FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Generátor myšlenkových map z WooWoo dokumentů
Student:	Matěj Frnka
Vedoucí:	Ing. Tomáš Kalvoda, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1. Prozkoumejte nástroje umožňující pomocí webových technologií atraktivně vizualizovat myšlenkové mapy (vztahy mezi pojmy).

2. Seznamte se s WooWoo formátem využívaným při vytváření studijních textů k předmětům BI-PKM, BI-ZMA a BI-LA1.21 a se způsobem generování výstupů pomocí WooWoo šablon (aktuálně PDF a HTML verze textu).

3. Navrhněte WooWoo šablonu, která ze zdrojového kódu textu vygeneruje myšlenkovou mapu pojmů (definice, věty, další důležitá tvrzení, ...). Případně navrhněte nutná rozšíření zdrojového kódu o další anotace objektů v textu.

4. Šablonu implementujte a její správnou funkčnost otestujte. Dbejte na přehlednost a interaktivitu prezentace pojmů podle jejich důležitosti.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 15. října 2021 v Praze.

Bakalářská práce

GENERÁTOR MYŠLENKOVÝCH MAP Z WOOWOO DOKUMENTŮ

Matěj Frnka

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Tomáš Kalvoda, Ph.D.
11. května 2022

Obsah

Seznam obrázků

Seznam výpisů kódu

Děkuji vedoucímu této práce Ing. Tomáši Kalvodovi, Ph.D. za všechen čas, který věnoval této práci. Především za jeho rady ohledně návrhu šablony i webové aplikace, za pomoc s pochopením WooWoo, za časté konzultace, za rychlé odpovědi na zprávy v jakýchkoliv hodinách a za jeho vždy vstřícný přístup. Také děkuji za pomoc s korekturou práce, a to především Janu F., Josefíně L., Monice S a Matějovi K. Dále i Daně F., Tomáši F., Vojtěchovi M., Matyáši R a Matoušovi K.

Prohlášení

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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2022 .....

Abstrakt

Tato práce se zabývá vytvořením nástroje pro automatickou tvorbu myšlenkových map z WooWoo souborů napsaných podle specifikace *FIT-šablona*. Vytvořená myšlenková mapa slouží jako studijní materiál zobrazující vybrané pojmy z WooWoo souborů a znázorňující jejich vztahy. Studijní materiál je interaktivní a jednoduše dostupný pro studenty ve formě webové stránky.

V práci je řešena extrakce pojmů z textového souboru WooWoo pomocí stejnojmenné knihovny WooWoo, rozložení pojmů a jejich vztahů do myšlenkové mapy pomocí algoritmů pro vizualizaci grafů, určení důležitosti pojmů pomocí algoritmů pro určení centrality a interaktivní zobrazení pojmů a jejich vztahů jako myšlenkovou mapu ve webové aplikaci.

Nástroj se v době odevzdání této práce využívá na Fakultě informačních technologií ČVUT.

Klíčová slova WooWoo šablona, knihovna WooWoo, myšlenková mapa, FIT-šablona, generované studijní materiály, WooWoo

Abstract

This thesis is about the creation of a tool for generating mind maps from WooWoo files written according to the *FIT-template*. The generated mind map serves as a study material visualizing selected concepts from WooWoo files and displaying their relationships. The study material is an interactive web application easily accessible to students.

The thesis deals with extraction of concepts from WooWoo text files using the WooWoo library; with generation of a layout for the concepts and for their relationships in the mind map using graph drawing algorithms from graph theory; with figuring out the importance of concepts using centrality algorithms and with problems of interactively displaying concepts and their relationships as a mind map in a web application.

At the time of publication, the tool is being used at the Faculty of Information Technology, CTU.

Keywords WooWoo template, WooWoo library, mind map, FIT-template, generated study materials, WooWoo

Seznam zkratek

WooWoo	Write only once, write only once
ČVUT	České vysoké učení technické v Praze
FIT	Fakulta informačních technologií
API	Application Programming Interface (Programové rozhraní)

Úvod

Existuje mnoho různých studijních materiálů i způsobů studia. Někomu více vyhovuje čtení skript, někomu zase stálé opakování krátkých úseků. Tato práce se bude snažit přidat studijní materiál, jehož cílem bude vizualizovat pojmy do myšlenkové mapy. Tento nový studijní materiál by měl sloužit jako doplněk ke standardním textovým skriptům. Mělo by ho být možné vytvářet z již existujícího formátu souborů jménem WooWoo. Výstup by měl graficky zobrazovat pojmy a jejich vzájemné závislosti.

Inspiraci k tvorbě této práce jsem našel při studiu definic z lineární algebry. Často se mi stávalo, že jsem ve studijních textech narazil na definici, která se odkazovala na jiné definice, které jsem buď neznal, anebo jsem si nebyl přesně jistý jejich zněním. Ve studijním textu ale nebyl způsob, jak nahlédnout na předchozí definici a pak zase pokračovat tam, kde jsem přestal. Chtěl jsem nějaký nástroj, ve kterém bych mohl lehce přecházet mezi definicemi podle závislostí.

Po chvíli experimentování jsem došel k tomu, že nejpřehlednější je zobrazit definice jako myšlenkovou mapu, kde jsou všechny definice propojené s definicemi, na kterých jsou závislé. Více o myšlenkových mapách v Kapitole 2.

Abych mapu vytvořil, použil jsem nástroj `www.orgpad.com`, manuálně jsem do něj přidal všechny definice a propojil je podle závislostí. Nástroj fungoval dobře, ale manuální zadávání bylo nudné a zdlouhavé. Cílem této práce je nahradit manuální práci automatickým generováním myšlenkových map a zároveň vylepšit grafické zobrazení přímo pro účely studia.

K tomu bude využita již existující technologie jménem WooWoo. WooWoo představuje formát textových souborů nazývaných WooWoo soubory a označených příponou .woo, knihovnu WooWoo, sloužící ke zpracování WooWoo souborů a šablony, využívající knihovnu WooWoo k vytvoření reprezentace dat z WooWoo souborů v různých formátech. Samotné soubory WooWoo nijak neurčují vzhled finálního dokumentu, jen text označují a přidávají k němu metadata. Šablony využívající knihovnu WooWoo pak metadata a označení zpracují a dále využijí například k vytvoření PDF dokumentu.

Motivace k využití WooWoo je především v tom, že již existují zdrojové soubory ve formátu WooWoo pro několik předmětů na Českém vysokém učení technickém v Praze na Fakultě informačních technologií a aktivně se využívají pro tvorbu skript do formátu PDF, HTML a pro generování Anki kartiček. Tyto již existující zdrojové soubory WooWoo tedy bude možné využít i ke generování myšlenkových map.

Knihovna WooWoo je udržována a rozvíjena Ing. Tomášem Kalvodou, Ph.D. a šablony pro tvorbu dokumentů z WooWoo souborů pomalu přibývají. Více o WooWoo souborech v Kapitole 1.

Cíle

Hlavním cílem této práce je vytvořit novou šablonu, která bude z WooWoo souborů generovat myšlenkovou mapu a zobrazí ji jako webovou aplikaci. Myšlenková mapa by měla být interaktivní a přehledná.

Tento cíl se dá rozložit do menších cílů, které je potřeba splnit k dosažení hlavního cíle:

1. Seznámit se s formátem WooWoo, knihovnou WooWoo, se šablonami pro generování dokumentů ze zdrojových kódů .woo a s pojmem myšlenková mapa.
2. Určit požadavky pro myšlenkovou mapu.
3. Prozkoumat existující nástroje pro vizualizaci myšlenkových map, zvážit jejich využitelnost pro analyzované požadavky.
4. Navrhnout WooWoo šablonu, nutná rozšíření zdrojového kódu a způsob vizualizace myšlenkových map.
5. Implementovat návrh, implementaci otestovat.

Kapitola 1
WooWoo

V této kapitole je do detailu rozebráno, jak vypadá WooWoo projekt, co jsou to WooWoo soubory a jaká je jejich základní syntaxe, co je to knihovna WooWoo a co v této práci představuje pojem šablona. Na konci jsou analyzovány existující nástroje související s WooWoo, které bude možné využít v této práci.

1.1 Úvod do WooWoo

Úplně pro začátek je důležité ujasnit následující pojmy:

Formát WooWoo představuje textový formát dodržující pravidla určené v Kapitole 1.4.

WooWoo soubor je textový soubor s příponou .woo ve formátu WooWoo.

Knihovna WooWoo je knihovna v jazyce Ruby pro zpracovávání WooWoo souborů. Více v Kapitole 1.3.

Šablona je program využívající Knihovnu WooWoo. Vytváří interpretaci dat z WooWoo souborů. Více v Kapitole 1.5.

WooWoo samo o sobě představuje ekosystém všech výše zmíněných pojmů.

WooWoo se využívá k tvorbě dokumentů a jejich zobrazení v různých formátech. Momentálně se využívá k tvorbě studijních materiálů pro předměty BI-PKM, BI-ZMA, BI-LA1, BI-LA2 a BI-MA1[1] na Fakultě informačních technologií spadající pod České vysoké učení technické.

Vývoj začal Ing. Tomáš Kalvoda, Ph.D. v roce 2017. V době odevzdání této práce se na vývoji nadále pokračuje.

WooWoo[2] soubory se označují příponou .woo. Jde o dokumenty v textovém formátu podobné například LaTeXovým souborům .tex. Soubory se tedy dají vytvářet v jednoduchých textových editorech jako je notepad nebo Atom. Pro editor Atom existuje rozšíření popisované v [1], které zobrazuje živý náhled dokumentu a přehledně prezentuje jeho logickou strukturu. Podobně jako LaTeX má syntaxe WooWoo souborů určitá pravidla, která musí být dodržena a podle kterých se pak obsah v souborech zpracovává. Více o pravidlech je v Kapitole 1.4.

Jak zkratka *Write only once, Write only once* (WooWoo) napovídá, WooWoo se využívá pro ušetření práce a zobrazení stejného dokumentu v mnoha různých formátech. Autor napíše dokument pouze jednou a za pomocí WooWoo pak může generovat dokumenty v různých formátech.

[1] Příklad HTML skript MA1: `https://courses.fit.cvut.cz/BI-MA1/@master/textbook/`

[2] Zkratka pro *Write only once, Write only once* (because why write it twice)

Navíc jsou tyto vygenerované dokumenty vzhledově konzistentní s ostatními dokumenty vytvořenými pomocí stejné šablony. Momentálně se WooWoo využívá pro generování do formátu PDF, do webových stránek a pro tvorbu Anki[3] kartiček. Vyvíjí se ale i další šablony, například šablona pro tvorbu dokumentů pro čtečky e-knih.

Návrh WooWoo knihovny a šablon dovoluje jednoduchou rozšiřitelnost pro přidávání dalších šablon, čehož se využívá i v této práci.

Protože knihovna a formát WooWoo jsou stále poměrně nové[4] a mimo Fakultu informačních technologií na ČVUT zatím nevyužívané, jsou zdroje s informacemi o WooWoo limitované. Jediný oficiální dokument o WooWoo je specifikace [2], která popisuje WooWoo formát a *FIT-šablonu*. Pro získání mnohých informací o WooWoo tedy nezbývá jiná možnost, než vyčtení ze zdrojového kódu nebo jejich získání od tvůrce WooWoo, Ing. Tomáše Kalvody, Ph.D.

1.2 Soubory a složky WooWoo projektu

K pochopení fungování WooWoo je užitečné znát alespoň základy složkové struktury použité v projektech. Na Obrázku 1.1 je příklad WooWoo projektu. Symbolem * jsou označené soubory a složky, které v každém projektu nutně musí být.

```
WooWoo
├── build ..... automaticky generovaná složka s výstupy
│   ├── template1 ..... složka s výstupy šablony template1
│   └── template2 ..... složka s výstupy šablony template2
├── templates* ..... složka se šablonami
│   ├── template1 ..... šablona template1
│   │   └── tasks.rb ..... soubor obsahující Rake[5] tasks
│   └── template2 ..... šablona template2
│       ├── helpers.rb ..... soubor s pomocnými metody pro .erb soubory ve složce erb
│       ├── tasks.rb ..... soubor obsahující Rake tasks
│       └── erb ..... složka s Embedded Ruby soubory
│           ├── Solution.html.erb ..... definice objektu Solution (více v Sekci 1.4.2)
│           └── solution.html.erb ..... definice prostředí solution (více v Sekci 1.4.3)
├── Rakefile* ..... soubor odkazující na Rake tasks v šablonách
├── Woofile* ..... parametry pro WooWoo knihovnu
└── root.woo* ..... soubor ke zpracování
```

Obrázek 1.1 Složková struktura WooWoo projektu.

V příloze ve složce `examples/sample woowoo project` jsou k dispozici soubory popisované na Obrázku 1.1 i s jejich obsahem. Níže je stručný popis souborů, jejichž význam přímo nevyplývá z popisu složkové struktury na Obrázku 1.1.

Woofile: Základem každého projektu je soubor Woofile obsahující informace ve formátu YAML[6]. Tento soubor musí vždy obsahovat proměnnou *root_file*, která určuje relativní cestu od Woofile k WooWoo souboru ke zpracování. Na Obrázku 1.1 je tento soubor root.woo. Dále může obsahovat jakékoliv další parametry, které mohou šablony nebo knihovna WooWoo využít k definici jejich chování. Všechny parametry ve Woofile jsou dostupné šablonám. Některé šablony mohou vyžadovat určité parametry ke správnému fungování.

root.woo: Soubor obsahující text, ze kterého šablony generují dokumenty. Obsah tohoto souboru je hlouběji popsán v Kapitole 1.4. Na Obrázku 1.1 je soubor označen jako povinný,

[3] Více informací o Anki na `https://apps.ankiweb.net/`
[4] První kód byl nahrán do verzovacího systému Gitlab v roce 2017.
[6] Více o formátu YAML na `https://yaml.org/spec/1.2.2/`

nemusí v projektu ale být přesně pod tímto jménem, stačí, když jeho jméno odpovídá parametru *root_file* v souboru Woofile.

Rakefile: Soubor s odkazy na Rake *tasks* jednotlivých šablon. Je využívaný knihovnou Rake, pomocí které se WooWoo spouští. Odkazy na Rake *tasks* šablon je možné definovat automaticky pomocí knihovny WooWoo. Šablony pomocí Rake *tasks* definují spustitelné operace.

soubory ve složce erb: Soubory *Embedded RuBy*[7] jsou soubory šablon. Slouží k transformaci textu dokumentu ve WooWoo formátu do jiné podoby. Například šablona pro tvorbu webových stránek by tento soubor mohla využít pro obalení paragrafu HTML znakem pro paragraf – `<p> ..paragraf </p>`

1.3 Knihovna WooWoo

Z konzultací s vedoucím práce a tvůrcem WooWoo vyšlo najevo, že knihovna WooWoo je využívána ke zpracování textu z dokumentů formátu WooWoo do objektů v programovacím jazyce Ruby a k aplikování stylů definovaných šablonou, která tuto knihovnu využívá.

Ze zdrojových kódů je vidět, že WooWoo je napsáno v jazyce Ruby verze 3.1.2. Instaluje se pomocí nástroje Rake jako *gem*[8].

Knihovna je testována a vyvíjena pomocí knihovny rspec[9], což je knihovna určena k metodice *test driven development*. *Test driven development* je dle [3] způsob vývoje softwaru, kde testy specifikují a validují co bude kód dělat. Nejprve se tedy napíšou testy a pokud selžou, upraví se funkčnost programu tak, aby testy proběhly úspěšně.

1.4 Formát WooWoo

Formát WooWoo definuje takzvané **výrazy**. Výrazy označují části textu a přidávají k nim metadata. Šablony pak podle typu výrazu a metadat rozhodnou, jak daný text zpracují. Například podle typu výrazu mohou rozhodnout, že text bude psán velkými písmeny a podle meta-bloku určí, že bude text psán žlutou barvou. Výrazy jsou popsány v následujících podkapitolách.

Podobně jako například v programovacím jazyce Python záleží ve formátu WooWoo na odsazení. Z již existujících souborů WooWoo je vidět, že se text odsazuje dvěma mezerami. U výrazů níže je tedy důležité dávat pozor na odsazení.

Pokud není řečeno jinak, všechny následující podsekce parafrázují z [2].

1.4.1 Výraz *sekce*

Každý dokument lze rozdělit na libovolné množství *sekcí*. Každá *sekce* musí mít typ a nadpis a může obsahovat takzvaný *meta-blok*, ve kterém mohou být další informace ve formátu *YAML*.

Začátek *sekce* začíná tečkou, po které ihned následuje název typu, který začíná velkým písmenem. Po typu následuje mezera a nadpis. Na novém řádku pak může být *meta-blok*. Ukázka *sekce* je ve Výpisu 1.1.

Ze zdrojových souborů předmětu BI-MA1[10] vychází, že *sekce* nemají odsazené tělo, jen označují bod v dokumentu. Do *sekce* tedy náleží veškerý text až po další výraz *sekce*.

[7] Více o *Embedded RuBy* na `https://docs.ruby-lang.org/en/2.3.0/ERB.html`
[8] Více o *gems* na `https://guides.rubygems.org/what-is-a-gem/`
[9] Více informací o rspec na `https://rspec.info/`
[10] Zdrojové soubory jsou dostupné pro studenty a zaměstnance ČVUT FIT na `https://gitlab.fit.cvut.cz/BI-MA1/bi-ma1`

```
.Typ Nadpis
  metadata: meta-blok
  format: Ukázka

Text text...
```

Výpis kódu 1.1 Ukázka *sekce* ve formátu WooWoo. [2], upraveno

```
.TypObjektu:
  meta-blok

  blok


  blok


...
```

Výpis kódu 1.2 Ukázka *objektu* se dvěma *bloky*. [2], upraveno

1.4.2 Výrazy *blok* a *objekt*

Sekce se dále dělí na *bloky* a *objekty*. Každý *objekt* má typ, označuje se typem začínajícím velkým písmenem obalen zepředu znakem tečky a zezadu znakem dvojtečky. Stejně jako u *sekce* může po řádku s typem být ihned na dalším řádku *meta-blok* s dodatečnými informacemi.

Po *meta-bloku* (případně rovnou po *objektu*, pokud *meta-blok* není přítomen) může být libovolné množství *bloků*. *Bloky* jsou od sebe odděleny dvěma prázdnými mezerami a jsou odsazeny od úrovně *objektu* dvěma mezerami. Ukázka *Bloku* se dvěma *objekty* je ve Výpisu 1.2.

1.4.3 Výrazy *vnitřní* a *vnější prostředí*

Nakonec lze označit text pomocí *prostředí*. *Prostředí* se dělí na dva typy – *vnější prostředí* a *vnitřní prostředí*. Obě *prostředí* mají stejnou funkci, liší se jen ve způsobu označení. Podobně jako *objekty* se *prostředí* označují pomocí jejich typu. Na rozdíl od *objektů* ale typ začíná malým písmenem.

1.4.3.1 Výraz *vnitřní prostředí*

Vnitřní prostředí se hodí k označení textu obsaženého v delším textu, například pro označení jednoho slova uprostřed věty. Výraz je možné zapsat dvěma způsoby. Buď krátce formátem `... text text vnitrniProstredi:telo text text ...`, nebo delším způsobem, ve formátu `... text text "telo telo".vnitrniProstredi text text ...`. V obou ukázkách představuje slovo `telo` obsah *prostředí* a `vnitrniProstredi` typ *prostředí*.

Delší formát dovoluje přidat *meta-blok* k *prostředí*, ukázka připojení *metabloku* je ve Výpisu 1.3.

Dále existují dvě zkrácená zapsání využívající znaky # a @ místo teček. Jejich význam definují šablony.

```
... text text ...
... text "telo telo".vnitrniProstredi.1 text ...
... text text ...
  1:
    klic: hodnota


...
```

Výpis kódu 1.3 Ukázka *vnitřního prostředí* s *meta-blokem*.

```
!vnejsiProstredi:
  meta-blok

  text
  text
  text
  ...


...
```

Výpis kódu 1.4 Ukázka křehkého *vnějšího prostředí* s *meta-blokem*.

1.4.3.2 Výraz *vnější prostředí*

Vnější prostředí jsou vhodná pro delší samostatné části textu, jako je výpis kódu nebo dlouhá matematická rovnice. Značí se podobně jako *objekty*, jen začínají malým písmenem a mohou obsahovat vždy pouze jeden *blok*. Na začátku výrazu může kromě znaku tečky být i znak vykřičníku. Znak vykřičníku jako prefix *vnějšího prostředí* označí obsah *prostředí* jako *křehké*, knihovna WooWoo obsah nebude nijak zpracovávat a předá ho šabloně takové, jaké je. Ukázka textu ve křehkém *vnějším prostředí* je ve Výpisu 1.4.

Pokud nějaký *blok* nemá označení žádného *vnějšího prostředí*, je zpracován, jako kdyby byl ve *výchozím prostředí* definovaném v šabloně.

1.4.4 Speciální výrazy

WooWoo knihovna má dva speciální výrazy, výraz *include* a *komentář*. Tyto výrazy mají chování definované přímo knihovnou WooWoo.

Dle konzultací s tvůrcem WooWoo má výraz *include* formát `.include slozka/soubor.woo`, kde `.include` představuje identifikátor výrazu a `slozka/soubor.woo` představuje jeho parametr určující cestu k souboru ve formátu WooWoo. Výraz vloží obsah souboru v dané cestě na místo samotného výrazu.

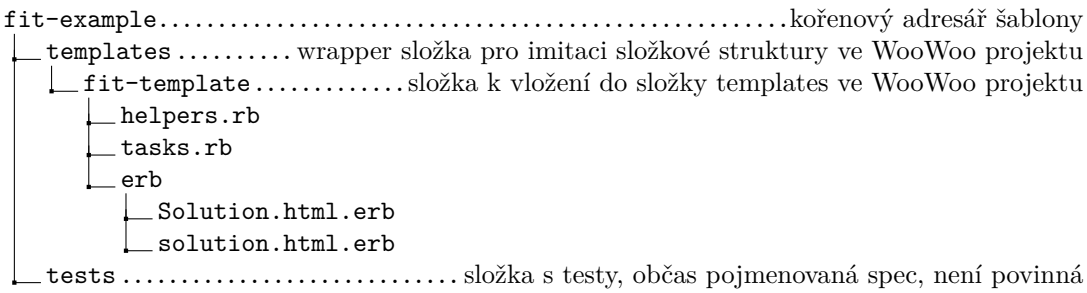
Komentář se značí znakem `%`. Knihovna WooWoo ignoruje text od `%` do konce řádku. Díky tomu lze do zdrojových souborů přidat například informace o licenci nebo komentáře ke zdrojovému kódu, aniž by je knihovna WooWoo předala šablonám.

1.5 Šablony

Jak již bylo zmíněno výše, pro WooWoo neexistuje mnoho dokumentace. Nikde není popsané, co je a co není šablona; je ale zřejmé, že šablony zpracovávají WooWoo soubory a vytvoří z nich reprezentaci jejich dat. V této práci je za šablonu WooWoo považován program, který využívá knihovnu WooWoo k vytvoření reprezentace dat z WooWoo souborů, je spustitelný pomocí knihovny Rake a využívá Woofile ve formátu YAML ke konfiguraci.

Šablony dodávají vzhled obsahu WooWoo souborů podle výrazů v Kapitole 1.4. Tvůrce dokumentů WooWoo tedy musí vědět, jaké šablony bude chtít využívat a určit podle nich typy výrazů. Například je vhodné znát, jaké označení nadpisů šablona očekává a podle toho označit nadpisy v dokumentu.

1.5.1 Standardní šablona



Obrázek 1.2 Ukázka složkové struktury pro šablonu jménem fit-example.

V této sekci je popsána standardní souborová struktura WooWoo šablony a předpokládaný obsah souborů. V případě jiné souborové struktury nebo obsahu souborů nemusí WooWoo knihovna využívaná šablonou fungovat správně.

Složková struktura standardní šablony je zobrazena na Obrázku 1.2. Složka šablony ve složce templates je při použití nakopírována do WooWoo projektu, kde se spouští pomocí Rakefile WooWoo projektu. V kořenovém adresáři jsou pak soubory týkající se vývoje šablony, jako například testy nebo soubory s návodem k užití.

Hlavní soubor šablon je již dříve zmíněný tasks.rb, který definuje chování celé šablony. Obsahuje Rake *tasks* s Ruby kódem, které mají za úkol vytvořit reprezentaci WooWoo souborů nebo je nějak analyzovat. Častý Rake *task* bývá *task build*, který zpracuje WooWoo soubory a vytvoří jejich reprezentaci, například PDF soubor.

Vzhled výrazů je typicky definován pomocí *Embedded RuBy* souborů, které jsou s pomocí WooWoo knihovny aplikovány na text označený výrazy *objekt* nebo *prostředí*. Jak je v Kapitole 1.4 popsáno, každý ze zmíněných výrazů má nějaký typ. Podle typu se vybere stejnojmenný *Embedded RuBy* soubor ze složky erb[11] a transformuje se podle něj text.

Například pro text ve *vnějším prostředí* typu *pragraph* se vybere soubor jménem paragraph.erb a aplikuje se na obsah *prostředí*. Ukázka, jak by soubor paragraph.erb mohl vypadat, je ve Výpisu 1.5. Výpis je převzat z fit-html šablony a lehce upraven. Jeho aplikováním se vloží obsah ve výrazu mezi `<p class="woowoo-paragraph">` a `</p>`.

Stejně jako typy *sekcí* a *objektů* začínají velkým písmenem, začínají i soubory aplikovatelné na *sekce* a *objekty* velkým písmenem. Malým písmenem pak začínají soubory aplikovatelné na *prostředí*. Toto rozlišování souborů malými a velkými písmeny může způsobovat problémy na operačních systémech, které nerozlišují velká a malá písmena u názvů. Mohou vzniknout konflikty u souborů typů *prostředí* a *objetk*, které se liší pouze velikostí počátečního písmena.

[11] Viz složková struktura na Obrázku 1.1

```
<p class="woowoo-paragraph">
  <%= r content %>
</p>
```

Výpis kódu 1.5 Ukázka *Embedded RuBy* souboru jménem paragraph.html.erb. Převzato z šablony fit-html, upraveno.

```
.Chapter Introduction
  label: chap-introduction

.Section Newton's method
  label: sec-newton

.Subsection Equations summary
  label: subsec-eq-summary
```

Výpis kódu 1.6 Ukázka označení kapitoly, sekce a podsekce dle specifikace *FIT-šablona*.

1.5.2 Specifikace *FIT-šablona*

Aby bylo možné využít více šablon pro jeden WooWoo soubor, vznikla specifikace *FIT-šablona*; ta předepisuje typy *sekcí*, *objektů* a *vnějších a vnitřních prostředí* a určuje, co mají představovat. Dále definuje povinná a nepovinná metadata očekávané v *meta-bloku* zmíněných výrazů.

V této práci je pro představu stručně rozebrána malá část specifikace. Nejnovější verze celé specifikace je dostupná pro studenty a zaměstnance ČVUT FIT na `https://gitlab.fit.cvut.cz/woowoo/fit-html/blob/master/README.md`. Zastaralejší, ale veřejně dostupná, specifikace je na `https://kam.fit.cvut.cz/deploy/woowoo-specs/woowoo-specs.pdf`.

***Sekce*:** Dle [2] strukturuje *FIT-šablona* dokument do kapitol, sekcí a podsekcí. Tyto výrazy mají jeden povinný argument *label*, který slouží jako identifikátor. Ve Výpisu 1.6 je ukázka kapitoly, sekce a podsekce.

***Objekty a Bloky*:** Bloky dle [2] ve *FIT-šabloně* představují paragrafy textu. Speciální typy *objektů* mohou být například `.Definition` nebo `.Corollary`. Více informací je ve specifikaci zmíněné výše.

***Prostředí*:** Mezi *vnější prostředí* dle [2] patří například `!codeblock`, který slouží k zobrazení většího množství kódu nebo `.enumerate`, které vytvoří očíslovaný seznam.

Vnitřní prostředí jsou například `.math`, nebo `.eqref`.

Zkrácené zápisy *Vnitřního prostředí*: Dle [2] znak # následovaný metadatem *label* nějakého výrazu vytvoří odkaz na daný výraz. Příklad takového zápisu je `"Text"#ref`. Slovo `ref` zde představuje hodnotu parametru *label* ve výrazu, ke kterému má vést odkaz a `Text` představuje zobrazovaný text.

Znak @ slouží jako odkaz mimo dokument. Má stejný význam jako *prostředí* typu *reference*. Zkrácený zápis je ve Výpisu 1.7.

```
"beautiful page".reference.1"
  1:
    url: 'link'

"beautiful page"@1"
  1: 'link'
```

Výpis kódu 1.7 Ukázka zkráceného zápisu *reference* pomocí zkratky @.

1.6 Relevantní existující WooWoo nástroje

1.6.1 Šablona fit-html

Šablona fit-html slouží ke zpracování WooWoo souborů dle specifikace *FIT-šablona*. Jejím výstupem je webová stránka[12].

Protože ze zadání má být i výsledná myšlenková mapa zobrazena jako webová stránka, mohou být styly šablony fit-html ze složky erb[13] znovupoužitelné i pro tuto práci.

1.6.2 Sémantické sítě

Je vhodné zmínit, že již existuje práce [4], která se zabývá znázorněním souvislostí výrazů. V jejím abstraktu je psáno následující. „Tato [myšlena práce [4]] bakalářská práce se zabývá zpracováním matematických pojmů do podoby sémantické sítě a využití těchto vztahů k obohacení výsledné vizualizace těchto matematických pojmů. Za tímto účelem byl vytvořen program zpracovávající matematická data ve specifickém formátu, extrahuje sémantické vztahy mezi pojmy a následně vytváří výstup ve webovém rozhraní.“

Cíl práce [4] je podobný jako cíl této práce. Liší se ale v tom, že práce [4] zpracovává pouze matematické pojmy a pouze pro předmět BI-ZMA, zatím co tato práce se snaží o obecnější zpracování. Další rozdíl je ve vizualizaci. V práci [4] bylo řečeno, že sémantickou mapu není vhodné vizualizovat graficky [myšleno jako myšlenkovou mapu] z důvodu vysokého počtu pojmů ve studijním textu. Mohlo by se ale najít využití pro vizualizaci pouze některých pojmů, což by ale kvůli chybějícím informacím omezilo její význam. Tato práce se i tak pokusí o takový způsob vizualizace.

Při konzultaci s Ing. Tomášem Kalvodou, Ph.D., vedoucím obou prací, bylo zjištěno, že Ing. Tomáš Kalvoda, Ph.D. vytvořil šablonu jménem fit-knowledge určenou pro automatickou extrakci výrazů a jejich referencí z WooWoo souborů. Šablona v práci [4] nakonec nebyla využita, místo toho práce spoléhala na ruční vytváření podkladů v jiném formátu.

[12] Ukázka výstupu šablony: `https://courses.fit.cvut.cz/BI-LA1/@master/textbook/`
[13] Složka popsána v Kapitole 1.2.

Kapitola 2
Myšlenková mapa

V této kapitole je definován pojem Myšlenková mapa, dále jsou prozkoumány některé nástroje pro její tvorbu a vizualizaci.

2.1 O myšlenkových mapách

Název „Myšlenková mapa“ vymyslel a popularizoval Tony Buzen v 70. letech 20. století v televizní show a sérii knih *Use Your Head*. Tam Buzen využíval myšlenkové mapy k vysvětlení jednoho konceptu, který typicky napsal do středu papíru a rozšiřoval ho různě se větvícími čarami s popisky vedoucími od středového konceptu.

Myšlenkové mapy byly ale využívané už dlouho před tím, dle [5] už například Leonardem da Vincim. Kvůli jejich dlouhé existenci není známý jejich vynálezce. Je možné, že v průběhu historie lidstva byla myšlenková mapa několikrát vynalezena a zase zapomenuta.

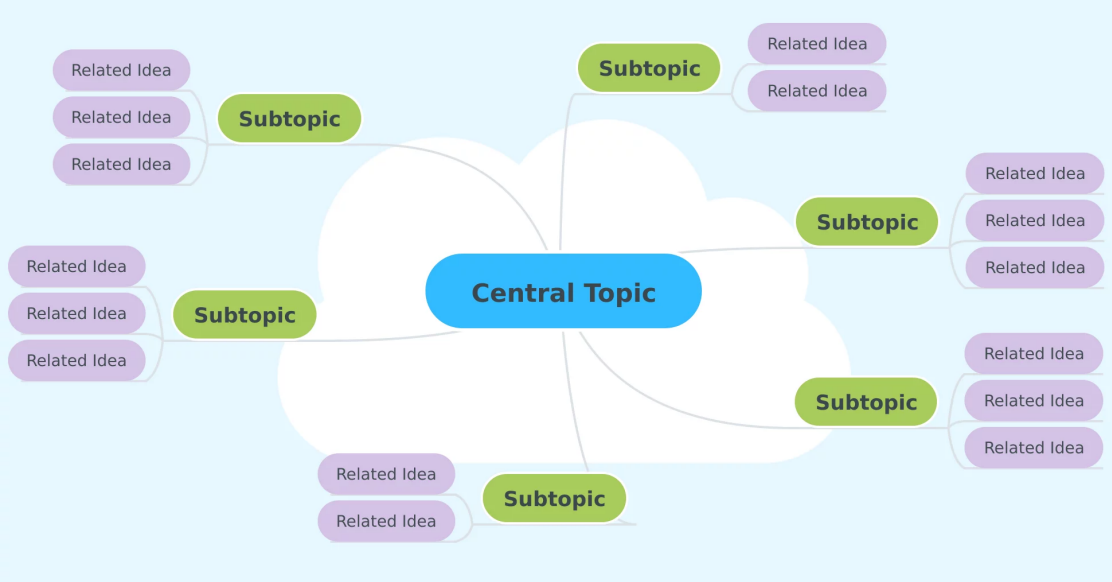
V nynější době se myšlenková mapa často využívá pro rozvedení jednoho informačního bodu. Tento bod bývá ve středu myšlenkové mapy a od něj se hierarchicky odvíjejí další úrovně pojmů doplňující tento hlavní pojem. Používá se jako nástroj zápisu informací při vymýšlení nových nápadů nebo řešení problémů při aktivitách jako je brainstorming.

Myšlenkové mapy ale nejsou užitečné jen pro zápis informací, dají se využít i ke studijním účelům. Bylo provedeno několik výzkumů řešících účinnost myšlenkových map jako studijních materiálů. Dle [6] na vysokých školách myšlenkové mapy sice nenahradí standardní učební metody jako jsou skripta, přednášky a cvičení, ale jsou užitečná pro spojení materiálů do souvislostí. Mohou sloužit k podobným účelům jako osobní konzultace, ale zaberou mnohem méně času pro vyučující.

2.1.1 Definice

Dle [7] je myšlenková mapa diagram využívaný k vizuální organizaci informací; myšlenková mapa je hierarchická a zobrazuje vztahy mezi jednotlivými informačními prvky.

Je ale důležité podotknout, že neexistuje shoda na jedné definici myšlenkové mapy. Například dle [8] je myšlenková mapa diagram reprezentující úkoly, slova, koncepty nebo předměty propojené a poskládané okolo centrálního konceptu v nelineárním rozložení, které dovoluje uživatelům sestavit intuitivní strukturu okolo centrálního konceptu. Podle této definice tedy musí existovat jeden centrální informační prvek, od kterého se odvíjejí ostatní. V této práci se budeme řídit první zmíněnou definicí kvůli její obecnosti. V této práci jsou vztahy nazývány také reference. Informační prvky jsou nazývány také jako pojmy nebo vrcholy. Ve WooWoo souborech jsou tyto pojmy a vrcholy označovány jako výrazy.



Obrázek 2.1 Ukázka klasické myšlenkové mapy.

2.2 Myšlenková mapa jako graf

Na myšlenkové mapy se dá pohlížet také jako na graf z teorie grafů, kde informační prvky představují vrcholy grafu a jejich vztahy představují hrany. Tento graf bývá většinou orientovaný, souvislý a acyklický. Orientace hran pak zpravidla směřuje od informačních prvků do pojmů na nich závislých, neboli prvků hierarchicky níže. Pohled na myšlenkovou mapu jako na graf může být užitečný například při strojové tvorbě rozložení pojmů v myšlenkové mapě. V sekcích níže jsou popsány některé algoritmy pro tvorbu rozložení grafu a pro určení důležitosti vrcholů.

2.2.1 Rozložení vrcholů

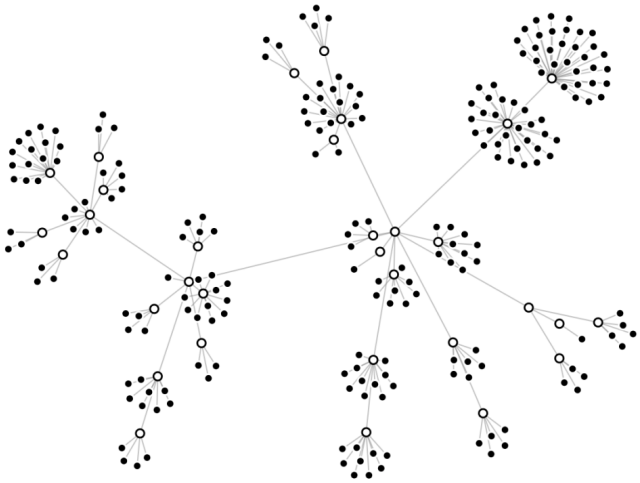
Určení vhodného rozložení grafu je stěžejní pro přehlednou myšlenkovou mapu. Problém rozložení grafu je řešen v oborech jako je informatika, bioinformatika, kartografie nebo sociologie. Existuje proto i mnoho různých typů rozložení, některé velmi specializované na určité obory, jiné zas poměrně obecné. Pro tvorbu myšlenkové mapy byly vybrány a dále rozebrány dva nejvhodnější algoritmy, které jsou dále popsány níže.

2.2.1.1 Force-Directed rozložení

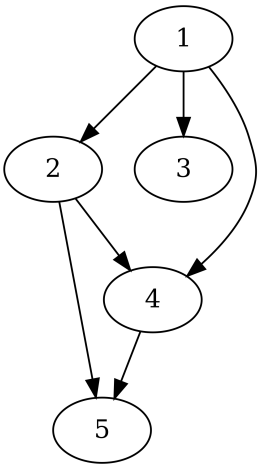
Algoritmy pro *Force-Directed* rozložení, občas nazývané *Spring layout* nebo pružinkový model. Dle [9] slouží k tvorbě rozložení neorientovaných grafů; grafy kreslené pomocí těchto algoritmů bývají esteticky pěkné a s malým množstvím překřížených hran. Algoritmy fungují na principu přidání přitažlivosti mezi propojenými hranami. Ukázka takových rozložení je na Obrázku 2.2a.

2.2.1.2 Hierarchické rozložení

Hierarchické rozložení je dle [10] vhodné pro zobrazení acyklických orientovaných grafů. Například při preferovaném směru hran odshora dolů by vrcholy měly být rozloženy horizontálně do řad a propojeny hranami směřujícími směrem dolů. Ukázka *hierarchického* rozložení vytvořeného pomocí nástroje Graphviz je na obrázku 2.2b.



(a) Ukázka pružinkového rozložení. [11]



(b) Ukázka hierarchického rozložení vytvořeného nástrojem Graphviz.

Obrázek 2.2 Srovnání pružinkového a hierarchického rozložení.

2.2.2 Důležitost vrcholů

Algoritmů pro určení důležitosti, občas také nazýváno *centrality*, existuje mnoho. Při analýze bylo vyzkoušeno mnoho různých algoritmů. Z nich byly vybrány 4 rozdílné způsoby určení důležitosti, každý fungující na odlišném principu pro pokrytí co největšího množství způsobů. Tyto algoritmy jsou hlouběji analyzovány níže.

2.2.2.1 *Degree centrality*

Degree centrality určuje důležitost vrcholů podle jejich stupně, tedy podle počtu příchozích a odchozích hran vrcholu. Vzorec pro výpočet vrcholu p je tedy velmi jednoduchý:

$$d_p = \deg(p)$$

$\deg(p)$ představuje stupeň vrcholu p. Dále mohou existovat variace pro určení důležitosti pouze pro příchozí nebo odchozí hrany.

2.2.2.2 *Betweenness centrality*

Dle [12] určuje *betweenness centrality* důležitost vrcholů podle počtu nejkratších cest mezi všemi vrcholy procházejících daným vrcholem. Vzorec pro výpočet důležitosti vrcholu v je:

$$b_v = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Kde V je množina všech vrcholů, $\sigma(s,t)$ je počet nejkratších cest z vrcholu s do vrcholu t, $\sigma(s,t|v)$ je počet nejkratších cest procházejících vrcholem v kde $v \neq s \wedge v \neq t$ a pokud $s = t$, pak $\sigma(s,t) = 1$.

2.2.2.3 *Pagerank*

Pagerank je známý především pro jeho využití vyhledávačem Google pro hodnocení webových stránek při vyhledávání.

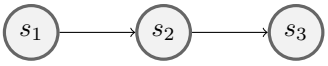
Dle [13] se důležitost vrcholů měří podle počtu příchozích hran a podle důležitosti vrcholů ze kterých tyto hrany vycházejí. Výpočet důležitosti vrcholu p je:

$$x_p = (1 - d) + d \sum_{q \in a[p]} \frac{x_q}{h_q}$$

Kde $d \in (0, 1)$, h_q je počet odchozích hran a $a[p]$ je množina vrcholů s hranou vedoucí do vrcholu p.

2.2.2.4 *Trophic levels*

Trophic levels se využívá při určování úrovně druhu v potravním řetězci. Čím vyšší hodnota, tím výš by měl druh v potravním řetězci být. Dle [14] je nejjednodušší *trophic* struktura a zároveň paradigma následující graf:



u kterého jsou *trophic levels* $s_1 = 0$, $s_2 = 1$ a $s_3 = 2$.

Dále dle [14] platí, že pokud do jednoho vrcholu vede více hran jejichž zdrojové vrcholy mají různé *trophic levels* nebo pokud graf obsahuje cykly, je k cestám třeba přiřadit váhy a podle nich pak vypočítat vážený průměr.

2.3 Nástroje pro tvorbu myšlenkových map

V této kapitole jsou popsány nástroje užitečné pro vytvoření myšlenkové mapy. Protože existuje jen malé množství nástrojů, které dokáží vytvořit rozložení myšlenkové mapy, určit důležitost jejich vrcholů, a nakonec i vše vizualizovat, jsou nástroje níže rozděleny do dvou sekcí. Sekce s nástroji pro vizualizaci a sekce s nástroji pro tvorbu rozložení a určení důležitosti. Určení důležitosti není nezbytné pro vytvoření myšlenkové mapy, je ale zmíněno v zadání této práce.

2.3.1 Vizualizace myšlenkové mapy

2.3.1.1 Orgpad

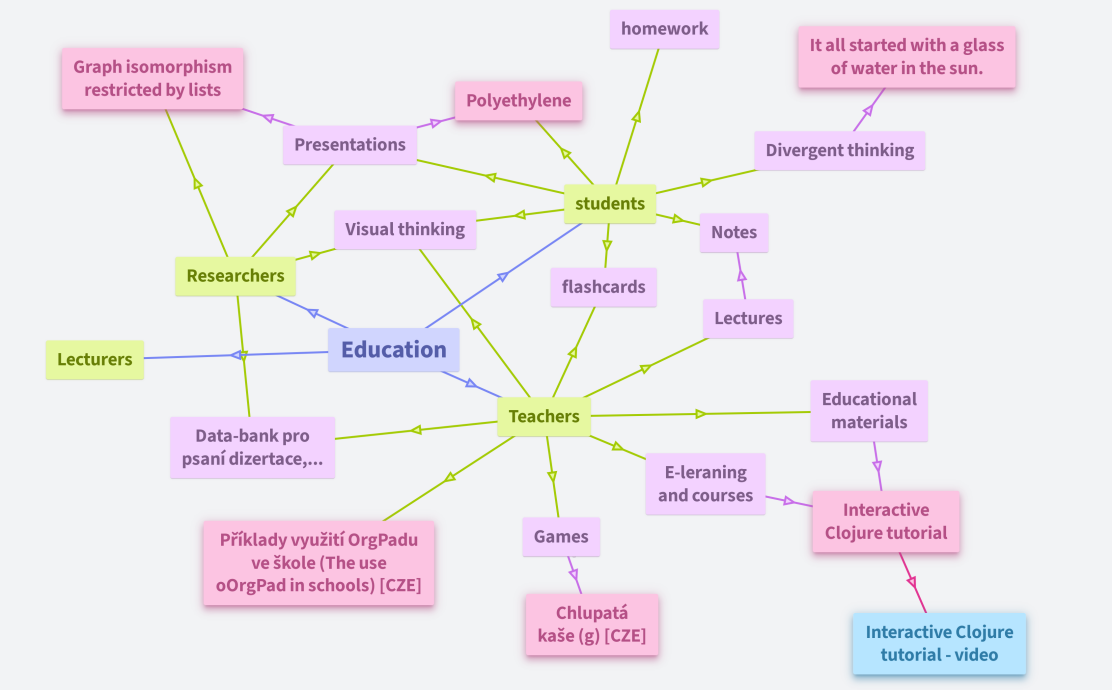
Orgpad[1] je *closed source* webová aplikace pro tvorbu myšlenkových map. *Closed source* znamená, že její kód není veřejný. Orgpad je veřejně dostupný jako webová stránka na adrese `www.orgpad.com`. Jako studijní materiál je doporučována například zaměstnancem ČVUT FIT Ing. Eliškou Šestákovou v [15]. Ukázka softwaru je zobrazena na Obrázku 2.3.

Orgpad je webová aplikace dostupná na všech zařízeních s webovým prohlížečem, je tedy funkční na počítačích i na mobilních telefonech a tabletech. Aplikace je určena k manuálnímu použití. To znamená, že nemá dokumentované API pro import dat z externích zdrojů.

Aplikace Orgpad nepodporuje automatickou tvorbu rozložení pojmů. U pojmů není možné měnit jejich velikost, lze ale upravit barvu pozadí. Mezi pojmy se dá pohybovat myší, případně prstem na dotykové obrazovce. Pojmy lze kliknutím přepínat mezi minimalizovaným a maximalizovaným zobrazením. Není možné zobrazovat speciální matematické výrazy jako jsou zlomky, do obsahu pojmu je ale možné vkládat obrázky, ve kterých by mohly tyto výrazy být zobrazeny.

Vytvořené myšlenkové mapy je možné sdílet s ostatními uživateli pomocí odkazu. Verze zdarma dovoluje jednomu uživateli vytvořit maximálně 3 myšlenkové mapy. Pro vytvoření většího množství je potřeba zakoupit licenci.

[1] Více o Orgpad na `https://www.orgpad.com/`



Obrázek 2.3 Ukázka softwaru Orgpad. [16]

2.3.1.2 Obsidian

Obsidian[2] je *closed source* textový editor vizualizující reference v textu pomocí interaktivního grafu. Ukázka editoru je na Obrázku 2.4. Jeho primární funkce slouží jako zápisník, zobrazení do grafu je spíše doplňující funkce.

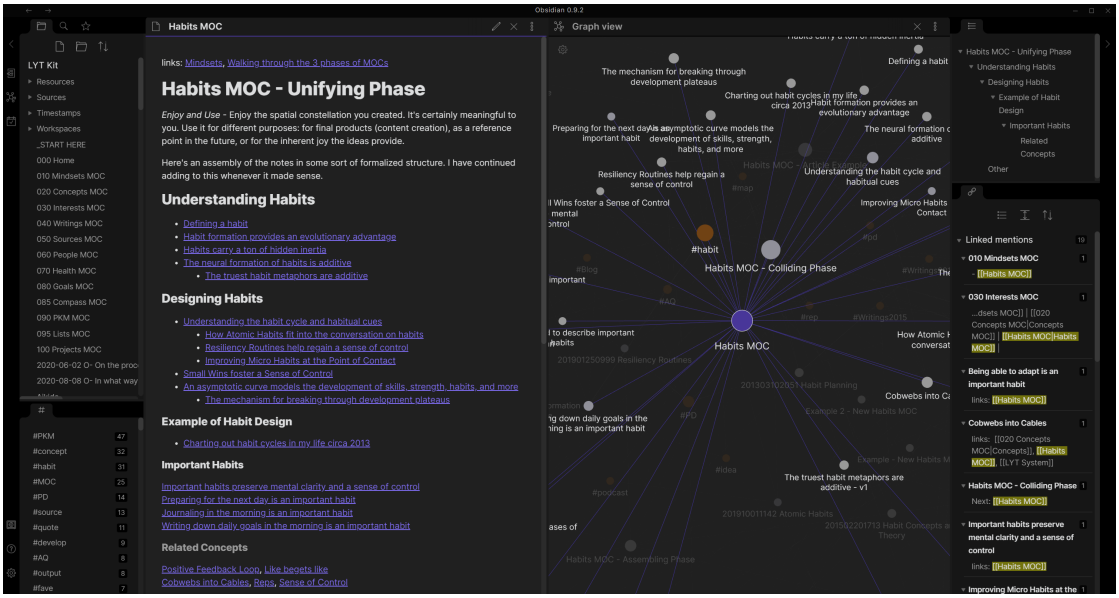
Jedná se o multiplatformní aplikaci funkční na operačních systémech Windows, Mac, na Linuxových systémech a na mobilních zařízeních Android a iOS. Dále je možné publikovat dokumenty na web za měsíční poplatek.

Zdrojové kódy jsou v textovém formátu, tudíž jsou čitelné i bez aplikace. Je možné zobrazit obrázky a speciální matematické symboly pomocí integrované knihovny MathJax. Dále je možné tvořit odkazy na jiné pojmy v jiných souborech. Podle těchto odkazů se pak vytvoří myšlenková mapa.

Rozložení pojmů v myšlenkové mapě nelze nijak definovat, vytvoří se automaticky ve stylu *force-directed* rozložení. Algoritmus pro tvorbu rozložení je nedeterministický, tudíž po každém spuštění vypadá rozložení jinak. Důležitost pojmů v myšlenkové mapě je reprezentována velikostí a určena algoritmem *degree centrality*. Nástroj Obsidian nedovoluje vlastní způsob určení důležitosti. Samotné zobrazení grafu je uživatelsky konfigurovatelné, lze upravovat sílu přitažlivosti vrcholů, barvu vrcholů, jejich velikost a velikost nadpisů.

V myšlenkové mapě editoru Obsidian je na první pohled obtížné zjistit, jaké pojmy patří mezi ty základní – na kterých je (i nepřímo) závislých mnoho dalších pojmů – a jaké jsou jedny z posledních – tedy není na nich závislých mnoho pojmů.

[2] Více informací o editoru na `https://obsidian.md/`



Obrázek 2.4 Ukázka editoru Obsidian. [17]

2.3.1.3 D3

Nástroje zmíněné výše jsou již hotové aplikace. D3[3] je oproti nim knihovna. Dle [18] je psaná v jazyce JavaScript a slouží k manipulaci s dokumenty podle dat. D3 zobrazuje data pomocí HTML, SVG a CSS. Například dovoluje jednoduše vytvořit jeden prvek HTML pro každý prvek v nějakém poli dat.

Knihovna je velmi flexibilní, díky tomu je možné vytvořit cokoliv, co je možné vytvořit se standardní kombinací JavaScriptu, HTML, CSS. Ukázka grafu vytvořeného pomocí knihovny D3 je zobrazena na Obrázku 2.5.

Dle [18] podporuje D3 všechny *moderní* prohlížeče, tedy vše, kromě prohlížeče Internet Explorer 8. Je testována na prohlížečích Firefox, Chrome, Safari, Opera, Internet Explorer 9+, Android a iOS.

2.3.1.4 Dracula.js

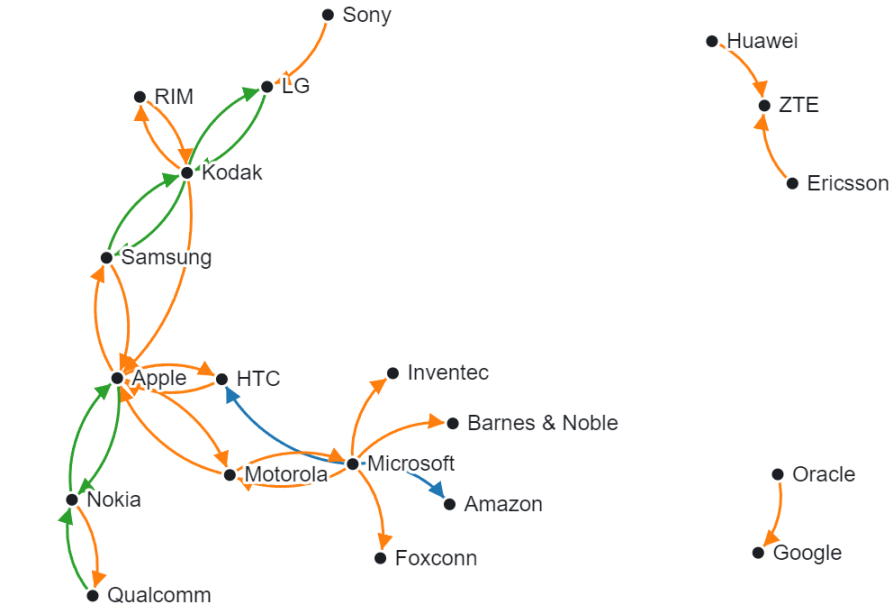
Dracula.js[4] je také knihovna v jazyce JavaScript. Na rozdíl od D3 je ale zaměřená přímo na vizualizaci grafů. Dracula.js je souhrn nástrojů pro zobrazení a rozložení grafů a sítí, dále obsahuje různé související algoritmy z teorie grafů.

Knihovna není dokumentovaná. Z vlastního testování bylo zjištěno, že funguje minimálně na Chromium prohlížečích a v prohlížeči Firefox. Na dotykových zařízeních je chování lehce omezené v tom, že nelze přesouvat vrcholy grafu.

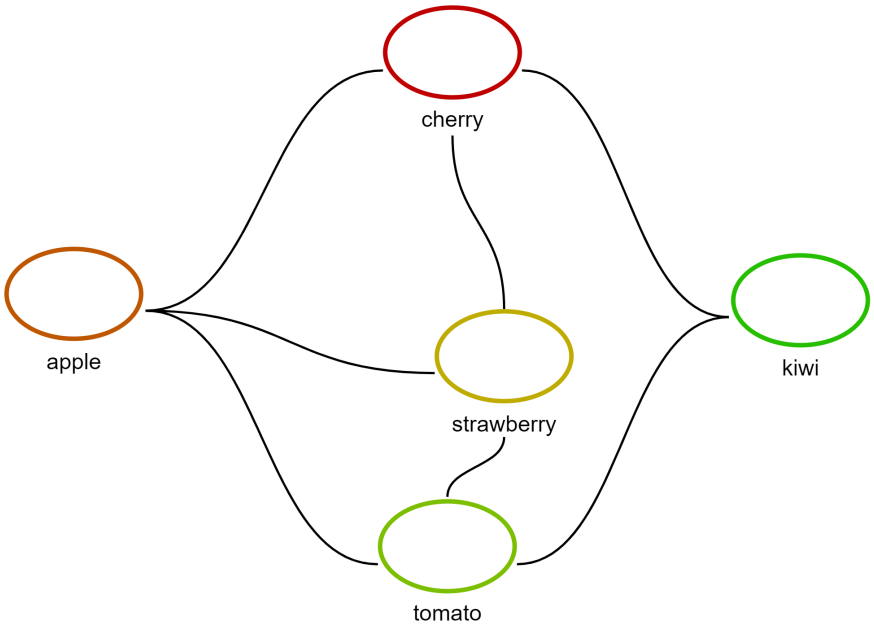
Grafy jsou vykreslovány do formátu SVG. Knihovna nedovoluje zásadní změny ve vykreslování prvků. Samotná struktura a vzhled vrcholů i hran je uzamčena v knihovně. Ukázka zobrazení grafu je na Obrázku 2.6.

Pro rozložení grafů využívá knihovna *force-directed rozložení*, které je ale nedeterministické. Je ovšem možné použít rozložení vytvořené jiným nástrojem.

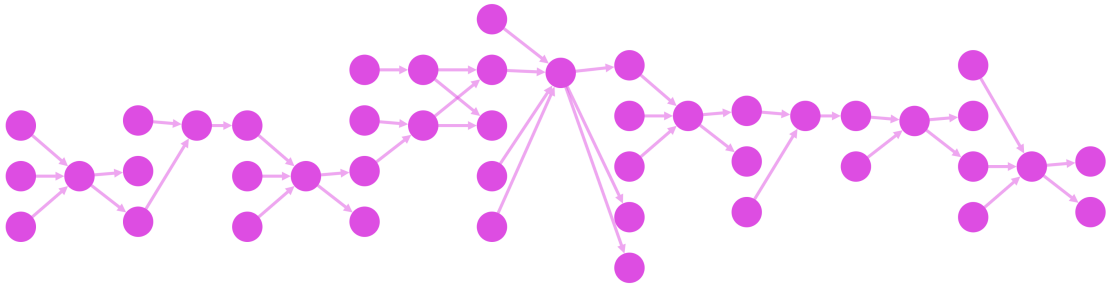
[3] Více info o D3 na `https://d3js.org/`
[4] Více informací o Dracula.js na `https://www.graphdracula.net/`



Obrázek 2.5 Ukázka grafu vytvořeného pomocí knihovny D3. [19]



Obrázek 2.6 Ukázka grafu vytvořeného pomocí knihovny Dracula.js. [20]



Obrázek 2.7 Ukázka rozložení vytvořeného pomocí Klay a zobrazeného pomocí Cytoscape. [21]

2.3.1.5 Cytoscape

Cytoscape je *open source* program určený pro vizualizaci komplexních sítí. Primární využití slouží pro analýzu genů v biologii. Cytoscape je samostatný program určený podobně jako Orgpad pro uživatelské využití. Existuje ale i knihovna Cytoscape, určená pro strojové vytváření grafů a jejich rozložení v jazyce Javascript.

Knihovna Cytoscape obsahuje algoritmy pro tvorbu rozložení a určení váhy, zároveň umí zobrazit graf na webové stránce. Více o tvorbě rozložení a určování důležitosti v Sekci 2.3.2.4. Sama o sobě umí knihovna vykreslovat graf do HTML *canvas*. Dále dovoluje vytvořit vlastní způsob vykreslování. Neexistuje dokumentace pro tvorbu vlastních vykreslovačů.

2.3.2 Rozložení a důležitost vrcholů

V této kapitole jsou rozebrány nástroje určené pro tvorbu rozložení a pro určení důležitosti vrcholů. Na začátku každé podkapitoly je uvedeno, zda nástroj umí vytvářet rozložení a zda umí určit důležitost vrcholů.

2.3.2.1 D3-force

Rozložení grafu: ANO

Důležitost vrcholů: NE

Knihovna D3 force slouží jako rozšíření knihovny D3. Dle [11] vytváří *force-directed* rozložení grafu pomocí algoritmu *Velocity Verlet integration*, což je algoritmus využívaný k výpočtu kinematických rovnic pro tělesa.

Výhody knihovny jsou v tom, že je integrována s knihovnou D3, běží tedy v aplikaci klienta. Díky tomu je možné za běhu aplikace skrývat a zobrazovat některé vrcholy a upravovat podle toho rozložení všech vrcholů. Knihovna například dovoluje pohybovat s vrcholy tak, že na to ostatní vrcholy také reagují.

Nevýhodou je to, že knihovna nepodporuje kromě *force-directed* žádné jiné rozložení.

2.3.2.2 Graphviz

Rozložení grafu: ANO

Důležitost vrcholů: NE

Dle [22] je Graphviz *open source* software pro vizualizaci grafů, k zápisu grafů používá jazyk DOT[5], který slouží k popisu grafů a jejich podgrafů, vrcholů, hran a k určení vzhledu vrcholů

[5]Kapitalizace je důležitá, dot psáno malými písmeny představuje generátor rozložení.

a hran. V praxi jsou grafy většinou zapsány do formátu DOT externími aplikacemi. Formát DOT ale může být upraven i ručně.

Graphviz nabízí 8 generátorů rozložení popsaných níže:

Dot generátor rozložení slouží k tvorbě hierarchických grafů. Více o hierarchických grafech v Sekci 2.2.1.2. Ukázka grafu vytvořeného pomocí tohoto generátoru je na Obrázku 2.2b.

Neato slouží k tvorbě pružinkových modelů. Více informací v Sekci 2.2.1.1. Neato generátor je vhodný pro nepříliš velké grafy s cirka 100 vrcholy.

Twopi vytváří kruhové zobrazení grafů. Všechny vrcholy uspořádá do jednoho kruhu.

Circo podobně jako *Twopi* vytváří kruhové zobrazení. Vrcholy ale může uspořádat do více kruhů.

Fdp je podobné jako Neato, je vhodné pro větší grafy.

Sfdp je fdp generátor pro ještě větší grafy.

Osage slouží pro tvorbu rozložení nesouvislých grafů.

Patchwork lze také využít pro rozložení nesouvislých grafů. Vrcholy grafů jsou vykresleny jako obdélníky přilehlé k sobě.

Dle [22] je možné Graphviz používat pomocí příkazové řádky. Vstupní graf se zadá pomocí formátu DOT, výstup může být v mnoha formátech, například jako obrázek ve formátu JPEG nebo jako objekt ve formátu JSON.

Nástroj Graphviz lze ovládat buď v uživatelském rozhraní, nebo pomocí příkazové řádky. Existují ale i API pro komunikaci s příkazovou řádkou v různých programovacích jazycích. Například pro jazyk Python existuje knihovna NetworkX popsaná v následující kapitole a pro Ruby knihovna Ruby-graphviz[6].

Je ale důležité zmínit, že vývoj knihovny Ruby-graphviz není moc aktivní, například zatím nedovoluje export do formátu JSON, což samotný nástroj Graphviz umí.

2.3.2.3 NetworkX

Rozložení grafu: ANO

Důležitost vrcholů: ANO

Knihovna NetworkX je pro jazyk Python. Dle [23] slouží pro tvorbu, manipulaci a studium struktury, dynamiky a funkcí komplexních sítí. S pomocí NetworkX lze ukládat sítě v mnoha formátech, generovat mnoho různých typů sítí, analyzovat strukturu sítí, tvořit síťové modely, navrhovat nové síťové algoritmy, vykreslovat sítě a mnoho dalšího.

Dle [23] má knihovna základní možnosti pro vizualizaci grafů, ale její primární zaměření je spíše na jejich analýzu. I přesto má NetworkX několik způsobů vykreslování například pro tvorbu *force-directed* rozložení, náhodného rozložení nebo planárního rozložení. Pro pokročilejší vykreslování využívá NetworkX nástroj Graphviz. Je tedy možné vizualizovat grafy nástrojem Graphviz s pomocí NetworkX.

Pro určení důležitosti vrcholů v grafu jsou v knihovně implementovány mimo jiné i algoritmy *degree centrality, betweenness centrality, pagerank a trophic levels* popsané v Sekci 2.2.2.

[6] Více o knihovně na `https://github.com/glejeune/Ruby-Graphviz/`

2.3.2.4 Cytoscape

Rozložení grafu: ANO

Důležitost vrcholů: ANO

Cytoscape již byla zmíněna v Sekci 2.3.1.5. Samotná knihovna Cytoscape implementuje pouze základní nástroje pro tvorbu rozložení, dovoluje ale použití externích rozšíření. Mezi těmito rozšířeními je například lehce upravená knihovna D3-force popsaná v Sekci 2.3.2.1 nebo knihovna Klay, která slouží pro tvorbu hierarchického rozložení. Ukázka jejího výstupu je na Obrázku 2.7.

Pro určení důležitosti vrcholů implementuje knihovna mimo jiné i algoritmy *degree centrality, betweenness centrality a pagerank* popsané v Sekci 2.2.2.

Kapitola 3
Požadavky

V této kapitole jsou stanoveny funkční a nefunkční požadavky a dále jsou identifikovány způsoby využití (také nazýváno use cases nebo případy využití).

Funkční požadavky:

F1 Zobrazení pojmů (neboli výrazů) z WooWoo souborů.

F2 Zobrazení souvislostí mezi pojmy.

F3 Vytvoření přehledného rozložení pojmů a souvislostí do myšlenkové mapy.

F4 Interaktivní přecházení mezi pojmy podle závislostí.

F5 Znázornění důležitosti pojmů.

F6 Správná interpretace WooWoo výrazů včetně matematických symbolů dle specifikace *FIT-šablona*.

F7 Oddalování a přibližování v myšlenkové mapě.

F8 Zjištění důležitosti pojmů.

Nefunkční požadavky:

N1 Zobrazení myšlenkové mapy na počítačích i na mobilních telefonech.

N2 Zobrazení myšlenkové mapy pomocí webových technologií.

N3 Zpracování WooWoo souborů na systémech Linux nebo pomocí technologie Docker.

N4 Myšlenková mapa musí být automaticky vytvořitelná z WooWoo souborů.

Způsoby využití

Aplikace má pouze dva způsoby využití.

U1 Zobrazení a používání myšlenkové mapy.

U2 Vytvoření myšlenkové mapy z WooWoo souborů.

Tyto dva případy využití budou využívat dva aktéři.

Uživatel: Aktér Uživatel bude typicky student, využívající aplikaci ke studiu. Jeho jediný případ využití je U1.

Tvůrce: Aktér Tvůrce je generalizace aktéra Uživatel. Tvůrce má tedy případ využití U1 a dále i U2. Těchto aktérů bude malé množství, budou to buď vyučující, nebo počítače.

Kapitola 4

Návrh

V této kapitole jsou nejprve vybrány algoritmy a nástroje podle analýzy v Kapitole 2, následně je podle těchto nástrojů vytvořen návrh.

Protože je ze zadání vyžadována šablona WooWoo, je zřejmé, že bude aplikace rozdělena na šablonu zpracovávající WooWoo soubory a na klienta zobrazujícího data vytvořené šablonou. S tímto rozdělením se počítá v následujících kapitolách.

4.1 Výběr algoritmů

4.1.1 Rozložení vrcholů

Způsobů a nástrojů pro tvorbu rozložení grafů existuje mnoho. Jen nástroj Graphviz obsahuje osm algoritmů pro tvorbu rozložení. Když se ale vyřadí algoritmy pro vytváření kruhů z cyklických grafů, pro uspořádání nesouvislých grafů a pro tvorbu velmi specializovaných a pro myšlenkovou mapu nepoužitelných zobrazení, zbydou pouze algoritmy pro tvorbu hierarchických nebo pružinkových grafů. Podobně je to i u ostatních nástrojů, jako je například Cytoscape.

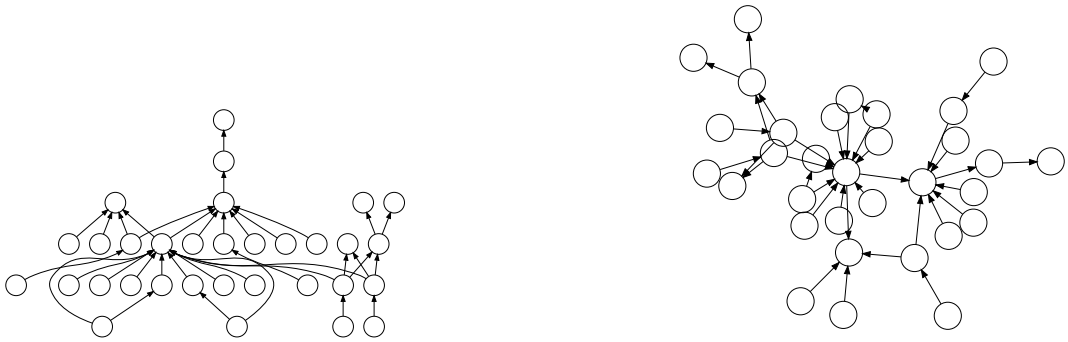
Pružinkové a hierarchické způsoby rozložení jsou porovnány na Obrázku 4.1 na datech z prvních dvou kapitol předmětu BI-MA1.

Rozhodování není nijak omezeno nástroji, dle analýzy v Sekci 2.3.2 je dostupných dost nástrojů pro tvorbu obou rozložení, a to i na straně klienta, i na straně šablony.

Výhoda pružinkového způsobu rozložení je, že lépe poukazuje na pojmy s vysokým množstvím závislých pojmů – tedy vrcholy s velkým množstvím vstupních hran. Tyto pojmy pak typicky zobrazuje doprostřed grafu. Dále dobře znázorňuje provázanost vrcholů a člení provázané vrcholy do skupin.

Na druhou stranu hierarchické rozložení, zobrazené na Obrázku 4.1a, je, jak název napovídá, dobré ke zvýraznění hierarchie v grafu.

Po zvážení obou možností bylo vybráno hierarchické rozložení. Dobře zobrazuje nejzákladnější pojmy bez mnoha závislostí na ostatních pojmech v horní části mapy na rozdíl od pružinkového zobrazení, které zdůrazňuje pojmy s velkým množstvím závislých pojmů do středu mapy. Dále by myšlenková mapa dle vybrané definice měla zdůrazňovat hierarchii pojmů, což dělá hierarchické zobrazení velmi dobře.



(a) Hierarchické rozložení.

(b) Pružinkové rozložení.

Obrázek 4.1 Srovnání hierarchického a pružinkového rozložení na datech z prvních dvou kapitol předmětu BI-MA1.

4.1.2 Důležitost vrcholů

Algoritmy analyzované v Sekci 2.2.2 lze rozdělit do dvou kategorií. Na algoritmy, které při výpočtu váhy vrcholu berou v potaz váhu ostatních vrcholů, a na algoritmy, které ji v potaz neberou. *Betweenness centrality* a *degree centrality* patří do kategorie, která váhu ostatních vrcholů nebere v potaz, a algoritmy *pagerank* a *trophic levels* patří do kategorie algoritmů, které je v potaz berou.

Na Obrázku 4.2 jsou zobrazeny porovnávané algoritmy využívající stejná data, jako při výběru rozložení v Sekci 4.1.1. Váhy jsou normalizované lineární transformací na interval $\langle 0, 100 \rangle$, nejnižší hodnota je tedy 0 a nejvyšší 100.

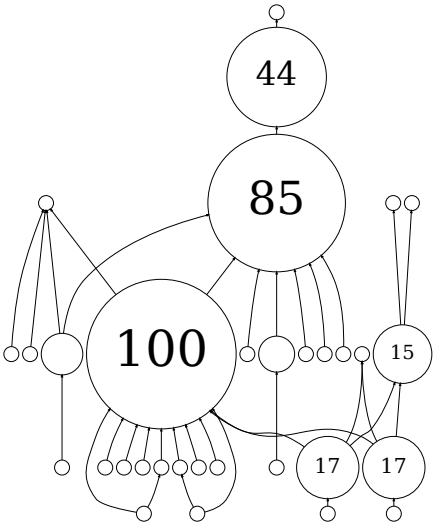
Stejně jako pro nástroje na tvorbu rozložení je i pro výpočet důležitosti dostatečné množství nástrojů. Jen pro algoritmus *trophic levels* existuje implementace pouze v knihovně NetworkX. Ostatní rozložení lze vytvářet i na straně klienta ve webovém prohlížeči, i při extrahování dat z WooWoo souborů.

Při rozhodování byl nejprve vyloučen algoritmus *degree centrality*, protože nepřináší žádnou přidanou hodnotu k přehlednosti. Zvýrazňuje pouze počet referencí pojmu, což je na první pohled viditelné i bez dalšího zvýraznění.

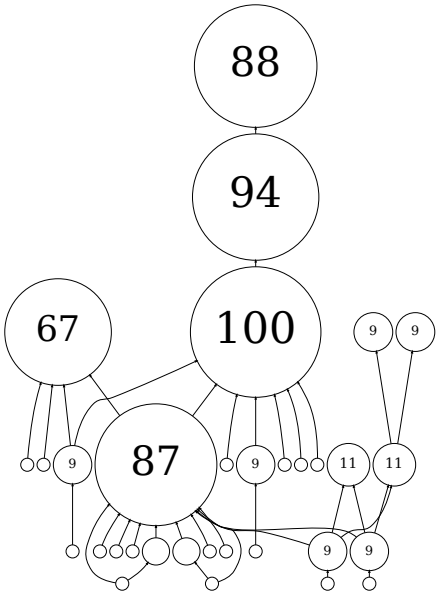
Dále ani algoritmus *betweenness centrality* není pro účely této práce ideální. Nezvýrazňuje důležité vrcholy, přes které sice nevedou žádné cesty, ale mnoho cest v nich končí. Příkladem takového vrcholu je nejvýše položený vrchol na Obrázku 4.2a. Dále jsou algoritmem zvýrazňovány nejvíce vrcholy uprostřed cest, od nich pak důležitost oběma směry podobnou mírou ustupuje. Pro účely myšlenkové mapy by bylo vhodnější, kdyby nejdůležitější vrcholy byly ty, ve kterých je zakončeno mnoho cest.

Ze zbývajících dvou možností byl po konzultaci s vedoucím práce nejprve vybrán algoritmus *trophic levels*, který je zobrazen na Obrázku 4.2d. Hlavní důvod pro výběr *trophic leves* oproti algoritmu *pagerank* byl ten, že *pagerank* dával příliš velkou váhu pojmům s velkým množstvím závislostí a téměř žádnou všem ostatním; změna parametru d nepomohla. Na druhou stranu algoritmus *trophic levels* váhu vrcholů průběžně zvyšoval.

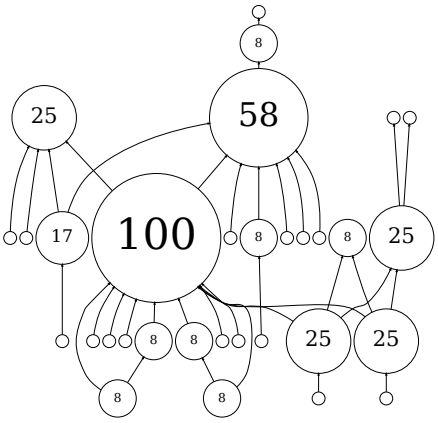
Po dalším zvážení bylo ale z algoritmu *trophic levels* ustoupeno kvůli neintuitivnímu chování pro využití v myšlenkové mapě. Protože se při určování důležitosti vrcholu dělá průměr důležitostí vstupních vrcholů, může dojít k tomu, že přidáním vstupních vrcholů s malou důležitostí klesne důležitost. Příklad tohoto chování je uveden na Obrázku 4.3, kde jsou k červenému vrcholu na Obrázku 4.3a přidány další dva vrcholy, čímž důležitost klesne. Výsledek je zobrazen na Obrázku 4.3b. Při tom by měla důležitost červeného vrcholu na Obrázku 4.3b stoupnout, protože se zvýší počet závislých vrcholů.



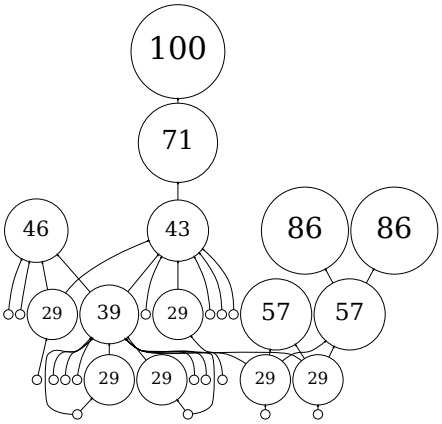
(a) *Betweenness centrality*



(b) *Pagerank*, $d = 0.85$

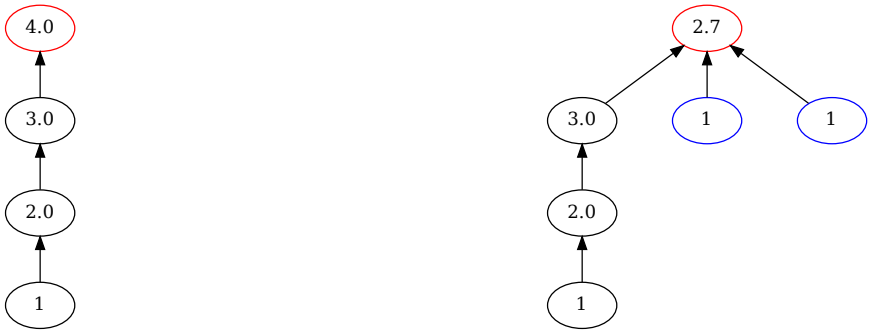


(c) *Degree centrality*



(d) *Trophic levels*

Obrázek 4.2 Srovnání algoritmů pro určení důležitosti na datech z prvních dvou kapitol předmětu BI-MA1, vyšší čísla znamenají vyšší důležitost, normalizováno na interval $\langle 0, 100 \rangle$.



(a) Vrchol před přidáním dalších závislostí.

(b) Vrchol po přidání závislostí.

Obrázek 4.3 Chování algoritmu *trophic levels*.

Proto bylo nakonec zvoleno využití algoritmu *pagerank*. Pro vyřešení problému s příliš velkým rozdílem váhy bude využita nelineární transformace – každá váha bude odmocněna třetí odmocninou. Tím se velké váhy zmenší více než malé, a jejich velikost se tak přiblíží. Hodnoty pak budou normalizovány na interval $\langle 0, 100 \rangle$.

4.2 Porovnání nástrojů

V této kapitole jsou porovnány nástroje zmíněné v Kapitole 2.3. U nástrojů je zváženo, zda umožňují vytvořit hierarchické rozložení a určit důležitost algoritmem *pagerank*. Následně jsou nástroje porovnávány podle toho, zda splňují funkční a nefunkční požadavky definované v Kapitole 3, podle jejich flexibility a možnosti budoucích úprav a podle toho, zda jsou dokumentované.

Požadavky se dají rozdělit do tří kategorií. Na požadavky týkající se vizualizace (F1, F2, F4, F5, F6, N1, N2), tvorby rozložení (F3) a určení důležitosti (F8). Nakonec zbývá nefunkční požadavek N4, který se týká všech kategorií. Každý nástroj dle něj musí mít možnost automaticky pracovat s daty bez ručního zasažení, například musí mít rozhraní do příkazové řádky nebo jiné přístupné API.

4.2.1 Nástroje pro vizualizaci

Nástroje pro vizualizaci by měly splňovat funkční požadavky F1, F2, F4, F5, F6 a nefunkční požadavky N1, N2. Budou využívány aktérem Uživatel i aktérem Tvůrce.

4.2.1.1 Orgpad

Protože je Orgpad webová aplikace, která funguje na počítačích i na mobilních zařízeních, jsou nefunkční požadavky N1 a N2 splněny. Problém může nastávat u automatické tvorby myšlenkových map z dat z WooWoo souborů. Pro splnění požadavku N4 by bylo nutné komunikovat s nedokumentovaným API, které by se v budoucnosti klidně mohlo změnit.

Funkční požadavky F1, F2 a F7 jsou splněny bez jakékoliv potřebné implementace. Po referencích nelze interaktivně přecházet jedním kliknutím, požadavek F4 tedy není splněn. Důležitost pojmů v požadavku F5 lze vyznačit například počtem vykřičníků v nadpise pojmu. Požadavek je tedy splněn.

Speciální výrazy dle požadavku F6 mohou být zobrazeny jako obrázek. Obrázky je ale potřeba někde generovat, což by zkomplikovalo vývoj. Orgpad umožňuje vlastní rozložení pojmů, je tedy možné docílit hierarchického rozložení.

S ohledem na složitost je sice samotný nástroj pro vizualizaci již hotový, není ale zamýšlen pro automatické generování myšlenkových map. Import dat by tedy byl velmi komplikovaný a mohl by po změnách nástroje přestat fungovat.

Flexibilita není vysoká, kontrolu nad budoucností nástroje mají jeho vývojáři, takže hrozí, že některé potřebné funkce budou v budoucnosti odebrány nebo změněny.

4.2.1.2 Obsidian

Nástroj Obsidian nedovoluje určení vlastního rozložení a jediné vestavěné je typu pružinkového rozložení. Nevyhovuje tedy rozhodnutí o způsobu rozložení ze Sekce 4.1.1. Proto ho není třeba dále porovnávat.

4.2.1.3 D3

Nefunkční požadavky N1 a N2 jsou splněny, protože je knihovna D3 funkční a testovaná na všech moderních prohlížečích i na dotykových obrazovkách. Knihovna je přímo dělaná pro zobrazování dat z jiného zdroje, požadavek N4 je tedy splněn.

Funkčních požadavků F1, F2, F4, F5 a F7 je možné dosáhnout využitím D3 s pomocí Javascriptu, HTML, CSS a SVG. Samotná knihovna D3 nedokáže zobrazit speciální matematické výrazy, ani k tomu není určena. Protože jde o knihovnu, je možné zkombinovat ji s dalšími knihovnami, které mohou sloužit ke zobrazení speciálních výrazů. Tím pádem se požadavek F6 dá splnit například využitím knihovny MathJax, použité i v šabloně fit-html.

Knihovna je velmi flexibilní a dobře dokumentována. Dovoluje určení vlastního rozložení i vlastního určení důležitosti dle výběru z Kapitoly 4.1. S vysokou flexibilitou ale přichází i delší doba vývoje.

4.2.1.4 Dracula.js

Požadavky N1 a N2 jsou díky zobrazení jako webová aplikace splněny, a to i přes to, že na dotykových displejích není možné přesouvat pojmy myšlenkových map. Požadavek N4 je splněn, protože lze s Dracula.js manipulovat s pomocí Javascriptu.

Funkčních požadavků F1, F2, F4, F5 a F7 lze podobně jako u knihovny D3 dosáhnout společně s pomocí Javascriptu. Protože je knihovna Dracula.js specializovaná na vykreslování grafů, jsou požadavky F1 a F2 implementačně jednodušší k vytvoření minimálního funkčního prototypu, pro další rozšiřování ale může uzavřené fungování knihovny spíše škodit. Je možné, že k dosažení uživatelsky přívětivého a hezkého zobrazení by bylo potřeba upravit samotné zdrojové kódy knihovny.

Stejně jako u D3 je splněn i požadavek F6 pomocí přidání další knihovny. Dále knihovna umí vytvářet rozložení, jde ale pouze o pružinkové rozložení, které nevyhovuje výběru z Kapitoly 4.1. Rozložení lze ale určit i z externích zdrojů.

Celkově je knihovna kvůli specifickému zaměření méně flexibilní. Vývoj základních prvků jako zobrazení vrcholů a referencí by byl rychlý, kvůli uzamčení vzhledu vrcholů grafu do samotné knihovny by ale mohlo být potřeba mnoho kompromisů nebo zásahů do fungování knihovny, což by celkový vývoj prodloužilo.

4.2.1.5 Cytoscape

Nástroj splňuje požadavky N1 a N2, protože se jedná o webovou aplikaci dostupnou a funkční i na dotykových zařízeních.

Knihovna dovoluje využití externích dat, požadavek N4 je tedy splněn. Je možné definovat vlastní polohu pojmů i jejich velikost, je tedy možné docílit hierarchického rozložení a nastavení důležitosti podle algoritmu *pagerank*. Dále knihovna umožňuje tvorbu vlastního zobrazení, více informací v Sekci 4.2.2.4.

Požadavků F1 a F2 lze docílit knihovnou bez jakýchkoliv úprav, pro splnění požadavku F5 a nastavení vlastního vzhledu je ale potřeba vytvořit vlastní *renderer*, k jehož tvorbě neexistuje dokumentace. Knihovna implementuje vlastní přibližování a pohyb v myšlenkové mapě, tím je splněn požadavek F7. Existující implementace ale může způsobovat problémy pro Požadavek F4, který vyžaduje přesuny zobrazení na jiné pojmy. Mohlo by být obtížné dosáhnout plynulého přesunu.

Celkově je knihovna specializovaná na grafy a zároveň poměrně flexibilní, její nevýhodou je ale chybějící dokumentace pro tvorbu vlastního *rendereru*.

4.2.2 Nástroje pro tvorbu rozložení a určení důležitosti

V této kapitole je rozebráno řešení požadavků F3 a F8, tedy vytvoření rozložení grafu a zváženo, zda nástroje umí vytvořit hierarchické rozložení a určit důležitost podle *pagerank* algoritmu. Nástroje pro určení důležitosti budou využívány aktérem Tvůrce, a pokud bude zvolena technologie běžící v klientské aplikaci, i aktérem Uživatel.

4.2.2.1 D3-force

D3 neumí vytvářet hierarchické rozložení, proto dále není zvažováno.

4.2.2.2 Graphviz

Program Graphviz je vhodný pro vytvoření rozložení jednou a pak jeho zobrazování všem klientům. Pro tvorbu rozložení by tedy bylo možné dedikovat více času a výpočetní síly, protože by se nemuselo vytvářet pokaždé při spuštění klientské aplikace. Nástroj Graphviz splňuje, F3 ale nesplňuje požadavek F8. Protože by nástroj Graphviz běžel na systému zpracovávajícím soubory WooWoo, musí splňovat i požadavek N3. Ten je splněn tím, že nástroj funguje na systému Linux.

Nástroj nabízí velké množství konfigurovatelných rozložení, ale pouze rozložení *dot* lze využít k tvorbě hierarchického rozložení.

Pro implementaci by bylo užitečné, kdyby šlo využít Graphviz už v šabloně, tedy v jazyce Ruby. Byla nalezena pouze knihovna Ruby-graphviz, zmíněná v Kapitole 2.3.2.2, která není udržovaná a nepodporuje export do formátů JSON nebo XML.

Byl vytvořen návrh na úpravu knihovny (*pull request*), který by dovolil export do formátu JSON. V době psaní bakalářské práce ale nebyl přijat.

Další možnost je napsání vlastního překladače do formátu DOT, využít ho k přeložení dat a následně Graphviz spustit systémovým příkazem.

Vzhledem k tomu, že žádná výše zmíněná možnost o využití nástroje Graphviz v programovacím jazyce Ruby není triviální, jeví se jako nejlepší možnost využití nástroje Graphviz s pomocí knihovny NetworkX v jazyce Python.

4.2.2.3 NetworkX

NetworkX implementuje algoritmus *pagerank* a umí vytvořit hierarchické rozložení integrací s nástrojem Graphviz. Splňuje tedy požadavky F8 a přes nástroj Graphviz i F3. Dále splňuje i Požadavek N3, který je nutný, protože by NetworkX běžel jako součást šablony.

Nevýhodou knihovny je, že je psána v jazyce Python, na rozdíl od šablony, která je v jazyce Ruby.

4.2.2.4 Cytoscape

Knihovna Cytoscape funguje na straně klienta, takže na rozdíl od knihovny NetworkX nevadí, že není v jazyce Ruby, protože se očekává, že aplikace klienta bude oddělená od šablony.

Umí vytvářet hierarchické rozložení i určit důležitost pomocí *pagerank*, splňuje tedy požadavky F3 i F8.

4.3 Výběr nástrojů

4.3.1 Nástroje pro vizualizaci

Možné způsoby vizualizace jsou:

- Napojení se na API aplikace Orgpad, díky čemuž by nebylo třeba implementovat jakoukoliv grafickou část. Tato možnost ale není vhodná, protože by se API mohlo kdykoliv v budoucnu.
- Využití knihovny Dracula.js, která je ale nevhodná pro jakékoliv větší změny vzhledu kvůli neflexibilnímu způsobu vykreslování.
- Vykreslování pomocí knihovny Cytoscape. Knihovna má možnost využití vlastního *rendereru*. Problém je pouze v nedostatku dokumentace, případně ve změně určitého chování.
- Použití D3, které je dobře dokumentované a velmi flexibilní. Kvůli flexibilitě a nespecializaci na grafy by vývoj mohl trvat déle.

Nástroj Orgpad a knihovna Dracula.js nejsou vhodné nástroje pro tvorbu myšlenkové mapy. Na druhou stranu Cytoscape i D3 splňují všechny požadavky a nemají mnoho nevýhod. Jde tedy především o volbu mezi hotovou částí implementace se špatnou dokumentací anebo možností vytvořit cokoliv s pravděpodobně delší dobou vývoje.

Nakonec byla zvolena knihovna D3 pro její flexibilitu a dokumentaci.

4.3.2 Nástroje pro tvorbu rozložení a určení důležitosti

Pro tvorbu rozložení lze využít buď knihovnu Cytoscape pro vytváření rozložení na straně klienta nebo Graphviz pro tvorbu rozložení společně při extrakci pojmů z WooWoo souborů. Knihovna NetworkX není zvažována, protože k tvorbě rozložení také využívá Graphviz.

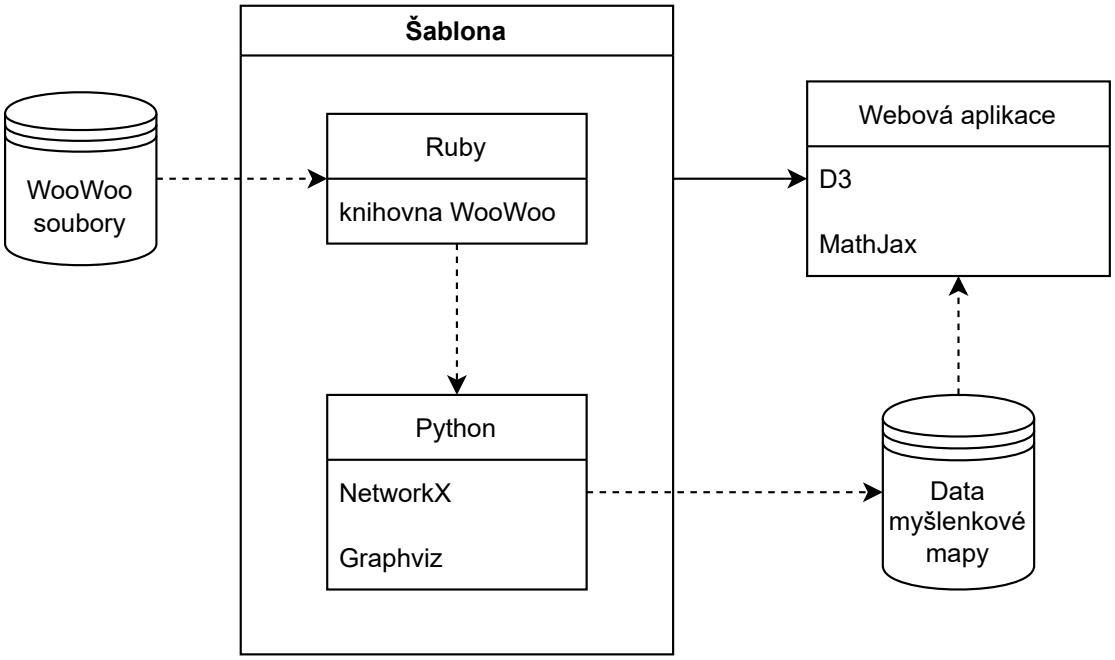
Hlavní rozdíl mezi knihovnami je tedy to, že Cytoscape je určena pro tvorbu rozložení ve webovém prohlížeči, tedy při otevření webové stránky, zatímco Graphviz by byl využit pouze jednou při extrakci dat z WooWoo souborů.

Protože není potřeba měnit rozložení za běhu klientské aplikace, byla zvolena knihovna Graphviz. Díky tomu nebude třeba vytvářet nové rozložení při každém otevření klientské aplikace a bude na jeho tvorbě možné strávit více času, protože tím nebude prodlužována doba načítání klientské aplikace.

K určení důležitosti pojmů bude využita knihovna NetworkX. Mezi implementacemi algoritmu *pagerank* nejsou mezi knihovnami velké rozdíly. Knihovna NetworkX má ale oproti ostatním knihovnám výhodu v tom, že integruje nástroj Graphviz, který je používaný pro tvorbu rozložení, tím pádem řeší dva problémy najednou.

4.4 Návrh

Na Obrázku 4.4 jsou zobrazeny komponenty aplikace. Přerušovanou čárou je vyznačen tok dat mezi komponenty, souvislou čarou jsou označeny vygenerované soubory. Tvary válce představují data a obdélníky představují jednotlivé části aplikace. Níže je popsána nejdříve sekce Šablona a po ní návrh webové aplikace vytvořené šablonou.



Obrázek 4.4 Návrh aplikace a využitých technologií.

4.4.1 Návrh šablony

Šablona bude sloužit pro tvorbu myšlenkové mapy, bude ji využívat aktér Tvůrce. Uživatelů tedy bude velmi malé množství, pravděpodobně v řádech jednotek až desítek.

Bude fungovat stejným způsobem jako typická šablona popsána v Sekci 1.5.1, tedy bude spouštěna pomocí nástroje Rake, konkrétně příkazem `fitknowledge:graph`, kde *fit* znamená, že šablona pracuje se specifikací *FIT-šablona*, *knowledge* je název šablony a *graph* je název úkolu, tedy vytvoření myšlenkové mapy.

4.4.1.1 Zpracování WooWoo souborů

Nástrojem Rake se spustí Ruby část šablony. Ta s pomocí WooWoo knihovny načte data ze souboru Woofile. Důležitá bude především cesta k iniciálnímu WooWoo souboru a typy pojmů, které mají být v myšlenkové mapě zobrazeny.

Woofile bude tedy muset obsahovat následující parametry:

root_file: cesta k WooWoo souboru ke zpracování

mindmap.parse_types: pole typů pojmů ke zobrazení v myšlenkové mapě

S pomocí WooWoo knihovny pak bude zpracován WooWoo soubor[1] specifikovaný ve Woofile. Ze souboru budou vybrány pojmy ke zobrazení, mezi nimy budou nalezeny vzájemné odkazy. Na jejich obsah pak budou pomocí knihovny WooWoo aplikovány styly z *Embedded RuBy* souborů.

Odkazy mezi pojmy budou určeny dvěma způsoby. Zpětně bude možné zjistit, jakým způsobem byl odkaz získán.

- Jedním způsobem bude určení závislostí podle odkazů z *prostředích* typu *full_reference*, *reference* a podle jejich zkratek # a @.

[1] Zpracovány budou i soubory vložené výrazem `.include`.

- Druhým způsobem bude pokus o nalezení důkazu k danému pojmu a vytěžení odkazů i z něj. Pokud bude ve WooWoo souboru *objekt* typu *Proof* pod daným pojmem s rozestupem maximálně jednoho jiného *objektu* (tedy v maximální vzdálenosti 2), vytěží se odkazy způsobem zmíněným v předchozím bodě i z důkazu.

Dále musí být obsah pojmů upraven do formátu HTML, aby ho bylo možné zobrazit ve finální webové aplikaci. Toho může být jednoduše dosaženo opět pomocí knihovny WooWoo a *Embedded RuBy* souborů ve složce erb.

Pojmy, upravené do formátu HTML, a informace o odkazech mezi pojmy pak budou ve formátu JSON předány modulu v jazyce Python.

4.4.1.2 Tvorba rozložení a určení váhy myšlenkové mapy

Druhá část šablony bude v jazyce Python 3, který je v práci dále nazýván také pouze Python. V této části bude vytvořen graf, kde pojmy budou vrcholy grafu a odkazy budou hrany. Z grafu bude pomocí algoritmu *dot* nástrojem Graphviz vytvořeno rozložení a pomocí algoritmu *pagerank* implementovaného knihovnou NetworkX určena důležitost. Důležitost bude normalizována na intervalu $\langle 0, 100 \rangle$.

Data budou exportována ve formátu JSON do složky určené parametrem *builder.build_dir*, případně do složky *./build* relativně od Woofile, pokud parametr není definován.

4.4.1.3 Výstupní data

Výstupem šablony budou zdrojové soubory webové aplikace a data (dále výstupní data) pro tuto aplikaci. Výstupní data budou ve formátu JSON s následujícími hodnotami:

canvas: informace o velikosti vytvořeného rozložení myšlenkové mapy, šířka uložena v proměnné x a výška v y

timestamp: datum vytvoření.

meta: kopie dat *dokument* a *mindmap* z Woofile.

nodes: pojmy myšlenkové mapy – Jejich název bude jedinečný identifikátor získaný z *meta-bloku* z proměnné *label* případně náhodně vygenerované, pokud proměnná *label* není definována.

Pojmy budou obsahovat následující proměnné:

x, y: souřadnice pojmů – Souřadnice $(0, 0)$ bude v levém horním rohu. Jejich hodnota se bude zvyšovat směrem doprava a dolů. Osa y bude oproti standardnímu souřadnicovému systému obráceně.

referenced_by: pole jedinečných identifikátorů pojmů závislých na tomto pojmu

required_in_proof_of: pole jedinečných identifikátorů pojmů s důkazy závislými na tomto pojmu

label: jedinečný identifikátor

type: typ výrazu ve WooWoo souboru

title: nadpis pojmu ve formátu HTML

content: obsah pojmu ve formátu HTML

weight: důležitost pojmu

chapter: WooWoo kapitola obsahující tento pojem – Může být *null*, pokud výraz v žádné sekci není.

section: WooWoo sekce obsahující tento pojem – Může být *null*, pokud výraz v žádné sekci není.

```
{
  "canvas": {
    "x": 500,
    "y": 500
  },
  "meta": {
    "root_file": "bi-ma1.woo",
    "document": {
      "title": "Matematická analýza 1"
    },
    "mindmap": {
      "parse_types": [
        "Definition",
        "Theorem"
      ]
    }
  },
  "nodes": {
    "theorem01": {
      "x": 250,
      "y": 250,
      "type": "Theorem",
      "label": "theorem01",
      "title": "Theorem title!",
      "referenced_by": [],
      "required_in_proof_of": [],
      "content": "<p>Obsah</p>",
      "weight": "100",
      "chapter": "chap",
      "section": "sec"
    }
  }
}
```

Výpis kódu 4.1 Ukázka výstupních dat z šablony.

Ukázka výstupních dat je ve Výpisu 4.1.

Zdrojové soubory aplikace budou do složky pouze překopírovány z interních souborů vytvářené šablony. Soubory šablony a data budou odděleny, aby bylo možné jednoduše měnit zdroj dat pro uživatelské zobrazení i uživatelské zobrazení pro data z šablony. Díky tomu je webová aplikace i šablona znovupoužitelná a lehce nahraditelná.

4.4.2 Návrh webové aplikace

Webová aplikace bude využívat již zmíněnou knihovnu D3 pro vytvoření prvků v myšlenkové mapě a knihovnu MathJax pro zobrazení matematických výrazů. Dále bude využívat sadu ikonek *fontawesome*[2].

Pojmy budou vykresleny pomocí HTML prvků s absolutní pozicí a reference mezi pojmy jako Bézierovy SVG křivky. Důležitost pojmů bude znázorněna velikostí písma.

[2] Více informací na `https://fontawesome.com/`

V aplikaci budou dva módy zobrazení. Takzvané přiblížené zobrazení, soustředící se na jeden pojem, a oddálené zobrazení, zobrazující více pojmů. Na Obrázku 4.5a je hrubý návrh oddáleného zobrazení a na Obrázku 4.5b hrubý návrh zobrazení při přiblížení na pojem. V oddáleném zobrazení je u pojmů napsán pouze jejich nadpis, při přiblížení je pak vidět i textový obsah pojmu. Různé typy pojmů budou barevně odděleny. Reference přímo zmíněné v pojmu budou černé a reference vycházejících z důkazu pojmu budou modré. Tedy černé reference budou vycházet ze vstupních dat *referenced_by* a modré z *required_in_proof_of*. Legenda, popisující významy barev, bude zobrazitelná po kliknutí na tlačítko ***i*** v levém horním rohu.

Aplikace bude psána v jazyce Javascript, konkrétně v ECMAScript 2015 (ES6). Pro vyšší přehlednost budou třídy členěny do modulů, které jsou dle [24] podporovány ve všech používaných prohlížečích kromě prohlížečů Internet Explorer, Opera Mini a UC prohlížeč pro Android. Pokud bude zjištěno, že je podpora těchto prohlížečů důležitá, bude možné upravit šablonu a využít nástroj jako je například Babel[3], který slouží k přeložení kódu do starší verze Javascriptu. V návrhu níže jsou zmiňovány rozhraní (anglicky *interface*), které v Javascriptu není možné vytvořit. Program by bylo možné napsat v nadstavbě Javascriptu, jako je třeba Typescript[4], to je ale pro nevelkou aplikaci jako je tato zbytečné. Proto nebudou tyto rozhraní v aplikaci existovat. Třídy, které je implementují, budou pouze dodržovat pravidla stanovená těmito rozhraními.

Aplikace bude dělena na několik tříd, hlavní dvě budou *GraphGenerator* a statická třída *Movement*.

4.4.2.1 GraphGenerator

GraphGenerator bude sloužit k vytvoření HTML a SVG z JSON dat. Prvky budou vykreslovány do dvou vrstev, ve vrchní vrstvě budou prvky HTML a ve spodní vrstvě prvky SVG.

Třída GraphGenerator bude implementovat *Observer design pattern*, tedy bude mít možnost registrace takzvaných posluchačů (anglicky *listeners*). Tito posluchači budou implementovat dvě funkce předepsané rozhraním *IHoverListener*. Funkci `hoverStart(mouseEvent, label)`, která bude volána při najetí myši na pojem a funkci `hoverEnd(mouseEvent, label)`, která bude volána, když myš opustí pojem. Parametry budou u obou funkcí stejné. Parametr *mouseEvent* bude Javascript objekt[5] a *label* bude identifikátor daného pojmu.

Cílem této třídy bude při spuštění vykreslit myšlenkovou mapu, při běhu aplikace pak bude sloužit už pouze k volání posluchačů při přejetí myší přes pojmy.

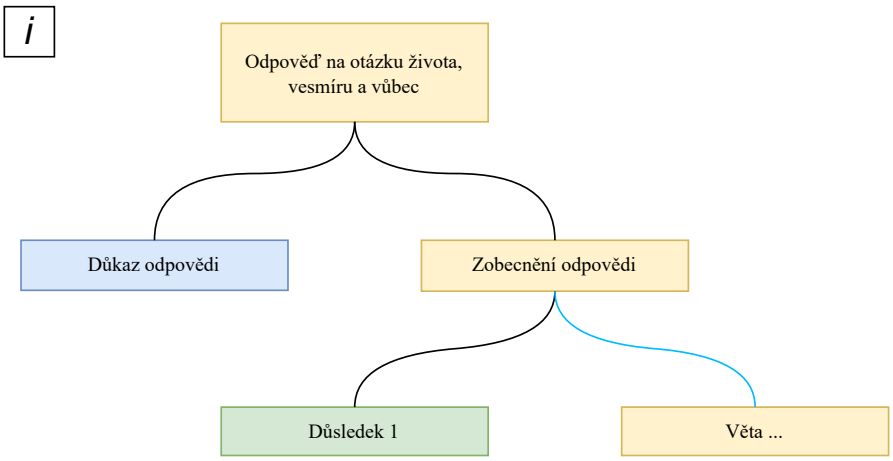
4.4.2.2 Movement

Movement bude statická třída sloužící k přibližování, oddalování a přesunu zobrazení po osách x a y. Bude obsahovat funkce pro přiblížení na určité pojmy a bude zařizovat pohyb, přibližování a oddalování pomocí myši nebo dotyku.

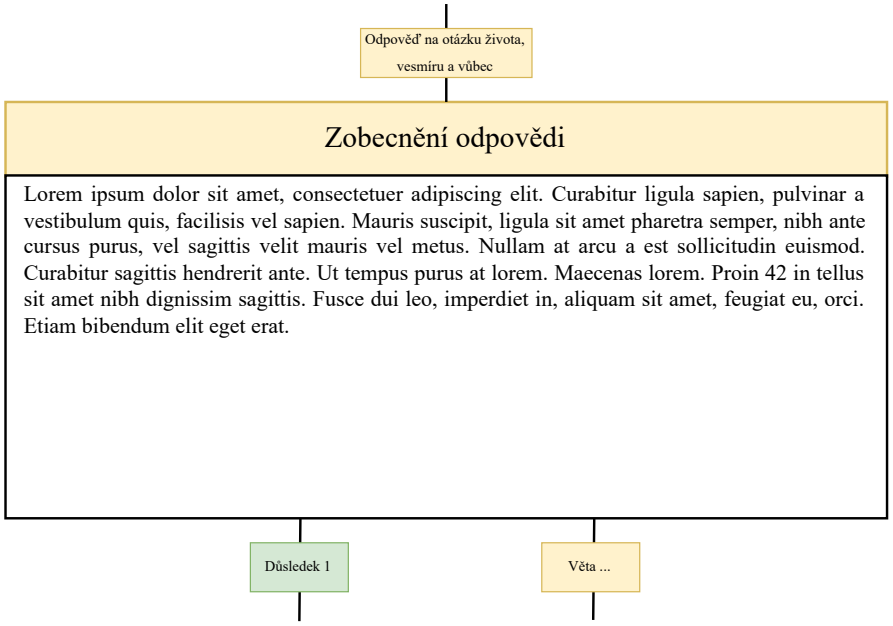
Při kliknutí na pojem v myšlenkové mapě třída *Movement* přiblíží zobrazení tak, aby byl pojem přiblížen jako na Obrázku 4.5b, při kliknutí mimo pojem pak třída zobrazení oddálí, aby vypadalo jako na Obrázku 4.5a.

Stejně jako *GraphGenerator* bude třída *Movement* implementovat *Observer design pattern*. Bude registrovat třídy implementující rozhraní *ISelectionListener*, které bude také obsahovat dvě funkce. Funkci `selectEvent(element, label)`, volanou při přiblížení na pojem, a funkci `deselect(element)`, volanou při oddálení z pojmu. Parametr *element* bude představovat Javasript odkaz na HTML element s pojmem a parametr *label* bude identifikátor objektu.

[3] Více informací o Balabel na `https://babeljs.io/docs/en/`
[4] Více o Typescriptu na `https://www.typescriptlang.org/`
[5] Dokumentace na `https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent`



(a) Hrubý návrh oddáleného zobrazení pojmů.



(b) Hrubý návrh přiblíženého zobrazení pojmu.

Obrázek 4.5 Návrh vzhledu webové aplikace.

4.4.2.3 FocusListener

Třída *FocusListener* bude implementovat rozhraní *ISelectionListener*. Jejím cílem bude zobrazit obsah pojmu, vygenerovat odkazy ve formě obdélníků nad a pod pojmem a změnit polohu křivek tak, aby odpovídaly změněné velikosti pojmu s obsahem. Zobrazený pojem by měl po zavolání funkce `selectEvent(element, label)` vypadat jako Obrázek 4.5b.

Po zavolání funkce `deselect(element)` vrátí tato třída pojem zpět do původního stavu.

4.4.2.4 UrlManager

UrlManager bude mít na starost historii prohlížeče. Bude implementovat rozhraní *ISelectionListener*. Při vybrání pojmu se do url přidá znak # a za něj *i/*, značící přiblížení k pojmu, následované identifikátorem pojmu. Při zrušení vybrání se pak do url přidá *o/* následované identifikátorem pojmu. Při změně url, například vrácením se v historii, pak třída *UrlManager* pomocí třídy *Movement* přesune zobrazení na pojem určený identifikátorem a přiblíží na něj, pokud url obsahuje *i/*, nebo pojem vycentruje, ale nepřiblíží, pokud je v url *o/*. Příkladná url při přiblížení na pojem s identifikátorem „definice-1“ by byla `www.mindmap.cz/na/teto/url#i/definice-1`

4.4.2.5 ReferenceBoldnessListener

Tato třída bude sloužit k zvýrazňování referencí končících nebo začínajících v pojmu při najetí myší na daný pojem. Reference budou zvýrazňovány pouze v oddáleném režimu.

Třída tedy bude implementovat rozhraní *IHoverListener*, aby věděla, kdy jaké reference zvýraznit, i rozhraní *ISelectionListener*, aby věděla, jestli reference zvýrazňovat, nebo jestli je zrovna přiblížen nějaký pojem, a proto reference není třeba zvýrazňovat.

4.4.2.6 ElementToggler

Tato třída bude zobrazovat a skrývat legendu při kliknutí na tlačítko informací v levém horním rohu.

4.5 Testování

V této sekci jsou vybrány knihovny pro testování a je navržen způsob testů.

4.5.1 Porovnání a výběr nástrojů

Protože byly v předchozích kapitolách vybrány nástroje v jazycích Ruby, Python a Javascript, budou k testování potřebné 3 různé nástroje, pro každý jazyk jeden.

K testování v Ruby lze využít vestavěnou knihovnu ruby/unit[6], která obsahuje základní funkcionalitu pro testování. Její výhodou je to, že ji není třeba instalovat navíc. Další možnost je aplikační rámec (anglicky *framework*) rspec[7], který je mimo normální testování použitelný i pro *test driven development*. *Framework* je využíván pro testování WooWoo knihovny nebo fit-pdf šablony. Testy lze spustit jednoduše příkazem *spec*. Pro tento projekt byl zvolen framework rspec, především pro zachování konzistence s ostatními WooWoo projekty.

Pro testování Python části byli zvažovány *frameworky* pytest[8] a unittest[9]. V *frameworcích* není příliš velký rozdíl, oba obsahují vestavěné funkce pro porovnávání výsledků a dovolují spouštění kódu před a po proběhnutí testů.

[6] Více informací o ruby/unit na `https://en.wikibooks.org/wiki/Ruby_Programming/Unit_testing`
[7] Framework rspec je popsán v Kapitole 1.3, více infomací na `https://rspec.info/`
[8] Více informací o pytest na `https://docs.pytest.org/en/7.1.x/`
[9] Více informací o unittest na `https://docs.python.org/3/library/unittest.html`

Framework unittest je vestavěn do Pythonu, takže ho není třeba instalovat. *Framework* pytest zase dovoluje spustit všechny testy jedním příkazem. Pro tento projekt byl vybrán *framework* pytest pro jeho jednoduché spouštění, které funguje stejně jako spouštění *frameworku* rspec.

Pro testování Javascriptu byly zvažovány dva typy možností a jejich případná kombinace. První možností je využití nástroje jako je Cypress[10], kde je webová aplikace spuštěna na lokálním serveru a jsou na ni prováděny testy stejným způsobem, jakým by je prováděl sám uživatel. Tento způsob testování testuje celou aplikaci jako celek a nazývá se integrační testování. Druhá možnost je využití *frameworku* Jasmine[11] nebo Jest[12] pro testování menších částí.

Po vytvoření prototypu obou způsobů bylo usouzeno, že bude postačující využít nástroj Cypress pro integrační testování a unit testování neprovádět. Důvodem je to, že většina funkcí webové aplikace něco přímo vytváří na webové stránce, tedy nějak upravuje HTML nebo CSS webové aplikace, a není při tom moc provázaná s ostatními funkcemi. Jejich testování by probíhalo tak, že by se vytvořil virtuální *DOM*, například pomocí knihovny jsdom[13], následně by se spustila testovaná funkce. Po jejím dokončení by se zkontrolovalo, jestli se v *DOM* vytvořilo to, co mělo. Integrační testování v tomto případě dělá to stejné, jeho výhodou je ale to, že je nezávislé na Javascript kódu, tudíž je menší šance, že se testy rozbijí změnami v kódu. Nevýhodou nástroje Cypress je to, že je od určité velikosti projektu zpoplatněn. Pro tento projekt je ale dostatečná i verze zdarma.

4.5.2 Návrh testů

Testy by měly být jednoduše spustitelné, proto bude existovat jeden Ruby soubor, který postupně spustí všechny tři testovací nástroje. Po skončení soubor vrátí status 0, pokud proběhnou všechny testy úspěšně, nebo status 1, pokud některý z testů selže. Je důležité, aby testování pokračovalo dále i po selhání jednoho testu, aby byly odhaleny veškeré selhávající testy při jednom spuštění.

Spuštění *frameworků* rspec pro otestování Ruby a pytest pro otestování Pythonu bude provedené zavoláním systémového příkazu pro jejich spuštění. Pro spuštění testů Javascriptu nástrojem Cypress bude potřeba spustit lokální server. Ten bude spuštěn v adresáři, do kterého budou přesunuty soubory webové aplikace a testovací data simulující výstup šablony. Po skončení testů budou tyto soubory z adresáře vymazány.

Veškeré testy budou spuštěny automaticky při jakékoliv změně v git repozitáři. Jelikož ČVUT FIT využívá nástroj Gitlab, bude automatické spouštění uskutečněno pomocí Gitlab *continuous integration*.

[10] Více o Cypress na `https://docs.cypress.io/`
[11] Více informací o Jasmine na `https://jasmine.github.io/`
[12] Více informací o Jest na `https://jestjs.io/`
[13] Více informací o jsdom na `https://github.com/jsdom/jsdom`

Kapitola 5
Implementace

V této kapitole je popsán průběh vývoje šablony a webové aplikace. Jsou zmíněny komplikace, které při vývoji nastaly a způsob, jakým byly vyřešeny.

Tvorba šablony probíhala agilním vývojem ve dvoutýdenních iteracích. Na konci každé iterace následovala schůze s vedoucím práce Ing. Tomášem Kalvodou Ph.D., se kterým byly probrány výsledky iterace a vymyšleny cíle pro další iteraci. Komunikace probíhala i mimo stanovené schůze pomocí platformy Teams.

5.1 Implementace šablony

5.1.1 Zpracování WooWoo souborů

Zpracování souborů WooWoo bylo dle návrhu implementováno v jazyce Ruby. Jak je zmíněno v Sekci 1.6.2, šablona pro nalezení pojmů a referencí již existuje. Nachází se v repozitáři[1] na Gitlabu školy ČVUT v Praze na Fakultě informačních technologií a jmenuje se fit-knowledge. Účel šablony byl tak podobný účelu této šablony, že namísto využití části jejího kódu k vytvoření nové šablony byla využita celá šablona včetně jejího repozitáře, který nebyl v době začátku práce ničím využívaný. Tato šablona tedy pokračuje ve vývoji šablony fit-knowledge, a nese proto i stejné jméno.

Pro získání referencí byl lehce upraven *rake task build*. Byla přidána možnost výběru typů pojmů, které mají být zpracovány. Bylo zrušeno získávání referencí z poznámek pod čarou, protože bylo při konzultacích usouzeno, že neobsahují relevantní informace. Bylo přidáno vyhledávání referencí z důkazů pojmu, a nakonec bylo přidáno formátování obsahu pojmů do formátu HTML.

K formátování pojmů byla využita knihovna WooWoo a její možnost aplikování stylu pomocí *Embedded RuBy*. Šablona fit-html zmíněná v Sekci 1.6.1 také využívá *Embedded RuBy* pro formátování obsahu WooWoo souborů do formátu HTML. Při vývoji byly tedy převzaty všechny *.erb* soubory ze složky erb v šabloně fit-html, některé byly následně upraveny a nepotřebné smazány. Tím se ušetřilo mnoho času při vývoji a zároveň se docílilo konzistentního formátování textu myšlenkové mapy s webovými stránkami z šablony fit-html.

[1] Veřejně nedostupný, vlastněný Ing. Tomášem Kalvodou, Ph.D. na `https://gitlab.fit.cvut.cz/woowoo/fit-knowledge`

```
exec_output = `python #{executable_path} #{output_file} #{destination_file}`
```

Výpis kódu 5.1 Nebezpečný způsob spuštění Python skriptu.

```
stdout, stderr, status = Open3.capture3(
    "python",
    executable_path,
    output_file,
    destination_file
)
```

Výpis kódu 5.2 Využívaný způsob spuštění Python skriptu.

Pro tvorbu rozložení byl vytvořen nový *rake task* jménem *graph*. Ten nejprve spustí dříve zmíněný *rake task build*, který vytvoří JSON soubor s pojmy a s jejich referencemi. Následně překopíruje soubory webové aplikace do cílové složky a poté spustí Python část šablony, která k webové aplikaci doplní data.

Přecházení mezi jazyky přidává další závislosti šabloně, ideální by bylo vše udělat v jazyce Ruby, ten ale nemá dostupné knihovny k docílení požadavků.

Python je z Ruby spouštěn pomocí systémového příkazu. Původně bylo ke spuštění využito příkazu ve Výpisu 5.1, kód měl ale dva nedostatky.

V případě, že by uživatel neměl Python 3 spustitelný příkazem *python*, by šablona nedokázala Python 3 spustit a tím by nemohla vytvořit rozložení a určit důležitost pojmů. Další problém byl, že parametr *destination_file* je uživatelsky definovaný v souboru Woofile. Toto byl bezpečnostní problém, protože tím bylo možné někomu spouštět neočekávané příkazy v příkazové řádce. Spuštění neočekávaných skriptů bylo možné tím, že byl jako uživatelsky definovaný parametr cesty ve Woofile nastaven příkaz, který středníkem ukončil systémový příkaz a spustil svůj vlastní. Příkladem takového parametru je *";bash virus.sh*.

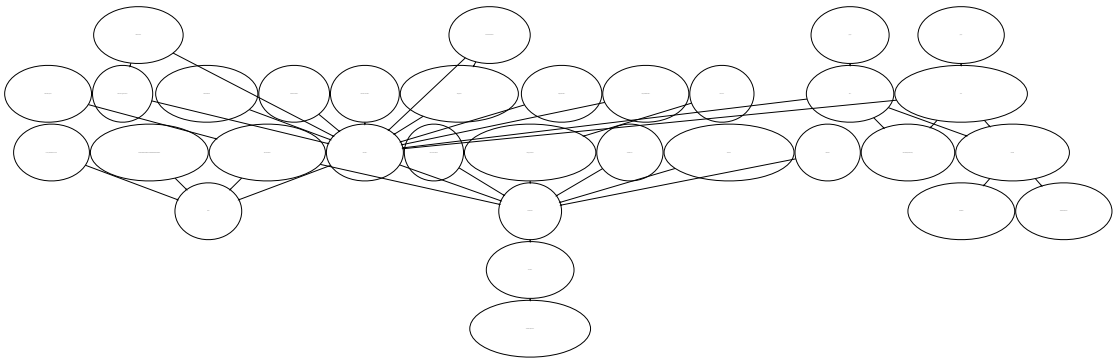
Bezpečnostní problém byl vyřešen použitím knihovny Open3[2], která je vestavěná do Ruby, není tedy potřeba instalovat další závislosti. Open3 lze využít k bezpečnému předání potenciálně škodlivých parametrů. Problém s Pythonem, který není spustitelný příkazem *python*, v šabloně není řešen. Pokud uživatel potřebuje spouštět například Python 2 příkazem *python*, bude mít ke spuštění možnost využít technologii Docker[3], pro kterou byl do git repozitáře šablony přidán soubor Dockerfile.

5.1.2 Tvorba rozložení a určení váhy myšlenkové mapy

Tvorba rozložení a myšlenkové mapy byla implementována pomocí Python skriptu *graph.py*. Ten přijímá dva argumenty, vstupní a výstupní soubor. Vstupní soubor je cesta k JSON souboru vytvořeném Ruby částí šablony pomocí *rake task build* a výstupní soubor je cílová destinace určena Ruby částí ze souboru Woofile. Skript *graph.py* tedy ze vstupního souboru nejprve načte data, z nich pak vytvoří objekt *DiGraph* z knihovny NetworkX, který představuje orientovaný graf.

Vytvořený graf pak pomocí knihovny NetworkX přepíše do objektu obalujícího formát *DOT*, který integruje knihovnu NetworkX s nástrojem Graphviz. Na tento objekt pak pomocí Graphviz aplikuje rozložení *dot*.

[2] Více o Open3 na `https://github.com/ruby/open3`
[3] Více o technologii Docker na `https://www.docker.com/`



Obrázek 5.1 Výstupní rozložení nástroje Graphviz.

Při vývoji bylo zjištěno, že rozložení vytvořené bez jakékoliv další konfigurace není pro myšlenkovou mapu vhodné. Vrcholy jsou příliš blízko u sebe, takže se v přiblíženém zobrazení navzájem překrývají. Některé reference vedly místo nejkratší cesty cestou kolem grafu a překrývaly se tak, že vypadaly jako jednotná čára.

Proto bylo potřeba přidat několik parametrů pro nástroj Graphviz. Vrcholům byla nastavena fixovaná výška a šířka pomocí nastavení atributů *fixedsize* na hodnotu True *height* na 15 a *width* 25. Kvůli stejné šířce pak byly vrcholy vykresleny do sloupců, kvůli čemuž bylo u některých referencí obtížné poznat, zda vedou do pojmu umístěného o jeden či o více pojmů níže. Aby se vyřešil tento problém, je k šířce přičteno číslo v rozsahu $\langle -10, 10 \rangle$ vytvořené pomocí hashovací funkce md5 z identifikátoru pojmu. Hashovací funkce místo náhodného čísla je použita proto, aby měly pojmy po každém spuštění stejné umístění a rozložení tak bylo plně deterministické.

Dále je pro lépe vypadající zobrazení nastavena váha referencí získaných z důkazů na hodnotu 5 a váha ostatních referencí na hodnotu 1. Čím vyšší je hodnota váhy, tím více se Graphviz snaží udělat referenci svislejší. Pro celé zobrazení je pak nastaven parametr *splines* na False, čímž jsou vynuceny rovné reference. Tím se předejde referencím, které vedou nepřehlednou klikatou cestou. Dále je nastaven parametr *mclimit* na hodnotu 100. Tento parametr určuje, po kolika nezlepšujících se pokusech o vytvoření zobrazení bez křížících se hran bude tvorba rozložení ukončena.

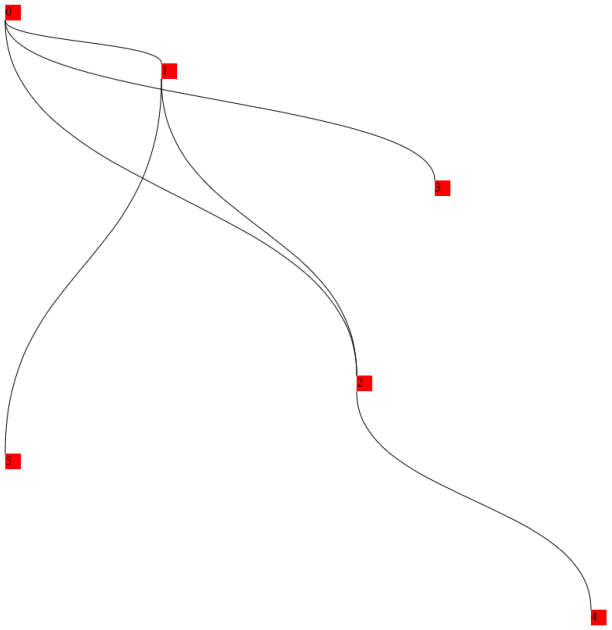
Po této konfiguraci vznikne rozložení zobrazené na Obrázku 5.1. Výstupní rozložení je hlavou dolů. To je tím, že Graphviz bere jako bod souřadnice $(0, 0)$ levý dolní roh a webové zobrazení dle návrhu jako levý horní roh. Graphviz dovoluje měnit směr zobrazení, na rozložení to ale nemá vliv a proto není třeba graf otáčet.

Pro určení váhy byla využita knihovna NetworkX, hodnoty váhy jsou odmocněny třetí odmocninou a normalizovány na interval $\langle 0, 100 \rangle$.

Výsledný graf je po určení váhy a rozložení exportován spolu s daty o velikosti celkového grafu a metadaty *mindmap* a *document* z Woofile ve formátu JSON do výstupní složky určené parametrem Python skriptu. Mimo hodnoty určené v návrhu v Kapitole 4.4.1.3 obsahuje výstupní soubor navíc dodatečné informace o pojmech. Tato data mohou být využita v budoucnu jinými aplikacemi, zvýšená velikost souboru jejím přidáním je zanedbatelná. Na druhou stranu z Woofile jsou překopírovány pouze data *mindmap* a *document*. Ostatní data by mohly obsahovat informace jako jsou ssh klíče k repozitářům, které by neměly být vystavěny veřejnosti.

5.2 Webová aplikace

Díky oddělení dat a vzhledu mohla být nejdříve vyvíjena webová aplikace a až poté šablona pro vytvoření dat. Před vytvořením šablony byly využívány ručně vytvořená data simulující reálný výstup.



Obrázek 5.2 Prototyp aplikace po základním zobrazení pojmů a referencí.

Nejdříve bylo implementováno zobrazení pojmů a jejich referencí podle dat, dále bylo přidáno posouvání zobrazení, přibližování a oddalování myší a dotykem, to bylo provedeno pomocí úprav css stylu `transform: translate(10px, 55px) scale(0.5);`, který byl aplikován na prvek zapouzdřující HTML prvky s pojmy a s referencemi. Po změnách byla vytvořena aplikace na Obrázku 5.2.

Následně byl upraven vzhled pojmů tak, aby vypadal více jako v návrhu na Obrázku 4.5a. Bylo přidáno tělo pojmů a z šablony fit-html byla převzata knihovna MathJax a její konfigurace. Kvůli delší době kompilace matematických výrazů knihovnou MathJax byla přidána načítací obrazovka, která zmizí po načtení pojmů.

Ve třídě *Movement* byly pomocí knihovny D3 implementovány funkce k přesunutí zobrazení na určitou polohu a k přesunutí zobrazení na pojem. Tyto funkce byly využity pro přiblížení při kliknutí na pojem a při oddálení zpět při kliknutí mimo pojem.

Důležitost pojmů byla v oddáleném zobrazení zvýrazněna velikostí písma a v přiblíženém zobrazení vykřičníky v pravém horním rohu. Čím více vykřičníků, tím je pojem důležitější. Vedle vykřičníků byla přidána ikonka odkazující na zobrazený pojem v studijním textu z šablony fit-html. Díky tomu, že obě šablony využívají stejný jedinečný identifikátor, získaný z parametru *label* z *meta-bloku* výrazu, případně z hashe výrazu z knihovny WooWoo, lze doplnit za url kapitolu a sekci získanou z JSON dat, následně znak „#“ a identifikátor, čímž webový prohlížeč zobrazí správnou kapitolu a přesune zobrazení na HTML prvek s daným identifikátorem. Pro možnost odkázání je třeba znát, na jaké stránce se studijní texty z fit-html nacházejí. Proto byl ve Woofile zaveden nový parametr `mindmap.textbook_url`, do kterého lze vyplnit url adresa skript. Pokud parametr není vyplněn, ikonka se nezobrazí.

Většina dalšího chování a vzhledu byla přidána pomocí tříd, které implementují rozhraní *IHoverListener* a *ISelectionListener*. Byla přidána třída *ReferenceBoldnessListener* pro zvýraznění referencí pojmu, na kterém se nachází kurzor myši. Třída *FocusListener* pro zobrazení názvů referencí nad a pod pojmem při jeho přiblížení. Nakonec byla přidána třída *UrlManager* spravující historii v prohlížeči.

Různé typy pojmů jsou barevně rozlišeny, jejich soubor se styly je oddělený od ostatních stylů, takže je přehledný. Tím by úprava barev nebo přidání nového pojmu v budoucnu mělo být

jednoduché. Momentálně jsou v aplikaci barevně rozlišené pojmy (neboli výrazy ve WooWoo) typu *Definition, Theorem, Lemma* a *Proposition*. Dále je definován výchozí styl pro všechny ostatní pojmy.

Pro přehledné zobrazení na různě velkých obrazovkách byl vytvořen nový soubor jménem responsive.css. V něm byly upraveny styly pro tři různé velikosti obrazovek, pro obrazovky užší než 600 pixelů, pro obrazovky o velikosti mezi 600 a 1000 pixely a nakonec pro obrazovky širší než 2000 pixelů. Pro obrazovky s šířkou v rozmezí 1001 a 1999 pixelů jsou styly v souboru responsive.css nezměněné. Díky využití responsivních jednotek *em* stačilo u obrazovek s šířkou nad 600 pixelu změnit základní velikost písma pro rodičovský prvek všech ostatních prvků, a velikost se propagovala i na všechny ostatní styly. U zobrazení s šířkou pod 600 pixelů bylo třeba změnit i další parametry, jako například velikost pojmů v legendě.

U pojmů s velkým množstvím referencí bylo v jejich přiblíženém zobrazení problematické zobrazení obdélníků s názvy referencí vykreslených nad a pod pojmem. V případě, že se všechny reference nevešly na obrazovku, bylo přidáno chování, které umožňovalo procházet referencemi pomocí kolečka myši nebo přejetím prstu po dotykové obrazovce.

Výsledná aplikace s daty z prvních dvou kapitol předmětu BI-MA1 je pak zobrazená na Obrázku 5.3.

Aplikace byla navržena a implementována pro jednoduchou rozšiřitelnost. Další funkce je jednoduché přidat registrací dalších posluchačů. Nové typy pojmů je možné zavést úpravou stylů, které jsou od ostatních stylů přehledně odděleny v jiném souboru.

5.3 Implementace testů

Implementace testů proběhla dle návrhu v Kapitole 4.5.2. Veškeré soubory, které se týkaly testů, byly vloženy do složky *tests*. Ta byla umístěna do kořenového adresáře. Jako podadresáře byly vytvořeny čtyři složky, složka pro každý z jazyků a složka resources, do které byly umístěny soubory využívané k testování.

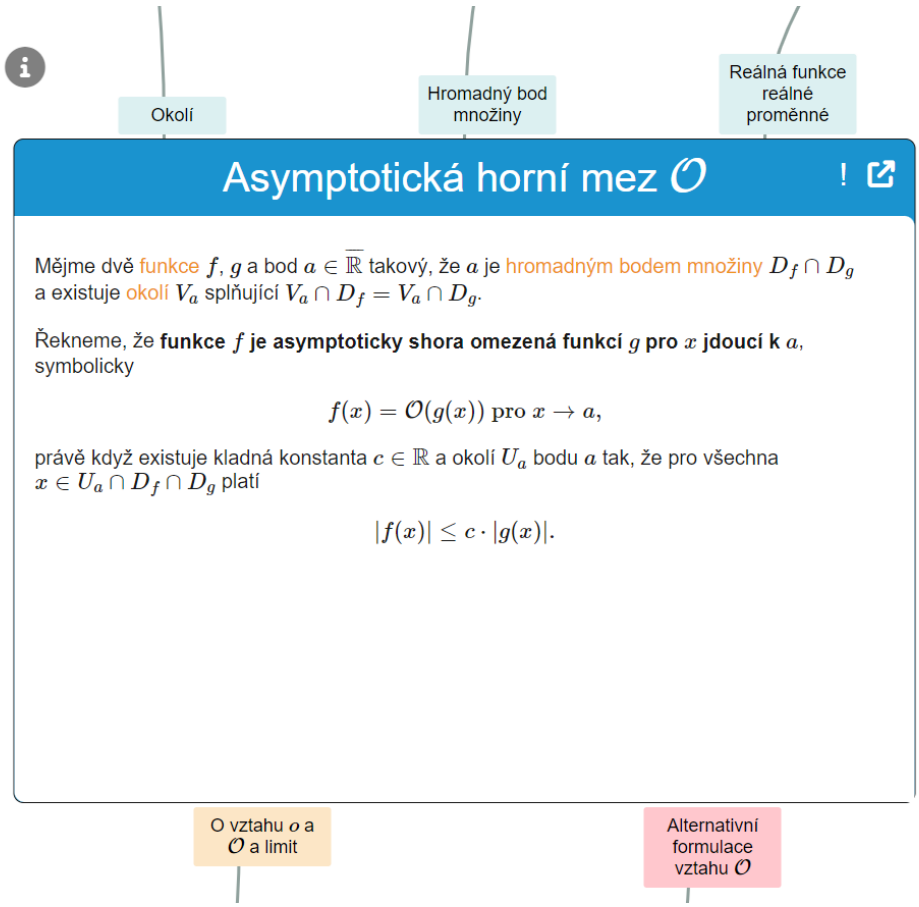
Spouštění testů Python a Ruby lze provést systémovým příkazem *pytest* a *rspec*. Pro spuštění testů webové aplikace byl vytvořen Ruby soubor jménem run_cypress.rb, který spustí lokální server, následně provede testy nástrojem cypress a nakonec lokální server vypne. Pro spuštění všech tří testů najednou byl vytvořen Ruby soubor run_tests.rb

V Python části šablony byly všechny funkce pokryty testy. V Ruby části byla pomocí *mocků*[4] mimikována funkcionalita WooWoo knihovny. Všechny významné funkce byly pokryty unit testy. Pro testování webové aplikace byl vytvořen testovací JSON soubor, na kterém je aplikace testována. Lokální server byl implementován pomocí vestavěné Ruby knihovny jménem Webrick[5]. Nástrojem cypress bylo otestováno přibližování a oddalování na prvky při kliknutí, přesouvání přes reference z prvku na prvek, historie url a zobrazení legendy.

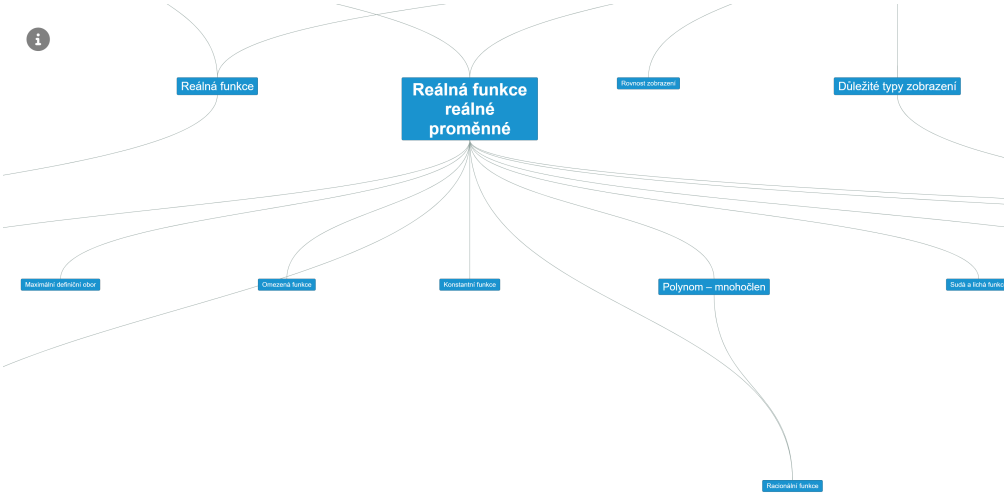
Testy se automaticky spouští při každém nahrání změn v repozitáři Gitlab. Pokud jsou změny provedeny na git *branch* jménem master, jsou ještě před spuštěním testů vytvořené dva Docker *images*[6] , *image* jménem *knowledge* a *image* jménem *knowledge-dev*. Oba jsou následně publikovány do Gitlab *Registry*. Docker *image knowledge* je určený ke standardnímu využití šablony. Obsahuje tedy nainstalovanou knihovnu WooWoo spolu s dalšími závislostmi, které jsou potřebné ke spuštění. *Image knowledge-dev* rozšiřuje *image knowledge* tím, že přidává závislosti potřebné ke spouštění testů. Oba Docker *images* jsou vytvořeny ze stejného souboru Dockerfile pomocí *multi-stage build*[7].

Pokud změny nejsou provedeny na git *branch* master, nový Docker *image* nebude vytvořen. Všechny tři části testů jsou spuštěny paralelně. Pokud žádný z testů neselže, je testování označené jako úspěšné.

[4] Více o *mock* na `https://devopedia.org/mock-testing`
[5] Více informací o knihovně Webrick na `https://github.com/ruby/webrick`
[6] Více o Docker *images* na `https://docs.docker.com/get-started/overview/`
[7] Více o Docker *multi-stage build* na `https://docs.docker.com/develop/develop-images/multistage-build/`



(a) Vzhled přiblíženého zobrazení v dokončené aplikaci.



(b) Vzhled oddáleného zobrazení v dokončené aplikaci.

Obrázek 5.3 Výsledná webová aplikace.

Kapitola 6
Závěr

Cílem práce bylo prozkoumat webové technologie pro vizualizaci myšlenkové mapy, seznámit se s WooWoo formátem a následné navržení a implementace WooWoo šablony, která z WooWoo souborů vygeneruje myšlenkovou mapu.

V rešerši byly analyzovány WooWoo soubory, WooWoo šablony i knihovna WooWoo. Bylo rozebráno několik možností pro zobrazení, rozložení i určení důležitosti pojmů myšlenkové mapy. Díky tomu pak byly v návrhu aplikace učiněny informované rozhodnutí o volbě nástrojů.

Při implementaci byly znovu použity již funkční a používané prvky z existujících WooWoo šablon, složité algoritmy byly přenechány na knihovnu NetworkX a nástroj Graphviz, čímž se výrazně zkrátila doba vývoje a s tím i riziko výskytu chyb. Celá šablona je testována automatickými unit testy, které v budoucím vývoji předejdou nechtěným změnám.

Uživatelské rozhraní myšlenkové mapy ve webové aplikaci využívá co nejmenší množství knihoven, díky tomu zůstává velikost celkové stránky malá a nenáročná na internetové připojení. Zároveň dovoluje rychlou navigaci podle referencí, plynulé přibližování na celý text pojmů a následné oddálení zpět na pouhý nadpis, zobrazení speciálních matematických pojmů a možnost navigace zpět pomocí historie prohlížeče.

Šablona se aktivně využívá společně se šablonami fit-html, fit-pdf a šablonou pro tvorbu Anki kartiček. V době vydání této práce je myšlenková mapa dostupná pro všechny studenty FIT ČVUT na `https://courses.fit.cvut.cz/BI-MA1/@master/textbook/mindmap/`. Studenti se na ni mohou dostat přes webové stránky předmětu.

Na výsledném nástroji je ještě mnoho prostoru pro zlepšení. Nejvíce by šlo zapracovat na vzhledu a na rozložení.

Cílem této práce bylo vytvořit funkční interaktivní aplikaci, hezký vzhled byla až sekundární cíl. Pro jeho zdokonalení by bylo třeba grafika, který by při vývoji poskytl vizuální předlohy. Další krok ke zlepšení by tedy mohla být spolupráce s grafikem/grafičkou k výběru hezčí palety barev a k zvýraznění typů pojmů (Definice, Věta, …) způsobem, který by nevyžadoval legendu.

Tvorba rozložení myšlenkové mapy je komplikovaný problém. Využité rozložení dobře zobrazuje hierarchii, mohlo by ale být vyšší a méně široké. Určitě by pomohla implementace vlastního nástroje na tvorbu rozložení, které by nějak kombinovalo pružinkové a hierarchické zobrazení.

Příloha A

Návod k použití

Doporučený způsob spuštění šablony WooWoo je pomocí technologie Docker. V repozitáři dostupném na `https://gitlab.fit.cvut.cz/woowoo/fit-knowledge` jsou dva Docker *images*. *Image knowledge*, který obsahuje všechny závislosti pro spuštění šablony, a *image knowledge-dev*, který navíc obsahuje závislosti pro spuštění testů. Protože se tyto Docker *images* vytvářejí průběžně při každé změně v repozitáři a knihovna WooWoo, na které jsou závislé, se rychle mění, může se stát, že tyto nejaktuálnější *images* nebudou plně funkční. Proto existují verze *knowledge:stable* a *knowledge-dev:stable*, které byly nahrány v době odevzdání této práce a plně fungují.

Pro úspěšné spuštění následujících příkazů je potřeba se přihlásit do *registry*:

```
docker login gitlab.fit.cvut.cz:5000
```

Příkazy v Docker lze spouštět následujícím příkazem[1]:

```
docker run -it -v PATH:/src \
  gitlab.fit.cvut.cz:5000/woowoo/fit-knowledge/knowledge:stable COMMAND
```

Je potřeba nahradit `PATH` cestou k WooWoo projektu a `COMMAND` spouštěným příkazem. List akcí, které šablona nabízí, lze zobrazit příkazem `rake -T`. Myšlenková mapa lze vytvořit příkazem `rake fitknowledge:graph`.

Pro vyzkoušení na WooWoo souborech předmětu BI-MA1 spusťte následující příkazy:

Stáhněte si repozitář předmětu BI-MA1:

```
git clone git@gitlab.fit.cvut.cz:BI-MA1/bi-ma1.git
cd bi-ma1
git checkout bp-frnka
```

V souborech BI-MA1 se WooWoo soubory nacházejí ve složce src/textbook. Pro použití této šablony (funkční i na case insensitive souborových systémech) proveďte následující:

- Ve složce textbook vytvořte adresář templates.
- V následujícím příkazu nahraďte PATH cestou ke složce textbook.

[1] Symbol \ představuje pokračování příkazu na další řádce.

```
docker run -it -p 8000:8000 \
  -v PATH:/src gitlab.fit.cvut.cz:5000/woowoo/fit-knowledge/knowledge:stable \
  /bin/bash

ln -s /opt/fit-knowledge/templates/fit-knowledge/ /src/templates/
```

- Pokud používáte operační systém Windows:

```
apt-get install dos2unix

dos2unix *
```

- Pro zobrazení možných akcí použijte `rake -T`
- Pro vytvoření myšlenkové mapy použijte `rake fitknowledge:graph`. Soubory se vytvoří do podsložky build.
- Pro zobrazení myšlenkové mapy použijte `rake fitknowledge:serve` a přejděte na webovou stránku `localhost:8000`

Pro spuštění testů spusťe následující příkazy. Testy běží také automaticky v repozitáři `https://gitlab.fit.cvut.cz/woowoo/fit-knowledge`. Zažádejte Ing. Tomáše Kalvodu, Ph.D. pro přístup.

```
docker run -it \
  gitlab.fit.cvut.cz:5000/woowoo/fit-knowledge/knowledge-dev:stable \
  /bin/bash

cd /opt/fit-knowledge/tests
```

Testy lze spustit:

- Veškeré testy: `ruby run_tests.rb`
- Testy webové aplikace: `ruby run_cypress.rb`
- Testy Python části šablony: `pytest`
- Testy Ruby části šablony: `rspec`

Bibliografie

1. STRAKA, David. *Editor zdrojových kódů WooWoo dokumentů*. Thákurova 9, 160 00 Praha 6, 2021. Bakalářská práce. České vysoké účení technické, Fakulta informačních technologií.
2. KALVODA, Tomáš. *WooWoo Specs* [online]. 2021 [cit. 2022-03-25]. Dostupné z: `https://kam.fit.cvut.cz/deploy/woowoo-specs/woowoo-specs.pdf`.
3. HAMILTON, Thomas. *What is Test Driven Development (TDD)? Tutorial with Example: What is Test Driven Development(TDD)?* [Online] [cit. 2022-03-24]. Dostupné z: `https://www.guru99.com/test-driven-development.html`.
4. ŠODEK, Ondřej. *Sémantické sítě matematických znalostí*. Thákurova 9, 160 00 Praha 6, 2020. Bakalářská práce. České vysoké účení technické, Fakulta informačních technologií.
5. ČERNÝ, Michal; CHYTKOVÁ, Dagmar. *Myšlenkové mapy pro studenty*. BIZBOOKS, 2014. ISBN 978-8026502678.
6. HAY, David; KINCHIN, Ian; LYGO-BAKER, Simon. Making learning visible: The role of concept mapping in higher education. *Studies in Higher Education*. 2008, roč. 33. Dostupné z DOI: `10.1080/03075070802049251`.
7. HOPPER, Carolyn H. *Practicing College Learning Strategies*. Cengage Learning, 2015. ISBN 978-1305109599. Dostupné také z: `https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=1305109597`.
8. MINDMAPPING.COM. *What is a Mind Map?* [Online]. © 2022 [cit. 2022-03-28]. Dostupné z: `https://www.MindMapping.com`.
9. KOBOUROV, Stephen G. *Spring Embedders and Force Directed Graph Drawing Algorithms*. arXiv, 2012. Dostupné z DOI: `10.48550/ARXIV.1201.3011`.
10. KAUFMANN, Michael; WAGNER, Dorothea. *Drawing Graphs: Methods and Models*. 2001. ISBN 978-3-540-42062-0. Dostupné z DOI: `10.1007/3-540-44969-8`.
11. D3. *d3-force* [online]. D3, 2022 [cit. 2022-04-08]. Dostupné z: `https://github.com/d3/d3-force`.
12. BRANDES, Ulrik. On Variants of Shortest-Path Betweenness Centrality and their Generic Computation. *Social Networks*. 2008, roč. 30, s. 136–145. Dostupné z DOI: `10.1016/j.socnet.2007.11.001`.
13. BRIN, Sergey; PAGE, Lawrence. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*. 1998, roč. 30, č. 1, s. 107–117. ISSN 0169-7552. Dostupné z DOI: `https://doi.org/10.1016/S0169-7552(98)00110-X`.
14. LEVINE, Stephen. Several measures of trophic structure applicable to complex food webs. *Journal of Theoretical Biology*. 1980, roč. 83, č. 2, s. 195–207. ISSN 0022-5193. Dostupné z DOI: `https://doi.org/10.1016/0022-5193(80)90288-X`.

15. ŠESTÁKOVÁ, Eliška. *Implementujeme digitální zettelkasten (2. část): džungle aplikací* [online]. 2021-12-30 [cit. 2022-04-23]. Dostupné z: https://eliskasestakova.cz/digitalni-zettelkasten-2/.

16. ORGPAD. *OrgPad* [online] [cit. 2022-04-06]. Dostupné z: https://www.orgpad.com/.

17. OBSIDIAN. *Obsidian* [online]. © 2022 [cit. 2022-04-06]. Dostupné z: https://obsidian.md/.

18. D3. *D3 Wiki* [online]. Bostock, 2021 [cit. 2022-04-07]. Dostupné z: https://github.com/d3/d3/wiki.

19. BOSTOCK, Mike. *Mobile Patent Suits / D3 / Observable* [Observable] [online]. 2020-01-16 [cit. 2022-04-07]. Dostupné z: https://observablehq.com/@d3/mobile-patent-suits.

20. GRAPHDRACULA. *Dracula Graph Library* [Dracula Graph Library] [online]. © 2022 [cit. 2022-04-14]. Dostupné z: https://www.graphdracula.net/.

21. FRANZ, Max; LOPES, Christian T.; HUCK, Gerardo; DONG, Yue; SUMER, Onur; BADER, Gary D. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*. 2015, roč. 32, č. 2, s. 309–311. ISSN 1367-4803. Dostupné z DOI: 10.1093/bioinformatics/btv557.

22. GRAPHVIZ. *Graphviz* [online]. 2022-04-04 [cit. 2022-04-12]. Dostupné z: https://graphviz.org/about/.

23. HAGBERG, Aric A.; SCHULT, Daniel A.; SWART, Pieter J. Exploring Network Structure, Dynamics, and Function using NetworkX. In: VAROQUAUX, Gaël; VAUGHT, Travis; MILLMAN, Jarrod (ed.). *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, 2008, s. 11–15.

24. DEVERIA, Alexis. *Can I use...* [Online] [cit. 2022-04-05]. Dostupné z: https://caniuse.com/?search=es6.

Obsah přiloženého média

- `fit-knowledge` ..... adresář s výslednou šablonou
- `examples` ..... ukázky práce
  - `mindmap` ..... ukázka výsledné myšlenkové mapy na datech z předmětu BI-MA1
  - `sample woowoo project` ..... ukázka WooWoo projektu zmíněném v Kapitole
- `text` ..... LaTeX soubory bakalářské práce
- `thesis.pdf` ..... text práce ve formátu PDF
- `readme.txt` ..... stručný popis obsahu média