

Jádro frameworku je tvořeno kontejnerem Inversion of Control (dále jen IoC). Princip IoC je postaven na principu vytváření, provázání a spravování životního cyklu objektů pomocí kontejneru. Tyto objekty se nazývají *Beans* a kontejner je injektuje na potřebná místa. Pojem injektování je znám jako návrhový vzor Dependency Injection (dále jen DI) neboli česky *vkládání závislostí*. DI slouží ke snížení závislostí mezi jednotlivými částmi programu.

Spring nám nabízí několik dalších projektů, které poskytují podporu pro různé architektury aplikací. Pro vývoj kolaboračního pluginu jsem využila následující Spring projekty:

- **Spring Boot**
  Mezi hlavní výhody této knihovny patří tvorba soběstačných (stand-alone)

aplikací, již zmíněná rychlost při startu projektu, snadná konfigurace a zabudovaný server Apache Tomcat, který lze spustit přímo z hlavní třídy bez jakékoli konfigurace a nutnosti nasazování WAR sousborů. Spring Boot opouští těžkopádné konfigurační soubory ve formátu XML v tradiční formě, které využívá Spring Framework a přechází na anotace.

- **Spring Data JPA**
  Poskytuje konzistentní přístup k uloženým datům v relační či nerelační databázi, přístup ke cloudovým datovým službám apod. Knihovna Spring Data JPA používá ve výchozím nastavení implementaci JPA[1] od Hibernate. Kromě nastavení přístupu k databázi není třeba nastavovat téměř nic dalšího.

- **Spring MVC**
  Jedná se o framework postavený na Servlet API, který zjednodušuje vývoj webových aplikací. Řídí se návrhovým vzorem Model-View-Controller. Zarhnuje vývoj REST webové služby.

- **Spring Security**
  Spring Security je standardem pro řešení problematiky autentizace a autorizace ve Spring aplikacích.

- **Spring Framework WebSocket**
  Spring Framework poskytuje podporu pro protokol WebSocket. *„Funguje jako vrstva aplikace, která přidává podporu pro komunikaci v reálném čase ... Ve Spring Frameworku je fallback implementován na základě protokolu SockJS. Komunikace mezi klientem a serverem bude na straně klienta probíhat pomocí „sockjs-client“, který emuluje W3C WebSocket API. Jelikož je protokol WebSocket nízkoúrovňový, tj. pracuje velmi blízko protokolu TCP, nespecifikuje žádný přesně daný styl (formát, nebo protokol) zpráv (styl zpráv může být specifikovaný při navazování WebSocket komunikace v hlavičce zprávy), proto Spring Framework přidává podporu pro protokol STOMP[2], která definuje, jak zprávy budou vypadat.“* [20]

5.1.2 Frontendová část

React

React je JavaScriptová knihovna pro tvorbu uživatelského rozhraní. [21] *„Používá koncept komponent jako částí uživatelského rozhraní a zjednodušujejejich budování. V kontextu Reactu jsou komponenty způsob, jakým zapouzdřit logiku a interakce určité části rozhraní. Tyto komponenty jsou reaktivní, a pro optimalizaci jejich aktualizací se využívá virtuální DOM[3]. Díky tomu je výkon aplikací využívající React velice vysoký. React byl vytvořen zaměst-nancem Facebooku, Jordanem Walkem, a vydán v roce 2013. V současnostije Facebookem také dále vyvíjen a udržován.“* [22]

[1] Java Persistance API
[2] Simple (nebo Streaming) Text Oriented Message Protocol
[3] Document Object Model

React knihovny

Dále jsem pro vývoj webové aplikace využila knihovny, které jsou uvedeny v tabulce 5.1.

Název knihovny	Popis
React router	Slouží k implementaci směrování (tzv. routing) v aplikaci.
Material UI	Poskytuje implementaci různých druhů hotových komponent (tlačítka, kartičky, tabulky, modální okna, formuláře apod.).
React final form	Používá se pro tvorbu formulářů a jejich validaci. Umožňuje spravovat stav bez nutnosti použití hooku *useEffect()*.

Tabulka 5.1: Tabulka použitých React knihoven.

5.1.3 Rozšíření Freeplane aplikace

Apache HttpComponents Core

HttpClient je implementace HTTP agenta. Je založen na HttpCore a je kompatibilní s HTTP/1.1. Poskytuje komponenty pro autentizaci na straně klienta, správu HTTP stavů a správu HTTP připojení.

Apache HttpComponents Client

HttpCore je sada transportních komponent pro HTTP přenos, které lze použít k vytvoření vlastních HTTP služeb na straně klienta a serveru s minimálními nároky.

Spring Boot Starter WebSocket

Jedná se o knihovnu, která byla popsána výše v sekci 5.1.1. V kolaboračním pluginu desktopové aplikace se používá pro hotovou implementaci WebSocket a STOMP klienta při komunikaci se serverem.

5.2 Popis struktury kolaboračního modulu

V této sekci popíši jednotlivé části kolaboračního pluginu a způsob, jakým mezi sebou komunikují.

5.2.1 Backendová část

Jádrem aplikační vrstvy kolaboračního pluginu je backendová aplikace Spring Boot. Kód této aplikace je rozdělen na balíčky. Jednotlivé balíčky reprezentují logicky provázané celky, které jsou používány k vykonávání určité činnosti. Aplikace se skládá z následujících balíčků:

- **config** – obsahuje nastavení pro zabezpečení aplikace a nastavení pro protokol WebSocket;

- **dao** – balíček obsahující definice rozhraní, které jsou využívány knihovnou Spring Data JPA pro poskytnutí jednoduchého způsobu práce s databázovou vrstvou;
- **exception** – obsahuje mnou vytvořené výjimky používané k oznámení o chybovém stavu aplikace;
- **listener** – obsahuje posluchač údalostí, konkrétně posluchač událostí připojení a odpojení k protokolu WebSocket;
- **model** – obsahuje třídy reprezentující entity využívané k načítání a ukládání do databáze;
- **rest** – obsahuje třídy, na jejichž základě Spring poskytuje RESTové rozhraní pro komunikaci s aplikací;
- **security** – obsahuje implementace základních rozhraní pro autentizaci a pomocnou třídu se statickou metodou pro získání aktuálního autorizovaného uživatele;
- **service** – v tomto balíčku je obsažená business logika aplikace.

5.2.2 Frontendová část

Základ implementace React aplikace se nachází ve složce /src, která má následující strukturu:

- **utils**

  Tato složka obsahuje soubor *request.js* s pomocnou funkcí *request* pro odeslání a zpracování požadavků na server. Jako parametry tato funkce bere *url* – URL[4] API endpointu a *options*, kam se vkládají hlavičky, body a metoda requestu. Tuto funkce je vidět na ukázce kódu 5.1.

```
export const request = async (
    url,
    options
) => {
    try {
        const response = await fetch(url, options);
        if (!response.ok) {
            return Promise.reject({
                status: response.status,
                statusText: response.statusText,
            });
        }

        if (options.method === 'GET') {
            const data = await response.json();
            return data;
        }
```

[4]Uniform Resource Locator

```
        else {
            const data = await response;
            return data;
        }
    } catch (error) {
        return Promise.reject(error);
    }
};
```

Zdrojový kód 5.1: Ukázka pomocné funkce *request*

- **actions**

  Zde se nachází soubor s metody požadavků GET, DELETE, POST, PUT, které posílám na backend. V základu těchto metod se používá pomocná funkce *request*, která je popsána a ukázána výše. Příklad takové metody je vidět na ukázce kódu 5.2.

```
// GET all users
export const getAllUsers = async () => {
    const token = localStorage.getItem("token");
    const requestOptions = {
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `${token}`
        },
    };

    const response = await request(`/users`, requestOptions
    ).then(data => {
        return data;
    })
    return await response
}
```

Zdrojový kód 5.2: Ukázka metody GET pro načítání uživatelů

- **components** se dělí na:
  - **fields** obsahuje komponenty s poli, které byly použity ve formulářích.
  - **modules** obsahuje komponenty s implementací celých stránek.
  - **ui** obsahuje komponenty s implementací samostatných elementů, které byly použity na stránkách.
- **img**

  Tato složka je určena k ukládání obrázků. Zde je uloženo logo aplikace Freeplane.

- soubor **App.js**, kde se nachází hlavní komponenta App. Zde se staví routing aplikace pomocí knihovny React Router. Směrování uživatele na jednotlivé stránky se provádí na základě jeho role.

5.2.3 Rozšíření Freeplane aplikace

Podle návodu[5] na stránkách Freeplane desktopová aplikace byla rozšířena o kolaborační plugin. Hlavní třídou, která aktivuje samotný plugin a registruje nové akce (Actions), je `Activator.java`. Zdrojový kód této třídy je uveden na 5.3.

```
public class Activator implements BundleActivator {

@Override
public void start(BundleContext context) throws Exception {
    registerMindMapModeExtension(context);
}

private void registerMindMapModeExtension(final BundleContext context){
    final Hashtable<String, String[]> props = new Hashtable<>();
    props.put("mode",
                new String[]{MModeController.MODENAME,
                SModeController.MODENAME}
    );
    context.registerService(
        IModeControllerExtensionProvider.class.getName(),
        (IModeControllerExtensionProvider) modeController -> {
            if (modeController.getModeName().equals("MindMap")) {
                modeController.addAction(new OpenWebPluginAction());
                modeController.addAction(new ServerConnectionAction());
                modeController.addAction(new OpenServerMindMapAction());
                modeController.addAction(new SaveServerMindMapAction());
            }
        }, props);
    }

    @Override
    public void stop(BundleContext context) throws Exception {
    }
}
```

Zdrojový kód 5.3: Ukázka třídy pro aktivace pluginu v desktopové aplikaci Freeplane

Do pluginu jsem přidala akce na připojení k serveru, přesměrování do webové aplikace, otevření myšlenkové mapy ze serveru pro možnost kolaboraci a uložení mapy na server. Pro spojení se serverem jsem vytvořila třídu ConnectionService, která odpovídá za autorizaci, odesílání požadavků a nastavení komunikace WebSocket.

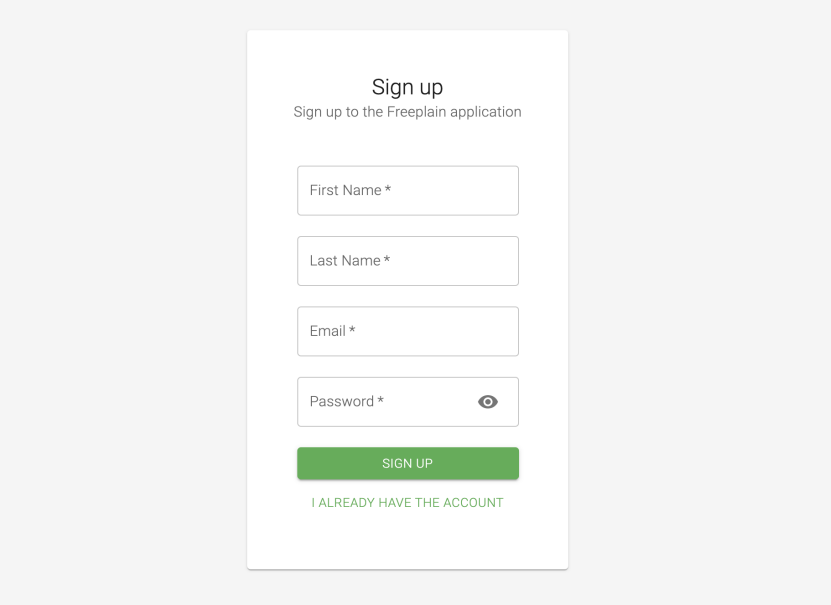
5.3 Realizace požadavků na kolaborační modul

V této podkapitole více rozeberu funkční a nefunkční požadavky, které mají prioritu *must have* nebo *should have*.

[5] dostupné z `https://www.freeplane.org/wiki/index.php/Plugin_development` k 1.1.2022

FP1.1 Registrace e-mailem a heslem

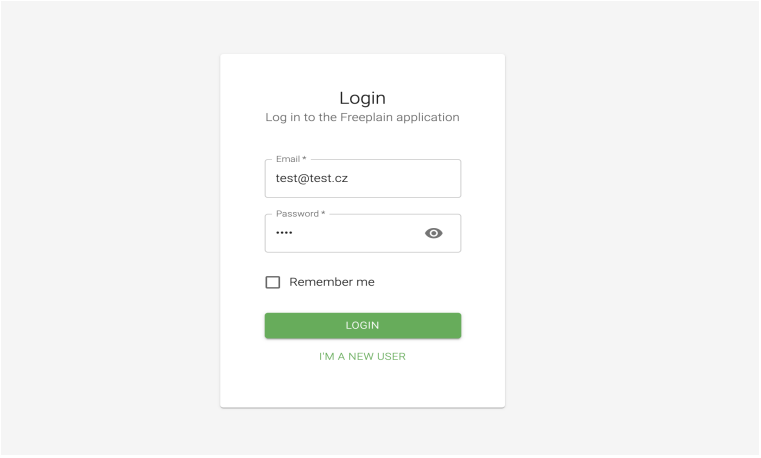
Ve webové části kolaboračního pluginu byla vytvořena stránka s formulářem pro registraci. Na polích formuláře je nastavena validace pomocí knihovny React Final Form. Na obrázku 5.1 je vidět, jak vypadá registrační formulář.



Obrázek 5.1: Registrační formulář kolaboračního modulu [vytvořeno autorem]

FP2.1 Přihlášení e-mailem a heslem

Ve webové části kolaboračního pluginu byla vytvořena stránka s přihlašovacím formulářem. Na polích formuláře je také nastavena validace pomocí knihovny React Final Form. Na obrázku 5.2 je vidět, jak vypadá přihlašovací formulář.



Obrázek 5.2: Přihlašovací formulář kolaboračního modulu [vytvořeno autorem]

FP2.3 Automatické přihlášení

V přihlašovacím formuláři je navíc zaškrtávací pole *remember me*, které je vidět na obrázku 5.2. Uživatel zaškrtne toto pole, pokud bude chtít, aby aplikace zapa-

matovala jeho relaci a opakovaně se neptala na přihlášení. Pokud uživatel nechá toto pole nezaškrtnutým, pak aplikace si zapamatuje relaci uživatele pouze na 60 minut. Po uplynutí této doby uživatel bude upozorněn na odhlášení z aplikace a přesměrován na stránku login.

FP3 Role uživatelů na úrovni aplikace

Role uživatelů na úrovni aplikace se řeší pomocí knihovny Spring Security. V databázi v tabulce *User*, ve sloupci *isAdmin* eviduji příznak, jestli daný uživatel je administrátorem. Po úspěšné autentizaci se uživatelům přidělují různá oprávnění (tzv. Granted Authorities). Role se překládá na *GrantedAuthority* s prefixem „ROLE_“ a vkládá se do bezpečnostního kontextu. Kolekci přeložených *GrantedAuthority* vrací třída `UserDetails`, která implementuje stejnojmenné rozhrání.

Metody, které vyžadují zvyšená práva administrátora, jsou zabezpečeny na úrovni kontrolérů pomocí anotace ve zdrojovém kódu 5.5.

```
1 @PreAuthorize("hasRole('ROLE_ADMIN')")
```

Zdrojový kód 5.4: Ukázka anotace pro zabezpečení na základě rolí

Také pro některé endpointy, kde je potřeba ověřit roli administrátora, využívám metodu pro načítání aktuálně přihlášeného uživatele `getCurrentUserDetails()`. Přes tuto metodu pak přistupuji k roli uživatele.

```
1  public static UserDetails getCurrentUserDetails() {
2      final SecurityContext context = SecurityContextHolder.getContext();
3      if (context.getAuthentication() != null &&
4              context.getAuthentication().getPrincipal()
5                      instanceof UserDetails) {
6          return (UserDetails)context.getAuthentication().getPrincipal();
7      } else {
8          return null;
9      }
10 }
```

Zdrojový kód 5.5: Ukázka metody pro načítání aktuálně přihlášeného uživatele

FP4.1 Sdílení mind mapy odkazem

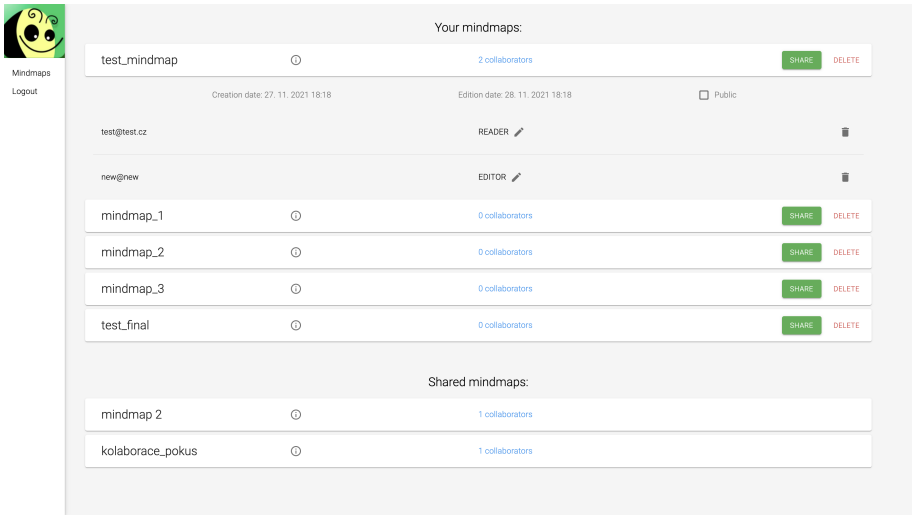
Ve webové části kolaboračního pluginu byla naimplementována možnost sdílení myšlenkové mapy odkazem.

FP5.1 Role uživatelů na úrovni mind mapy

Podle analýzy by kolaborační modul měl definovat role uživatelů na úrovni myšlenkové mapy pomocí třech rolí: *Owner*, *Reader* a *Editor*. Vlastníka mind mapy poznám podle sloupce *creator_id* v databázi v tabulce *Mindmap*. Role *Reader* a *Editor* eviduji v samostatné tabulce *Collaboration*, kam ukládám identifikační číslo uživatele, identifikační číslo myšlenkové mapy a roli. Kdyby se stalo, že uživatel provolá endpoint pro cizí myšlenkovou mapu, tak se mu vrátí HTTP status kód *HTTP 403 FORBIDDEN*.

FP5.2 Způsoby přístupů k mind mapě

Podle analýzy kolaborační modul by měl rozlišovat 2 způsoby přístupů k mind mapě. První způsob přístupu spočívá v tom, že se mapa zobrazí v seznamu vlastních mind map. Druhým způsobem je přístup ke sdíleným mind mapám, kde se mapa zobrazí v seznamu se sdílenými mapami. Na obrázku 5.3 je vidět, jak to vypádá ve webové části kolaboračního modulu.



Obrázek 5.3: Ukázka přehledu vlastních a sdílených myšlenkových map [vytvořeno autorem]

FP5.3 Přidělení práv na úpravu mind map

Ve formuláři při odeslání pozvánky ke kolaboraci se vypňuje e-mail uživatele, pro kterého pak mind mapa bude nasdílena.

FP6 Kolaborace

Pro kolaboraci mezi uživatele byla využita technologie WebSocket, která je popsána v sekci 5.1.1. Kolaborace byla implementována tak, že při otevření myšlenkové mapy ze serveru v desktopové aplikaci se spouští WebSocket komunikace se serverem. Serverovou konfiguraci je vidět na zdrojovém kódu 5.7. V metodě konfigurace *registerStompEndpoints* se registruje STOMP endpoint na základě fallbacku SockJs. Druhá metodá *configureMessageBroker* konfiguruje STOMP message broker. Destinace s prefixy `/queue` a `/topic` budou zpracovávány vestavěným brokerem STOMP, když destinace s prefixem `/app` budou zpracovávány pomocí anotovaných kontrolérů Springu.

Pro vysílání skrz WebSocket je potřeba také nastavit javovského `SockJsClient` a `WebSocketStompClient` na stráně klienta. Dále naplním potřebné hlavičky pro navázání spojení a spustím příkaz CONNECT. Celkové nastavení WebSocket v klientské části vypadá následovně.

```
1 List<Transport> transports = new ArrayList<>();
2 transports.add(new WebSocketTransport(new StandardWebSocketClient()));
3 transports.add(new RestTemplateXhrTransport());
4 WebSocketClient webSocketClient = new SockJsClient(transports);
5
```

```
WebSocketStompClient webSocketStompClient = new WebSocketStompClient(
                webSocketClient);
webSocketStompClient.setMessageConverter(new StringMessageConverter());

StompHeaders stompHeaders = new StompHeaders();
stompHeaders.set("email", email);
stompHeaders.set("mindmap", mindmapId);
stompHeaders.set("Authorization", this.authorizationHeader);

WebSocketHttpHeaders webSocketHttpHeaders = new WebSocketHttpHeaders();
webSocketHttpHeaders.add("Authorization", this.authorizationHeader);

ListenableFuture<StompSession> session = webSocketStompClient.connect(
                "http://localhost:8080/websocket",
                webSocketHttpHeaders,
                stompHeaders,
                stompSessionHandler());
```

Zdrojový kód 5.6: Ukázka klientské konfigurace Websockets

Jakmile se připojením vytvoří nová session, klient se přihlásí k odběru zpráv běžícího brokera, konkrétné se přihlásí k destinacím `/app/subscribe` a `/queue/user/:e-mail /responses`, kde :e-mail je e-mail uživatele.

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig
implements WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/websocket");
        registry.addEndpoint("/websocket").withSockJS();
    }

    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.enableSimpleBroker("/queue", "/topic");
        registry.setApplicationDestinationPrefixes("/app");
        registry.setPreservePublishOrder(true);
    }

    @Bean
    public SocketSessionRegistry SocketSessionRegistry() {
        return new SocketSessionRegistry();
    }
}
```

Zdrojový kód 5.7: Ukázka serverové konfigurace WebSocket

Zprávy od STOMP klientů a zprávy odeslané opačným směrem lze zpracovávat pomocí anotovaných kontrolérů Springu a šablony zpráv.

`@SubscribeMapping` anotace se používá pro jednorázové zasílání zpráv z aplikace klientům. Server odešle příkaz MESSAGE do stejné destinace `/app/subscribe` bez zapojení brokera.
Anotace `@MessageMapping` se používá pro opakované zasílání zpráv z aplikace klientům. V mém případě do destinace `/app/request` klient posílá první zprávu obsahující e-mail uživatele a identifikační číslo myšlenkové mapy, kterou si vybral uživatel dřív. Server pomocí metody anatované `@MessageMapping` přijme příkaz SEND od klienta, následně rozparsuje zprávu, ověří, jestli myšlenková mapa je momentálně uzamčená a odešle odpověď příkazem MESSAGE do explicitní destinace `/queue/user/:e-mail/responses`. Implementace této metody je uvedena ve zdrojovém kódu 5.8.

```
@MessageMapping("/request")
public void handleRequestMessage(String message) {
    if (message == null) {
        throw new RuntimeException(
                String.format("'%s' is rejected", message)
        );
    }

    String response = "not locked";
    String mindmapId = message.substring(0, message.lastIndexOf(","));
    String email = message.substring(message.lastIndexOf(",") + 1);

    MindmapLockInfo lockInfo = webAgentSessionRegistry
                .getMindMapLockInfo(mindmapId);
    if (lockInfo != null) {
        if (!lockInfo.getLockedBy().equals(email)) {
            response = "locked";
        }
    }
    logger.info("Message with response: {}", response);
    this.messageSendingOperations.convertAndSend(
                "/queue/user/" + email + "/responses", response
    );
}
```

Zdrojový kód 5.8: Ukázka anotované metody serveru *handleRequestMessage*

Pokud mapa bude uzamčená jiným uživatelem, aplikace zobrazí okno s upozorněním. Když nebude uzamčená, uživatel bude moct upravovat myšlenkovou mapu. Po provedení úprav je potřeba myšlenkovou mapu uložit explicitně pomocí záložky *Save on server* v menu.

NFP1 Hashování hesel

K hashování a solení hesel byla použita metoda *encode* třídy `BCryptPasswordEncoder` z Spring Security frameworku.

Kapitola 6

Testování

Tato kapitola se věnuje testování a zhodnocení dosažených výsledků.

6.1 Testování pomocí aplikace Postman

Aplikační rozhraní se během vývoje testovalo pomocí nástroje Postman. Postman je zajímavým nástrojem na vývoj a testování API, který umožňuje vytvářet a odesílat HTTP požadavky na server. Pomocí Postman jsem vytvořila sadu požadavků pro každý z kontrolérů. Kolekce požadavků je součástí příloh.

6.2 Uživatelské testy

Cílem tohoto testování bylo ověřit funkcionalitu výsledného kolaboračního modulu na třech uživatelích podle testovacích scénářů.

Testovací scénář č. 1

1. Spustit desktopovou aplikaci Freeplane.
2. Otevřít webové rozhraní z desktopové aplikace.
3. Přihlásit se jako administrátor a zaškrtnout *remember me* (přihlášovací údaje – e-mail: *test@test.cz*, heslo: *1234*).
4. Otevřít seznam všech uživatelů.
5. Vytvořit účet pro nového uživatele (registrační údaje – jméno: *Alice*, příjmení: *Novakova*, e-mail: *alice@mail.com*, heslo: *alice123*).
6. Přidat nově přidaného uživatele jako kolaborátora pro myšlenkovou mapu s rolí *READER*.
7. Upravit jeho roli na *EDITOR*.
8. Uzavřít webovou stránku a vyzkoušet otevření webové aplikace z Freeplane ještě jednou.
9. Odhlásit se.

Testovací scénář č. 2

1. Spustit desktopovou aplikaci Freeplane.
2. Přejít na stránky webové aplikace z desktopové aplikace.
3. Přihlásit se jako normální uživatel (přihlášovací údaje – e-mail: *bob@test.cz*, heslo: *1234*).
4. Přejít na seznam myšlenkových map.
5. Rozkliknout seznam kolaborátorů jedné z myšlenkových map.
6. Smazat libovolného kolaborátora.
7. Nasdílet myšlenkovou mapu pro nového uživatele (e-mail: *alice@mail.com*, role: *READER*).
8. Přejít zpátky na desktopovou aplikaci.
9. Připojit se k serveru.
10. Otevřít myšlenkovou mapu ze serveru.
11. Spustit aplikaci Freeplane v druhém okně.
12. Připojit se k serveru v druhém okně aplikace (přihlášovací údaje – e-mail: *alice@mail.com*, heslo: *alice123*).
13. Otevřít stejnou mind mapu jako v bodě 10.
14. Vrátit se k prvnímu oknu.
15. Uložit myšlenkovou mapu na server.

6.3 Zhodnocení výsledků testování

Skoro všechny pokyny uživatele provedli bez zásadních problémů, ale samozřejmě testování odhalilo některá slabá místa kolaboračního modulu. Uživatelské přípomínky jsou popsány níže:

- **Složité spuštění kolaboračního modulu**

  Spuštění celého balíku aplikace je docela náročný proces. Nasazení serverové aplikace by mohlo zjednodušit testování kolaboračního modulu a simulace kolaborace. Odpadne potřeba mít otevřená dvě okna desktopové aplikace Freeplane a každý z uživatelů by normálně spustil aplikaci Freeplane a začal kolaborovat.

- **Otevření webové aplikace**

  Uživatelům nebylo jasno, jak se dá otevřít webovou aplikaci z desktopové aplikace Freeplane. Příčinou je to, že tlačítko na otevření webové aplikace se nachází pod záložkou *Server* v horním menu desktopové aplikace. Bylo navrhnuto posunout toto tlačítko pod samostatnou záložku *Web*.

- **Vynucené obnovení stránek webové aplikace**

  Uživatelům vadily hlášky a následné obnovení stránky po aktualizaci dat kolaborátorů. To je způsobeno tím, že struktura HTTP odpovědi pro GET metodu *getAllMindmaps* je velice vnořená a složitá, tím pádem to komplikuje aktualizaci stavu komponenty Mindmaps a Groups. Aktualizaci dat jsem vyřešila vynuceným obnovením stránky.

Závěr

Cílem této bakalářské práce bylo zanalyzovat existující nástroje pro tvorbu myšlenkových map s podporou kolaborace, definovat požadavky pro kolaboraci a uživatelské scénáře pro práci s nástrojem, navrhnout vlastní řešení a naimplementovat toto rozšíření do aplikace Freeplane.

V rámci analýzy jsem prozkoumala několik nástrojů pro tvorbu myšlenkových map s podporou kolaborace a rozebrala původní návrh kolaboračního serveru od komunity vývojářů Freeplane. Tento výstup mi pak sloužil jako zamyšlení se nad vlastním návrhem kolaboračního modulu. Během návrhu vlastního řešení jsem nadefinovala funkční a nefunkční požadavky a detailně rozepsala uživatelské scénáře. Ve fázi návrhu architektury kolaboračního modulu jsem porovnala monolitickou a mikroservisní architekturu, na základě čehož jsem se rozhodla pro monolickou architekturu.

Výstupem je funkční kolaborační modul, který se skládá z pluginu integrovaného do desktopové aplikace Freeplane, samostatného serveru a uživatelského rozhraní vytvořeného pomocí JavaScriptové knihovny React. Serverová část je implementována v jazyce Java za použití frameworku Spring Boot.

Tato práce mi přinesla mnoho cenných zkušeností v oblasti analýzy softwaru a programování. Během vývoje jsem narazila na problémy se spuštěním a pochopením kódu aplikace Freeplane. Nicméně se mi nakonec povedlo rozšířit desktopovou aplikaci Freeplane o kolaborační plugin a rozběhnout základní kolaboraci pomocí technologie WebSocket, se kterou jsem dřív neměla žádné zkušenosti. Také v rámci analýzy řešení založeného na sémantických technologiích jsem se poprvé seznamila s oblastí ontologie a sémantického webu.

Bibliografie

1. BUZAN, Tony; BUZAN, Barry. *Myšlenkové mapy: probuďte svoji kreativitu, zlepšete svou paměť, změňte svůj život.* Brno: BizBooks, 2012. ISBN 978-80-265-0030-8.
2. KRÁLIKOVÁ, Dominika. *Webová aplikace pro interaktivní myšlenkové mapy.* 2019. B.S. thesis. České vysoké učení technické v Praze, Fakulta informačních technologií.
3. GUPTA, Yash; DEWAN, Himanshu; LEEKHA, Alka. Real-time monitoring using AJAX and WebSockets. *Journal of Statistics and Management Systems.* 2020, roč. 23, č. 1, s. 125–134.
4. ORACLE. 2014. Dostupné také z: `https://docs.oracle.com/javaee/7/tutorial/websocket001.htm#BABDABHF`.
5. MORDY, James. *The Best Free and Open Source Mind Mapping Software.* 2020. Dostupné také z: `https://www.goodfirms.co/blog/best-free-and-open-source-mind-mapping-software`.
6. MEISTERLABS. *MindMeister.* MindMeister, 2020. Dostupné také z: `https://www.mindmeister.com/`.
7. MEISTERLABS [online]. [N.d.] [cit. 2020-12-30]. Dostupné z: `https://developers.mindmeister.com/`.
8. JURÁNKOVÁ, Kristýna. *Tvorba myšlenkových map v online prostředí.* Medium, 2017. Dostupné také z: `https://medium.com/@kristyna.jurankova/tvorba-my%C5%A1lenkov%C3%BDch-map-v-online-prost%C5%99ed%C3%AD-83bb9ef9936b#_ftnref1`.
9. Bitbucket, 2015. Dostupné také z: `https://bitbucket.org/wisemapping/wisemapping-open-source/src/master/`.
10. *PostgreSQL 9.0.23 Documentation.* 2005-2020. Dostupné také z: `https://developer.mozilla.org/en-US/docs/Web`.
11. ALLEN, David. 2020. Dostupné také z: `https://gettingthingsdone.com/what-is-gtd/`.
12. LTD, XMind. *XMind - Full-featured mind mapping and brainstorming tool.* 2020. Dostupné také z: `https://www.xmind.net/`.
13. VAVŘÍKOVÁ, Monika. 2017. Dostupné také z: `https://www.aspectworks.com/cs/mikroservisy-a-jejich-monitoring`.
14. ŽUFÁNEK, Jakub. *Podnikový informační systém postavený na architektuře microservice.* Univerzita Pardubice, 2019.

15. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, Fakulta elektrotechnická. *Mondis systém*. 2012. Dostupné také z: `https://www.mondis.cz/web/portal/mondis-system`.

16. BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The semantic web. *Scientific American*. 2001, roč. 284, č. 5, s. 1–5.

17. KŘEMEN, Petr; MIČKA, Pavel; BLAŠKO, Miroslav; ŠMÍD, Marek. Ontology-driven mindmapping. In: *Proceedings of the 8th International Conference on Semantic Systems*. 2012, s. 125–132.

18. SALAI, Dominik. *Myšlenkové mapy s použitím sémantických technologií*. 2019. Dipl. České vysoké učení technické v Praze, Fakulta Elektrotechnická.

19. *Spring Boot Reference Documentation 2.6.2*. 2012-2021. Dostupné také z: `https://spring.io/projects/spring-boot`.

20. POKORNÝ, Miroslav. *Real-time komunikace ve webových aplikacích*. Univerzita Hradec Králové, 2017.

21. *React official documentation*. Facebook Open Source, 2021. Dostupné také z: `https://reactjs.org`. verze 17.0.2.

22. MARTINEK, Vlastimil. *Srovnání moderních javascriptových frameworků Vue.js a React.js*. 2018. Masarykova univerzita, Fakulta informatiky.

Přílohy

A Elektronické přílohy

- *server* - zdrojový kód Spring Boot serveru kolaboračního modulu
- *client* - zdrojový kód React webové aplikace
- *freeplane* - zdrojový kód desktopové aplikace Freeplane rozšířené o kolaborační plugin
- *postman.json* - sada dotazů pro testovácí nástroj Postman
- *README.MD* - soubor s návodem na spuštění kolaboračního modulu

B Ukázky kódu

B.1 Ukázka dat metody do


```
"data":{
   "changelists":[
      [
         {
            "idea_id":1700162750,
            "type":"Reposition",
            "old_data":{
               "pos":[
                  -317,
                  -242
               ]
            },
            "new_data":{
               "pos":[
                  -414,
                  -242
               ]
            },
            "id":991
         },
         {
            "idea_id":1700162750,
            "type":"Title",
            "old_data":{
               "title":"uzel"
            },
            "new_data":{
               "id":1700162750,
               "title":"novy nazev uzlu"
            },
            "id":992
         }
      ]
   ],
   "idea_id":1700162750
}
```

B.2 Ukázka API v XMind

```
String getFile();  //  return the name of the current default file
void setFile(String file);  // set the name of the current default file

void save();    // save the workbook to the current default file
void save(String file);  // save the workbook to the specified file name
void save(OutputStream output);  // save the workbook to
the specified OutputStream
void save(IOutputTarget target);  // save the workbook to the target

void saveTemp();

void setTempStorage(IStorage storage);
IStorage getTempStorage();
void setTempLocation(String tempLocation);
String getTempLocation();
```

C Porovnání kolaboračního pluginu s existujícími nástroji

	Mindmeister	Wisemapping	Freemind	Freeplane
Obecná kritéria				
Operační systém				
Windows	✓	✓	✓	✓
Mac OS X	✓	✓	✓	✓
Linux	✓	✓	✓	✓
Úložiště				
Cloudové	✓	✓		✓
Lokální			✓	✓
Autorizace				
Ano	✓	✓		✓
Ne			✓	
Undo/Redo				
Ano	✓	✓	✓	✓
Ne				
Kritéria pro kolaboraci				
Sledování úprav real-time				
Ano	✓			
Ne		✓		✓
Přidělení práv na mapu				
Ano	✓	✓		✓
Ne			✓	
Způsoby sdílení mapy				
Odkaz	✓	✓		✓
Odeslání pozvánky mailem	✓	✓		
Připojení k mapě přes port			✓	
Paralelní úprava mapy				
Ano	✓			
Ne				
Uzamčení mapy na dobu úprav		✓		✓
Uzamčení části mapy				
Verzování				
Ano	✓	✓		
Ne				✓
Rollback změn				
Ano		✓		
Ne	✓			✓
Rezoluce konfliktů				
Ano				
Ne	✓	✓		✓

Tabulka 1: Porovnání kolaboračního pluginu s existujícími nástroji