F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Kooperativní myšlenkové mapy

Michal Kalista
Softwarové inženýrství a technologie

Květen 2023
Vedoucí práce: Ing. Petr Aubrecht, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kalista** Jméno: **Michal** Osobní číslo: **499273**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Kooperativní myšlenkové mapy

Název bakalářské práce anglicky:

Cooperative Mindmaps

Pokyny pro vypracování:

Myšlenkové mapy jsou velmi užitečným pomocníkem při zpracování nějakého tématu, zapisování dosavadních znalostí a při analýze.
V současnosti existuje dostatek nástrojů pro jejich pořizování. Existují i online editory, které je možné používat odkudkoliv. Jejich nedostatkem je ale možnost spolupráce několika lidí najednou, jako je tomu například u Google Office, kdy několik uživatelů může najednou psát do jednoho dokumentů. Aplikace myšlenkových map se omezuje na možnost uložení a načtení do cloudu, což např. při online meetingu nutí mít jediného zapisovatele. Také zde nastupuje oblast placených vlastností daných aplikací. Dále neposkytují tyto aplikace žádné použitelné API, které by dovolilo práci s mapou nebo rozšíření zobrazovaných informací.
Tématem této práce je webová aplikace, která dovolí několika uživatelům najednou měnit jednu myšlenkovou mapu. Uzly a hrany mají základní množinu atributů, jako je text, barva a ikona (pro začátek stačí UNICODE znak).
Rozšíření dobudoucna zahrnuje REST API pro ovládání mapy, např. integrace s nějakým dalších systémem, která bude upravovat stav uzlů, případně je vytvářet a mazat.
Dalším rozšířením je použít tento software jako knihovnu, přidat nové atributy a předefinovat vizualizaci uzlů, např. progressbar zobrazující, nakolik je daná položka zpracována.
Požadavky
1. Zanalyzujte a popište aktuální stav software pro online myšlenkové mapy.
2. Implementujte webovou aplikaci podle zadání. Aplikace musí umět editovat myšlenkovou mapu jak pomocí myši, tak ideálně i z klávesnice (je to rychlejší). Změny provedené jakýmkoliv uživatelem se musí okamžitě zobrazit na všech ostatních instancích. Je potřeba vyřešit konflikty, a to co nejjednodušejí, přinejhorším zrušením změny.
3. Implementaci vyzkoušejte z uživatelského hlediska - dva uživatelé mění současně mapu a zároveň vidí změny druhého. Protože unit testy nedávají v tomto případě valný smysl, zauvažujte o automatických testech alespoň pro editaci mapy jedním uživatelem, případně to samé pro REST rozhraní.
4. Řešení implementujte v technologii JakartaEE. Technologie na uživatelské rozhraní je na vůli studenta, musí ji odůvodnit.

Seznam doporučené literatury:

[1] The Jakarta EE 8 Tutorial: https://eclipse-ee4j.github.io/jakartaee-tutorial/
[2] Jakarta EE Cookbook - Second Edition, https://www.packtpub.com/product/jakarta-ee-cookbook/
[3] Patterns of Enterprise Application Architecture — Martin Fowler
[4] Popis domény: https://en.wikipedia.org/wiki/List_of_concept-_and_mind-mapping_software
[5] Dostupné implementace: FreePlane, wisemapping.com, coggle.it, MindMup, https://www.popplet.com/, https://bubbl.us

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Aubrecht, Ph.D. katedra počítačů FEL

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

Datum zadání bakalářské práce: **30.01.2023** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **22.09.2024**

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

III. PŘEVZETÍ ZADÁNÍ

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

. _____ _____

Datum převzetí zadání Podpis studenta

Poděkování / Prohlášení

Chtěl bych poděkovat svému vedoucímu Ing. Petru Aubrechtovi, Ph.D. za vedení práce, jeho vstřícnost a cenné rady, bez kterých by práce nedosahovala výsledné míry propracovanosti. Také bych chtěl poděkovat své rodině za podporu při studiu.

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

.....

Abstrakt / Abstract

Myšlenková mapa je oblíbený nástroj pro grafické znázornění myšlenkových postupů. Při týmové tvorbě a sdílení myšlenek je důležitá možnost spolupráce – kooperace nad mapou. Dostupné nástroje pro tvorbu a úpravu map poskytují nedostatečnou možnost kooperace nebo je možnost kooperace finančně omezena.

Cílem této práce je analýza dostupných nástrojů a vytvoření nové aplikace pro tvorbu a úpravu myšlenkových map. Hlavním požadavkem na vytvářenou aplikaci je možnost kooperace.

Klíčová slova: myšlenková mapa, kooperace, Jakarta EE

A mind map is a popular tool for graphically representing thought processes. When team members share ideas, the possibility of cooperation is important. The tools available for creating and editing maps provide an insufficient possibility of cooperation, or the possibility of cooperation is financially limited.

The goal of this work is the analysis of available tools and the creation of a new application for creating and editing mind maps. The main requirement for the application being created is the possibility of cooperation.

Keywords: mind map, cooperation, Jakarta EE

Title translation: Cooperative mind map

/ Obsah

/ Obrázky

Kapitola 1
Úvod

Myšlenková mapa je grafické uspořádání myšlenek a problémů do souvislostí. Je to nástroj, který pomáhá člověku analyzovat problém a uspořádat dílčí podproblémy do vztahů. Za vynálezce myšlenkové mapy je považován Tony Buzan. Podle něj lze myšlenkové mapy používat například k plánování, brainstormingu, výběru postupu, učení, nebo organizaci myšlenek [1–2].

Kooperace je označení pro spolupráci [3]. Zde kooperace vyjadřuje spolupráci více lidí najednou tak, aby nebyli limitováni možnostmi editoru. Všichni uživatelé mohou používat editor současně. Zároveň změny každého uživatele se ihned projevují ostatním uživatelům. Ilustrativním příkladem zde může být fungování sdílených dokumentů google, kde všichni uživatelé mohou upravovat obsah a změny jsou ihned patrné.

Pro vytváření a editování myšlenkových map existuje celá řada editorů. Některé editory lze používat i online. Téměř všechny ale postrádají možnost kooperace více členů, anebo ji umožňují jen částečně. Z tohoto důvodu myšlenkovou mapu ve většině případů vytváří jen jeden člen a ostatní členové jen dávají náměty. Tento postup je méně efektivní, než kdyby s mapou mohli pracovat všichni členové najednou. Ze zbývajících editorů, velká většina umožňuje kooperaci jen ve zpoplatněných verzích. Verze bez poplatků bývá jen ukázková s velmi omezeným počtem upravovatelných map.

Cílem této práce je analyzovat již existující dostupné nástroje a navrhnout a implementovat nástroj vlastní pro vytváření a úpravu myšlenkových map. Tento nástroj má uživatelům umožňovat týmovou spolupráci nad myšlenkovou mapou, aniž by její úprava byla omezena. Má být zajištěna kooperace ve smyslu výše popsaném. Všichni uživatelé mají vidět výsledek právě provedených změn, aniž by museli znovu načíst aplikaci. Uzly aplikace se mají repozicovat tak, aby výsledný diagram byl přehledný. Uzly mají být jednoznačně pospojovány, nejlépe pomocí Beziérových křivek. Aplikace má být snadno ovladatelná, nejlépe pomocí klávesnice.

V této práci bych si chtěl vyzkoušet obousměrnou komunikaci mezi serverem a klientem. Naučit se pracovat s technologií Jakarta EE a také se naučit vykreslovat grafický výstup v podobě diagramu a měnit ho v závislosti na změně dat.

Kapitola 2
Analýza

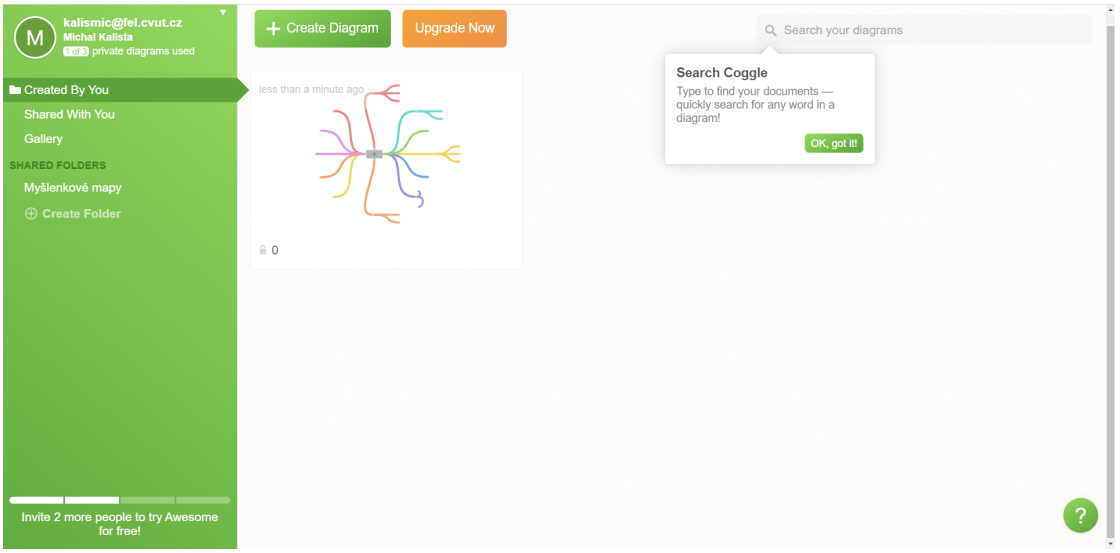
2.1 Existující řešení

V této kapitole jsou uvedeny již dostupné konkurenční nástroje, které zpracovávají tuto funkcionalitu. Každý nástroj je zde představen jako celek, následně se zaměřuji na požadovanou funkcionalitu. Pro každý nástroj jsou uvedeny výhody a nevýhody tohoto řešení. Všechny aplikace, které jsou zde uváděny, jsou online editory, které fungují pouze v prohlížeči a k jejich používání není nutné nic instalovat.

2.1.1 Coggle

Nástroj *Coggle*, který je dostupný na adrese coggle.it[1], se zaměřuje na vytváření myšlenkových map a vývojových diagramů [4].

Registrace do nástroje *Coggle* je zdarma. Autentizaci lze uskutečnit pomocí Google, Office 365, Apple nebo zadáním zaregistrovaného emailu. Po přihlášení se uživatel přesune na domovskou stránku, odkud lze vytvářet nové diagramy, zvát další uživatele ke sdílení a tvořit sdílené složky. Ukázka domovské stránky je vidět na obrázku 2.1.



Obrázek 2.1. Ukázka výchozí stránky nástroje *Coggle*

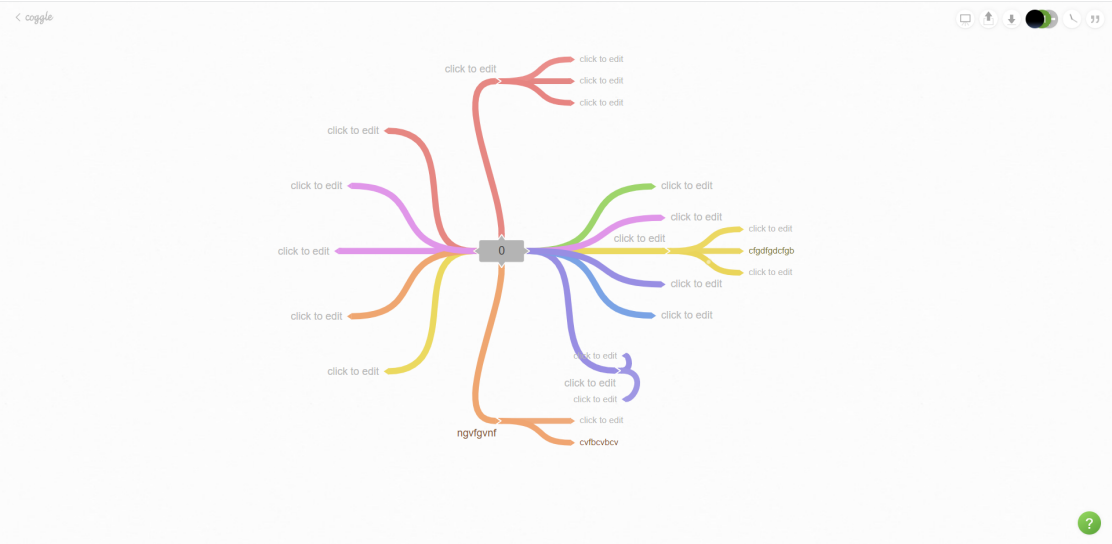
Uživatel, který používá tento nástroj ve verzi zdarma, je omezen počtem úkonů, které může udělat. Může například udělat pouze tři osobní diagramy, přičemž za osobní diagram se pokládá i diagram sdílený s jiným uživatelem. Počet veřejných diagramů omezený není. Uživatel může své diagramy sdílet s ostatními uživateli, a to prostřednictvím připojení uživatele pomocí emailové adresy, nebo prostřednictvím odkazu na mapu. Připojením pomocí odkazu se uživatel dostane pouze do režimu pozorovatele.

[1] `https://coggle.it`

V rámci nástroje *Coggle* může uživatel členit diagramy do sdílených složek. Složky lze vytvářet, editovat, mazat a přesouvat mezi nimi diagramy. Počet složek není omezený [4].

V rámci myšlenkové mapy může uživatel uzly vytvářet, editovat a mazat. *Coggle* znázorňuje hlavně cesty grafu. Uzly jsou znázorněny jen jako textový popisek. Odlišné znázornění uzlů je nutno nastavit. Oproti jiným editorům, lze v *Coggle* cesty spojovat a tvořit cykly. Do myšlenkové mapy je možné vkládat obrázky ve formátu jpg, png a gif a soubory mm a txt. Soubory jsou do diagramu vkládány jako uzly. Nástroj umožňuje vkládat do uzlů mnoho druhů emotikonů. Diagramy lze také exportovat ve formátu pdf, png, txt, vsdx a mm souboru [5].

Nástroj *Coggle* se snaží zaujmout svým nevšedním návrhem. Vkládané cesty jsou široké, velmi barevné a často nevšedně zvlněné. To není vždy úplně výhoda. Výsledné prostředí působí trochu nepřehledně. Kolem diagramu nejsou téměř žádné ikony, a proto je pro nového uživatele těžké s prostředím začít flexibilně pracovat. Veškeré ovládání uzlů a cest se objeví až při kliknutí na graf v kruhovém okolí myši. Prostředí pro tvorbu myšlenkových map je vidět na obrázku 2.2.



Obrázek 2.2. Ukázka prostředí pro tvorbu mapy v nástroji *Coggle*

Coggle poskytuje značnou míru kooperace. Všichni členové, se kterými je mapa sdílena pomocí emailu, mohou vytvářet a upravovat uzly a cesty. Jeden uzel nemůžou současně upravovat dva uživatelé. Dokud je uzel v režimu úpravy, ostatní uživatelé nevidí, co se na uzlu mění. Nově přidané uzly se zobrazují ostatním uživatelům zcela plynně. Mapa se při přidávání a ubírání uzlů nepřeškálovává. Za nedostatek lze považovat nedůsledné ošetření odebírání větví grafů, které jsou právě editovány. Tedy pokud jeden uživatel edituje uzel větve, může jiný uživatel v současné chvíli celou větev smazat a tím zrušit celou práci druhého uživatele, kterému zmizí větev doslova pod rukama.

Nadstandardně *Coggle* zdarma poskytuje uživatelům funkci chatu a revizi všech změn, ke které je možné se vrátit v podobě nového diagramu, pokud ji chce uživatel upravovat. Historie chatu je však v bezplatné verzi omezena. Stejně tak i poskytování odkazu, kterým by se dal diagram upravovat i bez přihlášení.

Vykreslování diagramů je v aplikaci *Coggle* realizováno pomocí technologie SVG.

Výhody:

- Snadná spolupráce
- Historie revizí
- Chat
- Možnost cyklických grafů

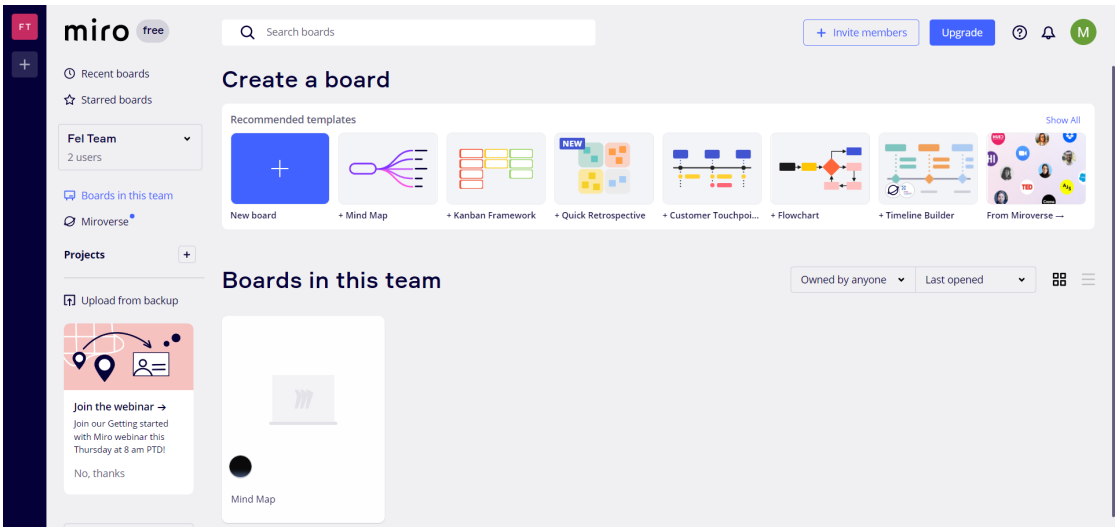
Nevýhody:

- Nestandardní design, který stěžuje uživatelům práci
- V bezplatné verzi vytvoření pouze tří soukromých map
- Nedokonalé ošetření práce více uživatelů se stejnou větví diagramu
- Nemožnost přidávat poznámky do grafu

2.1.2 Miro

Miro je rozsáhlá platforma pro spolupráci týmu, ale hodí se i pro osobní užití při plánování a organizaci. Vzniklo původně jako nástroj návrhářské agentury pro lepší komunikaci se zákazníkem, který se nemůže přímo zúčastnit diskuse nad produktem. Proto je také navrženo jako rozsáhlá interaktivní nástěnka [6–7].

Miro je dostupné na adrese miro.com[2]. Registrace do nástroje *Miro* je zdarma. Autentizaci lze provést skrze Google, Office 365, Slack nebo zadáním zaregistrovaného emailu. Po přihlášení do aplikace se uživateli zobrazí domovská stránka, odkud může vytvářet nové nástěnky, projekty, zvát další uživatele ke sdílení, vytvářet týmy a vybírat z předem připravených šablon pro tvorbu nástěnek. Ukázka výchozího zobrazení nástroje *Miro* je vidět na obrázku 2.3.



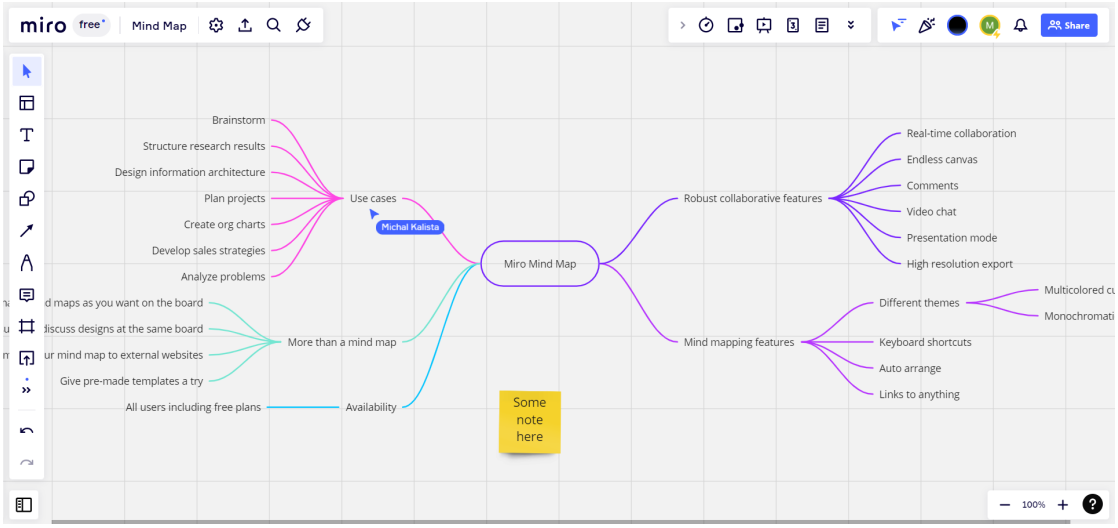
Obrázek 2.3. Ukázka výchozí stránky nástroje *Miro*

Pro uživatele, který si neplatí předplatné pro tento nástroj, je výběr značně omezený. Takový uživatel může vytvořit jen tři nástěnky, které může editovat. Pokud bude přizván ke spolupráci na další nástěnce, nebude tuto nástěnku moci upravovat. Ve volné verzi jsou pro uživatele předpřipraveny šablony pro nástěnky, uživatel nemá možnost vytvářet šablony nové nebo stávající šablony upravovat. Uživatel má možnost v základní verzi vytvořit jeden tým, se kterým bude nástěnku sdílet. Tento tým může mít zdarma pouze

[2] `https://miro.com`

dva členy. Uživatel po celou dobu pracuje v režimu sdílení s týmem. Pro tento tým nelze vytvořit projekt, který seskupuje nástěnky do jednoho celku [8].

V rámci nástěnky má uživatel možnost vytvářet, upravovat a mazat uzly myšlenkové mapy, nebo jiné, předem šablonou připravené, mapy. Uzly a stromy je také možné duplikovat. Do nástěnky lze vkládat poznámky, základní tvary, kreslit, či přidávat emotikony. Dále *Miro* umožňuje do nástěnky vkládat obrázky nebo dokumenty. V základní verzi je možné nástěnku exportovat jako obrázek ve formátu jpg a pdf, ale pouze v omezené kvalitě, která pro obrázek ve formátu jpg činí 1530 x 1020 pixelů. Obsah mapy lze vyexportovat jako csv soubor. Prostředí pro tvorbu myšlenkových map je vidět na obrázku 2.4.



Obrázek 2.4. Ukázka prostředí pro tvorbu mapy v nástroji *Miro*

Narozdíl od jiných nástrojů, *Miro* poskytuje vysokou míru spolupráce nad tvorbou mapy. Uzly mohou vytvářet a upravovat oba dva členové týmu. Jeden uzel nelze upravovat dvěma uživateli zároveň. Změny se promítají přímo do zobrazení nástěnky druhého uživatele se zpožděním úměrným trvání síťové komunikace. Mapa je stabilizovaná, po přidání uzlu jiným uživatelem zůstane upravovaný uzel na stejném místě. Nelze smazat uzel, jehož potomek je právě upravován. *Miro* umožňuje zapnout i zobrazování pohybu myši druhého uživatele.

Miro poskytuje integraci s nástroji jako je Zoom, Slack, Microsoft Teams, Trello Zapier, Google Drive, Microsoft One Drive, Dropbox, Box, Sketch, Unsplash a další. Bezplatně je také poskytováno REST rozhraní [8].

Ve verzi zdarma *Miro* poskytuje historii úprav.

V porovnání s ostatními nástroji, *Miro* má mapu vytvořenou pomocí technologie canvas.

Výhody:

- Jednoduché prostředí
- Snadná spolupráce
- Integrace s množstvím jiných nástrojů
- Možnost vkládání poznámek a komentářů
- Revize historie

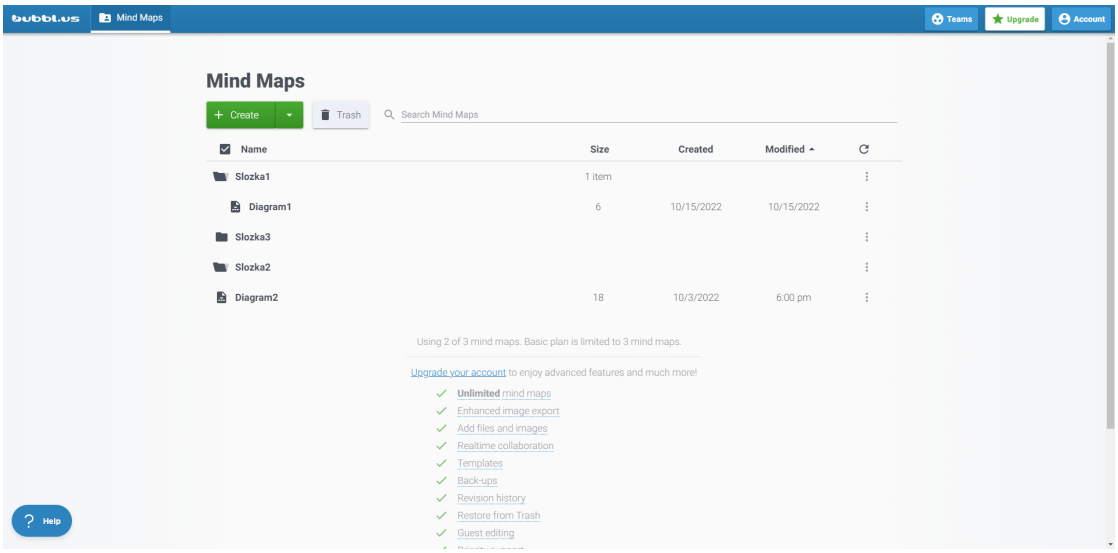
Nevýhody:

- V bezplatné verzi pouze vytvoření jednoho týmu
- V bezplatné verzi přiřazení pouze dvou členů do jednoho týmu
- V bezplatné verzi pouze vytvoření třech nástěnek

2.1.3 Bubbl.us

Nástroj *Bubbl.us* je dostupný na stejnojmenné adrese[3]. Svou funkcionalitou se zaměřuje na vytváření myšlenkových map, vývojových diagramů a jiné znázorňování myšlenkových postupů.

Do nástroje *Bubbl.us* se lze zdarma zaregistrovat. Přihlášení probíhá zadáním emailu a hesla nebo prostřednictvím Facebooku a Googlu. Po přihlášení se uživatel ocitne na domovské stránce, odkud může vytvářet nové diagramy, vybírat ze šablon, vytvářet složky diagramů a spravovat týmy. Po celou dobu práce v nástroji *Bubbl.us* je přítomna horní lišta, která dovoluje snadno přepínat mezi otevřenými diagramy a týmy v rámci jedné karty prohlížeče. Ukázka domovské stránky nástroje *Bubbl.us* je vidět na obrázku 2.5.



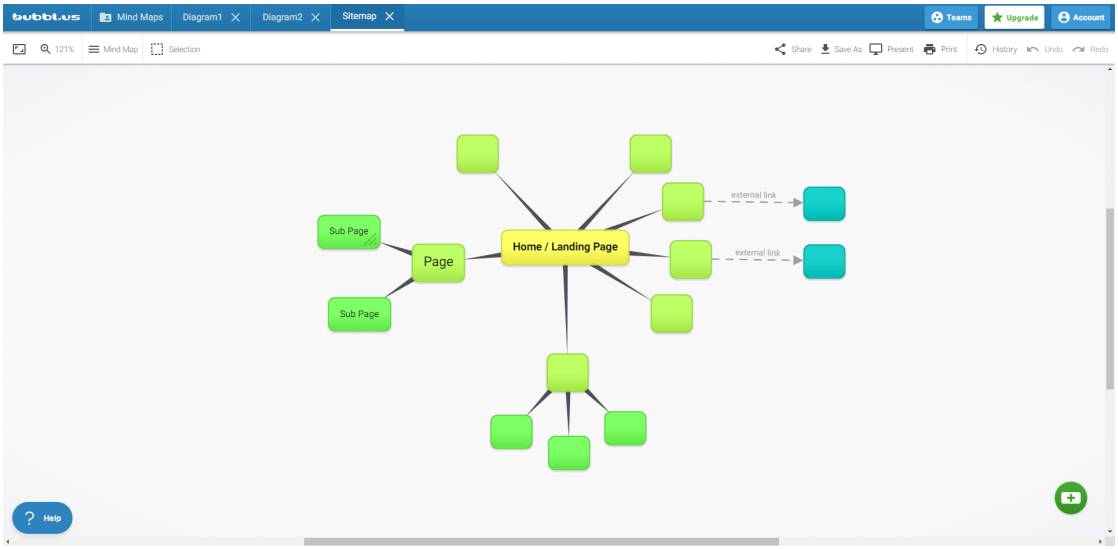
Obrázek 2.5. Ukázka výchozí stránky nástroje *Bubbl.us*

Použití nástroje *Bubbl.us* ve verzi zdarma je značně omezeno. Uživatel může vytvořit pouze tři diagramy. Své diagramy může uživatel sdílet prostřednictvím odkazu [9]. Každý takový diagram si může zobrazit kdokoli, kdo má odkaz. Zobrazení diagramu pomocí odkazu není omezeno nutností přihlásit se do nástroje *Bubbl.us*. Diagram nelze pomocí odkazu upravovat, je jen ve verzi pro čtení. Zobrazený diagram se chová pouze jako statický obrázek. Pokud majitel diagramu diagram upravuje, pozorovatelům se výsledek projeví až po opětovném načtení stránky. Diagramy lze v prostředí hlavní stránky dělit do složek, jejichž počet není omezen. Složky lze vytvářet, přejmenovávat a mazat. Správa týmů však není pro prostředí zdarma umožněna.

V prostředí diagramu lze uzly vytvářet, editovat a mazat. Myšlenkovou mapu lze exportovat ve formátu jpg, png, html a txt, případně vytisknout do pdf [10]. Do mapy lze vložit jeden soubor prostřednictvím uzlu mapy. Pokud je souborem obrázek, zobrazí se v uzlu, pokud jde o jiný soubor, zobrazí se jen odkaz na stažení. Vkládání dalších souborů či vkládání souborů mimo prostředí uzlu je umožněno jen pro placené verze.

[3] `https://bubbl.us/`

Do uzlu lze vložit emotikon. V základní verzi si uživatel musí vybrat pouze z pěti symbolů. Ke každému uzlu lze vložit poznámku, která se zobrazí po najetí myši. Prostředí myšlenkové mapy je vidět na obrázku 2.6.



Obrázek 2.6. Ukázka prostředí pro tvorbu mapy v nástroji *Bubbl.us*

Bubbl.us neposkytuje v základní verzi kooperaci. Kooperace je povolena až v placených verzích. Stejně tak neomezený počet vytvářených map, přidávání souborů, revize historie, funkce hosta, správa týmů či vylepšená možnost exportu obrázku diagramu [9].

Pro vykreslování diagramů je částečně využíváno technologie SVG. Tato technologie je však používána pouze pro vykreslování cest v grafu, uzly jsou vykreslovány pomocí `<div>` HTML elementu.

Výhody:

- Jednoduché prostředí
- Možnost snadného přepínání mezi grafy v rámci jedné karty prohlížeče
- Možnost vkládání poznámek

Nevýhody:

- V bezplatné verzi chybí možnost kooperace
- V bezplatné verzi pouze sdílení statického obrázku grafu
- V bezplatné verzi vložení pouze jednoho souboru
- V bezplatné verzi pouze vytvoření třech map
- V bezplatné verzi velmi omezená množina emotikonů

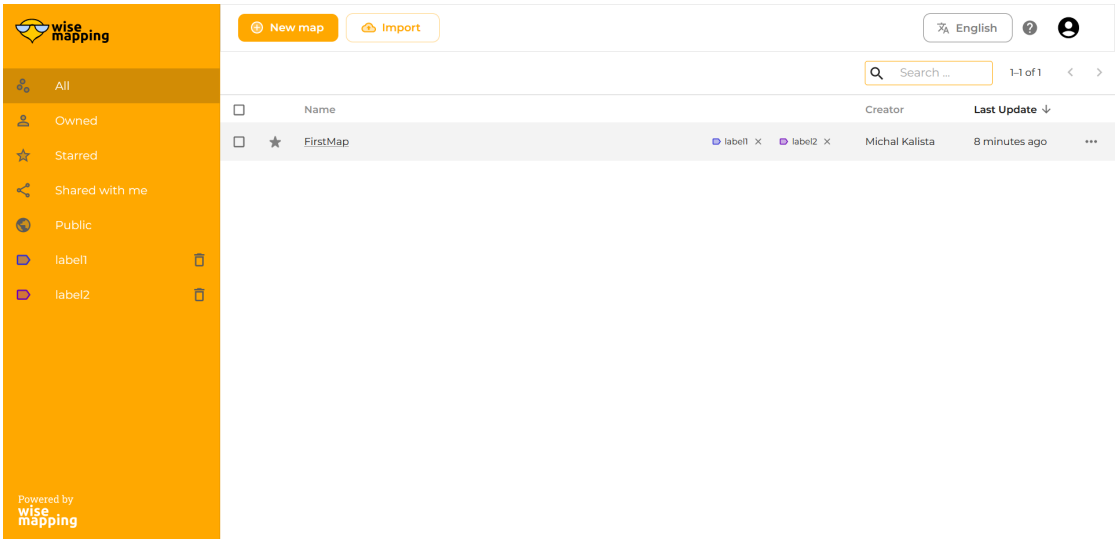
2.1.4 WiseMapping

Jedná se o open-source software dostupný na adrese www.wisemapping.com[4]. Tento nástroj je navržen na tvorbu myšlenkových map. Všechny jeho funkce jsou poskytovány zdarma [11].

Registrace lze provádět pouze prostřednictvím zadání emailu a hesla. *WiseMapping* neposkytuje registraci skrze účty jiných nástrojů pomocí protokolu OAuth 2.0.

[4] `https://www.wisemapping.com/`

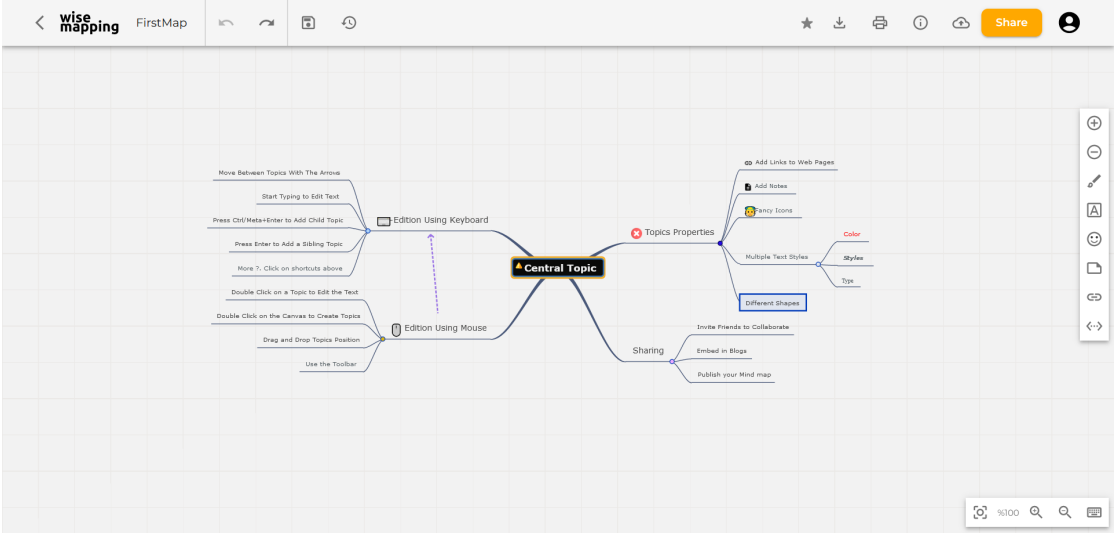
Po přihlášení je uživateli zobrazena domovská stránka. Z domovské stránky lze vytvářet diagramy nové, sdílet diagramy s ostatními uživateli, publikovat diagramy, či je označovat štítky pro lepší možnost filtrování. Mapy je možné importovat v podobě uložených souborů nástrojů *WiseMapping* a *Freemind* [12]. Ukázka domovské stránky nástroje *WiseMapping* je na obrázku 2.7.



Obrázek 2.7. Ukázka výchozí stránky nástroje *WiseMapping*

Narozdíl od ostatních zde uváděných nástrojů *WiseMapping* není omezen počtem úkonů. Každý uživatel může vytvořit libovolný počet map a označit je libovolným počtem štítků. Každý diagram jde sdílet s libovolným počtem uživatelů nástroje *WiseMapping* prostřednictvím zadání emailu. Pro sdílení je možné si vybrat dvě role uživatele. Roli editora, s kterou může uživatel diagram upravovat, a diváka, který diagram upravovat nemůže. Změny, provedené jinými uživateli se projeví až po novém načtení stránky. Uživatelé mohou také své mapy publikovat. Publikované mapy jsou dostupné pomocí odkaz pro všechny. Mapy spuštěné pomocí odkazu nelze upravovat Ke každé mapě lze také přidat popis.

V prostředí diagramu nástroje *WiseMapping* lze uzly vytvářet, mazat a editovat. Ke každému uzlu lze vložit poznámku. Soubory do diagramu není možné vkládat. Je umožněno vkládat ke každému uzlu předdefinované emotikony. Výslednou myšlenkovou mapu lze vyexportovat v obrázkovém formátu SVG, JPEG, či png, nebo v textovém formátu txt, md, nebo jako mapu nástroje *WiseMapping* (wxml) či *Freemind* 1.0.1 (mm). V neposlední řadě lze mapu vytisknout [12]. Prostředí nástroje *WiseMapping* je vidět na obrázku 2.8.



Obrázek 2.8. Ukázka prostředí pro tvorbu mapy v nástroji *WiseMapping*

Nástroj *WiseMapping* neposkytuje možnost kooperace v reálném čase. Změny ostatních uživatelů se v mapě projeví až po novém načtení stránky. Současně na mapě může pracovat pouze jeden uživatel. Ostatním uživatelům se v této chvíli jeví mapa jako uživatelům v roli diváka. Jako nadstandardní funkci poskytuje *WiseMapping* revizi historie.

K vykreslování mapy používá *WiseMapping* technologii SVG.

Výhody:

- Jednoduché a přehledné prostředí
- Možnost revize historie
- Možnost vkládání poznámek
- Zdarma bez omezení
- Možnost snadného sdílení a nastavování práv
- Možnost štítkování a filtrování map

Nevýhody:

- Chybí možnost kooperace
- Nelze vkládat soubory
- V případě, že někdo s mapou pracuje, mají ostatní uživatelé úpravu zakázánu

2.2 Shrnutí

Z předchozí analýzy vychází několik poznatků pro mou práci.

- V dostupných nástrojích z velké části chybí možnost kooperace více uživatelů zároveň.
- Ze zbývajících nástrojů, pokud nástroj kooperaci umožňuje, je buď zpoplatněna, nebo je značně omezena nejčastěji počtem map.
- Nástroje se často snaží zaujmout svým nestandardním stylováním diagramu. Toto stylování ale znepřehlední prostředí diagramu. Průchod aplikací je poté složitý a ovládání není očividné.
- Pravidelní uživatelé ocení spíše přehlednou aplikaci s plně funkční základní funkcionalitou, než přeplněnou aplikaci s nestandardními a často i nesouvisejícími funkcemi.

- Výhodou všech porovnávaných nástrojů je snadné ovládání prostřednictvím klávesnice i prostřednictvím myši.
- Všechny aplikace umožňují exportování do základních grafických souborů. U některých je omezena velikost při bezplatné verzi.
- Mezi velmi užitečné, avšak zcela nadstandardní funkcionality, které práci s nástroji usnadňují, patří například funkce chatu, revize historie či importování mapy. Tuto možnost však většina nástrojů poskytuje až v placených verzích.
- Možnost vkládání poznámek do mapy, či emotikonů do uzlu je výhodou, kterou poskytuje většina nástrojů. Stejně tak i barevné označení uzlu.
- Velká většina porovnávaných řešení vykresluje diagram pomocí technologie SVG. Použil jsem ji také.
- Všechny aplikace umožňují základní práci s uzlem mapy jako je vytváření uzlů nových, úprava obsahu uzlu a mazání uzlů. Některé nadstandardně pak kopírování, přesouvání a vkládání uzlů a sbalování podstromů.
- Pro práci s mapou je vhodné členění, štítkování, nebo filtrování a popis. Tuto funkcionalitu poskytují v omezené míře všechny porovnávané nástroje. Některé nástroje tuto funkcionalitu nabízí hlavně ve verzi s předplatným.

Kapitola 3
Návrh

3.1 Systémové požadavky

3.1.1 Funkční požadavky

Pro správné stanovení funkcionality vytvářené aplikace je nutné stanovit požadavky funkční. Při stanovování funkčních požadavků jsem vycházel z kapitoly 2 a z vlastní zkušenosti. Některé funkční požadavky nejsou v aplikaci zahrnuty, protože byli vyhodnoceny jako rozšiřující funkcionalita. Jedná se o požadavky s označením FR08, FR12, FR13, FR14 a FR15.

FR01 – Přihlášení
Systém umožní uživateli přihlásit se ke svému účtu.

FR02 – Zobrazení seznamu vlastních map
Systém zobrazí uživateli seznam vlastních map.

FR03 – Zobrazení seznamu sdílených map
Systém zobrazí uživateli seznam map, které jsou s uživatelem sdílené.

FR04 – Vytváření map
Systém umožní uživateli vytvářet mapy nové.

FR05 – Mazání map
Systém umožní uživateli mazat stávající vlastní mapy.

FR06 – Sdílení map
Systém umožní uživateli sdílet mapy vlastní.

FR07 – Úprava map
Systém umožní uživateli upravovat mapy vlastní a mapy sdílené. Mezi úpravy patří úprava názvu mapy a struktury mapy.

FR08 – Nastavení přístupových práv
Systém umožní uživateli upravovat přístupová práva map vlastních a map sdílených, pro která má nastavena práva pro sdílení.

FR09 – Tvorba uzlů
Systém umožní uživateli vytvářet uzly nové v rámci vlastní mapy nebo mapy sdílené. Uživatel může vytvářet uzly potomka aktivního uzlu, nebo uzly sourozenecké. Sourozenecké uzly nelze vytvářet z pozice prvního kořenového uzlu.

FR10 – Mazání uzlů
Systém umožní uživateli mazat uzly z map vlastních a map sdílených.

FR11 – Úprava uzlů
Systém umožní uživateli upravovat obsah uzlů map vlastních a map sdílených.

FR12 – Kopírování a vkládání uzlů
Systém umožní uživateli kopírovat a vkládat uzly v mapách vlastních a mapách sdílených, pro která má nastavena práva pro úpravu.

FR13 – Sbalování a rozbalování podstromů mapy
Systém umožní uživateli sbalovat a rozbalovat podstromy v mapách vlastních a mapách sdílených, pro která má nastavena práva pro úpravu.

FR14 – Importování souborů
Systém umožní uživateli importovat základní obrazové soubory do map vlastních a map sdílených, pro která má nastavena práva pro úpravu.

FR15 – Exportování souborů
Systém umožní uživateli exportovat mapy vlastní a mapy sdílené do základních obrazových souborů.

3.1.2 Kvalitativní a kvantitativní požadavky

Kromě funkčních požadavků jsem pro aplikaci stanovil i požadavky kvantitativní a kvalitativní. Při stanovování kvalitativních a kvantitativních požadavků jsem vycházel z kapitoly 2 a z vlastní zkušenosti.

NFR01 – Přehlednost
Myšlenkové mapy jsou vytvářené pro zvýšení přehlednosti, proto musí být výsledná aplikace přehledná.

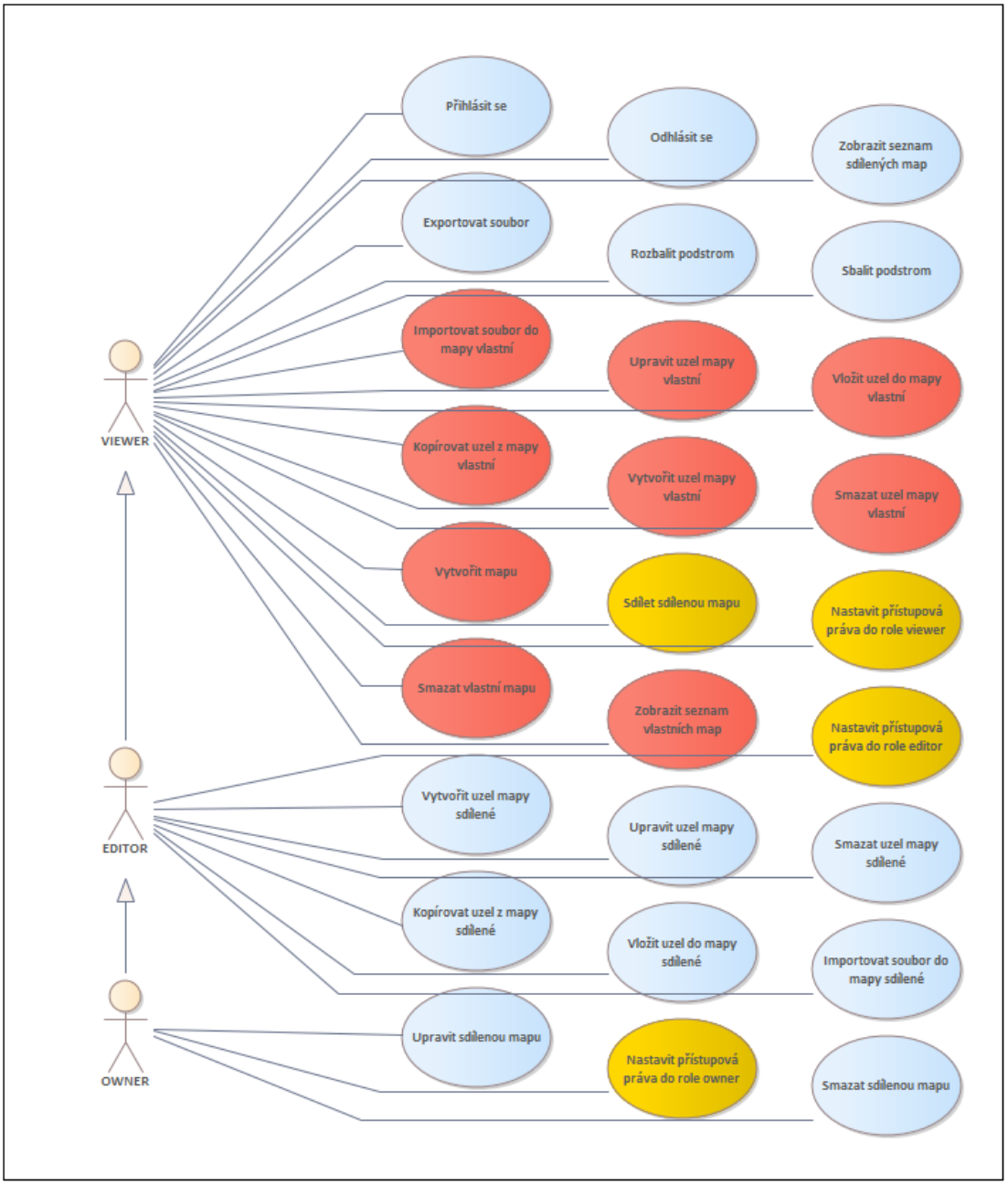
NFR02 – Snadná ovladatelnost
Nesnadné ovládání tvorby mapy odradí uživatele od používání aplikace. Aplikace musí umožňovat ovládání mapy skrze události myši i skrze události klávesnice.

NFR03 – Konzistence
Pro zajištění kvalitní týmové spolupráce je nezbytné, aby každý uživatel viděl mapu ve shodné podobě.

NFR04 – Dostupnost
Aby se mohli uživatelé aplikace spolehnout, že se ke svým mapám vždy dostanou, je potřeba zajistit vysokou dostupnost aplikace.

NFR05 – Výkon
Aplikaci je také potřeba zajistit proti většímu náporu uživatelů.

3.2 Případy užití



Obrázek 3.1. Diagram případů užití.

Pro přehlednost jsou na obrázku 3.1 uváděny i případy užití odvozené z funkčních požadavků. Jsou zde definovány tři role. Role jsou zde definovány z pohledu přístupu ke sdíleným mapám na role Viewer, Editor a Owner. Mapami sdílenými jsou zde myšleny mapy, ke kterým má uživatel nastaven přístup skrze entitu MapRights. Mezi mapy sdílené se nepočítají mapy vlastní, tedy takové mapy, které uživatel vytvořil. Případy užití jsou dále ovlivněny proměnnou *canShared*, která upravuje, zda může uživatel sdílenou mapu dále sdílet a proměnnou *canMapCreate*, která upravuje, zda může uživatel vytvářet mapy vlastní a operovat s nimi. Případy užití podbarvené červenou barvou mají

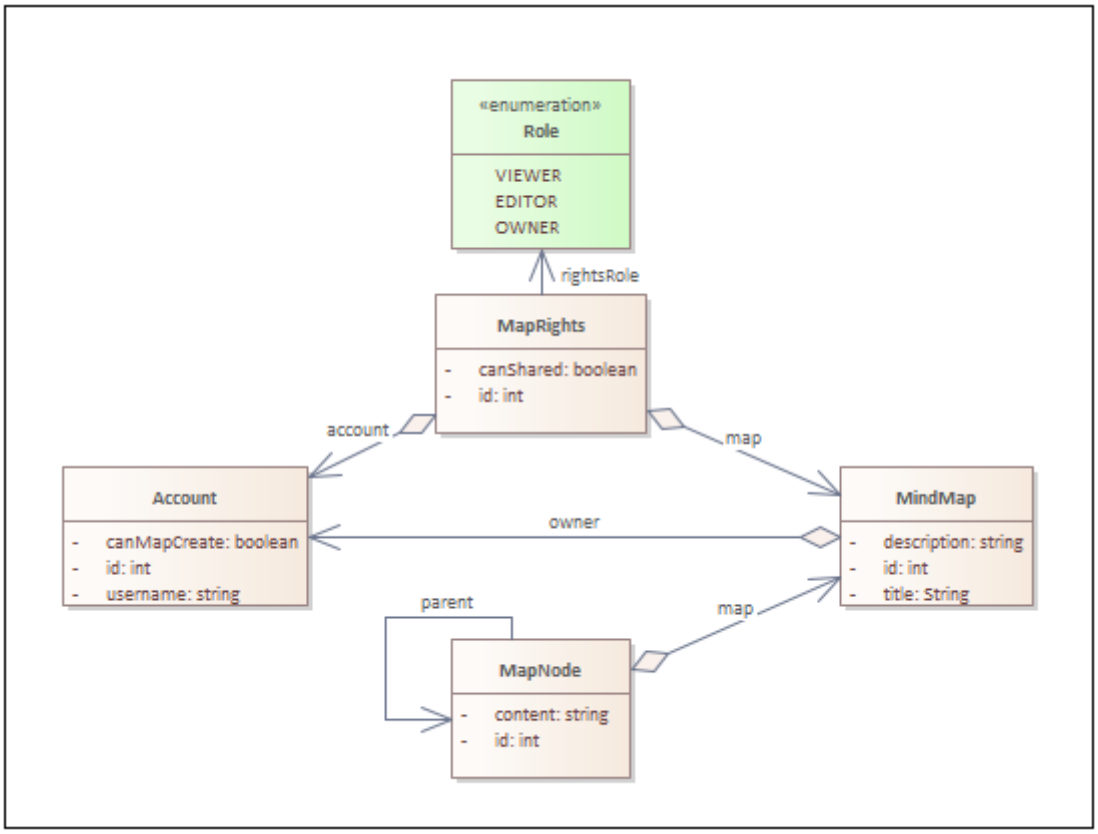
nastavenou proměnnou *canMapCreate* na hodnotu *true*. Případy podbarvené žlutou barvou mají nastavenou hodnotu *canShared* na *true*.

Uživatelské role nejsou ve vytvořené aplikaci používány k úpravě přístupových práv. Všichni uživatelé mají přístupová práva shodná. Stejně jako u funkčních požadavků v kapitole 3.1.1, nebyly všechny případy užití implementovány. Mezi nadstandardní nezahrnuté funkcionality patří případy užití: *Exportovat soubor, Rozbalit podstrom, Sbalit podstrom, Importovat soubor do mapy vlastní, Kopírovat uzel z mapy vlastní, Vložit uzel do mapy vlastní, Sdílet sdílenou mapu, Nastavit přístupová práva do role viewer, editor a owner, Kopírovat uzel z mapy sdílené, Vložit uzel do mapy sdílené a Importovat soubor do mapy sdílené, Smazat sdílenou mapu.*

3.3 Datový model

Datový model, který je vidět na obrázku 3.2, není složitý. Zde je datový model zapsán pomocí UML notace.

- *MindMap* – zastupuje entitu diagramu spolu s jeho vlastnostmi, odkazuje na účet vlastníka
- *MapNode* – zastupuje každý jednotlivý uzel s odkazem na svého rodiče a na mapu, v níž se nachází.
- *Account* – zastupuje uživatelský účet
- *MapRights* – zprostředkovává přístup k mapám, které jsou s uživatelem sdílené odkazováním na entitu účtu a entitu mapy, upravuje práva přístupu k sdíleným mapám



Obrázek 3.2. Diagram datového modelu.

3.4 Použité technologie

3.4.1 Jakarta EE

Jakarta EE je souhrn specifikací a standardů pro vývoj business aplikací v programovacím jazyce Java. Jakarta EE vznikla přejmenováním specifikace Java EE 8 při přechodu technologie Java EE od firmy Oracle pod Eclipse Foundation. Jakarta EE rozšiřuje sadu specifikací Java SE o komponenty umožňující snazší vývoj enterprise aplikací [13–14].

Mezi sadu specifikací použitých v této práci patří například:

- Jakarta Servlet
  - Definuje rozhraní pro dynamické obsluhování HTTP požadavků a tvorbu odpovědí obvykle pro HTML stránky.
  - V mé práci je pomocí servletu vykreslováno uživatelské rozhraní.
- Jakarta WebSocket
  - Rozhraní pro plně duplexní obousměrnou komunikaci mezi klientskou a serverovou částí aplikace s nízkou latencí.
  - V mé práci je websocket použit ke komunikaci při změnách v diagramu.
- Jakarta RESTful Web Services
  - Rozhraní pro vytváření web services, které reagují na HTTP metody a zachovávají principy REST.
  - Rozhraní je v mé práci provoláváno téměř s každou změnou aplikace, kterou je nutné uložit do databáze.
- Jakarta Persistence
  - Specifikace pro objektově-relační mapování mezi tabulkami relační databáze a třídami Jakarty.
  - V mé práci je použito rozhraní Jakarta Persistence konkrétně v implementaci EclipseLink.
- Jakarta Transaction
  - Rozhraní pro definování a správu transakcí.
- Jakarta Enterprise Beans
  - Rozhraní komponent zapouzdřujících business logiku aplikace [14–16].

Jakarta EE byla vybrána pro tuto práci již v zadání. Důvodem je zejména vysoká stabilita a zpětná kompatibilita. Konkrétně byla využita Jakarta EE ve verzi 8. Verze byla zvolena s ohledem na podporu Jakarty EE 8 na zvoleném aplikačním serveru [17].

3.4.2 Payara Server

Jako aplikační server byl zvolen Payara server. Payara server je odvozen z aplikačního serveru GlassFish open source edition. Hlavním důvodem, proč byl zvolen server Payara, je otevřený zdrojový kód, poskytování serveru zdarma a více časté a více pravidelné zveřejňování nových verzí než u serveru GlassFish [18]. Payara server je zde použit ve verzi 5.2022.3.

3.4.3 PostgreSQL

Jako úložiště byl vybrán relační databázový systém PostgreSQL. PostgreSQL je rozšířením jazyka SQL. Jedná se o systém s otevřeným zdrojovým kódem. To byl také jeden z důvodů, proč byl vybrán právě PostgreSQL. Dále také z důvodu dobré podpory ze strany Jakarta EE a robustnosti systému [19].

3.4.4 SVG

Pro vykreslování diagramů jsem využil technologii SVG. Tato technologie byla zvolena v návaznosti na kapitolu 2. Velká část dostupných nástrojů používá pro vykreslování právě technologii SVG. SVG je značkovací jazyk založený na XML. Jedná se o formát vektorový, který je přímo začleněn mezi HTML do webové stránky [20].

Konkurenční technologií k SVG je Canvas. Oproti SVG má ale v tomto případě nevýhody. Canvas je rastrový formát, který nelze stylovat pomocí CSS. V případě malého počtu objektů a velkých ploch je Canvas méně výkonný než SVG [21]. Zde vykreslovaný obsah obsahuje vektorové obrazce, které zachovávají předem dané stylování. Výsledný diagram navíc s velkou pravděpodobností nebude obsahovat velké množství objektů.

3.4.5 Technologie uživatelského rozhraní

Protože aplikace není rozsáhlá, nebylo využito žádného frameworku pro tvorbu uživatelského rozhraní. Nejtěžší částí uživatelského rozhraní je vykreslování diagramu pomocí SVG, které je popsáno výše. Použití frameworku by v tomto případě vykreslování nezjednodušilo. Pro interakci na frontendu byl využit Javascript také z důvodu vysoké rychlosti a nízké zátěže pro server [22]. Pro generovaní HTML byl využit Jakarta Servlet. Jakarta Servlet byl upřednostněn před JSF a JSP kvůli jednoduchosti a vysoké výkonnosti.

3.4.6 Docker

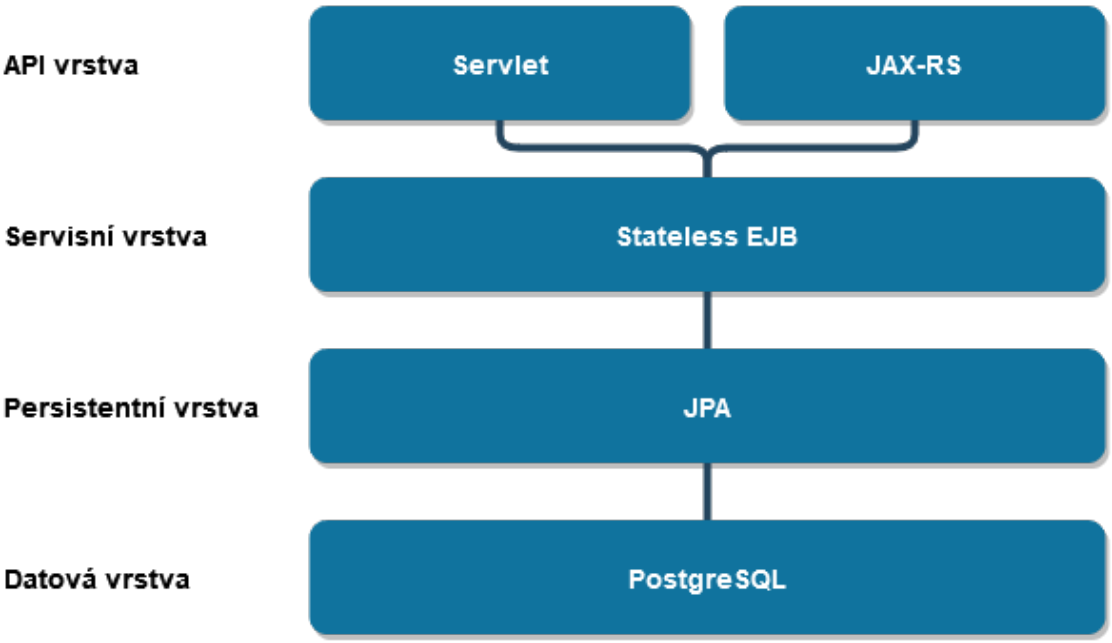
Docker je open-source nástroj pro snadné nasazování aplikací pomocí kontejnerů. Kontejnerizace přináší abstrakci od operačního systému, na kterém daný kontejner běží. Výsledný kontejner proto může běžet na jakémkoli stroji, který Docker umožňuje. Na rozdíl od virtuálních strojů, kontejnery pro svůj běh využívají původní operační systém, a proto spotřebovávají daleko méně hardwarových zdrojů než virtuální stroje [23–24].

V mé aplikaci jsou pro nasazení do produkčního prostředí vytvořeny dva Docker kontejnery. Konkrétně kontejner pro Payara server, na kterém běží samotná aplikace a kontejner pro PostgreSQL databázi. Pro spouštění kontejnerů současně se správným nastavením je využito Docker Compose.

Kapitola 4
Implementace

Aplikace je rozdělena do vrstev. Vrstvy architektury backendové části aplikace jsou vidět na obrázku 4.1. Spodní vrstvu tvoří databáze typu PostgreSQL. Pro backend aplikace je podle zadání použita Jakarta EE. Nad databází je vrstva JPA konkrétně v implementaci EclipseLink. Vrstva JPA slouží k mapování objektů na tabulkové záznamy v databázi. Nad JPA lze nalézt servisní beany sloužící jako logická jednotka backendové části aplikace. Nad nimi se nachází REST rozhraní pro zajištění komunikace s klientem a Servlety, které generují html data pro uživatele.

Na klientu běží Javascript, který zpracovává data ze serveru a obsluhuje interakce uživatele. Klient posílá požadavky na server a provolává REST rozhraní. Na základě odpovědi Javascript překresluje klienta. Pro překreslování diagramu je komunikace mezi serverem a klientem v této části aplikace zajištěna protokolem WebSocket. Zde se jedná o rozhraní Jakarta WebSocket. WebSocket zajišťuje obousměrnou komunikaci. Na klientské straně je zajištěno překreslování SVG elementů pomocí Javascriptu na základě dat přijatých WebSocketem.



Obrázek 4.1. Architektrura aplikace.

4.1 REST rozhraní

V této sekci je uvedeno dostupné REST rozhraní pro aplikaci, rozdělené podle entit datového modelu, ke kterým se jednotlivé požadavky váží.

4.1.1 Účet

- Získání objektu účtu podle jména uživatele
  - HTTP metoda: `GET`
  - Cesta: `/resources/account/username/{username}/`
  - Metoda vrací objekt účtu na základě uživatelského jména definovaného v proměnné `username`.
- Vytvoření účtu
  - HTTP metoda: `POST`
  - Cesta: `/resources/account/`
  - Metoda vytváří nový účet, objekt účtu je posílán v jazyce JSON v těle metody. Struktura JSON objektu je shodná s entitou Account v kapitole 3.3. Zpět se vrací objekt nově vytvořeného účtu.
- Úprava účtu
  - HTTP metoda: `PUT`
  - Cesta: `/resources/account/`
  - Metoda upravuje proměnnou canMapShared.
- Přihlášení
  - HTTP metoda: `POST`
  - Cesta: `/resources/account/login/`
  - Metoda přihlašuje uživatele na základě entity poslané v těle požadavku. Poslaná entita obsahuje uživatelské jméno a heslo.
- Odhlášení
  - HTTP metoda: `GET`
  - Cesta: `/resources/account/logout/`
  - Metoda odhlašuje přihlášeného uživatele.
- Me
  - HTTP metoda: `GET`
  - Cesta: `/resources/account/me/`
  - Metoda vrací objekt obsahující informace o právě přihlášeném uživateli. Objekt obsahuje jméno uživatele a příslušnost k uživatelským rolím.

4.1.2 Mapa

- Vytvoření mapy
  - HTTP metoda: `POST`
  - Cesta: `/resources/map/{username}`
  - Metoda vytváří novou mapu, objekt mapy je posílán v jazyce JSON v těle metody. Struktura JSON objektu je shodná s entitou MindMap v kapitole 3.3.
- Úprava názvu mapy
  - HTTP metoda: `PUT`
  - Cesta: `/resources/map/`
  - Metoda upravuje název mapy, objekt mapy je posílán v jazyce JSON v těle metody. Struktura JSON objektu je shodná s entitou MindMap v kapitole 3.3.

- Smazaní mapy
  - HTTP metoda: `DELETE`
  - Cesta: `/resources/map/{mapId}`
  - Metoda maže mapu na základě id `mapId`.
- Získání objektu mapy
  - HTTP metoda: `GET`
  - Cesta: `/resources/map/{mapId}`
  - Metoda vrací objekt mapy podle atributu id definovaného v proměnné `mapId`.
- Získání všech map účtu
  - HTTP metoda: `GET`
  - Cesta: `/resources/map/author/{username}`
  - Metoda vrací všechny mapy příslušící k účtu s uživatelským jménem definovaným v proměnné `username`.

4.1.3 Uzel

- Vytvoření uzlu
  - HTTP metoda: `POST`
  - Cesta: `/resources/node/{mapId}/{parentId}`
  - Metoda vytváří nový uzel v mapě s id `mapId`. Rodičovský uzel nově vzniklého uzlu má id `parentId` V těle požadavku je posílán objekt uzlu jazyce JSON. Struktura JSON objektu je shodná s entitou MapNode v kapitole 3.3.
- Úprava obsahu uzlu
  - HTTP metoda: `PUT`
  - Cesta: `/resources/node/`
  - Metoda upravuje uzel, který je posílaný v těle metody. Struktura JSON objektu je shodná s entitou MapNode v kapitole 3.3.
- Smazaní uzlu
  - HTTP metoda: `DELETE`
  - Cesta: `/resources/node/{nodeId}`
  - Metoda maže uzel s id `nodeId`.
- Získání objektu uzlu
  - HTTP metoda: `GET`
  - Cesta: `/resources/node/{nodeId}`
  - Metoda vrací uzel s id `nodeId`.
- Získání všech uzlů mapy
  - HTTP metoda: `GET`
  - Cesta: `/resources/node/map/{mapId}`
  - Metoda vrací všechny uzly příslušící k mapě s id definovaným v proměnné `mapId`.
- Získání všech potomků rodičovského uzlu
  - HTTP metoda: `GET`
  - Cesta: `/resources/node/parent/{parentId}`
  - Metoda vrací všechny uzly, které jsou potomky rodičovského uzlu s id definovaným v proměnné `parentId`.

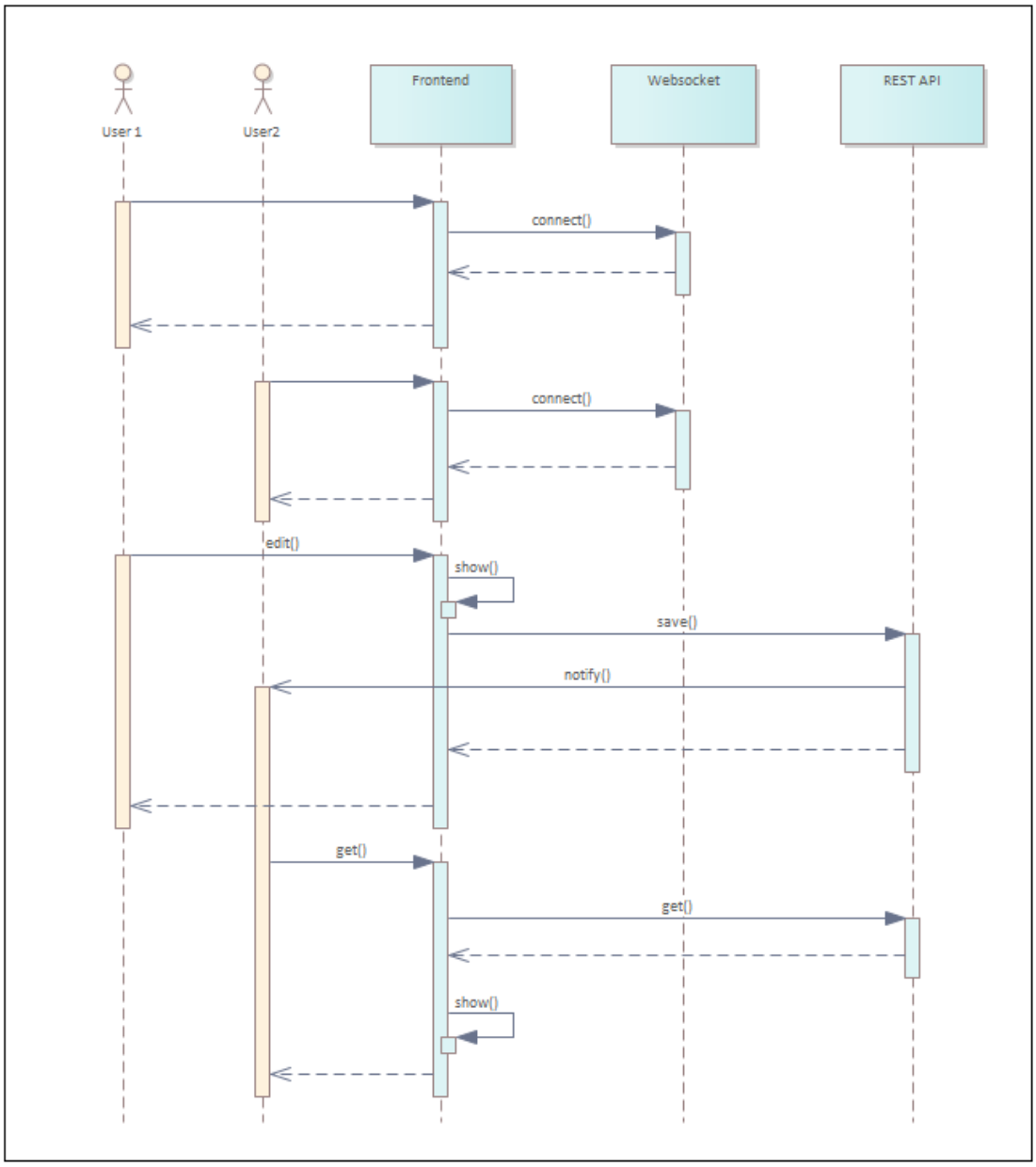
4.1.4 Práva

- Získání všech práv účtu
  - HTTP metoda: `GET`
  - Cesta: `/resources/rights/author/{accountUsername}/`
  - Metoda vrací všechny objekty práv příslušící k účtu s učivatelským jménem definovaným v proměnné `accountUsername`.
- Získání všech práv k mapě
  - HTTP metoda: `GET`
  - Cesta: `/resources/rights/map/{mapId}/`
  - Metoda vrací všechny objekty práv příslušících k mapě s id definovaným v proměnné `mapId`.
- Vytvoření práv
  - HTTP metoda: `POST`
  - Cesta: `/resources/rights/`
  - Metoda vytváří práva podle objektu posílaného v těle požadavku ve formě JSONu.
- Úprava práv
  - HTTP metoda: `PUT`
  - Cesta: `/resources/rights/`
  - Metoda upravuje objekt práv na základě entity poslané v těle požadavku ve formě JSONu.
- Odstranění práv
  - HTTP metoda: `DELETE`
  - Cesta: `/resources/rights/{rightsId}/`
  - Metoda maže objekt práv na základě id definovaného proměnnou `rightsId`.

4.2 WebSocket

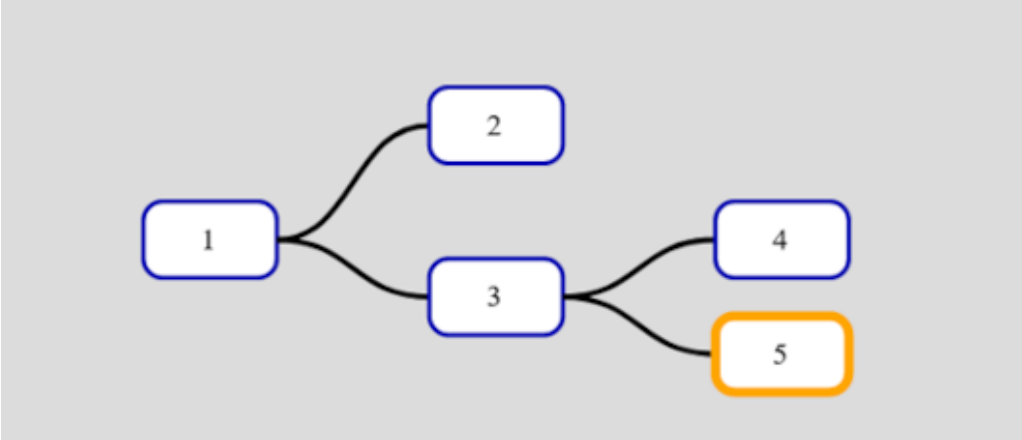
WebSocket je protokol pro obousměrnou komunikaci mezi klientem a serverem. Kvůli WebSocketu nemusí server čekat na požadavek od klienta, ale může data rovnou klientovi poslat. To činí aplikaci interaktivní. Zde je konkrétně použita specifikace Jakarta WebSocket pro definování rozhraní WebSocketu [25–26].

V této aplikaci posílá WebSocket informaci o každé změně v diagramu. Není zde posílán přímo SVG kód, ale pouze informace o tom, že došlo ke změně. Cílový klient si na základě této informace diagram aktualizuje pomocí REST rozhraní. Tato skutečnost je tu kvůli snížení síťové náročnosti při komunikaci pomocí WebSocketu.



Obrázek 4.2. Sekvenční diagram pro WebSocket.

4.3 Vykreslování diagramu




```
<svg id="map">
    <g id="6">
        <a href="#" target="_self" focusable="true" title="1">
            <rect x="115" y="230" rx="10" ry="10" width="70" height="40"></rect>
            <text x="145" y="255">1</text>
        </a>
        <g id="7">
            <path d="M 185 250 C 225 250, 225 190, 265 190"></path>
            <a href="#" target="_self" focusable="true" title="2">
                <rect x="265" y="170" rx="10" ry="10" width="70" height="40"></rect>
                <text x="288" y="196">2</text>
            </a>
        </g>
        <g id="8">
            <path d="M 185 250 C 225 250, 225 280, 265 280"></path>
            <a href="#" target="_self" focusable="true" title="3">
                <rect x="265" y="260" rx="10" ry="10" width="70" height="40"></rect>
                <text x="288" y="286">3</text>
            </a>
            <g id="9">
                <path d="M 335 280 C 375 280, 375 250, 415 250"></path>
                <a href="#" target="_self" focusable="true" title="4">
                    <rect x="415" y="230" rx="10" ry="10" width="70" height="40"></rect>
                    <text x="438" y="256">4</text>
                </a>
            </g>
            <g id="10">
                <path d="M 335 280 C 375 280, 375 310, 415 310"></path>
                <a href="#" target="_self" focusable="true" title="5">
                    <rect x="415" y="290" rx="10" ry="10" width="70" height="40"></rect>
                    <text x="438" y="316">5</text>
                </a>
            </g>
        </g>
    </g>
</svg>
```

Obrázek 4.3. Ukázka diagramu zapsaného pomocí SVG.

Pro vykreslování diagramu je používána technologie SVG. Data přijdou na klientskou část v podobě JSON objektu mapy. Na základě JSON objektu je diagram sestaven pomocí Javascriptu. Každému uzlu je přiřazena skupina SVG elementů s pevnou strukturou. Každý objekt uzlu si drží ukazatel na svou skupinu elementů, aby s ní při úpravě mohl pracovat.

Aby bylo možné se skupinou elementů snáze pracovat a řadit diagram do stromové struktury, jsou elementy drženy pomocí elementu `<g>`. Tento element není vykreslován. Další element, který skupina obsahuje je element představující obdélník `<rect>`. Dále textový element `<text>` držící obsah uzlu. Do skupiny SVG elementů je vkládán HTML element `<a>` kvůli fokusovatelnosti uzlu a obsluze událostí. První elementem skupiny pak bývá element cesty, vedoucí k rodičovskému uzlu, s názvem `<path>`. Tento element není obsažen v kořenovém uzlu. Cesta je vykreslována pomocí Bézierovy křivky natažené mezi body zadanými koncem rodičovského a začátkem uzlu potomka. Zbylé dva body jsou vypočteny z bodů předchozích. Vykreslované SVG elementy nelze do sebe vkládat, proto jsou do sebe vkládány celé skupiny označené elementem `<g>`. Každému vykreslovanému elementu je také nutné nastavit pozici.

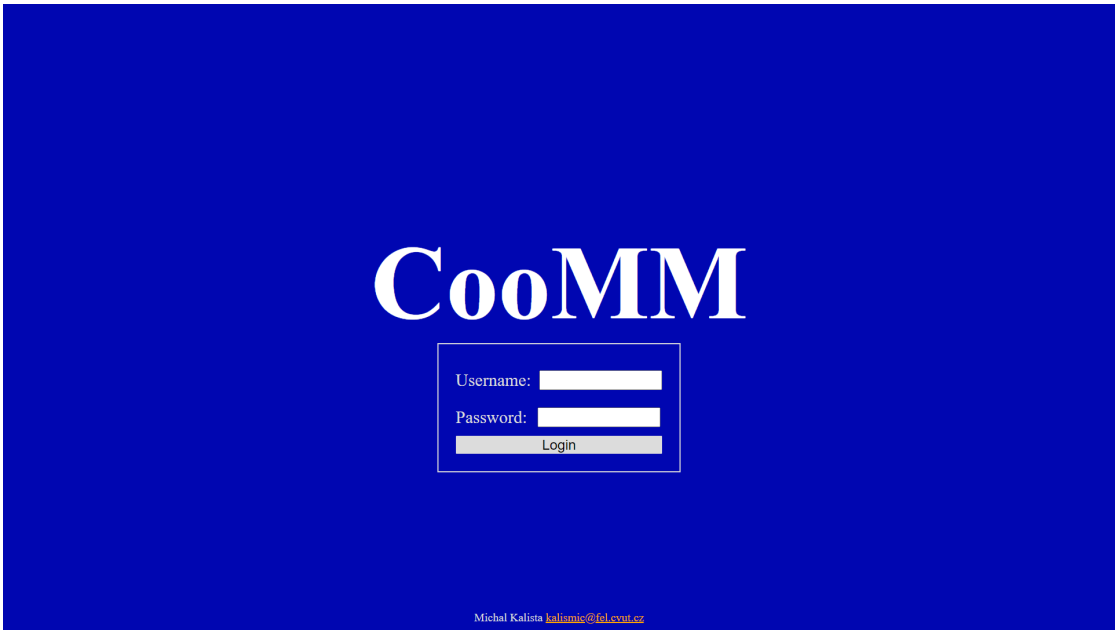
Při přidání nebo odebrání uzlu jsou přepočítávány pozice SVG elementů ve směru osy y tak, aby byl rodičovský uzel vždy uprostřed svých potomků. Dle příkladu se uzel s číslem 1 nachází uprostřed mezi uzli 2 a 5 ve směru osy y. Zároveň je dbáno na to, aby se uzly na konci stromu nepřekrývali. Proto taky v příkladu 2 je odsazená od 1 víc než 3, aby po přidání uzlu potomka k uzlu s názvem 2, nový uzel nepřekrýval uzel 4. Na obrázku 4.3 můžete vidět ukázku uspořádání SVG elementů v grafické podobě a zápis stejného diagramu pomocí SVG elementů.

Protože text není vždy stejně dlouhý, provádí se přepozicování i ve směru osy x. Přepozicování má více fází. Pokud velikost textu nepřesahuje obdélník, je text vystředěn. Pokud text přesáhne velikost obdélníku, začne se obdélník zvětšovat, ale jen po určitou dobu, a to směrem doprava. Při dalším zvětšení se text ořízne a na konec se přidají tři tečky. Celý obsah uzlu se uživatel dozví v případě stisknutí klávesy control v dialogovém okně, kde může měnit obsah uzlu.

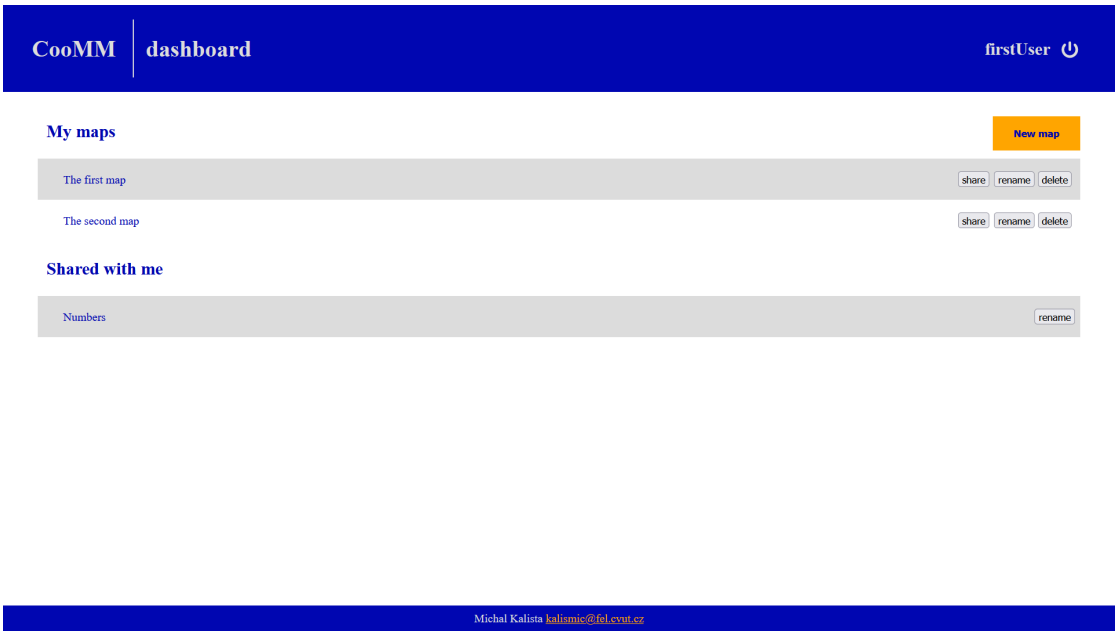
4.4 Průchod aplikací

Aplikace je rozložena na třech obrazovkách, stejně jako aplikace uváděné v analýze.

Na obrázku 4.4 je vidět úvodní obrazovka aplikace. Zde se může uživatel přihlásit pomocí jména a hesla.



Obrázek 4.4. Výchozí obrazovka aplikace.



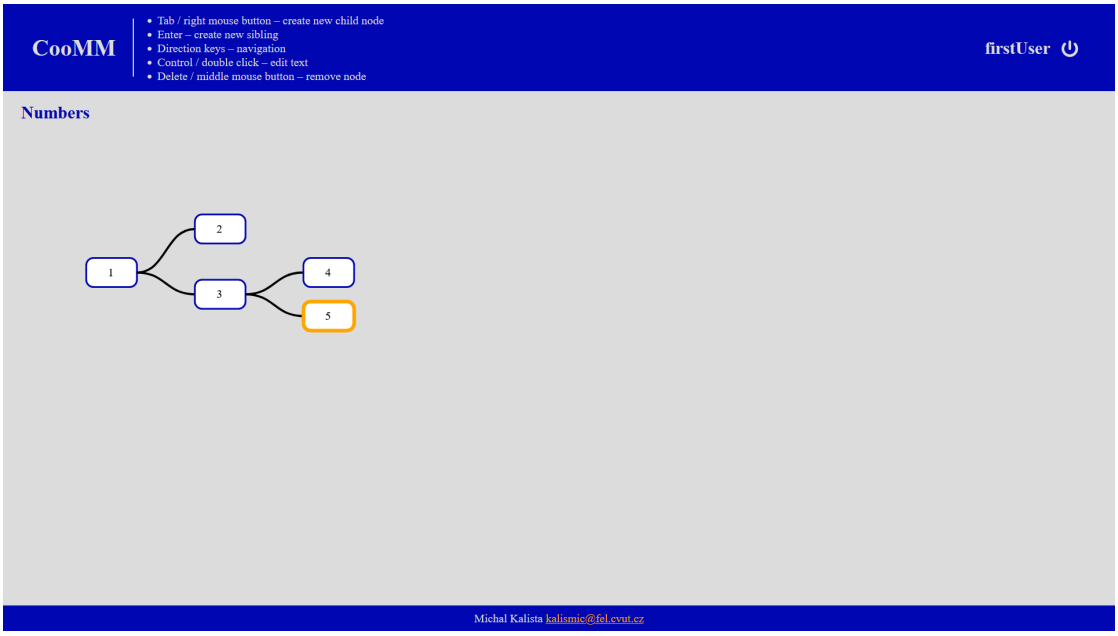
Obrázek 4.5. Centrální obrazovka aplikace.

Po přihlášení je uživatel přesměrován na centrální obrazovku, která je vidět na obrázku 4.5. Hlavní funkcionalita práce s mapou je soustředěna právě na tuto obrazovku. V pravém horním rohu je vidět uživatelské jméno účtu, pomocí kterého je v dané chvíli aplikace prohlížena. Toto jméno zde zůstává po celou dobu práce s aplikací. Vedle něj zle spatřit symbol umožňující odhlášení z aplikace. V levém horním rohu je název aplikace,

pomocí kterého se uživatel dostane na centrální obrazovku. V hlavní části stránky je výpis uživatelských map a map s uživatelem sdílených. Vlastní mapy může uživatel přejmenovávat, mazat a sdílet s uživateli jinými. Mapy sdílené může pouze přejmenovávat. Po kliknutí na název mapy je uživatel přesměrován na obrazovku diagramu. Z centrální obrazovky uživatel může vytvářet mapy nové po kliknutí na oranžové tlačítko nad výpisem.

Zobrazení obrazovky diagramu je vidět na obrázku 4.6. Postavení uživatelského jména a názvu aplikace zůstalo stejné. Vedle názvu navíc přibyla nápověda, jak aplikaci ovládat a pod ním název právě zobrazené mapy. Kořenový uzel mapy je vždy zobrazen v levé části aplikace vertikálně ve středu stránky. Od něj jsou jeho potomci pospojovány pomocí Bézierových křivek ve směru vpravo. Aktuální uzel je označen pomocí oranžového rámečku. Mačkáním na příslušné klávesy lze z tohoto uzlu provádět aktivity pro tvorbu diagramu. Navigace mezi uzly je zpřístupněna pomocí kláves šipek. Pokud není označen uzel žádný, je potřeba uzel vybrat pomocí myši. Přidávání potomků uzlu je zpřístupněno pod klávesou tabulátor. Přidávání sesterských uzlů pomocí klávesy enter. Kořenovému uzlu nelze přidávat sesterské uzly. Změna textu uzlu je zpřístupněna klávesou control v dialogovém okně. Uzly lze mazat pomocí klávesy delete. Mapu lze tvořit i používáním myši. Vytváření uzlu je zpřístupněno kliknutím pravým tlačítkem myši na rodičovský uzel, mazání kliknutí prostředního tlačítka myši na uzel, který chceme smazat a dvojklik je vyhrazen pro úpravu obsahu uzlu.

Změní-li uživatel svou mapu, změny se okamžitě projeví všem uživatelům, se kterými je mapa sdílená. Všichni uživatelé proto vidí diagram stejně, v aktuálním stavu.



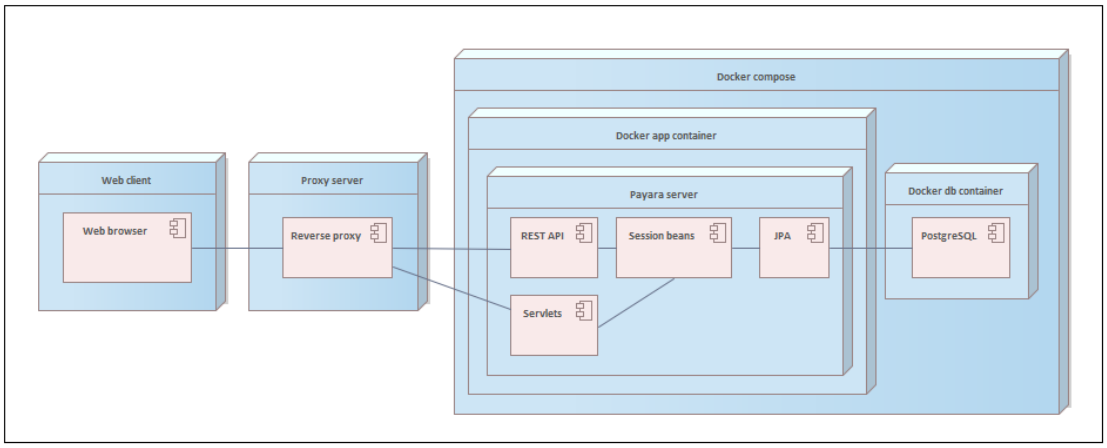
Obrázek 4.6. Obrazovka diagramu v aplikaci.

Kapitola 5
Nasazení

Při vývoji jsem aplikaci nasazoval na lokální Payara server na port 8080 s cestou `/CooMM`. Databáze běžela lokálně na portu 5432. Pro produkční prostředí by bylo ale výhodnější použití Dockeru tak, jak je znázorněno nasazení na obrázku 5.1.

Aplikace běží na aplikační serveru Payara na výchozím portu 8080, aplikační konzole Payara serveru na portu 4848. Z aplikační konzole lze upravovat například napojení na databázi. Aplikační server je pomocí Dockeru zakontejnerizován do kontejneru aplikace. Databáze běží na databázovém serveru na portu 5432 a také je zakontejnerizována do Docker kontejneru.

Oba dva kontejnery jsou spouštěny pomocí Docker compose. Docker compose také definuje překlad portů pro Payara server na 9580 a jeho administrativní konzoli na port 9548. Databáze běží v Docker compose na portu 9584. Na těchto portech spolu kontejnery komunikují. Rest API a servlety komunikují s prohlížečem uživatele skrze Revers proxy.



Obrázek 5.1. Diagram nasazení.

Kapitola 6
Testování

Aplikace byla otestována pomocí jednotkových a integračních testů. Pro testování byla použita knihovna JUnit 5[1].

Pomocí jednotkových testů byla otestována pouze netriviální funkcionalita, které aplikace neobsahuje velké množství.

Více jsem se soustředil na itegrační testování celé backendové části aplikace. Protože testy interagují s databází, je žádoucí, aby byla databáze před začátkem testování prázdná, aby data, která databáze již obsahuje, neovlivňovala testování například shodným jménem uživatele. Proto bylo integrační testování vyčleněno do samotné aplikace, aby mohlo být prováděno jen když je připojena testovací (prázdná) databáze. Vývojářům to také dává možnost přeskočit fázi integračního testování a tím zrychlit sestavení a nasazení aplikace.

Integrační testování probíhá voláním konkrétních metod REST rozhraní. Protože je testování vyčleněno do samostatného projektu, nelze metody volat přímo. Pro volání metod je nutné použít HTTP požadavky. Abych předešel málo přehlednému a na chyby náročnému způsobu tvorby HTTP požadavků tradiční cestou, použil jsem pro testovaní technologii MicroProfile Rest Client, která je popsána v kapitole 6.1. MicroProfile Rest Client je implementován ve formě rozhraní. Narozdíl od tradičního způsobu tvorby požadavků, klienta stačí jednou založit a poté už jsou jen volány metody definované v rozhraní.

Rozhraní MicroProfile Rest Clientu musí být při změně REST rozhraní také upraveno. Kromě rozhraní MicroProfile Rest Clientu je nutné také vytvořit objekty, které jsou posílány v těle požadavků ve formě JSON objektů a které také přichází v odpovědi. Tyto objekty musí vlastnostmi kopírovat objekty používané v REST rozhraní testované aplikace. I tyto objekty se mohou s úpravou testované aplikace měnit. Proto jsem využil technologie OpenApi Generator, abych si vygeneroval rozhraní MicroProfile Rest Clientu a datové objekty, které jsou v REST rozhraní používány. Tato technologie je popsána v kapitole 6.2 i s konkrétním využitím v mé aplikaci.

6.1 MicroProfile Rest Client

MicroProfile Rest Client je specifikace umožňující vytváření jednoduchého klienta pro typově bezpečný přístup k RESTful službám založených na Jakarta EE. MicroProfile Rest Client odstiňuje vývojáře od nutnosti psát HTTP volání tradičním způsobem. Vývojář tak vytváří požadavek pouze zavoláním metody, která je definována v klientu. U definice metody jsou anotacemi nastavovány parametry požadavku jako HTTP metoda, cesta, na které je daný požadavek dostupný, nebo jaký typ dat metoda přijímá. Samotnou metodu však vývojář nikdy neimplementuje, klient je definován jako Jakarta EE rozhraní. Rozhraní klienta je označeno anotací `@RegisterRestClient`. Před použitím je nutné klienta vytvořit pomocí RestClientBuilderu a definovaného rozhraní. Tato technologie poskytuje vývojáři menší chybovost při psaní kódu a snazší úpravy [27–28].

[1] `https://junit.org/junit5/`

6.2 OpenApi Generator

OpenApi Generator slouží ke generování klientského rozhraní, serverových stubů, dokumentace a konfigurace na základě OpenApi specifikace. OpenApi specifikace může být ve formě YAML nebo JSON souboru. OpenApi Generator podporuje generování ze specifikace verze 2.0 a novější. OpenApi Generator zahrnuje generování do velkého množství různých jazyků a frameworků, například Java, C++, C#, PHP, Python, Ruby, Scala a další. OpenApi Generator lze instalovat, používat jako plugin nebo využít online generátor [29–31].

V této práci jsem OpenApi Generator použil ke generování Microprofile Rest Clientu a DTO objektů, se kterými rozhraní klientu pracuje. Pro generování jsem použil OpenApi Generator Docker Image s konfiguračním souborem config.json. V mém konfiguračním souboru jsou nastavovány jména balíčků, generátor pro jazyk Java, použití knihovny microprofile nebo vstupní OpenApi specifikace. OpenApi specifikaci lze po nasazení na Payara server najít na výchozím portu *4848/openapi/*. Na obrázku 6.1 je vidět příklad části OpenApi specifikace a k ní příslušná vygenerovaná metoda MicroProfile Rest Clientu. Na následujícím řádku je vidět příkaz pro spuštění docker image v mé aplikaci.

```
docker run --rm -v ${PWD}:/local openapitools/openapi-generator-cli
generate -o /local/out/CoommTestApp -c /local/out/config.json
```

```yaml
put:
  operationId: updateAccount
  requestBody:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/AccountDto'
  responses:
    default:
      content:
        application/json:
          schema:
            type: object
      description: Default Response.
```

```java
@RegisterRestClient
@RegisterProvider(ApiExceptionMapper.class)
@Path("/resources")
public interface DefaultApi {

    @PUT
    @Path("/account")
    @Consumes({"application/json"})
    @Produces({"application/json"})
    Response updateAccount(AccountDto accountDto)
     throws ApiException, ProcessingException;
```

Obrázek 6.1. Příklad metody MicroProfile Rest Clientu vygenerované z OpenApi specifikace.

Kapitola 7
Závěr

V analytické části práce jsem se zaměřil na analýzu dostupných nástrojů pro vytváření myšlenkových map. U dostupných řešení jsem porovnával možnost spolupráce více lidí na tvorbě mapy v reálném čase, kvalitu ovládání mapy, přehlednost, možnost používat nástroj zdarma a nadstandardní funkcionality, které uživatelům ulehčují práci. Zjistil jsem, že dostupné nástroje často neposkytují funkcionality umožňující kooperaci více lidí v jeden okamžik. Ty nástroje, které takovou funkci mají, jsou dostupné pro pravidelné používání jen za poplatek. Aplikace jsou často navrženy tak, aby zaujaly svým návrhem, to ale často prostředí znepřehlední.

V implementační části práce jsem nástroj na tvorbu myšlenkových map vytvářel. Cílem bylo vytvořit aplikaci umožňující kooperaci lidí, která nebude omezena nutností platit poplatky. Ve své práci jsem vycházel z předchozí analýzy, a proto jsem se snažil udělat aplikaci jednoduchou a snadno ovladatelnou. Výsledná práce proto neřeší žádnou nadstandardní funkcionalitu. Podstatná část práce byla vykreslování diagramu, který je řešen pomocí technologie SVG. SVG je vykreslováno pomocí Javascriptu z JSON objektu, který přijde v odpovědi na HTTP požadavek. Pro snazší manipulaci jsou SVG elementy skládány do stromové struktury. Aby byla zajištěna přehlednost, jsou elementy uzlů přepozicovávány. Pro zajištění spolupráce v reálném čase je využita obousměrná komunikace. Obousměrnou komunikaci mezi klientem a serverem zajišťuje Jakarta WebSocket. Výsledná aplikace byla otestována pomocí integračních testů pro REST rozhraní v samostatné testovací aplikaci. Pro snazší testování byl využit OpenApi Generator a MicroProfile Rest Client.

Literatura

[1] *Co jsou myšlenkové mapy?*
https://kisk.phil.muni.cz/kreativita/temata/myslenkove-mapy/co-jsou-myslenkove-mapy. (Citováno 04. 05. 2023).

[2] *Myšlenkové mapy.*
https://www.mzk.cz/sites/mzk.cz/files/mentalni_mapy.pdf. (Citováno 04. 05. 2023).

[3] *Sociologická encyklopedie - kooperace.*
https://encyklopedie.soc.cas.cz/w/Kooperace. (Citováno 04. 05. 2023).

[4] *Coggle.*
https://coggle.it. (Citováno 03. 12. 2022).

[5] *Coggle Mind Map.*
https://project-management.com/coggle-software-review/. (Citováno 03. 12. 2022).

[6] Vojtěch Hamerský. *Interaktivní tabule s Miro*. 2021.
https://www.ped.muni.cz/digcompedu/clanky/interaktivni-tabule-s-miro. (Citováno 28. 11. 2022).

[7] *About Miro, the leading visual collaboration platform.*
https://miro.com/about/. (Citováno 27. 11. 2022).

[8] *Miro pricing.*
https://miro.com/pricing/. (Citováno 03. 12. 2022).

[9] *Bubbl.us pricing.*
https://bubbl.us/pricing. (Citováno 10. 12. 2022).

[10] Saad Mohammad. *Bubbl.us review*. 2020.
https://www.techradar.com/reviews/bubblus. (Citováno 10. 12. 2022).

[11] Jenny Chang. *WiseMapping review*. 2022.
https://reviews.financesonline.com/p/wisemapping/. (Citováno 12. 12. 2022).

[12] Paulo Veiga. *WiseMapping features*. 2014.
https://wisemapping.atlassian.net/wiki/spaces/WS/pages/524318/Features. (Citováno 12. 12. 2022).

[13] *Jakarta EE: An open source framework for developing cloud native Java applications.*
https://jakarta.ee/about/jakarta-ee/. (Citováno 29. 12. 2022).

[14] *Differences between Java EE and Java SE.*
https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html. (Citováno 29. 12. 2022).

[15] *Overview of Enterprise Applications.*
https://docs.oracle.com/javaee/6/firstcup/doc/gcrky.html. (Citováno 9. 1. 2023).

[16] *Java EE Servers*.
https://docs.oracle.com/javaee/6/firstcup/doc/gcrkq.html. (Citováno 9. 1. 2023).

[17] *Payara release notes*.
https://docs.payara.fish/community/docs/Release Notes/Release Notes 5.2022.3.html. (Citováno 29. 12. 2022).

[18] *GlassFish 5 x Payara Server 5 – Comparison*.
https://www.payara.fish/glassfish-vs-payara-server-5/. (Citováno 30. 12. 2022).

[19] *PostgreSQL about*.
https://www.postgresql.org/about/. (Citováno 30. 12. 2022).

[20] *SVG: Scalable Vector Graphics*.
https://developer.mozilla.org/en-US/docs/Web/SVG. (Citováno 30. 12. 2022).

[21] George John. *What is the difference between SVG and HTML5 Canvas?* 2020.
https://www.tutorialspoint.com/What-is-the-difference-between-SVG-and-HTML5-Canvas. (Citováno 08. 05. 2023).

[22] *What is "Vanilla JavaScript"?*
https://www.javatpoint.com/what-is-vanilla-javascript. (Citováno 08. 05. 2023).

[23] Prakhar Srivastav. *Docker for beginners*.
https://docker-curriculum.com/. (Citováno 15. 05. 2023).

[24] *Docker – 1. Představení, instalace a základní operace*. 2022.
https://www.websupport.cz/podpora/kb/docker-1-predstaveni-instalace-a-zakladni-operace/. (Citováno 15. 05. 2023).

[25] *Jakarta WebSocket*.
https://projects.eclipse.org/projects/ee4j.websocket. (Citováno 16. 05. 2023).

[26] *Chapter 14. Creating Jakarta WebSocket applications*.
https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html/development_guide/creating_websocket_applications. (Citováno 16. 05. 2023).

[27] *Rest Client for MicroProfile*.
https://microprofile.io/project/eclipse/microprofile-rest-client. (Citováno 14. 05. 2023).

[28] Andy McCright. *Introducing MicroProfile Rest Client 1.0*. 2018.
https://openliberty.io/blog/2018/01/31/mpRestClient.html. (Citováno 14. 05. 2023).

[29] *OpenAPI Generator*.
https://openapi-generator.tech/. (Citováno 14. 05. 2023).

[30] Kristopher Sandoval. *Introduction to OpenAPI Generator*. 2020.
https://nordicapis.com/introduction-to-openapi-generator/. (Citováno 14. 05. 2023).

[31] Gunter Rotsaert. *Generate Server Code Using OpenAPI Generator*. 2022.
https://mydeveloperplanet.com/2022/02/08/generate-server-code-using-openapi-generator/. (Citováno 14. 05. 2023).