FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Interaktivní mapa průchodu studiem na FIT ČVUT
Student:	Bc. Klára Matoušková
Vedoucí:	doc. Ing. Robert Pergl, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat prototyp aplikace pro vizualizaci obsahu studia různých oborů z hlediska předmětů a jejich sylabů.

1. Seznamte se se strukturou studijních programů a sylabů předmětů z hlediska průchodu studiem, tématickými souvislostmi a návaznostmi (Bílá kniha).
2. Proveďte rešerši vizualizačních knihoven pro web frontend aplikace.
3. Vytvořte konceptualizaci a návrh reprezentace problémové domény.
4. Navrhněte vhodné interaktivní vizualizace problémové domény – z hlediska oborů, návazností předmětů v čase i obsahově. Vemte též v potaz různé verze předmětů (akreditace).
5. Implementujte navržené vizualizace pomocí vybrané knihovny/knihoven.
6. Otestujte výsledek, a demonstrujte na vhodných ukázkách.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 21. října 2022 v Praze.

Diplomová práce

INTERAKTIVNÍ MAPA PRŮCHODU STUDIEM NA FIT ČVUT

Bc. Klára Matoušková

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: doc. Ing. Robert Pergl, Ph.D.
2. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Ráda bych poděkovala mému vedoucímu doc. Ing. Robertu Perglovi, Ph.D za odborné vedení a konzultace k této práci. Dále děkuji své rodině za podporu během mého studia.

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 2. května 2023 .....

Abstrakt

Projekt Interaktivní mapy průchodu studia na FIT ČVUT klade za cíl přehodnocení sylabů předmětů, programů, specializací a celkové prezentace struktury studia na FIT ČVUT. Tato práce se zabývá web front-end částí aplikace, ve které jsou realizovány interaktivní vizualizace z několika pohledů sylabu.

Součástí práce je rešerše existujících způsobů vizualizace se zaměřením na síťové a hierarchické vizualizace. Na základě analýzy je navrženo uživatelské rozhraní aplikace s vizualizacemi. Výsledkem práce je prototyp aplikace implementovaný pomocí vizualizační knihovny D3.js, která je v aplikaci integrována s knihovnou React.

Klíčová slova fit, čvut, sylabus, vizualizace, graf, síťový graf, hierarchie, uživatelské rozhraní, uživatelské testování, ux, ui, web front-end, d3.js, react, javascript

Abstract

The project of Interactive Map of Study at FIT CTU aims to reassess the syllabus of subjects, programs, specializations, and the overall presentation of the study structure at FIT CTU. This thesis focuses on the web front-end part of the application, in which interactive visualizations from several syllabus perspectives are realized.

The thesis includes research into existing visualization methods, focusing on network and hierarchy visualizations. Based on the analysis, a user interface with interactive visualizations is designed for the application. The result of the thesis is a prototype of the application implemented using the D3.js visualization library, which is also integrated with the React library in the application.

Keywords fit, cvut, sylab, visualization, graph, network graph, hierarchy, user interface, usability testing, ux, ui, web front-end, d3.js, react, javascript

Seznam zkratek

2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
API	Application Programming Interface
BFS	Breadth-first search
BK	Bílá kniha ČVUT
CSS	Cascading Style Sheets
CSV	Comma-separated values
ČVUT	České vysoké učení technické v Praze
D3	Data-driven documents
DFS	Depth-first search
DOM	Document Object Model
FIT	Fakulta informačních technologií
GEFX	Graph Exchange XML Format
GLSL	OpenGL Shading Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript Syntax Extension
KOS	Komponenta studium
MŠMT	Ministerstvo školství, mládeže a tělovýchovy
PDF	Portable Document Format
RAM	Random-access memory
REST	Representational state transfer
SASS	Syntactically Awesome Stylesheet
SPA	Single-page application
SQL	Structured Query Language
SSR	Server-side rendering
SVG	Scalable Vector Graphics
UC	Use case
UI	User interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User experience
VŠCHT	Vysoká škola chemicko-technologická v Praze
WebGL	Web Graphics Library
XML	Extensible Markup Language

Úvod

Doména sylabu fakulty vysoké školy obsahuje nemalé množství entit, pojmů, termínů a konceptů. Studenti se zapisují do studijních programů, vybírají si předměty do svého zápisu v dalším semestru, nebo hledají jiné souvislosti a návaznosti daného předmětu.

Jako primární informační zdroj o struktuře sylabu pro ČVUT slouží Bílá kniha ČVUT. Tento systém běží už několikátým rokem, avšak s postupem času upadá mimo prostředí moderních webových aplikací. Současné řešení tedy není pro studenty atraktivní a nenabízí veškeré informace, které by jak studenti, ale i vyučující potřebovali. Je tedy potřeba se zamyslet nad obnovou tohoto systému nebo realizaci nové nadstavby, která by různé systémy propojovala dohromady a nabízela by tím nové pohledy do studia.

Právě zlepšení současné situace na FIT ČVUT je společnou motivací několika závěrečných prací věnující se této problematice. A to jak z hlediska struktury sylabu, detekce změn v sylabu, nebo formy zobrazení obsahu studia. V komplexní doméně se přehled informací může stát komplikovaným, nebo i nevhodným s tradičním zobrazením uživatelského rozhraní webové aplikace. S nástupem moderních webových technologií je dnes otevřená možnost interaktivních webových aplikací, a to i interaktivních vizualizací.

Vizualizace nabízejí flexibilní pohledy na data a dokážou je transformovat tak, jak tradiční metody zobrazení dat nedokážou. Jedná se především o různé typy grafů nebo diagramů. Tím na ně lze aplikovat i užitečné algoritmy, které poskytnou nový pohled do studia. Zároveň se jedná o atraktivní alternativu pro zkoumání relací a entit v sylabu. Právě návrhu a implementaci těchto vizualizací se věnuje tato práce.

Cíle práce a metodika

Cílem této práce je návrh a implementace prototypu vhodných interaktivních vizualizací obsahu studia na FIT ČVUT. Účelem je navrhnout vizualizace z několika pohledů, které budou demonstrovat danou část sylabu a její problematiku. Dále je zapotřebí vizualizace implementovat pomocí vhodných metod a technologií pro web front-end část aplikace.

Teoretická část práce (Část I) se zabývá analytickou (či rešeršní) částí práce a obsahuje celkem pět kapitol.

Kapitola Sylabus (Kapitola 1) je analyzována současná struktura a situace sylabů na FIT ČVUT. Největší důraz je kladen na Bílou knihu, která slouží jako jeden z primárních zdrojů informací této domény. Jsou zmíněny i ostatní aplikace a systémy, které sylabem zabývají.

Kapitola Vizualizace (Kapitola 2) se věnuje rešerši síťových a hierarchických vizualizací. Také je zmíněn všeobecný úvod do teorie grafů.

Kapitola Vizualizační aplikace (Kapitola 3) se zabývá analýzou současných řešení včetně analýzy uživatelského rozhraní těchto aplikací. Aplikace jsou analyzovány z hlediska UI a nabízených funkcionalit pro vizualizace.

V kapitole Vizualizační knihovny (Kapitola 4) jsou porovnány web front-end vizualizační knihovny pro implementaci prototypu. Výběr knihovny závisí na několika faktorech, ale především na technologii, funkcionalitě, možnosti přizpůsobení, dokumentaci a podpoře.

První část práce je zakončena kapitolou Shrnutí teoretické části (Kapitola 5), která obsahuje souhrn poznatků z předchozích kapitol.

Praktická část práce (Část II) se věnuje návrhu, implementaci prototypu a jeho testování.

V kapitole Návrh (Kapitola 6) jsou úvodem stanoveny požadavky a případy užití aplikace. Dále jsou navrženy interaktivní vizualizace a jejich uživatelské rozhraní pomocí wireframů a zásad Nielsenovy heuristiky.

V kapitole Implementace prototypu (Kapitola 7) je návrh z předchozí kapitoly realizován. Primární knihovnou pro implementaci vizualizací je D3.js. Jádro web front-end aplikace je implementováno pomocí knihovny React.

V kapitole Testování (Kapitola 8) je provedeno a vyhodnoceno uživatelské testování prototypu. Uživatelské testování bylo provedeno dle standardních postupů s potenciálními uživateli aplikace, moderátor a testovacími scénáři.

Poslední kapitola Závěr (Kapitola 9) reflektuje tuto práci, shrnuje její výsledek a diskutuje možnost budoucího vývoje.

Část I

Teoretická část

Kapitola 1
Sylabus

Tato kapitola se zabývá rešerší sylabu FIT ČVUT. Hlavním zdrojem informací je Bílá kniha ČVUT, která obsahuje strukturu akreditací sylabu. Jednotlivé pojmy týkající se akreditací jsou rozebrány v této kapitole. Jsou zmíněny i další systémy, které se akreditací zabývají. V poslední sekci jsou rozebrány změny a rozdíly po přechodu do nových akreditací.

Sylabus představuje obsah studia, studijního programu, předmětu, kurzu či jiného vzdělávacího prostředku [1]. Pod tímto termínem jsou zahrnuty veškeré informace, které popisují jeho formu a obsah. Například předmět obsahuje název, anotaci, osnovu přednášek a cvičení, klíčová slova, zkratku, doporučené materiály atd. Tedy jakým způsobem jsou tyto informace cílové skupině prezentovány je velmi důležité - informace by měly dodržovat danou strukturu, konzistenci a dostatečné informace. Na základě sylabu si každý student vybírá vhodné studium.

1.1 Bílá kniha

Přehled sylabu fakulty zajišťuje elektronická *Bílá kniha ČVUT* (dále jen BK) [2, 3]. Obsahuje přehled pro všechny fakulty a součásti ČVUT včetně Fakulty Informačních Technologií (*FIT*). V rámci fakulty FIT obsahuje několik entit sylabu, pro které má fakulta udělenou akreditaci. Tato část je zaměřena na všeobecnou strukturu akreditací v BK, v sekci 1.1.6 se zaměřím na rozdíly a změny týkající se nové akreditace programů a specializací fakulty.

Akreditací je považováno určité oprávnění k činnosti, v tomto případě pro vzdělávací program vysokoškolského studia. V rámci školství definuje MŠMT (*Ministerstvo školství, mládeže a tělovýchovy*) standardní délku platnosti akreditace na šest let [4]. Za celou životnost FIT proběhly celkově dvě vlny akreditací a tím i obměny struktury sylabu. V roce 2009 byla fakulta založena a tím vznikl první bakalářský program *Informatika 2009*. O rok později byl přidán navazující program pro magisterské studium.

BK obsahuje primárně entity akreditace tvořící hierarchickou strukturu [5], tedy uspořádání nadřazenosti a podřazenosti. Mezi tyto entity patří:

- studijní programy
- specializace/obory
- studijní plány

Tedy každý studijní plán je součástí specializace/oboru a každá vypsaná specializace je zařazena do studijního programu. Mezi sekundární entity (netvořící nutně hierarchickou strukturu) patří:

- skupiny předmětů
- doporučený průchod
- předměty
- katedry

1.1.1 Studijní program

Nejvíce nadřazená entita je *studijní program*. Fakulta má vypsaných několik studijních programů, do kterých se studenti hlásí při výběru studia. Programy mohou být disjunktně rozděleny podle úrovně studia:

Bakalářské Jedná se o nejnižší úroveň studia, kterou fakulta vypisuje. Standardní délkou studia jsou tři roky. V rámci zkratek se označuje písmenem `B`.

Magisterské Jedná se o navazující studium na bakalářské programy. Standardní délkou studia jsou dva roky. V rámci zkratek se označuje písmenem `N` (nová akreditace viz sekce) nebo `M` (stará akreditace).

Doktorské Jedná se o nejvyšší úroveň studia. Standardní délkou studia jsou čtyři roky. V rámci zkratek se označuje písmenem `P`.

Následně jsou rozděleny podle jazyka výuky:

Čeština Tento jazyk je určen především pro české (či česky mluvící) studenty.

Angličtina Tento jazyk je určen především pro zahraniční (či anglicky mluvící) studenty.

V každé úrovni studia má fakulta vždy jeden aktuální studijní program *Informatika* v českém a anglickém jazyce. Fakulta vypsala celkem deset studijních programů, z nichž čtyři jsou bakalářské, čtyři jsou magisterské a dva jsou doktorské. Varianty bakalářského a magisterského studia obsahují české a anglické verze, avšak doktorský program obsahuje jediný jazyk výuky - češtinu.

Samostatné názvy programů nejsou jednoznačné - nově akreditované programy nesou univerzální název *Informatika* a ty staré jsou doplněny o rok získání jejich akreditace (např. *Informatika* a *Informatika 2009*). Zkratky jsou vytvořeny podle kombinace úrovně studia a jazyka výuky (např. `BIE` - bakalářská Informatika s anglickou výukou).

1.1.2 Specializace/obor

Součástí programu jsou *specializace* nebo *obory*. Od druhé iterace akreditací studijních programů je na FIT používán výhradně termín specializace. Avšak významově se jedná o to samé (více v sekci 1.1.6.1). Specializace představuje zaměření studenta - jakým směrem se specializuje. Tento tématický směr je odkázán na katedru (viz. sekce 1.1.5), která zajišťuje danou specializaci.

Specializace nesou název podle všeobecného zastřešení jejího obsahu a případně roku přidělení akreditace (např. *Počítačová grafika 2021*). Zkratky jsou vytvořeny podle kombinace jejího programu a zkratky samostatného názvu specializace (např. `BI-PG21` - specializace Počítačová grafika v programu Informatika).

1.1.3 Studijní plán

Pro každý obor/specializaci je definován minimálně jeden *studijní plán*, což je předpis, které předměty má student úspěšně absolvovat. Ve většině případech má specializace pouze jeden studijní plán a v některých dva plány. Avšak není to pravidlem - např. specializace Teoretická informatika

Studijní plán se dělí na dva typy podle formy studia:

Prezenční Tato forma studia je určena pro prezenční studenty. Neobsahuje implicitně žádnou zkratku.

Kombinovaný Tato forma studia je určena pro dojíždějící studenty. V rámci zkratek se označuje písmenem `K`.

1.1.3.1 Skupina předmětů

Studijní plán neobsahuje konkrétní předměty, které student musí absolvovat, avšak obsahuje určitou *skupinu předmětů*, která má vlastní *roli*. Předměty jsou tedy seskupeny do několika skupin podle role, kterou v plánu hrají. Platí, že skupina předmětů může být ve více studijních plánech. V BK je definováno celkem dvanáct rolí skupin předmětů, dělí se na *povinné* (začínající písmenem `P`) a *volitelné* (začínající písmenem `V`).

- **PE** - Povinné ekonomické
- **PJ** - Povinná zkouška z angličtiny
- **PO** - Povinné předměty oboru
- **PP** - Povinné předměty programu
- **PS** - Povinné předměty specializace
- **PT** - Povinná tělesná výchova, sportovní kurzy
- **PV** - Povinně volitelné předměty
- **PZ** - Povinně předměty zaměření
- **V** - Volitelné předměty
- **VE** - Povinně volitelné ekonomicko-manažerské
- **VH** - Povinně volitelné humanitní
- **VO** - Volitelné předměty oboru/specializace

V dané skupině předmětů je určen minimální počet kreditů (viz. předměty v sekci 1.1.4) sečtenými napříč předměty. Student je povinen splnit daný minimální počet kreditů skupiny podle jeho volby předmětů (s danou rolí).

Název studijních plánů je téměř identický se specializací, do které zapadají. Tvoří se kombinací úrovně studia programu, dané specializace, formy studia a případně roku přidělení akreditace (např. *Bc. specializace Počítačová grafika, kombi., 2021*). Studijní plány nedefinují žádnou zkratku.

Jména skupin předmětů jsou podle její role a dané specializace/programu, do které patří (např. *Povinné předměty bakalářského programu Informatika, verze 2021*). Zkratka je tvořena daným programem, do kterého studijní plán patří, formou studijního plánu a samostatnou zkratkou role (např. `BIK-PP.21`).

1.1.3.2 Doporučený průchod

Doporučený průchod studijním plánem je konkrétní instancí studijního plánu. Jsou zveřejněny děkanem fakulty v elektronické BK. Zde jsou skupiny předmětů nahrazeny již konkrétními předměty - doporučenými předměty. Průchod studijním plánem cestou doporučeného průchodu je populární nejen ze jednoduchosti, ale z chronologického uspořádání předmětů podle jejich znalostí. Doporučený průchod je totiž uspořádán tak, aby sylaby předmětů na sebe navzájem navazovaly (je-li to možné) a počet kreditů nepřesahoval průměrný počet zapsaných kreditů na semestr.

1.1.4 Předmět

Studijní plán se skládá z několika *předmětů*. Platí, že daný předmět může být součástí několika studijních plánů. V rámci studijního plánu mají tyto předměty vlastní roli (viz. role v sekci 1.1.3). Role představuje určitou „míru důležitosti“ předmětu ve studijním plánu a pro stejný předmět se může lišit v odlišných plánech (např. předmět BI-LA2.21 má ve studijním plánu Počítačové inženýrství 2021 a Počítačová grafika 2021 roli `PS` - povinný předmět specializace, avšak v ostatních plánech má roli volitelného předmětu `V`).

Celkově je předmět bohatá entita ohledně jeho sylabu - obsahuje několik relevantních informací jak pro studenty i vyučující. Hlavní atributy předmětu jsou:

- zakončení
- kredity
- rozsah
- jazyk výuky
- semestr výuky
- garant předmětu
- přednášející
- cvičící
- zajišťující katedra
- anotace
- požadavky
- osnova přednášek
- osnova cvičení
- cíle studia
- studijní materiály
- rozvrh na semestr
- předmět je náhradou

1.1.4.1 Zakončení předmětu

Úspěšné absolvování předmětu vyjadřuje atribut *zakončení předmětu*. Předmět lze na fakultě zakončit pěti různými způsoby:

Z Předmět je zakončen pouze zápočtem. Nemá známkové ohodnocení.

Z, ZK Předmět je zakončen zápočtem a zkouškou. Má známkové ohodnocení na základě bodů ze semestru a ze zkoušky.

ZK Předmět je zakončen zkouškou. Má známkové ohodnocení na základě bodů ze zkoušky.

KZ Předmět je zakončen klasifikovaným zápočtem. Má známkové ohodnocení na základě bodů ze semestru.

1.1.4.2 Rozsah předmětu

Časovou náročnost předmětu (zhruba) vyjadřuje atribut předmětu *rozsah předmětu*. Například rozsah 2P+2C znamená, že předmět bude mít dvě hodiny přednášek a dvě hodiny cvičení týdně.

1.1.4.3 Předmět je náhradou

Atribut „*předmět je náhradou za*“ obsahuje seznam předmětů, které byly nahrazeny daným předmětem. Například předmět BIE-LIN je náhradou za BIE-LA1.21. Více informací k náhradám není v BK specifikováno, zřejmě se jedná o vzájemnou uznatelnost předmětů mezi akreditacemi (více v sekci 1.1.6.2).

1.1.5 Katedra

Různé specializace a předměty jsou zajišťovány *katedrou* podle jejich tématické obsaženosti. Každá katedra má vlastního garanta a jsou k ní vázaní zaměstnanci fakulty. U katedry jsou důležité osoby - garant předmětu, rozvrhář katedry a referent katedry pro studium. Garant předmětu zodpovídá za obsahovou náplň předmětu.

BK obsahuje seznamy předmětů pod danou katedrou. Každá katedra navíc obsahuje svůj vlastní kód a zkratku. K dnešnímu datu obsahuje FIT celkem šest kateder:

- **KTI** - Katedra teoretické informatiky
- **KSI** - Katedra softwarového inženýrství
- **KCN** - Katedra číslicového návrhu
- **KIB** - Katedra informační bezpečnosti
- **KPS** - Katedra počítačových systémů
- **KAM** - Katedra aplikované matematiky

1.1.6 Nová akreditace 2021

V roce 2021 byly akreditovány nové programy pro bakalářské studium [6]. Do studia se vždy lze přihlásit pouze do nového programu akreditace. Výjimkou jsou ti studenti, kteří skončili v průběhu studia staré akreditace a znovu nastoupili do nově akreditovaného programu (mohou zažádat o přestup do svého původního programu). Tedy ve stejném semestru mohou být aktivní např. dvě bakalářské akreditace programu, ale aktivně se vyučuje pouze jedna (více v sekci 1.1.6.2). Získání nové akreditace přineslo několik změn (a potenciálně problémů) do sylabu.

1.1.6.1 Obory a specializace

V předchozích akreditacích se programy dělily na obory, kde některé obsahovaly zaměření (např. bakalářský obor Webové a softwarové inženýrství obsahoval celkem tři zaměření - *Softwarové inženýrství*, *Počítačová grafika*, *Webové a softwarové inženýrství*).

Nová pravidla MŠMT *obory* nahradila *specializacemi*. Současně s oborem zanikl pojem *zaměření*, které v minulosti nebylo téměř vůbec využito. Nové specializace byly přeneseny ze starých oborů, nebo byly vytvořeny nové z předešlých (rozštěpením, sloučením). V současnosti neexistuje návaznost mezi akreditacemi pro specializace, které jsou sesterské (přenesené ze starých oborů). To samé platí pro specializace, které vznikly rozštěpením nebo sloučením předešlých. Lze pouze předpokládat, že na základě podobnosti stavby studijního plánu a názvu budou dané specializace „ekvivalentní".

1.1.6.2 Návaznost předmětů

Podobně jako přechod z oborů na specializace, tak i jejich předměty byly nahrazeny novými verzemi. Pro vzájemnou uznatelnost předmětů slouží tabulka pravidel uznávání předmětu mezi bakalářskými programy [7]. Tabulka 1.1 určuje pravidla pro vzájemné uznávání předmětů, a to směrem ekvivalence nebo implikace starý na nový a opačně nový na starý.

Tabulka 1.1 Příklad vzájemné uznatelnosti předmětů nové bakalářské akreditace 2021 a staré akreditace 2009.

Starý ⇔ Nový		Starý ⇒ Nový		Starý ⇐ Nový	
Starý	Nový	Starý	Nový	Starý	Nový
BI-BEZ	BI-KAB.21	BI-LIN	BI-LA1.21	BI-LIN	BI-LA1.21 + BI-LA2.21
BI-CAO	BI-TZP.21	BI-ZMA	BI-MA1.21	BI-ZMA	BI-MA1.21 + BI-MA2.21
BI-DPR	BI-TDP.21	BI-MLO + BI-ZDM	BI-DML.21	BI-ZDM	BI-DML.21 + BI-MA2.21
BI-EMP	BI-EPP.21			BI-MLO	BI-DML.21 + BI-LOG.21

Jak lze vidět v tabulce, některé předměty prošly značnou změnou a byly buď rozštěpeny nebo sloučeny. Například předměty BI-DML.21 a BI-MA2.21 lze ve staré akreditaci uznat dohromady pouze jako jeden předmět BI-ZDM. A z druhé strany předměty BI-MLO a BI-ZDM lze v nové akreditaci uznat dohromady pouze jako jeden předmět BI-DML.21. Některé předměty netvoří značné problémy a jsou vzájemně (ekvivalentně) uznatelné, např. předměty BI-BEZ a BI-KAB.21.

1.2 Registr vysokých škol

Registr vysokých škol [8] obsahuje přehled a historii o akreditacích českých vysokých škol. Pro FIT obsahuje detailní informace o akreditacích programů a specializací/oborů. Na rozdíl od BK obsahuje navíc např. kód studijního oboru/programu pro lepší identifikaci. Další důležitou informací je *platnost akreditace*. V BK není nikde zmíněna celková životnost (platnost) akreditace - tedy datum získání akreditace a datum platnosti akreditace. Tyto informace by programy a studijní obory zařadily chronologicky na časovou osu.

Další atributem, který není uveden v BK, je *způsob získání akreditace*. Registr zaznamenává i akce ohledně prodloužení akreditací - jedním způsobem získání je totiž *prodloužení akreditace*. Fakulta jej využije v případě dosažení šestileté lhůty (např. bakalářská specializace Teoretická informatika, která má prodlouženou akreditaci až do konce roku 2024). Dalším způsobem je *rozšíření akreditace* (např. bakalářská specializace Znalostní inženýrství, která získala akreditaci rozšířením 09.10.2014) nebo *udělení akreditace* (např. nová akreditace bakalářského programu Informatika v roce 2021).

1.3 KOS

Systém komponenta studium [9] (dále jen KOS) slouží jako systém pro zapisování předmětů, studijních plánů a tvorby rozvrhu studentem. Pro zaměstnance fakulty slouží primárně pro vypisování zkouškových/zápočtových termínů předmětu a udělování klasifikace studentům.

Detail předmětu nabízí podobné informace jako detailní pohled v BK, avšak obsahuje navíc atributy:

- literatura k předmětu
- webová stránka věnující se předmětu
- klíčová slova

Literatura k předmětu je obdobná informace jako studijní materiály v BK. Webová stránka věnující se předmětu obsahuje odkaz do externích systémů a to buď FIT ČVUT Course Pages (viz. 1.4) nebo Moodle-výuka [10]. Tyto systémy slouží jako elektronická databáze studijních materiálů pro studenty a všechny informace, které se daného předmětu v semestru týkají. Klíčová slova jsou seznam slov, které daný předmět vystihují. Slova jsou zaměřena hlavně na tématický obsah předmětu, a jaké hlavní témata se v něm vyučují (typicky nula až deset klíčových slov na předmět). Bohužel nemají v sylabu hlubší význam, neboť několik předmětů tento má tento atribut zcela prázdný, nebo obsahuje irelevantní informace ohledně daného předmětu.

1.3.1 Nový systém KOS

V roce 2021 byl spuštěn nový webový systém KOS [11]. Stejně jako starý KOS obsahuje detailní sylabus předmětu, který je téměř obsahově identický. Nabízí několik minoritních změn, avšak zajímavou změnou je atribut *zástupnost*.

Zástupnost obsahuje seznam předmětů, které jsou s daným předmětem propojený. Například předmět nové bakalářské akreditace BI-DML.21 (více o nové akreditaci v sekci 1.1.6) v semestru B221 obsahuje dvě zástupnosti - předměty BIE-DML.21 a BIK-DML.21. Dle již dříve zmíněných zkratek lze vypozorovat, že první předmět představuje anglickou verzi předmětu a druhý kombinovanou verzi předmětu v dané akreditaci. Typ zástupnosti pro oba předměty je ekvivalence. Další příklad je předmět NI-VEM, který obsahuje v zástupnosti předmět MI-VEM. Dle zkratek lze vypozorovat, že předmět MI-VEM pochází z předchozí akreditace.

Lze tedy říci, že tento systém nabízí „propojení“ verzí předmětů - v rámci jedné akreditace (anglická/česká a prezenční/kombinovaná), nebo přes akreditace (v čase).

1.4 FIT ČVUT Course Pages

Primárním zdrojem pro přehled sylabu předmětu pro daný semestr je systém *FIT ČVUT Course Pages* [12] . Systém obsahuje všechny vypsané semestry a verze předmětů v těchto semestrech. Pro studenty je systém nezbytnou částí studia, neboť zde mohou najít veškeré materiály výuky (přednášky, cvičení), ale zároveň i novinky ohledně výuky předmětu. Samostatné stránky systému jsou vygenerovány aplikací Course Pages, který vygeneruje statické stránky z AsciiDoc souborů napsanými vyučujícími a zaměstnanci fakulty. Proto je obsah modulární, ale zároveň nemá danou strukturu [13].

1.5 Anketa ČVUT

V rámci všech fakult ČVUT mohou studenti vyplnit anketní lístky v systému *Anketa ČVUT* [14]. Po přihlášení jsou dostupné otevřené anketní lístky, které student může vyplnit. K dispozici je také archív výsledků anket. Student vidí jen ankety své fakulty rozdělené podle semestrů. Ohodnoceny mohou být předměty v daném semestru, ale také i role vyučujícího v předmětu v daném semestru. Za zmínku stojí, že tento systém funguje od semestru B182. Předchozím anketním systémem fakulty byl *Anketa FIT ČVUT*, který byl aktivní do zmíněného semestru.

1.6 Fittable

Dalším systémem, který je spojený se sylabem předmětů je *Fittable* [15]. Jedná se o jednoduchou aplikaci, která nabízí zobrazení studentova rozvrhu. V aplikaci lze také vyhledávat rozvrhy předmětů, vyučujících, ale i učeben. Je možné zobrazit i staré rozvrhy (v neaktuálním semestru). Vyučující v dané paralelce obsahují navíc odkaz do svého Usermap profilu, pokud ho mají. Aplikace podporuje i speciální události, jako je např. zkouškový nebo zápočtový termín.

1.7 Projekty FIT

Závěrečné práce spravuje systém *Projekty FIT* [16]. Systém má na starost správu závěrečných prací včetně vyhledání témat pro závěrečnou práci. Vedoucí závěrečných prací vypíšou svá témata, které následně student může vyhledat a vybrat si jej. Systém také nahradil KOS v procesu odevzdávání prací, kde studenti mohou jednoduše odevzdat práci v systému. Jedná se o systém, který se teoreticky propojuje s jednou formou témat (pro závěrečné práce) v sylabu.

1.8 Usermap

Identity zaměstnanců a studentů ČVUT spravuje systém *Usermap* [17]. Zde lze najít jakoukoliv osobu, která má danou roli na univerzitě. Veřejně přístupné osoby jsou zaměstnanci. Studenti se mohou přihlásit do svého profilu, ale dokud nemají přiřazenou zaměstnaneckou roli, tak je jejich profil nedostupný. V profilu lze dohledat veškeré role osoby, kontaktní údaje, údaje o identitě a osobní certifikáty. Možné je také změna hesel a dalších údajů.

Kapitola 2
Vizualizace

Tato kapitola se zabývá rešerší vizualizací. Jsou analyzovány různé typy vizualizací včetně síťových grafů a hierarchií. Také je zmíněn stručný základ ohledně teorie grafů. V jednotlivých sekcích jsou vizualizace analyzovány včetně jejich variant rozložení a možností interakce. Zmíněny jsou i vizuální prvky pro vizualizaci.

Vizualizací můžeme chápat jako grafickou reprezentaci informací a dat. Jejím účelem je transformovat a zobrazit data takovým způsobem, že na výsledném „plátně“ jsou vhodně zachycena. I jedny data obsahují několik pohledů, na které se mohou transformovat. Vizualizace se bude lišit v závislosti na jejím účelu, tedy hlavně to, co by měla definovat. Proto je důležité zvolit vhodný přístup pro reprezentaci dat, tzn. zvolit vhodné vizualizace.

Data pro vizualizaci mohou být více typů - nejčastěji ale však převažuje *kvantitativnost*, nebo *kvalitativnost* dat (a někdy i obojí). Kvantitativní data obsahují až tisíce řádků textového souboru nebo desítky listů tabulek. Jedná se o numerické hodnoty zaznamenané v několika fázích nebo iterací. Výsledná vizualizace může pozorovat např. vývoj a změnu těchto hodnot v čase a vizualizovat jejich množství. Naopak kvalitativní data nemohou být jednoduše vyměřena čísly. Jejich důležitost spočívá v textovém obsahu, který vyjadřují. Jedná se o vyjádření vztahů, struktur, myšlenek a souvislostí. Výsledná vizualizace může zkoumat např. závislosti a souvislosti slovních dat nebo hierarchii domény.

Jelikož získaná data jsou poměrně velká, bude třeba využít i několika „artistických“ prvků pro její správnou reprezentaci. Jedná se například o různé barvy, tvary, osy, popisky, velikosti, písma, tloušťky nakreslených čar a jiné grafické elementy. Prostor plátna je přece jenom v zásadě omezený na rozdíl od tabulek a textových dokumentů. Také je třeba myslet na uživatelskou přívětivost a intuici vůči elementům vizualizace. Podobně jako grafické návrhy pro vzhled webových stránek, tak i interaktivní vizualizace potřebují vhodné uživatelské rozhraní (*user interface*, dále jen UI).

Existuje zcela neomezeně mnoho možností, jak tyto data vizualizovat. Samozřejmostí je popularita již existujících řešení a daný standard pro vizualizace některé skupiny dat (např. časová osa pro znázornění chronologických dat). Avšak v dnešní době nejsme omezeni pouze na 2D plátna, ale stávají se populární i 3D prostorové vizualizace (např. za pomocí virtuální reality). Proto je třeba vizualizace řádně kategorizovat. Nejčastější vizualizace dat jsou reprezentovány pomocí grafů, diagramů a map - např. sloupcovými grafy, kruhovými diagramy, časovými osami, světovou mapou, myšlenkovou mapou ale i UML (*Unified Modeling Language*) diagramy.

V předchozí kapitole (viz. kapitola 1) byly analyzovány entity sylabu na FIT ČVUT. Účelem této práce je vybrat a navrhnout vhodné vizualizace pro znázornění vazeb a závislostí těchto entit, a to na základě tématického obsahu, ale i přes akreditace (tedy v čase). Proto jsou následující sekce věnovány rešerši pouze vybraným kategoriím vizualizací zaměřené na kvalitativní data:

- síťová vizualizace
- hierarchická vizualizace

2.1 Síťová vizualizace

Základním elementem síťových vizualizací je *síťová teorie* (network theory). Jedná se o podmnožinu *teorie grafů*. Sítě jsou definovány jako grafy, které obsahují atributy - jak pro vrcholy, ale i hrany [18]. Rozšiřují se s růstem online sociálních sítí (např. *Twitter* nebo *Facebook*), neboť dokážou vyjádřit jakým způsobem jsou informace propojené.

V teorii grafů můžeme (jednoduchý) graf popsat jako hrany mezi dvojicemi vrcholů pomocí dvouprvkových podmnožin vrcholů. Přesněji dle definice tzv. *obyčejného (neorientovaného) grafu* [19]:

▶ **Definice 2.1** (Neorientovaný graf)**.** *Neorientovaný graf je uspořádaná dvojice $G = (V, E)$, kde V je množina vrcholů a E je množina hran – množina vybraných dvouprvkových podmnožin množiny vrcholů.*

Obyčejný graf je tedy jednoduchý graf s vrcholy a obousměrnými hranami. Neorientované hrany jsou (ve většině případech) označovány jednoduchou spojnicí (hranou) bez orientace (obrázek 2.1. Definujeme také grafy jejichž hrana vyjadřuje orientaci (směr), tzv. *orientované grafy* [19]:

▶ **Definice 2.2** (Orientovaný graf)**.** *Orientovaný graf je uspořádaná dvojice $D = (V, E)$, kde $E \subseteq V \times V$.*

(a) Neorientovaný graf (b) Orientovaný graf

Obrázek 2.1 Zvýraznění hrany v orientovaném a neorientovaném grafu s vrcholy V=({A,B})

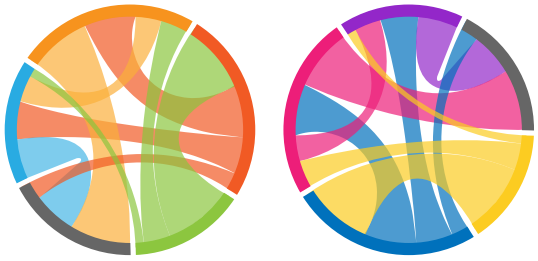
Síťové grafy jsou tedy reprezentovány grafy - buď s orientovanými, nebo neorientovanými hranami. Co může reprezentovat v reálném světě vrcholy a hrany grafu? Například již zmíněný příklad - vrcholy sítě jako uživatelé sociální sítě Twitter a hrany jako interakce mezi uživateli. Nebo lze rozšířit doménu o další typy entit - ve stejném grafu vrcholy reprezentují entity uživatelů, příspěvků (*tweetů*) aj. Takový to typ grafu se nazývá *znalostním grafem* (více v sekci 2.1.4).

2.1.1 Vzhled vrcholů a hran

Jakým způsobem budou grafové prvky reprezentovány je velice specifické, tedy existuje jich mnoho. Většinou jsou vrcholy reprezentovány kruhem a hrany jsou reprezentovány přímkou. V závislosti na orientaci hran jsou případně použity šipky pro vizualizaci orientaci hrany (viz obrázek 2.1). Avšak existují i robustnější vizualizace, např. oblouky místo přímek (tzv. *arc diagram*) nebo části kružnice místo vrcholů (např. *chord diagram* na obrázku 2.2).

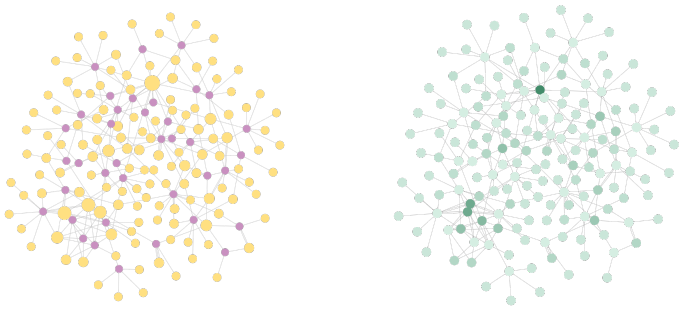
Vhodným elementem pro zobrazení vrcholů je využít jeho stupeň (*degree of centrality*), definováno [19]:

▶ **Definice 2.3** (Orientovaný graf)**.** *Stupněm vrcholu v v grafu G rozumíme počet hran vycházejících z v. Stupeň v v grafu G značíme $dG(v)$.*



Obrázek 2.2 Chord diagram s okraji reprezentující vrcholy a oblouky jako spojnice mezi nimi [20]

V husté síti vrcholů se stejnou velikostí nelze jednoduše vypozorovat vrcholy s největším „vlivem“ a závislostí. Jednoduché a elegantní řešení je škálovat velikost vrcholu - s rostoucím stupněm vrcholu roste i jeho velikost. Druhou variantou může být obarvení vrcholů *gradientem* - s rostoucím stupněm vrcholu je gradientové zabarvení výraznější. Tím je pro uživatele (pozorovatele) jednodušší zaměřit se na hlavní sekce grafu. Pro vrcholy je také typické různé obarvení, které je vhodné v případě, že vrcholy lze kategorizovat nebo zařadit do intervalu (myšleno jejich atribut), např. typ entity ve znalostním grafu (více v sekci 2.1.4). Na obrázku 2.3 lze vidět využití stupňů vrcholů a jeho zvýraznění v obou zmíněných formách výše.



(a) Velikost vrcholů (b) Zabarvení vrcholů

Obrázek 2.3 Síťový graf ve dvou formách vizualizace. Vizualizováno pomocí nástroje Neo4j Bloom.

Podobná situace platí i pro hrany. Čím silnější vazba mezi vrcholy, tím silnější přímka mezi nimi. Opět příklad se sociální sítí Twitter - čím více interakcí mezi dvěma uživateli, tím je mezi nimi silnější hrana.

2.1.2 Rozložení sítě

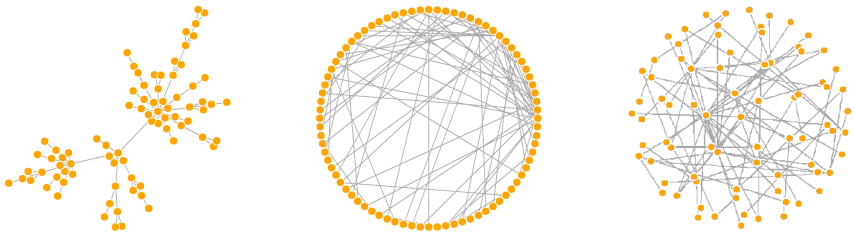
Mimo vizualizaci samostatných elementů grafu (vrcholů a hran) je důležité myslet na jeho rozložení (*layout*). Rozložení totiž přesně určuje, na jaké pozici se vrcholy na plátně nacházejí. S rostoucím počtem vrcholů se tato úloha může stát těžko vyřešitelná, tak aby síť byla vizuálně „příjemná“ (lze vidět na obrázku 2.4). Cílem je najít optimální rozložení vrcholů v síti. Mezi hlavní typy rozložení sítě patří [21]:

- force-directed
- radiální
- 3D sphere

2.1.2.1 Force-directed

Force-directed (tzv. „silově orientovaný“) rozložení umisťuje vrcholy na pozice pomocí simulující síly na vrcholech a hranách. Hlavní účel tohoto rozložení je umístit vrcholy zcela „přirozeně“, neboť algoritmy force-directed rozložení hledají nejvíce optimální souřadnice vrcholů na plátně. Jak lze vidět na obrázku 2.4, optimální rozložení kompletně mění vzhled sítě a přidává na hodnotě její přehlednosti.

Například ve vizualizační knihovně D3.js (*Data-driven documents*) [22] je toto rozložení interpolováno pomocí tří sil - přitažlivost/odpor vrcholů (*attraction/repulsion*), gravitace vrcholů ke středu grafu (*gravity*) a vzájemná přitažlivost podle hran (*link attraction*) [23]. Většina force-directed algoritmů je založeno na tomto principu, realizují ho např. algoritmy *Barabasi-Albert* nebo *Fruchteman-Reingold* [18]. Další variace force-directed algoritmu může vzít v potaz např. detekci komunit (viz. 2.1.3).



(a) Force-directed (b) Radiální (c) 3D Sphere

Obrázek 2.4 Síťový graf ve třech rozloženích [21]

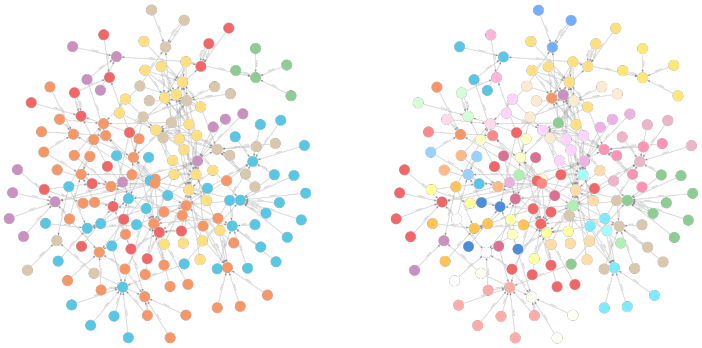
2.1.3 Detekce komunit

Detekce komunit (*community detection*) je populární problém v síťové analýze, který zahrnuje rozdělení vrcholů grafu do skupin na základě úzké souvislosti[24]. Skupiny zvané *komunity* jsou shluky obsahující tyto vrcholy. V rámci komunity jsou vrcholy úzce propojené, ale mezi komunitami jen řídce. Znamená to, že entity ve stejné komunitě jsou si nějakým způsobem blízké a často spolu interagují. Tímto se můžeme zaměřit se na společné vzory v síti a pozorovat určité „trendy“. Rozdělení grafu do komunit tak poskytuje nový úhel pohledu do sítě. Jedná se o sociální sítě, biologické sítě, komunikační sítě aj. Například sociální sítě typicky rozdělují její uživatele do několik komunit, kde každou lze dále detailněji analyzovat. Toto může být užitečné v jakékoliv aplikaci, kde je akce zaměřena na specifickou skupinu. Tím jsou modely pro automatické doporučení uživatele, produktu, reklamy [25], ale také modelování aktuálních témat na sociálních sítích [26].

Existuje mnoho algoritmů a metod pro detekci komunit v grafu, každá z nich má vlastní výhody a nevýhody. Například grafový databázový engine *Neo4j* implementuje ve své knihovně *Graph Data Science* metodu *Louvain* pro detekci komunit v síťovém grafu[27]. Je to tzv. *greedy* metoda a na rozdíl od některých jiných přístupů nevyžaduje na vstupu znát počet komunit, který se má detekovat [28]. Dalším algoritmem je např. *Label propagation*, který je také implementován ve stejné knihovně (použití lze vidět na obrázku 2.5).

Při vizualizaci síťového grafu může toto rozdělení přinést značnou transformaci vzhledu sítě. Místo husté a neorganizované sítě budou v síti vyznačeny jednotlivé skupiny a tím se stává mnohem více přehlednou, což je také důležité z uživatelského hlediska. Komunity jsou rozděleny odlišnými barvami a přitaženy simulovanými silami k sobě. Popřípadě mohou být obaleny pomocí konvexní obálky (*convex hull*) pro jasné definování komunity. Konvexní obálka představuje

nejmenší konvexní množinu, která danou množinu obsahuje [29]. Užitečný může být také štítek u komunity s názvem nejvíce vlivného vrcholu pro jednotné a rychlé definování skupiny.



(a) Louvain (b) Label propagation

Obrázek 2.5 Dvě varianty detekce komunit v aplikaci Neo4j Bloom

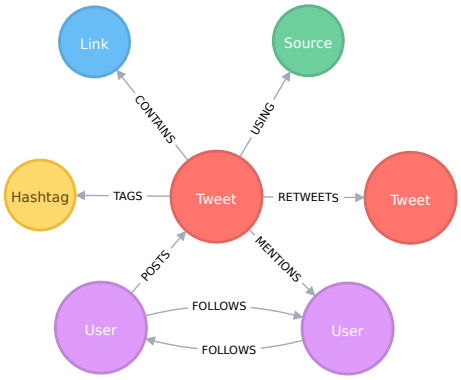
2.1.4 Znalostní graf

Znalostní graf (*knowledge graph*) je pojem, který nabývá několika definicí a v poslední řadě let se zvyšuje o jej zájem [30]. Poprvé se objevil v roce 2012 od společnosti Google, která jej reprezentuje jako digitální strukturu dat, které jsou propojené mezi sebou [31]. Využívá jej výhradně ve svém vyhledávacím enginu, který pomocí znalostního grafu dokáže zobrazit odkazy na relevantní informace k zadanému dotazu. Celková idea je propojit data na základě jejich atributů tak, aby byla rychle vyhledatelná.

Další oblastí, kde se pojem znalostních grafů vyskytuje jsou *grafové databáze*. Jedná se např. o databázový engine *Neo4j*, kde ukládání dat funguje na bázi znalostního grafu. Entity databáze jsou reprezentovány vrcholy a relace (ne)orientovanými hranami. Schéma databáze je pak reprezentováno jako znalostní graf, který mapuje všechny jeho možné typy entit a relací, které se v grafu vyskytují. Vizualizaci těchto schémat a grafů umožňuje nástroj *Neo4j Bloom* (více v sekci 3.1). Schéma lze vidět na obrázku 2.6, který reprezentuje část domény sociální sítě Twitter.

Tímto se dostáváme ke struktuře a vizualizaci grafu samotného. Znalostní graf totiž obsahuje vrcholy jako entity reálného světa a hrany jako relace mezi nimi [32]. Entity a hrany obsahují atributy, které jej popisuje. Například jak lze vidět ve schématu (obrázek 2.6), entita *Tweet* je v relaci sama se sebou přes atribut *retweets*, nebo s entitou *User* dvěma způsoby - přes atribut *posts* a *mentions*. Tím se odlišuje od klasického síťového grafu, který v zásadě nerozlišuje typy entit.

Pro tento model je důležité barevná odlišnost a *legenda grafu* pro jednotlivé typy entit (vrcholů). Velikost vrcholů je u všech entit stejná. V tomto typu zobrazení je kladen důraz na *význam* entit a hran. Tedy to, co vyjadřují a v jakém jsou navzájem vztahu. To samé platí pro hrany a jejich zobrazení, silnější a slabší hrany ve znalostním grafu v zásadě neexistují. Hlavní je význam hrany a relace k entitě. Hrany mohou být orientované, i neorientované a jsou charakterizovány vlastním atributem, který popisuje typ relace. Tím se také odlišuje od jednoduchého síťového grafu, který hrany odlišuje jen na základě spojujících vrcholů.



Obrázek 2.6 Schéma znalostního grafu jako schéma databáze v databázovém enginu Neo4j [33]

2.2 Hierarchická vizualizace

Hierarchie (již zmíněná v kapitole 1 v analýze entit BK) je uspořádání prvků do úrovní. Úroveň prvku definuje jeho pod/nadřazenost vůči ostatním prvkům. Také lze tento vztah označit jako tzv. *parent-child relationship* [23]. Jedná se o vizualizaci komplexnějších dat - kvalitativních, než kvantitativních. Důležité není množství a přesná numerická data, ale raději uspořádání a vztahy. Využití hierarchických vizualizací existuje v každé doméně: rodokmeny, uspořádání do kategorií nebo i tzv. org-charts (*organizational charts*). Taková vizualizace zobrazuje strukturu organizace a jejich zaměstnanců (typicky jejich role a hierarchické uspořádání).

Důležitý aspekt hierarchické vizualizace je její přehlednost a nadhled nad strukturou dat. Data lze vizualizovat „shora dolů“, kde nejvyšší úroveň je umístěna (nejčastěji) v horní části plátna, nebo taky pomocí mapy, která zanořuje grafické elementy „do sebe“ a tím definuje podřazenost a nadřazenost prvků. V této části jsou probrány oba přístupy.

2.2.1 Dendrogram

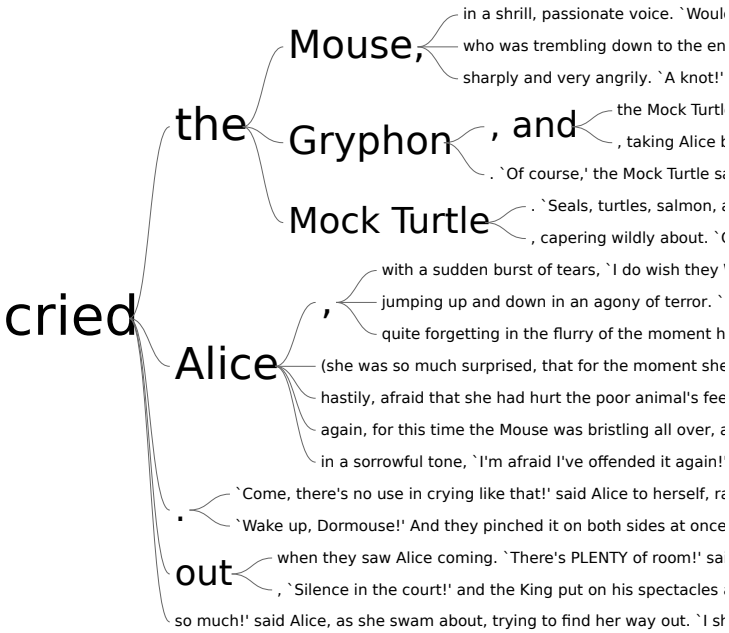
Od klasických sítí, které jsou založeny na (ne)orientovaných grafech, se posuneme ke *stromům*. Stromy představují souvislé grafy (tedy tvořící jednu komponentu) bez kružnic (tím je jasně daná pod/nadřazenost) [19]. Strom obsahuje jeden kořenový uzel (*root*), který se nachází na nejvyšší hladině hierarchie. Další vrcholy ve stromu hrají roli rodiče (*parent*) a potomka (*child*).

▶ **Definice 2.4** (Strom)**.** *Strom je jednoduchý souvislý graf T bez kružnic.*

Od stromů se dostáváme k dendrogramům. Jedním ze způsobů hierarchické vizualizace jsou *dendrogramy* (od slova *dendro* - znamená „strom“) [23]. Dendrogramy jsou diagramy vizualizující stromy. Obsahují vrcholy uspořádané v úrovních a hrany jako spojující závislost mezi nimi. Jsou vhodné pro vizualizaci propojenosti mezi jednotlivými rodiči a potomky, například pro analýzu stavby věty ve souvislém textu pomocí tzv. *word tree* [34] (viz. obrázek 2.7).

2.2.1.1 Rozložení

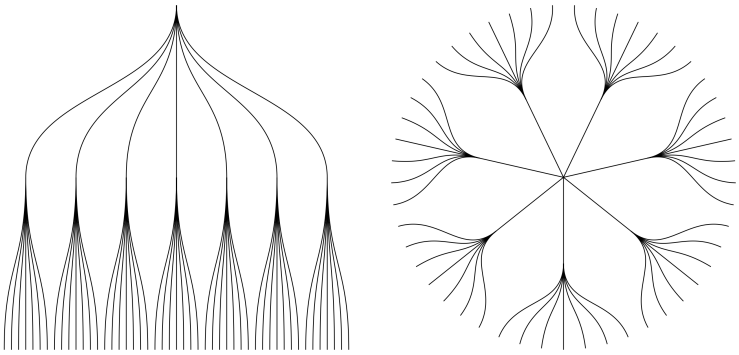
Dendrogramy mohou mít několik rozložení, podobně jako u sítí v sekci 2.1.2. Typickým rozložením je rozložení tzv. *shora dolů*, které umístí kořenový vrchol na nejvyšší možnou pozici vertikálně. Potomci se potom řadí na další hladiny směrem dolů (viz. obrázek 2.8). Obdobně lze kořen umístit na levou (popř. pravou) část horizontálně a potomci se řadí na další hladiny směrem zleva doprava (nebo naopak). Výhodou této metody je čtení grafu z jedné strany na druhou a



Obrázek 2.7 Word tree analyzuje jednotlivá slova textu a co po nich následuje. Text je vizualizován dendrogramem, který byl vykreslen pomocí nástroje od Jasona Daviese [35]

jasná vizualizace úrovní. Naopak nevýhodou je fakt, že tato metoda rychle spotřebuje veškeré dostupné místo a není tím prostorově efektivní.

Stejný strom lze rozložit i *radiálním* rozložením, kde kořen je naopak ve středu a jednotlivé hladiny stromu se spojují v kružnici. Každá hladina je potom definována kružnicí okolo kořene, čím větší průměr kružnice, tím větší zanoření (viz. obrázek 2.8). Výhodou této metody je rovnoměrné rozprostření potomků do všech směrů prostoru.



(a) Shora dolů (b) Radiální

Obrázek 2.8 Dendrogram ve dvou rozloženích [36]

2.2.2 Circle packing

Circle packing (v překladu „kruhové zabalení“) je dalším způsobem jak vizualizovat hierarchická data [20]. Data jsou zmapována do kruhového zabalení, kde každá kategorie je graficky umístěna do svého rodičovského kruhu. Dohromady tak tvoří vizualizaci vnořených kruhů. Na obrázku 2.9 lze vidět namapovaná hierarchie z klasického dendrogramu do kruhového zabalení.

Jednotlivé úrovně hierarchie jsou rozlišeny zabarvením a velikostí obalovaného kruhu. Velikost podkategorie je definována daty. Pokud není velikost specifikována, tak mají všechny kruhy rovnoměrně rozloženou velikost. Algoritmus optimálního rozložení pak dopočítá velikosti rodičovských kategorií. Barvy grafu lze stupňovat s úrovní zanoření (lze vidět na obrázku 2.9), nebo rozdělit podle např. nejvyšší podkategorie. Popisky kategorií jsou typicky umístěny uprostřed kruhů, avšak existuje i varianta umístit je např. podél kruhové cesty.

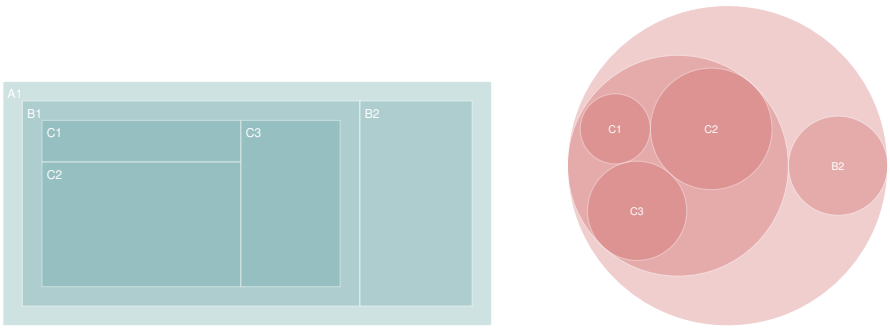
Výhoda této metody spočívá ve vizualizaci vnořených dat, než znázornění propojení a vztahů (viz. dendrogram výše). Také je její použití výhodné po znázornění velikosti kategorií. Avšak i přesto, že vizualizace může být vizuálně uspokojivá, tak tato metoda po sobě zanechává značné množství nevyužitého místa v kruzích. Při větším množství elementů lze rychle ztratit přehled nad hierarchií. Zároveň je kladen důraz na elementy na nejnižších úrovních, což může ale i nemusí být vždy žádané.

2.2.3 Treemap

Obdobná verze kruhového zabalení s obdélníky se nazývá *treemap* (v překladu „stromová mapa“) [20]. Treemap vizualizuje kategorie jako vnořené obdélníky, přičemž je schopný zobrazit i velikost každé kategorie přes obsah plochy. Každá kategorie (rodič) má svojí obdélníkovou plochu a její podkategorie (potomci) jsou vnořené uvnitř.

Velikost podkategorie je definovaná daty a při každé změně se změní i obsah obdélníkové plochy. Zároveň zůstává v proporci s kategoriemi na stejné úrovni. Pokud data nedefinují žádnou velikost kategorie tak je plocha rovnoměrně rozložena. Rozložení obdélníků záleží na použitém algoritmu, avšak tzv. *squarified algorithm* se pokouší obdélníky udržovat co nejblíže k podobě čtverce. Jiné algoritmy jej dodržovat nemusí, místo toho jsou v podobě např. „podlouhlých obdélníků“. Podobně jako u kruhového zabalení, tak pro různé úrovně nebo kategorie lze definovat zabarvení (např. na základě nejvyšší podkategorie).

Původ mapy spočívá ve vizualizaci adresáře souborů v systému. Dokáže totiž efektivně zobrazit hierarchii adresáře na malém plátně. Na obrázku 2.9 lze vidět vizualizaci mapy, která sahá do třetí úrovně hierarchie. Naopak nevýhodou je jednoduchá ztráta přehlednosti nad hierarchií s rostoucím počtem úrovní (podobně jako u kruhového zabalení výše).



(a) Treemap (b) Circle packing

Obrázek 2.9 Hierarchická vizualizace ve formě mapy ve dvou variantách [37]

Kapitola 3
Vizualizační aplikace

Tato kapitola se zabývá analýzou vybraných současných řešení aplikací, které obsahují vizualizaci jako jednu z hlavních komponent její funkcionality. Zaměřuje se na vizualizace, které byly analyzovány v předchozí kapitole. Důležitým bodem je prozkoumání interakce s vizualizací, ale i samostatné uživatelské rozhraní aplikace.

V předchozí kapitole byly analyzovány komplexní vizualizace dat, které se zaměřují více na závislosti a souvislosti dat. Průběžně bylo také zmíněno několik příkladů, kde se různé metody vizualizací v praktických úlohách využívají. V podstatě se může jednat o jakékoliv domény dnešního světa.

Tato kapitola analyzuje vybrané (webové) aplikace, které tyto vizualizace využívají (konkrétně tedy síťové a hierarchické vizualizace). Celkově rozšířenější variantou je vizualizace sítí, které jsou i flexibilnější. Všeobecně platí, že struktura hierarchie není přítomna v každém grafu, ale každá hierarchická struktura může být zobrazena síťovým grafem. Zároveň má v reálném světě taky více případů užití.

Rozdíl oproti klasickým webovým aplikacím je fakt, že uživatelé nejsou na takový to typ interakcí „zvyklý“. V dnešní existenci tisíce webových stránek byl vybudován jistý standard, který se osvědčil a je do dnes opakovaně používán. Navíc absolutní většina takových aplikací by ani nenašla ve svých případech užití prostor pro komplexní interaktivní vizualizace.

3.1 Neo4j Bloom

Každá aplikace potřebuje prostor pro ukládání dat, typicky databáze. Tradiční relační databáze pro ukládání dat využívají formu tabulek s atributy. Vztahy mezi tabulkami jsou vyjádřeny relacemi, které se tvoří za pomocí cizích klíčů (*foreign keys*) v tabulkách. Zároveň jsou omezeny na vztahy typu *many-to-many*, *one-to-many* (a naopak) apod. Znamená to, že několik řádek tabulky (tedy dat) jsou asociovány s několika (nebo jednou) řádek jiné dané tabulky. Tento koncept může být v určitých situacích omezující a nevhodný, protože spojení několika tabulek dohromady může být výpočetně náročné vůči relačnímu modelu databáze.

Grafové databáze jsou v dnešní době čím dál více populárnější variantou oproti klasickým relačním nebo dokumentovým. Pro ukládání dat využívají totiž formu *vrcholů* a *hran*, tedy grafu (oproti tabulkám nebo dokumentům). Entity jsou ve formě vrcholů a vztahy mezi entitami jsou vyjádřeny hranami mezi vrcholy. Jednotlivé informace a atributy entit (obdoba atributů tabulky v relačních databázích) jsou vázány k vrcholům. Avšak grafové databáze kladou důležitost i na vztahy mezi těmito vrcholy, tedy na hrany grafu.

V reálném světě je důležité důkladně porozumět vztahům, tedy co přesně *vyjadřují*, od kdy

do kdy byl vztah navázán apod. Relační databáze sice mají schopnost vyjadřovat různé vztahy a pohledy na data, ale ukázalo se, že manipulace se vztahy je složitá a často i výpočetně náročná. Jedná se především o spojování několika tabulek (tzv. *join*) nebo jiné komplikované procedury při získání dat a použití dotazovacích jazyků (*query language*), např. *SQL* pro relační databáze.

Přínos grafových databází je ve využití grafových vlastností. Nabízejí totiž vyjádření vztahů ve mnohem flexibilnější podobě a zároveň i jejich detailnější popis. V grafové databázi není třeba spojení několika tabulek dohromady za účelem vyjádření specifických vztahů a získání dat, nemají totiž žádné joiny apod. Důležitý je koncept: „vše je navzájem propojené“. Jako v klasické grafové teorii, i grafové databáze mají výhodu v použití různých grafových algoritmů. Například algoritmy pro detekci komunit (zmíněno v sekci 2.1.3), BFS (*Breadth-First-Search*) pro hledání nejkratší cesty apod.

Neo4j je *NoSQL* (někdy také „not only SQL“) [38] grafová databáze, která tento princip využívá [27]. Entity a relace mezi nimi jsou reprezentovány znalostním grafem (viz. v sekci 2.1.4). Celkově nabízí spoustu užitečných nástrojů pro příjemnější práci s grafovou databází. Jedním z těchto nástrojů je *Neo4j Bloom*, který mimo jiné nabízí [39]:

- pokročilé vyhledávání a filtrování v grafu
- zobrazení detailu entity a jejich relací
- perspektivy grafu
- grafové algoritmy aj.

3.1.1 Perspektivy

Jako první část při vstupu do aplikace je výběr *perspektivy* grafu. Perspektiva obsahuje podgraf původního grafu s definovaným nastavením. Výchozí perspektivou je původní graf s výchozím nastavením (žádné nastavení). Nastavením může být:

- filtrování entit a vztahů
- rozložení grafu
- aplikování algoritmů
- stylování (barvy, ikony, popisky)

Výběr perspektivy proběhne ještě před samostatným zobrazení obrazovky s grafem. Uživatel vlastní své perspektivy a jejich výběr je možné zobrazit v sekci *Perspectives*, které obsahují jejich seznam. Perspektiva je definována názvem a typy entit, které obsahuje. Lze je vytvářet, přejmenovat, smazat nebo exportovat. Díky této funkcionalitě je možné zobrazit graf z několika úhlů pohledu. Využití spočívá hlavně v business reprezentaci dat, neboť určité pohledy jsou relevantní pro dané oddělení uživatelů (obecně skupiny uživatelů). Celkově je tato funkcionalita užitečná pro personifikaci dat v grafu, např. na základě různých rolí uživatelů.

3.1.2 Prozkoumání grafu

Po výběru perspektivy je uživatel přemístěn do hlavní sekce aplikace, kde se nachází graf vybrané perspektivy. Graf lze prozkoumat především vyhledáváním nebo filtrováním.

3.1.2.1 Vyhledávání

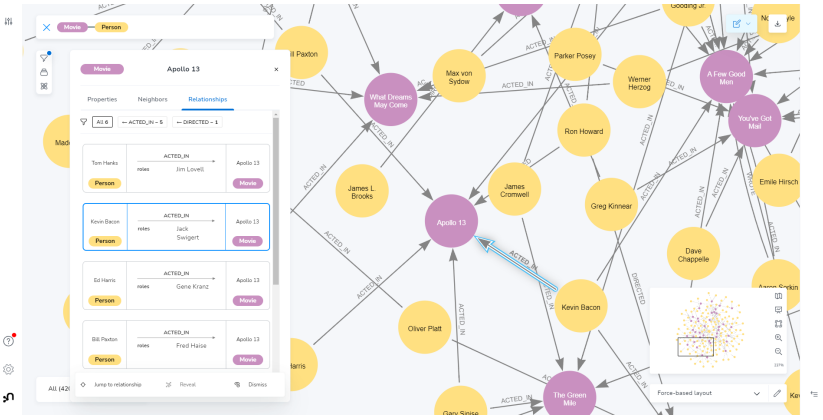
Vyhledávací panel se nachází v levém horním rohu scény s grafem. Vyhledávání je přizpůsobeno syntaktické podobě grafu. Lze najít samostatné entity na základě primárního atributu (jména, identifikace) nebo na základě jejího typu, který reprezentuje ve svém databázovém schématu (příklad schématu lze vidět na obrázku 2.6). Kromě hledání entit je možné hledat i konkrétní typy vztahů mezi entitami, nebo dokonce více entit podle definovaného hledacího vzoru. Výsledkem hledání je podgraf obsahující entity a vztahy, které odpovídají vyhledávajícímu filtru. Výsledek je také zobrazen ve formě klasického seznamu, který je rozbalen po kliknutí na panel výsledků v levém dolním rohu.

3.1.2.2 Filtrování

Další možností je filtrování grafu. Filtrování lze aplikovat na entity, tedy vrcholy, grafu. V levém horním rohu lze otevřít panel s možností přidání filtru. Jak již bylo řečeno předtím, každá entita a relace obsahuje vlastní informace ve formě atributů. Samostatný filtr lze provést na tyto atributy. Lze tedy aplikovat komplexní filtrování pro textové řetězce, numerické hodnoty ve formě intervalu a jiné. Panel také obsahuje možnost pro vypnutí/zapnutí jednoho filtru. Výsledkem filtru je graf se zašedlými (nesplňující filtr) a barevnými elementy (splňující filtr).

3.1.3 Detail entity a relace

Po dvojitém kliknutí na vrchol grafu reprezentující entitu je zobrazen panel s detailem entity. Panel entity obsahuje hlavičku a záložky pro atributy (*properties*), sousedy (*neighbors*) a relace (*relationships*). Hlavička obsahuje typ entity a jméno. Atributy obsahují možnost pro editaci atribut a seznam jejich hodnot. Sousedé jsou všechny entity které jsou spojeny s vybranou entitou hranou (odchozí i příchozí). Relace zobrazují seznam všech vztahů s danou entitou. Vztah je detailně reprezentován zdrojovým vrcholem a cílovým vrcholem, ale i hranou a jejími atributy (lze vidět na obrázku 3.1).



Obrázek 3.1 Neo4j Bloom: Detail entity *Apollo 13*

3.1.3.1 Rozšířená entita

Každou entitu lze vybrat a oddělit od grafu pomocí kontext menu, které se zobrazí po pravém kliknutí na entitu. Po oddělení lze použít možnost *expandování*, která zobrazí vrcholy a hrany pouze těch entit, které jsou s vybranou entitou sousední (lze vidět na obrázku 3.2). Graf tím získá podobu tzv. hvězdy (*star*).

Obrázek 3.2 Neo4j Bloom: Expandovaná entita *Keanu Reeves*

3.1.4 Aplikování algoritmů

Aplikace nabízí několik funkcionalit, které lze na graf aplikovat. V zásadě obsahuje několik typů algoritmů a pro každý typ několik možností provedení:

- detekce komunit
- stupně vrcholů
- nejkratší cesta

Panel s algoritmy lze otevřít v levém horním rohu rozhraní. Na výběr je celkem sedm algoritmů. Následně lze algoritmus aplikovat na graf. Aplikovaný algoritmus pak obsahuje možnosti po úpravu vzhledu grafu - pro detekci komunit odlišné barvy komunit (viz. obrázek 2.5) a pro stupně vrcholů gradientové obarvení, nebo velikost vrcholů (viz. obrázek 2.3). Odděleně lze také najít nejkratší cestu mezi dvěma vrcholy, cesta je potom znázorněna na grafu.

3.2 InfraNodus

Myšlenkové mapy jsou schopny vizualizovat komplexní nápady a myšlenky. Zároveň mají pozitivní vliv na rozvoj myšlení. Lze je vizualizovat pomocí již zmíněných síťových vizualizací. Síťové vizualizace se často používají s kombinací umělé inteligence (*aritifical intelligence*, dále jako AI) a textové analýzy. Aplikování kvalitativních textových dat na koncept síťového grafu umožní hlubší porozumění kontextu a tématických souvislostí domény. Vrcholy grafu reprezentují myšlenky a koncepty, zatímco hrany relace mezi nimi.

Takové principy využívá open-source webový nástroj *InfraNodus* [40]. Ve svých funkcionalitách využívá kombinaci různých přístupů přes textovou analýzu, síťovou analýzu, vizualizaci dat, umělou inteligenci aj. Tento nástroj poskytuje myšlenkovou mapu pro lepší organizaci a porozumění poznámek, na které je následovně aplikována síťová analýza. InfraNodus implementuje metodu pro textovou analýzu založenou na mapování textu do síťového grafu. Vizuál aplikace (front-end) byl implementován pomocí knihovny sigma.js (více v sekci 4.5). Pro backend část aplikace byla použita grafová databáze Neo4j (viz. sekce 3.1).

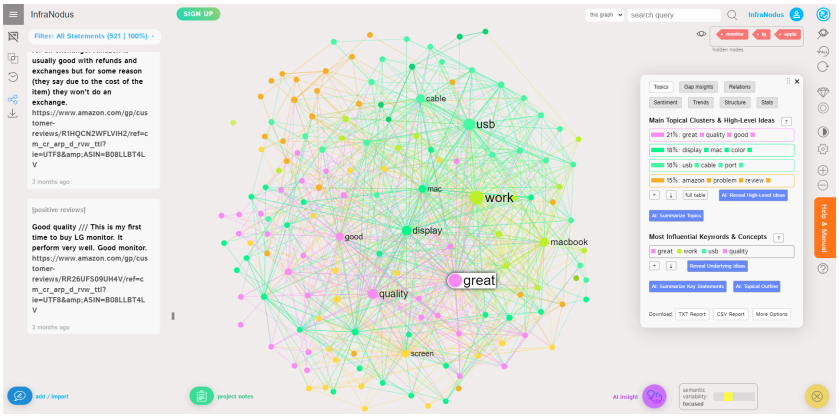
Mapování textu do grafu probíhá v několika krocích. Prvním je textová normalizace, kde všechny varianty slova jsou převedeny do jedné sjednocené (např. anglická slova „books“ a „booked“ se sjednotí do slova „book“). Dalším krokem je vymazání neúčelových slov. Taková slova nemají hlubší význam pro textovou analýzu. Jedná se většinou o předložky, spojky, ale i některá slovesa a podstatná jména (např. anglická slova „is“ nebo „a“ aj.) Poté je text převeden to grafové podoby. Vrcholy grafu reprezentují slova a hrany jejich společný výskyt. Tedy pokud nastává

hrana mezi dvěma slovy, tak byly nalezeny ve stejném textu. Váha hrany je určena podle počtu slov, které je mezi sebou odděluje. Toto byl stručný popis pouhé části procesu, který InfraNodus využívá. Detailní popis lze získat z jejich publikace [41].

3.2.1 Výběr grafu

Aplikace nabízí několik ukázkových grafů, které analyzují různé domény včetně zpravodajských článků dne, nejpopulárnějších příspěvků týdne ze sociální sítě Twitter nebo recenze produktů v obchodním řetězci Amazon. Také je možné vložit vlastní data a vygenerovat vlastní graf.

Po výběru grafu je zobrazena hlavní část aplikace s grafem. Pro tuto rešerši byl vybrán graf, který obsahuje recenze na produkt obchodního řetězce Amazon (lze vidět na obrázku 3.3).



Obrázek 3.3 InfraNodus: Rozhraní grafu recenzí produktu

3.2.2 Rozhraní grafu

Hlavní prostor obrazovky je věnován samostatnému grafu. Graf je implicitně zabarven podle detekovaných tématických shluků (více v sekci 3.2.4.2) a obsahuje neorientované hrany. Velikost vrcholu je daná stupněm vrcholu. Zdá se, že největším vrcholem je slovo „great“, což nejspíše implikuje, že zákazníci byli s produktem spokojeni (viz. obrázek 3.3).

3.2.3 Prozkoumání grafu

V grafu lze vyhledávat pomocí panelu v horní pravé části. Hledání vrcholů je provedeno jednoduše na základě obsahu slova. Nalezený vrchol je pak označen v grafu, ostatní vrcholy jsou skryty. Dále lze také skrýt nebo označit vrcholy podle filtru. Tento filtr je proveden na základě štítků v pravém horním rohu obrazovky. Štítek lze přidat do skrytých vrcholů nebo vybraných vrcholů. Vizualizace grafu se změní na základě filtru - šedé obarvení pro skryté vrcholy, barevné obarvení pro vybrané vrcholy a všechny jeho sousedy (lze vidět na obrázku 3.4). Zajímavá je funkcionalita, která zvýrazní vybraná slova v grafu také v datech v levé části obrazovky. Uživatel pak může rychle analyzovat i textová data.

3.2.4 Funkcionality

Aplikace nabízí spoustu funkcionalit pro detailnější analýzu grafu. Patří mezi ně:

- porovnání grafů

- tématické shluky
- analýza sentimentu textu
- exportování dat
- statistiky aj.

3.2.4.1 Porovnání

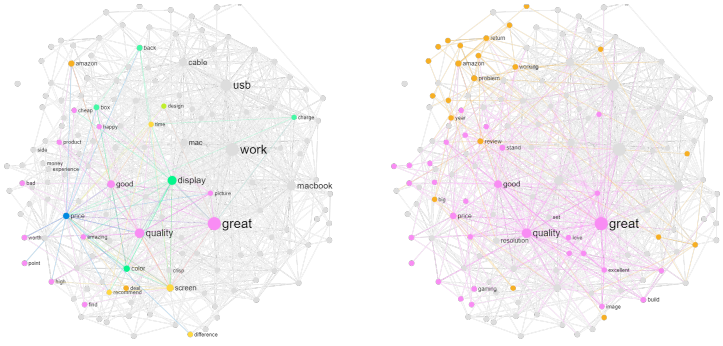
Zajímavou funkcionalitou je *porovnání grafů*. Grafy lze porovnat v levé liště pro rychlý přístup. Na výběr jsou celkem tři možnosti: průnik (*intersection*), rozdíl (*difference*) nebo kombinace obojího (*combination*). Výběr možnosti zobrazí novou stránku s grafem, na který bylo aplikováno porovnání. V podstatě se aplikuje vybraná operace na množinu vrcholů a hran, a poté se zobrazí jako nový graf. Nový graf lze analyzovat identicky jako ty předchozí, ze kterých vznikl. Tato funkcionalita může být užitečná při hledání změn např. v čase.

3.2.4.2 Statistiky

Aplikace nabízí pohled na různé statistiky grafu včetně analýzy sentimentu textu, trendů, numerických statistik a tématických shluků. Statistiky lze vidět v pravé části na panelu.

Analýza *sentimentu textu* je komplexním procesem, který výpočetně identifikuje a zařazuje názory vyjádřené pomocí textu. Účelem je výpočetně určit, s jakým postojem byl text autorem napsán [42]. InfraNodus analyzuje sentiment do tří kategorií - zda se slova přibližují spíše pozitivní, negativní nebo neutrální povaze. Kategorie lze filtrovat a tím zobrazit jen daný podgraf s daným sentimentem. Tato analýza je užitečná především v recenzích zákazníků nebo v jakémkoliv systému, kde je velké množství textových dat od uživatelů (komentáře, příspěvky na sociálních sítích apod.).

Tématické shluky jsou vytvořeny pomocí detekce komunit (viz. sekce 2.1.3). Kromě vizualizace komunit v grafu je dostupný panel s detailnějších popisem komunit (viz. obrázek 3.3). Každá komunita obsahuje vlastní řádek v seznamu, která obsahuje procentuální obsaženost v grafu a hlavní vlivný vrcholy. Při kliknutí na komunitu v seznamu se barevně zvýrazní také v grafu. Na obrázku 3.4 lze vidět dvě vybrané komunity.



(a) Vybrané slovo „price“ (b) Vybrané komunity

Obrázek 3.4 InfraNodus: Vizualizace síťového grafu

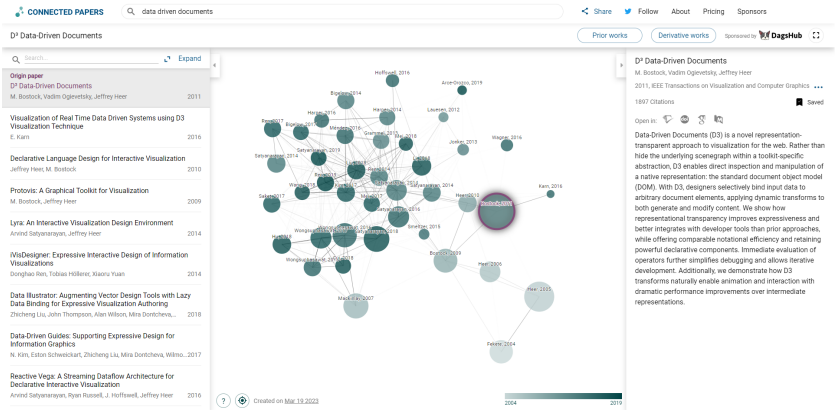
3.3 Connected Papers

Jak již bylo zmíněno, v reálném světě jsou souvislosti víceméně v každé doméně. I při psaní akademických publikací je zvykem, že autoři těchto publikací čerpají z různých zdrojů a následně je citují ve své bibliografii. Ve většině případů i citované publikace mají vlastní citované zdroje a tímto způsobem se řetězí dál. Avšak dohledání těchto zdrojů může být problematické, neboť publikované práce jsou roztroušeny po několika doménách jak v online nebo kamenných knihovnách. Zároveň autoři hledají publikace, které jsou nějakým způsobem podobné nebo obsahují podobnou tématiku.

Proto byl vytvořen nástroj *Connected Papers* [43]. Jedná se o unikátní nástroj, který dokáže vizualizovat nejen citované publikace na základě jejich tématické podobnosti. Pomáhá tím akademickým pracovníkům lépe prozkoumat publikace, které jsou pro ně relevantní. K realizaci využívá síťový graf, který vzájemné souvislosti a citace mezi publikacemi vizualizuje. Graf obsahuje nejen přímé citace, ale i ty, které jsou od publikace vzdálenější, ale stále tématicky relevantní.

3.3.1 Hledání publikací

Aplikace obsahuje databázi publikací, ve které je nejdříve potřeba nalézt danou publikaci. Po vyhledání publikace buď pod názvem, jménem autora nebo identifikací je zobrazena hlavní část s grafem citací. Pro tuto rešerši byla použita publikace *D^3 Data-Driven Documents* [22], která představila vizualizační knihovnu D3.js (více o knihovně v sekci 4.2). Vygenerovaný graf lze vidět na obrázku 3.5.



Obrázek 3.5 Connected Papers: Rozhraní grafu publikace

3.3.2 Rozhraní grafu

Síťový graf obsahuje vrcholy spojené neorientovanými hranami. Vrcholy reprezentují jednotlivé publikace a hrany propojení mezi nimi. Jak již bylo řečeno výše, jedná se i o vzdálenější souvislosti než jen přímé citace. Vrcholy publikací jsou zabarvené neobvyklým způsobem, a to gradientem podle roku vydání publikace. Čím výraznější barva, tím je publikace novější (relevantní k dnešnímu datu). Starší publikace jsou tím méně výraznější a spíše zašedlé. Velikost vrcholu udává počet citací publikace. Podobné publikace mají výrazné hrany a jsou umístěny do shluku, zatím co méně souvislé publikace naopak. Při kliknutí na citovanou publikaci je zobrazena nejkratší cesta k ní. Všechny publikace zobrazené v grafu jsou navíc zobrazené jako seznam po levé straně obrazovky (lze vidět na obrázku 3.5).

3.3.3 Detail publikace

Při kliknutí na vrchol se po pravé straně obrazovky zobrazí panel s detailem publikace. Kromě jejího názvu s autory obsahuje i další užitečné informace o publikaci - počet citací publikace, rok a doména vydání publikace, seznam externích odkazů na publikaci a samostatný abstrakt publikace.

Seznam odkazů do externích systémů odkazuje na systémy zaměřené na publikaci a vyhledávání akademických publikací (např. Google Scholar). Je především důležitou funkcionalitou, neboť není třeba hledat publikaci v grafu znovu v těchto systémech. Jsou rovnou propojeny přes přímý odkaz (lze vidět na obrázku 3.5).

Kapitola 4
Vizualizační knihovny

Tato kapitola se zabývá rešerší vizualizačních knihoven pro web front-end aplikace, tedy knihovny na bázi jazyka JavaScript. Knihovny jsou analyzovány především z hlediska technologií, použitelnosti a jejich funkcionality. Analyzovány jsou také metody vykreslování pro web front-end aplikace, které dané knihovny používají.

JavaScript (nebo také jako JS) je progamovací jazyk, který se používá po boku HTML (*HyperText Markup Language*) a CSS (*Cascade Style Sheets*). JavaScript se nejčastěji používá pro klientskou část aplikace (*front-end*), ale v některých případech i pro serverovou část (*back-end*). Tato trojice tvoří základy tzv. *World Wide Web* technologie, neboli technologie webových stránek.

HTML je základem pro zobrazení dokumentů ve webových prohlížečích. Představuje „kostru" toho, co je na stránce zobrazeno. Využívá při tom značky (*tagy*) pro definování elementů. O vzhled a grafické elementy stránky se stará CSS. Pro umožnění interaktivity a dynamiky uživatelského rozhraní je právě již zmíněný jazyk JavaScript.

Od vzniku jazyka JavaScript vzniklo také několik knihoven stavějící na bázi tohoto jazyka. Proto většina z nich nese v názvu zkratu *.js*. V dnešní době je čistý JavaScript jen lehce používán, jeho místo nahradily odvozené knihovny a frameworky.

Za zmínku stojí koncept DOM (*Document Object Model*). Jedná se o objektovou reprezentaci HTML dokumentu, která reprezentuje rozhraní mezi JS a HTML dokumentem. Účelem DOM je reflektovat hierarchickou strukturu HTML dokumentu a umožnit tím jazyku JavaScript vytvořit dynamické stránky [44]. S tím souvisí i funkcionalita JS knihoven, které mohou princip DOM využívat ve svém API (*application programming interface*).

Právě několika vybraným knihovnám se bude věnovat tato kapitola. Při rešerši a výběru vhodné knihovny pro praktickou část práce je důležité nahlížet několika faktorům, které eventuálně pomohou při výběru vhodné knihovny pro praktickou část práce:

- princip a technologie
- funkcionality
- možnosti přizpůsobení
- dokumentace
- podpora a relevantnost

4.1 Vykreslovací metody

Vizualizační knihovny mohou využívat pro realizaci svých vizualizací několika vykreslovacích metod pro web front-end aplikace. Tyto metody umožňují vykreslování komplexních elementů ve webovém prohlížeči. Mezi primární metody patří:

- SVG
- HTML5 Canvas
- WebGL

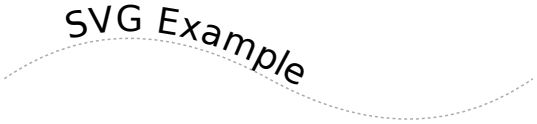
4.1.1 SVG

SVG (*Scalable Vector Graphics*) je formát založený na struktuře XML (*Extensible Markup Language*) používaný pro tvorbu 2D vektorové grafiky [45]. SVG formát je bezeztrátový a založen na matematické reprezentaci tvarů. Tedy na rozdíl od ztrátových formátů (např. JPEG) neztrácí kvalitu při jejím škálování. Tento formát je ideální pro tvorbu grafiky jako jsou loga nebo ikony, které se skládají z jednoduchých tvarů a potřebují být škálovatelné nebo interaktivní.

Přínos HTML5 je integrovaná podpora právě pro SVG formát. Například následující kód (viz. kód 4.1) vygeneruje příslušné SVG (viz. obrázek 4.1). Důležité použité značky jsou `<svg>`, `<path>`, `<text>` a `<textPath>`. Tato metoda je využívána např. v knihovně D3.js (sekce 4.2) nebo G6.js (sekce 4.3).

Výpis kódu 4.1 Ukázka `<svg>` elementu

```
<svg
  viewBox="0␣0␣400␣150"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <path
    id="myPath"
    d="M20,75␣Q100,20␣200,75␣T380,75"
    stroke="#aaaaaa"
    stroke-dasharray="2"
    fill="none"/>
  <text
    font-size="24"
    font-weight="400"
    fill="black"
    dy="-5px"
    dx="50">
    <textPath xlink:href="#myPath">SVG Example</textPath>
  </text>
</svg>
```



Obrázek 4.1 Ukázka SVG vizualizace reprezentovanou kódem 4.1.

4.1.2 HTML5 Canvas

Další metodou je využití elementu *HTML5 Canvas*. Jedná se to 2D vykreslovací API, které umožňuje dynamické vykreslování *bitmap* ve webových aplikacích. V HTML dokumentu je deklarováno značkou `<canvas>`. Interaktivita vizualizace je umožněna především přes JavaScript, a to za pomocí již zmíněného DOM. Za pomocí JS skriptu lze přes DOM vybrat element `<canvas>` a následně do něj vykreslit grafické elementy (lze vidět v ukázce kódu 4.2).

Avšak na rozdíl od SVG formátu, HTML5 Canvas není bezeztrátovou grafikou. Naopak je rychlý a efektivní, tedy vhodný pro komplexní animace a neustálé překreslování. Příklad lze vidět v kódu 4.2, kde elementy `<canvas>` a `<script>` se nachází v `<body>` elementu HTML dokumentu. Tato metoda je využívána např. v knihovně D3.js (sekce 4.2) nebo vis.js (sekce 4.4).

Výpis kódu 4.2 Ukázka `<canvas>` elementu

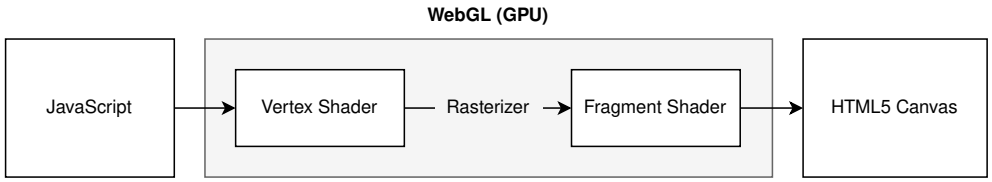
```
<canvas id="myCanvas" width="400" height="150"/>
<script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");

    ctx.beginPath();
    ...
</script>
```

4.1.3 WebGL

WebGL (*Web Graphics Library*) je JavaScript API, které umožňuje vykreslování jak 2D tak 3D grafiky ve webovém prohlížeči [46]. WebGL používá GPU (*graphics processing unit*, v překladu grafická karta) ke zpracování komplexních výpočtů, čímž dělá vykreslování rychlejším a efektivnějším oproti zmíněným variantám výše. Tato metoda funguje na bázi shaderů. Veškeré tvary a obrazce vykresluje pomocí trojúhelníků, jejichž pozice se vypočítají ve *vertex shaderu* a následně se každý pixel obarví ve *fragment shaderu*. Shadery se píšou v jazyce GLSL (*OpenGL Shading Language*). Práce se samostatnými daty trojúhelníků funguje na úrovni bufferů.

Jak už nejspíše zní z popisu, tato metoda se značně liší od předchozích dvou metod. Je to kvůli faktu, že WebGL je založeno na OpenGL a dodržuje jeho principy pro využití GPU. Zároveň nabízí možnost 3D vizualizace, které jednoduché SVG neumí. Rozdílem mezi OpenGL a WebGL je zavedení WebGL do webového prostředí pomocí `<canvas>`. JavaScript zpracuje inicializaci a zpracování dat, následně jsou data transformována oběma shadery a nakonec vykreslena do `<canvas>` (lze vidět na obrázku 4.2). Tato metoda je využívána např. knihovnou Sigma.js (sekce 4.5).



Obrázek 4.2 Proces zpracování pomocí metody WebGL. Překresleno podle [47].

4.2 D3.js

Zkratka D3 znamená *data-driven documents* a dává název knihovně D3.js. Tvůrce JavaScript knihovny Mike Bostock vytvořil D3.js za účelem „využití moderních web standardů“ jako jsou HTML5, CSS3 a SVG [22]. Tato knihovna nabízí všechny stavební bloky potřebné k vytvoření tradičních, ale i komplexních vizualizací a to od jednoduchých spojnicových grafů až po geografické mapy. Do dnešní doby je stále široce využívanou knihovnou obzvlášť pro její možnost přizpůsobení a vytvoření prakticky čehokoliv, co vývojář navrhne.

4.2.1 Princip a technologie

D3.js nabízí vykreslování pomocí SVG i HTML5 Canvas (viz. výše). Tradiční způsob obsahuje tvorbu vizualizací pomocí řetězení metod v JavaScriptu, kde základem je tzv. *selecting and binding* dat. Selecting znamená *výběr* a binding znamená *vazba*. D3 umožňuje vybírání elementů HTML přes jeho DOM (již vysvětleno v úvodu kapitoly), a následně k nim naváže data. Navázání dat umožňuje všem vnitřním elementům přistoupit k nim za pomocí D3 metod. To může být užitečné při tvorbě a přidávání grafických elementů [23].

Příklad typického D3 kódu lze vidět v kódu 4.3, kde je použito vykreslování pomocí SVG (viz. sekce 4.1.1). Nejdříve se vybere `<svg>` element z DOM pomocí `d3.select()` a poté se do něj přidá element `<circle>` pomocí `d3.append()`. Následně se upraví jeho styly (podobně jako CSS) pomocí metody `style()` a nakonec atribut `r` elementu `<circle>` je nastaven na 20.

D3 kód lze napsat i jiným způsobem než zřetězováním metod, např. s využitím moderních web front-end knihoven jako je React aj. V takovém případě lze uvažovat o použití D3 jako vizualizační kernel pro aplikaci, zatímco vytváření a aktualizaci elementů provádět pomocí zmíněných knihoven (o integraci s React knihovnou více v sekci 7.4).

Výpis kódu 4.3 Ukázka D3.js kódu

```
var svg = d3.select('svg')

svg
  .append('circle')
  .style('fill', 'orange')
  .attr('r', 20)
```

4.2.2 Funkcionality

Hlavní sílou této knihovny je princip „stavebních bloků“. Knihovna nabízí všechny potřebné prostředky k vytvoření všech možných grafů, ale postup je delší než zavolání jedné funkce pro jeho vytvoření. Obsaženy jsou funkcionality pro síťové grafy i hierarchické rozložení. Mezi základní funkcionality knihovny patří:

- force-directed a radiální rozložení
- dendrogram
- circle packing a treemap
- geografické mapy
- škálovací funkce
- animace a přechody
- výpočet křivek, přímek a jiných tvarů

D3.js je tedy vhodným kandidátem pro přizpůsobené vizualizace. Zmíněné funkcionality obsahují bohaté API, které lze přepoužít prakticky kdekoliv. Kvůli tomu může být učící křivka s knihovnou velmi strmá (tzv. *steep learning curve*) a záleží na případu užití zda ji použít. Knihovna nativně podporuje formáty CSV a JSON.

4.2.3 Podpora a dokumentace

V době psaní této práce je poslední verzí D3.js verze *7.8.3*, která vyšla v březnu roku 2023. Knihovna je tedy aktivní s precizní dokumentací, která je dostupná i v několika jazycích včetně původní angličtiny. Zároveň díky vybudované popularitě lze říci, že knihovna má i značnou podporu komunity v rámci tvorby užitečných návodů a tipů. Dokonce byly vydány i navazující knihovny pro různé účely (např. *d3-milestones* nebo *react-d3-library* aj.) nebo oficiální návody od samotného tvůrce D3.js knihovny.

4.3 G6.js

G6.js je JavaScript knihovna navržená pro zjednodušení komplexních vizualizací jako jsou síťové grafy nebo jiné typy grafů [48]. Svoji simplicitu nabízí pomocí modulů, kde veškerá konfigurace grafů je umožněna přes atributy modulu.

4.3.1 Princip a technologie

G6.js primárně nabízí vykreslování pomocí HTML5 Canvas. Nabízí i možnost SVG vykreslování, ale tato varianta je momentálně omezená. Vizualizace se tvoří přímo v JS kódu, kde hlavním principem je vytvoření nového grafu přes `G6` API (např. `G6.Graph()`, `G6.TreeGraph()`). Přes stejné API lze také vytvářet pomocné UI komponenty, které se typicky využívají v rozhraní vizualizací (např. `G6.ToolBar()`, `G6.TimeBar()`).

Příklad jednoduchého síťového grafu lze vidět na ukázce v kódu 4.4. Nový objekt grafu je vytvořen pomocí `G6.Graph()` metody, které je předán objekt s nastavením grafu. Atribut `container` představuje HTML5 Canvas, které je získáno z DOM. Jak lze vidět, nastavení vzhledu grafu je provedeno přímo v JS kódu, kde jsou nastaveny atributy např. `defaultNode`. Po konfiguraci grafu je na graf zavolána metoda `data()` pro předání dat a nakonec metoda `render()` pro jeho vykreslení.

Výpis kódu 4.4 Ukázka G6.js kódu

```
const graph = new G6.Graph({
  container: "container",
  width: 500,
  height: 500,
  plugins: [tooltip],
  modes: {
    default: ['drag-canvas', 'activate-relations']
  },
  defaultNode: {
    shape: "circle",
    style: {
      fill: "#9EC9FF"
    },
  }
});
graph.data(data);
graph.render();
```

4.3.2 Funkcionality

G6.js se zaměřuje především na komplexní vizualizace jako jsou síťové grafy a všechny elementy jich se týkající. Nechybí ale ani vizualizace jednodušších grafů jako jsou např. pie graf, radar graf, spojnicový graf atd. Celkově nabízí široký výběr možností pro grafy jako jsou:

- force-directed, radiální a mřížkové rozložení
- dendrogram a jiné stromové struktury
- shlukování vrcholů
- flow chart
- entity relationship (ER) diagram
- časové osy vývoje grafů
- výpočet křivek, přímek a jiných tvarů
- animace a přechody
- legenda grafu

Knihovna obsahuje bohaté API pro vývoj nejen síťových grafů a i značné množství ukázek jeho využití. Do určité míry jsou vizualizace přizpůsobitelné pomocí objektového rozhraní při definování nastavení grafu (jak bylo ukázáno výše). Navíc lze pomocí API vytvořit různé komponenty, které mohou být po bohu samostatné vizualizace (popis, časová osa, legenda, panel pro manipulaci přiblížení atd.).

4.3.3 Podpora a dokumentace

V době psaní této práce je poslední verzí G6.js verze *4.8.8*, která vyšla v březnu roku 2023. Knihovna se tedy zdá být aktivní. První verze vyšla okolo roku 2020, avšak do této doby je dostupná jen část dokumentace v anglickém jazyce. Zároveň kvůli relativně krátkému času dostupnosti knihovny nejsou k dispozici anglické návody pro použití, které by vytvořila komunita. Tato knihovna se totiž (zatím) do mezinárodní komunity nerozšířila. Kvůli těmto faktorům není G6.js momentálně na pozici, která by zajišťovala spolehlivý vývoj.

4.4 vis.js

Vis.js je JavaScript knihovna navržená pro jednoduché použití a vykreslení interaktivních dat [49]. Obsahuje celkem čtyři komponenty - DataSet, Timeline, Network, Graph2D a Graph3D. V rámci této anylýzy je relevantní část *Network* pro vizualizaci síťových grafů.

4.4.1 Princip a technologie

Vis.js Network používá HTML5 Canvas pro vykreslení vizualizace. Deklarace grafu, která je založená na modulech (podobně jako v G6.js viz. 4.3) probíhá v JS skriptu. Stačí získat `<canvas>` z DOM, připravit data a objekt konfigurace `options` pro graf. Následně vše vložit jako parametry metody `vis.Network()`.

4.4.2 Funkcionality

Objekt `options` obsahuje konfiguraci pro graf. Zde je nastavitelné jakékoliv vlastní přizpůsobení vzhledu a interakce grafu. Celkem obsahuje asi osm modulů, kde každý má na starost vlastní typ funkcionality. Jedná se moduly `edges`, `nodes`, `groups`, `layout`, `interaction`, `manipulation` a `physics`. Moduly nabízí funkcionality včetně:

- vzhled vrcholů a hran
- force-directed rozložení
- hierarchie
- seskupení vrcholů
- přiblížení/oddálení
- přidání/smazání vrcholů a hran

V zásadě se jedná o základní a typické interakce se síťovým grafem. Knihovna nenabízí značné množství vlastního přizpůsobení a nemá pro síťový graf velké API.

4.4.3 Podpora a dokumentace

V době psaní této práce je poslední verzí vis.js verze *9.1.6*, která vyšla v březnu roku 2023. Knihovna je aktivní a obsahuje i dokumentaci v angličtině. Avšak dokumentace není dobře strukturovaná a neobsahuje ani praktické ukázky použití. Knihovna není mezi komunitou populární, tudíž neobsahuje ani přínos od ní. Existují nadstavby pro knihovnu jako např. *vis-react* pro React nebo *vue-vis-network* pro vue.js, ale nejsou aktivní ani udržované.

4.5 Sigma.js

Sigma.js je JavaScript knihovna s účelem vizualizovat grafy s tisíce vrcholy ve webovém prohlížeči [50]. Zaměřuje se výhradně na síťové grafy (popř. hierarchie).

4.5.1 Princip a technologie

Knihovna byla navržená pro spolupráci s knihovnou *Graphology*, která nabízí různé algoritmy pro síťové grafy [51]. Data síťového grafu jsou zpracována touto knihovnou a následně vykresleny pomocí Sigma.js. Knihovna využívá WebGL pro vykreslení na HTML5 Canvas. Díky této konstrukci je možné vykreslit grafy s tisíce vrcholy, jak sama se knihovna prezentuje. Práce s knihovnami probíhá především v JS skriptech.

4.5.2 Funkcionality

Sigma.js se zaměřuje výhradně na síťové a popřípadě hierarchické grafy. Veškeré funkcionality pro zpracování dat grafu zpracovává knihovna Graphology a vykreslování Sigma.js. Vzhledem k tomu, že Sigma.js využívá WebGL, tak jsou možnosti přizpůsobení možné, ale relativně složité. Vývojář totiž musí mít navíc znalost jazyka GLSL pro vytváření shaderů (viz. WebGL v sekci 4.1.3), což přidává komplexitě na práci s knihovnou. Tvorba a přizpůsobení grafických elementů v GLSL je značně složitější než SVG nebo HTML5 Canvas. Knihovna Graphology obsahuje funkcionality včetně:

- orientované i neorientované grafy
- force-directed a radiální rozložení
- hierarchické rozložení
- detekce komunit
- hledání vrcholů a hran
- hledání sousedů
- hledání nejkratší cesty mezi vrcholy aj.

Zajímavou funkcionalitou knihovny Sigma.js je podpora GEXF (*Graph Exchange XML Format*) formátu. Jedná se o speciální formát, který popisuje komplexní síťové grafy. Příklad formátu lze vidět v kódu 4.5. Jedná se o jednoduchý orientovaný statický graf obsahující dva vrcholy a jednu hranu mezi nimi. Dále jsou podporovány formáty JSON a CSV.

Výpis kódu 4.5 Ukázka GEXF formátu

```
<gexf xmlns="http://gexf.net/1.2" version="1.2">
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
        </edges>
    </graph>
</gexf>
```

4.5.3 Podpora a dokumentace

V době psaní této práce je poslední verzí Sigma.js verze *3.0.0-alpha3*, která vyšla v březnu roku 2023. Knihovna je aktivní s dokumentací v angličtině. Knihovna není v komunitě rozšířená v takové míře jako je D3.js, ale oficiální dokumentace obsahuje několik příkladů pro její využití i s příklady pro GLSL. Sigma.js obsahuje i nadstavby pro integraci s moderními web front-end knihovnami jako je React (konkrétně *react-sigma*).

Kapitola 5

Shrnutí teoretické části

V této kapitole jsou shrnuty klíčové poznatky z teoretické části. Tedy z kapitol o sylabu, vizualizaci, vizualizačních aplikacích a vizualizačních knihovnách. V závěru je jsou porovnány vizualizační knihovny pro výběr k implementaci prototypu.

5.1 Sylabus

V kapitole o sylabu (viz. kapitola 1) bylo zjištěno, že studium má tři úrovně - bakalářské, magisterské a doktorské. Každá úroveň obsahuje vlastní akreditace ve formě studijního programu. Studijní program lze kategorizovat podle jazyka jeho výuky - aktuálně čeština nebo angličtina.

Každá akreditace má vlastní platnost. Během doby platnosti je akreditace aktivní, po ukončení se stává neaktivní. Platnost akreditací dvou entit, které se reprezentují ekvivalentně se může časově překrývat (např. bakalářský program staré a nové akreditace).

Pomocí MŠMT bylo zjištěno, že akreditovány jsou studijní programy nebo také specializace/obory programu. Nový pojem „specializace“ je obdobou starého pojmu „obor se zaměřením“. Každý studijní program má několik specializací. Specializace lze kategorizovat podle kateder.

Každá specializace vypisuje studijní plány, do kterých se studenti hlásí. Studijní plán je buď ve prezenční nebo kombinované formě. Speciálním případem je fáze studia bez studentova zaměření, který je výchoze zapsán každému novému nastupujícímu studentovi.

Každý studijní plán obsahuje skupiny předmětů, kde student musí splnit určité podmínky (minimální počet kreditů a předmětů). Skupina předmětů je charakterizována vlastní rolí. Předměty ve skupině reprezentují danou roli. Stejný předmět může v jiném studijní plánu hrát jinou roli.

Předměty obsahují několik klíčových atributů (typ zakončení, jazyk, kredity atd.). Atributy obsahující osnovu přednášek a cvičení obsahují rámcové témata předmětu. Atribut klíčových slov je zanedbaný a nekonzistentní. Verze předmětu lze rozdělit na verze v akreditaci a verze přes akreditace (nebo také pohyb v čase přes semestry). Některé propojení verzí přes akreditace jsou definované v novém systému KOS. Tabulka obsahující pravidla pro vzájemné uznávání předmětů mezi akreditacemi definuje ekvivalentní nebo jednostranný směr uznatelnosti předmětů. V této oblasti lze říci, že některé předměty byly rozštěpeny na více předmětů a některé byly naopak spojeny s příchodem nové akreditace.

Ve stejné akreditaci má předmět dva typy verzí. Jeden typ je podle formy studia - kombinovaná a prezenční forma. Druhým typem je jazyk výuky - česká a anglická verze předmětu. Předmět patří do dané katedry.

Vyučující je zaměstnancem dané katedry, ale navíc může hrát roli jako garant, přednášející nebo cvičící v rámci předmětu. Dále může hrát roli jako garant studijního plánu.

Ve studiu lze identifikovat hierarchický vztah pro entity (postupně):

- studijní program
- specializace/obor
- studijní plán
- skupina předmětů
- předmět

5.2 Vizualizace

V kapitole o vizualizacích (viz. kapitola 2) byly analyzovány především síťové vizualizace a hierarchické vizualizace. Obě možnosti jsou založené na konceptu grafu.

Síťové grafy zobrazují komplexní domény s několika souvislostmi. Důležité je rozložení grafu, kde optimálním a zároveň nejvíce uživatelsky přívětivým je force-directed rozložení. Doména grafu může obsahovat jeden typ vrcholů (např. slova při textové analýze viz. kapitola 3) nebo také více typů entit jako ve znalostním grafu (viz. sekce 2.1.4).

Jednoduchá hierarchická vizualizace může být reprezentována dendrogramem (viz. sekce 2.2.1) ve formě stromu. Další jsou možnosti, které jsou úspornější ve formě map. Jedná se o circle packing (kruhové zanoření) nebo treemap (stromová mapa).

Některé aplikace s vizualizací nabízejí určitou míru personifikace, např. ve formě ukládání vlastních grafů a jeho nastavení, nebo možnost nahrát text pro textovou a grafovou analýzu (viz. sekce 3.2). Dále aplikují různé algoritmy včetně detekce komunit. Typické interakce obsahují pohybování s plátnem a možnost zobrazit detaily entity (vrcholu) po jeho výběru kliknutím. Přidané informace, které ve vizualizaci nelze jednoduše zobrazit, jsou zobrazeny právě na postranních panelech v uživatelském rozhraní aplikace. Většinou obsahují základní orientační informace, přesměrování na další zobrazení nebo třeba i odkazy na relevantní externí systémy.

5.3 Vizualizační knihovny

V kapitole s rešerší vizualizačních knihoven (viz. kapitola 4) byly rozebrány tři možnosti vykreslování ve webových aplikacích - SVG, HTML5 Canvas a WebGL. SVG je nejkvalitnější variantou, neboť představuje bezeztrátový formát, ale může se potýkat s výkonnostními problémy. HTML5 Canvas a WebGL jsou bitmapové možnosti (tedy ztrátové), ale naopak jsou velmi výkonné a WebGL dokonce nejoptimálnější, kvůli zpracování pomocí GPU.

Ve stejné kapitole byly analyzovány vybrané knihovny, které jsou potenciální kandidáti pro implementaci prototypu. Knihovny mají své silné i slabé stránky v rámci několika faktorů. Pro výběr je důležité vzít v potaz vykreslování, dokumentaci, funkcionality a možnosti přizpůsobení. V následující tabulce 5.1 lze vidět porovnání na těchto faktorech (+ značí výhodu, - značí nevýhodu, ∼ značí průměr). Pro implementaci prototypu je nejvhodnější D3.js obzvlášť kvůli spolehlivé dokumentaci a přizpůsobení, které zároveň není složité jako u WebGL.

Tabulka 5.1 Porovnání vizualizačních knihoven pro web front-end aplikace

Knihovna	Vykreslování	Dokumentace	Funkcionality	Přizpůsobení
D3.js	Canvas, SVG	+	+	+
G6.js	Canvas, SVG	-	+	+
vis.js	Canvas	-	∼	-
Sigma.js	WebGL	+	+	∼

Část II

Praktická část

Kapitola 6
Návrh

Tato kapitola se zabývá návrhem vizualizací pro prototyp. Nejdříve jsou definovány požadavky, které by aplikace měla splňovat. Pomocí předchozí analýzy je vytvořena konceptualizace problémové domény. Dále je popsána aplikace a tím definovány případy užití a návrh uživatelského rozhraní s vizualizací. V závěru jsou vytvořeny komponentové modely pro implementaci vizualizací.

6.1 Požadavky

Úvodem jsou definovány požadavky pro návrh a vývoj aplikace. Funkční požadavky obsahují požadavky především ohledně vlastností vizualizací a co by měly splňovat. Nefunkční požadavky obsahují naopak vlastnosti aplikace jako jsou technologie, knihovny, programovací jazyky aj.

6.1.1 Funkční požadavky

F1: Vizualizace obsahují interaktivitu
Jako jeden z požadavků je interaktivnost vizualizací. Vizualizace nebudou staticky vykreslené, ale budou interaktivní na základě uživatelského vstupu. Je důležité, aby uživatel mohl vizualizace nejen pozorovat ale i prozkoumávat, popřípadě mít možnost měnit jejich vzhled a efektivně v nich vyhledat potřebné informace.

Priorita Vysoká

Kategorie Funkčnost

Náročnost Střední

F2: Vizualizace zobrazuje přehled z hlediska specializací/oborů
V předchozích kapitolách byly analyzovány entity sylabu, kde z jedna z entit byla specializace, nebo také ekvivalentně obor. Aplikace bude obsahovat formu vizualizace, která bude zobrazovat přehled akreditovaných specializací/oborů v semestru.

Priorita Vysoká

Kategorie Funkčnost

Náročnost Střední

F3: Vizualizace zobrazuje návaznost předmětů v čase
V předchozích kapitolách byly analyzovány předměty a jejich verze. Verze předmětu v rámci času jsou jednotlivé běhy předmětu v semestru, ale také i verze přes akreditace. Momentální řešení nenabízí jasné propojenosti verzí předmětů v čase. Aplikace bude obsahovat formu vizualizace, která bude zobrazovat návaznost verzí předmětu přes semestry, ale i různé akreditace.

Priorita Vysoká

Kategorie Funkčnost

Náročnost Střední

F4: Vizualizace zobrazuje obsahovou (tématickou) návaznost předmětů
V předchozích kapitolách byly analyzovány předměty a jejich obsaženost. Tématická obsaženost předmětů je momentálně umožněna přes klíčová slova nebo osnovu přednášek a cvičení. Aplikace bude obsahovat formu vizualizace, která bude zobrazovat obsahovou a tématickou návaznost předmětů v semestru.

Priorita Vysoká

Kategorie Funkčnost

Náročnost Střední

6.1.2 Nefunkční požadavky

NF1: Webová aplikace
Aplikace bude implementována ve formě webové aplikace. Architektura aplikace bude vybrána v pozdější kapitole v implementaci prototypu.

Priorita Vysoká

Kategorie Implementace

Náročnost Nízká

NF2: Implementace vizualizací pomocí D3.js
V předchozí kapitole (viz. kapitola 5) byla shrnuta rešerše vizualizačních knihoven a na základě několika faktorů vybrána knihovna pro implementaci prototypu vizualizací. Aplikace bude implementována primárně pomocí knihovny D3.js.

Priorita Střední

Kategorie Implementace

Náročnost Vysoká

NF3: Získání dat z REST API
Data pro tvorbu vizualizací budou získána pomocí REST API. V pozdějších iteracích tohoto projektu bude aplikace využívat REST API přímo z back-end části aplikace. První verzi back-end části začal vyvíjet Tochilin Illia ve své bakalářské práci ve stejném akademickém roce jako byla psána tato práce.

Priorita Vysoká

Kategorie Podporovatelnost

Náročnost Nízká

NF4: Rozhraní v českém jazyce
Aplikace bude implementovat uživatelské rozhraní v českém jazyce včetně implementovaných vizualizací.

Priorita Nízká

Kategorie Vhodnost k použití

Náročnost Nízká

6.2 Konceptuální model domény

Na základě předchozí analýzy sylabu na FIT ČVUT (viz. kapitola 1) byla vytvořena konceptualizace problémové domény. Vytvořený model je konceptuálním modelem domény se základními a klíčovými entitami, které jsou v problémové doméně relevantní v době psaní této práce. Diagram modelu (viz. 6.1) se řídí podle UML zásad doménového modelu a obsahuje hlavní třídy ve formě „konceptů". Lze jej zařadit do kategorie tzv. *class diagramů*.

Nové (a jiné) konceptualizace studia analyzuje Martina Chomyšinová ve své bakalářské práci ve stejném akademickém roce jako byla psána tato práce. Na konceptualizaci a návrhu pro sledování změn v sylabech předmětů pracuje Samuel Händl ve své bakalářské práci ve stejném akademickém roce jako byla psána tato práce. Konceptualizaci sylabu také řeší Tochilin Illia, který ve své bakalářské práci pracuje na back-endové části aplikace.

V této části budou popsány třídy (entity), které mají významnou roli v návrhové části práce. Detailní popis a analýza entit je zmíněna v analytické části o sylabu (viz. kapitola 1). Diagram modelu neobsahuje atributy entit, neboť hlavním zaměřením jsou relace a vztahy mezi entitami. Některé klíčové atributy jsou zmíněny níže v popisu. Model nebere v potaz klasifikaci studentů, zápis studentů, výukové materiály nebo zaměstnance univerzity s jinou rolí než vyučující.

Akreditace *Akreditace* může být typem *specializace/obor* nebo *studijní program*. Definuje její typ získání (prodloužení, udělení akreditace aj.) dobu platnosti přes datum získání akreditace a datum platnosti akreditace. Také definuje jazyk výuky (typicky čeština nebo angličtina).

Studijní program *Studijní program* je nejvýše nadřazená entita sylabu fakulty (kromě jí samotné). Programy vytváří *fakulta*. Programy jsou definovány svým kódem (zkratkou), názvem a odkazem do externích systémů (BK).

Úroveň studia Každý program má *úroveň studia* (bakalářské, magisterské, doktorské). Úroveň studia může být také ekvivalentně definovaná jako typ programu.

Specializace/obor *Specializace* a *obor* představují ekvivalentní entity (od zavedení nové akreditace viz. 1.1.6). Studijní program obsahuje několik specializací, specializace náleží jednomu studijnímu programu. Specializace jsou definovány svým kódem (zkratkou), názvem a odkazem do externích systémů (BK).

Studijní plán *Studijní plán* má formu studia. Studijní plán je přiřazen k jedné specializaci, specializace obsahuje několik studijních plánů. Každý studijní plán může mít několik doporučených průchodů studiem, každý průchod náleží jednomu studijnímu plánu. Studijní plán obsahuje skupinu předmětů a má svého garanta. Studijní plány jsou definovány svým kódem (zkratkou), názvem a odkazem do externích systémů (BK).

Forma studia *Forma studia* může být buď *prezenční* nebo *kombinovaná*. Student je zapsán do studijního plánu s definovanou formou studia.

Role ve studijním plánu Předmět a skupina předmětů hrají *roli* ve studijním plánu. Seznam dostupných rolí viz. v analytické části (kapitola 1) nebo níže.

Předmět *Předmět* má výuku v semestru výuky (zimní, letní, nebo obojí). Obsahuje sadu klíčových slov a je zařazen do skupiny předmětů. Zároveň může být i doporučeným předmětem v doporučeném průchodu. Předmět může být zástupností jiných předmětů a naopak. Předmět je definován svým názvem, kódem (zkratkou), jazykem výuky, počtem kreditů, způsobem zakončení, rozsahem, osnovou přednášek a cvičení a svými odkazy do externích systémů (KOS, BK).

Běh předmětu *Běh předmětu* je instance předmětu v konkrétním semestru výuky. Běh předmětu má svého garanta, své paralelky a je zapisován studenty do jejich studia. Obsahuje odkazy do externích systémů (Fittable, Anketa ČVUT).

Osoba *Osoba* může být typem *student* nebo *vyučující*. Je definována svým jménem, příjmením, e-mailem a uživatelským jménem (*username*).

Vyučující *Vyučující* je typem *osoby*. Vyučující pracuje na katedře fakulty a může mít několik rolí vyučujícího (*přednášející*, *cvičící* a *garant*). Také může hrát roli garanta ve studijním plánu a skupině předmětů. Vyučující má své odkazy do externích systémů (Usermap, Fittable).

Garant *Garant* běhu předmětu je rolí vyučujícího. Běh předmětu má alespoň jednoho garanta, garant může být garantem více běhů předmětů.

Cvičící *Cvičící* paralelky předmětu je rolí vyučujícího. Paralelka má alespoň jednoho cvičícího, cvičící vyučuje ve více paralelkách běhů předmětů.

Přednášející *Přednášející* paralelky předmětu je rolí vyučujícího. Paralelka má alespoň jednoho přednášejícího, přednášející vyučuje ve více paralelkách běhů předmětů.

Katedra *Katedra* zajišťuje studijní plány a zaměstnává vyučující fakulty. Katedra je definována svým názvem, kódem (zkratkou) a obsahuje své odkazy do externích systémů (BK).

6.3 Případy užití

Modely případu užití (*use case*s) popisují a zachycují funkcionality systému (aplikace). Poskytují abstraktní náhled na systém především z pohledu uživatelů (*aktérů*), které tyto činnosti vykonávají. Diagramy dodržují zásady UML, kde (typicky) na levé straně vystupuje aktér a na pravé straně množina případů užití. Aktéři a případy užití jsou spojeny.

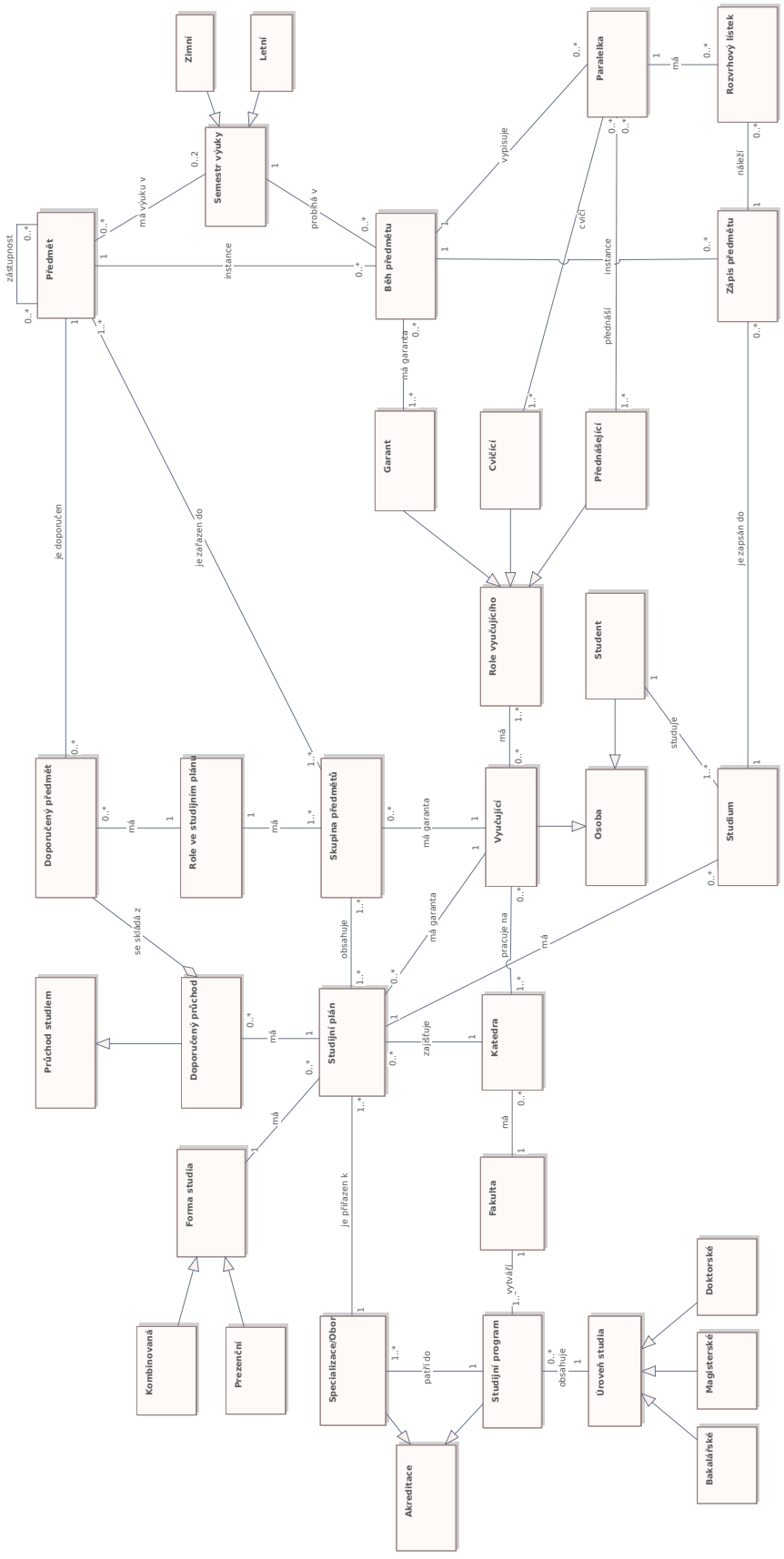
6.3.1 Aktéři

Aktérem případů užití může být osoba, skupin osob, externí systém nebo i čas. Tato práce se zaměřuje především na aktéry anonymních (nepřihlášených) uživatelů, neboť koncept personifikace a autentizace nebyl v době psaní této práce realizován ani konceptualizován.

Anonymní uživatel V této práci je kladen důraz především na *anonymního uživatele* aplikace, podobně jako je přístupná BK. Anonymní uživatel zahrnuje aktéry *studenta* a *vyučujícího*, které tvoří nejdůležitější cílovou skupinu aplikace.

Student V budoucích iteracích projektu je *student* potenciálním aktérem pro personifikaci. V této fázi projektu není personifikace pro studenta zahrnuta, avšak každý student může hrát roli anonymního uživatele a jeho případů užití.

Vyučující V budoucích iteracích projektu je *vyučující* potenciálním aktérem pro personifikaci a správních rolí systému. V této fázi projektu jsou zmíněny základní akce v této roli (jako přidávání a odebírání entit), avšak každý vyučující může hrát roli anonymního uživatele a jeho případů užití.



Obrázek 6.1 Konceptuální model: Problémová doména

6.3.2 Znalostní mapa

6.3.2.1 UC1: Zobrazení znalostní mapy

Znalostní mapa zobrazuje klíčové entity sylabu a jejich relace. Jedná se o entity *předmět*, *vyučující*, *téma*, *studijní plán* a *katedra*. Mapa lze zobrazit pro daný semestr.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází v systému.
Scénář:

1. Aktér vybere v postranní liště možnost domovské stránky pod tlačítkem „Domů“.
2. Systém zobrazí domovskou stránku (nástěnku).
3. Aktér vybere možnost znalostní mapy pod tlačítkem „Mapa“.
4. Systém zobrazí mapu v aktuálním semestru s entitami.

Alternativní scénář:

1. Pokračuje od bodu 2 standardního scénáře.
2. Aktér vybere semestr pro zobrazení mapy v semestru.
3. Systém zobrazí mapu ve vybraném semestru.

6.3.2.2 UC2: Zobrazení tématických shluků

Tématické shluky transformují mapu a vybrané entity do upravené podoby. Shluky obsahují množinu předmětů/témat, které mají mezi sebou silnou provázanost.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér zapne zobrazení statistik pomocí tlačítka „Zobrazit tématické shluky“.
2. Systém zobrazí seznam nalezených tématických shluků.
   a. Procentuální obsaženost shluku v mapě
   b. Hlavní vlivné téma/předmět
   c. Ostatní obsažená témata/předměty shluku
3. Systém zobrazí vizualizaci shluků na místech zájmu.

6.3.2.3 UC3: Filtrování mapy

Systém nabízí více možností filtrování entit v mapě na základě jejího typu nebo atribut.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere možnost filtrování:
   a. Filtrování typů entit v horní liště zobrazené mapy.
      i. Aktér vybere typ entity po zobrazení/skrytí pomocí tlačítka s názvem typu.
   b. Filtrování atributů entit v pravém rohu zobrazené mapy.
      i. Aktér vybere dané filtry.
      i. Aktér aplikuje filtry tlačítkem „Filtrovat“.

6.3.2.4 UC4: Hledání v mapě

Mapa umožňuje efektivní vyhledávání entit. Následně je vizualizace zaměřena na vybranou entitu.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere formulář pro hledání entit.
2. Systém zobrazí seznam dostupných entit roztříděných podle typů.
3. Aktér zadá jméno nebo zkratku vyžadované entity.
4. Systém zaměří mapu na entitu a zvýrazní jí.

6.3.2.5 UC5: Přidání entity/relace

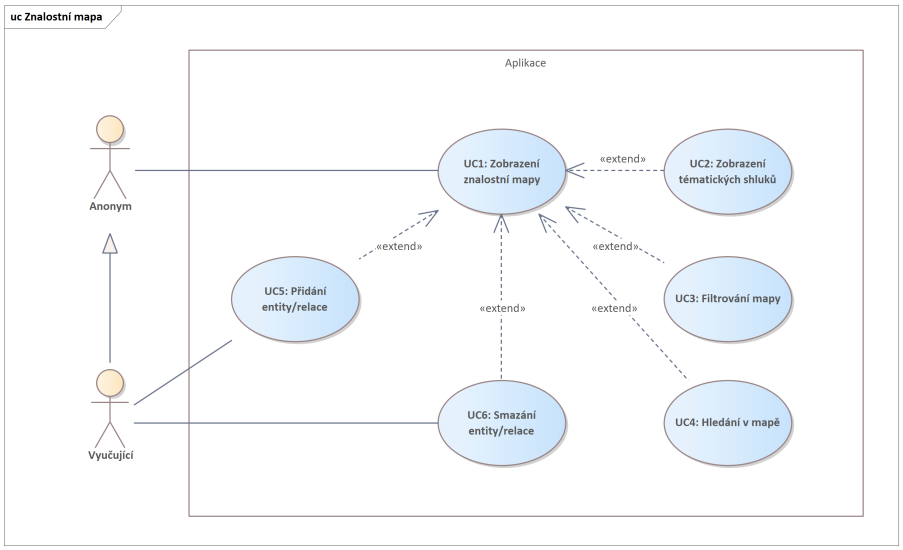
Vyučující má možnost upravit mapu pomocí přidání entity/relace.
Aktéři: Vyučující
Počáteční podmínka: Vyučující se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere nové přidání tlačítkem „Přidat“.
2. Systém zobrazí možnost přidání:
   a. Entity
   b. Relace
3. Aktér vybere danou možnost přidání pomocí tlačítka „Entita“ nebo „Relace“.
4. Systém zobrazí formulář pro přidání.
5. Aktér vyplní potřebné informace.

6.3.2.6 UC6: Smazání entity/relace

Vyučující má možnost upravit mapu pomocí smazání entity/relace.
Aktéři: Vyučující
Počáteční podmínka: Vyučující se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere entitu/relaci v mapě a otevře akce entity.
2. Systém zobrazí možnost smazání entity/relace pomocí tlačítka „Smazat“.
3. Aktér potvrdí smazání entity/relace.



Obrázek 6.2 Diagram případu užití: Znalostní mapa

6.3.3 Informace entity

6.3.3.1 UC1: Zobrazení detailu entity

Každá entita obsahuje navíc detailní informace ve formě informačního panelu. Detail entity se dělí na základní informace (hlavičku) a atributy na základě typu entity.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází v systému se zobrazenou vizualizací s entitami.
Scénář:

1. Aktér vybere entitu v grafu pomocí kliknutí na vizualizaci.
2. Systém zobrazí postranní panel s detailem entity:
   a. Základní informace (hlavička)
   b. Detailní informace na základě typu entity (atributy)

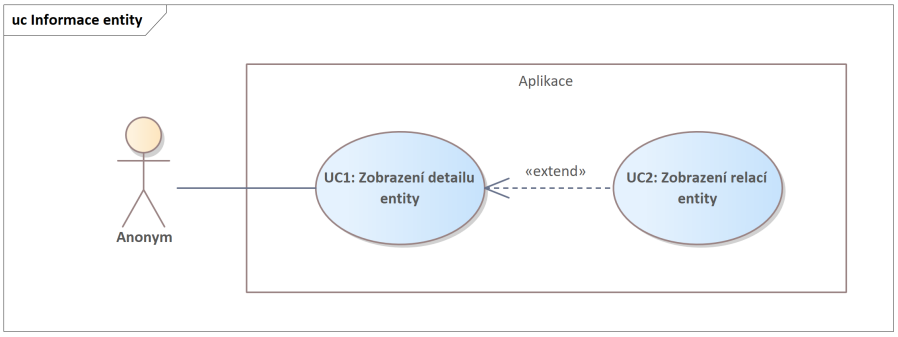
Alternativní scénář:

1. Aktér vybere možnost zobrazení detailu entity pomocí záložky „Detail“.
2. Systém zobrazí panel s detailem entity.

6.3.3.2 UC2: Zobrazení relace entity

Každá entita obsahuje navíc informace závislostí ve formě informačního panelu.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér má zobrazený detail entity.
Scénář:

1. Aktér vybere možnost zobrazení relace entity pomocí záložky „Závislosti“.
2. Systém zobrazí panel s relacemi entity.



Obrázek 6.3 Diagram případu užití: Informace o entitě

6.3.4 Pohledy entity

6.3.4.1 UC1: Zobrazení znalostního grafu entity

Znalostní graf entity se zaměřuje na přímé závislosti (relace) entity ve znalostní mapě v daném semestru.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere entitu v mapě pomocí kliknutí na vizualizaci.
2. Systém zobrazí možnosti akce entity.
3. Aktér vybere možnost závislosti entity pomocí tlačítka „Závislosti“.

6.3.4.2 UC2: Rozšíření/kolapsování entity

Znalostní graf lze rozšířit o sousedy vybrané entity. V opačném případě (entita je rozšířená) jej lze i zpětně „kolapsovat“ (*collapse*).
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostním grafu entity.
Scénář:

1. Aktér vybere entitu ve znalostním grafu entity pomocí kliknutí na vizualizaci.
2. Pokud byla entita:
   a. Kolapsovaná: Systém rozšíří entitu o její relace v grafu.
   b. Rozšířená: Systém kolapsuje entitu o její relace v grafu.

6.3.4.3 UC3: Zobrazení aktivity entity

Aktivita entity rozšíří znalostní graf entity o její aktivitu v čase a jaké entity byly změněny.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere entitu v mapě pomocí kliknutí na vizualizaci.
2. Systém zobrazí možnosti akce entity.
3. Aktér vybere možnost závislosti entity pomocí tlačítka „Aktivita“.

6.3.4.4 UC4: Zobrazení změny

Změny v aktivitě entity se indikují ve vizualizaci.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází v aktivitě entity.
Scénář:

1. Aktér vybere v postranním panelu vybranou změnu.
2. Systém zobrazí změnu ve vizualizaci:
   a. Přidané entity/relace
   b. Odebrané entity/relace
   c. Změněné entity/relace

6.3.4.5 UC5: Zobrazení verzí entity

Verze entity zobrazují návaznosti entit (konkrétně předmětu) v čase přes semestry a různé akreditace.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve znalostní mapě.
Scénář:

1. Aktér vybere entitu v mapě pomocí kliknutí na vizualizaci.
2. Systém zobrazí možnosti akce entity.
3. Aktér vybere možnost závislosti entity pomocí tlačítka „Verze“.

6.3.4.6 UC6: Zobrazení předchůdců

Předchůdci verze entity reprezentují všechny entity, které měly podíl na vzniku dané entity. Předchůdců entity může být vícero než jeden.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází ve verzích entity
Scénář:

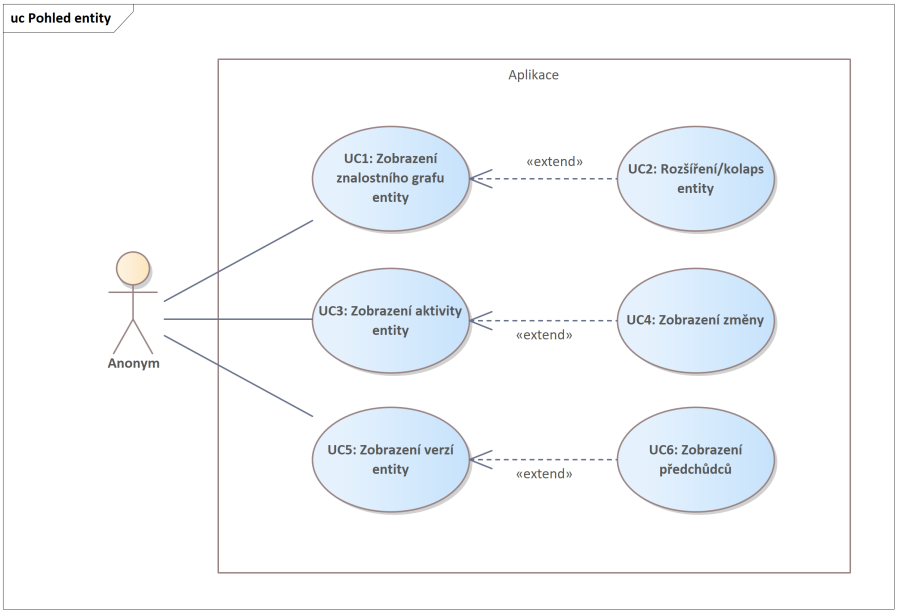
1. Aktér vybere entitu v mapě pomocí kliknutí na vizualizaci.
2. Systém zobrazí předchůdce verze entity pomocí vizualizace.

6.3.5 Přehled studia

6.3.5.1 UC1: Zobrazení přehledu studia

Přehled studia definuje hierarchii studia z pohledu programů, specializací/oborů, studijních plánů a předmětů specializace.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází v systému.
Scénář:

1. Aktér vybere v postranní liště možnost domovské stránky pod tlačítkem „Domů“.
2. Systém zobrazí domovskou stránku (nástěnku).
3. Aktér vybere možnost přehledu studia.
4. Systém zobrazí vizualizaci přehledu studia v aktuálním semestru pomocí hierarchie.

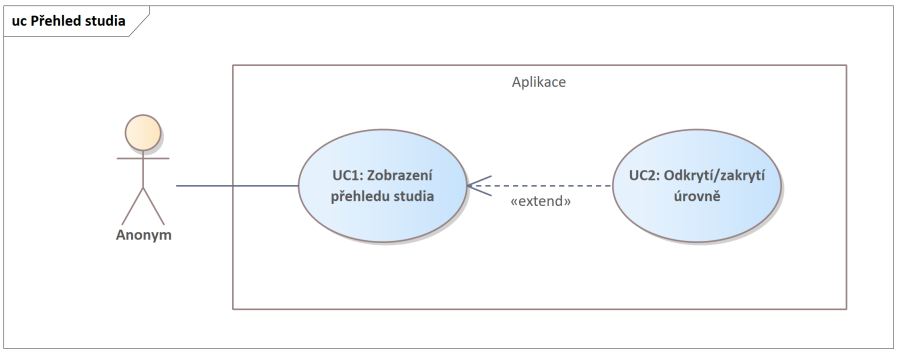


Obrázek 6.4 Diagram případu užití: Pohledy entity

6.3.5.2 UC2: Odkrytí/zakrytí úrovně

Hierarchická struktura lze postupně odkrývat (popřípadě zakrývat v opačném případě) pro větší zanoření úrovně v hierarchii.
Aktéři: Anonymní uživatel
Počáteční podmínka: Aktér se nachází v přehledu studia
Scénář:

1. Aktér vybere úroveň hierarchie pomocí kliknutí na vizualizaci.
2. Pokud byla úroveň:
   a. Odkrytá: Systém zakryje úroveň hierarchie ve vizualizaci.
   b. Zakrytá: Systém odkryje úroveň hierarchie ve vizualizaci.



Obrázek 6.5 Diagram případu užití: Přehled studia

6.4 Popis aplikace

Podle definovaných funkčních požadavků a navržených případů užití byla navrhnuta mapa aplikace ve formě diagramu úloh (viz. sekce 6.4.1). Návrh aplikace probíhal agilním přístupem, kde oproti lineárním přístupu (tzv. *vodopád*) jsou využity krátké opakující se iterace pro zlepšení návrhu. Výhodou je rychlá reakce na změny při reprezentaci návrhu cílovému klientovi.

6.4.1 Diagram úloh

Diagram úloh (*task graph*) pomáhá vytvořit pohled na průchod aplikací a jejím uživatelským rozhraním [52]. Hlavním účelem je zachytit tok akcí a přechod mezi jednotlivými stavy rozhraní aplikace. Jednotlivé úlohy (*tasks*) reprezentují akce, které uživatel má na stránce k dispozici. Provedení dané úlohy přesměruje uživatele na stav s vizuální změnou rozhraní aplikace (např. otevření podokna, zobrazení další sekce pomocí záložek, nebo otevření kompletně nového okna se stránkou).

Diagram úloh pro aplikaci lze vidět na obrázku 6.6. Úlohy (akce) jsou reprezentovány bílým elementem s kulatými rohy, stavy jsou reprezentovány šedým elementem s hranatými rohy. Diagram obsahuje výhradně akce pro anonymního uživatele, na kterého je práce zaměřená. Jak lze vidět na obrázku, některé úlohy korespondují s případy užití (viz. výše), kde bylo důležité specifikovat funkcionality a aktéry. Zde je důležité zobrazit průchod rozhraním aplikace. Úlohy a stavy jsou navíc rozděleny do šesti logických sekcí rozhraní aplikace:

Domovská stránka Domovská stránka slouží jako jednoduchý rozcestník aplikace. Na domovskou stránku se lze dostat z každého stavu rozhraní aplikace, proto v diagramu není zobrazena žádná úloha, která by směrovala na tuto stránku. Stránka nabízí dvě úlohy, první pro zobrazení znalostní mapy a druhou pro zobrazení přehledu studia.

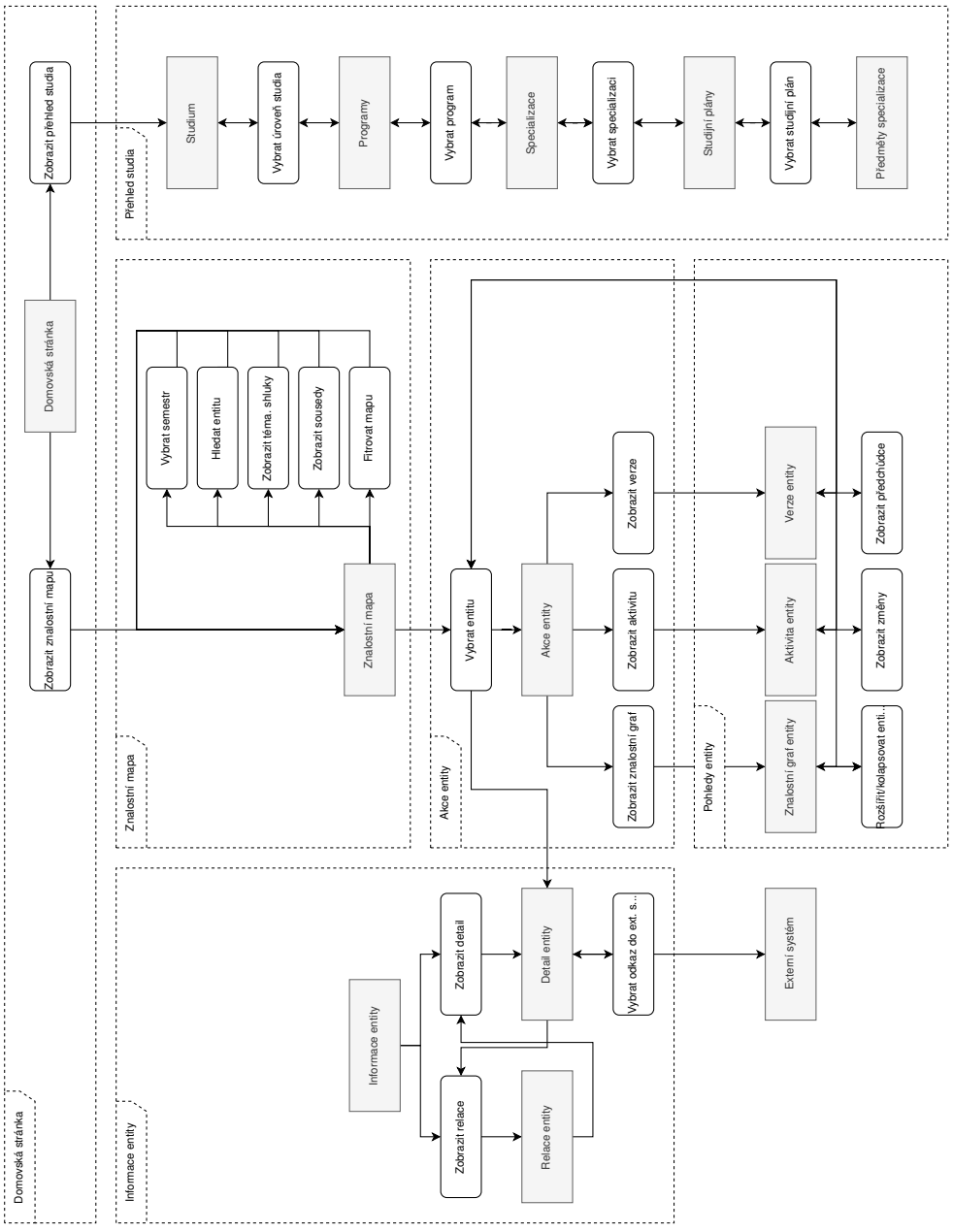
Znalostní mapa Znalostní mapa obsahuje znalostní graf pro entity sylabu (předmět, vyučující, téma, studijní plán a katedra) s relacemi. Mapa nabízí několik úloh, které změní její stav (filtrování, hledání, změna semestru, zobrazení tématických shluků a sousedů), ale existují v rámci mapy a tudíž tyto úlohy vedou zpět do znalostní mapy.

Akce entity Akce entity se zobrazí při vybrání entity v několika definovaných stavech rozhraní aplikace. Akce jsou reprezentovány podoknem a obsahují úlohy pro zobrazení pohledů entity (znalostního grafu, aktivity a verzí entity). Akce lze otevřít i ze samotných pohledů entity. Výběr entity zároveň směruje na informace entity.

Informace entity Informace entity obsahují dva vzájemně přepínatelné panely, první pro detail entity a druhé pro její relace. V detailu lze navíc přesměrovat z aplikace do vybraného externího systému pomocí odkazu (BK, Fittable, KOS, CoursesFIT, Anketa ČVUT, Usermap). Informace entity jsou ve formě postranního panelu a neobsahují žádné další úlohy.

Pohledy entity Pohledy entity se zaměřují na samostatnou entitu (závislosti, aktivity a verzí entity). Ve znalostním grafu entity lze rozšířit/zúžit entitu pro rozšíření/zúžení grafu její relací. V aktivitě entity lze zobrazit změny v grafu. Ve verzích entity lze zobrazit předchůdce dané verze. Z každého pohledu lze opět zobrazit akce entity.

Přehled studia Přehled studia je víceméně „izolovaná“ sekce, neboť prochází hierarchií entit (úrovně studia, studijní program, specializace/obor, studijní plán, předměty specializace) navzájem mezi sebou.



Obrázek 6.6 Diagram úloh: Anonymní uživatel

6.5 Uživatelské rozhraní

Návrh uživatelského rozhraní (*user interface*, také jako UI) se zabývá návrhem grafických elementů rozhraní aplikace. Příbuzným pojmem je také uživatelská zkušenost (*user experience*, také jako UX), který klade důraz na použitelnost funkcionalit cílovou skupinou uživatelů. Společným cílem UI a UX je vytvořit takové rozhraní, které je uživatelsky přívětivé, ale především užitečné a vhodné pro cílovou skupinu uživatelů.

V této práci je návrh uživatelského rozhraní poněkud netradiční. Hlavní komponentou jsou totiž interaktivní vizualizace, které se (ve webovém prostředí) uchytily až s nástupem modernějších technologií. Klasický návrh uživatelského rozhraní utilizuje elementy jako tlačítka, tabulky, seznamy, obrázky, formuláře aj. Zatímco vizualizace mohou používat tyto pomocné prvky, jejich zájem je soustředěn do geometrických elementů jako obrazce, šipky a animace.

6.5.1 Nielsonova heuristika

Při návrhu jakéhokoliv uživatelského rozhraní je doporučeno držet se zásad *Nielsenovy heuristiky* podle Jakoba Nielsena [53]. Jedná se o soubor desíti pravidel, které slouží jako univerzální návod pro dosáhnutí lepší použitelnosti a intuitivního ovládání uživatelského rozhraní. Pravidla byla vytvořena z dlouhodobých pozorování uživatelů a jejich interakcí s uživatelským rozhraním:

Viditelnost stavu systému Uživatel by měl být informován o tom, kde se v systému nachází (informace o stavu systému).

Shoda mezi systémem a realitou Používané termíny by měly být v souladu s terminologií cílové skupiny (metafory z reálného světa, přirozené pojmenování).

Minimální zodpovědnost Při neúmyslné akci uživatele by uživatel měl nést minimální zodpovědnost a mít možnost vrátit se zpět v systému.

Konzistence a obecné standardy Používané elementy by měly být konzistentní a vypadat podobně, popř. dodržovat obecně dané standardy.

Prevence chyb Uživatel by neměl kontrolovat svůj vstup, ale rozhraní ano. Kontrola při zadání vstupu zabrání chybám v systému.

Navádět než vzpomenout si Systém uživatele navádí, uživatel má vše viditelné. Uživatel by měl použít minimum své paměti pro používání systému.

Flexibilita a efektivita Systém by měl mít k dispozici funkcionality i pro zkušené uživatele systému a tím je provádět efektivně.

Minimalistický Uživatel by měl v systému vidět jen ty informace, které jsou pro něj relevantní. Grafika je smysluplná a relevantní.

Smysluplné chybové hlášky Při chybě je zobrazena chybová hláška, která informuje uživatele dostatečně.

Nápověda a dokumentace Systém by měl být intuitivní a použitelný bez nápovědy, ale nápověda je k dispozici.

6.5.2 Drátěný model

Pro návrh uživatelského rozhraní se používají drátěné modely (*wireframes*). Účelem drátěných modelů je zachytit hlavní myšlenku návrhu. V podstatě se jedná o nakreslené obrazovky aplikace,

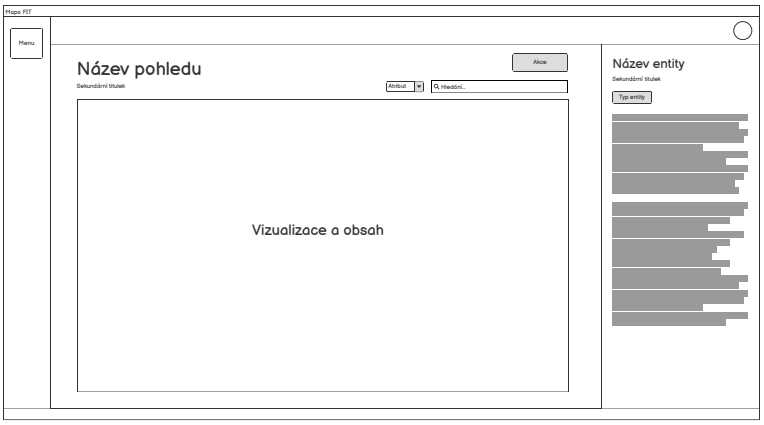
které nejsou interaktivní. Jeho podoba (většinou) neodpovídá výslednému produktu, neboť zachycuje elementy na obrazovce jen přibližně. Proto jsou drátěné modely (většinou) jednoduché, černobílé a obsahují demonstrativní výplňový text (např. *lorem ipsum*).

Dnešní moderní nástroje umožňují i lehké napodobení interaktivnosti ve formě přepínání nakreslených obrazovek při kliknutí do ní. Tato funkcionalita je možná např. pomocí formátu PDF (*Portable Document Format*). Drátěné modely vytvořené v této kapitole byly vytvořeny pomocí nástroje *Balsamiq Wireframes*. Tento nástroj umožňuje interaktivní přepínání nakreslených obrazovek. Vytvořené drátěné modely v této kapitole lze zobrazit v přílohách v interaktivním formátu PDF.

6.5.2.1 Low-fidelity

Nízkoúrovňový (*low-fidelity*) model je nejjednodušší formou drátěného modelu. Zde je kladen důraz na typ elementů a jejich umístění na obrazovce. Na obrázku 6.7 lze vidět první model aplikace ve formě nízkoúrovňového modelu. Základem aplikace je prostorné plátno pro vizualizace, které je podepřeno záhlavím. Záhlaví obsahuje hlavní titulek s názvem pohledu vizualizace, podtitulek pro dodatečné informace po levé straně a doplňující elementy po pravé straně. Po pravé straně lze vidět panel, který také podporuje vizualizaci ve formě dodatečného obsahu informací. Hlavní komponentou panelu je jeho hlavička s názvem, podtitulkem entity a typem entity. Po levé straně je malý panel s menu pro rychlou navigaci.

Toto rozložení bylo zvoleno především z důvodu zaměření na samotné vizualizace. Díky tomuto rozložení mají vizualizace dostatek prostoru, ale zároveň jsou podporovány dalšími tradičními elementy grafického návrhu. V budoucích iteracích projektu se může toto rozložení měnit, ale vizualizace vždy potřebují vlastní plátno a dostatek prostoru pro vykreslení a čitelnost.



Obrázek 6.7 Low-fidelity drátěný model: První model základu aplikace

6.5.2.2 High-fidelity

Vysokoúrovňový (*high-fidelity*) model je další úrovní drátěného modelu. Avšak na rozdíl od nízkoúrovňových obsahuje detailnější informace, typicky z reálného prostředí a výsledného produktu. Účelem je zobrazit cílový vzhled a interakce uživatelského rozhraní. Některé vysokoúrovňové modely zahrnují i barvu elementů. V této kapitole jsou všechny vysokoúrovňové modely černobílé z důvodu efektivity, avšak odlišné zabarvení je znázorněno odstínem šedi.

6.5.2.3 Znalostní mapa

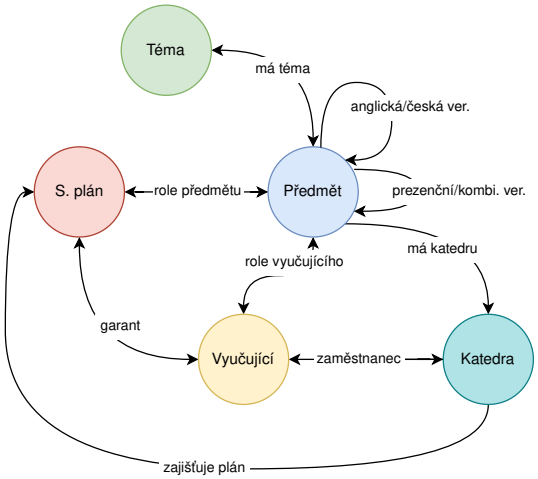
První vizualizací je znalostní mapa. Jedná se o orientovaný graf s entitami jako vrcholy grafu, kde entity se dělí podle typu. Graf je v optimálním rozložení (force-directed rozložení), neboť je takto nejčitelnější. Mapa je zachycena pro daný semestr. Znalostní graf byl již zmíněn v analýze (viz. sekce 2.1.4). Šipky relací značí jejich orientaci, kde některé jsou obousměrné. Návrh této verze grafu obsahuje následující entity a relace (schéma grafu lze vidět na obrázku 6.8):

- Předmět –*česká v.*→ Předmět
- Předmět –*anglická v.*→ Předmět
- Předmět –*prezenční v.*→ Předmět
- Předmět –*kombinovaná v.*→ Předmět
- Předmět ←*má téma*→ Téma
- Předmět ←*katedra*→ Katedra
- Předmět ←*role vyučujícího*→ Vyučující
  - Předmět ←*cvičící*→ Vyučující
  - Předmět ←*přednášející*→ Vyučující
  - Předmět ←*garant*→ Vyučující
- Předmět ←*role předmětu*→ Studijní plán
  - Předmět ←*PP*→ Studijní plán
  - Předmět ←*PS*→ Studijní plán
  - Předmět ←*PV*→ Studijní plán
  - Předmět ←*PT*→ Studijní plán
  - Předmět ←*PO*→ Studijní plán
  - Předmět ←*PJ*→ Studijní plán
  - Předmět ←*PZ*→ Studijní plán
- Vyučující ←*zaměstnanec*→ Katedra
- Vyučující ←*garant*→ Studijní plán
- Katedra ←*zajišťuje*→ Studijní plán

Role předmětu je substituovaná konkrétní rolí (viz. seznam nebo schéma) a podobně i *role vyučujícího*. Graf vystihuje nejvíce robustní a důležité vztahy v sylabu (viz. konceptuální model v 6.1 nebo analýza sylabu v kapitole 1).

Model grafu nezahrnuje studijní programy a specializace, neboť se nachází v hierarchickém vztahu (společně se studijním plánem) a neměly by ve vizualizaci takový přínos jako ostatní entity, které obsahují robustnější propojení. *Studijní plán* je spojovým bodem mezi specializacemi a předměty. Zároveň přináší další relace s katedrou a vyučujícím, proto byl v mapě obsažen.

Téma předmětu je navržená entita, která představuje (v daném momentě) všeobecný pojem. V analýze byly zmíněny osnovy přednášek, cvičení a klíčová slova, které obsahují témata předmětů. V budoucích iteracích projektu je v zájmu vytvoření nové konceptualizace sylabu a především témat předmětu. V rámci použitelnosti se jedná o důležitou funkcionalitu, která by pro studenty mohla mít značný přínos. Dále lze graf transformovat a využít této relace pro zpracování tématických shluků (viz. sekce 6.5.2.4).

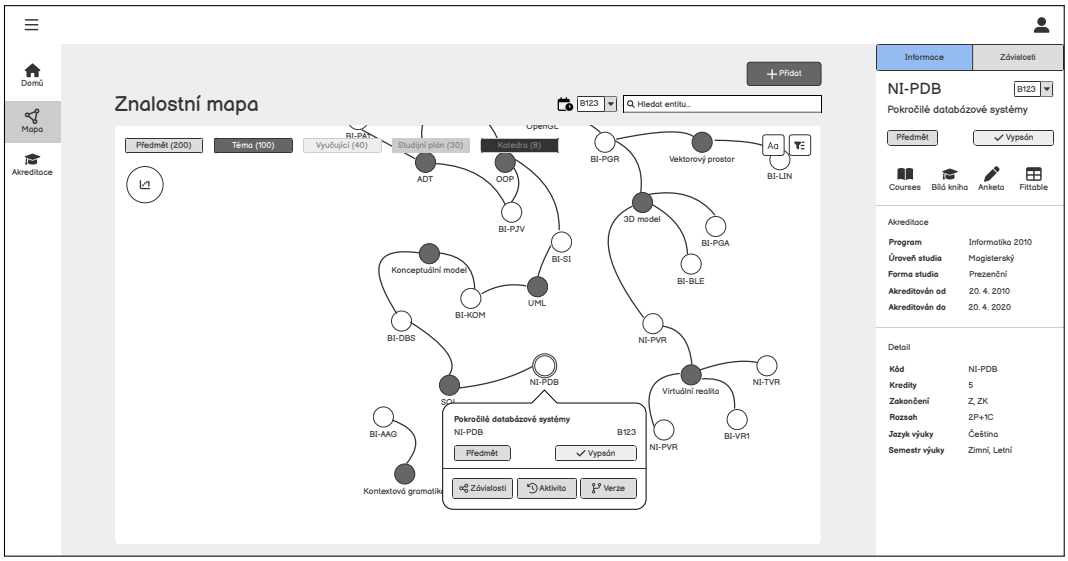


Obrázek 6.8 Znalostní mapa: Schéma znalostního grafu

Vizualizace grafu je udržuje simplicitu. Kruhové vrcholy grafu jsou obarveny podle typu entity, který reprezentuje. Název entity je reprezentován popiskem. Některé názvy mohou být poněkud dlouhé, proto je preferencí dát přednost zkrácenému názvu. Orientované hrany mezi vrcholy jsou reprezentovány šipkami a obsahují popisek podél své hrany.

Aby nebyl uživatel zahlcen množstvím vrcholů a hran v grafu, tak byly navrženy dva typy filtrů. První typ filtruje entity podle typu a je jednoduše aktivovatelný kliknutím na dané tlačítko v horní části grafu. Druhý typ filtru filtruje mapu podle dodatečných atribut entit, především předmětů, které obsahují nejvíce atribut. Otevřít a aplikovat tento filtr je možné přes tlačítko v pravém horním rohu. K dispozici je také vyhledávací panel v záhlaví nad vizualizací grafu. Vyhledávací panel vyhledá jednu entitu v mapě a zvýrazní jí.

Při výběru entity kliknutím na vrchol grafu je zobrazen panel se základní informací o entitě a její možné akce. Základní informace reflektují ty v postranním panelu (více v 6.5.2.5), aby rozhraní udržovalo konzistenci. Možné akce entity zobrazí její pohledy (závislosti, aktivita, verze). Návrh rozhraní lze vidět na obrázku 6.9.

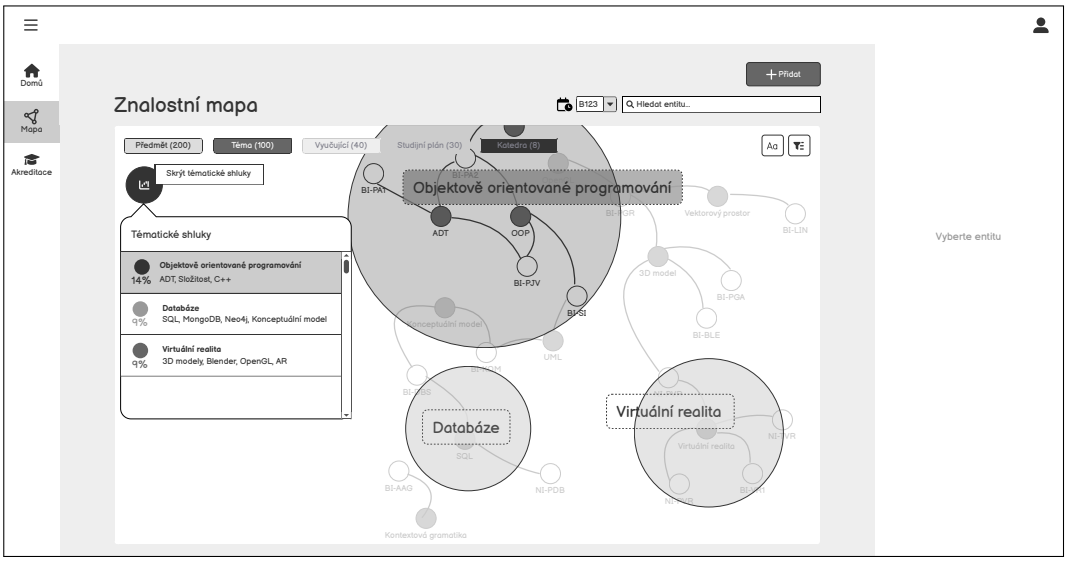


Obrázek 6.9 High-fidelity drátěný model: Znalostní mapa

6.5.2.4 Tématické shluky

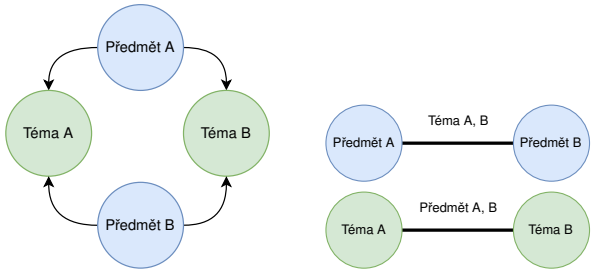
Další funkcionalitou jsou tématické sluky. Tato funkcionalita vychází z detekce komunit v grafu (viz. sekce 2.1.3). Díky relací témat a předmětů ve znalostní mapě lze graf transformovat do podoby, která bude zobrazovat shluky vizuálně.

První model shluků obsahoval detekci komunit v původním grafu s oběma typy entit (předmět a téma, viz. obrázek 6.10). Detekce komunit ve znalostním grafu s různými typy entit je validní a možná, ale vzhledem k tomu, že takové množství entit může být obrovské, tak byla vytvořena druhá verze modelu.



Obrázek 6.10 Původní (starý) model: Tématické shluky

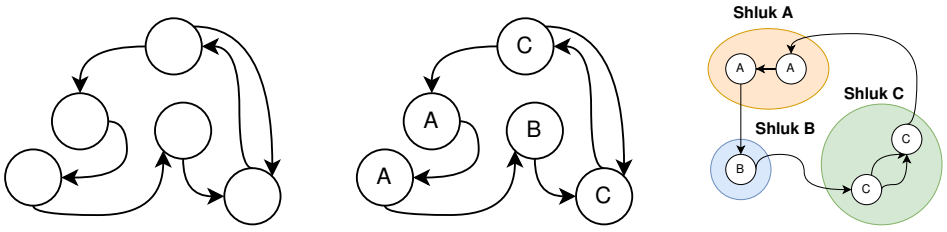
Druhý model shluků pracuje s ideou, že eliminuje jeden typ entity a tím ponechá v grafu jen téma nebo předmět. Graf se stane neorientovaným grafem. Hrana mezi vrcholy představuje skutečnost, že dané entity sdílí společné téma nebo předmět a také určuje, který to je (viz. obrázek 6.11). Tím se značně zredukuje množství vrcholů v grafu, který zároveň ponechá důležité informace. Největší přínos je možnost obou dvou variant a přepínat mezi nimi. Nevýhodou jsou entity, které mají vazbu jen na jednu entitu a tím nejsou prostředníkem. Takové relace nejsou zachyceny v transformovaném modelu.



(a) Znalostní graf (b) Transformovaný graf

Obrázek 6.11 Tématické shluky: Transformovaný graf

Shluky musí být nejdříve v grafu detekovány, poté je možné pracovat s vrcholy. Samostatné shluky jsou vizualizovány pomocí atrakce vrcholů ve shluku a konvexní obálky, která je obaluje (viz. obrázek 6.12). Dalším elementem je nejvíce vlivný vrchol shluku. Takový vrchol nese pojmenování celého shluku a je zobrazen v místě zájmu ve vizualizaci. Pro detailnější informace o detekovaných shlucích je k dispozici pomocný panel po levé straně grafu. Panel obsahuje informace o obsaženosti shluků.



(a) Orientovaný graf (b) Detekované shluky (c) Vizualizace shluků

Obrázek 6.12 Tématické shluky: Vizualizace shluků

6.5.2.5 Informace entity

Při výběru entity v mapě (nebo i v pohledech) se zobrazí informační panel s dvěma záložkami. První záložka zobrazuje detailní informace entity. Hlavní část tvoří hlavička, která koresponduje s menším panelem ve vizualizaci (viz. sekce 6.5.2.3). Sekundární část tvoří samostatné atributy entity a liší se dle typu. V této práci nebude rozebrán sekundární obsah panelu. Vizualizaci panelu lze vidět na obrázku 6.9.

Hlavička obsahuje název, zkrácený název, semestr, typ entity a status entity. Typ entity koresponduje s typem ve znalostní mapě. Status entity zobrazuje zda je entita aktivní, tedy je relevantní vůči probíhajícímu semestru. Status entity zobrazuje např. stavy vyučujících (zda jsou stále zaměstnanci fakulty), stavy předmětů a studijních plánů (zda jsou akreditované a platné). Pod hlavičkou je seznam odkazů do externích systémů, které jsou relevantní vůči danému typu entity. Odkazy také odpovídají semestru (verzi) entity, např. vyhodnocení ankety předmětu v daném semestru. Možné odkazy pro entity:

- Course Pages FIT ČVUT
- Anketa ČVUT
- KOS
- Fittable
- Bílá kniha ČVUT
- Projekty FIT
- Usermap

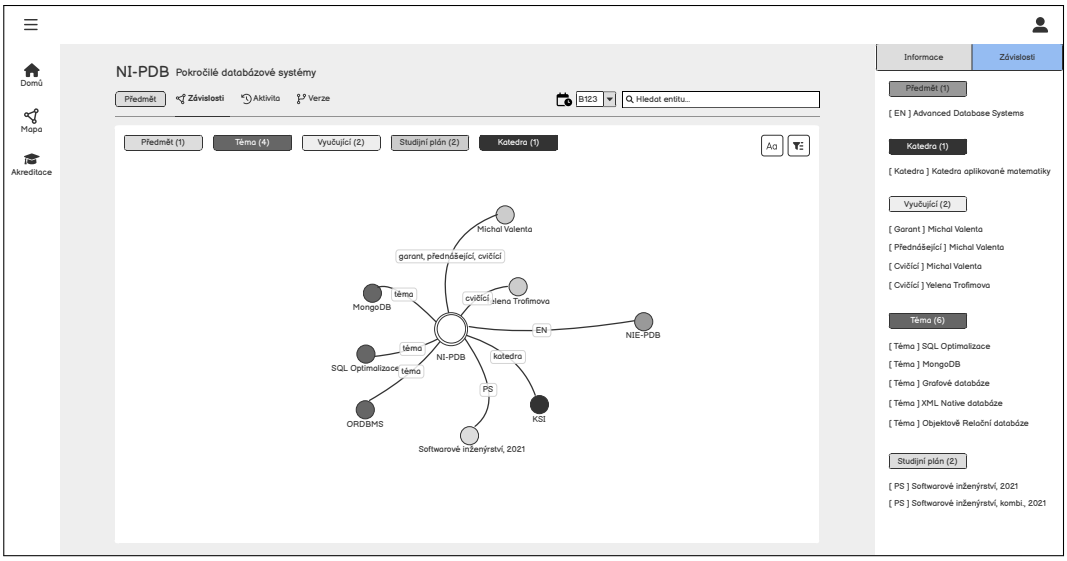
Druhá záložka zobrazuje všechny závislosti (relace) entity roztříděné podle typů entit. Relace má svůj popisek s názvem a typem relace, např. zda je vyučující cvičícím, přednášejícím, nebo garantem předmětu. Vizualizaci záložky s relacemi lze vidět na obrázku 6.13.

6.5.2.6 Závislosti entity

Závislosti entity zobrazují znalostní graf z mapy (viz. sekce 6.5.2.3), ale jen přímé závislosti vybrané entity. Motivací pro tento pohled bylo zobrazení všech potřebných informací na jednom místě a přehledně. Zároveň mít možnost rychle navigovat mezi závislostmi. Pohled obsahuje podobné funkcionality (filtry, hledání, výběr entity, změna semestru, vzhled grafu), ale místo tématických shluků obsahuje funkcionalitu *rozšíření grafu*.

Při výběru entity se opět zobrazí panel se základními informacemi, ale také se daná entita rozvětví o své sousedy. Následně jej lze rozšiřovat dál. Naopak při výběru rozšířené entity je entita opět uzavřena (zúžena).

Rozhraní má odlišné záhlaví, kde titulek reprezentuje název entity (popř. zkrácený název). Také obsahuje tři záložky, každou pro jiný pohled entity (závislosti, aktivita, verze). Vizualizaci lze vidět na obrázku 6.13.

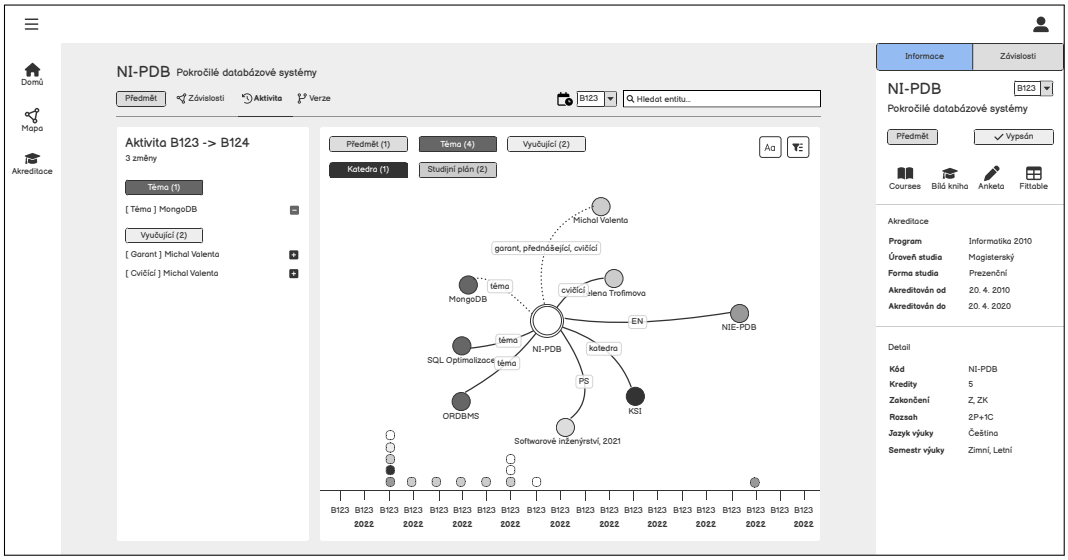


Obrázek 6.13 High-fidelity drátěný model: Závislosti entity

6.5.2.7 Aktivita entity

Pro zobrazení změn závislostí entity v čase slouží aktivita entity. Obohatí znalostní graf o časovou osu, která obsahuje změny v kompaktním zobrazení přes barvy typů entit. Pohled stále zobrazuje entitu a její závislosti v daném semestru, avšak po levé straně je přidán panel s aktivitou. V panelu je seznam provedených změn během dvou sousedních semestrů (předešlého a současného). Tedy co bylo přidáno, odebráno a případně změněno (atributy entity). Graf entity zároveň vizuálně zobrazuje dané změny (viz. obrázek 6.14).

Aktivita entity je hlavně záležitost do nadcházejících iteracích projektu. Na konceptualizaci a návrhu pro sledování změn v sylabech předmětů pracuje Samuel Händl ve své bakalářské práci ve stejném akademickém roce jako byla psána tato práce.

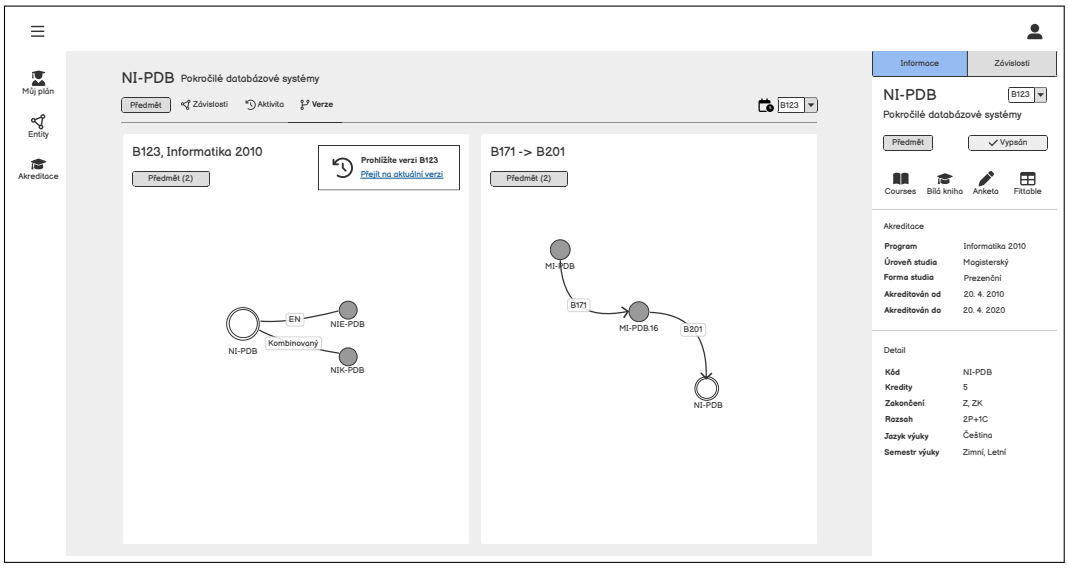


Obrázek 6.14 High-fidelity drátěný model: Aktivita entity

6.5.2.8 Verze entity

Verze entity se týká primárně předmětu, avšak v teorii se dá aplikovat i na jiné entity. Zobrazuje její pohyb v čase přes semestry a jakým způsobem se měnila ona samotná (tedy nejedná se o to samé jako aktivita entity, kde se sledují změny závislostí). Motivací pro tuto vizualizaci byly předměty, které na sebe časově navazují, ale neexistuje mezi nimi provázanost.

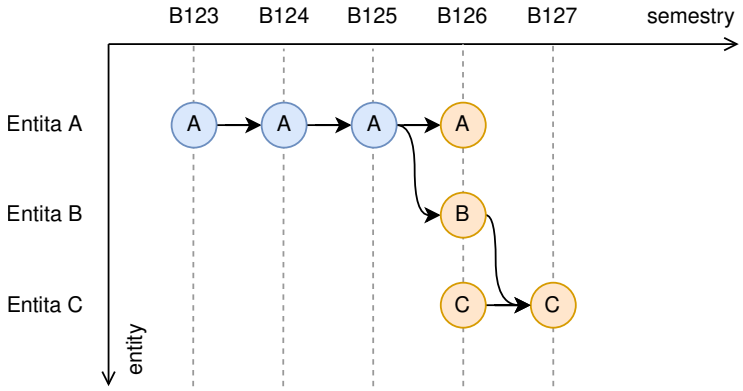
První model obsahoval dva pohledy na verze - v semestru a přes semestry. Výhodou bylo zobrazení všech možných verzí najednou, ale naopak se plátno rozdělilo a tím se i zmenšil prostor. Navíc verze v semestru neměla hlubší význam - tyto závislosti lze zobrazit i pomocí závislostí entity (viz. výše) a většinou jsou maximálně dvě (forma studia a jazyk). Návrh vizualizace přes semestry taky nebyl řádně promyšlen, protože se jednalo o nepřehledný graf cesty (viz. 6.15).



Obrázek 6.15 Původní (starý) model: Verze entity

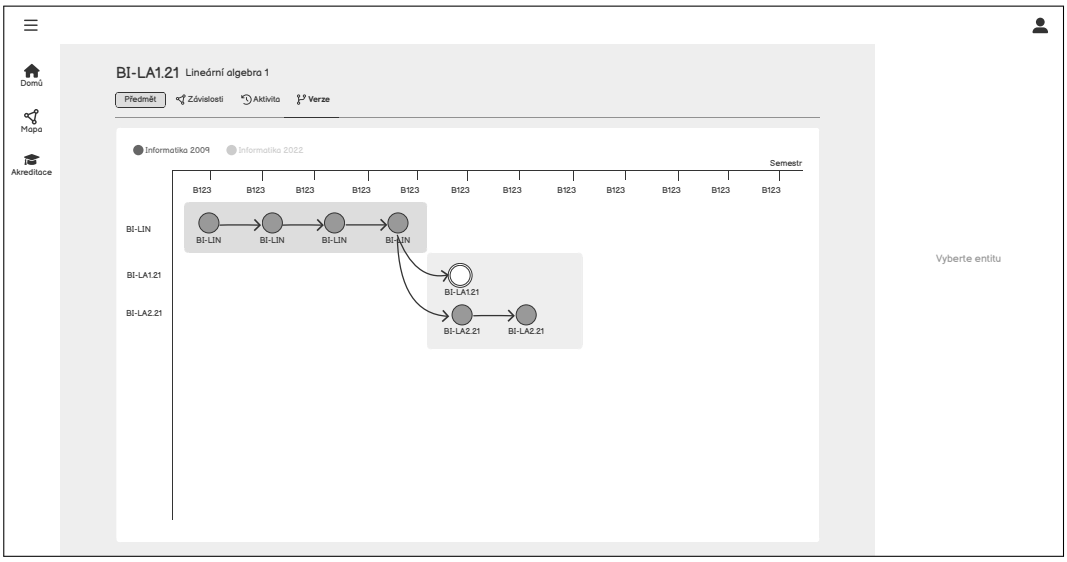
Z prvního modelu vznikl druhý model, který se zaměřuje na verze přes semestry (v čase). Při analýze sylabu se ukázalo, že předměty mohou mít schopnost se rozštěpit nebo naopak sloučit (např. vzájemnou uznatelnost při nové akreditaci). Proto byla navržena vizualizace, která je schopná zobrazit jednoduché posuny v čase (lineární), ale i ty komplexnější (sloučení, rozštěpení).

Verze entity zobrazí graf, který obsahuje všechny entity co se v čase podílely na vzniku vybrané entity. Lze jej přirovnat k rodokmenu, avšak zde může být libovolný počet rodičů i potomků (viz. obrázek 6.16). Vizualizace obsahuje dvě osy - vertikální pro entity a horizontální pro čas (semestry). V horizontální ose se postupně řadí entity, které jsou na sebe závislé. Hrana mezi entitami znamená návaznost v čase. Ve vertikální ose se řadí různé entity, které vznikly nebo se naopak podílely na vzniku.



Obrázek 6.16 Verze entity: Vizualizace grafu

Příklad lze vidět na obrázku 6.17 a entitě BI-LA1.21. Předmět vznikl původem z předmětu BI-LIN. Předmět BI-LIN se rozštěpil na dva předměty příchodem nové akreditace. Akreditace (v tomto případě studijní programy) jsou navíc zvýrazněny odlišnou barvou v grafu a tím zobrazují, zda byla změna způsobena novou akreditací. Entity lze vybrat a zobrazit jejich detail (v postranním panelu) podobně jako u znalostní mapy, závislostí nebo aktivit entity.



Obrázek 6.17 High-fidelity drátěný model: Verze entity

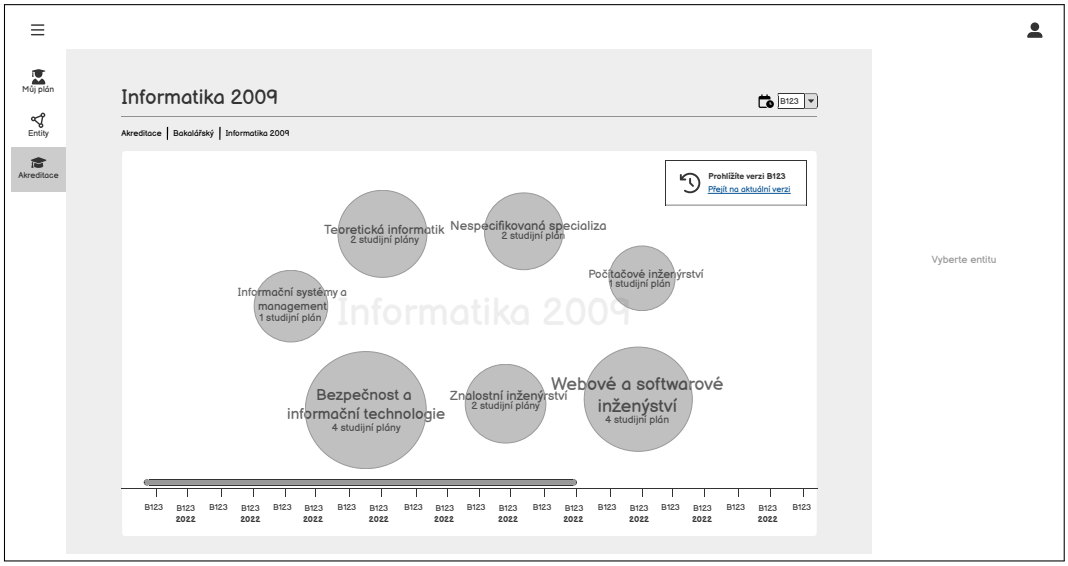
6.5.2.9 Přehled studia

Poslední vizualizace se zaměřuje na zobrazení průchodu studia a možné cesty studenta ve výběru specializace. Jak již bylo zmíněno dříve, tyto entity se drží hierarchické struktury. Jedná se o studijní programy, specializace/obory, studijní plány a popř. předměty specializace.

Původní model se zaměřoval na dynamický průchod přes různé úrovně hierarchie. První úroveň reprezentovala úrovně studia (kořen). Ta se následně rozdělila na jednotlivé programy a dále podle struktury zmíněné výše. Hlavní vizualizace vždy obsahovala zobrazenou jen jednu úroveň ve formě tzv. *word cloud*, kde text reprezentuje primární element. Potomci vrcholu byly doplněny o kruhové elementy podobně jako u tzv. *bubble chart* a ty byly škálovány dle velikosti jejich potomků. Celkově potomci obsahovaly svůj název a sekundární název obsahující počet a typ potomků (viz. obrázek 6.17). Při výběru entity nastane přechod vizualizace do další úrovně hierarchie. Při výběru aktuálně zobrazené entity (uprostřed plátna) se vizualizace posune zpět o úroveň výše.

Původní idea také obsahovala časovou osu, kde byla zobrazena životnost (platnost) akreditací, konkrétně studijních programů. Rozlišovaly se pomocí zabarvení podle úrovně studia (bakalářský, magisterský, doktorský). Časová osa sice může poskytnout další formu pohledu, ale byla spíše podpůrnou funkcionalitou pro účel zobrazení hierarchie. Zároveň může působit nepřehledně pro uživatele, obzvlášť při několika paralelních linek, které by musely najít prostor na plátně. Je tedy vhodným kandidátem pro separátní scénu.

Ohledně zobrazení entit byl problém, že vizualizace sice působila přehledně a intuitivně, ale nenesla samostatnou ideu hierarchie správně. V předchozí kapitole (viz. kapitola 2) byly zmíněny různé způsoby pro její zobrazení a zmíněné principy obsahovaly buď stromové zobrazení (dendrogram) nebo mapu zanoření (circle packing, treemap). Cílem druhého modelu bylo vylepšení vizualizace pomocí těchto principů.

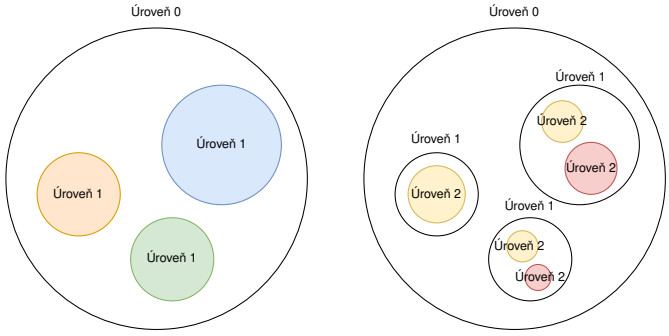


Obrázek 6.18 Původní (starý) model: Přehled studia

Jelikož se původní model podobal nejvíce kruhovému zanoření (circle packing), tak byla navržena vizualizace jím inspirovaná. Hierarchie je tedy přehlednější a intuitivnější. Druhý model také vyřadil časovou osu a zaměřil se více na vlastnosti jednotlivých úrovní.

Kruhové zanoření obsahuje vnořené elementy. Klasický model obsahuje viditelnost od kořene až do poslední úrovně, čímž při hlubokém zanoření může způsobovat problémy s čitelností. Druhý model vizualizace obsahuje interaktivní odkrytí potomků úrovně společně s automatickým přiblížením na potomka. Tímto se čitelnost vizualizace zvýší a zároveň uživatel vždy ví, v jakém

stavu se nachází. Stále existuje možnost posunout se o úroveň výše výběrem daného rodiče, kde úrovně jsou zakryty a vizualizace oddálena na pozici rodiče (viz. obrázek 6.19).

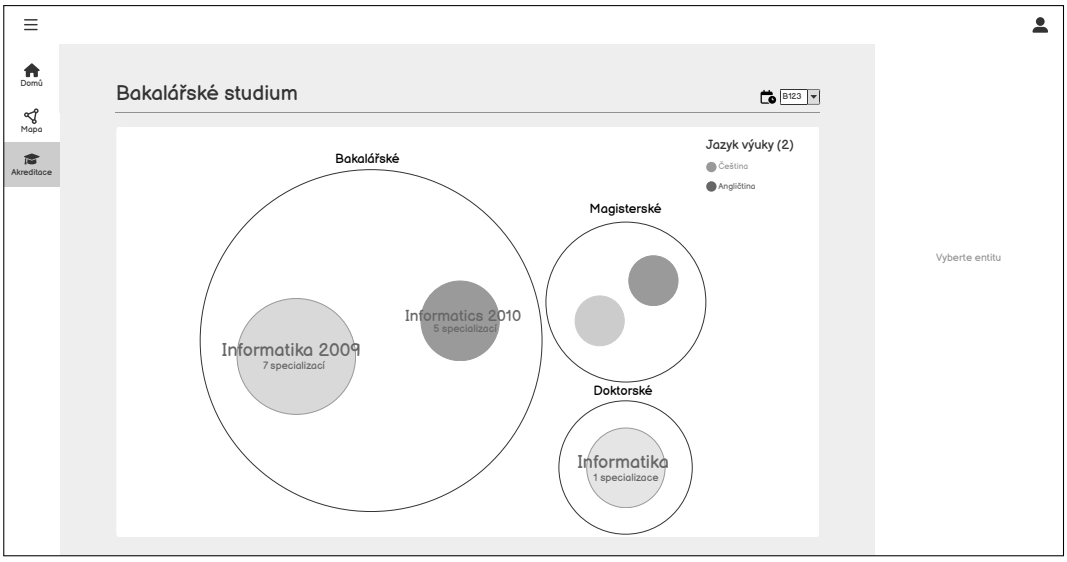


(a) Úroveň 0 (b) Úroveň 1

Obrázek 6.19 Přehled studia: Vizualizace odkrytí/zakrytí úrovní

Vzhled vizualizace je podobný pro potomky jako v prvním modelu (zabarvené s primárním a sekundárním titulkem). Zabarvení potomků je pro každou úroveň přizpůsobeno podle odlišných atributů s podpůrnou legendou v pravém horním rohu. Odkryté úrovně změní vizualizaci na jednoduchý kruh s názvem entity podél kruhu v horní části (viz. obrázek 6.20). Zabarvení dle zobrazených potomků hierarchie:

1. Úroveň studia (bakalářské, magisterské, doktorské)
2. Studijní programy (jazyk výuky)
3. Specializace (zajišťovaná katedra)
4. Studijní plány (forma studia)



Obrázek 6.20 High-fidelity drátěný model: Přehled studia

Kapitola 7
Implementace prototypu

Tato kapitola se zabývá implementací prototypu navržené aplikace a vizualizací. Nejdříve jsou diskutovány použité technologie pro jeho tvorbu. Detailněji jsou rozebrány hlavní knihovny D3.js a React. Následně je popsána architektura samotné aplikace a její komponenty. Dále jsou zmíněny zajímavosti ohledně implementace vizualizací. Poslední sekce kapitoly je věnována optimalizaci, kde jsou uvedeny možnosti pro zlepšení výkonu a budoucí vývoj. Výsledkem této části je prototyp aplikace a vizualizací.

7.1 Použité technologie

Webová aplikace se skládá z několika částí, avšak primárně ze tří technologií. Kostru stránky a její elementy představuje HTML. O vzhled stránky se stará CSS. Pro dynamiku rozhraní je používán JavaScript. Všechny tyto technologie dohromady tvoří základy pro webové stránky. Byly již detailněji popsané na začátku kapitoly 4, kde je také provedena rešerše vykreslování a vizualizačních knihoven pro web front-end aplikace.

Implementace prototypu využívá zmíněné technologie výše, avšak pro tvorbu moderních webových stránek jsou v dnešní době často využity i další technologie pro usnadnění vývoje. Řešení primárně zahrnuje práci s knihovnami *React* (viz. sekce 7.1.2) a *D3.js* (viz. sekce 7.1.1).

7.1.1 D3.js

Pro realizaci vizualizací byla použita knihovna D3.js (viz. rešerše knihovny v sekci 4.2). Důvodem zvolení této knihovny je především široká komunita, možnost přizpůsobení a spolehlivá dokumentace. Knihovna navíc nabízí vykreslování pomocí `<svg>` i `<canvas>`. V rámci tvorby vizualizací nabízí síťové grafy, dendrogramy, hierarchické mapy, ale i klasické spojnicové grafy. Všechny vizualizace jsou vytvořeny ze stavebních bloků, které knihovna nabízí. Díky tomu není realizace navržených vizualizací omezena a lze prakticky vytvořit téměř cokoliv.

7.1.2 React

Pro tvorbu aplikace byla použita knihovna *React*. Jedná se o populární knihovnu, která poskytuje implementaci web front-end aplikace. Knihovna byla vyvinuta společností Facebook. Mezi hlavní principy knihovny patří využití virtuálního DOM, který optimalizuje vykreslování a překresluje jen ty elementy, které to vyžadují (více také v sekci 7.4.1).

Vývoj v Reactu je postaven na JSX (*JavaScript Syntax Extension*) komponentách. JSX rozšiřuje syntaxi JavaScript kódu, který tím podporuje provázanost mezi vykreslováním ele-

mentů (podobně jako HTML elementy) a další logiky pro správu stavů, eventů aj. Například výraz `<h1>{title}</h1>` je JSX výrazem, kde proměnná `title` je později kompilátorem nahrazena příslušnou hodnotou. Pomocí JSX je psaní kódu v JS přehlednější a pohodlnější.

Dříve byl React zaměřen na tzv. *class komponenty*, kde každá vytvořená komponenta rozšiřovala třídu `React.Component`. Životní cyklus komponenty se skládá primárně ze tří částí - *mounting* (nasazení), *updating* (aktualizace) a *unmounting* (vysazení). Všechny stavy odpovídají příslušným metodám, které třída nabízí. Následně pomocí metody `render()` byly elementy ve formě JSX vykresleny.

Postupem času konvergoval vývoj komponent na tzv. *funkcionální komponenty*, které také byly použity po vývoj této aplikace. Jak již plyne z názvu, funkcionální komponenty jsou JavaScript funkcemi. Vykreslování zajišťuje samotné tělo funkce (na rozdíl od metody v class komponentě). Životní cyklus komponenty zajišťují procedury *Hooks*. Od verze React 16.8.0 jsou v knihovně implementovány stabilní verze Hooks [54]. Tyto procedury v podstatě zajišťují různé funkcionality pro správu stavů, překreslování, cachování nebo dokonce i vlastní přepoužitelné logiky do komponent.

V implementaci prototypu je React využit jako jádro web front-end aplikace. Následně je vizualizační knihovna D3.js integrována do prostředí React komponent. Zároveň jsou diskutovány různé metody integrace, které jsou v této situaci k dispozici (viz. sekce 7.4). Motivací pro vzájemnou integraci těchto dvou knihoven je rozdělení zodpovědností (*seperation of concerns*) a tím i přehlednější strukturu kódu pro budoucí vývoj.

7.1.3 Material UI

Pro jednoduchou a rychlou tvorbu stylizovaných komponent byla použita knihovna Material UI. Tato knihovna je exkluzivně dostupná pro vývoj v Reactu a nabízí široké množství komponent pro vývoj moderní web front-end aplikace. Knihovna nabízí API pro tlačítka, formuláře, avatary, seznamy, dialogy a mnohé další.

7.1.4 SASS/SCSS

SASS (*Syntactically Awesome Stylesheet*) je CSS preprocessorem, který rozšiřuje jeho syntaxi. Rozšiřuje jej o proměnné, podmínky, funkce aj. Zároveň tím nabízí lepší přehled a udržitelnost CSS stylů v projektu. Před použitím SASS stylů je nutné přeložit je do CSS, aby jej mohl webový prohlížeč vykreslit. Platí, že každý CSS kód je kompatibilní se SASS, avšak ne naopak.

7.1.5 Lodash

Komplexnější práce s daty je při implementaci vizualizací očekávaná a při manipulaci v JS se může stát repetitivní a nepřehledná. *Lodash* je elegantní JS knihovna řešící tento problém. Nabízí několik funkcionalit pro manipulaci s daty a jednoduché funkce pro každodenní transformaci dat (např. `groupBy()` kolekce podle hodnoty nebo `upperFirst()` pro kapitalizace řetězce).

7.1.6 Axios

Pro získání dat byla použita knihovna *Axios*. Knihovna je HTTP klientem pro webový prohlížeč a nabízí rozhraní pro jednoduché posílání požadavků a správu odpovědí. Podporuje standardní HTTP metody RESTful služby, kde nejdůležitějšími jsou `GET`, `POST`, `PUT`, `DELETE`, `PATCH`.

7.1.7 Mock Service Worker

Pro mockování dat byla využita knihovna *Mock Service Worker* (také jako MSW). Hlavním principem je využití Service Worker API, které funguje jako proxy mezi prohlížečem a sítí. MSW zaregistruje Service Worker, který naslouchá odchozím požadavkům aplikace, tyto požadavky přesměruje a MSW vrátí mockovanou odpověď. Díky tomuto principu je možné aplikaci dodat přímo s mockovanými daty ve vývojovém (*develop*) prostředí (více v sekci 7.3.1).

7.1.8 Storybook

Dokumentace komponent byla vytvořena pomocí *Storybook*. Tato knihovna byla vytvořena pro dokumentování komponent v MDX formátu a interaktivním prostředím. MDX kombinuje formáty Markdown (MD) pro běžnou dokumentaci ve formátovaném textu a JSX. Díky této kombinaci je dokumentování komponent aplikace elegantní a zároveň interaktivní s dostupnou ukázkou.

7.2 Architektura aplikace

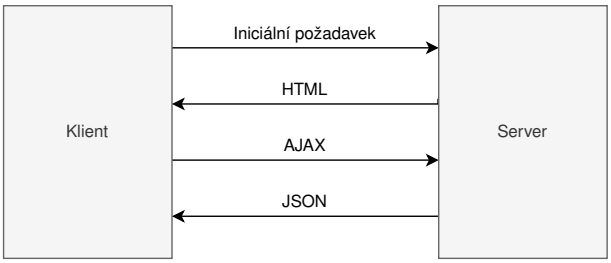
Pro tvorbu webové aplikace je k dispozici více možností architektur. Dále je užitečné zmínit strukturu samostatného projektu aplikace vytvořeného pomocí knihovny React.

7.2.1 Single-page application

Dříve se webová aplikace skládala z několika HTML dokumentů, které se generovaly na serveru a následně byly poslány klientovi pro vykreslení. Při každém přechodu na další stránku byl poslán nový požadavek na HTML dokument na server, což eventuálně může tvořit značnou neefektivitu. S příchodem moderních webových technologií se uchytilo řešení *single-page application* (SPA).

SPA je architektura pro webové aplikace, která načte jediný HTML dokument ze serveru a následně aktualizuje jeho tělo pomocí JS skriptů [55]. Při první návštěvě je stažena kostra stránky, styly a balíky JavaScriptu. Server pak posílá požadovaná data nové stránky, které se aktualizují v těle hlavního dokumentu. Jedná se o standard moderních webových stránek, proto je i implementace prototypu provedena pomocí této architektury. V rámci React aplikace je klientské směrování aplikace uskutečněno pomocí knihovny React Router.

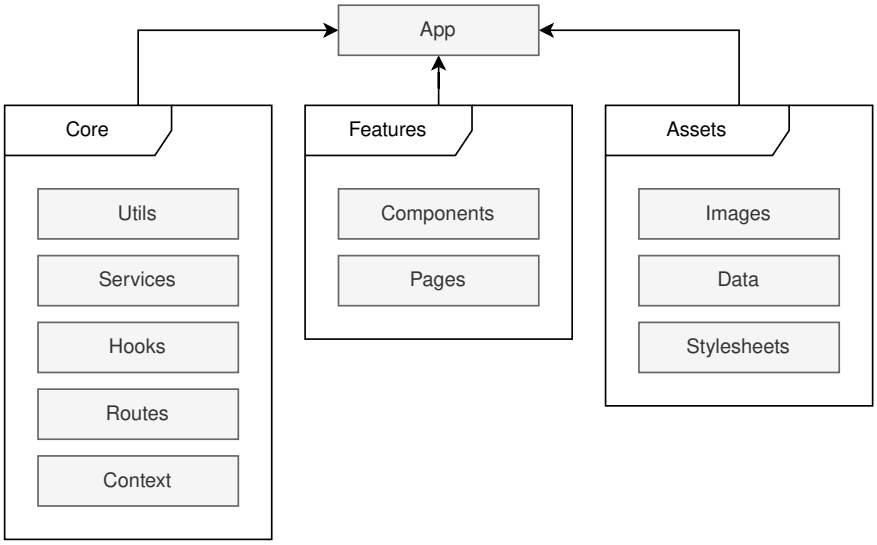
Zároveň bylo použito vykreslování na straně klienta (*client-side rendering*), které je vhodným kandidátem pro jednoduché aplikace. Efektivnější řešení naopak obsahuje vykreslování na straně serveru (*server-side rendering*, také jako SSR), které vygeneruje HTML dokumenty na serveru (na rozdíl od klienta).



Obrázek 7.1 Životní cyklus SPA aplikace

7.2.2 Struktura aplikace

Struktura aplikace je rozdělena do tří hlavních modulů, které obsahují implementované komponenty. Na nejvyšší úrovni se nachází hlavní komponenta `App`, která vykreslí hlavní dokument a importuje další moduly. Jádro aplikace tvoří `Core`, kde se nachází funkcionality dostupné napříč aplikací. Dalším modulem je `Features` modul, který obsahuje přepoužitelné komponenty a stránky aplikace. Poslední částí jsou `Assets`, které obsahují různé typy zdrojů (obrázky, data aj.). Strukturu lze vidět na obrázku 7.2.



Obrázek 7.2 Struktura prototypu aplikace

7.2.2.1 Core

Utils Utility obsahují různé pomocné funkcionality dostupné napříč aplikací. Jedná se o funkcionality pro zobrazení vizualizací.

Services Služby obsahují funkcionalitu pro volání API. Každý soubor exportuje funkce pro přepoužití přes celou aplikaci. Funkce asynchronně získají data pomocí knihovny *Axios*. Například `MapService` zajišťuje data pro znalostní mapu v semestru.

Hooks Hooks obsahují vlastní Hook procedury, které jsou přepoužitelné v jakékoliv funkcionální komponentě aplikace. Vlastní Hook definuje určité chování, které je repetitivní (např. fetchování dat). Jako vlastní Hook je definován `useD3` (viz. sekce 7.4).

Routes Klientské směrování aplikace je umožněno pomocí *React Router*. Díky této službě je možné navigovat mezi stavy a komponentami SPA. Modul obsahuje jednotlivé definice stránek a jejich obsahu. Tedy pod jakým URL se zobrazí daná komponenta. Ty jsou následně namapovány na `Route` komponentu.

Context Kontext byl navržen pro sdílení dat, které jsou k dispozici napříč celou aplikací nebo části aplikace. Ke kontextu mají přístup všechny jeho potomci. Přístup ke kontextu je pomocí jednoho z Hooků, konkrétně tedy `useContext`. Aplikace obsahuje jednoduchý kontext pro udržení stavu bočního panelu, vybrané entity a vybraného semestru.

7.2.2.2 Features

Components Komponenty obsahují podsložky s JSX, SCSS a popř. MDX soubory. Každý soubor obsahuje exportuje právě jednu komponentu. Komponenty jsou rozděleny dle typu jejich účelu. Hlavní komponenty vizualizace se nachází ve složce `visualisation` a jsou rozděleny do `CirclePacking`, `Knowledge`, `Timeline` a `NetworkCluster`.

Pages Zde lze nalézt komponenty reprezentující stránky aplikace. Aplikace je jednoduchá a neobsahuje mnoho stránek. Všechny stránky mají základ definovaný v komponentě `DefaultPage`.

7.2.2.3 Assets

Images Obsahuje obrázky, které byly použity při tvorbě domovské stránky, kde si uživatel může vybrat vizualizaci k zobrazení.

Data Obsahuje JSON data pro MSW mockování. Také jsou k dispozici některá JSON schémata, které byly použity k vygenerování dat.

Stylesheets Obsahuje moduly SCSS, které definují exporty a proměnné přepoužitelné v React komponentách nebo jiných SCSS souborech. Styly komponent jsou přítomné přímo v jejich složce ve `Features` nebo jsou vloženy inline.

7.3 Práce s daty

V této sekci jsou zmíněny veškeré informace o získání a formátu dat, s kterýma aplikace pracuje. Jelikož je aplikace zaměřena na vizualizace, tak je nezbytné pracovat s (alespoň trochu) smysluplnými daty. Struktura dat pro jednotlivé vizualizace je dále zmíněna v dokumentaci.

7.3.1 Mockování dat

Pro realizaci vizualizací bylo potřebné *mockování* dat, jinak řečeno vyvíjet a demonstrovat výsledky na falešných datech. Důvodem je ranný vývoj back-endové části aplikace v práci, kterou se zabývá Tochilin Illia ve své bakalářské práci ve stejném akademickém roce jako byla psána tato práce. Výhodou mockování je brzký vývoj front-endové části, která v budoucích iteracích projektu může rychleji a efektivněji navázat na back-endovou část. Naopak nevýhodou je nespolehlivost dat (struktura, konzistence) a složité testování na vizualizacích.

7.3.2 REST API

Pro získání dat ze serveru bylo využito REST (*Representational State Transfer*) API. V této fázi projektu je tvorba API v ranném vývoji, avšak bylo definováno použití REST. Jelikož je aplikace soustředěna na anonymního uživatele a zobrazení vizualizací, tak byla volána HTTP metoda `GET`. Aplikace interaguje s mockovaným API. Zároveň jsou využity dočasné URI (*Uniform Resource Identifier*) adresy, nebo také endpointy, na které jsou posílány požadavky.

7.3.3 Formát dat

Data z mockovaného serveru jsou přenášena ve formátu JSON (*JavaScript Object Notation*). Další typickou variantou pro vizualizace je CSV (*comma-separated values*) formát, avšak nenabízí objektovou strukturu jako JSON včetně hierarchie. Zároveň je JSON vhodnější pro zpracování ve webové aplikaci a vykreslení jiných dat nejen ty pro vizualizace, proto je vhodným kandidátem pro aplikaci.

7.4 Integrace D3.js s React

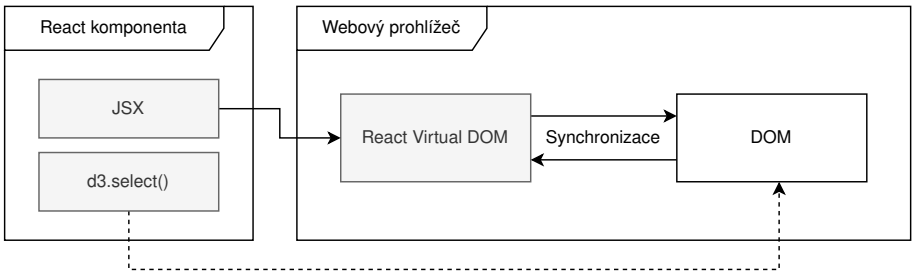
V dnešní době je vývoj web front-endu aplikace skoro neobejde bez využití moderních knihoven nebo frameworků jako je např. React, Vue.js nebo Angular. Proto je integrace D3.js a jakékoliv jiné JavaScript knihovny specializované na vizualizaci nevyhnutelná. Avšak při integraci dvou zcela různých knihoven můžou nastat různé komplikace a tím je třeba zvážit různé metody postupu.

Otázkou může být, proč se integrováním těchto dvou knihoven zabývat. První důvod byl již zmíněn výše, a tím je modernizace vývoje web front-endu. S tím úzce souvisí oddělení zodpovědností do jednotlivých komponent (např. v knihovně React). Zároveň JSX nabízí podporu pro `<svg>`, a tím lze deklarovat vizuální elementy přímo v React JSX formátu. React obecně nabízí mnoho funkcionalit pro lepší manipulaci a integraci (např. Hooks).

7.4.1 Problém s DOM

V sekci 7.1.2 byl nastíněn princip knihovny React, která využívá virtuální DOM pro vykreslování komponent. Virtuální DOM je reprezentace skutečného DOM dokumentu v paměti. Při detekované změně stavu komponenty je potřeba komponentu aktualizovat a znovu ji vykreslit v uživatelském rozhraní. Taková změna je nejdříve propsána do virtuálního DOM. Následně je vypočtena změna mezi starým a aktualizovaným virtuálním DOM. Teprve po zjištění rozdílů je změna propsána do skutečného DOM. Lze tedy říci, že React neinteraguje s DOM přímo, ale přes „proxy“ virtuální DOM. Díky tomuto principu je React schopný minimalizovat počet DOM operací nutné pro aktualizaci elementů v dokumentu. Stává se tak efektivním a výkonným řešením pro vývoj web front-end aplikace.

Naopak D3.js obsahuje API, které je založeno na připevnění dat ke skutečnému DOM. Pomocí D3.js je možné přidávat potomky a data nebo upravovat styly a atributy vizualizačních elementů. Již mohlo být napovězeno, kde spočívá problém při využití obou knihoven zároveň. Obě knihovny React a D3.js manipulují s DOM, avšak každá z jiného směru. React nejdříve vypočte změny pomocí virtuálního DOM a pak jej aplikuje na skutečný DOM, zatímco D3.js manipuluje s DOM přímo (viz. obrázek 7.3).



Obrázek 7.3 Problém s DOM při integraci D3.js a React

7.4.2 Metody integrování

V této sekci jsou popsány dvě hlavní metody pro integraci D3.js a React. První řešení dává větší zodpovědnost D3.js pro vykreslení vizualizačních elementů. Naopak druhé řešení se snaží využít vykreslování pomocí React všude, kde je to možné.

V obou metodách je (do určité míry) využit princip získání reference pomocí `useRef()`. Tento Hook udržuje selekci elementu pomocí `d3.select()`. Typicky je vybrán nejvíce horní element, který se má vykreslit. Metoda vrátí selekci elementu a ten je vykreslen v `renderFunc()` funkci. Jelikož je potřeba překreslit vizualizaci vždy, kdy jsou změněna data, tak je vykreslení obaleno v

`useLayoutEffect()`. Při jakékoliv změně v závislostech `dependencies` je vizualizace překreslena. Dohromady byl vytvořen univerzální Hook `useD3()`, který tuto logiku obaluje (viz. kód 7.1).

Výpis kódu 7.1 Custom Hook `useD3()`

```
export const useD3 = (renderFunc, dependencies) => {
    const ref = useRef(null)
    useLayoutEffect(() => {
        if(ref)
            renderFunc(d3.select(ref.current))
        return () => {}
    }, dependencies)

    return ref
}
```

7.4.2.1 Využití D3.js pro vykreslení

První metoda je tou jednodušší variantou. Hlavním principem je manipulace s DOM pomocí D3.js, tedy přidávání a úprava elementů přes D3.js API. Element je opět vybrán přes `d3.select()` a následně je pomocí řetězení metod upravován. Například metoda `append('circle')` přidá danému elementu `<circle>` element.

Samotná vizualizace je tedy tvořena řetězením metod D3.js API v JavaScriptu. Nyní je potřeba část pro integraci s knihovnou React. V JSX je definován `<svg>` element sloužící jako kontejner vizualizace, kterému budou přidávány další elementy. Pro přístup k tomuto elementu je potřeba mít jeho referenci. K tomu je vhodné přepoužít `useD3()` Hook, který byl popsán výše. Jako vykreslovací funkci je definován D3.js kód, který manipuluje s DOM přímo (viz. výše). Hook vrátí referenci na selekci, v tomto případě na `svg` a ta je propojena atributem `ref` v JSX (viz. kód 7.2). Typicky lze takto obalit celou vizualizaci a následně zajistit synchronizaci kontejneru `<svg>` pomocí Reactu.

Výpis kódu 7.2 Ukázka integrace D3.js a React pomocí `useD3()` a `<svg>`

```
const Node = ({ node }) => {
    const ref = useD3(
        (svg) => {
            svg.append('circle')
            .datum(node)
            .attr('r', d => d.r)
            .attr('cx', d => d.x)
            .attr('cy', d => d.y)
        }, [node])

     return (
        <svg ref={ref}></svg>
     )
}
```

7.4.2.2 Využití JSX pro vykreslení

Druhá metoda se zaměřuje na častější využití JSX. V JSX lze definovat i jiné `<svg>` elementy jako např. `<circle>` nebo `<path>`. Pomocí definování těchto značek v JSX lze elementy vykreslit přes virtuální DOM. Jediným problémem je naplnění elementu daty, ke kterému později některé funkcionality a algoritmy D3.js přistupují (např. při tvorbě síťového grafu s force-directed rozložením). Pokud jsou takové funkcionality využity, lze opět použít `useD3()` na daný element.

Zde jsou data předána přes metodu `datum()` pro jeden element a `data()` pro více elementů (viz. kód 7.3).

Nevýhodou je složitá správná exekuce této metody. Na rozdíl od první metody výše, je třeba dbát na synchronizaci obzvlášť při kombinaci JSX a `d3.select()`. Tím vznikají problémy při vykreslování, kde React často přepisuje DOM, s kterým D3.js manipuluje. Výhodou je přehlednost kódu a rozšiřitelnost, neboť JSX lze elegantně rozdělit do React komponent.

Výpis kódu 7.3 Ukázka integrace D3.js a React pomocí `useD3()` a `<svg>`

```
const Node = ({ node }) => {
    const ref = useD3((g) => {
        g.datum(node)
    }, [node])

    return (
        <circle
          ref={ref}
          r={node.r}
          cx={node.x}
          cy={node.y}>
        </circle>
    )
}
```

7.5 Implementace vizualizací

V této sekci jsou stručně popsány postupy při implementaci vizualizací a různé funkcionality, které nabízí. V implementaci byly použity funkcionální React komponenty společně s druhou metodou integrace přes JSX.

V implementaci prototypu není přítomna aktivita entity (viz. sekce 6.5.2.7). Důvodem byla priorita ostatních navržených vizualizací a také fakt, že aktivita prezentovala modifikovaný znalostní graf entity, který byl implementován. Není přítomen ani detailní obsah postranního panelu entity. Také je třeba zmínit, že dané vizualizace nejsou optimalizované. Diskuze o možné optimalizaci je detailněji rozebrána v sekci 7.6).

Další obrázky a videa výsledné aplikace s vhodnou ukázkou lze nalézt v přílohách práce. Ukázky několika variant pro vizualizace jsou k dispozici v interaktivní dokumentaci (více o dokumentaci v sekci 7.7).

7.5.1 Přehled studia

Vizualizace byla implementována pro univerzální použití a vizualizaci hierarchii pomocí kruhového zanoření. Graf je inicializován v komponentně `CirclePackingGraph` a následně je vykreslen v komponentě `CirclePackingGraphView`. První komponenta se stará o dodatečné filtrování dat, které by je třeba oddělit od vykreslení.

Hierarchie Hierarchické rozložení je vytvořeno metodou `d3.pack()` pro kruhové zabalení. Metoda vypočítá umístění a velikost vrcholů v hierarchické struktuře, následně vrátí objekt hierarchie. Z objektu lze získat potomky metodou `descendants()`. Přes potomky se dále iteruje a vykresluje každý vrchol. Na základě těchto metod je vytvořen vlastní princip a vzhled.

Vrcholy Vrchol je dvojího typu - rodičem (komponenta `CirclePackingNode`) nebo listem (komponenta `CirclePackingLeaf`). Rodič je vykreslen jako kružnice obalující vrcholy listů. Barvy

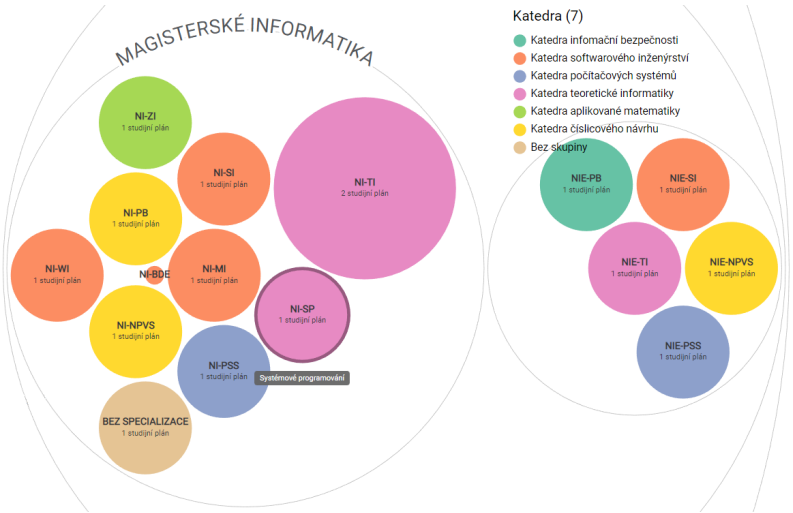
listů jsou vytvořeny pomocí škály `d3.scaleOrdinal()`, která je definována přes rozsah `range()` a doménu `domain()`.

Popisky Popisky vrcholů se liší pro rodiče a listy. V případě rodiče je popisek úrovně vykreslen podél kružnice pomocí elementu `<textPath>`. Je možné vykreslit buď popisek dané úrovně, nebo i přidat popisek nad úrovně (např. popisek „Magisterská Informatika“ je složen ze dvou úrovní). Popisky listů obsahují navíc sekundární popisek odhalují počet elementů v něm zapouzdřeným.

Animace Při výběru vrcholu kliknutím je provedena změna úrovně a tím i animace přiblížení nebo oddálení. Výběrem rodiče se provede oddálení, naopak výběrem listu se provede přiblížení. Při změně úrovně se aktualizují i barevné škály specifikované pro danou úroveň.

Legenda Každá úroveň obsahuje vlastní legendu, která je interaktivní a při přejetí myši zvýrazní dané skupiny. Legenda je vlastní univerzální komponentou `Legend`, která je použita i v dalších vizualizacích.

Na obrázku 7.4 lze vidět stránku s přehledem studia na úrovni specializací. Přehled studia zobrazuje hierarchii úrovně studia, programů, specializací a studijních plánů. Každá úroveň je navíc barevně rozdělena podle určité skupiny. Na úrovni studia je tím úroveň studia, na úrovni programů je tím jazyk programu, na úrovni specializace je tím zajišťovaná katedra a na úrovni studijního plánu je tím kombinovaná nebo prezenční forma. Ve vizualizaci vrcholů jako listů byly použity zkratky těchto entit, neboť dlouhé názvy byly nepřehledné. Plný název je zobrazen po přejetí myši nebo po zapouzdření do úrovně.



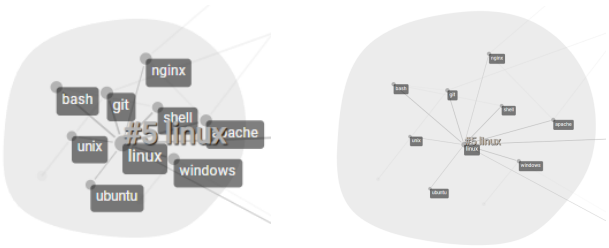
Obrázek 7.4 Prototyp: Přehled studia

7.5.2 Tématické shluky

Vizualizace byla implementována pro univerzální použití domény vrcholů, které mezi sebou navazují hranu pomocí dané spojitosti (např. dvě témata sdílí předmět nebo dva štítky se vyskytují ve stejném příspěvku) a následně je pro každý vrchol uveden shluk (komunita), do které patří. Ovlivňujícím vrcholem komunity je pak ten, který má největší stupeň v daném shluku. Graf je inicializován v komponentě `NetworkClusterGraph` a následně je vykreslen pomocí komponenty `NetworkClusterGraphView`. První komponenta se stará o dodatečné filtrování dat, které by je třeba oddělit od vykreslení.

Force-directed Pro rozmístění vrcholů síťového grafu bylo použito force-directed rozložení, které je inicializováno metodou `d3.forceSimulation()`. Rozložení lze dále rozvíjet řetězením metody `force()`, která nabízí několik variant. V implementaci byly použity varianty `link` (hrany), `charge` (přitažlivost vrcholů), `center` (přitažlivost ke středu) a `collide` (odpor bez kolizí).

Sémantické zvětšení Sémantické zvětšení umožňuje zanechat stejnou velikost elementů před i po přiblížení grafu, ale jejich pozice se změní. Pomocí metody `d3.zoom()` je inicializováno zvětšení. Řešením je aplikovat atribut přiblížení ze získaného objektu `transform` a lineárně interpolovat velikosti prvků přes `transform.k` atribut. Sémantické zvětšení je výhodné především pro grafy s velkým množstvím vrcholů, kde prozkoumání dané oblasti je klíčové (obrázek 7.5).



(a) Oddálený shluk (b) Přiblížený shluk

Obrázek 7.5 Prototyp: Sémantické zvětšení

Shluky Shluky jsou obaleny konvexní obálkou (komponenta `NetworkClusterHull`). Pomocí metody `d3.line().curve(d3.curveBasisClosed)` je inicializována uzavřená křivka. Křivce je třeba dodat data bodů konvexní obálky. Body obálky jsou vypočteny přes `d3.polygonHull()`. Nakonec jsou předány do atributu `d` elementu `<path>`. Shluky jsou rozlišeny barvou pomocí škály (podobně jako v předchozím grafu). Graf obsahuje funkcionalitu pro zapnutí a vypnutí shluků.

Vzhled grafu Vrcholy grafu mají odlišnou velikost podle jeho stupně. Podobně tak i hrany mají vizuální význam a mění svou tloušťku podle počtu hodnot na ní. Přejetím myši přes vrchol jsou zvýrazněni jeho sousedi s popisky. Podobně lze takto zobrazit i hrany. Různé stavy lze vidět na obrázku. 7.6.

Na obrázku 7.6 lze vidět příklad grafu Stack Overflow štítků. Štítky navazují spojitost pokud byly zmíněny ve společném příspěvku [56]. Data byly vhodným kandidátem pro realizaci této vizualizace, neboť obsahují již vypočtené shluky a zobrazují danou skutečnost (nejsou falešná). Nereálná data by mohly tvořit komplikace u testování prototypu a tím by i vizualizace ztrácela smysl. Později jej lze nahradit za tématické souvislosti v sylabu (např. vyučovaná témata v předmětu nebo klíčová slova předmětu). Tyto data nebylo možné namockovat, neboť je nutné získat smysluplné informace o shlucích (aby dané témata patřily správně k sobě).

Například vrcholy „Django“ a „PostgreSQL“ obsahují spojitost (hranu) v grafu štítků. Na této hraně je následovně zobrazen počet společných hodnot (v tomto případě příspěvků, ve kterých se společně vyskytovaly). Na obrázku lze také vidět, že nejvíce vlivným vrcholem shluku vrcholu „Django“ je vrchol „Python“. Ten pak následovně nese jméno celého shluku je zobrazen patřičným titulkem.

(a) Vypnuté shluky (b) Zapnuté shluky (c) Sousedé vrcholu

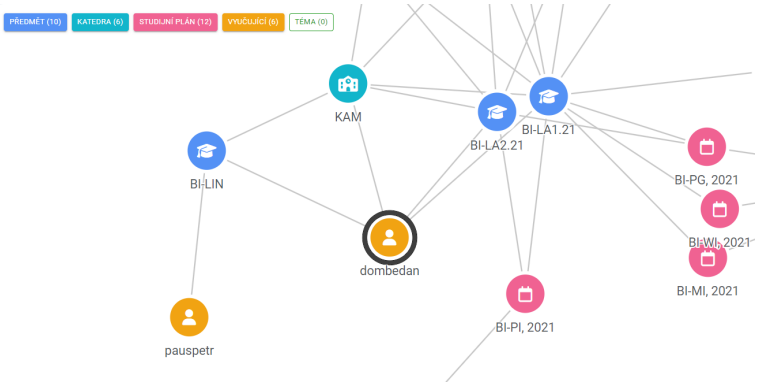
Obrázek 7.6 Prototyp: Pohledy tématických shluků

7.5.3 Znalostní mapa

Vizualizace byla inspirována znalostním grafem (např. grafové databáze). Byla navrhnuta pro univerzální zobrazení relací a entit různého typu. V zásadě lze graf rozdělit na dvě varianty - mapu a graf ve tvaru hvězdy pro přímé závislosti vybrané entity. Pro lepší přehlednost je doporučena varianta s rozšířenou entitou (hvězda). Mapa slouží pro celkový pohled a možnosti v ní vyhledávat. Graf je inicializován v komponentě `KnowledgeGraph` a následně je vykreslen pomocí komponenty `KnowledgeGraphView`. První komponenta se stará o dodatečné filtrování dat, které by je třeba oddělit od vykreslení.

Filtrování a hledání Znalostní grafy (mapu i rozšířenou entitu ve tvaru hvězdy) lze jednoduše filtrovat dle typu entity. Tlačítka pro filtrování slouží zároveň jako legenda pro orientaci v grafu (počet a barva daného typu). Hledání entity je umožněno pomocí vyhledávacího panelu (komponenta `EntityGroupSearch`), který seskupuje entity dle typu a nabízí automatické doplnění výsledků. Při výběru entity (hledáním nebo kliknutím na vrchol) se provede automatické animované přiblížení na vybraný vrchol. Na obrázku 7.7 lze vidět vybranou entitu v mapě s filtry.

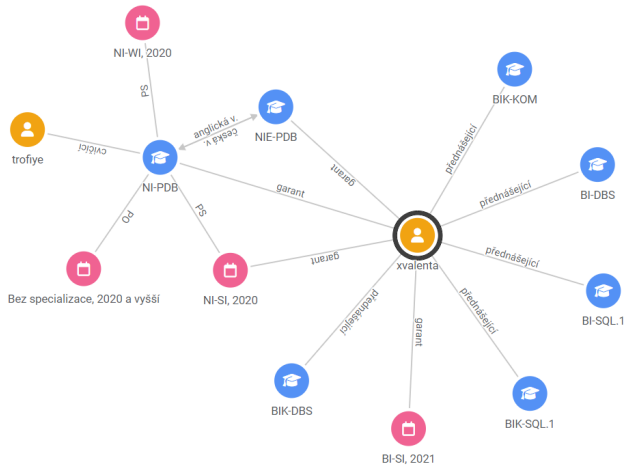
Panel entity Panel entity obsahuje hlavičku podle návrhu. Otevře se při každém výběru entity. Zároveň dočasně obsahuje tlačítka pro zobrazení závislostí a verzí entity, které uživatele přenesou do nové vizualizace.



Obrázek 7.7 Prototyp: Znalostní mapa s filtry a vybranou entitou dombedan

Rozšířená entita Graf používá force-directed rozložení (podobně jako výše). Pro rozšířenou verzi entity (hvězdy) je rozložení přizpůsobeno. Podoba hvězdy je vytvořena za pomocí `d3.forceCollide()`. V závislosti na počtu sousedů je také škálovaná vzdálenost hran pomocí metody `distance()` u `d3.forceLink()`. Pro toto nastavení je k dispozici i možnost rozšíření nebo kolapsování sousedů entity. Rozšíření doplní graf o další vrstvu sousedů daného vrcholu, kolapsování naopak zavře všechny rozšířené sousedy v podstromu. Na obrázku 7.8 lze vidět znalostní graf s rozšířenou entitou.

Vzhled grafu Ikony vrcholů jsou vykresleny pomocí `<text>`. Trik je v použití speciálního písma (Font Awesome), který zobrazí Unicode znaky jako vybrané ikony. Jedná se o pravděpodobně nejvíce elegantní a efektivní řešení pro vykreslení ikon uvnitř vrcholu. Šipka u hran je reprezentována komponentou `ArrowMarker`, která obsahuje `<marker>` v podobě parametrizované šipky. Tento marker je definován ve speciálním elementu `<defs>`, kde se později odkazuje jako reference. Graf nabízí možnosti pro zobrazení nebo skrytí šipek u cílového nebo zdrojového vrcholu. Text podél hrany je vytvořen pomocí `<textPath>`.



Obrázek 7.8 Prototyp: Znalostní graf entity NI-PDB s rozšířenou entitou xvalenta

7.5.4 Verze entity

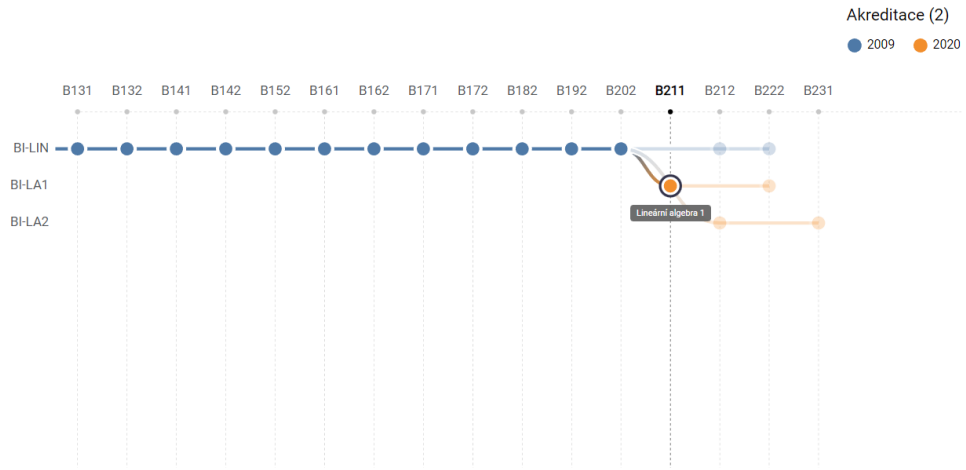
Vizualizace zobrazuje časovou osu a zaměřuje se na zobrazení verzí. Vrchol je vždy definován semestrem, jménem, skupinou a svými rodičovskými vrcholy. Tím lze zadefinovat předměty (vrcholy), které mají jak více rodičů, tak více potomků. Pokud je rodič stejnou entitou, tak se posune dál ve své horizontální ose. Jinak se osa rozvětví na další stupeň. Atribut skupiny určuje volitelné zařazení vrcholů do skupin, které jsou následně barevně odlišeny. Graf je inicializován v komponentě `TimelineGraph` a vykreslen v komponentně `TimelineGraphView`.

Osy Škálování os bylo vytvořeno pomocí `d3.scalePoint()` s danou doménou `domain()` a rozsahem `range()`. Obě dvě osy obsahují vlastní komponenty (`TimelineXAxis` a `TimelineYAxis`), které vykreslují vlastní přizpůsobenou osu. Pomocí táhnutí myši lze také graf posunout, ale osy se nepohybují přímo s vizualizací. Horizontální osa se pohybuje pouze horizontálně a vertikální pouze vertikálně (tedy obě jedním směrem). Popisky os tak zůstanou vždy viditelné a při přiblížení nebo oddálení grafu jsou rozsahy škál správně přemapovány.

Animace Při zobrazení grafu jsou hrany animovány od začátku časové osy postupně přes každý semestr až po poslední verzi. Funkcionalita je volitelná a lze nastavit i její rychlost. Animace byla vytvořena pomocí atributů `stroke-dasharray` a `stroke-dashoffset` elementu `<path>`.

Oříznutí grafu Aby se ve vizualizaci nepřekrývaly prvky při změně jejich polohy (viz. posun s grafem), tak mají oříznutý pohled pomocí komponenty `ClipPath`. Komponenta kombinuje `<clipPath>` a `<rect>` elementy a tím ořezává definovaný pohled. Opět se definuje v elementu `<defs>` a následně je v jiném elementu referencována. V tomto případě v obou osách a samostatné vizualizaci grafu, aby všechny elementy byly při změně pozice správně ořezány v pohledu.

Vzhled grafu Hrany obsahují barevný přechod (*gradient*) při spojení dvou vrcholů z odlišných skupin. Barevný přechod vytváří komponenta `LinearGradient`, která utilizuje lineární přechod elementu `<linearGradient>`. Opět se definuje v elementu `<defs>` a následně je v jiném elementu referencována. V tomto případně v atributu `stroke` elementu `<path>` pomocí kterého byla hrana mezi vrcholy vytvořena. Při přejetí myši na vrchol je zvýrazněna kompletní cesta, která do něj vede. Tím lze jednoduše pozorovat předchůdce dané verze. Pro hledání cesty byl použit jednoduchý algoritmus DFS (*depth-fist search*) s hledáním do hloubky. Kliknutím na vrchol je v postranním panelu zobrazen detail dané entity (podobně jako u znalostní mapy).



Obrázek 7.9 Prototyp: Verze předmětu BI-LA1

Na obrázku 7.9 lze vidět verze předmětu BI-LIN, BI-LA1 a BI-LA2 přes akreditace. Horizontální osa představuje semestry a vertikální osa všechny předměty, které se podílely na změně. V této situaci je vybraná verze BI-LA1 v semestru B211. Předchůdci této verze jsou jen verze předmětu BI-LIN, který je stále vypsán až do semestru B222, kde zanikne. Tato verze (BI-LA1) je také zařazena do nové akreditace 2020. Předešlá verze BI-LIN je zaražená v akreditaci 2009. Další entitou je BI-LA2, která se vyučuje v letním semestru akademického roku (na rozdíl od BI-LA1, která je v zimním). Tuto skutečnost lze také pozorovat na časové ose, kde BI-LA1 má vrcholy na přeskáčku s BI-LA2.

7.6 Optimalizace

Implementované vizualizace jsou vytvořeny bezeztrátovým formátem SVG. Tento formát nabízí kvalitu vizualizace, avšak není výkonný ve větším měřítku. S rostoucím počtem elementů se vykreslování zpomaluje obzvlášť ve webovém prohlížeči.

Proto je tato sekce věnována ideám, které by výkon vizualizace zlepšily. Účelem práce nebylo zaměřit se na optimální vizualizace s vysokým výkonem, ale věnovat se jejich obsahu a principu. Optimalizace je hlavně záležitostí budoucího vývoje a souvisí také s přehodnocením výběru technologie. Proto jsou zmíněny univerzální metody, které nejsou nutně závislé na knihovně React.

7.6.1 Vykreslovací metody

V kapitole 4.1 byly detailně rozebrány vykreslovací metody pro web front-end aplikace. Jednalo se o SVG, Canvas a WebGL. Zde již byly zmíněny veškeré výhody nevýhody těchto metod, avšak ohledně výkonu bylo nejlepším kandidátem WebGL, poté Canvas a nakonec SVG.

D3.js není stavěno na GPU vykreslování jako WebGL, avšak podporuje vykreslování pomocí HTML `<canvas>` elementu. React JSX také podporuje `<canvas>` a tím jej lze integrovat dohromady. Nevýhodou je fakt, že přidávání prvků do `<canvas>` lze provést jen přes jeho Context API. Tedy není možné použít JSX jako u `<svg>` elementu. Další nevýhodou je napojení dat, kde u `<svg>` lze použít jednoduchou metodu `data()`, avšak element `<canvas>` je bezstavový a tuto metodu nepodporuje. Proto je třeba vytvořit custom pomocný HTML element, do kterého již lze data vložit. Ukázku React JSX `<canvas>` elementu s Context API lze vidět v kódu 7.4. Kód je v podobném stylu jako byly prezentovány metody integrace v sekci 7.4.2.

Výpis kódu 7.4 Ukázka `<canvas>` elementu v D3.js a React pomocí `useD3()`

```
const Node = ({ node }) => {
    const ref = useD3(
        (canvas) => {
            var ctx = canvas.node().getContext('2d')
            ctx.beginPath()
            ctx.arc(node.x, node.y, node.r, 0, Math.PI * 2)
        }, [node])

     return (
        <canvas ref={ref}></canvas>
     )
}
```

7.6.2 Server-side rendering

Prototyp byl vytvořen jako SPA s vykreslováním na klientovi. Vizualizace jsou také vykresleny pomocí klienta a vzhledem k tomu, že se jedná o komplexnější a výkonně náročnější elementy, tak může způsobovat problémy s výkonem.

Pomocí vykreslování na serveru se výkon při vykreslování může značně zlepšit. SSR se od klientského vykreslování SPA liší tím, že veškeré komponenty předkreslí na serveru a následně pošle vygenerovaný HTML dokument klientovi (webovému prohlížeči). V podstatě jej posílá po částech a ne celý obsah při prvním načtení aplikace v prohlížeči. Princip SSR a D3.js je vygenerování vizualizace na serveru. Server následně pošle vygenerovaný dokument s vizualizací.

7.6.3 Integrace D3.js

V předchozí sekci byly zmíněny dvě hlavní metody pro integraci knihovny D3.js a React. Největší problém tvoří animace. Používání animací je s knihovnou D3.js jednoduché a většinou i vyžadované. Animace jsou prováděny při *update* fázi komponent, kdy je výhodnější použít D3.js. Pro React pak zbývá jen část *mount* a *unmount*. Také se lehce stává, že D3.js animace je přepsána aktualizací DOM z Reactu.

Proto je třeba zvážit metody integrace D3.js do React (nebo jiného frameworku pro tvorbu front-end webové aplikace). Zároveň je zcela jisté, že existuje i více variant těchto integračních metod a záleží na povaze aplikace a vizualizací, jakou metodu využít. Někdy může být výhodnější přenechat celé vykreslování na D3.js, které má své klady i zápory stejně jako ostatní metody.

Další variantou je pro implementaci využít nadstavby D3.js, které lze použít v Reactu. Není třeba řešit životní cyklus komponenty a její vykreslování a tím se eliminuje většina kódu. Výhodou je rychlé a jednoduché vytvoření vizualizace, neboť většina těchto knihoven potřebuje jen konfiguraci a data pro graf (podobně jako např. u G6.js, viz. 4.3). Naopak obrovskou nevýhodou je vlastní přizpůsobení, které je velmi omezené. Příkladem takové knihovny je např. *d3plus* [57], která nabízí wrapper pro React.

7.6.4 Formát dat

Poslední sekcí optimalizace je formát dat. Jedná se o víceméně jednoduchou variantu, jak rychleji optimalizovat výkon. V této práci je použit formát JSON, který je univerzálním standardem pro přenos dat z back-end části do front-end části aplikace. Zároveň v době psaní této práce není back-endová část přístupná a je v ranném vývoji, proto byla volba formátu víceméně otevřená.

Vizualizace mohou být někdy objemné a obsahovat velké množství dat. V závislosti na povaze vizualizace, je třeba zvážit využití jiného kompaktnějšího formátu. Typicky se jako alternativa používá CSV formát. CSV obsahuje hodnoty oddělené symbolem čárky a hraje roli 2D datového pole, proto je často používán společně s tabulkami. Výhodou CSV oproti JSON je kompaktnost a rychlejší zpracování velkého množství dat. Naopak nevýhodou je způsob zpracování, neboť v JavaScriptu je příjemnější práce s JSON objekty. JSON také nabízí hierarchickou strukturu kterou CSV neumí.

7.7 Dokumentace

Veškerá dokumentace je napsána v anglickém jazyce. Zdrojové kódy vizualizací jsou standardně dokumentovány pomocí JSDoc, z kterých lze následně vygenerovat HTML dokumentace kódu. Navíc je i možné lokálně spustit interaktivní dokumentaci pomocí nástroje Storybook, který obsahuje dokumentaci pro komponenty vizualizací.

Pomocí Storybook bylo vytvořeno celkově pět dokumentací (každá pro jeden typ vizualizace a legenda) a přes dvacet „stories“ (příběhů), kde každý příběh obsahuje interaktivní ukázku a možnost jednoduše nastavit nabízené parametry vizualizace. Není nutné lokálně spouštět celou aplikaci, pouze Storybook dokumentaci ve stejném repozitáři a vizualizace jsou k dispozici. Instrukce pro spuštění interaktivní dokumentace a aplikace na lokálním stroji jsou dostupné v příloze v dokumentu `README.md`.

Kapitola 8
Testování

Tato kapitola se zabývá testováním implementovaného prototypu společně s navrženým uživatelským rozhraním. Uživatelské testování bylo provedeno s potenciálními uživateli aplikace. Výsledkem je souhrn nedostatků a problémů uživatelského rozhraní, které byly odhaleny. Některé navržené řešení jsou implementovány v úpravě prototypu.

8.1 Uživatelské testování

Uživatelské testování je jednou z metod pro testování prototypů s uživateli. Tento termín je někdy nesprávně interpretován jako testování uživatelů, avšak tím není. Uživatelské testování se zabývá testováním uživatelského rozhraní a schopností uživatelů jej použít. Hlavním cílem je odhalení problémů v rozhraní a lepší porozumění uživatelům.

V této metodice hrají roli tři elementy - moderátor, úkoly (instrukce) a účastník (tester). Moderátor testování moderuje celý jeho průběh a předává účastníkovi úkoly. Ten se je následně pokusí splnit.

8.1.1 Pokrytí případů užití

Testovací scénáře byly postaveny na základě definovaných případů užití v kapitole 6.3. Některé případy užití nejsou přítomny, neboť nebyly implementovány v prototypu. Jedná se především o případy s aktéry jako vyučujícím (UC5 a UC6 ve znalostní mapě) a vizualizaci aktivit entity (UC3 a UC4 v pohledech entity). Pro připomenutí je uveden seznam pokrytých případů užití:

8.1.1.1 Přehled studia

- UC1: Zobrazení přehledu studia
- UC2: Odkrytí/zakrytí úrovně

8.1.1.2 Znalostní mapa

- UC1: Zobrazení znalostní mapy
- UC2: Zobrazení tématických shluků
- UC3: Filtrování mapy
- UC4: Hledání v mapě

8.1.1.3 Informace entity

- UC1: Zobrazení detailu entity

8.1.1.4 Pohledy entity

- UC1: Zobrazení znalostního grafu entity
- UC2: Rozšíření/kolapsování entity
- UC5: Zobrazení verzí entity
- UC6: Zobrazení předchůdců

8.1.2 Průběh testování

Prototyp aplikace byl otestován s mockovanými daty na zařízení s procesorem AMD Ryzen 5 5600H a RAM 16GB ve webovém prohlížeči Google Chrome. Data byly postaveny tak, aby co nejvíce odpovídaly reálné situaci. Celý průběh testování byl nahráván včetně kamery a záběru obrazovky. Testování probíhalo kontaktní formou. Moderátor testování postupně zadával testerům úkoly a následně pozoroval jejich průběh. V krajních situacích měl moderátor možnost zakročit a pokusit se testera navést na správné řešení. Po dokončení úkolů byl tester požádán o vyplnění dodatečného dotazníku.

Dle teorie J. Nielsena je nejúčinnější provádět uživatelské testování s pěti testery, neboť většina UI/UX problémů (okolo 75%) rozhraní je nalezena po testování pouhých pěti uživatelů [58]. Proto implementovaný prototyp otestovali celkově čtyři testeři reprezentující skupinu potenciálních uživatelů. Všichni testeři byli studenti vysoké školy a nikdy nepracovali s podobnou vizualizační aplikací. Odpovídají tedy cílové skupině uživatelů.

8.1.3 Testovací scénáře

Každý tester prošel celkem osm testovacích scénářů, každý z nich zaměřený na použití jedné vizualizace. Při testování se moderátor řídil dle testovacího scénáře. Tester obdržel jen část tohoto scénáře, kde byly instrukce pro vyplnění úkolu. V následující části je uveden příklad plné formy testovacího scénáře. Ostatní scénáře lze nalézt v přílohách této práce. Instrukce pro každý úkol jsou uvedeny v sekci 8.1.4, kde je také celkové vyhodnocení testování.

8.1.3.1 Ukázka scénáře

Název
T4: Filtrování ve znalostní mapě

Odhadovaný čas
2 min.

Účel testování
Testování interakce se znalostní mapou a postranním panelem. Uživatel by měl být schopný filtrovat mapu a efektivně zobrazit entity a její detail.

Pokrytí případů užití
Znalostní mapa - UC1, UC3
Informace entity - UC1

Počáteční bod
Uživatel se nachází v aplikaci.

Koncový bod
Uživatel se nachází ve stránce studijního plánu v Bílé knize.

Instrukce pro testera
V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny studijní plány, které nejsou zajišťovány žádnou katedrou. Vyberte vhodnou vizualizaci, vyberte jeden z těchto plánů a zobrazte si jeho obsah v Bílé knize.

Očekávané kroky

1. Zobrazení vizualizace znalostní mapy
2. Vybrání semestru B212 v mapě
3. Filtrování mapy na entity (studijní plán a katedra)
4. Vybrání entity studijního plánu v mapě (Bez specializace, 2020 a vyšší)
5. Vybrání odkazu Bílé knihy v postranním panelu entity

8.1.4 Vyhodnocení

Nejdříve je uveden krátký přehled testerů. V písemné části této práce bude jejich identita anonymizována. Nadále budou testeři ve vyhodnocení úkolů referováni jako tester A, B, C a D. Jak již bylo zmíněno, všichni testeři jsou studenti a nemají předešlé zkušenosti s podobnou vizualizační nebo grafovou aplikací. Přehled testerů lze vidět v tabulce 8.1.

Tabulka 8.1 Přehled testerů, kteří se zúčastnili uživatelského testování.

Tester	Graf. aplikace	Student	Ročník studia
A	Ne	ČVUT FIT	5
B	Ne	ČVUT FIT	5
C	Ne	ČVUT FIT	5
D	Ne	VŠCHT Ekonomie a Management	1

Scénáře T1 a T2 se věnovaly vizualizaci přehledu studia. Scénáře T3 a T4 se věnovaly tématickým okruhům. Scénáře T5 až T7 se věnovaly znalostní mapě. Poslední scénář T8 se věnoval verzím entity. Uživatelské rozhraní bylo vyhodnoceno na základě následující stupnice:

100 % Testující byl schopen projít celým testovacím scénářem bez výhrad.

75 % Testující prošel testovacím scénářem s drobnou výpomocí moderátora.

50 % Testující prošel testovacím scénářem se častou pomocí moderátora.

25 % Testující prošel testovacím scénářem s krokovými instrukcemi od moderátora.

0 % Testující nebyl schopen dokončit scénář.

Následuje vyhodnocení jednotlivých úkolů ze scénářů. Procentuální ohodnocení je vypočítáno jako průměr hodnocení všech otestování. Také jsou zmíněny poznatky z pozorování a dotazníku.

8.1.4.1 Scénář T1

Právě jste dokončili bakalářské studium na FIT ČVUT a máte v plánu pokračovat dál v magisterském studiu. Ve svém bakalářském studiu jste studovali v českém jazyce a prezenční formě specializaci Umělá inteligence (zkratka BI-UI21). Pokud by to bylo možné, chtěli byste pokračovat ve stejné specializaci i na magisterském studiu (čeština), nebo alespoň ve nejvíce podobné specializaci, např. ve stejné katedře. Vyberte vhodnou vizualizaci a v ní tuto specializaci.

Složitost Vysoká

Hodnocení výsledku $(0 + 75 + 75 + 50) \ / \ 4 = 50\ \%$

Testeři byli trochu zmateni ze zadání tohoto scénáře. Souvislosti hledali jinde, než byl scénář zamýšlen. Tester A hledal specializaci rovnou na magisterské úrovni studia a byl zmaten ohledně jeho cíle. Tester C hledal v jiných vizualizacích, což také nebylo zamýšleno.

Rozdělení specializací do kateder podle barevné škály nebylo všem testerům intuitivní a nemohli odvodit, do jaké katedry specializace patří. Tester D očekával další úroveň zanoření podle katedry, kde by následně našel specializace, které pod ní spadají. Tester B si nejdříve také nepovšiml tohoto rozdělení, avšak nakonec došel ke správnému řešení. Tester C hledal podobnost ve stavbě předmětů.

Hlavní problém scénáře spočíval v zadání, které bylo lehce zavádějící a proto nebyl scénář úspěšný. Samostatné rozhraní vizualizace bylo dostačující, avšak navigace podle barev nebyla vždy intuitivní (obzvlášť na úrovni specializací).

8.1.4.2 Scénář T2

Studujete magisterské studium na FIT ČVUT v češtině. Právě studujete specializaci Počítačová bezpečnost (zkratka NI-PB). Váš kolega by chtěl studovat stejnou specializaci, avšak neumí češtinu a proto je nyní zapsán na magisterském studiu v angličtině. Zjistěte, zda může studovat tuto specializaci ve svém programu a popřípadě mu doporučte podobnou specializaci.

Složitost Lehká

Hodnocení výsledku $(100 + 100 + 100 + 100) \ / \ 4 = 100\ \%$

Scénář byl zaměřen na stejnou vizualizaci jako předchozí. Testeři byli již mírně obeznámeni s principem vizualizace a orientovali se lépe. Testeři neměli problém najít sesterskou specializaci v anglické verzi programu a proto byl scénář úspěšný.

8.1.4.3 Scénář T3

Po prvním semestru na FIT ČVUT jste se začali zajímat o programovací jazyk C++. Avšak jediný co víte je, že toto téma by mohlo mít souvislost s jazykem C# nebo Python. Vyberte vhodnou vizualizaci a najděte v ní téma C++ společně s jeho nejbližšími tématy.

Složitost Střední

Hodnocení výsledku $(100 + 0 + 75 + 25) \ / \ 4 = 50\ \%$

Někteří testeři byli zmateni principem vizualizace. Tester B hledal nejbližší témata jako předměty a zaměňoval tyto dva termíny. Lehce zmaten byl i tester C, který si nebyl jist rozdílu mezi těmito pojmy v grafu.

Testeři B a D také očekávali, že hodnoty hran půjdou zobrazit na kliknutí. Tester D očekával propojenost mezi hodnotami v panelu a samotnou vizualizací.

Problémy tedy tvořil princip vizualizace a nejasnost či zaměnitelnost pojmů téma a předmět. Testeři se také často odkazovali na dodatečný panel, jelikož vizualizace byla moc obsáhlá. Zmatení také působilo ve faktu, že graf obsahoval pouze technologie a odkazoval se na ně jako na témata.

8.1.4.4 Scénář T4

Zajímáte se o témata, které mají souvislost s tématem Hibernate. Abyste rozšířily své obzory, chtěli byste najít takové témata, které stále mají souvislost s okruhem Hibernate, ale i s jinými tématickými okruhy. Z nich vyberte takové téma, které má nejvíce společných předmětů (hodnot) s tématem z jiného okruhu. Dále zjistěte o jaké předměty se jedná.

Složitost Vysoká

Hodnocení výsledku $(50 + 0 + 100 + 0) \ / \ 4 = 37{,}5\ \%$

Problém byl podobný jako v předchozím scénáři a to v principu vizualizace a zaměnitelnosti pojmů téma a předmět. Také tvořily problémy hodnoty (předměty) na hranách. Někteří testeři si mysleli, že jedna hrana znamená jedna hodnota.

Testeři A si nepovšimli možnosti pro zobrazení shluků v grafu. Tester A nejdříve nepochopil princip hodnot hrany, avšak po nápovědě a ujasnění byl schopen najít dané předměty. Tester B hledal vrchol místo okruhu, který měl stejné pojmenování. Tester C pochopil princip vizualizace a se splněním úkolu neměl problémy.

Kombinace nelehkého zadání, problémy s vizualizací a pojmy vytvořila složitý úkol pro testery. Scénář nebyl tedy tak úspěšný, jak bylo očekáváno.

8.1.4.5 Scénář T5

V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny studijní plány, které nejsou zajišťovány žádnou katedrou. Vyberte vhodnou vizualizaci, vyberte jeden z těchto plánů a zobrazte si jeho obsah v Bílé knize.

Složitost Lehká

Hodnocení výsledku $(100 + 100 + 100 + 100) \ / \ 4 = 100\ \%$

Scénář byl úvodním scénářem pro vizualizaci znalostní mapy. Testeři správně aplikovali filtr na mapu a pochopily návaznosti pomocí hran. Testeři byli schopni zobrazit Bílou knihu pomocí postranního panelu. Testeři neměli problém splnit scénář a tím je scénář považován za úspěšný.

8.1.4.6 Scénář T6

V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny předměty, které mají jako garanta Michala Valentu. Vyberte vhodnou vizualizaci a vypište tyto předměty.

Složitost Střední

Hodnocení výsledku $(75 + 100 + 100 + 75) \ / \ 4 = 87{,}5\ \%$

Tento scénář měl podobný účel jako předchozí scénář. Navíc si měli testeři zobrazit přímé závislosti a popisky hran, aby zjistili jaký vyučující hraje roli garanta. Někteří testeři využili mapu pro zjištění závislosti místo funkce pro přímé závislosti. Testeři A a D si nepovšimli možnosti pro zobrazení popisků hran a očekávali, že budou automaticky zapnuté.

8.1.4.7 Scénář T7

V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny závislosti předmětu Lineární algebra. Poté Vás zajímá, jaké další předměty vyučuje garant tohoto předmětu. Vyberte vhodnou vizualizaci a zobrazte zmíněné závislosti.

Složitost Střední

Hodnocení výsledku (75 + 100 + 100 + 100) / 4 = 93,75 %

Tento scénář měl podobný účel jako předchozí scénář. Navíc měli testeři použít rozšiřování entit v grafu, aby se efektivněji dostali ke zmíněné závislosti. Tester A si nepovšiml možnosti pro zobrazení popisků a nejdříve hledal, který z vyučujících je garantem pomocí externích odkazů z postranního panelu.

8.1.4.8 Scénář T8

S nástupem nové akreditace byly provedeny změny v předmětu Lineární algebra. Zajímá Vás, jaký byl vývoj verzí této entity. Zjistěte v jakém semestru (popř. semestrech) nastala změna a o jakou změnu se jednalo (jaké předměty se podílely na této změně).

Složitost Lehká

Hodnocení výsledku (100 + 100 + 100 + 75) / 4 = 93,75 %

Scénář byl zaměřen na přechod ze znalostní mapy do grafu verzí entity a jeho navigaci v něm. Testeři neměli problém najít správnou entitu a zobrazit si verze dané entity. Testeři pochopili princip vizualizace a byli schopni správně odvodit informace z grafu. Tester D byl lehce zmaten z barevné legendy, která obsahovala akreditace s popiskami jejich roků jako názvů. Tester D neznal zkratky předmětů.

8.1.4.9 Shrnutí výsledků

V následující tabulce 8.2 lze vidět shrnutí vyhodnocení výsledků testování pro každý scénář. Jak lze vypozorovat z výsledků, největší problémy tvořily scénáře T1, T3 a T4.

Testeři byli prvním scénářem poněkud zmateni, protože byl lehce zavádějící. Avšak problém byl především v barevné rozlišitelnosti specializací podle katedry, který některé testery zarazil. Ostatní funkcionality a rozhraní byly ve vizualizaci uspokojivé a testerům přišla relativně jednoduchá i užitečná.

Scénáře T3 a T4 byly zaměřeny na vizualizaci tématických shluků a tvořily největší problémy v uživatelském testování. Testeři měli problémy s principem grafu a významu hodnot. Často se odkazovali na pomocný panel s daty, ve kterém preferovali hledání na rozdíl od vizualizace. Zároveň namockovaná testovací data nebyla nejvhodnější pro tématické okruhy v sylabu, protože vyjadřovala technologie. Samostatná idea shluků s tématy byla oceněna a byla ohodnocena jako užitečná pro studenty.

Nejúspěšnější byly scénáře T5, T6 a T7 pro znalostní mapu a T8 pro verze entity. Znalostní mapa byla intuitivní a nabízela prostředky pro rychlé dohledání entity. Také byl oceněn postranní panel s dodatečnými informacemi. Mapa měla drobné nedostatky, např. vypnuté popisky relací. Celkově byla ohodnocena velmi pozitivně a jako přínosná pro studenty.

Verze entity také byly intuitivní a každý tester pochopil význam dat v grafu. Zde bylo lehce matoucí zabarvení grafu dle akreditací, kde jednotlivé akreditace byly popsány rokem. Avšak testeři ohodnotili tuto vizualizaci jako neužitečnou pro roli studenta.

Tabulka 8.2 Vyhodnocení uživatelského rozhraní pro osm testovacích scénářů T1 až T8.

T1	T2	T3	T4	T5	T6	T7	T8
50 %	100 %	50 %	37,5 %	100 %	87,5 %	93,75 %	93,75 %

Byla vytvořena i tabulka nalezených problémů v rozhraní a funkčnosti aplikace, viz. 8.3. Problémy mají přidělenou závažnost na stupnici od 1 (nezávažné) po 5 (nejvíce závažné). U vybraných problémů je provedena úprava prototypu.

Tabulka 8.3 Odhalené problémy z uživatelského testování pro osm testovacích scénářů T1 až T8.

#	Problém	Scénář	Testeři	Závažnost	Návrh řešení	Implementováno
1	Barevné rozlišení jako rozdělení kateder v přehledu studia	T1	D, B	3	Přehodnocení významu barev	✗
2	Princip grafu tématických shluků a pojmů v něm	T3	A, B, C, D	5	Přehodnocení pojmů, zapojení postranního panelu, smazání nepotřebných informací	✗
3	Panel tématických shluků není interaktivně propojen s grafem	T4	A, B, C, D	5	Panel je nahrazen interaktivní legendou	✓
4	Vyhledávání v tématických shlucích není efektivní	T4	A, B, C, D	5	Vyhledávání je uskutečněno pomocí vyhledávacího panelu	✗
5	Graf tématických shluků je nekontrolovatelně v pohybu	T4	D	4	Přidání možnosti vypnutí a zapnutí pohybu s grafem	✓
6	Neviditelný vyhledávací panel	T5	C	2	Přestylizovat vyhledávací panel	✓
7	Vypnuté popisky u relací znalostního grafu	T6	A, D	3	Automaticky zapnuté popisky, zapnutí při přejetí myši přes hranu	✓
8	Nepřehledné zobrazení relací ve znalostním grafu	T6	D	3	Zvýraznění hrany při vybrání vrcholu	✓
9	Barevné rozlišení akreditací ve verzích	T8	D	2	Varianta grafu bez rozdělení do skupin	✓
10	Graf verzí nelze přiblížit	T8	C	3	Varianta grafu s možností přiblížení a oddálení	✓
11	Hrana reprezentující změnu ve verzích není intuitivní	T8	C, D	2	Přidání popisku pro hrany, které přesněji popisují danou akci nebo změnu	✗

Kapitola 9
Závěr

Diplomová práce se zabývá tvorbou vizualizací pro několik pohledů studia na FIT ČVUT. Hlavním cílem bylo navrhnout a implementovat prototyp web front-end aplikace, která tyto vizualizace následně realizuje. V návrhu bylo prioritou zaměřit se na tématické souvislosti předmětů, na pohledy specializací/oborů a na verze předmětů přes akreditace (v čase).

Práce se rozděluje na dvě části - teoretickou a praktickou část. První část práce je zaměřena teoretickou část a obsahuje rešerši problémové domény sylabu, vizualizací, současných řešení vizualizačních aplikací a knihoven pro vývoj web front-end aplikace.

V analýze sylabu byly identifikovány nejdůležitější aspekty a entity studia z primárního zdroje Bílé knihy ČVUT. Také byly analyzovány ostatní systémy, které fakulta spravuje nebo využívá. Důležité byly poznatky ohledně přechodu akreditace ze staré na nové, které inspirovaly následný návrh vizualizace verzí předmětů.

Po analýze problémové domény se práce zaměřuje na rešerši současných řešení vizualizací. Zde je kladen důraz na síťové grafy a hierarchie, neboť se jedná o komplexní vizualizace se záměrem pro kvalitativní data (na rozdíl např. od spojnicových grafů, které jsou více kvantitativní) a ty jsou pro účely práce vyžadovány. Dále je detailně probrán síťový graf včetně jeho rozložení, metodik vzhledu a některých algoritmů (např. detekce komunit). Nakonec jsou probrány možnosti hierarchické vizualizace, které později hrají roli v návrhu aplikace.

Analýza současných řešení vizualizačních aplikací obsahuje vybrané webové aplikace, které pracují s vizualizací jako s primárním zdrojem. Zde je důležité zaměření na jejich účel, způsoby vizualizace a uživatelského rozhraní. Poznatky z této kapitoly jsou následně zužitkovaný v návrhu aplikace.

Jako poslední z teoretické části je rešerše vizualizačních knihoven pro web front-end aplikace. Bylo potřeba vybrat správnou knihovnu na základě několika faktorů. Nejdříve byly zmíněny různé vykreslovací metody, které webové prohlížeče nabízejí. Jedná se o SVG, HTML5 Canvas a WebGL. Jako výsledná knihovna byla vybrána D3.js s možností implementace pro SVG nebo Canvas, která umožnila velmi široké vlastní přizpůsobení při tvorbě vizualizací.

Praktická část začíná návrhem aplikace s vizualizacemi. Nejdříve byly specifikovány požadavky a vytvořeny případy užití. Návrh se zaměřoval primárně na anonymního uživatele aplikace (podobně jako je tomu v Bílé knize). Dále byl navržen princip vizualizací a následně jejich uživatelské rozhraní. Výsledkem je primárně drátěný model, který popisuje jednotlivé interaktivní vizualizace.

Vytvořený návrh byl realizován formou SPA prototypu. Pro vývoj jádra web front-end aplikace byla použita knihovna React. V implementaci byla provedena integrace knihoven D3.js a React. Následně byly vytvořeny komponenty vizualizací (a ostatních UI elementů), ze kterých byly sestaveny jednotlivé části aplikace. Kapitola také obsahuje diskuzi o možné optimalizaci vizualizací do budoucího vývoje.

Výsledný prototyp byl otestován a podroben uživatelskému testování. Testeři představovali potenciální uživatele aplikace a tím byli i vhodnými kandidáty. Testování bylo velice přínosné a odhalilo několik nedostatků v návrhu a implementaci vizualizací. Celkově se ukázalo, že aplikace splňuje požadavky uživatelů v dostatečné míře a celková idea vizualizací byla podpořena. Veškeré problémy a poznatky z testování byly sepsány do výsledné tabulky. Největší problém tvořila vizualizace tématických shluků, kde se vyskytlo zmatení ohledně terminologie a testeři měli problém s vyhledáváním v grafu. Naopak nejúspěšnější byla znalostní mapa a verze entity.

Budoucí vývoj aplikace vyžaduje přehodnocení vykreslovací metody pro znalostní mapu, neboť se jedná o výkonnostně nejnáročnější vizualizaci. Zároveň se jedná o rozsáhle používanou formu vizualizace, tedy je možné jí nahradit již implementovanou variantou. Tím se ovšem ztratí možnost vlastního přizpůsobení funkcionalit a vzhledu grafu. Ostatní vizualizace jsou pro SVG vykreslení dostatečné pro typický objem dat.

Práce splnila veškeré požadavky a naplnila očekávání, které v úvodu měla. Výsledek zahrnuje všechny typy požadovaných vizualizací, které jsou demonstrovány na vhodných ukázkách v kapitole zabývající se implementací. Vizualizace jsou přepoužitelné, ale i otevřené pro pokračování v budoucím vývoji tohoto projektu.

Příloha A

Drátěný model

Interaktivní drátěný model je také k dispozici na médiu ve složce `wireframes`.

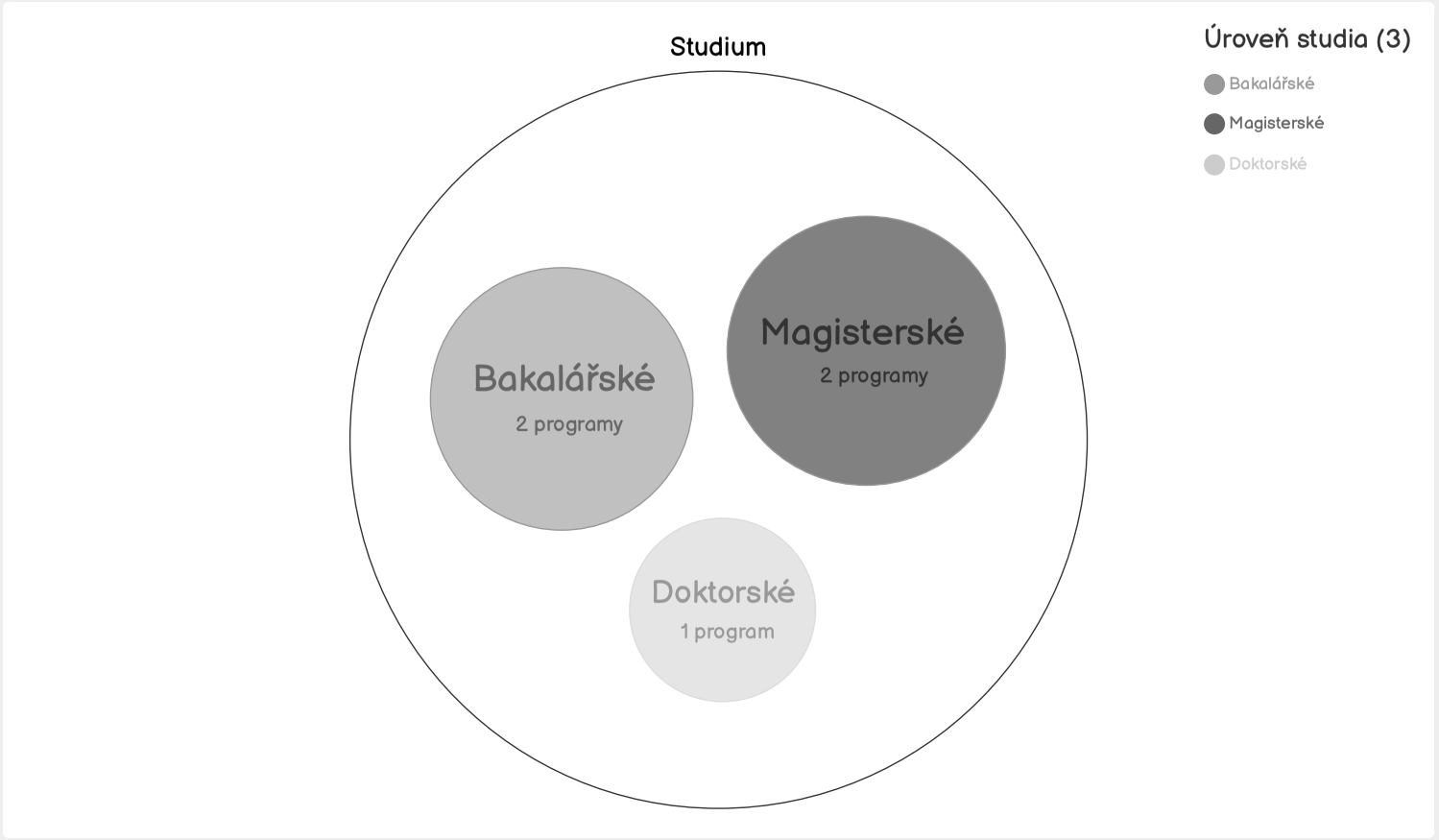
Domů

Mapa

Akreditace

Studium FIT

B123



Vyberte entitu

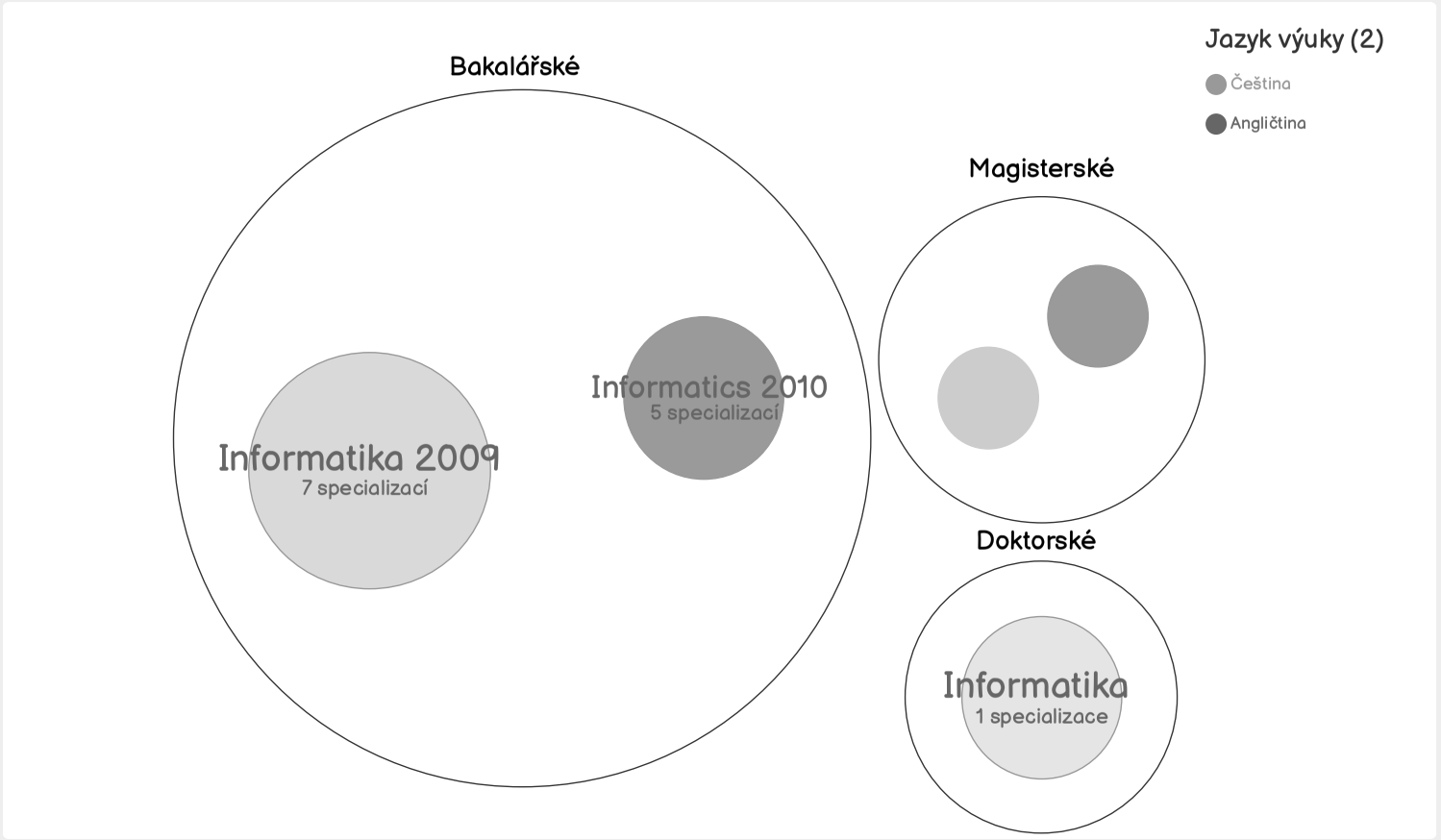
Domů

Mapa

Akreditace

Bakalářské studium

B123



Vyberte entitu

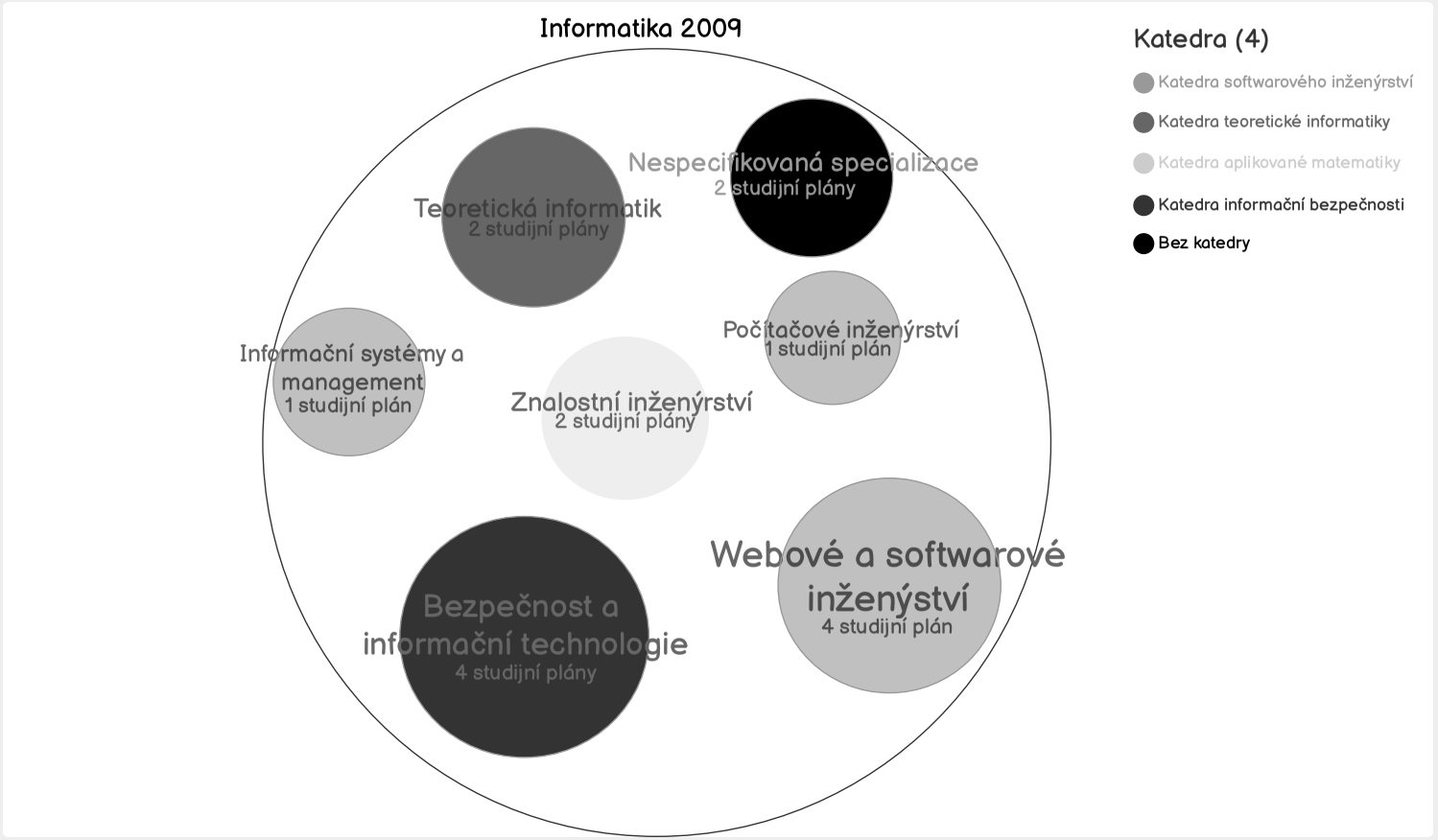
Domů

Mapa

Akreditace

Informatika 2009

B123



Vyberte entitu

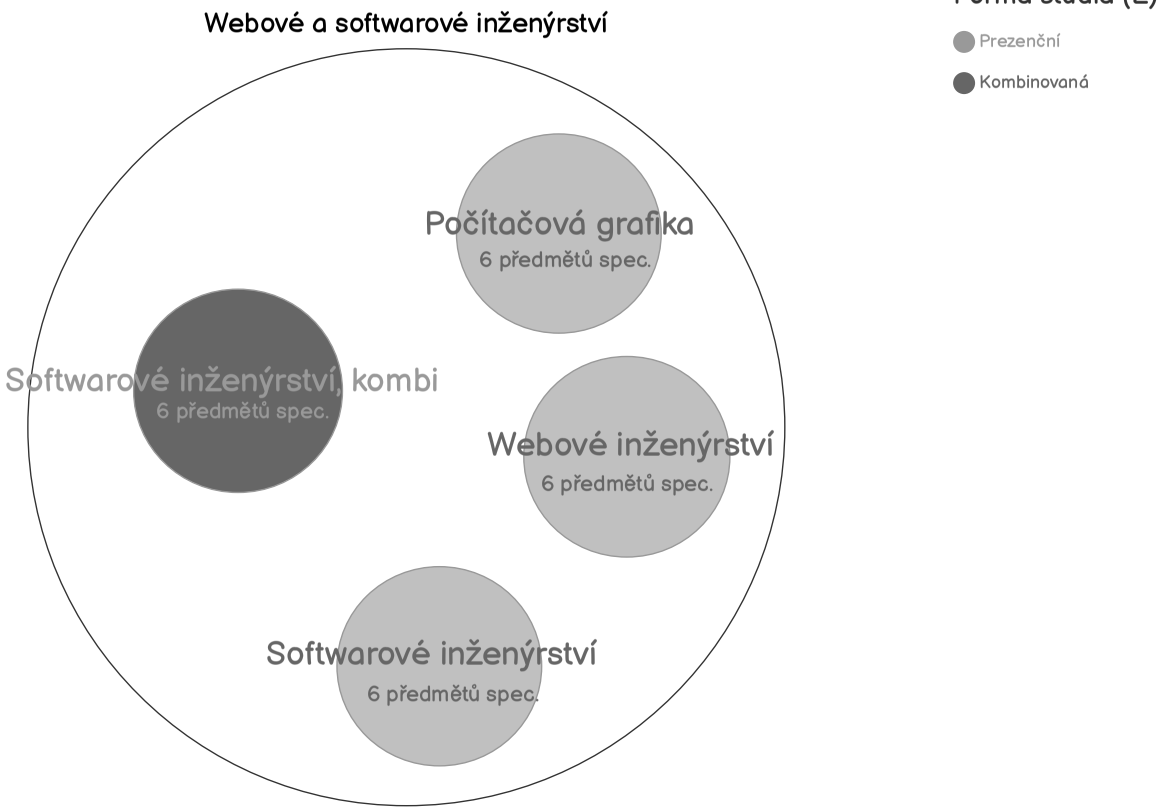
Domů

Mapa

Akreditace

Webové a softwarové inženýrství

B123



Vyberte entitu

Domů

Mapa

Akreditace

Znalostní mapa

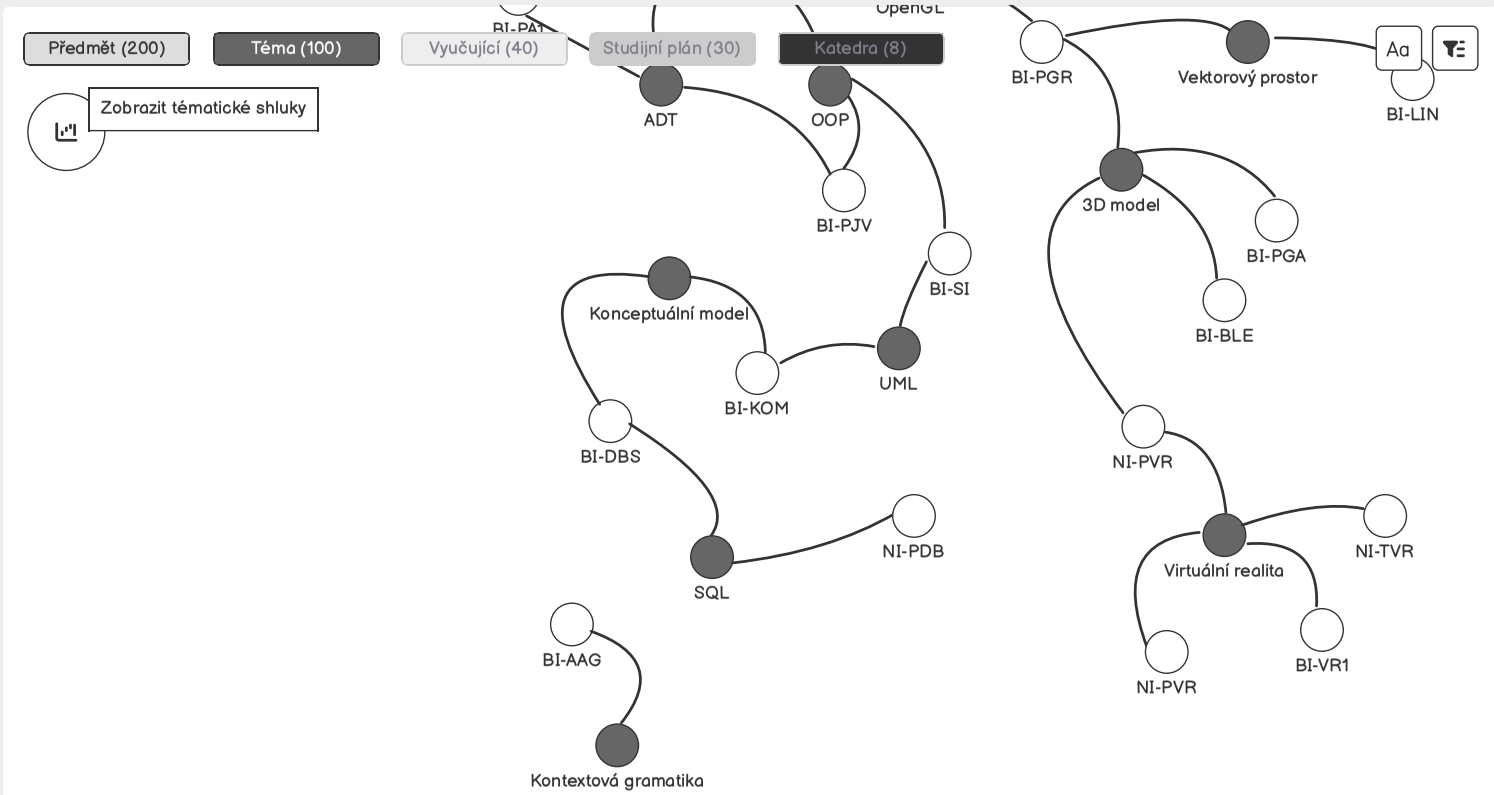
Přidat

B123

Hledat entitu...

Předmět (200) | Téma (100) | Vyučující (40) | Studijní plán (30) | Katedra (8)

Zobrazit tématické shluky



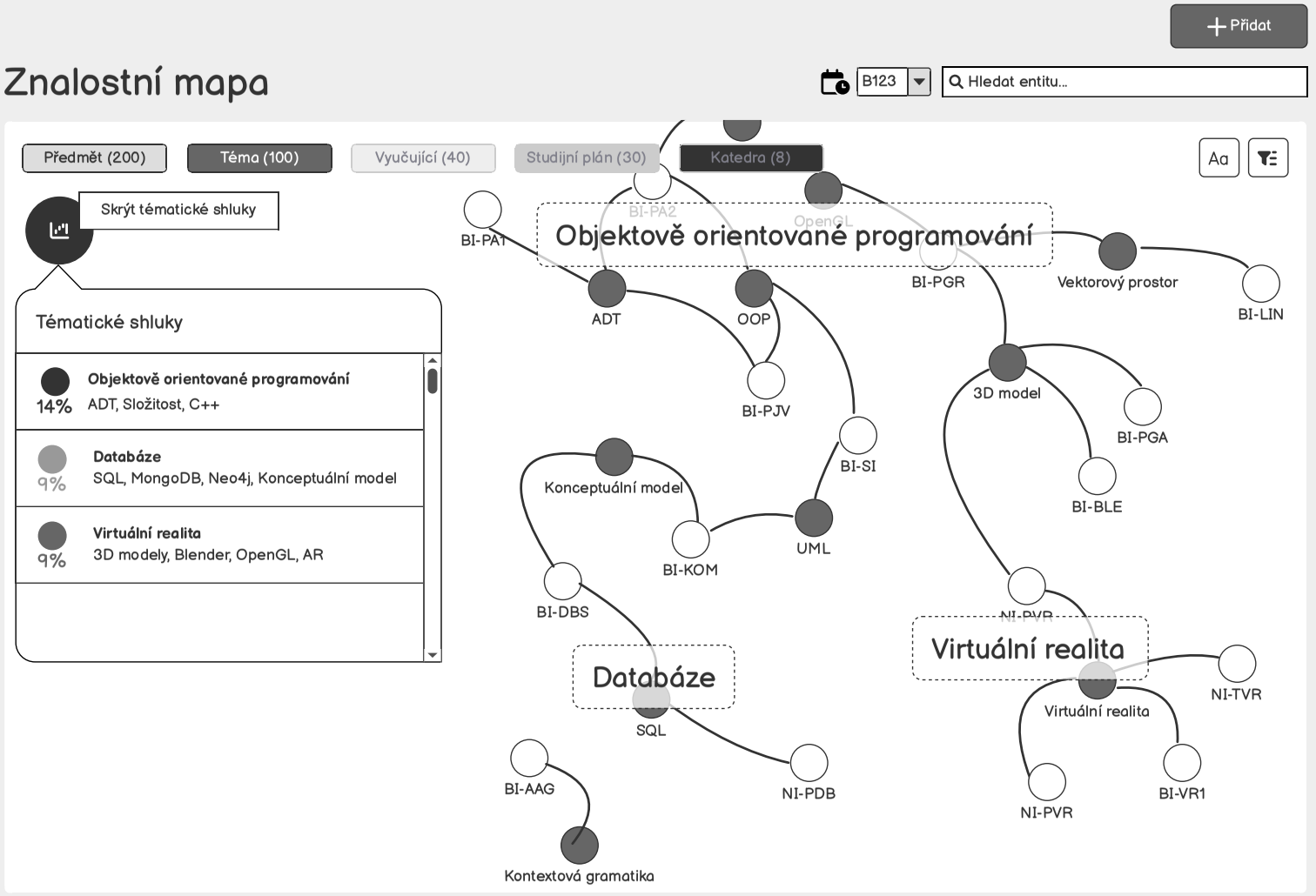
Vyberte entitu

Domů

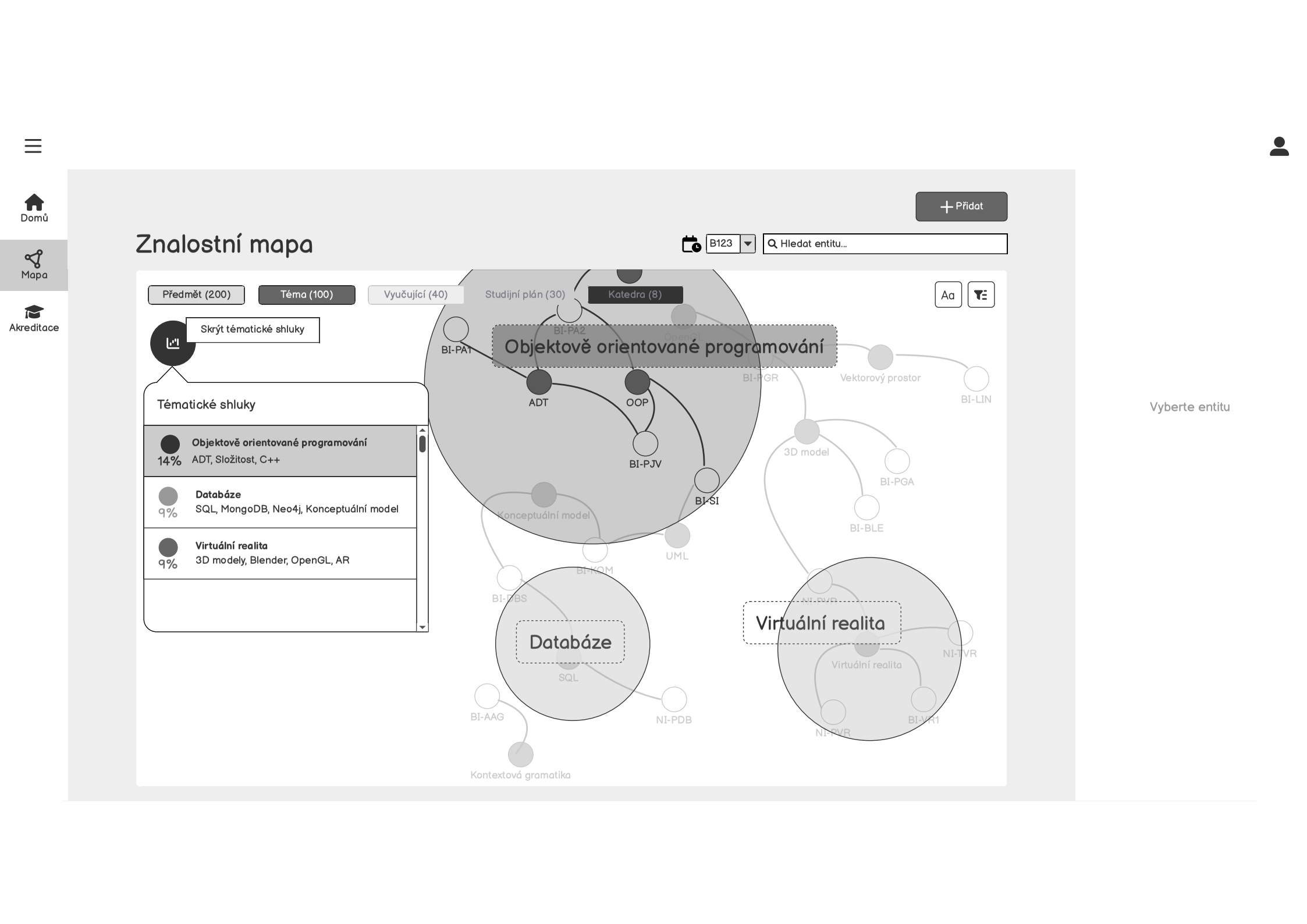
Mapa

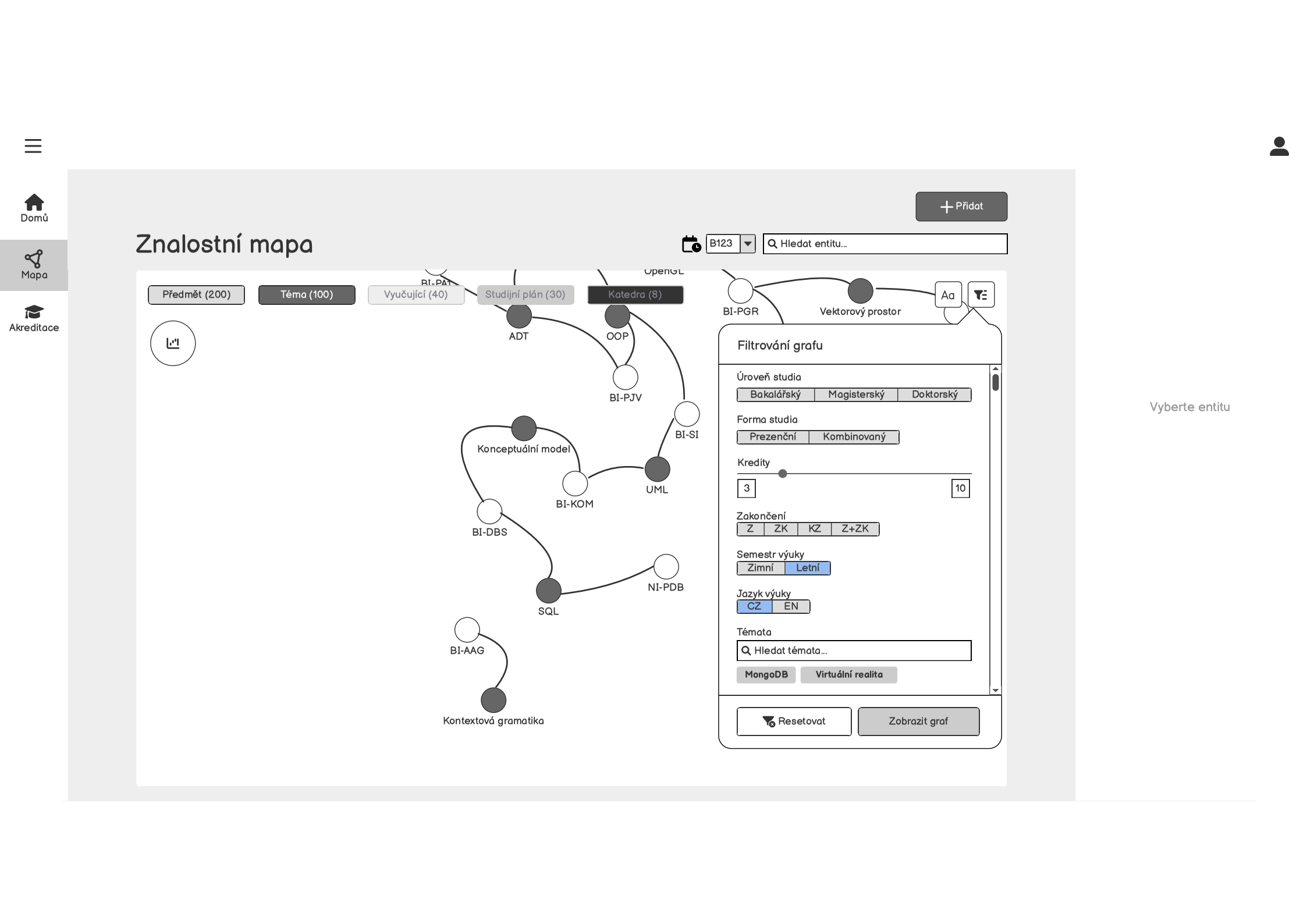
Akreditace

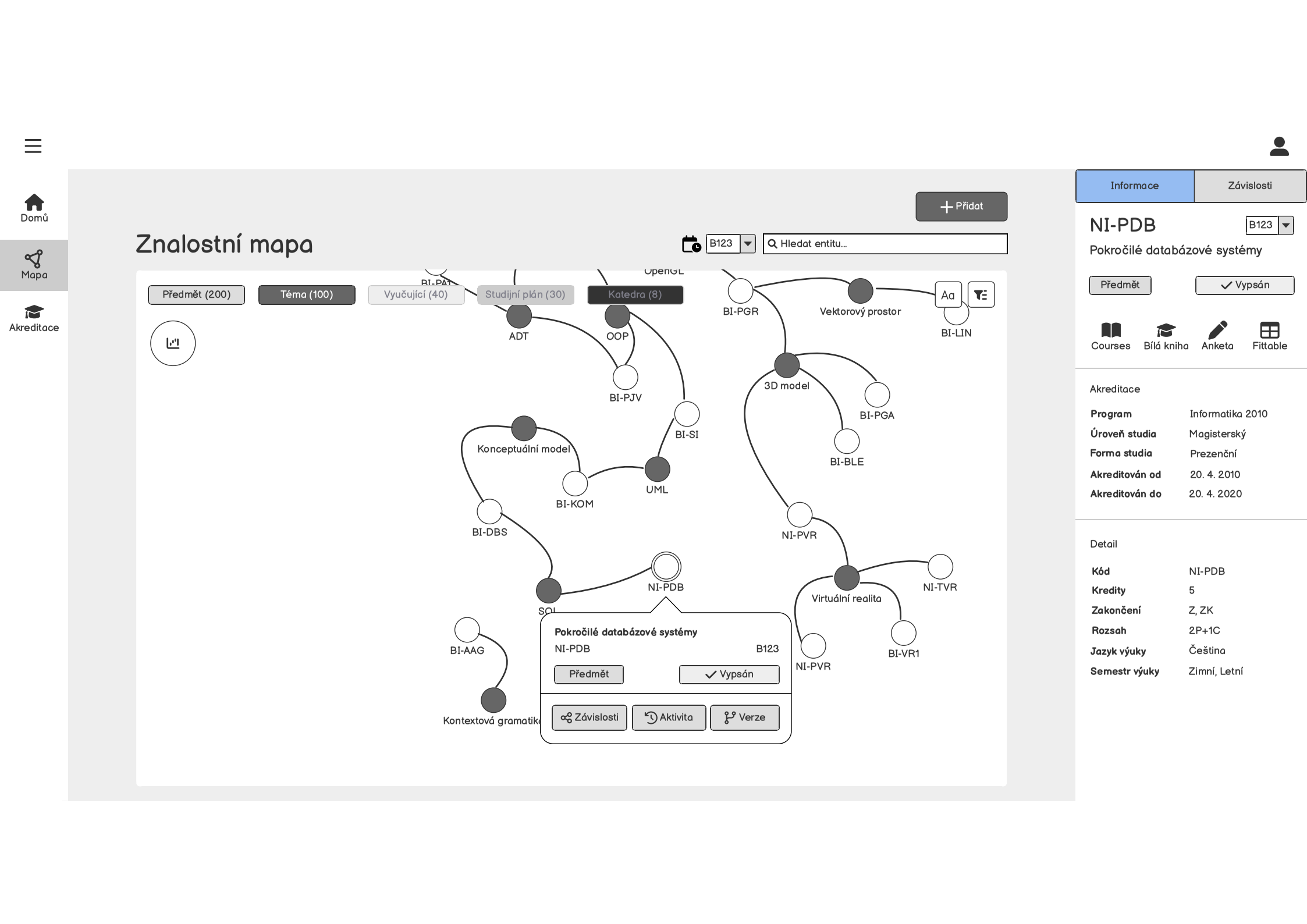
Znalostní mapa



Vyberte entitu


Domů
Mapa
Akreditace
Přidat
Znalostní mapa
B123
Hledat entitu...
Předmět (200)
Téma (100)
Vyučující (40)
Studijní plán (30)
Katedra (8)
Aa
Skrýt tématické shluky
Tématické shluky
Objektově orientované programování
ADT, Složitost, C++
14%
Databáze
SQL, MongoDB, Neo4j, Konceptuální model
9%
Virtuální realita
3D modely, Blender, OpenGL, AR
9%
BI-PA1
BI-PA2
ADT
OOP
BI-PJV
BI-SI
Objektově orientované programování
BI-PGR
Vektorový prostor
BI-LIN
3D model
BI-PGA
BI-BLE
Konceptuální model
UML
BI-KOM
BI-DBS
Databáze
SQL
BI-AAG
NI-PDB
Kontextová gramatika
Virtuální realita
NI-TVR
Virtuální realita
NI-PVR
BI-VR1
Vyberte entitu


Domů
Mapa
Akreditace
Přidat
Znalostní mapa
B123
Hledat entitu...
Předmět (200)
Téma (100)
Vyučující (40)
Studijní plán (30)
Katedra (8)
OpenGL
BI-PA1
ADT
OOP
BI-PJV
BI-SI
Konceptuální model
BI-KOM
UML
BI-DBS
SQL
NI-PDB
BI-AAG
Kontextová gramatika
BI-PGR
Vektorový prostor
Aa
Filtrování grafu
Úroveň studia
Bakalářský
Magisterský
Doktorský
Forma studia
Prezenční
Kombinovaný
Kredity
3
10
Zakončení
Z
ZK
KZ
Z+ZK
Semestr výuky
Zimní
Letní
Jazyk výuky
CZ
EN
Témata
Hledat témata...
MongoDB
Virtuální realita
Resetovat
Zobrazit graf
Vyberte entitu


Domů
Mapa
Akreditace
Přidat
Znalostní mapa
B123
Hledat entitu...
Předmět (200)
Téma (100)
Vyučující (40)
Studijní plán (30)
Katedra (8)
ADT
OOP
BI-PJV
BI-SI
Konceptuální model
UML
BI-KOM
BI-DBS
SQL
NI-PDB
BI-AAG
Kontextová gramatika
BI-PGR
Vektorový prostor
Aa
BI-LIN
3D model
BI-PGA
BI-BLE
NI-PVR
Virtuální realita
NI-TVR
NI-PVR
BI-VR1
Pokročilé databázové systémy
NI-PDB
B123
Předmět
Vypsán
Závislosti
Aktivita
Verze
Informace
Závislosti
NI-PDB
B123
Pokročilé databázové systémy
Předmět
Vypsán
Courses
Bílá kniha
Anketa
Fittable
Akreditace
Program
Informatika 2010
Úroveň studia
Magisterský
Forma studia
Prezenční
Akreditován od
20. 4. 2010
Akreditován do
20. 4. 2020
Detail
Kód
NI-PDB
Kredity
5
Zakončení
Z, ZK
Rozsah
2P+1C
Jazyk výuky
Čeština
Semestr výuky
Zimní, Letní

Domů
Mapa
Akreditace

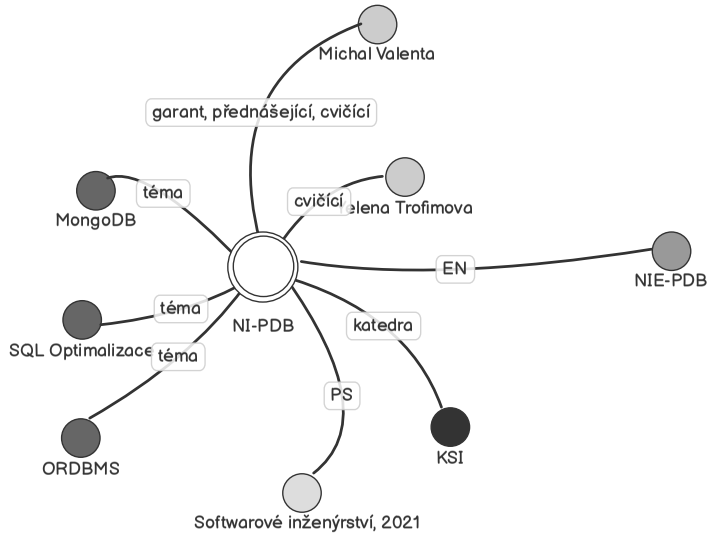
NI-PDB Pokročilé databázové systémy

Předmět | Závislosti | Aktivita | Verze

B123 | Hledat entitu...

Předmět (1) | Téma (4) | Vyučující (2) | Studijní plán (2) | Katedra (1)

Aa



Informace | Závislosti

Předmět (1)

[EN] Advanced Database Systems

Katedra (1)

[Katedra] Katedra aplikované matematiky

Vyučující (2)

[Garant] Michal Valenta

[Přednášející] Michal Valenta

[Cvičící] Michal Valenta

[Cvičící] Yelena Trofimova

Téma (6)

[Téma] SQL Optimalizace

[Téma] MongoDB

[Téma] Grafové databáze

[Téma] XML Native databáze

[Téma] Objektově Relační databáze

Studijní plán (2)

[PS] Softwarové inženýrství, 2021

[PS] Softwarové inženýrství, kombi., 2021

Domů
Mapa
Akreditace

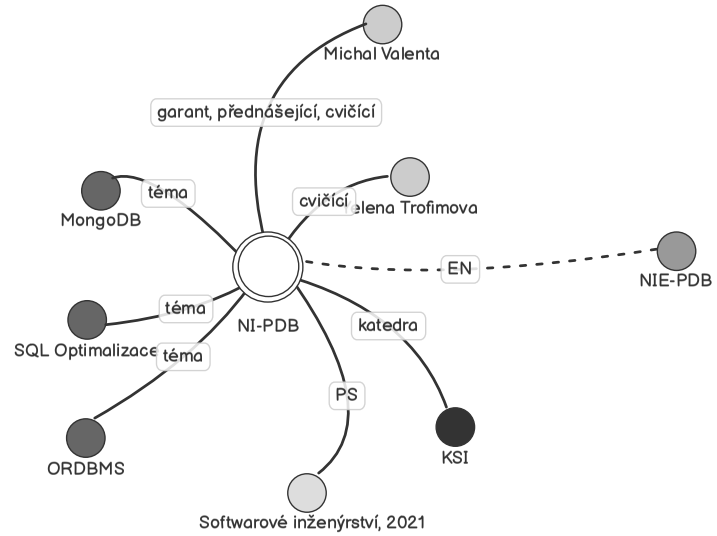
NI-PDB Pokročilé databázové systémy

Předmět | Závislosti | Aktivita | Verze

B123 | Hledat entitu...

Předmět (1) | Téma (4) | Vyučující (2) | Studijní plán (2) | Katedra (1)

Aa



Informace | Závislosti

Předmět (1)

[EN] Advanced Database Systems

Katedra (1)

[Katedra] Katedra aplikované matematiky

Vyučující (2)

[Garant] Michal Valenta

[Přednášející] Michal Valenta

[Cvičící] Michal Valenta

[Cvičící] Yelena Trofimova

Téma (6)

[Téma] SQL Optimalizace

[Téma] MongoDB

[Téma] Grafové databáze

[Téma] XML Native databáze

[Téma] Objektově Relační databáze

Studijní plán (2)

[PS] Softwarové inženýrství, 2021

[PS] Softwarové inženýrství, kombi., 2021

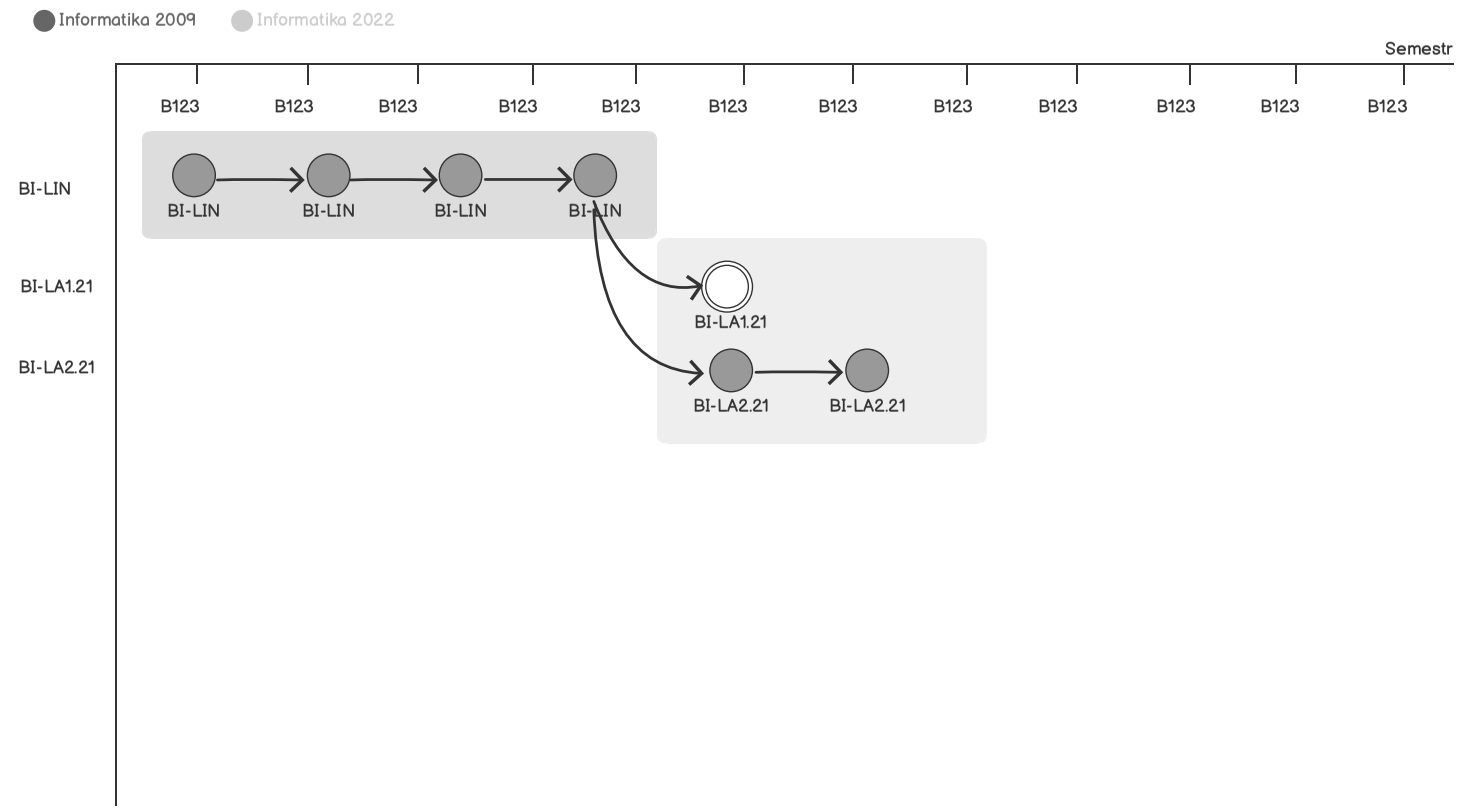
Domů

Mapa

Akreditace

BI-LA1.21 Lineární algebra 1

Předmět | Závislosti | Aktivita | **Verze**



Vyberte entitu

Domů
Mapa
Akreditace

NI-PDB Pokročilé databázové systémy

Předmět | Závislosti | **Aktivita** | Verze

B123 | Hledat entitu...

Aktivita B123 -> B124

3 změny

Téma (1)

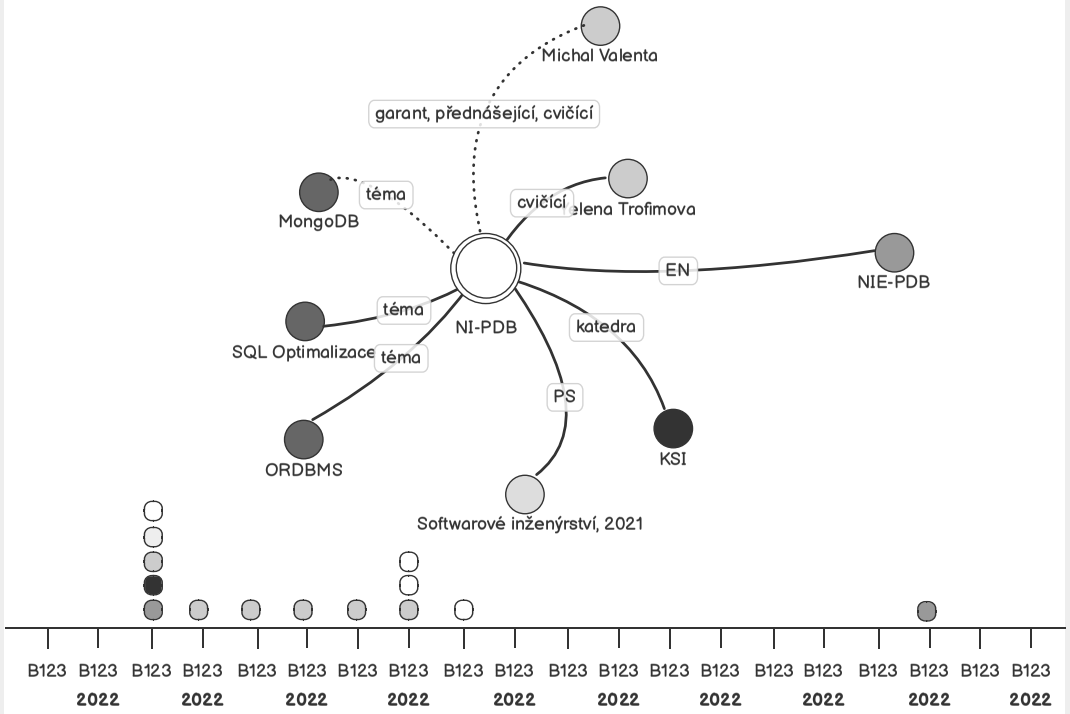
[Téma] MongoDB

Vyučující (2)

[Garant] Michal Valenta

[Cvičící] Michal Valenta

Předmět (1) | Téma (4) | Vyučující (2) | Katedra (1) | Studijní plán (2) | Aa



Informace | Závislosti

NI-PDB

B123

Pokročilé databázové systémy

Předmět | ✓ Vypsán

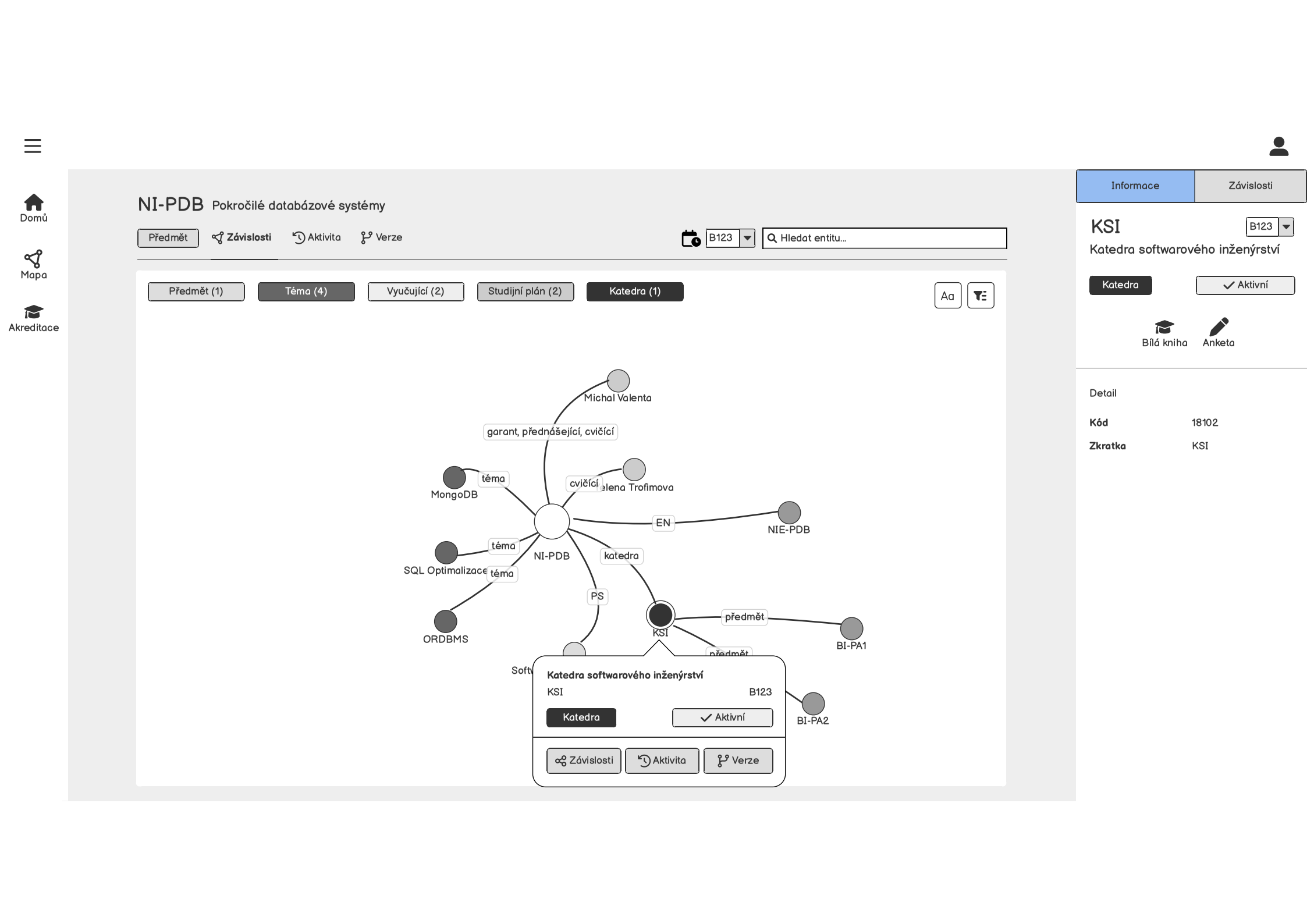
Courses | Bílá kniha | Anketa | Fittable

Akreditace

Program	Informatika 2010
Úroveň studia	Magisterský
Forma studia	Prezenční
Akreditován od	20. 4. 2010
Akreditován do	20. 4. 2020

Detail

Kód	NI-PDB
Kredity	5
Zakončení	Z, ZK
Rozsah	2P+1C
Jazyk výuky	Čeština
Semestr výuky	Zimní, Letní


Domů
Mapa
Akreditace
NI-PDB Pokročilé databázové systémy
Předmět
Závislosti
Aktivita
Verze
B123
Hledat entitu...
Předmět (1)
Téma (4)
Vyučující (2)
Studijní plán (2)
Katedra (1)
Aa
Michal Valenta
garant, přednášející, cvičící
MongoDB
téma
cvičící
Elena Trofimova
EN
NIE-PDB
téma
NI-PDB
katedra
SQL Optimalizace
téma
PS
ORDBMS
KSI
předmět
BI-PA1
předmět
Softv
Katedra softwarového inženýrství
KSI
B123
Katedra
Aktivní
BI-PA2
Závislosti
Aktivita
Verze
Informace
Závislosti
KSI
B123
Katedra softwarového inženýrství
Katedra
Aktivní
Bílá kniha
Anketa
Detail
Kód 18102
Zkratka KSI

Domů
Mapa
Akreditace

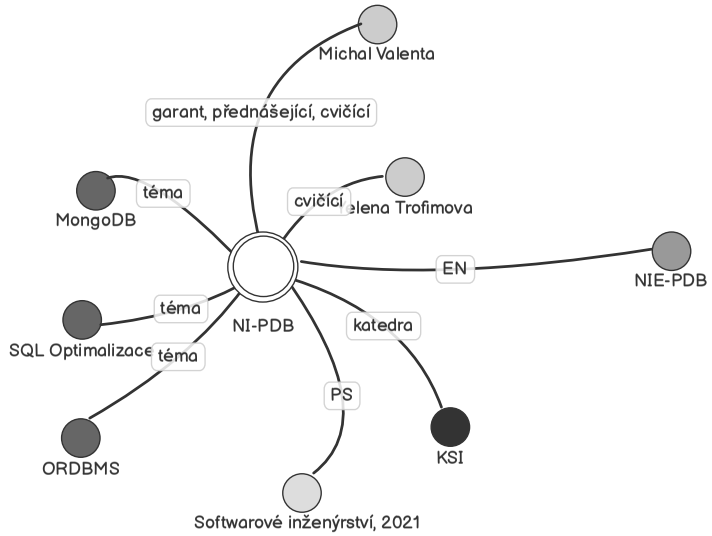
NI-PDB Pokročilé databázové systémy

Předmět | Závislosti | Aktivita | Verze

B123 | Hledat entitu...

Předmět (1) | Téma (4) | Vyučující (2) | Studijní plán (2) | Katedra (1)

Aa



Informace | Závislosti

NI-PDB

B123

Pokročilé databázové systémy

Předmět | ✓ Vypsán

Courses | Bílá kniha | Anketa | Fittable

Akreditace

Program	Informatika 2010
Úroveň studia	Magisterský
Forma studia	Prezenční
Akreditován od	20. 4. 2010
Akreditován do	20. 4. 2020

Detail

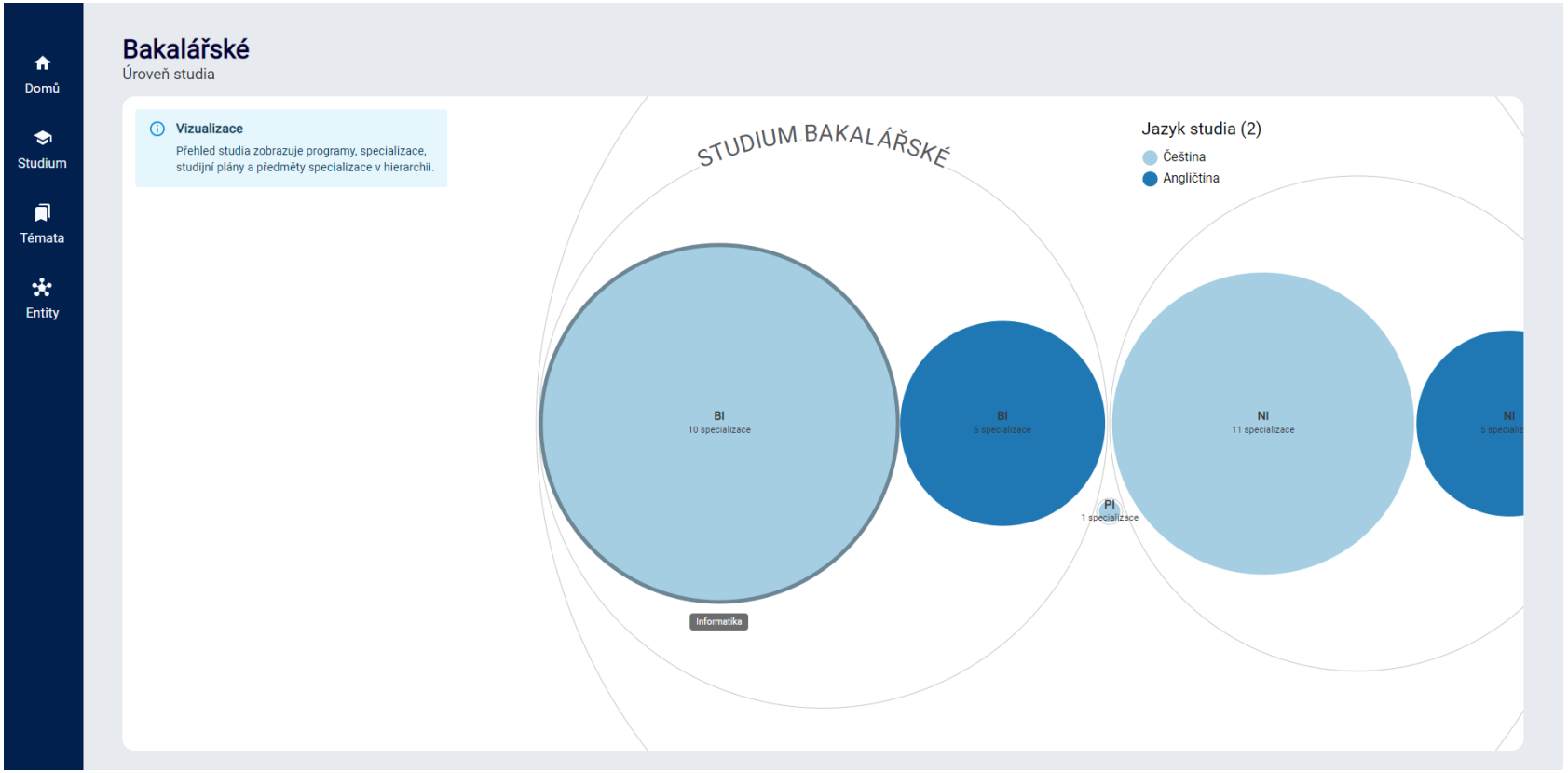
Kód	NI-PDB
Kredity	5
Zakončení	Z, ZK
Rozsah	2P+1C
Jazyk výuky	Čeština
Semestr výuky	Zimní, Letní

Příloha B

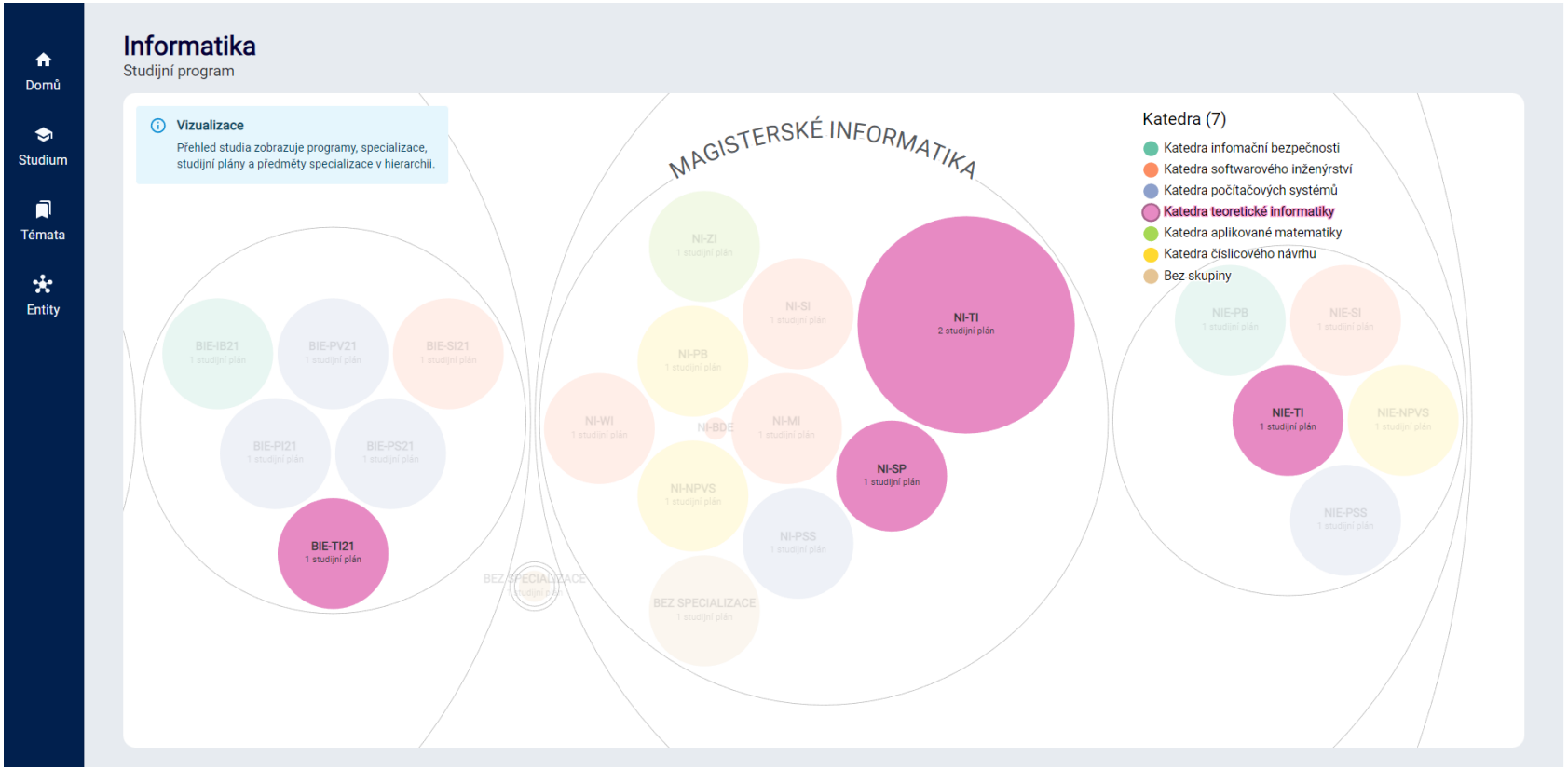
Ukázky prototypu

Video ukázky prototypu jsou také k dispozici na médiu ve složce `preview`.

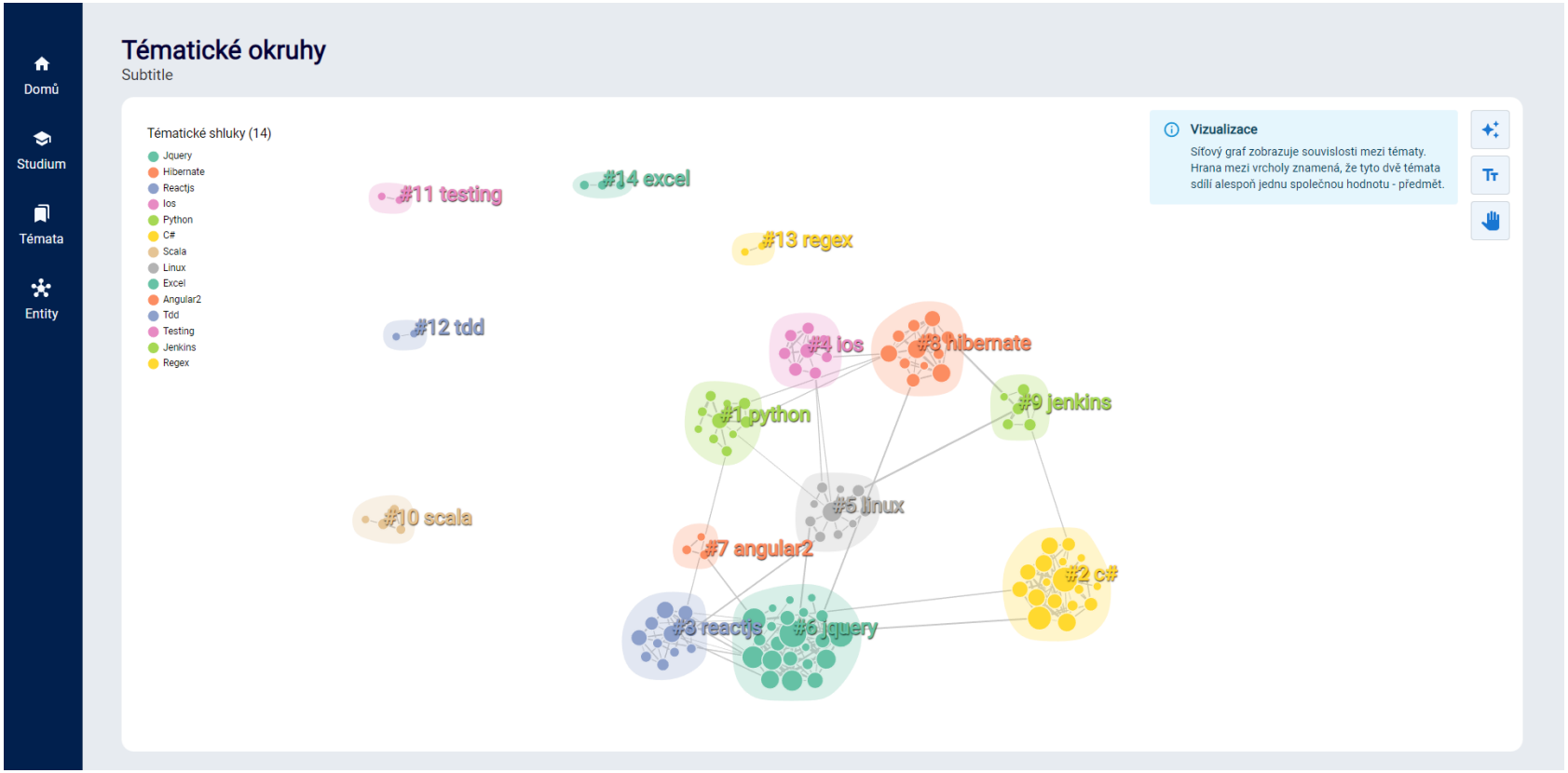
Obrázek B.1 Ukázka prototypu: Přehled studia



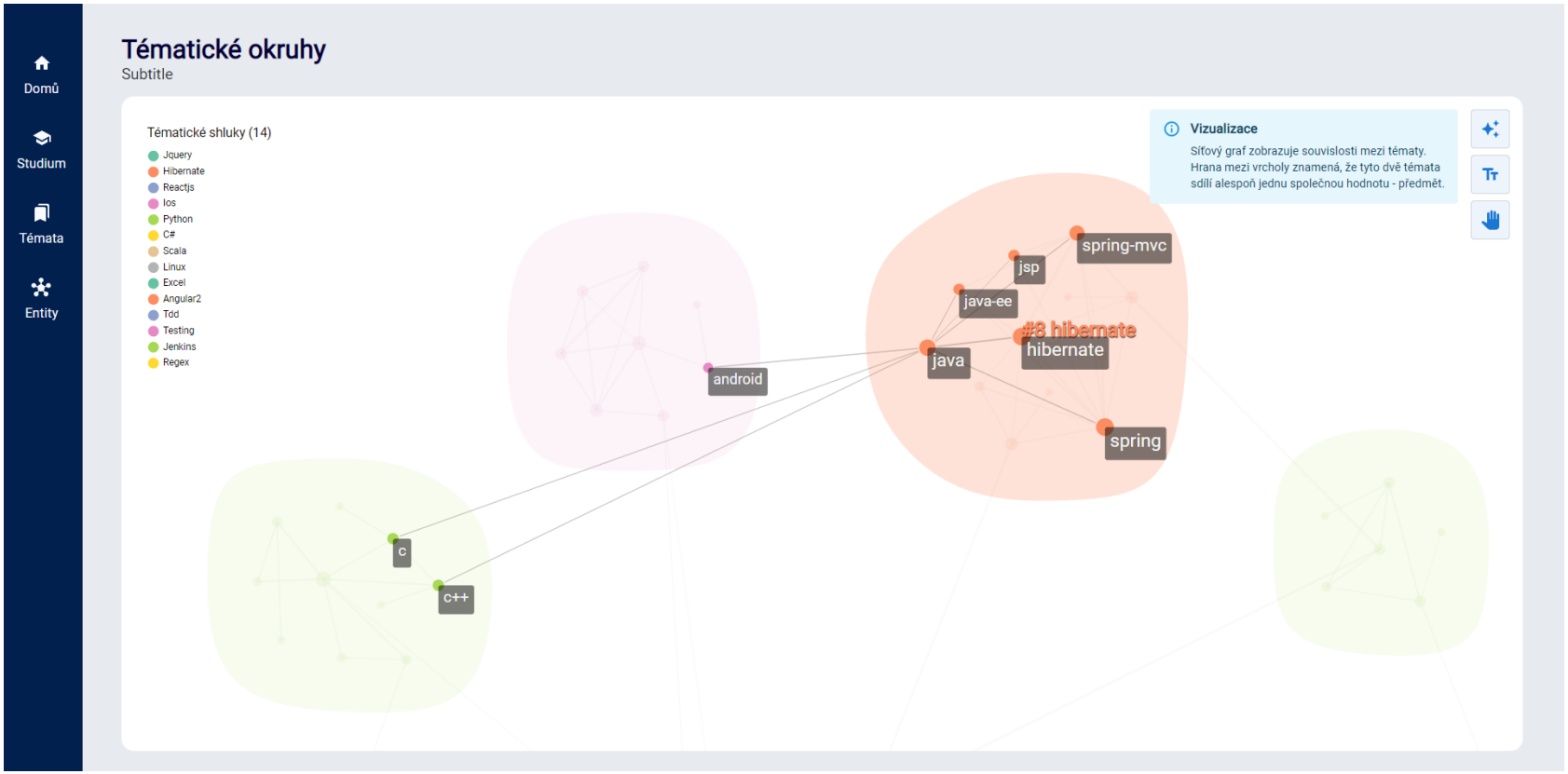
Obrázek B.2 Ukázka prototypu: Přehled studia



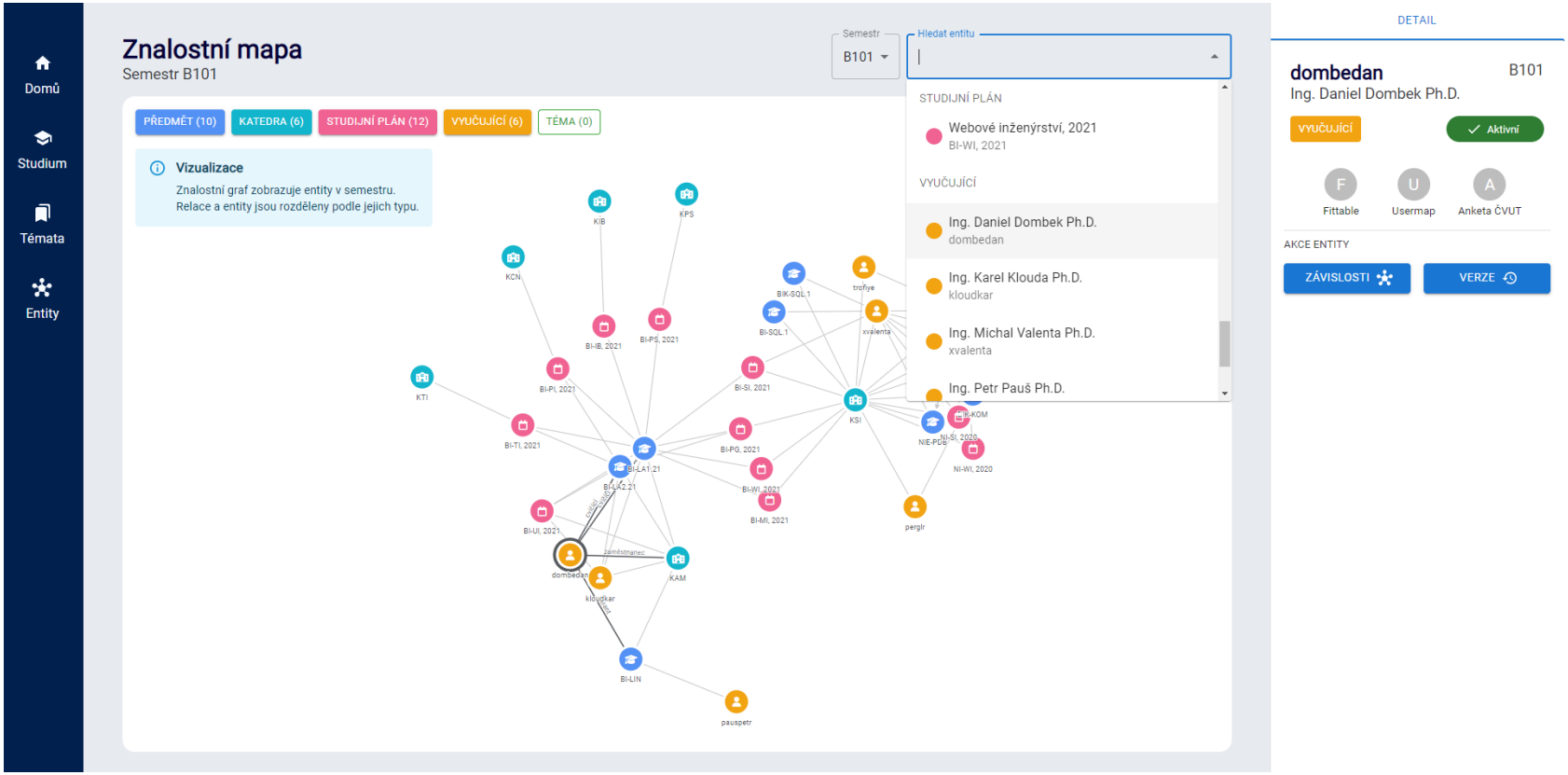
Obrázek B.3 Ukázka prototypu: Přehled studia



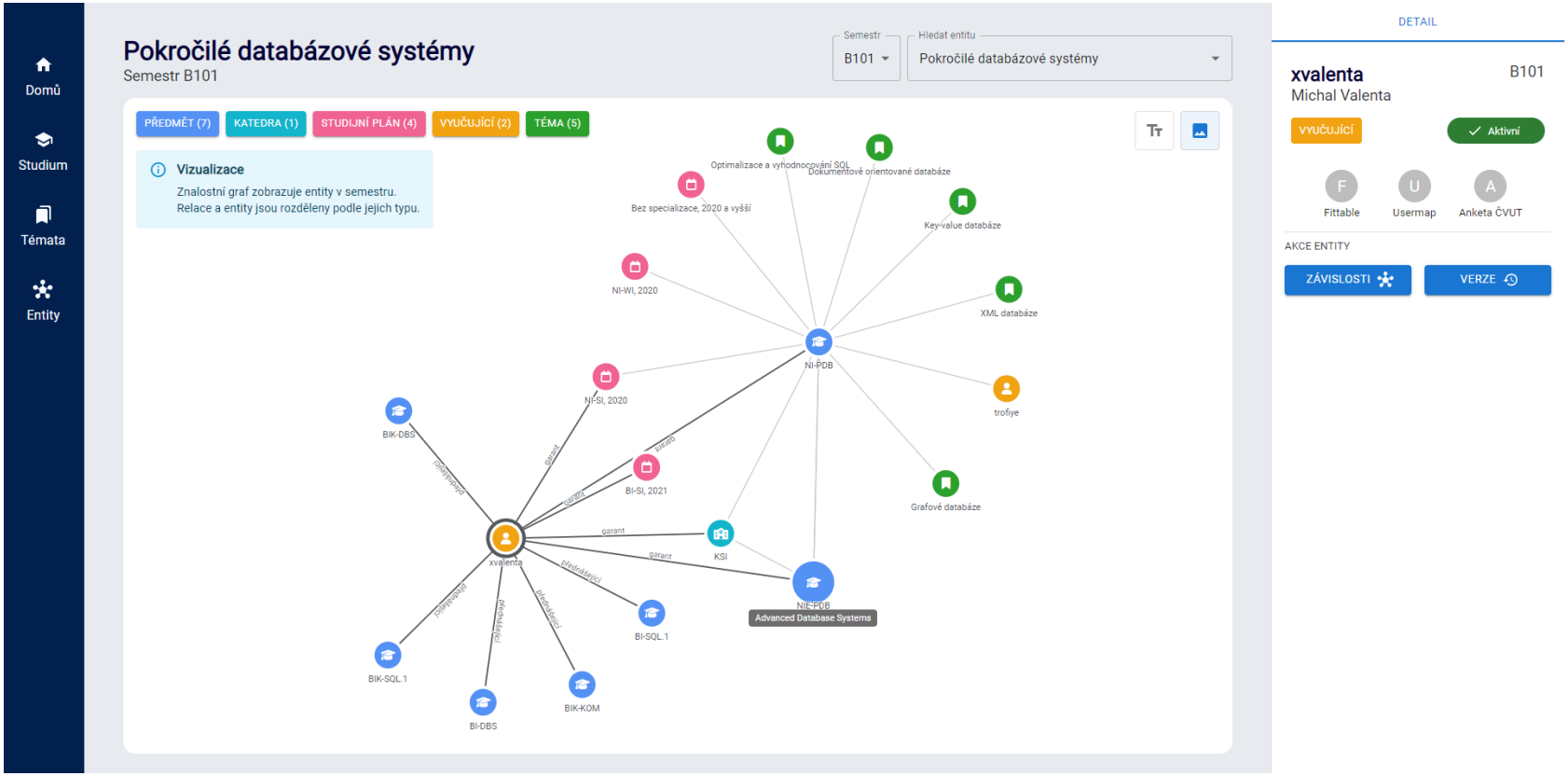
Obrázek B.4 Ukázka prototypu: Tématické shluky



Obrázek B.5 Ukázka prototypu: Tématické shluky



Obrázek B.6 Ukázka prototypu: Znalostní mapa



Obrázek B.7 Ukázka prototypu: Znalostní graf entity

Obrázek B.8 Ukázka prototypu: Verze entity

Obrázek B.9 Ukázka prototypu: Verze entity

Příloha C

Testovací scénáře

Formulář a nahrávky uživatelského testování jsou také dispozici na médiu ve složce `testing`.

C.0.1 Scénář T1

Název
T1: Vyhledání specializace v přehledu studia

Odhadovaný čas
3 min.

Účel testování
Testování interakce s vizualizací přehledu studia a schopnost uživatele navigovat mezi jednotlivými úrovněmi a skupinami podle legendy.

Pokrytí případů užití
Přehled studia - UC1, UC2

Počáteční bod
Uživatel se nachází v aplikaci.

Koncový bod
Uživatel vyhledal danou specializaci v přehledu studia.

Instrukce pro testera
Právě jste dokončili bakalářské studium na FIT ČVUT a máte v plánu pokračovat dál v magisterském studiu. Ve svém bakalářském studiu jste studovali v českém jazyce a prezenční formě specializaci Umělá inteligence (zkratka BI-UI21). Pokud by to bylo možné, chtěli byste pokračovat ve stejné specializaci i na magisterském studiu (čeština), nebo alespoň ve nejvíce podobné specializaci, např. ve stejné katedře. Vyberte vhodnou vizualizaci a v ní tuto specializaci.

Očekávané kroky

1. Výběr bakalářského programu s českým jazykem studia.
2. Vyhledání specializace (BI-UI21) a zjištění její katedry (katedra aplikované matematiky)
3. Výběr magisterského programu s českým jazykem studia

4. Výběr specializace (NI-ZI), která je zajišťována katedrou aplikované matematiky

C.0.2 Scénář T2

Název
T2: Vyhledání specializace v přehledu studia

Odhadovaný čas
2 min.

Účel testování
Testování interakce s vizualizací přehledu studia a schopnost uživatele navigovat mezi jednotlivými úrovněmi a skupinami podle legendy.

Pokrytí případů užití
Přehled studia - UC1, UC2

Počáteční bod
Uživatel se nachází v aplikaci.

Koncový bod
Uživatel vyhledal specializace v přehledu studia.

Instrukce pro testera
Studujete magisterské studium na FIT ČVUT v češtině. Právě studujete specializaci Počítačová bezpečnost (zkratka NI-PB). Váš kolega by chtěl studovat stejnou specializaci, avšak neumí češtinu a proto je nyní zapsán na magisterském studiu v angličtině. Zjistěte, zda může studovat tuto specializaci ve svém programu a popřípadě mu doporučte podobnou specializaci.

Očekávané kroky

1. Zobrazení vizualizace přehledu studia
2. Výběr magisterské úrovně studia
3. Výběr magisterského programu s českým jazykem studia.
4. Nalezení dané specializace (NI-PB)
5. Výběr specializace (NIE-PB) v anglickém programu

C.0.3 Scénář T3

Název
T3: Vyhledání tématu v tématických shlucích

Odhadovaný čas
3 min.

Účel testování
Testování interakce s vizualizací tématických shluků a navigace v síťovém grafu. Uživatel by měl být schopný využít dodatečných UI elementů pro lepší navigaci v grafu. Uživatel by měl mít uvědomění, co graf vyjadřuje a na jakém principu funguje.

Pokrytí případů užití
Znalostní mapa - UC2

Počáteční bod
Uživatel se nachází v aplikaci.

Koncový bod
Uživatel vyhledal dané témata.

Instrukce pro testera
Po prvním semestru na FIT ČVUT jste se začali zajímat o programovací jazyk C++. Avšak jediný co víte je, že toto téma by mohlo mít souvislost s jazykem C# nebo Python. Vyberte vhodnou vizualizaci a najděte v ní téma C++ společně s jeho nejbližšími tématy.

Očekávané kroky

1. Zobrazení vizualizace tématických shluků
2. Vyhledání a prozkoumání shluků C# a Python v panelu
3. Výběr shluku Python s přítomným tématem C++ v panelu
4. Zapnutí tématických shluků v grafu (volitelné)
5. Vyhledání tématu C++ s jeho sousedy v grafu (Java, Python, Qt, C)

C.0.4 Scénář T4

Název
T4: Vyhledání předmětů v tématických shlucích

Odhadovaný čas
3 min.

Účel testování
Testování interakce s vizualizací tématických shluků a navigace v síťovém grafu. Uživatel by měl být schopný využít dodatečných UI elementů pro lepší navigaci v grafu. Uživatel by měl mít uvědomění, co graf vyjadřuje a na jakém principu funguje.

Pokrytí případů užití
Znalostní mapa - UC2

Počáteční bod
Uživatel se nachází v tématických shlucích.

Koncový bod
Uživatel vyhledal dané předměty (hodnoty).

Instrukce pro testera
Zajímáte se o témata, které mají souvislost s tématem Hibernate. Abyste rozšířily své obzory, chtěli byste najít takové témata, které stále mají souvislost s okruhem Hibernate, ale i s jinými tématickými okruhy. Z nich vyberte takové téma, které má nejvíce společných předmětů (hodnot) s tématem z jiného okruhu. Dále zjistěte o jaké předměty se jedná.

Očekávané kroky

1. Zapnutí tématických shluků a popisků hran v grafu
2. Vyhledání okruhu Hibernate
3. Výběr témat, které mají souvislost (hranu) s jiným okruhem (Maven, Rest, Java)
4. Výběr tématu s nejvíce společných předmětů (Jenkins)
5. Vyhledání předmětů v panelu

C.0.5 Scénář T5

Název
Filtrování ve znalostní mapě

Odhadovaný čas
2 min.

Účel testování
Testování interakce se znalostní mapou a postranním panelem. Uživatel by měl být schopný filtrovat mapu a efektivně zobrazit entity.

Pokrytí případů užití
Znalostní mapa - UC1, UC3
Informace entity - UC1

Počáteční bod
Uživatel se nachází v aplikaci.

Koncový bod
Uživatel se nachází ve stránce studijního plánu v Bílé knize.

Instrukce pro testera
V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny studijní plány, které nejsou zajišťovány žádnou katedrou. Vyberte vhodnou vizualizaci, vyberte jeden z těchto plánů a zobrazte si jeho obsah v Bílé knize.

Očekávané kroky

1. Zobrazení vizualizace znalostní mapy
2. Vybrání semestru B212 v mapě
3. Filtrování mapy na entity (studijní plán a katedra)
4. Vybrání entity studijního plánu v mapě (Bez specializace, 2020 a vyšší)
5. Vybrání odkazu Bílé knihy v postranním panelu entity

C.0.6 Scénář T6

Název
Hledání v mapě a zobrazení závislostí entity

Odhadovaný čas
2 min.

Účel testování
Testování interakce se znalostní mapou a postranním panelem. Uživatel by měl být schopný vyhledat určitou entitu v mapě a zobrazit její závislosti.

Pokrytí případů užití
Znalostní mapa - UC3, UC4
Pohledy entity - UC1

Počáteční bod
Uživatel se nachází ve znalostní mapě.

Koncový bod
Uživatel má zobrazené dané předměty

Instrukce pro testera
V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny předměty, které mají jako garanta Michala Valentu. Vyberte vhodnou vizualizaci a vypište tyto předměty.

Očekávané kroky

1. Vybrání semestru B212 ve znalostní mapě
2. Hledání a výběr entity v mapě (xvalenta)
3. Zobrazení závislostí entity pomocí postranního panelu
4. Zapnutí zobrazení popisků relací ve znalostním grafu
5. Zobrazení předmětů v grafu (NIE-PDB a NI-PDB)

C.0.7 Scénář T7

Název
Zobrazení závislosti entity a rozšíření o sousední závislosti

Odhadovaný čas
2 min.

Účel testování
Testování interakce se znalostní mapou a postranním panelem. Uživatel by měl být schopný efektivně zobrazit sousední závislosti. Uživatel by měl být schopný použít funkcionality grafu jako je zobrazenní nebo skrytí popisků hran.

Pokrytí případů užití
Pohledy entity - UC1, UC2
Informace entity - UC1

Počáteční bod
Uživatel se nachází ve znalostní mapě.

Koncový bod
Uživatel má zobrazené závislosti entity a popř. rozšířenou entitu.

Instrukce pro testera
V akademickém roce právě probíhá semestr B212. V tomto semestru Vás zajímají všechny závislosti předmětu Lineární algebra. Poté Vás zajímá, jaké další předměty vyučuje garant tohoto předmětu. Vyberte vhodnou vizualizaci a zobrazte zmíněné závislosti.

Očekávané kroky

1. Vybrání semestru B212 ve znalostní mapě
2. Hledání a zobrazení závislostí entity (BI-LIN)
3. Rozšíření entity ve znalostním grafu entity (dombedan)
4. Zobrazení předmětů v grafu (BI-LA1.21 a BI-LA2.21)

C.0.8 Scénář T8

Scénář
Zobrazení a prozkoumání verzí entity

Odhadovaný čas
2 min.

Účel testování
Testování uživatelského rozhraní verzí entity a postranního panelu. Uživatel by měl být schopný zobrazit verze entity a porozumět jejímu principu.

Pokrytí případů užití
Pohledy entity - UC3, UC4

Počáteční bod
Uživatel se nachází ve znalostní mapě.

Koncový bod
Uživatel se nachází ve verzích entity

Instrukce pro testera
S nástupem nové akreditace byly provedeny změny v předmětu Lineární algebra. Zajímá Vás, jaký byl vývoj verzí této entity. Zjistěte v jakém semestru (popř. semestrech) nastala změna a o jakou změnu se jednalo (jaké předměty se podílely na této změně).

Očekávané kroky

1. Zobrazení závislostí entity (BI-LIN)
2. Výběr verzí entity v postranním panelu entity
3. Prozkoumání a interakce s vizualizací verzí entity

Bibliografie

1. syllabus, n. In: *OED Online* [online]. Oxford University Press, 2023 [cit. 2023-03-18]. Dostupné z: https://www.oed.com/view/Entry/196148?redirectedFrom=syllabus.
2. JIŘÍ KOSEK, ČVUT. *Bílá kniha ČVUT* [online]. ČVUT, 2010 [cit. 2023-03-18]. Dostupné z: https://bilakniha.cvut.cz/.
3. J. NOVÁK I.Halaška, FIT ČVUT. *Bílá kniha FIT ČVUT* [online]. FIT ČVUT, 2012 [cit. 2023-03-18]. Dostupné z: https://bk.fit.cvut.cz/.
4. QCM, s.r.o web@qcm.cz http://www.qcm.cz. *AKREDITACE VZDĚLÁVACÍ INSTITUCE* [online]. MŠMT, [b.r.] [cit. 2023-03-18]. Dostupné z: https://www.msmt.cz/vzdelavani/dalsi-vzdelavani/akreditace-vzdelavaci-instituce.
5. hierarchy, n. In: *OED Online* [online]. Oxford University Press, 2023 [cit. 2023-03-19]. Dostupné z: https://www.oed.com/view/Entry/86792?redirectedFrom=hierarchy.
6. PORTADESIGN.CZ. *Nová Akreditace Bakalářského studijního Programu a změny ve výuce matematiky* [online]. FIT ČVUT, 2021 [cit. 2023-03-18]. Dostupné z: https://fit.cvut.cz/cs/studium/pruvodce-studiem/bakalarske-a-magisterske-studium/nova-akreditace-bsp.
7. PORTADESIGN.CZ. *Uznávání Předmětů Mezi starým a nově Akreditovaným Bakalářským Programem* [online]. FIT ČVUT, 2021 [cit. 2023-03-19]. Dostupné z: https://fit.cvut.cz/cs/studium/pruvodce-studiem/bakalarske-a-magisterske-studium/uznavani-predmetu-mezi-bc-programy.
8. *České vysoké školy* [online]. MŠMT, 2022 [cit. 2023-03-18]. Dostupné z: https://regvssp.msmt.cz/registrvssp/cvslist.aspx.
9. *Kos* [online]. České vysoké učení technické v Praze, [b.r.] [cit. 2023-03-19]. Dostupné z: https://www.kos.cvut.cz/.
10. *Moodle-výuka* [online]. [B.r.]. [cit. 2023-03-19]. Dostupné z: https://moodle-vyuka.cvut.cz/.
11. *Kos* [online]. České vysoké učení technické v Praze, [b.r.] [cit. 2023-03-19]. Dostupné z: https://new.kos.cvut.cz/.
12. *FIT CTU Courses* [online]. Fakulta informačních technologií, České vysoké učení technické v Praze, [b.r.] [cit. 2023-03-19]. Dostupné z: https://courses.fit.cvut.cz/.
13. *ASCIIDOC* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: https://help.fit.cvut.cz/courses/content/asciidoc.html.
14. *Anketa ČVUT* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: https://anketa.is.cvut.cz/.

15. *Fittable beta: Fit čvut* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: https://timetable.fit.cvut.cz/new/.
16. [online]. [B.r.]. [cit. 2023-04-27]. Dostupné z: https://projects.fit.cvut.cz/.
17. ČVUT-VIC, Ing. Petr Karel. *Vyhledávání osob na čvut* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: https://usermap.cvut.cz/.
18. HEALY, Yan Holtz; CONOR. *Network diagram* [online]. from Data to Viz, [b.r.] [cit. 2023-03-20]. Dostupné z: https://www.data-to-viz.com/graph/network.html.
19. DOC. RNDR. PETR HLINĚNÝ, Ph.D. *Základy teorie grafů pro (nejen) informatiky* [online]. FI MU, 2010 [cit. 2023-03-20]. Dostupné z: https://is.muni.cz/do/1499/el/estud/fi/js10/grafy/Grafy-text10.pdf.
20. RIBECCA, Severino [online]. [B.r.]. [cit. 2023-03-19]. Dostupné z: https://datavizcatalogue.com/.
21. HOU, Janpu. *Network Visualization by igraph* [online]. RStudio, 2017 [cit. 2023-03-20]. Dostupné z: https://rpubs.com/JanpuHou/337696.
22. BOSTOCK, Mike. *Data-driven documents* [online]. [B.r.]. [cit. 2023-03-20]. Dostupné z: https://d3js.org/.
23. MEEKS, E. *D3.js in Action: Data visualization with JavaScript*. Manning Publications, 2017. ISBN 9781617294488. Dostupné také z: https://books.google.cz/books?id=UeUQMQAACAAJ.
24. *Community structure* [online]. Wikimedia Foundation, 2023 [cit. 2023-03-19]. Dostupné z: https://en.wikipedia.org/wiki/Community_structure.
25. BEDI, Punam; SHARMA, Chhavi. Community detection in social networks. *WIREs Data Mining and Knowledge Discovery*. 2016, roč. 6, č. 3, s. 115–135. Dostupné z DOI: https://doi.org/10.1002/widm.1178.
26. *Community detection algorithms - developer guides* [online]. Neo4j, [b.r.] [cit. 2023-03-20]. Dostupné z: https://neo4j.com/developer/graph-data-science/community-detection-graph-algorithms/.
27. NEO4J. *The leader in Graph databases* [online]. 2022. [cit. 2023-03-20]. Dostupné z: https://neo4j.com/.
28. MOORE, Karleigh; KHIM, Jimin; ROSS, Eli. *Greedy algorithms* [online]. [B.r.]. [cit. 2023-03-20]. Dostupné z: https://brilliant.org/wiki/greedy-algorithm/.
29. convex, adj. and n. In: *OED Online* [online]. Oxford University Press, 2023 [cit. 2023-03-25]. Dostupné z: https://www.oed.com/view/Entry/40795?redirectedFrom=convex+hull.
30. EHRLINGER, Lisa; WÖSS, Wolfram. Towards a Definition of Knowledge Graphs. In: 2016. Dostupné také z: https://www.researchgate.net/publication/323316736_Towards_a_Definition_of_Knowledge_Graphs.
31. SINGHAL, Amit. *Introducing the knowledge graph: Things, not strings* [online]. Google, 2012 [cit. 2023-03-20]. Dostupné z: https://blog.google/products/search/introducing-knowledge-graph-things-not/.
32. PAULHEIM, Heiko. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*. 2016, roč. 8, s. 489–508.
33. *Graph your twitter network in NEO4J* [online]. Neo4j, [b.r.] [cit. 2023-03-19]. Dostupné z: http://network.graphdemos.com/.
34. WATTENBERG, Martin; VIÉGAS, Fernanda B. The word tree, an interactive visual concordance. *IEEE transactions on visualization and computer graphics*. 2008, roč. 14, č. 6, s. 1221–1228.

35. DAVIES, Jason [online]. [B.r.]. [cit. 2023-03-20]. Dostupné z: https://www.jasondavies.com/wordtree/.
36. HOLTZ, Yan. *Dendrogram customization with R and ggraph* [online]. [B.r.]. [cit. 2023-03-19]. Dostupné z: https://r-graph-gallery.com/335-custom-ggraph-dendrogram.html.
37. COOK, Peter. *D3 hierarchies* [online]. [B.r.]. [cit. 2023-03-20]. Dostupné z: https://www.d3indepth.com/hierarchies/.
38. *What is nosql? NoSQL databases explained* [online]. [B.r.]. [cit. 2023-03-24]. Dostupné z: https://www.mongodb.com/nosql-explained.
39. *Neo4j bloom user interface guide - developer guides* [online]. [B.r.]. [cit. 2023-03-24]. Dostupné z: https://neo4j.com/developer/neo4j-bloom/.
40. *AI Text Network Analysis for Research and Exploration* [online]. [B.r.]. [cit. 2023-03-25]. Dostupné z: https://infranodus.com/.
41. PARANYUSHKIN, Dmitry. InfraNodus: Generating Insight Using Text Network Analysis. In: *The World Wide Web Conference*. San Francisco, CA, USA: Association for Computing Machinery, 2019, s. 3584–3589. WWW '19. ISBN 9781450366748. Dostupné z DOI: 10.1145/3308558.3314123.
42. CHANDLER, Daniel; MUNDAY, Rod. *sentiment analysis* [online]. Oxford University Press, 2016 [cit. 2023-03-25]. ISBN 9780191803093. Dostupné z DOI: 10.1093/acref/9780191803093.013.1306.
43. *Find and explore academic papers* [online]. [B.r.]. [cit. 2023-03-25]. Dostupné z: https://www.connectedpapers.com/.
44. ROBIE, Jonathan. *What is the Document Object Model?* [online]. [B.r.]. [cit. 2023-03-26]. Dostupné z: https://www.w3.org/TR/WD-DOM/introduction.html.
45. *SCALABLE VECTOR GRAPHICS (SVG)* [online]. [B.r.]. [cit. 2023-03-26]. Dostupné z: https://www.w3.org/Graphics/SVG/.
46. *WebGL* [online]. The Khronos Group, 2011 [cit. 2023-03-26]. Dostupné z: https://www.khronos.org/webgl/.
47. ROCA, Aral. *First steps in WebGL* [online]. [B.r.]. [cit. 2023-03-26]. Dostupné z: https://aralroca.com/blog/first-steps-in-webgl.
48. *G6 Graph Visualization Engine* [online]. AntV, [b.r.] [cit. 2023-03-27]. Dostupné z: https://g6.antv.antgroup.com/en.
49. *Vis.js community edition* [online]. [B.r.]. [cit. 2023-03-27]. Dostupné z: https://visjs.org/.
50. *sigma.js a JavaScript library aimed at visualizing graphs of thousands of nodes and edges* [online]. Sciences-Po médialab, OuestWare, [b.r.] [cit. 2023-03-27]. Dostupné z: https://www.sigmajs.org/.
51. PLIQUE, Guillaume. *Graphology, a robust and multipurpose Graph object for JavaScript.* [online]. Zenodo, 2023. 0.25.1-lib [cit. 2023-03-27]. Dostupné z DOI: 10.5281/zenodo.7695163.
52. PAVLÍČEK, Josef. *UI design steps* [online]. [B.r.]. [cit. 2023-04-03]. Dostupné z: https://docs.google.com/presentation/d/1CldSHyC8D8cN2LGrqUVJ2U5eEKn5z9MpkFPmzwZX4Zc/edit#slide=id.g9cc1296fe4_0_429.
53. NIELSEN, Jakob. Enhancing the Explanatory Power of Usability Heuristics. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Boston, Massachusetts, USA: Association for Computing Machinery, 1994, s. 152–158. CHI '94. ISBN 0897916506. Dostupné z DOI: 10.1145/191666.191729.

54. *Hooks FAQ* [online]. [B.r.]. [cit. 2023-04-10]. Dostupné z: https://legacy.reactjs.org/docs/hooks-faq.html#which-versions-of-react-include-hooks.

55. *Choose between traditional web apps and Single Page Apps* [online]. Microsoft, 2023 [cit. 2023-04-20]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.

56. *Stack Overflow Tag Network* [online]. Stack Overflow, 2017 [cit. 2023-05-01]. Dostupné z: https://www.kaggle.com/datasets/stackoverflow/stack-overflow-tag-network?select=stack_network_nodes.csv.

57. *Introduction* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: https://d3plus.org/.

58. NIELSEN, Jakob. *How many test users in a usability study?* [online]. 2012. [cit. 2023-04-27]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `wireframes` ..... vypracované wireframy aplikace
  - `diagrams` ..... vypracované (UML) diagramy návrhu
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `preview` ..... ukázky z implementace
- `testing` ..... výstupy z testování