FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Jednodušší navazování mezioborové spolupráce ve vědecké komunitě díky efektivnímu zobrazování vzájemných vztahů
Student:	Alisher Nurmatov
Vedoucí:	Ing. David Pešek
Studijní program:	Informatika
Obor / specializace:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

1. Specifikujte funkční a nefunkční požadavky na aplikaci.
2. Proveďte rešerši podobných nástrojů.
3. Navrhněte vlastní řešení, zvolte nejvhodnější implementační platformu.
4. Implementujte prototyp vizualizačního nástroje.
5. Proveďte testování na reálných datech.
6. Zhodnoťte ekonomický přínos systému pro fakultu a popř. možnou monetizaci nástroje.
7. Zpracujte další rozvoj (studii proveditelnosti) vizualizačního nástroje.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 30. prosince 2022 v Praze.

Bakalářská práce

JEDNODUŠŠÍ NAVAZOVÁNÍ MEZIOBOROVÉ SPOLUPRÁCE VE VĚDECKÉ KOMUNITĚ DÍKY EFEKTIVNÍMU ZOBRAZOVÁNÍ VZÁJEMNÝCH VZTAHŮ

Alisher Nurmatov

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. David Pešek
16. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci: Nurmatov Alisher. *Jednodušší navazování mezioborové spolupráce ve vědecké komunitě díky efektivnímu zobrazování vzájemných vztahů.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu Ing. Davidu Peškovi za jeho cenné rady, odborné vedení a podporu, kterou mi poskytoval po celou dobu přípravy této práce. Dále bych rád poděkoval své rodině za jejich neustálou podporu a povzbuzení během celého 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 16. května 2024

Abstrakt

Projekt klade za cíl vyvinout vizualizační nástroj pro jednodušší a interaktivní zobrazení mezioborové spolupráce. Tato práce se zabývá návrhem a vývojem prototypu takového nástroje, ve kterém jsou implementovány interaktivní vizualizace. Součástí je rešerše podobných nástrojů, definice funkčních a nefunkčních požadavků a testování.

Práce hodnotí ekonomický přínos pro fakultu ve formě ušetřeného času zaměstnanců. Součástí je také budoucí rozvoj dalších možností vizualizace a nových funkcí tohoto nástroje a studie proveditelnosti.

Klíčová slova čvut, vizualizace, graf, uživatelské rozhraní, nodej.js, neo4j, neovis.js, react, javascript, síťový graf

Abstract

The project aims to develop a visualization tool for simpler and more interactive display of interdisciplinary collaboration. This work involves the design and development of a prototype of such a tool, which incorporates interactive visualizations. It includes a review of similar tools, definition of functional and non-functional requirements, and testing.

The work evaluates the economic benefit for the faculty in terms of time saved by employees. It also includes future development of additional visualization options and new features of this tool and a feasibility study.

Keywords cvut, visualization, graph, user interface, node.js, neo4j, neovis.js, react, javascript, network graph

Seznam zkratek

HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
ČVUT	České vysoké učení technické v Praze
CSV	Comma-separated values
JSON	JavaScript Object Notation
FIT	Fakulta informačních technologií ČVUT v Praze
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
VIC	Výpočetní a informační centrum ČVUT
DOI	Digital Object Identifier
UML	Unified Modeling Language
SPA	Single-page application
REST	Representational State Transfer
RAM	Random access memory
vCPU	Virtual Central Processing Unit
URL	Uniform Resource Locator
UI	User interface
UX	User experience
UC	Use case
API	Application Programming Interface
USD	United States dollar
TCP	Transmission Control Protocol
SQL	Structured query language
GDPR	General Data Protection Regulation

Úvod

V dnešní době, kdy se svět neustále mění a přibývá množství nových objevů, je důležité propojovat akademický svět s průmyslem. Tato bakalářská práce se zaměřuje na vývoj nástroje, který pomůže lépe se orientovat ve velkém množství akademických publikací. Cílem je podpořit nejen vědecký výzkum, ale i zlepšit spolupráci mezi univerzitami a průmyslem, což může prospět naší fakultě.

Důležitost tématu je zřejmá z potřeby lépe porozumět a analyzovat vědecké informace. Výsledkem by mohlo být lepší propojení mezi teorií a praxí, což by mohlo pomoci najít nové možnosti pro společné projekty a spolupráce. Tento nástroj by mohl ukázat, kdo jsou hlavní odborníci v různých oborech a jaké nové myšlenky by mohly být užitečné pro vědu a průmysl.

Motivace k výběru tématu vychází z potřeby zefektivnit proces hledání informací a porozumění vědecké komunitě. V současné době, kdy je tolik informací, je klíčové mít nástroje, které pomohou rychle najít to, co potřebujeme a odhalit důležité vztahy mezi různými výzkumy.

Hlavním cílem této práce není vytvořit hotový komerční produkt, ale vyvinout první verzi nástroje, který ukáže, jak by se daly moderní techniky vizualizace využít pro zkoumání akademických publikací. Tento nástroj by měl sloužit jako základ pro další vývoj a testování v praxi. Jeho úkolem je usnadnit orientaci ve vědeckých pracích a podpořit hledání spolupráce mezi vědeckou komunitou a průmyslem.

Kromě toho se také práce zaměřuje na hodnocení ekonomického přínosu nástroje pro FIT ČVUT, vědeckou komunitu a zaměstnance fakulty. Taktéž se zpracuje budoucí rozvoj vyvinutého prototypu a posoudí se jeho proveditelnost.

Práce má následující strukturu. V první kapitole je rozebrána vizualizace dat jako taková a vybírá se vhodná vizualizace pro prototyp vizualizačního nástroje. Druhá kapitola se věnuje existujícím aplikacím a krátce popisuje postup práce s nimi. Třetí kapitola popisuje návrh vlastního řešení. Čtvrtá kapitola popisuje zvolené technologie pro implementaci vlastního řešení. Pátá kapitola se věnuje implementaci prototypu. V šesté kapitole je popsán postup

testování vyvinutého prototypu. Sedmá kapitola rozebírá ekonomický přínos nástroje pro FIT ČVUT. Osmá kapitola je věnována budoucímu rozvoji vizualizačního nástroje a studii jeho proveditelnosti.

Cíle práce

Cílem této bakalářské práce je vyvinout prototyp vizualizačního nástroje, jenž umožní efektivní zobrazení vztahů mezi autory a jejich pracemi. Tento nástroj by měl usnadnit uživatelům, jako jsou akademičtí pracovníci, výzkumníci, studenti a zaměstnanci fakulty, orientovat se lépe v komplexním světě akademických publikací a pochopit vzájemné vazby mezi autory.

V prvním kroku se práce soustředí na důkladnou specifikaci funkčních a nefunkčních požadavků. To zahrnuje určení, jaké vlastnosti musí nástroj obsahovat pro efektivní a přívětivou uživatelskou interakci, jak zajistit jeho rychlý a spolehlivý běh, a jaké technické požadavky musí splňovat pro bezpečné a škálovatelné využití.

Následuje fáze rešerše existujících vizualizačních nástrojů, jejímž cílem je porozumět současnému stavu technik a metod v této oblasti. Zjištění z této fáze pomůže identifikovat příležitosti pro zlepšení, které mohou být aplikovány při návrhu vlastního řešení.

Třetím krokem je samotný návrh vizualizačního nástroje, včetně rozhodnutí o implementační platformě. Tento proces zahrnuje výběr technologií, které umožní efektivní realizaci požadovaných funkcí a zároveň zajistí intuitivní práci s nástrojem.

Následuje implementace prototypu. V této fázi se teoretické plány a návrhy přemění na fungující software, jehož účinnost a správnost funkcionality bude ověřena testováním na reálných datech. Tímto testováním se zjistí, jak dobře nástroj plní svůj účel v praxi.

Dále práce přechází k hodnocení ekonomického přínosu systému. Tento krok se zaměří na posouzení, jak může nástroj přinést přidanou hodnotu pro fakultu nebo akademickou instituci, ať už jde o zlepšení výzkumných metod, usnadnění spolupráce nebo podporu vzdělávání.

Závěrem se práce věnuje možnostem dalšího rozvoje vizualizačního nástroje. Prozkoumá, jaké další funkce by mohly být přidány a zhodnotí jejich časovou náročnost a finanční náklady na jejich realizaci.

Tato bakalářská práce tak poskytne ucelený pohled na proces vývoje nástroje pro vizualizaci akademických vztahů, od počáteční analýzy potřeb

a možností až po praktickou realizaci a hodnocení přínosů. Práce má ambici nejen vytvořit užitečný nástroj pro akademickou komunitu, ale také přispět k rozvoji metod a technik vizualizace vědeckých dat.

Teoretická část

Kapitola 1
Vizualizace dat

Tato kapitola se věnuje vizualizaci dat. Stručně je popsán proces, který tento úkol obnáší. Jsou také stručně definovány pojmy z teorie grafů.

Vizualizace je proces převodu dat nebo informací do grafické podoby, která uživatelům umožňuje snadněji pochopit, analyzovat komplexní informace. Využívá grafické prvky jako jsou body, čáry, barvy a symboly k zobrazení vztahů a vzorců, které by mohly být v textové nebo numerické podobě obtížně interpretovatelné. Vizualizace dat nachází uplatnění v široké škále oborů, včetně vědy a průmyslu. Vizualizace je klíčová pro usnadnění rozhodování, hlubší porozumění komplexním informacím a efektivní komunikaci. Tento proces umožňuje zobrazit složité informace způsobem, který je rychle a snadno srozumitelný [1].

Vizualizovaná data lze rozdělit na dvě základní kategorie: kvantitativní a kvalitativní [2].

Kvantitativní data: mohou být měřena nebo počítána. Mohou reprezentovat cenu, rozměry nebo rychlost. Jsou opakem kvalitativních dat.

Kvalitativní data: jsou nečíselné hodnoty, které popisují vlastnosti nebo atributy. Mezi ně lze zařadit barvy, preference zákazníka nebo chutě. Jsou opakem kvantitativní dat.

Existuje mnoho různých způsobů, jak vizualizovat data, každý z nich se svými specifickými vlastnostmi hodí pro různé typy informací. Mezi běžné typy vizualizace patří grafy, jako jsou sloupcové, spojnicové, bodové a koláčové grafy. Pro složitější sady dat se často používají například teplotní mapy, síťové grafy nebo myšlenkové mapy.

Každá vizualizace může usnadnit pochopení dat a přispět k lepším rozhodnutím.

1.1 Výběr vhodné vizualizace

Na základě definice funkčních požadavků (viz. 3.1.1) byl pro vizualizační nástroj vybrán síťový graf. Tento typ grafu je vhodný pro vizualizaci a analýzu vztahů mezi různými entitami. V tomto případě se jedná o vztahy mezi autory a publikacemi.

1.1.1 Síťový graf

Nejdříve je nutné definovat, co graf vůbec je. Začne se tzv. *neorientovaným grafem.*

▶ **Definice 1.1** (Neorientovaný graf)**.** *[3] je uspořádaná dvojice (V, E), kde*

- *V je neprázdná konečná množina* ***vrcholů (také uzlů)****,*
- *E je množina* ***hran****.*

Hrana je dvouprvková podmnožina V (čili neuspořádaná dvojice vrcholů),

$$\textit{kterou značíme } \{u, v\}.$$

Obrázek 1.1 Neorientovaný graf s vrcholy X a Y

Neorientovaný graf obsahuje vrcholy a neorientované hrany. Tyto hrany jsou znázorněny čarou bez šipek, která indikuje, že mezi vrcholy není určený žádný směr. Pokud je nutné zvýraznit směr hrany mezi vrcholy, používají se tzv. *orientované grafy*, v nichž jsou hrany značeny šipkami ukazujícími směr od jednoho vrcholu k druhému.

▶ **Definice 1.2** (Orientovaný graf)**.** *[3] je uspořádaná dvojice (V, E), kde*

- *V je neprázdná konečná množina* ***vrcholů*** *a*
- *E je množina* ***orientovaných hran*** *(zobrazujeme jako šipky).*

Orientovaná hrana $(u, v) \in E$ *je uspořádaná dvojice vrcholů* $u, v \in V$.

Obrázek 1.2 Orientovaný graf s vrcholy X a Y

Síťový graf může být jak orientovaný, tak neorientovaný [4]. Volba mezi těmito dvěma typy je závislá na povaze vztahů, které je potřeba zobrazit. V tomto případě budou zobrazovány vztahy bez směru šipek, tedy neorientovaně. Vrcholy v grafu budou reprezentovat jednotlivé autory a publikace. Hrana mezi nimi bude symbolizovat vztah, že autor napsal danou publikaci. Tento přístup umožňuje vizualizovat a analyzovat vztahy mezi autory a jejich díly.

Kapitola 2

Existující vizualizační aplikace

Tato kapitola se věnuje rešerši podobných nástrojů, jejich vlastnostem a funkcím. Stručně je popsán postup práce s těmito nástroji a také jsou rozebrány ceny předplatného.

Za podobné nástroje se považují vizualizační aplikace, které na základě uživatelských dat poskytují tato data ve formě grafů. Jinými slovy, mohou být nápomocné při zkoumání dat tím, že poskytují jejich přehlednou vizualizaci. Zkoumané aplikace jsou k dispozici ve formě webových aplikací i jako klasické desktopové aplikace.

2.1 Connected Papers

Při psaní vědeckých publikací je nezbytné provádět literární rešerši, která vyžaduje procházení a analýzu existujících studií v daném oboru. Tento proces může být časově náročný a složitý, zejména kvůli obrovskému množství publikací a obtížnosti v identifikaci těch nejrelevantnějších. Nalezení publikací, které jsou přímo spojené s tématem bádání, vyžaduje důkladný výběr, který může být výzvou ve vědeckých oblastech, které se pohybují mezi různými obory.

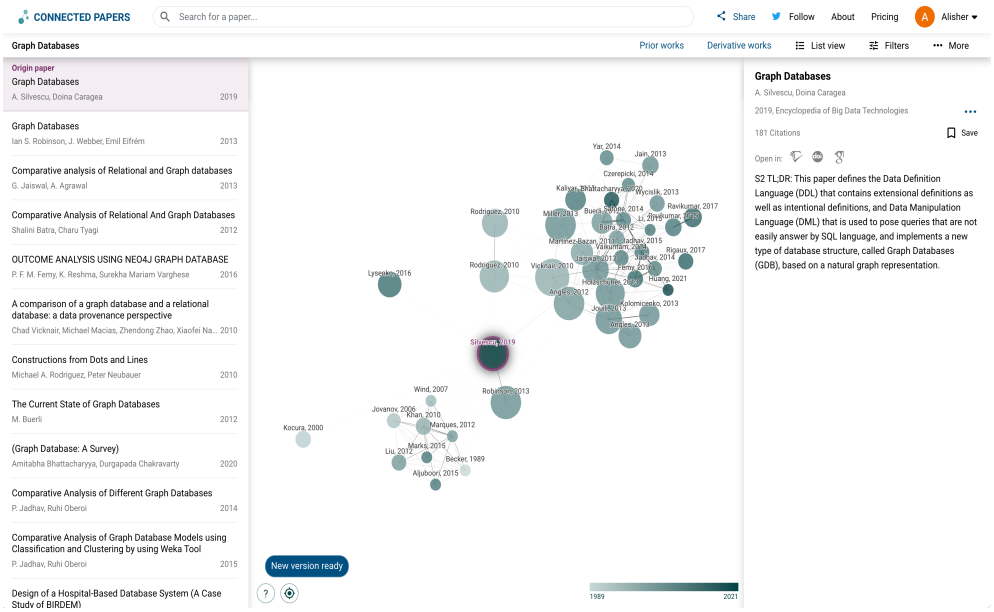
K řešení tohoto problému byl vytvořen nástroj Connected Papers, který se snaží vědce efektivně navigovat a pomáhat jim nalézat relevantní publikace. Connected Papers je webová aplikace určená k vizualizaci vědeckých prací a jejich vzájemných vztahů na základě citací. Tento nástroj umožňuje uživatelům objevovat nové články a zjišťovat, jak jsou jednotlivé práce propojeny, což ve vědecké komunitě velmi užitečné. Díky své schopnosti přehledně zobrazovat vědecké práce si Connected Papers získal popularitu mezi vědci a akademiky, kteří hledají inspiraci pro své další výzkumné projekty a spolupráce s dalšími vědci.

2.1.1 Vlastnosti a funkce

Connected Papers efektivně generuje síťový graf, v němž jednotlivé vrcholy představují vědecké publikace. Tyto vrcholy jsou vzájemně propojeny hranami, které reprezentují citace mezi publikacemi. Tento vizuální způsob zobrazení umožňuje uživatelům snadno identifikovat, jak jsou jednotlivé články vzájemně citovány, což napomáhá lepšímu porozumění vztahům a vlivům v rámci vědecké literatury.

2.1.1.1 Práce s nástrojem

Práce s nástrojem Connected Papers začíná vyhledáváním určité publikace, které se provádí pomocí názvu, klíčových slov, nebo DOI (Digital Object Identifier). Po zadání hledaného výrazu systém zobrazí seznam relevantních výsledků, z nichž uživatel může vybrat konkrétní publikaci. Po výběru publikace nástroj automaticky generuje síťový graf, který zobrazuje vztahy mezi vybranou publikací a dalšími souvisejícími pracemi. Graf ilustruje, jak jsou jednotlivé články propojeny na základě jejich vzájemných citací a tematické příbuznosti. Na obrázku níže je příklad grafu vytvořeného pro publikaci s názvem *Graph Databases*, která věnuje tématu grafových databází [5].



Obrázek 2.1 Connected Papers: Vizualizace pro určitou publikaci [6]

2.1.1.2 Síťový graf

Vrcholy v tomto grafu představují jednotlivé publikace, které jsou propojeny neorientovanými hranami založenými na citacích mezi nimi. Publikace

s tématickou podobností jsou umístěny blíže k sobě a hrany mezi nimi jsou zvýrazněny silnějšími barvami, což naznačuje jejich bližší vztah.

Velikost každého vrcholu je určena počtem citací, což uživatelům pomáhá rychle identifikovat klíčové a vlivné práce v daném grafu. Barva vrcholu představuje rok publikování – nejnovější publikace mají tmavší odstín, zatímco starší práce jsou zobrazeny světlejší barvou.

Po kliknutí na libovolnou publikaci nástroj zobrazí nejkratší cestu k této publikaci od zvoleného výchozího dokumentu, pro který byl síťový graf vygenerován. Na levé straně obrazovky se zobrazuje seznam všech publikací zobrazených v grafu, který usnadňuje navigaci a umožňuje rychlý přístup k detailům o jednotlivých pracích.

2.1.1.3 Detail vybrané publikace

Po kliknutí na publikaci se na pravé straně obrazovky zobrazí detaily o vybrané práci. Detail zahrnuje název publikace, datum jejího vydání a seznam autorů. Dále obsahuje informace jako počet citací, odkazy na externí systémy pro publikování vědeckých prací (např. arXiv) nebo na systémy pro vyhledávání vědeckých prací (např. Google Scholar). Nedílnou součástí detailu je také abstrakt publikace, který poskytuje rychlý přehled o obsahu a zaměření studie.

Jednou z dalších funkcí je možnost vygenerovat nový síťový graf pro vybranou publikaci, která uživatelům umožňuje hlouběji prozkoumat související práce a citace. V bezplatné verzi aplikace je však počet grafů, které lze vygenerovat, omezen, a pro neomezený přístup je nutné zakoupit předplatné.

2.1.2 Cena předplatného

Následující tabulka ukazuje, kolik stojí předplatné Connected Papers pro různé účely a kolik grafů měsíčně je pro tyto účely možné vygenerovat. Informace jsou aktuální k okamžiku psaní této práce [7].

Tabulka 2.1 Předplatné Connected Papers

Typ uživatele	Měsíční cena	Grafů měsíčně	Roční cena
účet zdarma	0 USD	5 grafů	0 USD
akademické účely	3 USD	neomezeně	36 USD
průmysl	10 USD	neomezeně	120 USD

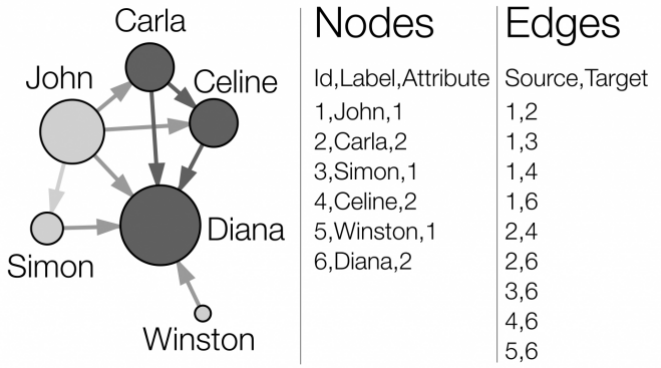
Aplikace umožňuje i skupinové předplatné pro ty, kdo by chtěli nástroj využívat jako tým. Pro ně platí stejné ceny jako pro individuální uživatele, ale jsou vynásobeny počtem lidí v týmu.

2.2 Gephi

Gephi je komplexní vizualizační software určený k vizualizaci grafů, který si klade za cíl pomoci uživatelům lépe porozumět komplexním datům. Tento nástroj umožňuje širokou škálu manipulací s grafy, včetně možnosti nastavit rozložení grafu, konfigurovat barvy, velikost vrcholů, čímž uživatelům umožňuje vizuálně zdůraznit klíčové aspekty dat [8].

2.2.1 Práce s nástrojem

Jedná se o desktopovou aplikaci. V Gephi je síťový graf tvořen dvěma základními prvky – seznamem vrcholů a seznamem hran. Grafy mohou být jak orientované, tak neorientované, a umožňují uživatelům vizualizovat a analyzovat různé typy vztahů a struktur dat. Vrcholy grafu představují jednotlivé entity, zatímco hrany ukazují vztahy mezi těmito entitami. Na obrázku níže je ilustrovaný jednoduchý síťový graf, který demonstruje síťový graf v Gephi [9].



Obrázek 2.2 Gephi: Síťový graf [9]

Na obrázku výše vidíme základní definice vrcholů a hran, které tvoří strukturu síťového grafu v nástroji Gephi. Do Gephi se tyto dva typy dat nahrávají zvlášť jako CSV soubory. V tomto případě vrcholy mají tři hlavní vlastnosti: unikátní identifikátor vrcholu, štítek, který značí jméno dívky nebo chlapce, a atribut reprezentující barvu vrcholu. Co se týče hran, ty nutně obsahují zdroj a cíl, přičemž každý z těchto prvků je definován identifikátorem vrcholu. Pokud bychom chtěli v Gephi pracovat s neorientovanými hranami, je třeba do souboru s hranami přidat sloupec s názvem „Type“ a přiřadit hodnotu „Undirected“ pro hranu.

2.2.2 Cena předplatného

Gephi je software, který je zcela zdarma a open-source. To znamená, že uživatelé nemusí platit žádné předplatné a mají plný přístup ke všem funkcím

a aktualizacím. Jako *open-source* nástroj mohou uživatelé také přispívat k jeho vývoji nebo upravovat jeho kód podle svých specifických potřeb.

2.3 Shrnutí

Tabulka níže shrnuje vlastnosti diskutovaných nástrojů. „Snadnost“ znamená, jak snadné je daný nástroj používat a jak rychle se uživatel naučí ho ovládat. „Zdarma“ indikuje, zda nástroj nevyžaduje předplatné. „Přívětivost“ odkazuje na uživatelskou přívětivost nástroje. V případě Gephi musí uživatel být obeznámen s pojmy z teorie grafů, a je poměrně těžké najít políčka, která uživatel musí zaškrtnout, aby vizualizace dávala smysl. Proto byl tento nástroj ohodnocen nízkou přívětivostí.

Tabulka 2.2 Porovnání vizualizačních nástrojů Gephi a Connected Papers

Nástroj	**Snadnost**	**Zdarma**	**Přívětivost**
Gephi	Střední	Ano	Nízká
Connected Papers	Vysoká	Ne	Vysoká

Kapitola 3
Návrh vlastního řešení

V této kapitole je prezentován návrh vizualizačního nástroje. Detailně jsou popsány způsoby, jakými byly stanoveny funkční a nefunkční požadavky a proč byla vybrána konkrétní implementační platforma.

3.1 Požadavky

Nyní jsou definovány požadavky pro návrh a vývoj aplikace.

Funkční požadavky popisují konkrétní funkce a chování, které software musí nabízet, například způsob, jakým software zpracovává data nebo interaguje s uživatelem. Jsou to přímo úkoly, které software musí vykonávat.

Nefunkční požadavky se zaměřují na atributy jako výkon, bezpečnost, škálovatelnost a uživatelská přívětivost. Tyto požadavky definují kvalitu a omezení systému, nikoliv jeho specifické chování. Výběr specifických technologií také spadá pod nefunkční požadavky, protože ovlivňuje, jak systém splňuje očekávání v oblastech jako jsou výkon, kompatibilita a bezpečnost.

3.1.1 Funkční požadavky

F1: Vizualizace zobrazuje graf autorů a jejich publikace
Rozhraní aplikace musí zobrazovat graf, ve kterém jsou autoři a jejich publikace reprezentováni jako uzly. Každý autor a každá publikace jsou zobrazeny jako samostatné uzly. Pokud autor napsal určitou publikaci, mezi odpovídajícími uzly autora a publikace bude vytvořena hrana, indikující tento vztah. Toto uspořádání umožňuje uživatelům vizuálně analyzovat spojení mezi autory a jejich publikacemi.

Priorita: Vysoká

Typ: Funkční požadavek

Složitost: Střední

F2: Vizualizace umožňuje interaktivitu
Rozhraní aplikace musí umožňovat interaktivní manipulaci s uzly grafu, které reprezentují autory a jejich publikace. Uživatelé by měli být schopni posouvat jednotlivé uzly a přibližovat je, což umožní podrobnější analýzu vztahů mezi autory a jejich pracemi.

Priorita: Vysoká

Typ: Funkční požadavek

Složitost: Střední

F3: Aplikace umožňuje nahrát graf
Aplikace musí umožnit nahrání grafu, kde graf je reprezentován formátem CSV. Uživatelé by měli mít možnost importovat data v CSV formátu, která obsahují informace o autorech, publikacích a vztazích mezi nimi, aby mohli tato data vizualizovat jako graf v rozhraní aplikace. Tento proces importu zajišťuje, že uživatelé mohou snadno a efektivně načítat a aktualizovat grafy založené na svých specifických datasetech.

Priorita: Vysoká

Typ: Funkční požadavek

Složitost: Vysoká

F4: Vizualizace zobrazuje spoluautory
Rozhraní aplikace musí nabízet možnost zobrazení všech prací a spoluautorů konkrétního autora, kterého si uživatel vybere. Po výběru autora aplikace automaticky zobrazí seznam jeho publikací a jména všech spoluautorů těchto prací. Toto zobrazení umožní uživatelům podrobněji prozkoumat akademický přínos a síťové spojení vybraného autora, což usnadňuje analýzu jeho vědeckého vlivu a kolaborativních vztahů.

Priorita: Vysoká

Typ: Funkční požadavek

Složitost: Střední

3.1.2 Nefunkční požadavky

NF1: Webová aplikace
Aplikace musí být implementována jako webová aplikace. Tento požadavek zajistí, že uživatelé mohou přistupovat k aplikaci přímo přes webový prohlížeč bez potřeby jakékoliv další instalace, což zvyšuje snadnost přístupu a používání.

Priorita: Vysoká

Typ: Nefunkční požadavek

Složitost: Střední

NF2: Rozhraní aplikace v anglickém jazyce
Rozhraní aplikace musí být v anglickém jazyce. Tento požadavek zaručuje, že aplikace bude přístupná a srozumitelná pro mezinárodní uživatele, což rozšiřuje její potenciální uživatelskou základnu a podporuje globální použitelnost.

Priorita: Vysoká

Typ: Nefunkční požadavek

Složitost: Nízká

NF3: Vizualizace je implementována pomocí Neovis.js
Vizualizace v aplikaci musí být implementována pomocí knihovny Neovis.js, která byla vybrána na základě doporučení oficiálních stránek Neo4j [10].

Priorita: Vysoká

Typ: Nefunkční požadavek

Složitost: Nízká

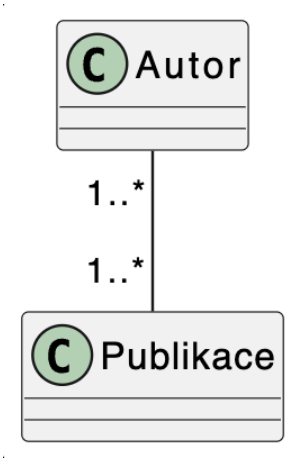
3.2 Doménový model

Doménový model je konceptuální reprezentace dat a vztahů mezi nimi v určité oblasti, která slouží k pochopení a organizaci domény problému. Model zahrnuje entity, jejich atributy a relace, aby odpovídaly konkrétním potřebám [11].

V této části budou popsány třídy (entity) a relace mezi nimi v rámci doménového modelu aplikace. Atributy tříd nebudou detailně uváděny, protože cílem je hlavně popsat relace mezi jednotlivými třídami. Klíčové atributy budou zmíněny v popisu níže.. Obrázek 3.1 níže ukazuje UML diagram popisující třídy a jejich vzájemné relace.

Autor *Autor* reprezentuje jednotlivce, kteří se podílejí na tvorbě vědeckých publikací. Autoři jsou odlišováni podle jejich jména.

Publikace *Publikace* reprezentuje vědecké práce nebo články, které byly napsány jedním nebo více autory. Publikace se rozlišují svými názvy.



Obrázek 3.1 Digram tříd: doménový model

3.3 Případy užití

Případy užití (*use cases*) je technika používaná v softwarovém inženýrství k definování funkcí aplikace z pohledu koncových uživatelů (*aktéři*). Poskytují náhled na to, jak aplikace funguje skrze různé scénáře, které popisují interakce mezi uživatelem a systémem [12]. V diagramu případů užití jsou aktéři obvykle umístěni na levé straně a případy užití, které ukazují funkce, s nimiž mohou aktéři pracovat, jsou na pravé straně.

3.3.1 Aktéři

Aktér v případech užití může být jednotlivec, skupina lidí, externí systém nebo dokonce čas jako periodické události spouštějící funkce. Tato práce se zaměřuje na interakce s aplikací pro nepřihlášené uživatele(*anonymní uživatelé*). Během psaní této práce nebylo navrženo rozdělení na uživatelské role, ani nebyly navrženy a implementovány autentizační mechanismy.

Nepřihlášený uživatel reprezentuje cílovou skupinu uživatelů

3.3.2 Vizualizace grafem

3.3.2.1 UC1: Vizualizace autorů

Vizualizační nástroj kreslí klíčové entity. Zde se jedná o *autory*.

Aktéři: Nepřihlášený uživatel

Počáteční podmínka: Aktér se nachází na vybraném grafu

Scénář:

1. Systém vykresluje autory ve formě uzlů grafu
2. Aktér vyhledává autory dle svých zájmů

3.3.2.2 UC2: Vizualizace publikací

Vizualizační nástroj kreslí klíčové entity. Zde se jedná o *publikace*.

Aktéři: Nepřihlášený uživatel

Počáteční podmínka: Aktér se nachází na vybraném grafu

Scénář:

1. Systém vykresluje publikace ve formě uzlů grafu
2. Aktér vyhledává publikace dle svých zájmů

3.3.2.3 UC3: Vizualizace grafu autorů a jejich publikací

Vizualizační nástroj kreslí vztahy mezi *publikacemi* a *autory*. Pokud *autor* napsal danou *publikaci*, vizualizační nástroj propojí dva uzly hranou.

Aktéři: Nepřihlášený uživatel

Počáteční podmínka: Aktér se nachází na vybraném grafu

Scénář:

1. Systém vykresluje graf autory, publikace a relace mezi nimi
2. Aktér prozkoumává výsledný graf dle svých potřeb

3.3.2.4 UC4: Vypisuje publikace vybraného autora a jeho spoluautory

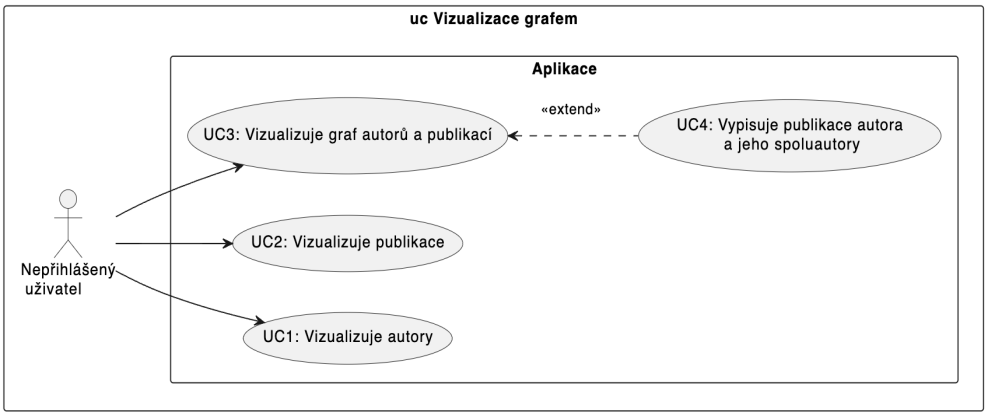
Vizualizační nástroj vypisuje *publikace* vybraného *autora* a spoluautory, kteří se podíleli na této *publikaci*

Aktéři: Nepřihlášený uživatel

Počáteční podmínka: Aktér se nachází na vybraném grafu

Scénář:

1. Systém vykresluje graf autory, publikace a relace mezi nimi
2. Aktér si vybere nějakého autora, o kterého má zájem
3. Systém vypíše publikace autora a všechny spoluautory pro každou publikaci
4. Aktér prozkoumává výpis dle svých zájmů



Obrázek 3.2 Diagram případu užití: Vizualizace grafem

3.3.3 Vypisování vizualizací

3.3.3.1 UC1: Vypisování seznamu nahraných vizualizací

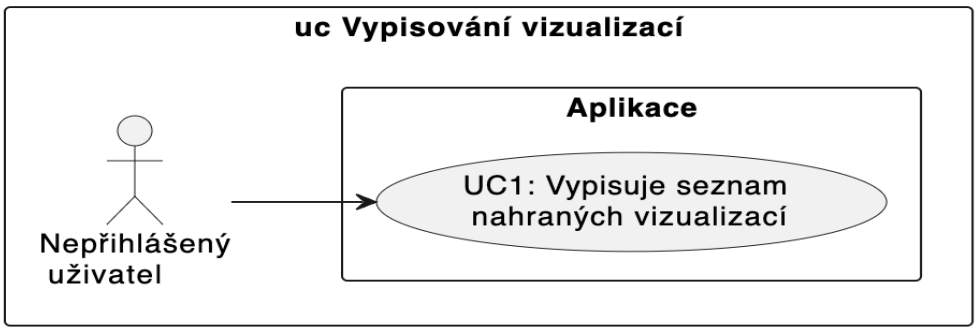
Systém vypisuje seznam uložených vizualizací.

Aktéři: Nepřihlášený uživatel

Počáteční podmínka: Aktér se nachází na seznamu vizualizací

Scénář:

1. Systém vypisuje seznam uložených vizualizací
2. Aktér si vybere konkrétní vizualizaci podle dle svých zájmů



Obrázek 3.3 Diagram případu užití: Vypisování vizualizací

3.3.4 Uložení vizualizace

3.3.4.1 UC1: Ukládání vizualizace

Jedná se o CSV soubor, který nahraje aktér. Systém posléze nabídne vizualizaci grafem.

Aktéři: Nepřihlášený uživatel

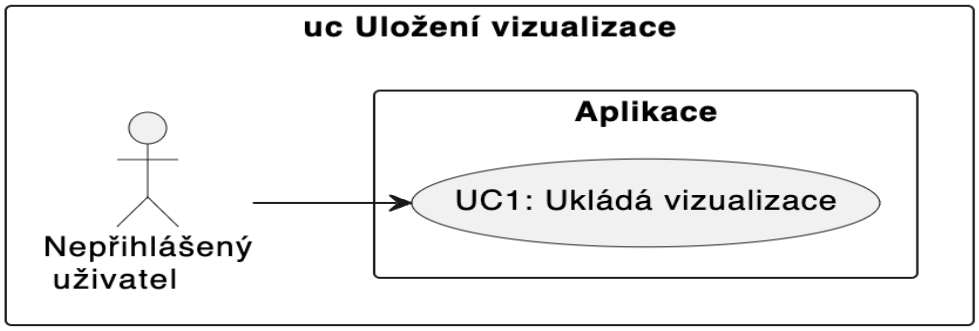
Počáteční podmínka: Aktér se nachází v sekci pro nahrávání

Scénář:

1. Aktér si nahraje soubor
2. Systém zpracuje soubor
3. Systém uloží data a nabídne vizualizaci

Alternativní Scénář:

3. Systém vrátí chybu a nahlásí aktérovi



Obrázek 3.4 Diagram případu užití: Uložení vizualizace

3.4 Uživatelské rozhraní

Nyní je věnována pozornost návrhu uživatelského rozhraní (*User interface*, též jako UI) aplikace. S UI je spojená i uživatelská zkušenost (*User experience*, též jako UX), která klade důraz na uživatelskou přívětivost a dostupnost. Tyto dvě techniky mají za cíl umožnit uživatelům snadno navigovat, manipulovat a interpretovat data zobrazená v grafu.

3.4.1 Nielsonova Heuristika

V průběhu let se Nielsonovy heuristické principy staly velmi osvědčenými a jejich aplikace v různých projektech po celém světě dokázala, že zlepšují použitelnost a zvyšují uživatelskou spokojenost. Tyto principy, které byly vyvinuty na základě rozsáhlých výzkumů a praktických testů, se opravdu osvědčily

v praxi. V dnešní době je proto běžně doporučováno, aby se tato pravidla dodržovala při tvorbě uživatelských rozhraní. Jedná se o sadu 10 pravidel [13].

1. **Viditelnost stavu systému** Uživatel musí mít vždy přehled o aktuálním stavu aplikace, zda čeká na uživatelský vstup nebo zpracovává data, což může být ukázáno například ikonou rotujícího kolečka nebo přesýpacích hodin.

2. **Shoda mezi systémem a reálným světem** Systém by měl používat jazyk srozumitelný pro uživatele a svým návrhem by měl napodobovat reálný svět. Například, ikona koše by měla vypadat jako skutečný odpadkový koš a funkce přetažení souboru do koše by měla soubory opravdu odstranit.

3. **Uživatelská kontrola a svoboda** Systém by měl umožnit uživatelům jednoduše se vrátit zpět z aktuálního stavu nebo zrušit prováděnou akci. Tato funkce je obvykle zajištěna pomocí tlačítek nebo odkazů označených jako „zpět“ nebo „storno“.

4. **Konzistence a standardy** Systém by měl být konzistentní vizuálně i ovládáním. Názvy stejných funkcí by měly být jednotné, aby nedocházelo k záměně, například mezi „uložit“ a „upravit“.

5. **Prevence chyb** Systém by měl aktivně zabránit chybám, například použitím potvrzovacích dialogů a varování. Uživatele je třeba upozornit, pokud nevyplní povinné položky formuláře, nebo před smazáním neprázdného adresáře.

6. **„Mrknu a vidím“** Aby bylo užívání systému co nejméně náročné, systém by měl zobrazovat jen relevantní volby a skrýt nebo zneplatnit nepoužitelné možnosti.

7. **Flexibilita a efektivita použití** Systém by měl být jednoduchý, ale také nabízet dostatek funkcí pro pokročilé uživatele, včetně klávesových zkratek a automatického doplňování polí.

8. **Estetický a minimalistický** Systém by měl zobrazovat jen nezbytné informace a volby pro rychlou a přehlednou práci.

9. **Pomoc při opravě chyb** Chybové zprávy by měly být jasné a přesné, poskytovat jednoduché řešení problémů a vysvětlovat, co se stalo.

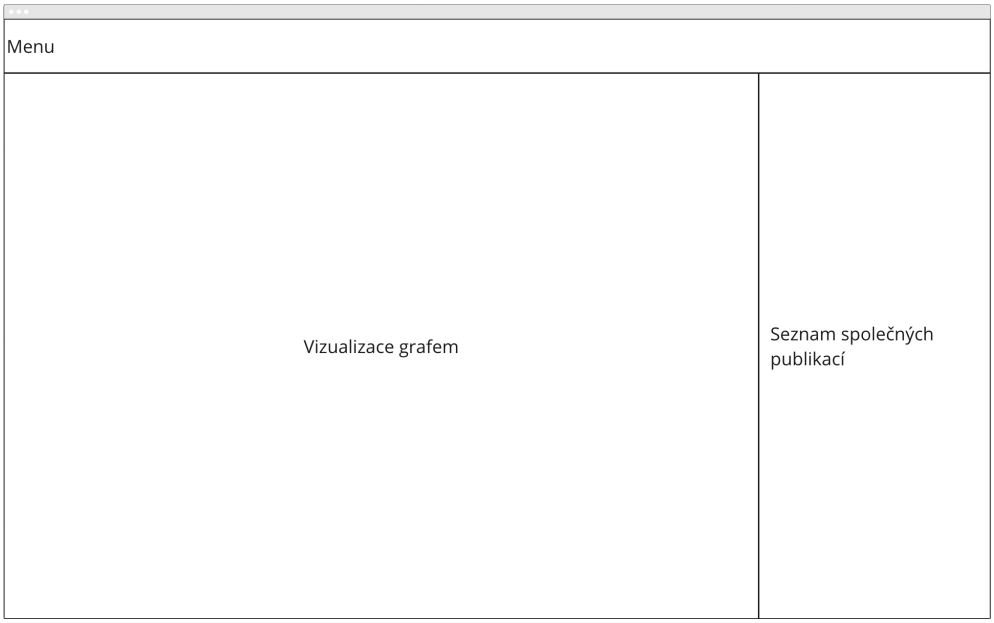
10. **Nápověda a dokumentace** I když je ideální, když systém může být používán bez nápovědy, měla by být dostupná užitečná nápověda a dokumentace, která je zaměřená na uživatelské úkoly a snadno přístupná.

3.4.2 Dratěný model

Při návrhu vizualizačního nástroje byl použit drátěný model (wireframe), který sloužil jako základní vizuální průvodce pro strukturu a rozložení aplikace. drátěný model se umožnil zaměřit na uspořádání obsahu, funkcionality a navigace. Využití drátěného modelu v raných fázích návrhu pomohlo rychle experimentovat s různými uspořádáními a efektivně optimalizovat uživatelské rozhraní. Wireframe také usnadňuje komunikaci mezi členy vývojového týmu a zainteresovanými stranami [14].

3.4.2.1 Low-fidelity

Nízkoúrovňový (*Low-fidelity* nebo *low-fi*) drátěný model, je jednoduchá vizuální reprezentace uživatelského rozhraní, která se zaměřuje na základní rozložení a strukturu, nezahrnuje detailní grafiku, barvy ani styly. Tyto modely jsou často rychle malovány ručně nebo s pomocí jednoduchého softwaru, který umožňuje snadno provádět změny [15].



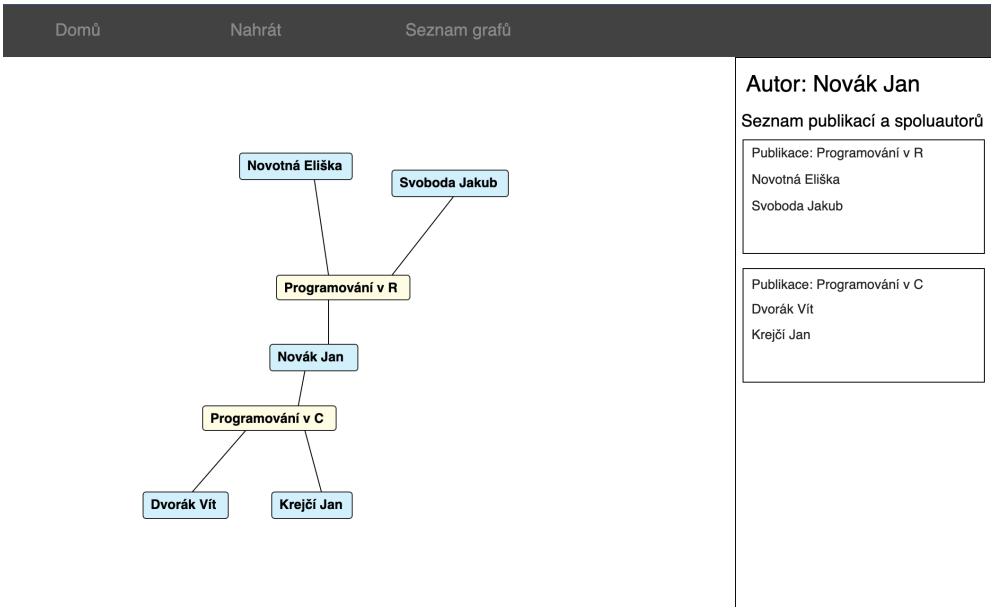
Obrázek 3.5 Dratěný model low-fidelity: Navržený vzhled aplikace [16]

Ve vizualizačním nástroji je na levé straně umístěna prostorná vykreslovací plocha, kde jsou zobrazovány vizualizace vztahů mezi autory a jejich publikacemi. V horní části je umístěna navigační lišta, která umožňuje snadné ovládání a přecházení mezi funkcemi nástroje. Na pravé straně je vyhrazen prostor pro výpis detailních informací, kde jsou poskytovány podrobné údaje o vybraných autorech a jejich pracích.

3.4.2.2 High-fidelity

Vysokoúrovňový (*High-fidelity* nebo *high-fi*) drátěný model je detailní prototyp, který je velmi podobný finálnímu vzhledu aplikace nebo webové stránky. Tento typ prototypu zahrnuje vizuální prvky, barvy, typografii, tlačítka a navigace [15].

Níže je uvedený podrobnější návrh finálního vzhledu aplikace. Zde můžeme pozorovat příklad vizualizace grafem a výpis publikací a spoluautorů pro autora Jana Nováka.



Obrázek 3.6 Dratěný model high-fidelity: Navržený vzhled aplikace [16]

Kapitola 4

Implementační platforma

V této kapitole bude rozebráno, které technologie byly zvoleny pro implementaci vizualizačního nástroje. Bude vysvětleno, proč byly konkrétní technologie vybrány a jak přispívají k celkové funkcionalitě a efektivitě nástroje.

Při výběru technologií pro vizualizační nástroj bylo zohledněno několik klíčových aspektů. Vybrané technologie jsou open-source a zdarma, což umožňuje snadné úpravy a eliminuje licenční poplatky. Dalším rozhodujícím faktorem byla aktivní komunita uživatelů, která podporuje rychlé řešení problémů. Posledním a nejméně důležitým faktorem byly zkušenosti autora s těmito technologiemi, které rovněž usnadnily efektivní vývoj. Tyto faktory vytvořily pevný základ pro výběr technologií pro projekt.

Vizualizační nástroj je strukturován do dvou hlavních částí – backend a frontend. Backend je zodpovědný za zpracování a ukládání dat, které zahrnuje transformaci dat a jejich uložení v databázi (serverová část). Na druhé straně, frontend slouží jako uživatelské rozhraní, kde jsou data vizualizována (klientská část). Umožňuje uživatelům nejen prohlížet výsledky ve formě grafů, ale také ukládat již nahrané vizualizace. Tato rozdělenost umožňuje efektivní správu a prezentaci dat, zatímco udržuje systém organizovaný a optimalizovaný pro uživatelské interakce.

Na straně serveru je vyžadováno efektivní zpracování uživatelských dat a jejich ukládání. To je nezbytné pro správnou funkčnost a rychlou odezvu vizualizačního nástroje. Tyto operace vyžadují technologii, která je schopná rychle zpracovat a reagovat na požadavky uživatelů. Níže bude zdůvodněno, proč byla vybrána zrovna technologie Node.js.

Na straně klienta se klade důraz na efektivní vizualizaci a interakci s uživatelem, což je klíčové pro uživatelskou přívětivost nástroje. Tato část systému vyžaduje technologie schopné vykreslovat grafy a zpracovávat uživatelské vstupy. Níže bude objasněno, jaké technologie byly zvoleny pro frontend, aby se zajistilo, že vizualizace jsou atraktivní a srozumitelné, což zvyšuje celkovou efektivitu prozkoumávání a analýzy dat.

4.1 Serverová část

Serverová část obsahuje dvě hlavní části. První z nich je Node.js server, který zpracovává uživatelské požadavky a je navržen jako REST API server. Druhou komponentou je databáze Neo4j, která slouží jako uložiště dat a umožňuje efektivní dotazování na tato data.

4.1.1 Neo4j

Neo4j je grafová databáze, která se specializuje na efektivní manipulaci a ukládání dat reprezentovaných ve formě grafů. Tato databáze je navržena na zpracování komplexních dotazů na propojená data, čímž umožňuje rychlé procházení grafů. Neo4j používá dotazovací jazyk Cypher, který je speciálně navržen pro práci s grafovými strukturami, který zjednodušuje vytváření a čtení dotazů na grafy. Neo4j je populární pro využití v doporučovacích systémech, sociálních sítích a složitých analytických platformách. Jedná se o nejpopulárnější grafovou databázi [17].

Pro uložení dat se využívá tzv. grafový model (viz 4.1.1.1), kde entity jsou ukládány jako vrcholy grafů. Každý vrchol může obsahovat libovolný počet dvojic ve tvaru klíč-hodnota, které reprezentují vlastnosti entity. Vztahy mezi entitami jsou reprezentovány hranami, které rovněž mohou nést informace ve tvaru dvojic klíč-hodnota, aby lépe vystihovaly povahu vztahu.

Tento způsob ukládání dat se výrazně liší od klasických relačních SQL databazí, kde se data ukládají do tabulek. V tradičních relačních databázích se vztahy mezi daty dopočítávají prostřednictvím spojení tabulek (operace JOIN), což může být velmi výpočetně náročné. V grafové databázi je tento nedostatek eliminován, neboť vztah mezi entitami je uložen přímo a není třeba ho dopočítávat. Tento způsob umožňuje rychlejší a efektivnější zpracování dotazů na propojená data.

4.1.1.1 Grafový model

V databázi Neo4j jsou informace organizovány jako vrcholy, vazby (také hrany) a vlastnosti. Vrcholy reprezentují entity, jako jsou autoři a publikace, zatímco vazby vyjadřují vztahy mezi těmito entitami, například spoluautorství nebo citační odkazy. Vlastnosti poskytují dodatečné informace o vrcholech a hranách, jako jsou názvy publikací, jména autorů, počet citací a další relevantní data [18].

Vrcholy jsou entity v grafu. Pro rozlišení entit se používá tzv. *label*, jako například „Autor“ nebo „Publikace“ Každý vrchol může obsahovat libovolný počet *label*ů.

Vazby jsou orientované hrany mezi vrcholy a vždy obsahují směr, avšak při dotazování mohou být navigovány v libovolném směru bez ztráty výkonu.

Stejně jako vrcholy, i vazby mohou obsahovat vlastnosti, což umožňuje přidávat dodatečné informace o charakteru vztahu mezi entitami. Vrcholy mohou mít libovolný počet vazeb.

4.1.2 Node.js

Node.js je JavaScript prostředí, které umožňuje vývojářům tvořit webové servery nebo programy pro příkazový řádek přímo v jazyce JavaScript. Tento nástroj je populární [19] mimo jiné kvůli své schopnosti obsloužit mnoho současně připojených klientů. Tato vlastnost poskytuje lepší škálovatelnost aplikací [20].

Je důležité zmínit NPM[1], což je balíčkovací systém pro Node.js. NPM umožňuje instalovat a spravovat závislosti projektu, ale také publikovat vlastní balíčky. Tento nástroj je součástí téměř každé aplikace napsané v Node.js, protože efektivně řeší externí knihovny a moduly pro vývoj aplikací [21].

V tomto případě Node.js poskytne možnost zpracovávat uživatelská data a vyřizovat jejich požadavky. Zajistí také přístup k databázi Neo4j, který umožní manipulaci a čtení uložených dat. Balíčkovací systém NPM poskytne open-source knihovny, které usnadní vývoj aplikace tím, že umožní instalovat závislosti bez nutnosti vytvářet vše od začátku.

4.2 Klientská část

Jako obecný přístup byl zvolen vývoj *single-page application* (SPA). Jedná se o architekturu webových aplikací, ve které se ze serveru načte jeden HTML dokument. Jeho tělo se poté přizpůsobuje a aktualizuje dle interakcí s uživatelem [22]. To umožňuje JavaScript. SPA architektura se volí v případě, že je potřeba poskytovat uživateli komplexní uživatelské rozhraní [23]. Komunikace se serverem probíhá pomocí webových API.

Na rozdíl od klasických webových stránek, kde téměř veškerá logika probíhá na serveru a klientovi je vracen hotový HTML dokument, SPA poskytuje dynamickou interaktivitu přímo na straně klienta. Tento přístup klade za cíl minimalizovat dobu načtení stránek a zvýšit rychlost odezvy aplikace.

4.2.1 React

React je JavaScript knihovna, která umožňuje vývoj SPA a je velmi populární díky velké komunitě uživatelů [19]. Byla vyvinuta společností Facebook. Uživatelské rozhraní v Reactu se skládá z tzv. komponent, které mohou reprezentovat různé uživatelské prvky jako formuláře, textová pole, tlačítka, obrázky

[1] JavaScript package manager. Oficiální stránky NPM popírají, že NPM znamená Node Package Manager

a další. Tyto komponenty umožňují rozdělit uživatelské rozhraní na nezávislé a znovu použitelné části.

Nedílnou součástí knihovny React je JavaScriptové rozšíření JSX, které umožňuje psát kód velmi podobný HTML. Toto rozšíření přináší další výhody, jako je například možnost řízení stavu komponenty nebo vykreslování obsahu proměnných [24].

Níže je demonstrována komponenta s názvem `SimpleComponent`, která vrací odstavec textu, ve kterém je proměnná *message* nahrazena za obsah této proměnné. Tento příklad ukazuje, jak lze v Reactu pomocí JSX snadno integrovat proměnné do HTML dokumentu, čímž umožňuje dynamické vykreslování obsahu na webové stránce. Posléze se tato komponenta dá použít v jiné komponentě zavoláním `<SimpleComponent />`.

Výpis kódu 4.1 Ukázka React komponenty

```
function SimpleComponent() {
    const message = "Some value"
    return (
        <p>Value is: {message}</p>
    )
}
export default SimpleComponent;
```

Hooks jsou funkce, které umožňují komponentám používat stav, optimalizaci výkonu, navigace mezi stránkami a další funkcionality. Byly představeny ve verzi React 16.8.0 [25]. Funkce hooks lze kombinovat a skládat do specifických funkcí, které pak mohou být opakovaně použity v různých částech aplikace, tím vzniká tzv. *Custom hook* funkce [26].

4.2.2 Neovis.js

Neovis.js je JavaScript knihovna, která umožňuje vizualizaci grafů z dat databáze Neo4j. Na pozadí využívá populární knihovnu vis.js, která ji umožňuje zobrazovat složitější grafy. Knihovna poskytuje možnost připojení přímo do databáze Neo4j a nabízí široké možnosti konfigurace vizualizace. Uživatelé mohou upravovat vzhled grafu, včetně barev a velikostí vrcholů a hran. Neovis.js také poskytuje možnosti pro interakci s grafem, jako jsou přiblížení, posouvání a výběr konkrétních vrcholů či hran, tím umožňuje uživatelům podrobně prozkoumat graf a vztahy v něm [10].

4.3 TypeScript

Na serverové i klientské části bude použit jazyk TypeScript. TypeScript umožňuje definovat rozhraní, třídy a vlastní typy. Kompilátor TypeScriptu převádí kód do JavaScriptu, který lze spustit v prohlížeči nebo v Node.js. Platí, že kód napsaný v JavaScriptu je validní TypeScript, avšak opačně to neplatí.

Praktická část

Kapitola 5

Implementace prototypu

Tato kapitola se věnuje implementaci prototypu. Rozebrána je jak klientská, tak i serverová část. Ukázána je i implementace databázového modelu a stručně jsou popsány dotazy.

5.1 Struktura systému

Níže je ilustrována struktura systému (viz. 5.1), ve které je uživatel schopen prostřednictvím webové aplikace plnit své úkoly. Webová aplikace posílá HTTP požadavky na backendový systém a dostává výsledky, které poskytne uživateli. Data jsou uložena v grafové databázi.

Node.js Server: zpracovává uživatelské požadavky a vrací výsledky. Je navržen jako REST API (viz. 5.3.1) server.

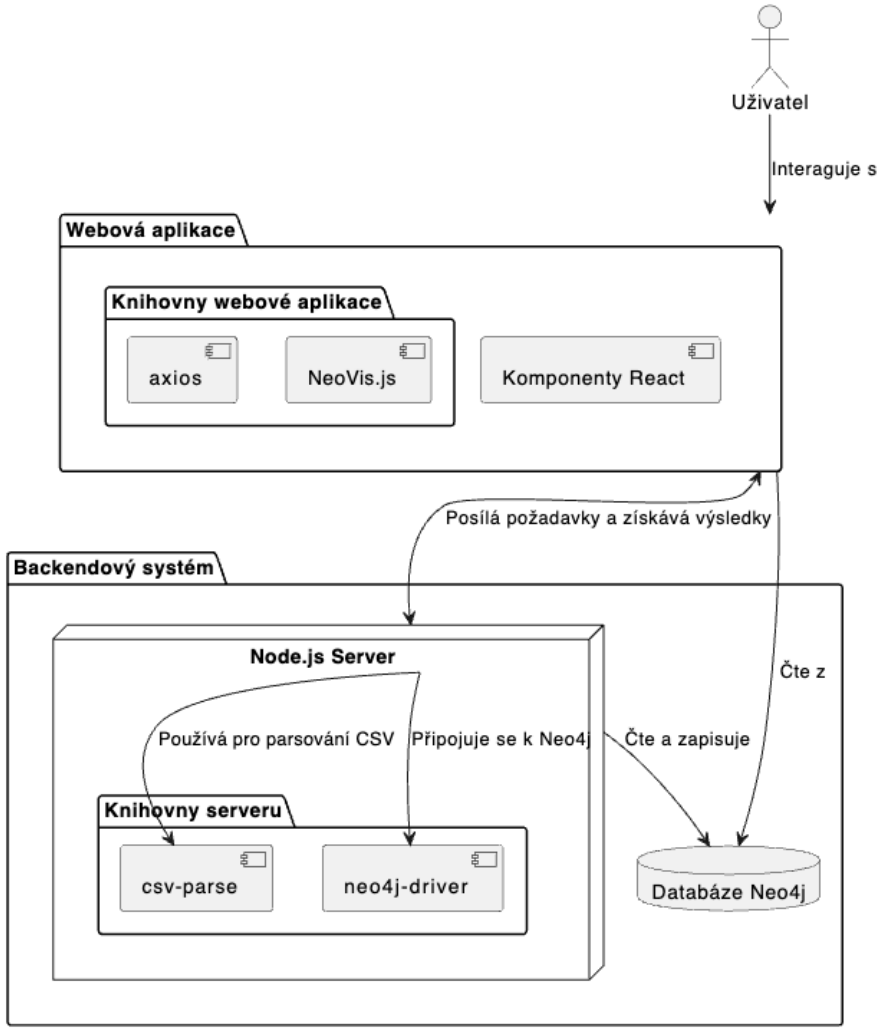
Webová aplikace: poskytuje uživatelské rozhraní a zajišťuje komunikaci se serverem.

Komponenty React: reprezentují uživatelské rozhraní.

Databáze Neo4j: slouží jako uložiště uživatelských dat.

Knihovny webové aplikace: usnadňují komunikaci se serverem a napomáhají ve vizualizaci dat.

Knihovny serveru: usnadňují zpracování dat a napomáhají v komunikaci s databází.



Obrázek 5.1 Ilustrace: struktura systému

5.2 Klientská část

Klientská část aplikace je koncipována jako webová aplikace. Tato sekce se zaměřuje na vývoj této části, přičemž jsou představeny technologie použité při jejím vývoji a způsob, jakým byly mezi sebou propojeny. Budou zde představeny jednotlivé React komponenty (viz 4.2.1) a jejich funkce, stejně jako způsob, jakým jsou tyto komponenty propojeny s knihovnou Neovis.js pro vizualizaci dat.

5.2.1 Využití Reactu

Nyní budou popsány jednotlivé komponenty, z nichž se skládá vizualizační nástroj. Bude probrána jejich struktura, funkce a význam pro aplikaci.

5.2.1.1 Popis komponent

App: je kořenovou komponentou celé aplikace, ve které je vykreslena navigace. Po přechodu z navigace na vybranou stránku je v těle komponenty `App` vykreslena odpovídající komponenta, která reaguje na změny v URL adrese. K tomuto účelu je využita knihovna React Router, která umožňuje dynamické překreslení obsahu aplikace dle aktuální adresy.

AuthorGraphComponent: je komponenta, která vizualizuje graf autorů a jejich publikací. Tato komponenta zobrazuje spoluautory vybraného autora a jejich společné publikace, umožňující uživatelům prohlížet a analyzovat akademické vztahy a kolaborace mezi vědci.

UploadListComponent: je komponenta, která zobrazuje seznam nahraných grafů, poskytující uživatelům přehled o všech dostupných datech, která mohou být vizualizována.

FileUploadComponent: je komponenta, která obsahuje formulář pro nahrání grafů. Umožňuje uživatelům přidávat nová data do systému prostřednictvím souborů ve formátu CSV.

DefaultComponent: je komponenta, která se zobrazuje jako výchozí na úvodní stránce a obsahuje krátké informace o aplikaci, jejím účelu a základních funkcích.

HorizontalNav: je komponenta, která zobrazuje horizontální navigaci a umožňuje uživatelům přechod mezi různými stránkami aplikace.

5.2.1.2 Ant design

Ant Design je rozsáhlá knihovna předpřipravených stylizovaných komponent, která je využívána pro urychlení vývoje webových aplikací. Tento přístup

umožňuje zrychlit implementaci díky nabídce atraktivně vypadajících komponent. Knihovna nabízí mnoho uživatelských prvků, včetně formulářů, seznamů, tlačítek, navigačních prvků a mnoha dalších, které usnadňují vývoj a zajišťují konzistentnost designu napříč aplikací.

5.2.1.3 Axios

Pro komunikaci ze strany klienta se serverem byla použita knihovna *Axios*. Tato knihovna umožňuje snadnější způsob odesílání HTTP požadavků. Také usnadňuje zpracování odpovědí ze serveru. Hlavní podporované metody odesílání dat jsou `GET`, `POST`, `DELETE`, `PATCH` a `PUT`.

5.2.1.4 Implementace komponent

Stručně bude uvedena implementace komponenty `UploadListComponent`. Ostatní komponenty v aplikaci využívají podobné funkce React a strukturu, proto se popis omezí na tuto komponentu.

Výpis kódu 5.1 Ukázka komponenty UploadListComponent

```
function UploadListComponent() {
  const [uploads, setUploads] = useState([])
  const fetchUploads = async () => {
    const response = await getUploads()
    setUploads(response.data)
  }
  useEffect(() => { fetchUploads() }, [])
  return (
    <Card>
      <Title level={2}>Uploads List</Title>
      <Table dataSource={uploads} />
    </Card>
  )
}
```

Komponenta *UploadListComponent* slouží k zobrazení seznamu nahraných grafů. Využívá hooky *useState* a *useEffect* pro udržení stavu seznamu nahraných grafů a pro provedení asynchronního načtení dat při inicializaci komponenty. Po načtení dat jsou nahrané grafy společně s datem jejich nahrání zobrazeny ve formě tabulky pomocí komponenty *Table* z knihovny *Ant Design*. Komponenta dále využívá komponenty *Title* a *Card*, které slouží k vytvoření nadpisu a vizuálního oddělení obsahu.

5.2.2 Integrace s Neovis.js

5.2.2.1 Protokol Bolt

Bolt je protokol, který umožňuje vykonávat dotazy a získávat výsledky z databáze, například pomoci jazyka Cypher. Zrychluje komunikaci mezi klientem

a databází. Přenáší se přes TCP nebo WebSocket spojení. Jedná se o binární protokol [27].

5.2.2.2 Vizualizace dat

Vizualizace probíhá vytvořením komponenty *AuthorGraphComponent*, která zobrazuje autory, publikace a jejich vazby. Využívá funkce hooks pro získání parametrů z URL adresy pomocí *useParams*, udržování stavu komponenty pomocí *useState* a překreslení grafu autorů podle parametru *label* 5.3.4.1 v URL adrese pomocí *useEffect*. Pokud je *label* definován, vytváří se instance objektu *NeoVis* s konfigurací získanou z funkce *getNeoVisConfig(label)*. Metoda *render* vykreslí graf do HTML oddílu *div*, který využívá dostupnou šířku a výšku obrazovky. Návratová funkce *useEffect* správně odstraní objekt pro vykreslování z paměti.

Výpis kódu 5.2 Ukázka vizualizace Neovis.js

```
const AuthorGraphComponent = () => {
  const { label } = useParams()
  const [author, setAuthor] = useState([])
  useEffect(() => {
    if (!label) return
    const neoViz = new NeoVis(getNeoVisConfig(label))
    neoViz.render()
    return () => neoViz.clearNetwork()
  }, [label])
  neoViz.registerOnEvent("clickNode", async (node) => {
  if (node.group === "Author") {
    const authorName = node.label
    const uploadLabel = label
    const response = await axios.get(
    `${BASE_URL}$/coauthors/${authorName}/${uploadLabel}`
    )
    setAuthor(response.data)
  }
  })
  return <Row>
    <Col span={18}>
      <div
      id="viz"
      style={{ width: "100%", height: "100vh" }}
      ></div>
    </Col>
    <Col span={6}>
      <List dataSource={author.data} />
    </Col>
  </Row>
}
```

Funkce *registerOnEvent* slouží k zachycení událostí. V kódu je použita k registraci události *clickNode*, která se spustí při kliknutí na vrchol v grafu. V této události se zpracovává kliknutí na vrchol typu *Author*, kdy se provede HTTP požadavek na server a aktualizuje se seznam jeho spoluautorů a jejich společných publikací v komponentě.

V rámci *Row* jsou vytvořeny dva sloupce *Col* zknihovny Ant Design. Definují rozložení stránky. Sloupce definují, kolik prostoru na stránce budou zabírat.

Druhý sloupec obsahuje komponentu *List* z *Ant Design*, která zobrazuje seznam spoluautorů a publikací. Tato komponenta získává data z proměnné *author.data* a zobrazuje je ve formě seznamu. Seznam autorů se překreslí na změnu stavu díky použití hooku *useState*.

Funkce *getNeoVisConfig* slouží k vytvoření konfiguračního objektu pro třídu *NeoVis*. Konfigurace obsahuje několik hodnot: *containerId*, který určuje identifikátor HTML elementu, do kterého bude vykreslen graf, objekt *neo4j* obsahuje informace potřebné k připojení k databázi Neo4j, včetně adresy serveru, uživatelského jména a hesla. Dále *labels* definují typy vrcholů a vlastnosti ze kterých se vloží text do vrcholů, *relationships* specifikují vztahy mezi vrcholy, *initialCypher* obsahuje výchozí dotaz pro načtení dat při inicializaci grafu.

Výpis kódu 5.3 Ukázka konfigurace Neovis.js

```
const getNeoVisConfig = (label) => ({
  containerId: "viz",
  neo4j: {
    serverUrl: "bolt://localhost:7687",
    serverUser: "neo4j",
    serverPassword: "neo4j",
  },
  labels: {
    Author: { label: "name" },
    Paper: { label: "title" }
  },
  relationships: { AUTHORED: {} },
  initialCypher: `
  MATCH (a:\`${label}\`)-[r:AUTHORED]->(p:\`${label}\`)
  RETURN *
  `,
})
```

Během reálného provozu je nutné dbát na to, aby uživatel *neo4j* měl z bezpečnostních důvodů pouze právo pro čtení. Toto opatření zajišťuje, že nedojde k zneužití práv ze strany klienta, například k nechtěnému smazání databáze.

5.3 Serverová část

Serverová část aplikace se skládá z Node.js a databáze Neo4j. Node.js umožňuje přistupovat k datům v databázi pomocí knihovny `neo4j-driver`. Zároveň zpracovává CSV soubory od uživatele pomocí knihovny `csv-parse` a ukládá data do databáze.

5.3.1 REST API

Pro získávání dat a nahrávání ze serveru bylo využito REST (*REpresentational State Transfer*) API , které umožňuje komunikaci mezi klientskou částí a serverem pomocí HTTP metod `GET`, `POST`, `PUT` a `DELETE` [28]. Metoda `GET` byla použita pro získávání dat, například pro načítání informací o společných publikacích a spoluautorech, zatímco metoda `POST` byla využita pro nahrávání CSV souborů od uživatele. Pro každou z těchto operací je definován specifický endpoint, na který se odesílají uživatelské požadavky.

5.3.2 Použité formáty dat

Byly použity formáty dat jako CSV a JSON. Většina komunikace mezi klientem a serverem probíhá ve formátu JSON, který umožňuje hierarchickou reprezentaci dat. Ve formátu CSV se zpracovává soubor s autory a publikacemi, který uživatelé nahrávají pro zpracování a následnou vizualizaci.

5.3.3 Připojení do databáze

Připojení k databázi Neo4j probíhá pomocí objektu `driver`, který se získává pomocí knihovny `neo4j-driver`. Prvním argumentem při vytváření tohoto objektu je URL adresa, kde běží databáze Neo4j. Druhým argumentem jsou autentizační údaje, které zahrnují uživatelské jméno a heslo.

Výpis kódu 5.4 Ukázka připojení do databáze

```
const driver = neo4j.driver(
  "bolt://localhost:7687",
  neo4j.auth.basic("neo4j", "neo4j")
)
const session = driver.session()
```

Z objektu `driver` se následně získává objekt `session`, na kterém se dají volat metody, jako je například `run`. Argumentem funkce `run` je Cypher dotaz.

5.3.3.1 Zpracování dat

V této sekci je popsán proces zpracování CSV souboru, který poskytne uživatel. Soubor obsahuje mnoho informací. Mezi těmito informacemi najdeme názvy publikací, jména autorů, počet citací, druh publikace a další relevantní údaje.

Pro účely vizualizace jsou relevantní hodnoty jmen autorů a názvů publikací. CSV soubor se zpracovává pomocí knihovny `csv-parse`.

Výpis kódu 5.5 Ukázka zpracování dat

```
const readStream = fs.createReadStream(filePath)
const parser = csvParse({ delimiter: "," })
const transformer = new Transform({
    objectMode: true,
    transform(record, _encoding, callback) {
      const authors = record["AUTHORS"]
      const paperTitle = record["NAME"]
      this.push({ authors, paperTitle })
      callback()
    }
})
const uploadLabel = "`"
 + `upload_${Date.now()}_`
 + `${nanoid()}` + "`"
readStream
  .pipe(parser)
  .pipe(transformer)
  .pipe(neo4jWriter(session, uploadLabel))
```

Prvně je vytvořen objekt `readStream`, který reprezentuje stream pro čtení dat ze souboru. Poté je vytvořen objekt parser, což je stream pro transformaci, který umožňuje jak čtení, tak i zápis. Zde je použit pouze pro čtení. V něm je nastavena konfigurace pro CSV soubor, jako je oddělovač. Následuje objekt `transformer`, který je také stream pro transformaci a slouží k extrakci jmen autorů a názvů publikací z CSV souboru v metodě `transform`. Je nastaven tak, aby pracoval s daty ve formě objektu a ne řetězců či bufferů pomocí atributu `objectMode`. Extrahovaná data jsou pomocí funkce `push` předána dalšímu zapisovacímu streamu `neo4jWriter`, který tato data uloží do databáze Neo4j. Volání funkce `callback` indikuje, že zpracování pro tento záznam je ukončeno a je připraven pro další data. Kódování dat lze nastavit pomocí argumentu `encoding`, ale zde není použito. Význam proměnné *uploadLabel* je popsán v databazovém modelu (viz. 5.3.4.1). Jedná se o label, který rozlišuje jednotlivá nahrání, používá knihovnu *nanoid* pro tvorbu unikátního identifikátoru na konci řetězce. Streamy jsou spojeny pomocí metody `pipe`, kdy výstup jednoho streamu se stane vstupem následujícího [29].

Výpis kódu 5.6 Ukázka ukladání dat

```
function neo4jWriter(session, uploadLabel) {
  return new Transform({
    objectMode: true,
    async transform(record, _encoding, callback)) {
      try {
        const { authors, paperTitle } = record
        for (const author of authors)
```

```
          await createAuthorAndPaper(session, author
          , paperTitle, uploadLabel)
        callback()
    }
  })
}
```

Tento kód využívá stejný princip `Transform` streamu, jak bylo popsáno výše. Přijatá data jsou uložena do databáze voláním funkce `createAuthorAndPaper`.

Výpis kódu 5.7 Ukázka funkce createAuthorAndPaper

```
async function createAuthorAndPaper(
  session,
  author,
  paperTitle,
  uploadLabel
) {
  const cypherQuery = `
    MERGE (author:Author {name: $authorName})
    MERGE (paper:Paper {title: $paperTitle})
    MERGE (author)-[:AUTHORED]->(paper)
    SET author:${uploadLabel}
    SET paper:${uploadLabel}
  `;
  const params = {
    authorName: author,
    paperTitle: paperTitle,
  };
  await session.run(cypherQuery, params);
}
```

Výše demonstrovaná funkce `createAuthorAndPaper` přijímá jméno autora, název publikace a *uploadLabel*. Vytváří se Cypher dotaz, jehož význam byl popsán v sekci o dotazech (viz. 5.3.4.2). V objektu `params` jsou uloženy parametry, které budou použity v Cypher dotazu jako proměnné. Do proměnné *uploadLabel* se vloží patřičná hodnota. Nakonec se spustí dotaz metodou `run` objektu `session`.

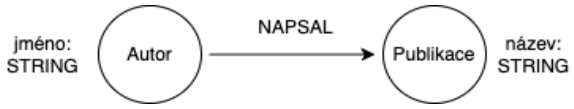
5.3.4 Implementace Neo4j

V této sekci je věnována pozornost datovému modelu, který byl použit ve vizualizačním nástroji. Budou také probrány dotazy, které byly použity pro získávání dat z databáze.

5.3.4.1 Databázový model

Databázový model, který byl použit ve vizualizačním nástroji je poměrně jednoduchý. Obsahuje dva typy entit: „Autor“ a „Publikace“, které jsou rozlišeny

příslušnými labely. Mezi těmito entitami je definována vazba „NAPSAL“, směřující od autora k publikaci. Autor obsahuje vlastnost jméno, které je řetězec znaků. Publikace obsahuje vlastnost název, který je též řetězcem znaků. Tento datový model je shrnut na obrázku níže, který vizualizuje strukturu a vztahy mezi entitami.



Obrázek 5.2 Neo4j: Databazový model

Aby se vrcholy v databázi jednoznačně identifikovaly k jakému nahrání patří, byl zaveden speciální label s formátem „upload_datum_*uid*“, kde *datum* je počet milisekund uplynulých od 1. ledna 1970 a *uid* je unikátní identifikátor. Tyto hodnoty jsou poskytnuty serverem Node.js. Tento label umožňuje rozlišení jednotlivých nahrání. Ačkoliv tento label není uveden na přiloženém obrázku, je nezbytný pro správnou funkčnost. Ve zdrojových kódech aplikace je tento label uložen v proměnné *uploadLabel*.

5.3.4.2 Vytvoření autora a publikace

Jedná se o Cypher dotaz, který používá klíčové slovo `MERGE`, které zajistí, že záznamy budou vytvořeny pouze v případě, že neexistují [30]. Zároveň vytváří vazbu `AUTHORED` mezi autorem a publikací. Směrem od autora k publikaci. Klíčové slovo `SET` nastaví label, identifikující dané nahrání pro tyto vrcholy autor a publikace.

Výpis kódu 5.8 Ukázka Cypher dotazu pro vytváření dat

```
MERGE (author:Author {name: $authorName})
MERGE (paper:Paper {title: $paperTitle})
MERGE (author)-[:AUTHORED]->(paper)
SET author:${uploadLabel}
SET paper:${uploadLabel}
```

5.3.4.3 Získávání labels

Tento Cypher dotaz využívá klíčová slova `MATCH` pro vyhledání všech vrcholů, `UNWIND` k rozbalení labelů každého vrcholu, získaných funkcí `labels`, na samostatné záznamy, `WITH DISTINCT` k odstranění duplicit a zajištění unikátnosti výsledků, `WHERE` k filtrování labelů, které začínají na „upload“ a `RETURN` k vrácení unikátních labelů.

Výpis kódu 5.9 Ukázka Cypher dotazu pro získávání labelů

```
MATCH (n)
UNWIND labels(n) AS label
WITH DISTINCT label
```

```
WHERE label STARTS WITH 'upload'
RETURN label
```

5.3.4.4 Získávání spoluautorů a publikací

Tento dotaz v Cypheru vyhledává autory a jejich spoluautory, kteří společně publikovali, využívá k tomu specifické labely identifikující konkrétní nahrání dat. Klíčové slovo `MATCH` je použito k nalezení autora podle zadaného jména, jeho publikace a spoluautory. Klauzule `WHERE` filtruje vrcholy na základě těchto labelů a `RETURN` vrátí informace o autorovi, názvy publikací a seznam spoluautorů jako kolekci.

Výpis kódu 5.10 Ukázka Cypher dotazu pro získávání spoluautorů a publikací

```
MATCH (author {name: $name})-[:AUTHORED]->(paper),
  (paper)<-[:AUTHORED]-(coAuthor)
WHERE $uploadLabel IN labels(author) AND
  $uploadLabel IN labels(paper) AND
  $uploadLabel IN labels(coAuthor)
RETURN author,
  paper.title AS title,
  collect(coAuthor) AS coAuthors
```

5.4 Shrnutí

Byl implementován prototyp vizualizačního nástroje, který splňuje stanovené požadavky. Tento nástroj je implementován jako SPA webová aplikace, která umožňuje dynamické a plynulé uživatelské rozhraní bez nutnosti opakovaného načítání celé stránky. Na serverové straně je implementován REST API server, který je vyvíjen pomocí technologií Node.js, zajišťující komunikaci s klientskou částí. Pro efektivní ukládání dat je použita databáze Neo4j, která umožňuje efektivní správu a vyhledávání dat.

Kapitola 6

Testování prototypu

Tato kapitola se věnuje testování nástroje. Rozebrány jsou jednotlivé testovací scénáře. Stručně jsou popsány jednotkové testy, které byly též využity.

Prototyp vizualizačního nástroje bude testován na reálných datech, která budou exportována ze systému V3S. Tato data poskytnou autentický základ pro ověření funkčnosti a výkonnosti nástroje v reálném provozním prostředí. Testování s reálnými daty umožní lépe hodnotit, jak efektivně nástroj zvládá skutečné použití. Tento přístup také pomůže identifikovat potenciální problémy a nedostatky v návrhu a implementaci, které by mohly být přehlédnuty při testování s fiktivními nebo simulovanými daty.

6.1 Splnění případů užití

Testovací scénáře pro vizualizační nástroj byly navrženy na základě případů užití, které pokrývají základní funkce systému. Tyto případy užití zahrnují vizualizaci grafem, výpis již nahraných grafů a nahrávání nových grafů. Každý z těchto scénářů je připraven tak, aby otestoval specifické funkce nástroje. Všechny testovací scénáře byly prováděny na souborech ze systému V3S, které reprezentují reálná data. Níže je podrobně popsán každý testovací scénář, včetně kroků, které jsou potřeba pro ověření správného fungování a interakce uživatelů s aplikací. Následující způsob testování je založen na metodě černé skříňky (black-box). To znamená, že tester nemá přístup k interní struktuře kódu. Tester zná vstupy a očekávané výsledky [31].

6.1.1 Ukládání grafu

Cíl testu: Cílem tohoto testovacího scénáře je ověřit, zda systém správně zpracovává a ukládá grafy z nahraných souborů ve formátu CSV pocházejících ze systému V3S. Test zkontroluje, jestli systém úspěšně načte CSV sou-

bor, správně extrahuje data a následně generuje graf, který je uživatelům nabídnut k vizualizaci.

Pokryté případy užití: Uložení vizualizace UC1

Předpoklady: Uživatel se nachází na úvodní stránce

Kroky testu: Jsou následující

1. Uživatel se přemístí pomocí horní lišty navigace na stránku s formulářem pro nahrávání
2. Systém poskytne formulář
3. Uživatel nahraje soubor
4. Systém zpracuje soubor a nabídne vizualizaci

Očekávané výsledky: Systém úspěšně zpracuje nahraný soubor. Po zpracování by měl systém uživateli nabídnout vizualizaci grafu prostřednictvím odkazu. Kliknutím na tento odkaz by uživatel měl vidět graf.

Shrnutí: Test ukládání grafu proběhl úspěšně, avšak systém původně nezobrazoval chyby, které nastaly. Tento nedostatek byl odstraněn a zapracován do další verze prototypu

6.1.2 Výpis nahraných grafů

Cíl testu: Cílem tohoto testovacího scénáře je ověřit, zda systém správně vypisuje seznam všech nahraných grafů v uživatelském rozhraní. Test zkontroluje, jestli jsou grafy správně a úplně zobrazeny v seznamu, a zda je možné každý graf vybrat pro další vizualizaci.

Pokryté případy užití: Vypisování vizualizací UC1

Předpoklady: Uživatel se nachází na úvodní stránce

Kroky testu: Jsou následující

1. Uživatel se přemístí pomocí horní lišty navigace na stránku se seznamem grafů
2. Systém vypisuje seznam nahraných grafů
3. Uživatel dostane k dispozici seznam nahraných grafů

Očekávané výsledky: Výpis tabulky, ve které jsou uvedeny název grafu a datum nahrání

Shrnutí: Test výpisu nahraných grafů proběhl úspěšně a bez zjištěných problémů. Systém správně zobrazil seznam všech nahraných grafů v uživatelském rozhraní, přičemž každý graf byl přístupný.

6.1.3 Vizualizace grafem

Cíl testu: Cílem tohoto testovacího scénáře je ověřit, zda systém správně vizualizuje grafy z nahraných dat. Test zkontroluje, jestli systém umožňuje uživatelům efektivně prohlížet a interagovat s grafem, včetně možnosti přiblížení, posouvání a výběru konkrétních vrcholů nebo hran. Dále test zkontroluje, jestli nástroj správně zobrazuje spoluautory vybraného autora a jejich publikace, na kterých se podíleli, tím umožňuje uživatelům snadno vizualizovat a analyzovat spolupráce.

Pokryté případy užití: Vizualizace grafem UC1, UC2, UC3, UC4

Předpoklady: Uživatel se nachází na výpisu grafů

Kroky testu: Jsou následující

1. Uživatel si vybere konkrétní graf
2. Systém zobrazí stránku s vizulizací
3. Uživatel si prozkoumá graf dle svých zájmů a vybere si konkrétního autora
4. Systém vypíše seznam publikací se spoluautory

Očekávané výsledky: Systém správně zobrazí graf na základě uložených dat. Umožní uživatelům s grafem interagovat. Systém je schopen zobrazit specifického autora, jeho spoluautory a společné publikace

Shrnutí: Test proběhl úspěšně. Byl správně zobrazen graf na základě uložených dat a bylo umožněno s grafem efektivně interagovat. Dále byli správně zobrazeni spoluautoři a jejich publikace, což poskytlo uživatelům hlubší pohled na akademické vztahy. Avšak byl zjištěn nedostatek, kdy při zobrazení příliš dlouhého seznamu společných publikací došlo k přetečení viditelné plochy. Tento problém byl následně identifikován a opraven v aktualizované verzi prototypu.

6.2 Jednotkové testy

Během vývoje vizualizačního nástroje byly využívány jednotkové testy. Tyto testy umožňují detailní ověření funkčnosti menších částí aplikace, typicky jednotlivých funkcí nebo metod. Jednotkové testy jsou psány přímo vývojáři, kteří pracují na projektu, a jejich přístup je založen na metodě bílé skříňky (white-box). To znamená, že tester, v tomto případě vývojář, má porozumění k interní struktuře kódu, který testuje. Tento princip umožňuje identifikaci a opravu specifických problémů [32].

Níže je ukázka kódu jednotkového testu, který testuje správnost dotazu při vytváření vrcholů pro autory, publikace a vytváření hrany, která symbolizuje

vztah, že autor napsal danou publikaci. Tento test ověřuje, zda je databázový dotaz správně formulován a zda se data správně ukládají do grafu.

Výpis kódu 6.1 Ukázka jednotkového testu

```
const mockSession = {
  run: jest.fn(),
}
describe("createAuthorAndPaper function", () => {
  it("should create an author and paper", async () => {
    const author = "John Doe"
    const paperTitle = "Sample Paper"
    const uploadLabel = "SampleUploadLabel"
    const expectedCypherQuery = `
      MERGE (author:Author {name: $authorName})
      MERGE (paper:Paper {title: $paperTitle})
      MERGE (author)-[:AUTHORED]->(paper)
      SET author:${uploadLabel}
      SET paper:${uploadLabel}
    `
    const expectedParams = {
      authorName: author,
      paperTitle: paperTitle,
    }
    await createAuthorAndPaper(
      mockSession,
      author,
      paperTitle,
      uploadLabel
    )
    const [args] = mockSession.run.mock.calls
    const [cypherQuery, params] = args

    should(cypherQuery).be.equal(expectedCypherQuery)
    should(params).be.deepEqual(expectedParams)
  })
})
```

Mocking Session Objektu: Test začíná vytvořením *mock* objektu pro `Session` z knihovny *neo4j-driver*. Mocking umožňuje smazat skutečnou implementaci testované funkce a kontrolovat externí volání [33].

Testovací scénář: V bloku `describe` je definován testovací scénář pro funkci `createAuthorAndPaper`. Tento scénář zahrnuje jeden test, který ověřuje, že funkce správně vytváří vrcholy a hranu v databázi.

Testovací Funkce: Uvnitř `it` bloku je asynchronní funkce, která spouští funkci `createAuthorAndPaper` s předem definovanými parametry pro autora a publikaci.

Ověření: Po spuštění funkce se ověří, že databázové volání bylo provedeno s očekávaným *Cypher* dotazem a parametry. K tomu se používá funkce `should` z knihovny `should`, která kontroluje, že dotaz a parametry odpovídají očekáváním.

6.3 Shrnutí

V této části práce bylo provedeno testování na reálných datech ze systému V3S. Testování odhalilo několik nedostatků, které byly následně odstraněny v další verzi prototypu. Kromě toho byly využity jednotkové testy, které pomohly ověřit funkčnost jednotlivých funkcí.

Kapitola 7
Ekonomický přínos

Tato kapitola se věnuje aktivitě zaměstnance fakulty a tomu, jak mu vizualizační nástroj práci zefektivňuje. Probrán je také způsob, jakým byla spočítána ekonomická výhoda pro fakultu.

Zde bude rozebrán ekonomický přínos vizualizačního nástroje, přičemž hlavní metrikou bude ušetřený čas zaměstnanců fakulty. Ušetřený čas přispěje k efektivnější práci zaměstnanců a povede k efektivnějším výdajům fakulty. Optimalizace času umožní zaměstnancům věnovat se dalším důležitým úkolům, čímž se zvýší produktivita a sníží se náklady spojené s provozem.

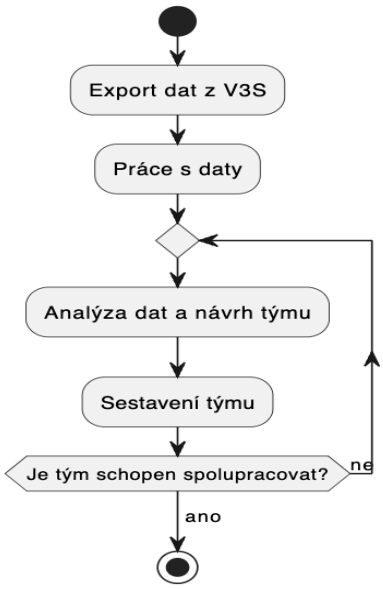
Metoda měření ušetřeného času spočívá v porovnání času stráveného na konkrétních úkolech před a po zavedení vizualizačního nástroje. Tato metoda umožňuje kvantifikovat, jaký dopad má implementace nástroje na efektivitu pracovních procesů. Měříme dobu, kterou zaměstnanci tráví vykonáváním specifických úkolů s a bez použití nástroje. Metoda poskytne představu o časových úsporách. Zmíněný přístup přinese měřitelné výsledky, které lze použít k vyhodnocení ekonomické hodnoty nástroje pro fakultu.

7.1 Pracovní postup

Pracovní postup se skládá ze čtyř kroků. Tyto kroky jsou základními stavebními kameny pro sestavení týmu pro nadcházející projekt a zahrnují: export dat, práci s daty, analýzu dat a návrh týmu, případné opakování postupu s úpravami týmu. Postup může být ilustrovan diagramem aktivit 7.1.

7.1.1 Export dat

Prvním krokem pracovního postupu je přihlášení zaměstnance do systému V3S, kde následně exportuje soubor obsahující informace o potenciálních členech týmu podle svých specifických potřeb.



Obrázek 7.1 Diagram aktivity: postup práce

7.1.2 Práce s daty

Po exportu dat si zaměstnanec v tabulkovém editoru, nejčastěji v aplikaci Microsoft Excel, upravuje a přizpůsobuje data podle svých preferencí. Využívá funkcí filtrů, třídění a kalkulačních nástrojů, aby data co nejlépe odpovídala potřebám sestavování efektivního týmu pro nadcházející projekt.

7.1.3 Analýza dat a návrh týmu

Po přípravě dat v tabulkovém editoru zaměstnanec přechází k analýze dat a návrhu týmu. V této fázi zkoumá podobné spolupráce, odborné dovednosti a dostupnost potenciálních členů týmu, aby sestavil skupinu pro realizaci projektu. Tento proces zahrnuje hodnocení kompatibility členů týmu a vyhledávání vhodných dovedností, které budou vyhovovat požadavkům projektu.

7.1.4 Opakování postupu

V procesu sestavování týmu může být nutné opakovat analýzu dat několikrát, aby bylo dosaženo optimálního složení týmu. Tato iterace umožňuje zaměstnanci upravit a přepracovat původní návrhy na základě zpětné vazby nebo nově zjištěných informací. Opakování analýzy dat je často řeší problémy s kompatibilitou členů týmu nebo pro doplnění potřebných dovedností. Tato fáze pomůže dosáhnout cíle a požadavky projektu.

7.2 Měření času

Jak bylo zmíněno v kapitole vizualizace dat (viz. 1), vizualizace mohou významně pomoci s analýzou dat a přispět k lepším rozhodnutím. Čas byl měřen metodou srovnání práce před a po zavedení vizualizačního nástroje. Před implementací tohoto nástroje trvala analýza dat a návrh týmu přibližně 3 pracovní dny za týden. Po zavedení nástroje byla tato činnost zrychlena na zhruba jeden den týdně. Pozorujeme tedy šetření pracovního času zaměstnance třikrát. Toto zlepšení efektivity práce názorně ukázáno v následující tabulce, která také ukazuje průměrný čas trávený touto činností za den. Počet dnů je převeden na minuty a celkový čas je rozložen do pracovního týdne, který obsahuje 5 pracovních dnů.

Tabulka 7.1 Porovnání času stráveného vyhledáváním před a po implementaci vizualizačního nástroje

Aktivita	Počet dnů za týden	Celkový čas za den (minuty)
Vyhledávání bez nástroje	3	$\frac{1440}{5} = 288$ minut
Vyhledávání s nástrojem	1	$\frac{480}{5} = 96$ minut

7.3 Finanční hodnocení ušetřeného času

Finanční hodnocení ušetřeného času je faktorem pro vyhodnocení celkového přínosu vizualizačního nástroje. Při posuzování ekonomického přínosu vizualizačního nástroje je důležité převést ušetřený čas na peněžní hodnotu.

7.3.1 Hodnota hodiny práce

Pro účely tohoto výpočtu se nebude uvádět konkrétní částka hodinové mzdy. Místo toho postup se zaměří na obecný výpočet a ilustraci potenciální úspory času, kterou vizualizační nástroj přináší.

7.3.2 Výpočet celkového ušetřeného času

Jak bylo zmíněno výše, implementace nástroje snížila potřebný čas pro vykonání určitých úkolů z 3 pracovních dnů na 1 pracovní den týdně. Převedeno na hodiny, to může znamenat, že zaměstnanec strávil o 64 hodin méně za měsíc na analýze dat (za předpokladu, že pracovní den trvá 8 hodin).

7.3.3 Příklad výpočtu

Uvedený výpočet celkové měsíční úspory 64 hodin platí pro jednoho zaměstnance. Pokud by bylo potřeba posoudit celkový dopad na větší skupinu zaměstnanců, lze tento výpočet jednoduše vynásobit počtem zaměstnanců, kteří nás-

troj používají. Jedná se například o zaměstnance oddělení pro spolupráci s průmyslem.

$$\text{CFÚ} = \text{PUH} \cdot \text{HM} = 64 \cdot \text{HM}$$

CFÚ: Celková finanční úspora za měsíc

PUH: Počet ušetřených hodin

HM: Hodinová mzda

7.4 Další zdroj příjmů

Další možností ekonomického přínosu je zlepšení kvality vytvořených publikací. Vizualizační nástroj poskytne vědcům fakulty jednodušší a efektivnější možnost hledání spoluautorů pro jejich nové publikace, zejména v oblastech, ve kterých původní autor přímo nepůsobí, ale chce, aby jeho publikace zahrnovala i tuto oblast. Takovým způsobem mohou vědci nebo skupiny vědců vytvořit kvalitnější publikace. Kvalitnější publikace se následně promítají do vyšších peněžních částek, které škola získává z různých zdrojů, například grantů.

7.5 Shrnutí

V této analýze bylo vyhodnoceno, jak implementace nového vizualizačního nástroje zvýšila efektivitu vyhledávání a analýzy dat. Před zavedením nástroje zaměstnanec strávil vyhledáváním 3 pracovní dny týdně, což se snížilo na 1 den týdně. To představuje úsporu 64 pracovních hodin měsíčně pro jednoho zaměstnance.

Tato časová úspora se promítá do významných finančních úspor a zvyšuje produktivitu práce, umožňuje zaměstnancům věnovat se dalším úkolům.

Byl zmíněn i další potenciální zdroj příjmů pro fakultu ve formě kvalitnějších publikací, které vědci vytvoří pomocí tohoto vizualizačního nástroje.

Kapitola 8

Další rozvoj a studie proveditelnosti

Tato kapitola se věnuje dalšímu rozvoji vizualizačního nástroje ve formě nových funkcí. Zaměřuje se na jejich technickou a finanční proveditelnost.

8.1 Rozšíření funkcí

8.1.1 Vizualizace grafem podle klíčových slov

Je vhodné začlenit novou vizualizační metodu založenou na klíčových slovech z publikací. V této vizualizaci by jednotlivé publikace vystupovaly jako vrcholy grafu. Propojení mezi publikacemi by bylo reprezentováno hranami, pokud by publikace sdílely alespoň jedno klíčové slovo. Po kliknutí na vrchol publikace by se automaticky zobrazil seznam autorů dané publikace. Tato vizualizace by usnadnila identifikaci souvisejících prací a pomohla by při orientaci v jiných oblastech výzkumu.

8.1.2 Filtrace grafu dle data publikace

Dalším vhodným rozšířením by bylo implementovat funkci filtrace grafu podle data publikace. Uživatelé by měli možnost zmenšit grafickou reprezentaci grafem tak, aby zobrazovala pouze publikace z konkrétního časového období. Tato funkcionalita by umožnila uživatelům lépe se orientovat v historii výzkumu nebo identifikovat nové trendy.

8.1.3 Vizualizace externích spolupracovníků

Je vhodné zahrnout funkci, která umožní vizuálně odlišit externí spolupracovníky, tj. vědce z jiných institucí, v síťovém grafu. Externí spolupracovníci

by mohli být označeni jinou barvou, která by zjednodušila identifikaci spolupráce mezi školami. Tato funkcionalita by podpořila uživatele v rychlejším identifikování klíčových externích vědců a jejich vlivu na akademické prostředí.

8.2 Technická proveditelnost

8.2.1 Vizualizace grafem podle klíčových slov

Jako řešení se nabízí zavedení nového typu vrcholu - klíčového slova. Pokud publikace obsahuje určité klíčové slovo, vytvoří se mezi těmito vrcholy hrana. V současné době nejsou klíčová slova povinným údajem, a proto exportovaný soubor ze systému V3S tyto informace neobsahuje. Bude proto vyžadována spolupráce s VIC ČVUT[1], aby bylo možné tato chybějící data doplnit do exportovaného souboru. Samotná implementace vizualizace bude velmi podobná již existujícímu typu vizualizace, která spojuje autory s publikacemi.

Časový odhad: 104 hodiny práce

Složitost implementace: střední

Priorita: vysoká

8.2.2 Filtrace grafu dle data publikace

V současné době exportovaný soubor obsahuje datum publikace. Proto je vhodné rozšířit entitu *Publikace* o vlastnost „datum publikace“. Dále je potřeba do vizualizace přidat políčko pro zadání rozsahu dat. Dalším krokem by také bylo přidání klauzule *WHERE* do příslušných databázových dotazů, která umožní filtrovat data podle rozsahu zadaného uživatelem.

Časový odhad: 24 hodiny práce

Složitost implementace: lehká

Priorita: nízká

8.2.3 Vizualizace externích spolupracovníků

Jedná se o rozšíření stávající funkcionality, které umožní vizuálně odlišit autory pracující v jiných institucích. Toho bude dosaženo barevným odlišením vrcholu typu *Autor*. K tomu by bylo vhodné zavést novou vlastnost pro vrchol typu autor, kde bude uvedena jeho instituce. V současné době exportovaný soubor ze systému V3S neobsahuje informace o instituci, a proto by byla vyžadována spolupráce s VIC ČVUT. Cílem je doplnění chybějících dat do exportovaného

[1] Výpočetní a Informační centrum ČVUT

souboru nebo získání těchto dat jiným způsobem a jejich následné uložení do databáze vizualizačního nástroje.

Časový odhad: 24 hodiny práce

Složitost implementace: lehká

Priorita: nízká

8.3 Časový harmonogram

Časový harmonogram vývoje nových funkcí je navržen s ohledem na odhady časové náročnosti jednotlivých funkcí. Jak bylo zmíněno výše, harmonogram zahrnuje také čas potřebný pro komunikaci s VIC ČVUT, aby bylo možné navrhnout úpravy, včetně přidání nových sloupců do exportovaného CSV souboru. Po navržení úprav souboru není nutné čekat na jejich skutečnou implementaci. Práce na nových funkcích může začít ihned, s tím, že během programování bude třeba mít na paměti plánované změny struktury souboru. Tyto změny je možné ověřit, například pomocí jednotkových testů, a to i s využitím CSV souboru, který již reflektuje navrhovaný nový formát.

Časový harmonogram vývoje nových funkcí je graficky znázorněn pomocí Ganttova diagramu, který ukázán níže. Ganttův diagram je chronologický pruhový graf, snadno čitelná časová osa, pomocí které lze přehledně vizualizovat podrobnosti projektu [34]. Diagram poskytuje vizuální přehled o plánování jednotlivých úkolů a jejich časového rozvržení. Ukazuje sekvenci a závislosti mezi činnostmi, včetně času vyhrazeného pro komunikaci s VIC ČVUT ohledně nutných úprav v exportovaném CSV souboru.

Obrázek 8.1 Ganttův diagram: vývoj nových funkcí

V diagramu se předpokládá, že na projektu rozšíření funkcí se začne pracovat od 2. září 2024.

8.4 Finanční proveditelnost

Finanční proveditelnost pro rozvoj nových funkcí vizualizačního nástroje se soustředí především na dva hlavní faktory: náklady na lidské zdroje a náklady

na technologické zdroje a infrastrukturu. Náklady na lidské zdroje jsou odhadnuty na základě pracovního času, který byl určen výše technické proveditelnosti.

8.4.1 Náklady na lidské zdroje

Náklady na lidské zdroje jsou vyjádřeny v člověkodnech. Člověkoden je pracovní čas jedné osoby, který odpovídá jednomu pracovnímu dni, tedy typicky 8 hodinám [35]. Každý člověkoden má také finanční ohodnocení, které se vyjadřuje sazbou za člověkoden. Tato sazba určuje, kolik stojí jeden den práce daného člověka.

Průměrný plat Node.js vývojáře v České republice je přibližně 650 Kč za hodinu, podle informací získaných ze serveru Jooble [36]. Tato data jsou relevantní ke dni psaní práce. Vzhledem k tomu, že jeden člověkoden odpovídá osmi hodinám práce, cena za jeden člověkoden činí 5200 Kč. Tato částka však nezahrnuje další náklady související se zaměstnáním, jako jsou sociální a zdravotní pojištění a příspěvky na důchod.

Není potřeba zvlášť najímat React vývojáře, poněvadž základ klientské aplikace je již hotový a není těžké se v něm vyznat a přidávat další funkce. Proto bude stačit Node.js vývojář, který dokáže rozšířit stávající funkcionalitu. Vzhledem k tomu, že navržené funkce projektu jsou na sobě nezávislé, je možné zapojit více vývojářů, kteří by pracovali paralelně na jednotlivých částech. Optimální počet vývojářů pro tento projekt by byl tři. Každý vývojář by se tak mohl věnovat jedné z funkcí, což by zefektivnilo vývojový proces a zkrátilo dobu potřebnou k realizaci nových funkcí.

Níže je tabulka, která shrnuje ceny práce za jednotlivé funkce. Souhrnně činí celkové náklady 98 800 Kč. Tato částka je však pouze orientační a může se ke dni zahájení projektu lišit, v závislosti na trhu práce a možných změnách v cenách pracovní síly.

Nebyla započítána cena práce VIC ČVUT, poněvadž konkrétní cena jejich práce nebyla stanovena a nebylo cílem zjišťovat tento soukromý údaj. Tato nejistota by měla být zohledněna při plánování rozpočtu vedením či zodpovědnými zaměstnanci fakulty. Je důležité mít na paměti tuto proměnnou při celkovém finančním hodnocení projektu, aby bylo možné reagovat na možné budoucí finanční nároky spojené s integrací a podporou od VIC ČVUT.

Funkce	Odhad hodin	Počet člověkodnů	Celkové náklady
Vizualizace grafem podle klíčových slov	104	13	67 600 Kč
Filtrace grafu dle data publikace	24	3	15 600 Kč
Vizualizace externích spolupracovníků	24	3	15 600 Kč

Tabulka 8.1 Odhad nákladů na lidské zdroje pro vývoj funkcí

8.4.2 Náklady na technologické zdroje a infrastrukturu

Náklady na technologické zdroje a infrastrukturu jsou následující. Vzhledem k tomu, že se využívají technologie, které jsou dostupné zdarma, což zahrnuje všechny softwarové nástroje a knihovny použité při vývoji, budou náklady spojené především s provozem serverů.

Provoz vizualizačního nástroje bude vyžadovat hosting serverů, pro který byla zvolena platforma Amazon Web Services (AWS). Tato platforma je známá svou spolehlivostí, škálovatelností a širokým spektrem služeb, které umožňují efektivní správu a provoz aplikací [37].

Infrastruktura serverů pro pilotní provoz byla zvolena poměrně jednoduchá. Celkem budou použity dva virtuální servery EC2 instance [2] od AWS. Na jednom serveru bude instalována serverová část aplikace, která zahrnuje databázi Neo4j a Node.js server. Druhý server bude sloužit jako webový server, který poskytne uživatelům přístup ke klientské části aplikace. Toto rozdělení zajišťuje efektivní využití zdrojů a optimalizaci výkonu, kdy každý server může být konfigurován a škálován nezávisle podle specifických potřeb jeho komponent.

Níže je uvedena tabulka s odhadem kalkulací cen za provoz, která byla vytvořena pomocí AWS Pricing Calculator [38]. Konfigurace virtuálních serverů byla zvolena tak, aby dobře převyšovala minimální požadavky na systém. Aplikace bude disponovat dostatečným výpočetním výkonem a pamětí pro plynulý běh, avšak konfigurace serverů se může změnit v závislosti na požadavcích během reálného provozu.

Tabulka 8.2 Náklady na provoz serverové a klientské části

Část aplikace	Měsíční náklady (USD)	Roční náklady (USD)
Serverová část	81,03	972,36
Klientská část	36,50	438,00

Celkové měsíční náklady na provoz činí 117,53 USD. Roční náklady pak dosahují 1410,36 USD. Tyto hodnoty zahrnují provoz obou virtuálních serverů, které společně zajišťují funkčnost serverové a klientské části aplikace.

Vizualizační nástroj není nijak vázán na poskytovatele AWS. Byly využité virtuální servery, které lze nakoupit i od jiných poskytovatelů. Ceny by byly velmi podobné.

8.5 Právní aspekty

V dnešní době je zásadní věnovat zvýšenou pozornost ochraně soukromí a citlivých údajů, což zahrnuje dodržování předpisů jako je Obecné nařízení o ochra-

[2] jedná se o komerční název pro virtuální server

ně osobních údajů (GDPR) [39]. Nicméně, tento vizualizační nástroj využívá veřejně dostupné údaje. Autoři publikací, poskytli svůj explicitní souhlas s jejich publikací a dalším využitím.

8.6 Shrnutí významu dalšího rozvoje

Byly uvedeny tři funkce, které by měly zvýšit užitečnost aplikace. Nejdůležitější z nich je vizualizace grafem podle klíčových slov, což bylo určeno na základě zpětné vazby poskytnuté oddělením pro spolupráci s průmyslem i od vědců z FIT. Tato funkce bude významně přispívat vědcům v hledání potenciálních spoluautorů, jak bylo zmíněno výše (viz. 7.4). Zbylé funkce, ačkoli jsou méně důležité, mohou také přinést užitek a zlepšit uživatelskou zkušenost s aplikací.

Kapitola 9
Závěr

Cílem této práce bylo vyvinout vizualizační nástroj pro jednodušší vizualizaci vztahů ve vědecké komunitě. V teoretické části byly definovány pojmy používané v práci a byly probrány již existující podobné vizualizační aplikace. Také byl proveden návrh vlastního řešení a byla zvolena implementační platforma.

V analýze podobných vizualizačních aplikací bylo důležité se zaměřit na jejich vlastnosti a funkce. Stručně byla probrána práce s nástroji. Též byl probrán způsob vizualizace dat a byly zhodnoceny i náklady na předplatné za použití těchto nástrojů.

V návrhu vlastního řešení bylo cílem navrhnout nové řešení, specifikovat funkční a nefunkční požadavky a popsat doménový model. Hlavní funkce prototypu byly navrženy pomocí případů užití, a uživatelské rozhraní bylo navrženo pomocí drátěných modelů a principů Nielsenovy heuristiky.

Nakonec byla zvolena implementační platforma. Klientská část prototypu byla implementována pomocí knihovny React. Serverová část byla implementována pomocí technologie Node.js a databáze Neo4j.

V praktické části byla provedena implementace prototypu a jeho testování na reálných datech. Nedílnou součástí bylo vyhodnocení ekonomického přínosu pro fakultu. Jako poslední byl vypracován další rozvoj a studie proveditelnosti.

V implementaci prototypu byly popsány jednotlivé implementované React komponenty, integrace těchto komponent s knihovnou Neovis.js a způsob vizualizace dat. Na straně serveru byl popsán REST API server, který byl implementován pomocí technologie Node.js. Též byl popsán způsob ukládání dat do databáze Neo4j a byly probrány použité dotazy.

Následně byl prototyp otestován na reálných datech. Otestován byl jak manuálně, tak i pomocí jednotkových testů.

Posléze byl vyhodnocen ekonomický přínos vizualizačního nástroje pro fakultu. Výstupem je hlavně ušetřený čas zaměstnanců fakulty. Dalším přínosem může být vyšší kvalita publikací.

V dalším rozvoji byly navrženy další vhodné funkce pro lepší uživatelskou

zkušenost a byl navržen další typ vizualizace. Součástí bylo také vyhodnocení technické a ekonomické proveditelnosti.

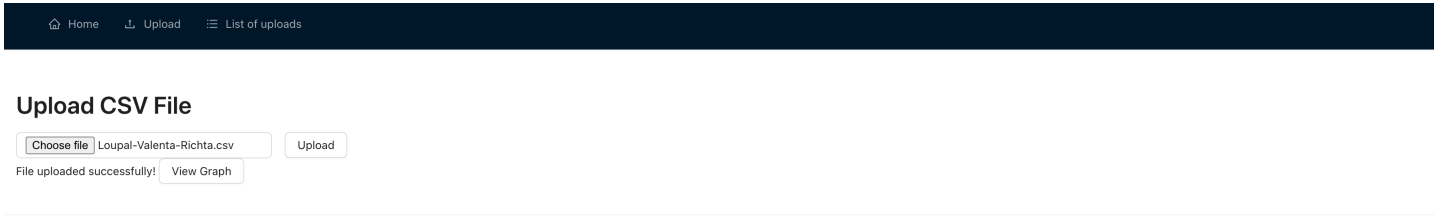
Příloha A

Vizualizace prototypu

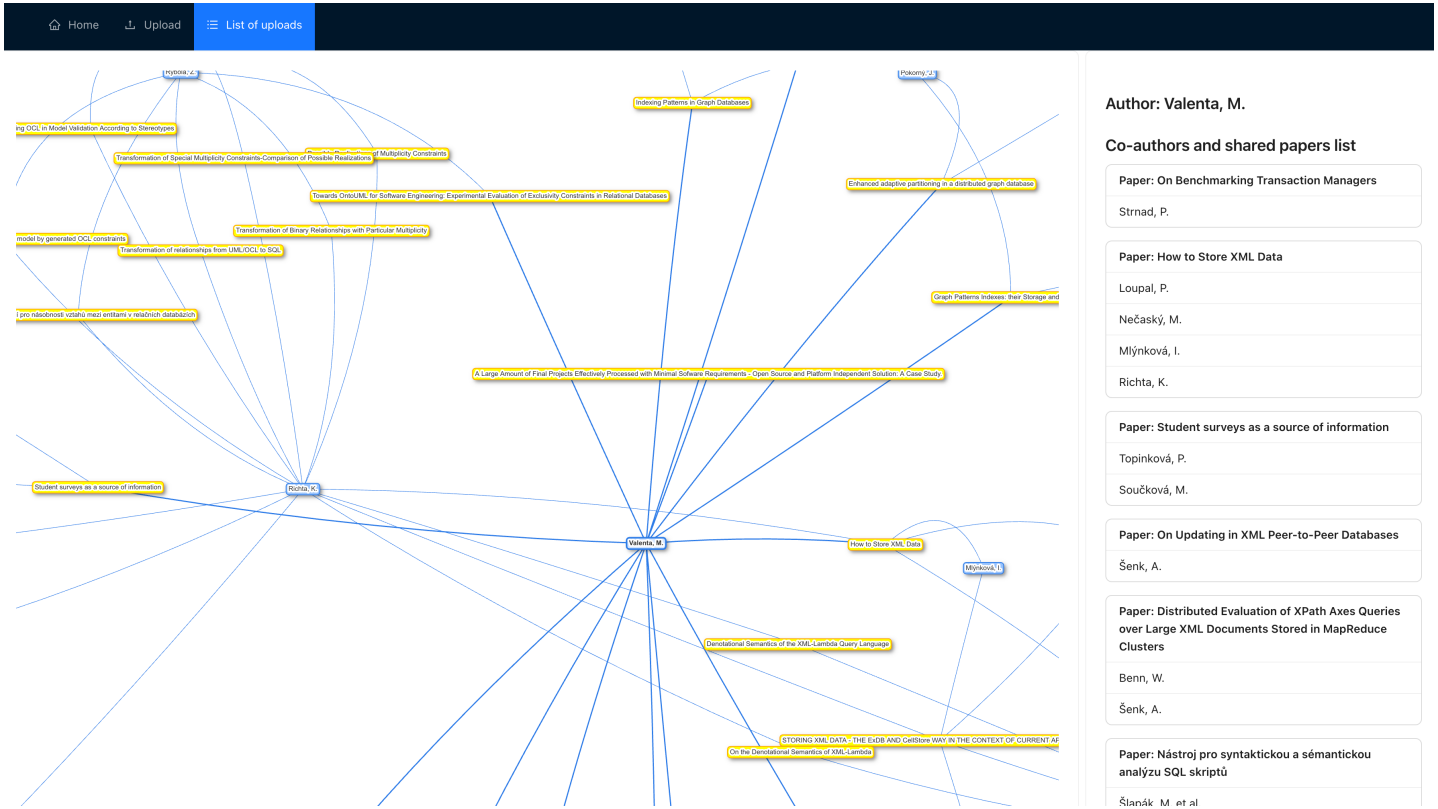
Na prvním obrázku je demonstrována domovská stránka aplikace. Na druhém obrázku je zobrazen formulář pro nahrávání souboru ze systému V3S. Na třetím obrázku je demonstrováno úspěšné nahrání souboru a odkaz na uložený graf, po rozkliknutí kterého se zobrazí vizualizace. Na čtvrtém obrázku je vizualizace nahraného souboru z předchozího obrázku. V souboru byly uloženy informace o publikacích autorů Valenty, Loupala a Richty. Na pátém obrázku je další vizualizace, která se týká autora Vítka. Na šestém obrázku je vzdálený pohled na předchozí vizualizaci pro Vítka. Na posledním obrázku je demonstrován seznam uložených grafů, který lze rozkliknout a získat tak již demonstrované vizualizace.

Obrázek A.1 Domovská obrazovka aplikace

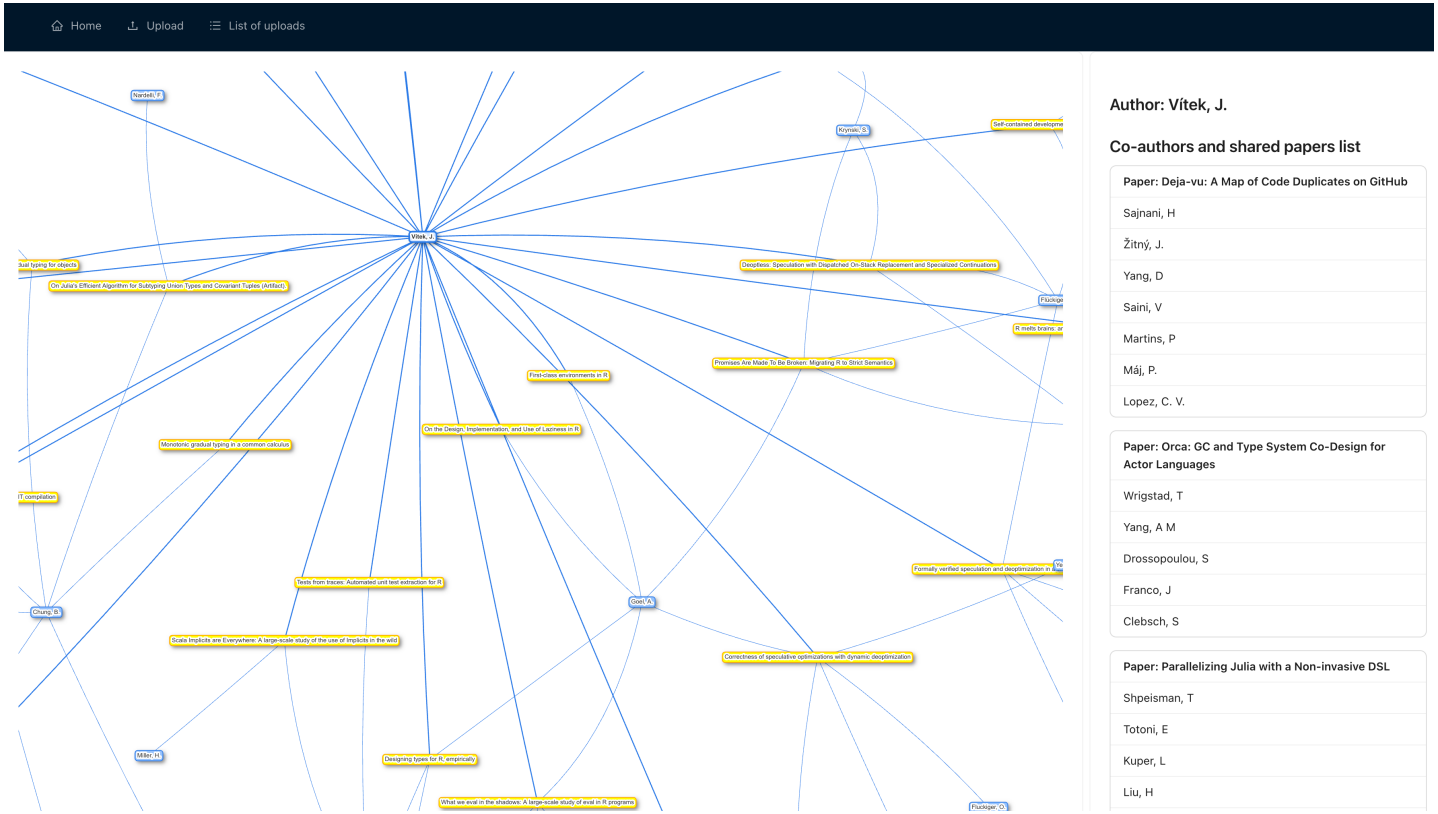
Obrázek A.2 Formulář pro nahrávání



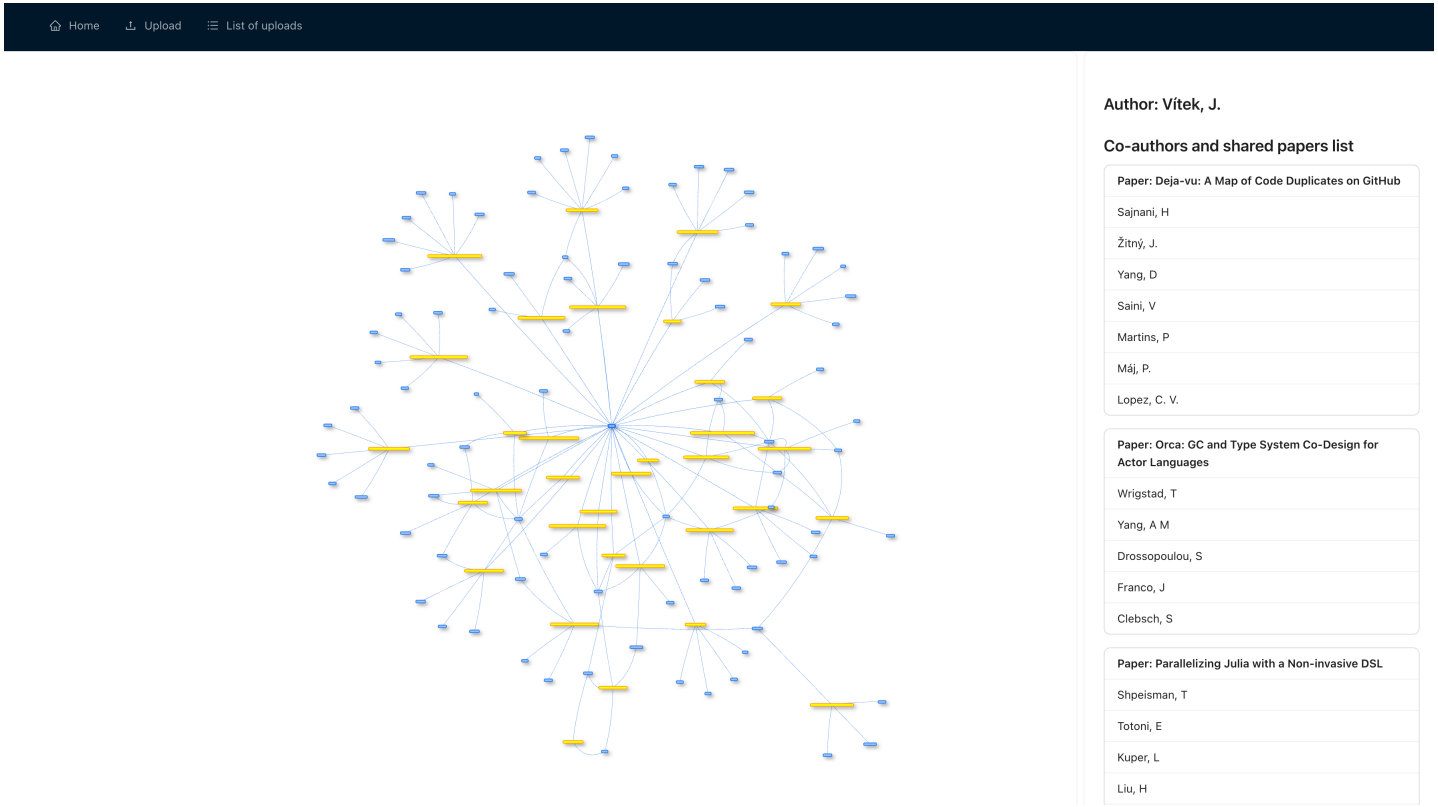
Obrázek A.3 Formulář po úspěšným nahrání



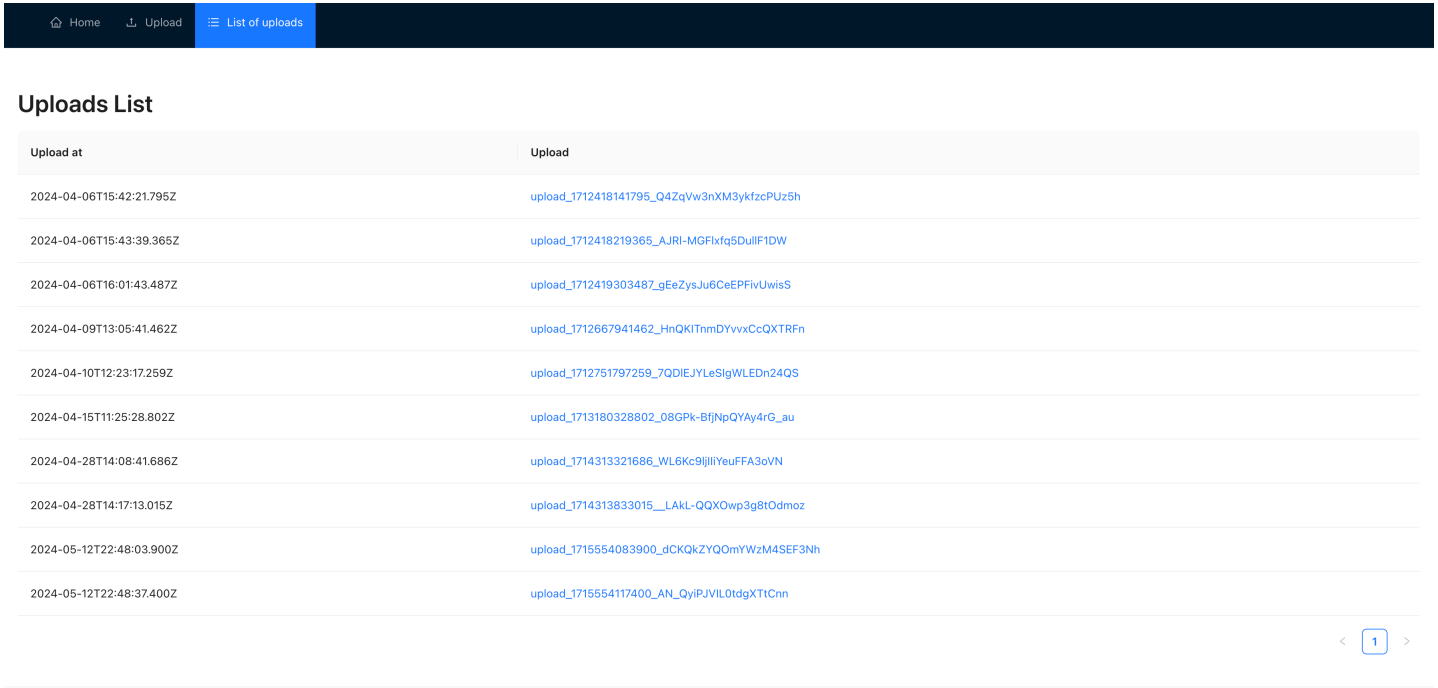
Obrázek A.4 Vizualizace pro autory Loupal, Valenta a Richta



Obrázek A.5 Vizualizace pro autora Vítek



Obrázek A.6 Vizualizace pro autora Vítek, vzdálený pohled

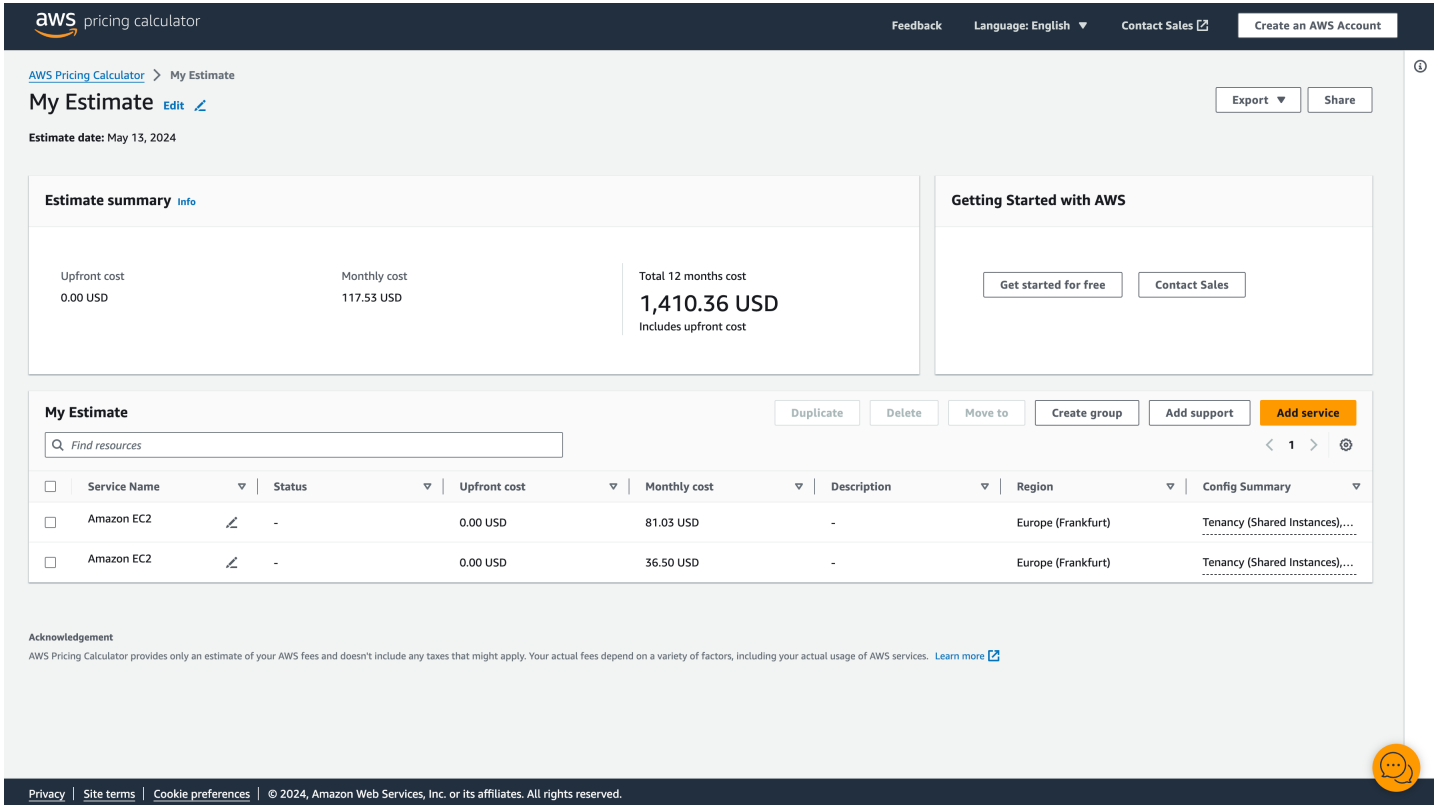


Obrázek A.7 Seznam nahraných vizualizací

Příloha B

Kalkulace infrastruktury AWS

Infrastruktura serveru zahrnuje virtuální server typu c5.xlarge pro serverovou část a jeden virtuální server typu c5a.large pro klientskou část. Tyto servery poskytují dostatečný výkon pro pilotní provoz, neboť jejich kapacity násobně přesahují minimální systémové požadavky využívaných technologií. Podrobný popis specifikací těchto serverů lze nalézt v tabulkách poskytovatele [40]. Konkrétně virtuální server c5.xlarge nabízí 4 vCPU (Virtual Central Processing Unit) a 8 GB RAM, zatímco virtuální server c5a.large disponuje 2 vCPU a 4 GB RAM.



Obrázek B.1 Kalkulačka AWS [38]

Bibliografie

1. Why your brain needs data visualization. In: *SAS analytics* [online]. [B.r.] [cit. 2024-03-22]. Dostupné z: https://www.sas.com/en_us/insights/articles/analytics/why-your-brain-needs-data-visualization.html.
2. Quantitative and qualitative data. In: *Australian Bureau of Statistics* [online]. [B.r.] [cit. 2024-03-21]. Dostupné z: https://www.abs.gov.au/statistics/understanding-statistics/statistical-terms-and-concepts/quantitative-and-qualitative-data.
3. KNOP, Dušan; OPLER Michal. Základy grafů. In: *BI-AG1* [online]. [B.r.] [cit. 2024-03-25]. Dostupné z: https://courses.fit.cvut.cz/BI-AG1/lectures/media/bi-ag1-p1-handout.pdf.
4. HOLTZ, Yan; HEALY Conor. Network diagram. In: *from Data to viz* [online]. [B.r.] [cit. 2024-03-25]. Dostupné z: https://www.data-to-viz.com/graph/network.html.
5. SAKR, Sherif; ZOMAYA, Albert Y. (ed.). Graph Databases. In: *Encyclopedia of Big Data Technologies*. Cham: Springer International Publishing, 2019, s. 835–835. ISBN 978-3-319-77525-8. Dostupné z DOI: 10.1007/978-3-319-77525-8_100147.
6. CONNECTED PAPERS. *Connected papers* [soft.]. 2024. Dostupné také z: https://www.connectedpapers.com/.
7. Speed up your literature search. In: *Connected papers* [online]. [B.r.] [cit. 2024-04-12]. Dostupné z: https://www.connectedpapers.com/pricing.
8. GEPHI – Introduction to Network Analysis. In: *Gephi* [online]. [B.r.] [cit. 2024-04-12]. Dostupné z: https://gephi.org/about/.
9. GRANDJEAN, Martin. GEPHI – Introduction to Network Analysis. In: *martingrandjean* [online]. [B.r.] [cit. 2024-04-12]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/gantt-chart-guide.

10. DE JONG, Niels. 15 Tools for Visualizing Your Neo4j Graph Database. In: *Neo4j Developer Blog* [online]. [B.r.] [cit. 2024-03-28]. Dostupné z: https://neo4j.com/developer-blog/15-tools-for-visualizing-your-neo4j-graph-database/.
11. Domain Model. In: *ictdemy* [online]. [B.r.] [cit. 2024-04-15]. Dostupné z: https://www.ictdemy.com/software-design/uml/uml-domain-model.
12. DALY, Nicky. What Is a Use Case? In: *Wrike* [online]. [B.r.] [cit. 2024-04-15]. Dostupné z: https://www.wrike.com/blog/what-is-a-use-case/.
13. NIELSEN, Jakob. 10 Usability Heuristics for User Interface Design. In: *Nielsen Norman Group* [online]. [B.r.] [cit. 2024-03-29]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.
14. 3 Benefits of Wireframing. In: *digital.ink* [online]. [B.r.] [cit. 2024-04-15]. Dostupné z: https://www.digital.ink/blog/3-benefits-wireframing/.
15. Low fidelity vs high fidelity wireframes: What's the difference? In: *uizard* [online]. [B.r.] [cit. 2024-04-20]. Dostupné z: https://uizard.io/blog/low-fidelity-vs-high-fidelity-wireframes/#what-is-a-low-fidelitywireframe.
16. MOQUPS. *Moqups App* [soft.]. 2024. Dostupné také z: https://app.moqups.com/.
17. DB-Engines Ranking of Graph DBMS. In: *db-engines* [online]. [B.r.] [cit. 2024-04-25]. Dostupné z: https://db-engines.com/en/ranking/graph+dbms.
18. What is a graph database? In: *Neo4j* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/.
19. 2023 Developer Survey. In: *Stack Overflow* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://survey.stackoverflow.co/2023/.
20. About Node.js. In: *Node.js* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://nodejs.org/en/about.
21. About npm. In: *NPM* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://docs.npmjs.com/about-npm.
22. SPA (Single-page application). In: *MDN* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/SPA.

23. Choose Between Traditional Web Apps and Single Page Apps (SPAs). In: *NPM* [online]. [B.r.] [cit. 2024-03-30]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.

24. Writing Markup with JSX. In: *React* [online]. [B.r.] [cit. 2024-03-31]. Dostupné z: https://react.dev/learn/writing-markup-with-jsx.

25. Hooks FAQ. In: *React* [online]. [B.r.] [cit. 2024-03-31]. Dostupné z: https://legacy.reactjs.org/docs/hooks-faq.html#which-versions-of-react-include-hooks.

26. Reusing Logic with Custom Hooks. In: *React* [online]. [B.r.] [cit. 2024-04-01]. Dostupné z: https://react.dev/learn/reusing-logic-with-custom-hooks.

27. Bolt Protocol. In: *Neo4j* [online]. [B.r.] [cit. 2024-04-01]. Dostupné z: https://neo4j.com/docs/bolt/current/bolt/.

28. What is a REST API. In: *Red Hat* [online]. [B.r.] [cit. 2024-04-10]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

29. NEVYHOŠTĚNÝ, Petr. Používáme Node.js streamy. In: *Zdroják* [online]. [B.r.] [cit. 2024-04-09]. Dostupné z: https://zdrojak.cz/clanky/pouzivame-node-js-streamy/.

30. MERGE. In: *Neo4j* [online]. [B.r.] [cit. 2024-04-01]. Dostupné z: https://neo4j.com/docs/cypher-manual/5/clauses/merge/.

31. What is Functional Testing? In: *testsigma* [online]. [B.r.] [cit. 2024-04-10]. Dostupné z: https://testsigma.com/guides/functional-testing/#Manual_Functional_Testing.

32. White Box Testing. In: *Imperva* [online]. [B.r.] [cit. 2024-05-07]. Dostupné z: https://www.imperva.com/learn/application-security/white-box-testing/.

33. Mock Functions. In: *jestjs* [online]. [B.r.] [cit. 2024-04-19]. Dostupné z: https://jestjs.io/docs/mock-functions.

34. MAU, Derek. Základní průvodce Ganttovými diagramy. In: *Microsoft-365* [online]. [B.r.] [cit. 2024-05-01]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/gantt-chart-guide.

35. Člověkoden (Man-day). In: *managementmania* [online]. [B.r.] [cit. 2024-05-01]. Dostupné z: https://managementmania.com/cs/clovekoden-manday.

36. Node.js developer platy. In: *jooble* [online]. [B.r.] [cit. 2024-05-01]. Dostupné z: https://cz.jooble.org/salary/node.js-developer.

37. Cloud computing with AWS. In: *Amazon AWS* [online]. [B.r.] [cit. 2024-05-02]. Dostupné z: `https://aws.amazon.com/what-is-aws/`.

38. AWS Pricing Calculator. In: *Calculator AWS* [online]. [B.r.] [cit. 2024-05-13]. Dostupné z: `https://calculator.aws/`.

39. Co je GDPR. In: *MVČR* [online]. [B.r.] [cit. 2024-05-01]. Dostupné z: `https://www.mvcr.cz/gdpr/clanek/co-je-gdpr.aspx`.

40. Amazon EC2 C5 Instances. In: *Amazon AWS* [online]. [B.r.] [cit. 2024-05-13]. Dostupné z: `https://aws.amazon.com/ec2/instance-types/c5/`.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace, včetně diagramů
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `wireframes` ..... navržený drátový model
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF