FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Efektivní navigace ve znalostní bázi MBI
Student:	Bc. Marek Neumann
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

MBI [1] je otevřená sada metodických pokynů a postupů ohledně strategického rozhodování, plánování i provozního řízení informatického oddělení firmy vyvinutá na VŠE Praha. Cílem této práce je revize a návrh nového přístupu k navigaci uživatele - typicky vedoucího ICT oddělení - v této znalostní bázi.

1. Seznamte se s aktuálním stavem portálu MBI [1] - jeho obsahem i strukturou.
2. Seznamte se s pracemi [2], [3] a [4], které se efektivní navigaci uživatele v MBI věnují.
3. Revidujte datový model MBI prezentovaný ve [2] a [4].
4. Analyzujte přístupy k intuitivní navigaci uživatelů v MBI zpracované ve [2] a [4].
5. Navrhněte vlastní způsob intuitivní navigace znalostní bází, který bude založen na revidovaném datovém modelu MBI.
6. Vytvořte funkční prototyp a proveďte na něm uživatelské testování.
7. Zhodnoťte výsledky včetně odhadu finančních nákladů a benefitů spojených s realizací návrhu, případně navrhněte další úpravy.

Seznam odborné literatury

[1] MBI portál https://mbi.vse.cz/
[2] Batík Ondřej: Dotazovací jazyk pro vztahy MBI objektů. Bakalářská práce na FIT ČVUT. 2016.
[3] Stránský Vojtěch: Vizualizace vztahů v informační bázi MBI. Bakalářská práce na FIT ČVUT. 2015.
[4] Stránský Vojtěch: Návrh a implementace software pro interaktivní práci s objekty MBI pomocí jazyka MBIQL. Magisterská práce na FIT ČVUT. 2017.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 19. ledna 2019

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Efektivní navigace ve znalostní bázi MBI

Bc. Marek Neumann

Vedoucí práce: Ing. Michal Valenta, Ph.D.

9. ledna 2020

Poděkování

Nejprve bych chtěl poděkovat vedoucímu mé diplomové práce za vstřícný přístup, poskytnuté konzultace a trpělivost při psaní této práce. Velké díky patří také mé rodině za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2020 .....

Odkaz na tuto práci

Neumann, Marek. *Efektivní navigace ve znalostní bázi MBI.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem této práce je analyzovat existující přístupy k navigaci v bázi MBI (Management Byznys Informatiky), navrhnout nový, intuitivní způsob navigace a ověřit tento návrh na prototypu.

Na základě analýzy současného stavu MBI a dosavadních přístupů k navigaci je navržen grafický dotazovací jazyk pro navigaci v MBI. Tento jazyk je poté porovnán s předchozím způsobem navigace z práce Ondřeje Batíka, a je navržena jeho možná implementace. Nakonec je proveden návrh a implementace prototypu aplikace pomocí frameworku Angular a knihoven D3.js a popoto.js s úložištěm dat v grafové databázi Neo4j. Tento prototyp demonstruje princip dotazování v navrženém dotazovacím jazyce.

Výsledkem práce je tedy analýza a návrh řešení grafického dotazovacího jazyka spolu s návrhem implementace, a prototypem aplikace, který demonstruje princip dotazování v tomto jazyce. Uživateli je tak nabídnuta alternativní možnost pro získávání informací z MBI interaktivně a bez složitých uživatelských rozhraní.

Klíčová slova dotazovací jazyk, interaktivní grafické prvky, grafová databáze, Neo4j, Cypher, JavaScript, Angular, D3.js, popoto.js

Abstract

The goal of this work is to analyze the existing methods for navigation in the MBI database (Management of Business Informatics), devise and design a new, intuitive way of navigation and verify the design on a prototype application.

Based on the analysis of the current state of MBI and the existing methods of navigation a new graphical query language for navigation in MBI is designed. The language is then compared with one of the previous methods of navigation from the work of Ondřej Batík, and a possible implementation of the language is devised. Finally, a prototype application of the language is designed and implemented in the framework Angular with visualization libraries D3.js and popoto.js and a data storage in the Neo4j database. This prototype demonstrates the fundamentals of the new query language.

The result of this work is the analysis and design of a graphical query language along with a concept for possible full-fledged implementation and a prototype which demonstrates the fundamentals of using the new language. Users are thus offered an alternative method to gain information from the MBI database in an interactive way without complex user interfaces.

Keywords query language, interactive graphical elements, graph database, Neo4j, Cypher, JavaScript, Angular, D3.js, popoto.js

Obsah

Seznam obrázků

Úvod

Motivace

V současné éře Internetu je k dispozici nespočet informací, a řízení informatiky ve firmách není žádnou výjimkou. Problémem již není dostupnost informací, ale jejich organizace a verifikace. Portál MBI (Management Byznys Informatiky) je webový zdroj, který se snaží tyto informace poskytovat ve srozumitelné podobě a vytvořit mezi nimi strukturu a kontext. Díky tomu však vzniká mnoho vztahů, které tvoří hustou síť navzájem propojených, souvisejících Objektů. Může tak být obtížné tyto vztahy prezentovat vztahy přehledným způsobem

Vhodným, a často používaným nástrojem pro vizualizaci vztahů jsou grafové databáze. Možnost vizualizace vztahů v bázi MBI pomocí grafové databáze byla poprvé popsána a realizována v práci „Vizualizace vztahů v informační bázi MBI“ [1], kde byly Objekty báze MBI transformovány a nahrány do databáze Neo4j, kde bylo možné se dotazovat pomocí Cypher. Tento přístup však představoval překážku pro pro netechnicky orientované uživatele.

Další, alternativní řešení bylo popsáno v práci „Dotazovací jazyk pro vztahy MBI Objektů“ [2], kde byl navržen dotazovací jazyk *MBIQL*. Ten využívá grafických prvků pro formování dotazů a následnou vizualizaci výsledků. Tento návrh byl posléze implementován v práci „Návrh a implementace SW pro interaktivní práci s Objekty MBI pomocí jazyka MBIQL“ [3]. Výstupem byla aplikace, která umožňuje dotazovat se nad grafovou databází pomocí grafického uživatelského rozhraní a výsledky dotazování poté zobrazit ve formě grafu.

Dosavadní přístupy tedy představují krok správným směrem k intuitivnější navigaci v bázi MBI. Je však názorem autora, že je možné zajít dále a plně využít vizuální reprezentace báze MBI pro zobrazení vztahů mezi Objekty.

Cílem této práce je tedy navrhnout způsob navigace v MBI založený na interakci přímo s vizuální formou grafové databáze. Nakonec pak realizovat

prototyp, který by demonstroval tento způsob navigace, a zhodnotit možné přínosy a náklady spojené s možnou realizací tohoto návrhu.

Struktura práce

Práce se skládá z následujících kapitol:

První kapitola (viz 1) seznamuje s Portálem MBI a jeho strukturou.

Druhá kapitola (viz 2) se zabývá způsoby dotazování a navigace v bázi MBI. Analyzuje dosavadní přístupy, zejména z pohledu uživatelské použitelnosti, a nastiňuje alternativní přístup.

Ve třetí kapitole (viz 3) je zanalyzován současný datový model pro grafovou databázi MBI a zhodnoceno zda je nutné provést změny modelu pro navrhovaný způsob navigace.

Čtvrtá kapitola (viz 4) se zabývá návrhem navigačního jazyka nad datovým modelem MBI.

Pátá kapitola (viz 5) srovnává navrhovaný navigační jazyk s přístupem jazyka MBIQL.

V šesté kapitole (viz 6) je popsán návrh implementace navigačního jazyka.

Sedmá kapitola (viz 7) popisuje návrh jednoduchého prototypu, který by demonstroval princip navigačního jazyka.

V osmé kapitole (viz 8) jsou popsány technologie zvolené k implementaci prototypu a důvody jejich volby. Jsou také uvedeny některé alternativní technologie.

Ve deváté kapitole (viz 9) je realizována implementace prototypu na základě předchozího návrhu a zvolených technologií.

V desáté kapitole (viz 10) je prototyp otestován z pohledu správnosti zobrazovaných dat a z hlediska uživatelské použitelnosti.

V poslední kapitole (viz 11) je zhodnocen celý návrh navigačního jazyka společně s přínosy a náklady spojenými s možnou realizací.

Kapitola 1

MBI (Management Byznys Informatiky)

1.1 Definice MBI

„MBI (Management Byznys Informatiky) je portál obsahující zobecněná řešení v řízení provozu a rozvoje IT, resp. podnikové informatiky. Jeho smyslem je sdílet a využívat znalosti a doporučení vyplývající z praxe a z dalších zdrojů.“ [4]

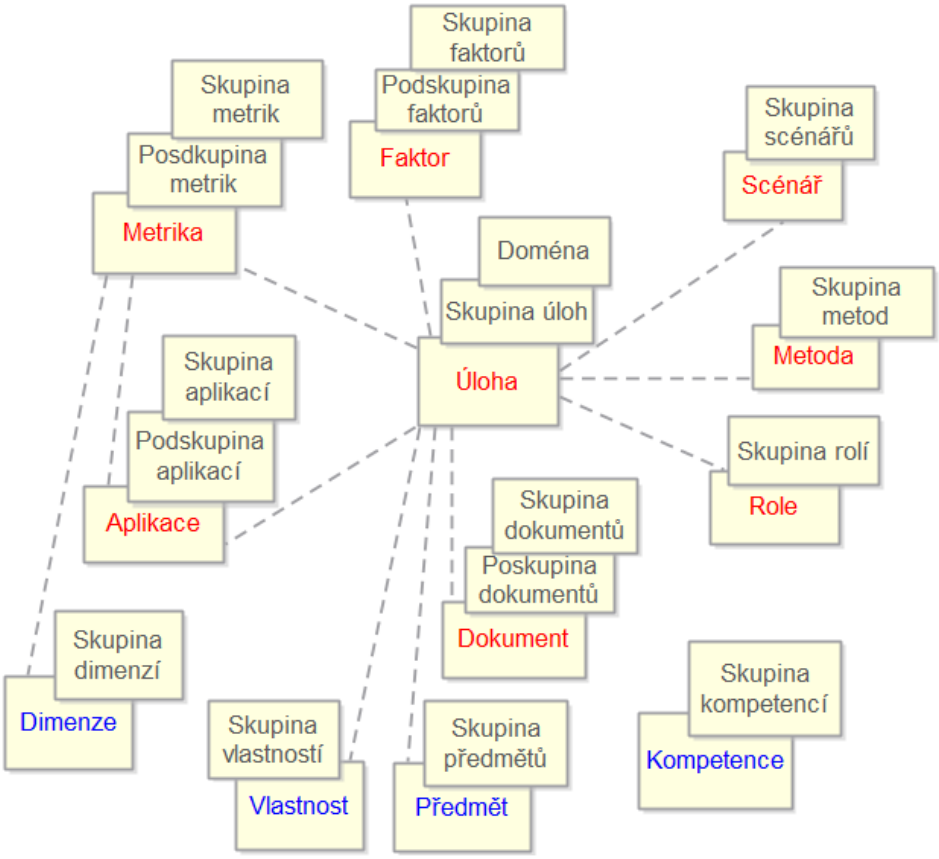
Jedná se tedy o sbírku postupů a metodik, které se zabývají řízením Informačních Technologií v podniku. Tyto postupy jsou organizovány do hierarchické struktury Objektů, které jsou navzájem mezi sebou propojeny. Momentálně jsou tyto informace dostupné přímo přes Portál MBI a přes dotazovací nástroje nad grafovou databází vytvořené v rámci předešlých prací [1] [2] [3].

1.2 O portálu MBI

Portál MBI byl vytvořen Katedrou informačních technologií na Vysoké škole ekonomické v Praze za účelem zpřístupnění informací o řízení podnikové informatiky pro širší veřejnost. Portál je v současné době přístupný pro kohokoliv, je pouze nutné se bezplatně registrovat.

Hlavní účel portálu je přispívat k zefektivnění práce podnikových analytiků, manažerů a specialistů zodpovědných za řízení informatiky ve společnostech. Poskytuje aktuální informace, zkušenosti a doporučení pro řešení konkrétních úkolů a to na různé úrovni podrobnosti podle potřeb uživatele. Kromě samotných informací jsou k dispozici také vzorové dokumenty, prezentace a aplikace, které si může uživatel stáhnout.

Doporučení, jak nejlépe se samotným portálem pracovat je možné najít přímo hlavní stránce portálu či na odkazu [5].



Obrázek 1.1: Struktura Objektů modelu MBI [2]

1.3 Struktura datového modelu MBI

Informační báze MBI je postavena na datovém modelu dvanácti navzájem propojených Objektů, ze kterých může uživatel čerpat informace. Tyto Objekty je možné dále rozdělit na hlavní a podpůrné. Každý z těchto Objektů nich má také svou vlastní hierarchickou strukturu.

Struktura všech Objektů a jejich vzájemné vazby jsou viditelné na obrázku 1.1. Podle vazeb uživatel získává přehled o tom, jak jsou mezi sebou Objekty provázány a měl by také být schopný si význam jednotlivých vazeb intuitivně odvodit.

Následující sekce čerpají z doporučení jak nejlépe používat portál MBI zmíněných výše [5].

1.3.1 Hlavní Objekty datového modelu

1.3.1.1 Úloha

Úloha představuje „centrální“ Objekt celého modelu a všechny ostatní Objekty jsou na ni přímo navázány, s výjimkou Objektu Kompetence. Tento Objekt tak slouží jako hlavní přístupový bod do modelu. Úlohy mají za cíl poskytovat informace a postupy pro řešení konkrétních problémů.

Příklady

- Úloha: Revize IT strategie dle požadavků byznysu
- Skupina: IT jako součást byznysu
- Doména: Strategické řízení IT

1.3.1.2 Scénáře

Scenář definuje konkrétní reálné situace a problémy a poskytuje informace o tom jak se v těchto situacích zachovat a problémy řešit. Na tento Objekt jsou vázána doporučená řešení Úloh MBI, Scénář tak lze vnímat jako alternativní přístupový bod do modelu.

Příklady

- Scénář: Analyzují se náklady na IT
- Skupina: Scénáře v řízení IT ekonomiky

1.3.1.3 Faktor

Faktor představuje souhrnné vyjádření pro nějaký předmět a s ním související podmínky řízení IT. Definuje existující prostředí informatiky a udržuje bázi v souladu s vývojovými trendy.

Příklady

- Faktor: Faktor velikosti podniku
- Podskupina: Velikost podniku
- Skupina: Byznys prostředí

1.3.1.4 Role

Role určuje typy pracovních pozic v podniku, které jsou vázané na úlohy, a jakým způsobem se podílejí na rozvoji informatiky i celého podniku. Role jsou členěny dle jejich kompetence na základě principu RACI [1].

[1] Responsible, Accontabble, Consulted, Informed

Příklady
- Role: IT architekt
- Skupina: Vývojáři

1.3.1.5 Dokument

Dokument je libovolná datová struktura, která reprezentuje vstup nebo výstup dané Úlohy. Může se jednat dokumenty papírové, elektronické, databáze, reporty, atd.

Příklady
- Dokument: Katalog datových zdrojů
- Podskupina: Dokumenty řízení datových zdrojů
- Skupina: Dokumenty řízení IT zdrojů

1.3.1.6 Metrika

Metrika poskytuje data, která slouží k řízení kvality a výkonnosti nejen informatiky, ale i jiných aktivit podniku. K Metrice jsou vázány další související metriky a definované analytické dimenze, které slouží k identifikaci a posouzení sledovaných ukazatelů.

Příklady
- Metrika: Rozpracovanost IT projektů
- Podskupina: Metriky plánování IT služeb
- Skupina: Metriky řízení IT služeb

1.3.1.7 Aplikace

aplikace zahrnuje veškeré analytické, plánovací a jiné aplikace a nástroje, které je možné použít pro podporu řízení IT.

Příklady
- Aplikace: EIOrig
- Podskupina: Aplikace pro podporu provozu podniku
- Skupina: Aplikace a nástroje pro řízení podniku

1.3.1.8 Metoda

Metoda představuje ověřené manažerské, analytické a jiné metodiky, metody a postupy, které lze aplikovat při řízení IT a realizovat cíle dané úlohy.

Příklady
- Aplikace: Metody plánování projektů
- Skupina: Metodiky a metody řízení projektů

1.3.2 Podpůrné Objekty modelu

Podpůrné Objekty poskytují pouze doplňující informace k datům obsaženým v hlavních Objektech a nepředstavují v bázi MBI tak významnou roli. Proto jsou označeny jako „podpůrné“.

1.3.2.1 Dimenze

Dimenze slouží jako analytická hlediska pro sledování a hodnocení jednotlivých Metrik.

Příklady
- Dimenze: IT projekty
- Skupina: IT služby a zdroje

1.3.2.2 Vlastnosti

Vlastnost představuje základní vlastnosti informatiky v podniku, které daná úloha ovlivňuje.

Příklady
- Vlastnost: Výkonnost podnikové informatiky
- Skupina: Vlastnosti podnikové informatiky jako celku

1.3.2.3 Objekt řízení

Předmět reprezentuje hlavní předměty řízení, které jednotlivé úlohy ovlivňují. V bázi MBI se instance Objektu Předmět vyskytují pod názvem Objekt řízení.

Příklady
- Předmět: IT služby
- Skupina: Předměty řízení podnikové informatiky

1.3.2.4 Kompetence

Kompetence představují přehled standardních kompetencí zaměstnanců podniku ve vztahu k II i podnikovému řízení. Stávají se součástí Objektu Role.

Příklady
- Kompetence: Integrace systémů
- Skupina: Technické kompetence

Kapitola 2

Navigace v bázi MBI

Následující kapitola se zabývá dotazováním a navigací v bázi MBI a vychází zejména z analýzy provedené v předchozích pracích [2] a [3]. Nejprve je shrnuta analýza uživatelů, poté jsou popsány jednotlivé metody dotazování a navigace, jejich výhody a nedostatky. Nakonec je popsán princip alternativního přístupu k navigaci, kterým se zabývá tato práce.

2.1 Analýza uživatelů a typů dotazů

2.1.1 Uživatelé

Uživatele Informační báze MBI můžeme zhruba rozdělit do tří skupin.

První skupinou jsou zaměstnanci firem, které nevyužívají informační technologie jako hlavní záměr podnikání [6]. Na ty je portál primárně zaměřen a měli by tak představovat nejčastější uživatele.

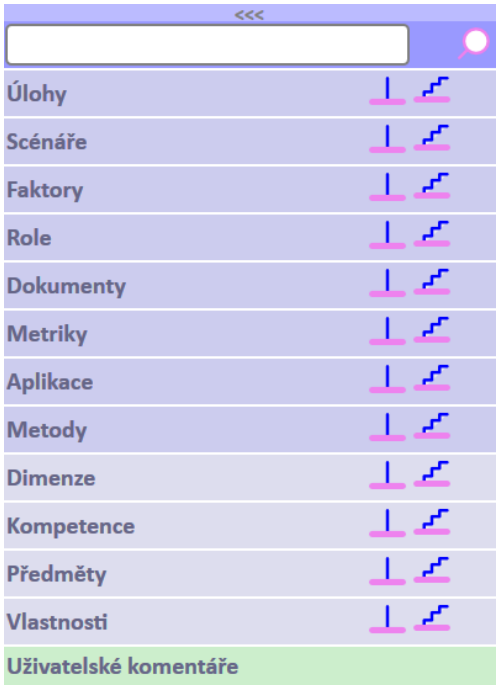
Druhou skupinu uživatelů, jsou potenciální manažeři z oboru podnikové informatiky, kterým se portál MBI nabízí jako ideální zdroj informací k profesnímu vývoji.

Poslední skupina představuje všechny ostatní uživatele se zájmem o obor, protože přístup do báze MBI má kdokoliv.

2.1.2 Typy dotazů

Dotazy, které by od identifikovaných skupin měly přicházet jsou směřované na oblast podnikové informatiky. Zejména dotazy od skupiny firemních zaměstnanců by měly být velmi cílené, protože budou pravděpodobně souviset s problémem či úkolem, který momentálně řeší.

Obecně lze za nejčastější dotazy považovat ty, které souvisí s Objekty Úloha a Scénář. Úloha obsahuje vazby téměř na všechny ostatní Objekty modelu a představuje tak ideální vstupní bod 1.1. Scénář potom představuje



Obrázek 2.1: Postranní menu portálu MBI [4]

specifické reálné situace a problémy, je tedy pravděpodobné, že bude přímo obsahovat řešení hledaných problémů.

Za nejméně časté dotazy lze považovat ty, směřované na podpůrné Objekty, které mají pouze doplňující význam pro model.

Na základě analýzy tedy lze usoudit, že nejčastější typy dotazů nejsou příliš složité a jasně specifikované. Nicméně je potřeba zohlednit volnost přístupu k Portálu MBI. Najde se tak nepochybně i řada komplexnějších dotazů, zejména od zájemců o obor z poslední skupiny.

2.2 Dotazování v Portálu MBI

Doporučený způsob dotazování a přístupu k Objektům v portálu MBI je skrze panel v levé části portálu 2.1. Přes tento panel je možné se dotazovat třemi dostupnými variantami [5].

- **Dotazování přes Objekt:** První metodou pro přístup do modelu je přímý výběr konkrétní instance Objektu. Po zvolení této metody se zobrazí seznam všech instancí vybraného Objektu v databázi.
- **Dotazování přes hierarchie:** Jako druhou možnost lze k přístupu ke konkrétní instanci Objektu využít hierarchii modelu. Nejprve se tedy

zvolí příslušná Skupina či Podskupina do které Objekt náleží. Nakonec se opět zvolí samotná instance Objektu. Není však potřeba volit konkrétní instanci, lze zobrazit prvky vybrané Skupiny či Podskupiny.

- **Dotazování přes vyhledávání:** Poslední metodou přístupu k Objektům je fulltextové vyhledávání nad všemi Objekty báze MBI. Uživatel zadá řetězec a Portál mu vrátí všechny Objekty, ve kterých se řetězec vyskytuje.

Po zvolení či nalezení požadovaného Objektu ze uživateli zobrazí detailní informace o Objektu. Je také možné si zobrazit matici souvisejících Objektů, případně z ní přejít na jednotlivé související Objekty.

2.2.1 Shrnutí stavu

Pro dotazování přímo přes Portálu MBI byly identifikovány následující výhody a nedostatky.

2.2.1.1 Výhody

Další předností Portálu je rozhraní postranního panelu k dotazování, které je přehledné a intuitivní, a umožňuje uživateli se rozhodnout jakým způsobem se nad Objekty dotazovat.

Další výhodou je množství informací, které uživatel obdrží o jednotlivých Objektech a několik úrovní podrobnosti, ve kterých si informace může zobrazit, od stručných až po velmi detailní. Je také možné s informacemi dále pracovat díky možnosti stažení různých dokumentů a šablon.

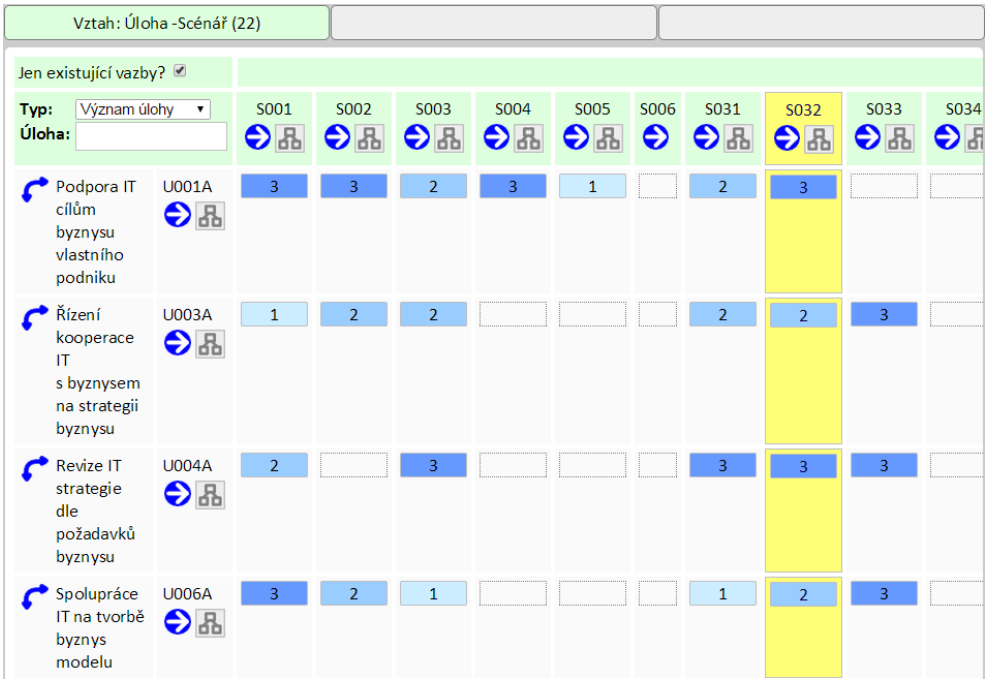
2.2.1.2 Nedostatky

Hlavní přednost portálu lze považovat i za potenciální nedostatek, a to množství informací o jednotlivých Objektech. Detailní stránky jednotlivých instancí Objektů mohou obsahovat velké množství dat a může tak být problematické se v nich zorientovat. Tento problém je částečně řešen různými úrovněmi podrobnosti.

Za hlavní nedostatek relevantní k tématu práce lze považovat zobrazování vztahů mezi Objekty. To je realizováno pomocí matice vztahů, která s rostoucím počtem vztahů nabývá na rozměrech ztrácí na přehlednosti, v extrémních případech se nemusí ani vejít na obrazovku 2.2.

2.2.1.3 Zhodnocení

Na základě předchozí analýzy detailněji popsané v práci [2] je vyvozen závěr, že Portál MBI je více než dostačující pro poskytování informací o samotných Objektech.



Obrázek 2.2: Příklad matice vztahů Portálu MBI [2]

Nicméně nedostatky nepřehlednosti vyplývající z vizualizace vztahů mezi Objekty by bylo vhodné navrhnout alternativní přístup, který mohou poskytnout právě grafové databáze.

2.3 Dotazování pomocí jazyka MBIQL

Na základě předešlé analýzy byl navržen jazyk MBIQL, který je detailně popsán v práci [2] a posléze implementován v práci [3]. Nejdůležitější body návrhu a implementace jsou popsány v této podkapitole, která čerpá z výše zmíněných prací.

2.3.1 Definice jazyka MBIQL

„*MBIQL (Management of Business Informatics Query Language) je graficky orientovaný a doménově specifický dotazovací jazyk, který je určen pro dotazování se na data MBI. Dotaz jazyka MBIQL je vytvořen pomocí jedné ze dvou metod: Specificky orientované dotazování a Dotazování pomocí vazeb. Výsledný dotaz je následně automaticky překládán do jazyka Cypher.*“ [2]

2.3.2 Rozdíl v technologii

Zásadním rozdílem mezi Portálem MBI a jazykem MBIQL je využití grafové databáze Neo4j jako úložiště dat a jazyka Cypher pro komunikaci s databází. Grafové databáze jsou vynikajícím nástrojem pro vizualizaci vztahů mezi Objekty a jeví se tak jako vhodný kandidát pro řešení problému matic vztahů Portálu MBI.

2.3.3 Datový model

V rámci návrhu MBIQL prošel změnou datový model MBI. K původním Objektům instancí byly plnohodnotně přidány i Objekty hierarchické, dále také tzv. „modelové“ meta Objekty, které sjednocují všechny instance daného Objektu (např. Objekty typu Úloha). Byly pro ně rovněž nadefinovány odpovídající typy vazeb.

2.3.4 Ovládání jazyka

Jazyk MBIQL využívá několik skupin grafických prvků pro vytváření dotazů, které slouží pro výběr a specifikaci Objektů a jejich vztahů. Po zformování dotazu pomocí těchto prvků je dotaz přeložen do jazyka Cypher a vykonán nad databází Neo4j. Výsledek dotazu je poté vizualizován jako graf uživateli.

K dispozici je několik způsobů dotazování, které jsou popsány níže, včetně rozdílů ve výsledné implementace.

2.3.4.1 Specificky orientované dotazování

Tento způsob dotazování se zaměřuje na jednoduché dotazy orientované na specifické instance Objektů. 2.3

Uživatel tedy zvolí typ Objektu, ke kterému náleží hledaná instance. Dále vybere jeden z možných atributů Objektu. Nakonec zvolí hodnotu, které by daný atribut měl nabývat, buď ze seznamu všech možných hodnot, nebo hodnotu zadá manuálně do Textového pole s našeptáváním hodnot.

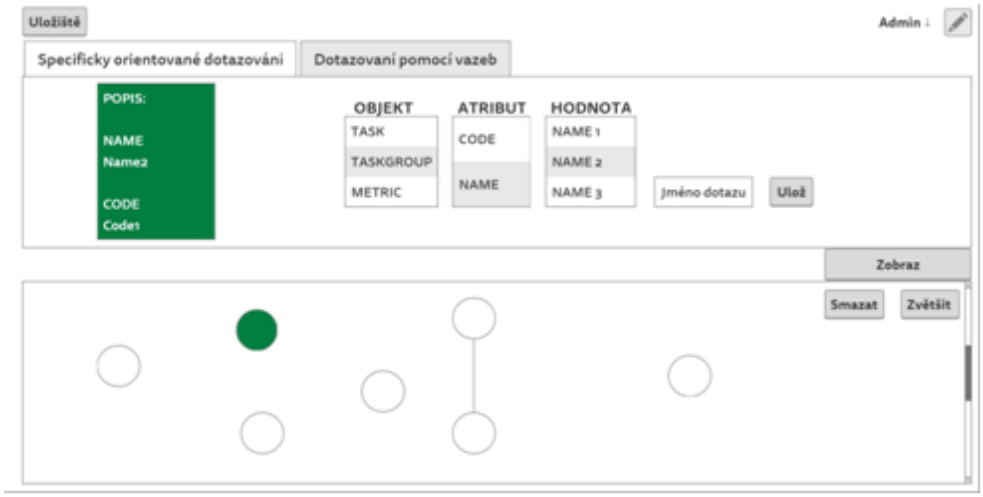
Dotaz je poté přeložen do jazyka Cypher a výsledek zobrazen.

Rozdíly v implementaci

- Uživatel je schopen volit kromě instančních Objektů i Objekty hierarchické.
- Atributy Objektů lze volit pouze ze seznamu, našeptávací textové pole zatím nebylo implementováno.

2.3.4.2 Dotazování pomocí vazeb

Tento způsob dotazování se zaměřuje na složitější dotazy, které se soustředí na vztahy mezi zvolenými Objekty MBI a hledání souvislostí mezi těmito



Obrázek 2.3: Wireframe - Specificky orientované dotazování [3]

Objekty. U každého Objektu lze specifikovat jeho atributy a jejich hodnoty. Formování dotazu je rozděleno na dvě části. 2.4

Struktura dotazu Nejprve uživatel ze seznamy vybere počáteční Objekt ze kterého začne vztahy mezi Objekty zkoumat. V následných cyklech poté volí vazby mezi Objekty. Lze volit vazby vedoucí z počátečního Objektu a posléze také vazby vedoucí z libovolného Objektu, který byl přidán vazbou v předchozím cyklu. Každou vazbu lze vybrat pouze jednou. Vazby lze mazat a volit tak jinou cestu grafem.

Lze také zvolit zda budou ve výsledné vizualizaci zahrnuty i Modelové uzly.

Specifikace dotazu V tomto kroku dotazování uživatel pracuje s Objekty a Vazbami, které zvolil v předchozím kroku. Nejprve zvolí prvky, které chce dále specifikovat. Pokud je zvoleno více prvků, přiřadí jim uživatel z nabídky logických operátorů (OR, AND). Pořadí prvků a operátorů může uživatel měnit.

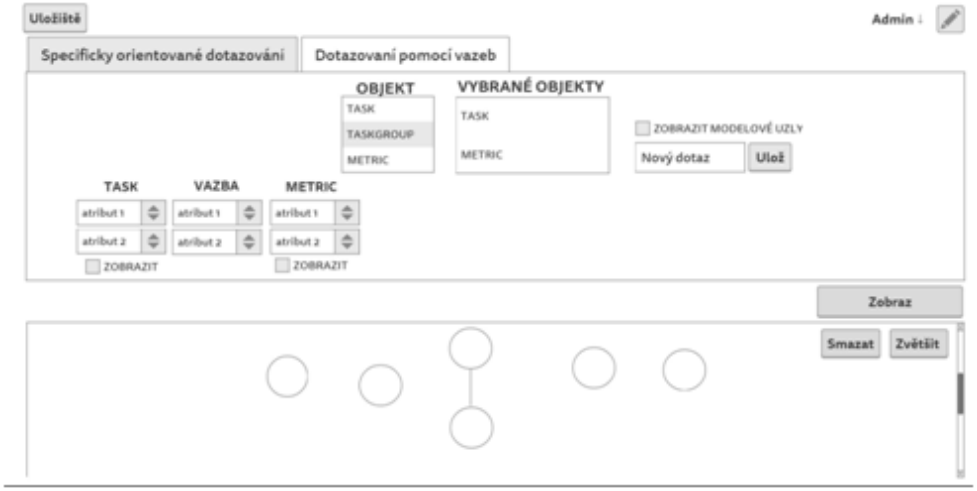
Následně zvolí uživatel postupně pro každý prvek požadované atributy, buď ze seznamu, nebo manuálně do textového pole. Mezi atributy je opět možné přiřadit logické operátory.

V posledním kroku uživatel zvolí návratové hodnoty dotazu ze zvolených prvků.

Poté je dotaz opět přeložen do jazyka Cypher a výsledek zobrazen uživateli.

Rozdíly v implementaci

- Uživatel nevolí vazby mezi Objekty, ale přímo související Objekty. Vazby jsou doplněny automaticky.



Obrázek 2.4: Wireframe - Dotazování pomocí vazeb [3]

- Uživateli jsou zobrazeny všechny zvolené prvky, nespecifikuje jejich vlastnosti v cyklu.
- Není zatím možné určovat logické operátory OR a AND.

2.3.4.3 Asistované procházení grafem

Poté co je jeden z předchozích typů dotazu přeložen a zobrazen, bylo navrženo několik rozšiřujících metod jak z výsledku získávat další informace. Ty byly souhrnně nazvány „Asistované procházení grafem“ a jsou popsány níže. Jedná se o částečný návrh funkcionality, který nesouvisí přímo s jazykem MBIQL a nebyl součástí implementace.

Zobrazování pomocí vazeb Tato metoda se spoléhá na nové typy Objektů a vazeb definované v novém datovém modelu. Uživatel by si mohl v rámci tohoto způsobu zobrazování zvolit, že chce ve výsledném grafu ponechat pouze uzly spojené jedním z typů vazeb (Instanční, Hierarchické, nebo Modelové).

Zobrazování pomocí jiných prvků Tato metoda spočívá v odstraňování typů Objektů z výsledného grafu. Konkrétně může uživatel zvolit, které Objekty nechce vidět v návratových hodnotách dotazu a z výsledného grafu jsou odstraněny. Lze se tak zaměřit na užší množinu uzlů.

Asistované procházení Poslední metodou je samotné procházení grafu. V tomto případě si uživatel zvolil ve výsledném grafu výchozí uzel. Následně by měl možnost si zvolit hranu, po které se z tohoto uzlu vydat k sousedícím

uzlům. Tento proces by uživatel opakoval pro nově přidané uzly. Neoznačené hrany a uzly, které nejsou napojené na zvolené uzly by byly odstraněny.

Uživatel by tento proces mohl v libovolné iteraci ukončit.

Rozdíly v implementaci Na rozdíl od první dvou typů dotazování nebyly tyto rozšiřující metody plně implementovány. Byl realizován pouze jednoduchý prototyp, který umožňuje ve výsledném grafu zobrazit všechny další (nezobrazené) související Objekty zvoleného uzlu.

2.3.5 Shrnutí stavu

Pro dotazování přímo pomocí MBIQL byly identifikovány následující výhody a nedostatky.

2.3.5.1 Výhody

Jednoznačnou výhodou implementace MBIQL je vizualizace vztahů mezi Objekty báze MBI pomocí grafu namísto matice. Graf umožňuje přehledně zobrazit relevantní vztahy bez redundancí a omezit tak množství nadbytečných informací poskytovaných uživateli.

Za pozitivní aspekt lze považovat i grafické rozhraní pro dotazování. Jedná se o značné zlepšení pro netechnicky orientované uživatele v porovnání s nutností používat jazyk Cypher v prvotním návrhu vizualizace MBI [1].

2.3.5.2 Nedostatky

Dotazovací rozhraní lze zároveň považovat i za nedostatek jazyka MBIQL. Ačkoliv je grafické rozhraní lepší alternativou k přímému dotazování v jazyku Cypher, v případě *Specificky orientovaného dotazování* je poskytována podobná funkcionalita jako postranní panel v Portálu MBI 2.1, avšak není možné provádět *Dotazování přes hierarchie* či *Dotazování přes vyhledávání*. *Dotazování přes hierarchie* je možné provádět skrze *Dotazování pomocí vazeb*, které je ale o poznání komplexnější v porovnání s dotazovacím rozhraním Portálu MBI.

Za další možný nedostatek lze považovat pouze velmi omezenou možnost interakce s výsledným grafem. výsledná vizualizace je tak poměrně statická a neumožňuje další interakci s modelem. Součástí návrhu jazyka MBIQL, ačkoliv ne přímo související, byly rozšiřující metody pro „Asistované procházení grafem“, jednalo se však pouze o nastínění částečné funkcionality a nebyly tak implementovány.

2.3.5.3 Zhodnocení

Na základě předchozí analýzy lze usoudit, že návrh a implementace MBIQL splnily hlavní cíl, kterým byl alternativní způsob vizualizace vztahů mezi Ob-

jekty namísto matice vztahů používané na Portálu MBI. Bylo toho však docíleno na úkor složitosti dotazování. Ačkoliv samotná vizualizace je výrazně přehlednější, zvýšila se komplexita dotazovacího procesu a rozhraní v porovnání s rozhraním Portálu MBI. Pro některé uživatele by tedy tento přístup stále nemusel být ideální.

Dále také implementace neumožňuje další interakci s výsledným grafem pro případné prozkoumávání vazeb mezi dotazovanými Objekty. V takovém případě je nutné zformovat další dotaz nad databází. Návrh jazyka MBIQL tuto funkcionalitu nastiňoval, ale pouze v omezené podobě.

2.4 Výsledek analýzy

V předešlých podkapitolách bylo uvedeno shrnutí analýzy uživatelských skupin báze MBI a byly zanalyzovány dosavadní způsoby dotazování a navigace. Z této analýzy lze vyvodit některé závěry:

- Samotné metody dotazování a zobrazování instancí Objektů na portálu MBI není nutné dále upravovat. Navzdory mírné nepřehlednosti se jedná o dobrý způsob jak prezentovat velké množství dokumentů, které portál obsahuje. Problém Portálu MBI spočívá ve vizualizaci vztahů mezi Objekty pomocí matic, které mohou být nepřehledné a nabývat velkých rozměrů.

- Jazyk MBIQL byl navržen jako alternativní způsob dotazování, který by přehledněji reprezentoval vztahy mezi Objekty pomocí grafové databáze. V tomto ohledu návrh uspěl, avšak na úkor složitosti dotazovacího rozhraní v porovnání s Portálem MBI. MBIQL ve výsledné podobě také neumožňuje přímou interakci s výsledným vizualizovaným grafem. Není tak možné dále prozkoumávat datový model bez zformování dalšího dotazu.

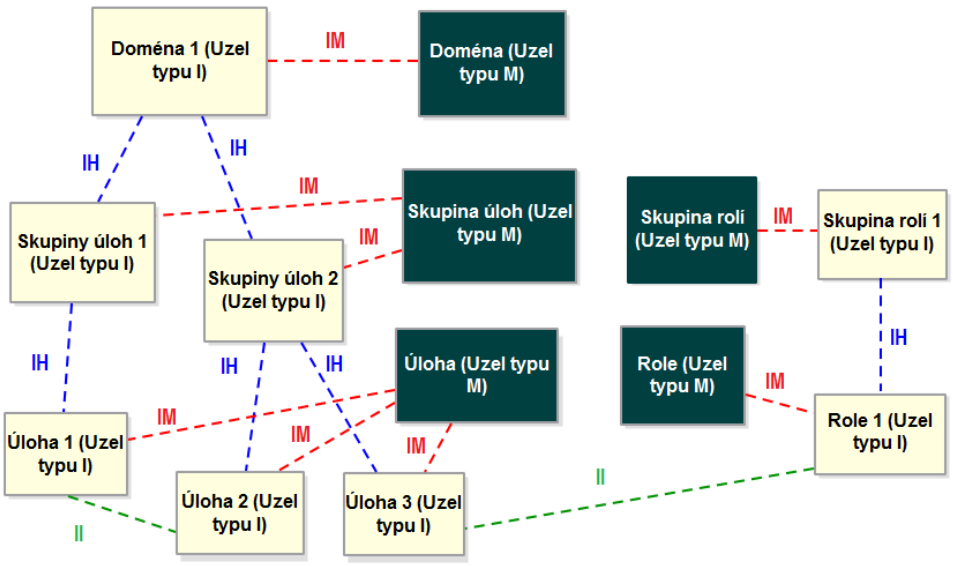
- Z analýzy uživatelů vyplývá, že navrhovaný způsob navigace by měl zohlednit potřebu dotazovat se nad samostatnými Objekty a zároveň umožnit navigovat vazby mezi Objekty datového modelu pro získání vhledu do struktury a kontextu modelu.

- Na základě předchozích závěrů se jako další krok, a cíl této práce, jeví navrhnout navigační jazyk v bázi MBI, který by přehledně prezentoval vztahy mezi Objekty, ale nevyžadoval komplexní, dotazovací, uživatelské rozhraní.

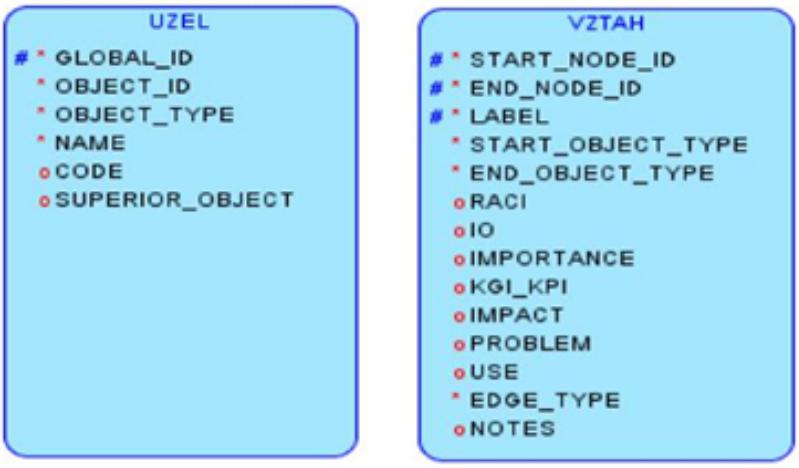
Kapitola 3

Schéma datového úložiště

Tato kapitola se zabývá strukturou databázového schématu pro import dat z MBI. Nejprve je popsána struktura Objektů schématu, kterými jsou *Uzel* (Objekty) a *Hrana* (Vazba), poté je schéma zhodnoceno z hlediska použitelnosti pro novou metodu navigace. Nakonec jsou zanalyzována data určená k importu do grafové databáze poskytnutá z MBI.



Obrázek 3.1: Nové schéma modelu MBI [2]



Obrázek 3.2: Základní entity datového úložiště [2]

3.1 Schéma Uzlu

Uzly grafu reprezentují Objekty modelu MBI. Pro Uzly jsou definovány následující atributy:

- **GLOBAL_ID**: Interní identifikátor uzlu. Je uměle generován pro účely grafové databáze a není součástí modelu MBI.
- **OBJECT_ID**: Identifikátor jednotlivých instancí Objektů MBI. Má následujícím tvar: [OBJECT_TYPE]-[číslo] (např.: TASK-101, ROLE-58).
- **OBJECT_TYPE**: Typ daného Objektu. Může nebývat jedné z 29 hodnot podle typů Objektů datového modelu (viz 1.1) a hodnoty MODEL pro modelové Objekty (např.: TASK, ROLE, TASKSGROUP).
- **NAME** - Název konkrétní instance Objektu (např.: „Pocty konfiguračních položek“).
- **CODE** - Pětimístná zkratka označující daný uzel (např.: UQ153A, TGQ400, DO700).
- **SUPERIOR_OBJECT** - Referencuje nadřazený Objekt v hierarchii modelu pomocí jeho OBJECT_ID (např.: ROLEGROUP-3 pro instanci ROLE-13).

3.2 Schéma Hrany

Hrany představují vztahy mezi Objekty modelu MBI. Pro Hrany jsou definovány atributy, které poskytují kontext pro vztahy souvisejících Objektů:

- **START_NODE_ID** : Identifikátor startovního uzlu jeho (OBJECT_ID).
- **END_NODE_ID** : Identifikátor koncového uzlu (jeho OBJECT_ID).
- **LABEL**: Popis hrany ve formátu [START_OBJECT_TYPE]_[END_OBJECT_TYPE].
- **START_OBJECT_TYPE**: Typ Objektu startovního uzlu (např.: METRIC)
- **END_OBJECT_TYPE**: Typ Objektu koncového uzlu (např.: ROLE)
- **RACI**: Typ vztahu mezi Objekty Role a Úloha, který Roli přiřazuje zodpovědnosti a kompetence v rámci vykonávání Úlohy. Vychází z principu matice RACI[2].
- **IO**: Typ vztahu mezi Objekty Dokument a Úloha, který určuje zda je Dokument pro danou Úlohu vstupní (**I**nput), výstupní (**O**utput), nebo je v rámci Úlohy pouze aktualizován (**U**pdate).
- **IMPORTANCE**: Typ vztahu který se vyskytuje mezi více typy Objektů. Vyjadřuje míru důležitosti na množině 1,2,3, vyšší znamená důležitější.
- **KGI_KPI**: Typ vztahu mezi Objekty Metrika a Úloha, který určuje význam Metriky pro Úlohu. KGI[3] Metrika indikuje zda Úloha dosahuje požadovaných cílů. KPI[4] Metrika představuje indikátor výkonnosti dané úlohy.
- **IMPACT**: Typ vztahu mezi Objekty Faktor a Úloha, který určuje míru dopadu určitého Faktoru na Úlohu. Míra je opět na množině 1,2,3, vyšší znamená vyšší dopad.
- **PROBLEM**: Typ vztahu mezi Objekty Scénář a Úloha, který specifikuje dílčí problémy a otázky Scénáře, které daná Úloha řeší. Problémy jsou poznamenány čísly, pokud Úloha řeší celý scénář, je použit znak „*“.
- **USE**: Typ vztahu mezi Objekty Metrika a Dimenze, který určuje četnost využití Dimenze při práci s danou Metrikou. Míra je opět na množině 1,2,3, vyšší znamená vyšší četnost.

[2] Responsible-Accountable-Consulted-Informed
[3] Key Goal Indicator
[4] Key Performance Indicator

- **EDGE_TYPE**: Nový atribut přidán v rámci změny datového modelu při návrhu MBIQL. Určuje typ Hrany z hlediska struktury modelu:
  - **Instance-Instance (II)**: Představuje Hrany mezi instancemi Objektů na stejné úrovni hierarchie modelu (např.: Úloha-Metrika)
  - **Instance-Hierarchy (IH)**: Představuje Hrany mezi instancemi Objektů na různé úrovni hierarchie modelu (např.: Úloha-Skupina úloh)
  - **Instance-Model (IM)**: Představuje Hrany mezi instancemi Objektů na libovolné úrovni hierarchie modelu a příslušným modelovým Objektem MBI.
- **NOTES**: Obsahuje textové poznámky o vztahu obalené do tagů jazyka HTML.

Popisy atributů byly čerpány z [5] a [2].

3.3 Zhodnocení modelu

Databázový model v jeho současné podobě je plně postačující pro novou metodu navigace v bázi MBI. Pro účely této práce tedy není nutné model rozšiřovat. Naopak, některé prvky modelu se jeví jako redundantní či zbytečné jak pro nový způsob navigace, tak pro transformaci dat z báze MBI do grafové databáze jako takové. Tyto prvky jsou v této části specifikovány.

3.3.1 Atributy Uzlu

V této sekci jsou zmíněny atributy Uzlů, které jsou potenciálně zbytečné či redundantní pro novou metodu navigace či samotnou tvorbu databázového schématu MBI v grafové databázi.

- **GLOBAL_ID**: Dodatečný identifikátor Uzlu není potřeba, Neo4j si již vytváří interní identifikátor pro každý vytvořený uzel. Atributy OBJECT_ID a CODE jsou navíc již v rámci Objektů MBI unikátní mohou sloužit jako možné identifikátory uzlů.
- **SUPERIOR_OBJECT** - Tento atribut není u Uzlů v grafové databázi nutný, jelikož jeho funkci již zastupují hierarchické Hrany typu IH.

3.3.2 Atributy Hrany

V této sekci jsou zmíněny atributy Hran, které jsou potenciálně zbytečné či redundantní pro novou metodu navigace či samotnout tvorbu databázového schématu MBI v grafové databázi.

- **EDGE_TYPE**: Typ hrany není nutné explicitně udávat, jelikož přirozeně vyplyne z atributů START_OBJECT_ID, END_OBJECT_ID, LABEL, START_OBJECT_TYPE a END_OBJECT_TYPE. Možné využití se nabízí v návrhu jazyka MBIQL v rámci *Asistovaného procházení grafem*, pokud si uživatel přeje zobrazit pouze hrany určitého typu. Tato funkcionalita však nebyla implementována.

3.4 Formát dat k importu

Data z báze MBI byla již v prvotním řešení transformována z formátu XML do CSV a uložena do databáze Neo4j [1]. V rámci návrhu a implementace jazyka MBIQL byla data poskytnuta již ve formátu CSV, nicméně bylo nutné vytvořit nové schéma datového úložiště z důvody změny samotného datového modelu MBI (viz obrázek 3.1) a pro potřeby jazyka MBIQL [2].

Pro účely této práce byla data z MBI rovněž poskytnuta ve formátu CSV a to ve třech souborech, jeden pro Uzly, druhý pro Hrany, třetí, který mapuje atribut OBJECT_TYPE Uzlů na jejich slovní přepis (např.: TASK -> Úloha).

Soubor pro Uzly obsahoval veškeré instanční Uzly, neobsahoval Uzly Modelové.

Soubor pro Hrany obsahoval všechny Hrany typu II, neobsahoval Hrany typu IH a IM, ty bylo nutné doplnit pomocí atributu SUPERIOR_OBJECT u Uzlů.

Kapitola 4

Návrh navigačního jazyka

Cíl této kapitoly je navrhnout grafický, doménově specifický navigační jazyk, který umožní procházet grafovou databází MBI. V rámci této kapitoly je jazyk definován. Poté jsou definovány vzorové dotazy pro demonstraci principu jazyka. Dále jsou popsány grafické prvky, které jazyk používá. Následně je popsán samotný princip a průběh navigace pomocí jazyka. Nakonec je průběh navigace demonstrován na vzorových dotazech.

4.1 Definice jazyka

MBINL (MBI Navigation Language) je doménově specifický, grafický navigační jazyk, určený k navigaci ve struktuře Objektů báze MBI. Jazyk zprostředkovává navigaci pomocí interaktivní konstrukce dotazovacího grafu. Tento graf je grafickou reprezentací Objektů MBI z grafové databáze. Reprezentace má podobu Uzlů představujících Objekty a Hran představujících Vztahy mezi Objekty. Při interakci s reprezentací je dotazována databáze MBI a reprezentace aktualizována dle získaných dat.

4.2 Vzorové dotazy

Následující vzorové dotazy budou použity k demonstraci principu navigace. Vzorové dotazy byly převzaty z návrhu jazyka MBIQL [2] aby bylo možné lépe porovnat rozdíly mezi oběma návrhy. Dva vzorové dotazy (3. a 4.) byly modifikovány pro demonstraci funkcionality, kterou návrh jazyka MBIQL neobsahuje.

1. *Zobraz instanci objektu Úloha, která má název: „Správa infrastruktury“.*

2. *Zobraz všechny Metody, které mají vztah k Úloze: „Propojení metrik byznysu a metrik IT“.*

3. *Zobraz Skupiny úloh, které nespadají do Domény: „Strategické řízení IT“.* (modifikován)

4. *Najdi pouze Faktory, které nejsou využity u žádného Scénáře.* (modifikován)

5. *Zjisti, do jaké Skupiny metrik patří Metrika: „Počty spravovaných technických prostředku“.*

6. *Zobraz kompletní hierarchii dvou Rolí, první s názvem: „Návrhář databází“, druhou s kódem: „R108“. K nim zobraz také všechny Úlohy, se kterými jsou vázány.*

4.3 Grafické prvky jazyka

Navigace či dotazování za pomoci grafických prvků dokáže poskytnout intuitivnější uživatelské rozhraní než formování dotazů pomocí textově orientovaných jazyků. Podobně jako jazyk MBIQL jsou proto grafické prvky součástí i tohoto návrhu. Aby se však omezilo navýšení komplexnosti rozhraní, grafické prvky jsou použity tak aby se minimalizovalo množství různých použitých grafických prvků.

Jako hlavní grafické prvky jsou použity následující:

- **Uzel**: Představuje Uzel v dotazovacím grafu a reprezentuje Objekt z modelu MBI. Uzel může reprezentovat typ Objektu, nebo specifickou instanci či množinu instancí typu Objektu. Během konstrukce dotazovacího grafu jsou k dispozici všechny Objekty modelu MBI 1.1.

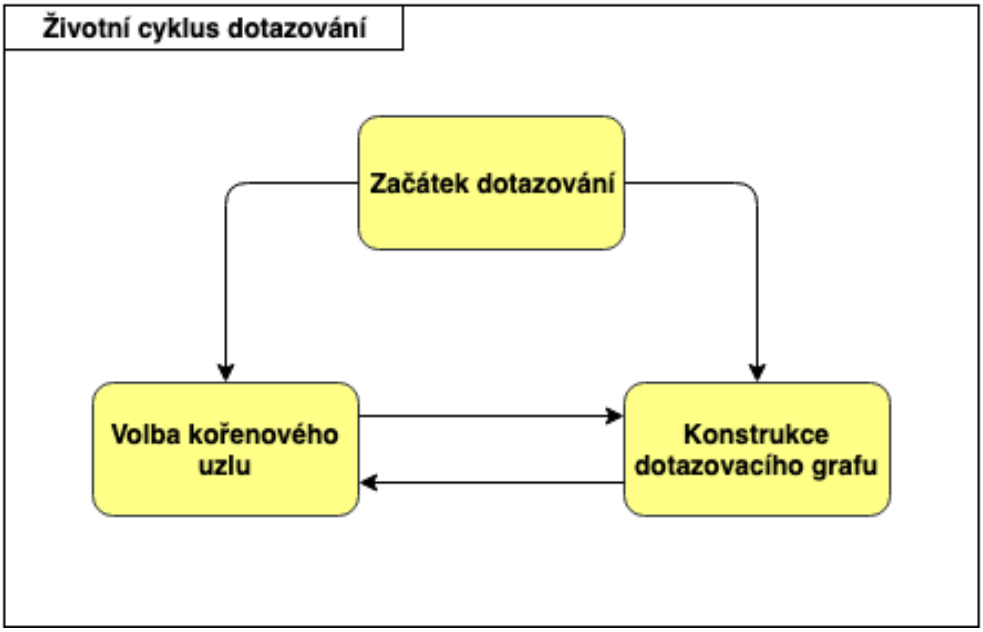
- **Hrana**: Představuje Hranu v dotazovacím grafu a reprezentuje Vazbu mezi dvěma Objekty z modelu MBI. Hrana může reprezentovat Vazbu mezi typy Objektů, nebo přiřazenými instancemi typů Objektů.

- **Plátno**: Obsahuje interaktivní, dotazovací graf, který je grafickou reprezentaci Objektů MBI.

Jako podpůrné grafické prvky jsou použity následující:

- **Seznam**: Je použit k volbě kořenového uzlu a k přiřazování konkrétních instancí Objektů k danému Uzlu.

- **Textové pole**: Je použito k vyhledávání konkrétních instancí Objektů k danému Uzlu. Slouží tedy jako alternativa pro Seznam.



Obrázek 4.1: Životní cyklus dotazování [2]

4.4 Navigace pomocí jazyka

Způsob, kterým jazyk MBINL pracuje s databází má za účel být méně statický než klasický způsob dotazování. Nejedná se tedy o jednorázový proces, kde uživatel vytvoří dotaz, a ten je následně vykonán a výsledky zobrazeny. Během interakce s dotazovacím grafem jsou vykonávány odpovídající, dílčí dotazy. Pomocí těchto dotazů je dotazovací graf aktualizován.

4.4.1 Průběh navigace

Navigace pomocí jazyka je rozdělena na dvě části:

- **Volba kořenového uzlu**: Kořenový uzel představuje Objekt MBI, o kterém chce uživatel získat informace. Jako výchozí kořenový uzel je použit Objekt typu *Úloha*.

- **Konstrukce dotazovacího grafu**: Po zvolení kořenového uzlu lze začít konstruovat dotazovací graf. To lze provádět přídáváním a odebíráním souvisejících Uzlů a jejich specifikací či negací.

Grafické znázornění průběhu navigace lze vidět na obrázku 4.1.

4.4.2 Průběh volby kořenového uzlu

Volba kořenového uzlu slouží pro specifikaci Objektu MBI, o kterém chce uživatel získávat další informace a představuje centrální Uzel celého dotazovacího grafu. Kořenový uzel může být změněn kdykoliv během konstrukce dotazovacího grafu. V takovém případě je dotazovací graf resetován do výchozího stavu, kdy obsahuje pouze zvolený kořenový uzel.

Pro zvolení kořenového uzlu je nutné vybrat typ Objektu ze Seznamu Objektů modelu MBI. Výchozí kořenový uzel je Objekt typu *Úloha*. Tento Objekt byl zvolen díky jeho centrální roli v datovém modelu MBI 1.1.

4.4.3 Průběh konstrukce dotazovacího grafu

Když je zvolen kořenový uzel, je možné začít konstruovat dotazovací graf. Konstrukce je založena na interakci přímo s dotazovacím grafem, který uživatel konstruuje a vede k postupné specifikaci instancí, kterých může nabývat Objekt kořenového uzlu. Interakce je prováděna pomocí několika dostupných operací, kterými může uživatel graf měnit. Tyto akce na sobě nejsou přímo závislé a nemají pevně danou posloupnost vykonávání:

- Přidání Uzlu
- Odebrání Uzlu
- Specifikace Uzlu
- Negace Uzlu

4.4.3.1 Přidání Uzlu

První z akcí, které umožňují interakci a dotazovacím grafem je přidání Uzlu do grafu. Jelikož Uzly představují Objekty MBI, při každé interakci je možné přidat pouze takové Uzly, které respektují Vztahy v modelu MBI. Lze tedy přidat pouze Uzly, které splňují jednu z následujících podmínek:

- Přidávaný Uzel má Vztah s kořenovým uzlem v modelu MBI.
- Přidávaný Uzel má Vztah s již přidaným Uzlem v modelu MBI.

Pro přidání Uzlu je uživatel zvolí již existující uzel na Plátně a poté určí, který související Uzel má být do grafu přidán. Nově přidaný Uzel je poté se zvoleným Uzlem propojen Hranou, která reprezentuje jejich Vztah v modelu MBI.

4.4.3.2 Odebrání Uzlu

Další operací umožňující interakci s dotazovacím grafem je odebrání Uzlu z grafu. Během této operace hraje roli existence kořenového uzlu. V případě „krajního Uzlu“, tedy takového, který má v grafu pouze jeden sousední Uzel, je z grafu odstraněn pouze tento Uzel. V případě „vnitřního Uzlu“, tedy takového, který má v grafu dva nebo více sousedních Uzlů by současný graf byl rozdělen na dva nebo více nesouvislých grafů. Bylo by tak nutné zvolit,

který z výsledných grafů bude dále použit a který odstraněn. Jelikož je zvolen kořenový uzel, který je předmětem zájmu, jsou vždy odstraněny všechny větve grafu, které z odstraňovaného Uzlu vedou směrem od kořenového uzlu, společně se samotným Uzlem určeným k odebrání.

Pro odebrání Uzlů je nutné zvolit existující uzel na Plátně a určit, že má být odstraněn

4.4.3.3 Specifikace Uzlu

Další operace, která umožňuje interakci s dotazovacím grafem je specifikace Uzlu v grafu. Specifikace uzlu představuje přiřazení specifické instance či instancí Objektu z modelu MBI k danému Uzlu. Pro každý Uzel je tak možné specifikovat množinu instancí, kterých může nabývat a omezit tak množinu potenciálních instancí, kterých mohou nabývat související Uzly. V případě specifikace více než jedné instance jsou využity logické operátory AND a OR. Tyto operátory umožňují specifikovat v jakém Vztahu jsou specifikované instance Objektů vzhledem k souvisejícím Uzlům. Je tedy možné vyjádřit, že související Uzly musí mít Vztah s každou specifikovanou instancí, nebo pouze s některými.

Pro specifikaci Uzlu je nutné nejprve zvolit existující uzel na Plátně. Poté je nutné specifikovat instanci Objektu, buď výběrem ze Seznamu odpovídajících instancí, nebo vyhledáním přes Textové pole. V případě přidání více instancí je také nutné specifikovat v jakém logickém vztahu mají být přiřazené instance (AND nebo OR).

4.4.3.4 Negace Uzlu

Poslední akcí umožňující interakci s dotazovacím grafem je negace Uzlu v grafu. Negace umožňuje vyjádřit, že daný Uzel nesmí mít Vztah se souvisejícími Uzly. Pokud Uzel není Specifikován (nemá přiřazené instance), negace vyjadřuje, že související Uzly mohou nabývat pouze takových instancí Objektů, které pro daný Uzel nemají definovaný vztah. Pokud je uzel Specifikován (má přiřazenu jednu či více instancí), negace vyjadřuje, že související Uzly mohou nabývat pouze takových instancí Objektů, které nemají definovaný vztah s přiřazenými instancemi daného Uzlu.

4.5 Ukázka vzorových dotazů

V této části je na vzorových dotazech ze sekce 4.2 demonstrován průběh navigace pomocí jazyka MBINL.

1. *Zobraz instanci Objektu Úloha, která má název: „Analýza datových zdrojů“.*

Jako výchozí kořenový uzel je již zvolen Objekt Úloha, není tedy nutné provádět *Volbu kořenového uzlu*. Pomocí *Specifikace uzlu* je kořenovému uzlu přiřazena instance s názvem „Analýza datových zdrojů“.

2. *Zobraz všechny Metody, které mají vztah k Úloze: „Analýzy výnosů z IT služeb“.*

   Nejprve je jako kořenový uzel zvolen Objekt Metoda. Pomocí *Přidání Uzlu* je pro kořenový uzel zvolen související Objekt Úloha a odpovídající Uzel je přidán do grafu. Pomocí *Specifikace Uzlu* je Uzlu Úloha přiřazena instance s názvem „Analýzy výnosů z IT služeb“.

3. *Zobraz Skupiny úloh, které nespadají do Domény: „Strategické řízení IT“.*

   Jedná se o modifikovaný dotaz. Nejprve je jako kořenový uzel zvolen Objekt Skupina úloh. Pomocí *Přidání Uzlu* je pro kořenový uzel zvolen související Objekt Doména a odpovídající Uzel je přidán do grafu. Pomocí *Specifikace Uzlu* je Uzlu Doména přiřazena instance s názvem „Strategické řízení IT“. Pomocí *Negace Uzlu* je Specifikovaný Uzel Doména (přiřazena instance) označen logickým operátorem NOT.

4. *Najdi pouze Faktory, které nejsou využity u žádného Scénáře.*

   Jedná se o modifikovaný dotaz. Nejprve je jako kořenový uzel zvolen Objekt Faktor. Pomocí *Přidání Uzlu* je pro kořenový uzel zvolen související Objekt Úloha a odpovídající Uzel je přidán do grafu. Opět pomocí *Přidání Uzlu* je pro Uzel Úloha zvolen související Objekt Scénář a odpovídající Uzel je opět přidán do grafu. Pomocí *Negace Uzlu* je Nespecifikovaný Uzel Scénář (bez přiřazené instance) označen logickým operátorem NOT.

5. *Zjisti, do jaké Skupiny metrik patří Metrika: „Finanční náklady na bezpečnost IT služeb“.*

   Nejprve je jako kořenový uzel zvolen Objekt Metrika. Pomocí *Přidání Uzlu* je pro kořenový uzel zvolen související Objekt Podskupina Metrik a odpovídající Uzel je přidán do grafu. Obdobně je pro Uzel Podskupina Metrik zvolen Objekt Skupina Metrik a odpovídající Uzel je opět přidán do grafu. Nakonec je pomocí *Specifikace Uzlu* k Uzlu Skupina Metrik přiřazena instance s názvem „Finanční náklady na bezpečnost IT služeb“.

6. *Zobraz kompletní hierarchii dvou Rolí, první s názvem: „Návrhář databází“, druhou s kódem: „R108“. K nim zobraz také všechny Úlohy, se kterými jsou vázány.*

   U tohoto dotazu by teoreticky bylo možné dotaz začít konstruovat z několika potenciálních kořenových uzlů. Jako nejužitečnější se však jeví

jako kořenový uzel ponechat Objekt Úloha, protože všechny ostatní Uzly budou reprezentovat specifické instance. *Volbu kořenového uzlu* lze tedy vynechat.

Pomocí *Přidání Uzlu* je pro kořenový uzel zvolen související Objekt Role a odpovídající Uzel je přidán do grafu. Opět pomocí *Přidání Uzlu* je pro Uzel Role zvolen související Objekt Skupina rolí a odpovídající Uzel je opět přidán do grafu. Pomocí *Specifikace Uzlu* je Uzlu Role přiřazena instance s názvem „Návrhář databází“. Dále je pomocí *Specifikace Uzlu* k Uzlu Role přiřazena instance s kódem „R108“ a specifikován logický vztah OR.

Kapitola 5

Srovnání s jazykem MBIQL

V této kapitole je nový navigační jazyk MBINL, definovaný v předchozí kapitole 4, srovnán s dotazovacím jazykem MBIQL, popsaným v sekci 2.3. V první části jsou popsány shodné či podobné aspekty obou jazyků. Ve druhé části jsou popsány hlavní rozdíly mezi oběma jazyky.

5.1 Shodné aspekty

- Oba jazyky využívají grafické prvky.
- Cílovou technologií obou jazyků je grafová databáze Neo4j.
- Oba jazyky na pozadí využívají dotazovací jazyk Cypher pro komunikaci s grafovou databází Neo4j.
- Podobně jako jazyk MBIQL byl nový jazyk MBINL navržen aby poskytl uživatelům intuitivnější způsob dotazování či navigace v bázi MBI.
- U obou jazyků se stále nabízí možnost budoucího rozšíření funkcionality.

5.2 Rozdílné aspekty

- Průběh dotazování obou jazyků je odlišný. Jazyk MBIQL má přesně definované dvě metody dotazování (Specificky orientované dotazování a Dotazování pomocí vazeb), které mají určitou posloupnost operací. Jazyk MBINL pouze vyžaduje zvolení kořenového uzlu grafu, poté má definovanou množinu operací, které jsou na sobě relativně nezávislé a samotný průběh dotazování je tak velmi volně definovaný.
- Oba jazyky využívají grafické prvky různým způsobem. Jazyk MBIQL má v rámci struktury dotazování definované jaké grafické prvky jsou použity v konkrétním kroku tvorby dotazu. Také se jedná zejména o prvky

využívané ve standardních uživatelských rozhraních a formulářích. V případě jazyka MBINL je hlavním grafickým prvkem Plátno, na kterém se nachází interaktivní dotazovací graf. Formování dotazu má tedy blíže k „sandboxu“ bez pevně dané struktury dotazování.

- Jazyk MBIQL je navržen nad schématem specifického datového úložiště. Dotazy v něm vzniklé jsou transformovány do dotazů v jazyce Cypher, které jsou použitelné pouze pro toto schéma. Pro jazyk MBINL toto omezení do určité míry platí také. Díky volněji definované struktuře dotazování je však obecnější a bylo by tak možné jej přizpůsobit na libovolné schéma.

Kapitola 6

Návrh implementace

Tato kapitola se zabývá návrhem implementace pro dotazovací nástroj jazyka MBINL. Nejprve jsou uvedeny modely, které popisují základní vlastnosti dotazovacího nástroje z hlediska funkcionality. Dále je definován koncept pro dotazovací nástroj. Nakonec jsou uvedeny technologické požadavky na implementaci nástroje.

6.1 Modely dotazovacího nástroje

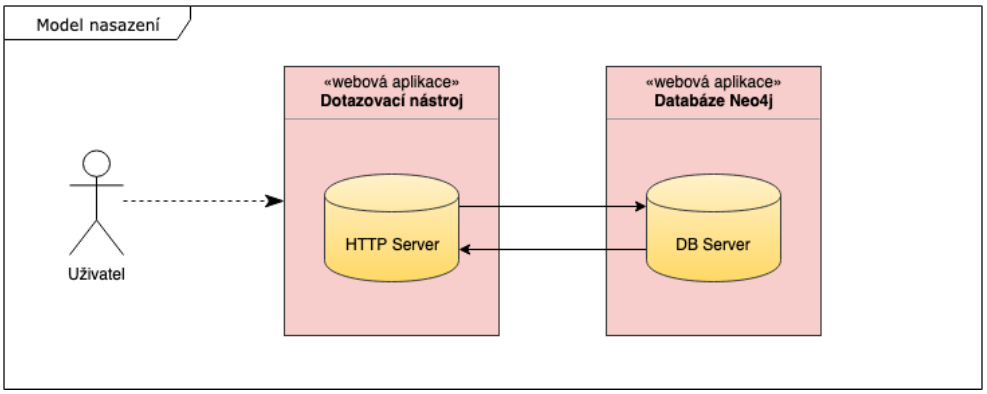
Základní vlastnosti dotazovacího nástroje jsou popsány dvěma modely. První model ukazuje jak by nástroj měl fungovat po nasazení na server. Druhý model specifikuje případy použití nástroje pro uživatele.

6.1.1 Model nasazení

Pro realizaci jazyka MBINL je nutné vyvinout softwarovou aplikaci, která bude sloužit jako dotazovací nástroj. Tato aplikace by po dokončení implementace měla být nasazena na HTTP server, ze kterého bude přístupná pro uživatele. Tento server by měl komunikovat s databázovým (DB) serverem, na kterém bude nasazena grafová databáze Neo4j. Získávání dat z databáze bude zprostředkováno pomocí jazyka Cypher. HTTP server tedy bude odesílat uživatelské dotazy na DB server, který HTTP serveru zpět odešla požadovaná data. Model nasazení lze vidět na obrázku 6.1.

6.1.2 Model použití

Účelem dotazovacího nástroje je získávat informace o Objektech MBI pomocí jazyka MBINL. Je tedy nutné aby dokázal nástroj vytvářet dotazovací graf. Dále by bylo užitečné, aby byl uživatel schopen spravovat vytvořené grafy. Nakonec by uživatel měl být schopen zobrazit si výsledky dotazu. Budou tedy dostupné následující možnosti práce s nástrojem:



Obrázek 6.1: Model nasazení

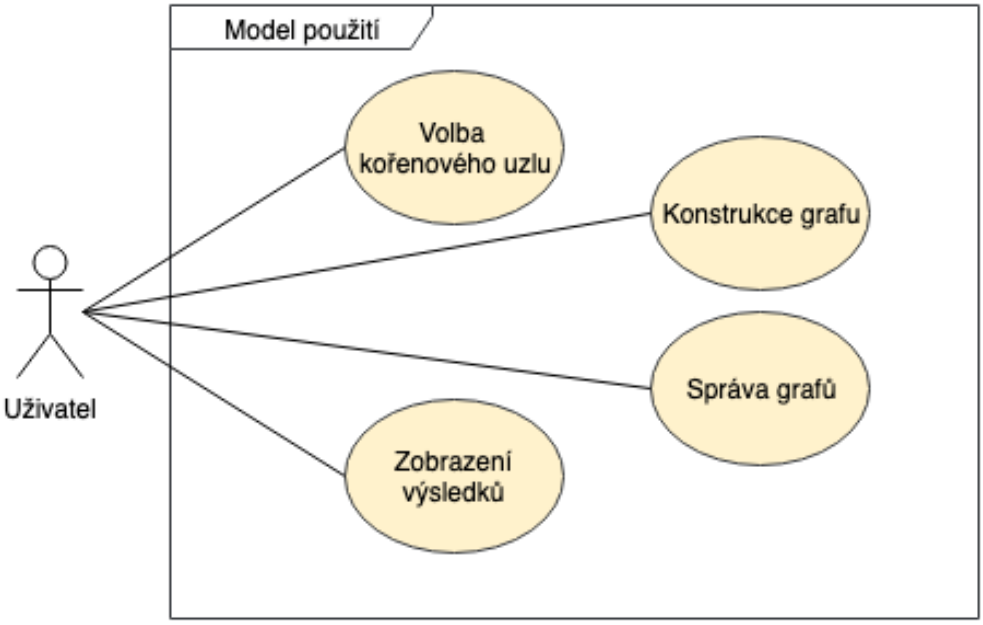
- **Volba kořenového uzlu**: Dle návrhu jazyka MBINL umožní nástroj uživateli zvolit kořenový uzel, který bude použit jako cíl dotazování.
- **Konstrukce dotazovacího grafu**: Dle návrhu jazyka MBINL umožní nástroj uživateli konstruovat dotazovací graf pomocí definovaných operací: *Přidání Uzlu, Odebrání Uzlu, Specifikace Uzlu, Negace Uzlu.*
- **Správa grafů**: Nástroj umožní uživateli ukládat zkonstruované grafy a opět nahrávat takto uložené grafy. Uživatel tak nebude muset pokaždé znovu konstruovat graf pro každý dotaz.
- **Zobrazení výsledků**: Kromě samotného dotazovacího grafu bude nástroj poskytovat možnost zobrazit výsledky odpovídající tomuto grafu. V rámci jazyka MBINL to znamená zobrazení množiny instancí kořenového uzlu, které odpovídají právě zkonstruovanému dotazovacímu grafu.

Model použití je znázorněn na obrázku 6.2.

6.2 Koncept dotazovacího nástroje

Před začátkem implementace je důležité zformovat koncept vyvíjené aplikace, který se odvíjí od modelu použití z předchozí sekce 6.2. Na základě tohoto modelu by dotazovací nástroj měl být rozdělen na čtyři hlavní části.

- **Volba kořenového uzlu**: Pro volbu kořenového uzlu bude sloužit grafický prvek *Seznam* v levé části aplikace. Seznam bude obsahovat seznam Objektů MBI zobrazených hierarchicky dle modelu MBI 1.1.
- **Konstrukce dotazovacího grafu**: Pro konstrukci dotazovacího grafu bude sloužit grafický prvek *Plátno* a bude zaujímat většinu aplikace. V tomto prvku bude zobrazen kořenový uzel jako základ dotazovacího



Obrázek 6.2: Model použití

grafu a následně veškeré Uzly a Hrany konstruovaného grafu. Operace *Specifikace Uzlu* bude využívat prvek *Seznam*, který se bude kontextově zobrazovat v pravé části *Plátna*. V horní části *Seznamu* se bude nacházet *Textové pole* pro vyhledávání v seznamu instancí.

- **Správa grafů**: Pro správu grafů bude opět sloužit grafický prvek *Seznam*, který se bude nacházet v pravé části aplikace. Uložené grafy budou zobrazeny zde a uživatel je bude moct nahrát na plátno. Tato akce přepíše jakýkoliv právě konstruoavný graf uloženým grafem.

- **Zobrazení výsledků**: Kromě samotného dotazovacího grafu bude možné zobrazit množinu instancí, kterých může nabývat kořenový uzel, jakožto zamýšlený cíl konstrukce grafu. Pro tento účel bude použit grafický prvek *Seznam*, který bude kromě názvu instancí zobrazovat i jejich další atributy. Tento prvek se bude nacházet ve spodní části aplikace pod *Plátnem*.

6.3 Technologické požadavky na implementaci

V této části jsou uvedeny požadavky na implementaci dotazovacího nástroje, které vyplynuly předchozích modelů aplikace a analýzy dotazování nad bází MBI:

- Práce s interaktivními grafickými prvky, zejména prvky *Uzel* a *Hrana*
- Funkcionalita pro ukládání a nahrávání dotazovacích grafů
- Podpora protokolu HTTP
- Přihlašování uživatelů
- Modul pro komunikaci s databází Neo4j

Pro implementaci front-end části aplikace by měly být použity následující jazyky:

- **HTML**: Slouží pro tvorbu základní struktury aplikace.
- **CSS**: Slouží pro tvorbu vizuální stránky aplikace.
- **JavaScript**: Slouží pro implementaci samotné funkcionality a zajištění responzivního chování aplikace.

Pro implementaci back-end části aplikace lze použít několik možných jazyků, které umožní splnit technologické požadavky na aplikaci. Některé z možných jazyků jsou:

- JavaScript (Node.js)
- PHP framework
- Java

Kapitola 7

Návrh prototypu

V této kapitole je popsán návrh prototypu dotazovacího nástroje, který poslouží k demonstraci principu navigace pomocí jazyka MBINL. Výsledek bude použit jako podklad pro realizaci.

Nejprve je popsán proces transformace a importu dat do grafové databáze Neo4j. Dále je specifikován návrh prototypu dle návrhu implementace jazyka MBINL 6. Podle potřeby jsou provedeny k revize a úpravy.

7.1 Transformace dat z MBI

Data z portálu MBI byla již v rámci předchozích řešení[2][3] transformována do současného schématu datového modelu 3.1 a uložena v grafové databázi Neo4j. Jak bylo zmíněno v sekci 3.3, současná podoba modelu MBI je pro jazyk MBINL dostačující a není potřeba jej rozšiřovat. Naopak, některé prvky modelu se jeví jako redundantní a v prototypu nebudou přímo použity. Jedná se o následující prvky:

- **Atributy Uzlu**: GLOBAL_ID, SUPERIOR_OBJECT
- **Atributy Hrany**: EDGE_TYPE

7.1.1 Import dat do databáze Neo4j

Prvotní způsob importu dat byl prováděn skrze program Batch importer [5], kde byly připraveny dva CSV soubory a ty naimportovány do databáze. V rámci implementace MBIQL byla databáze Neo4j vytvořena přímo v Javě a naplněna daty.

Program Batch Importer již řadu let není aktualizovaný a podporuje Neo4j nejvýše verze 2.0.0 (současná verse je 3.5.12). Jako způsob importu byl tedy

[5]Program je dostupný z: https://github.com/jexp/batch-import/tree/20

zvolen skript napsaný v jazyce Python s využitím ovladače neomodel [6]. Tento skript zajistí transformaci dat a naplnění databáze.

Před naplněním databáze je potřeba Uzly a Vztahy připravit. Vstupní data jsou poskytnuta od týmu MBI v podobě CSV souborů:

- **finalMeta**: Obsahuje mapování atributu OBJECT_TYPE u Uzlů a LABEL u Hran na formát v přirozeném jazyce.
- **finalNode**: Obsahuje všechny Objekty z báze MBI.
- **finalRelation**: Obsahuje vztahy mezi Objekty MBI.

V souboru *finalRelation* existují pouze Vztahy typu INSTANCE-INSTANCE, tudíž je nutné vytvořit hierarchické Vztahy (INSTANCE-HIERARCHY). Tyto Vztahy lze vytvořit pomocí atributu SUPERIOR_OBJECT v souboru *finalNode*.

Po těchto úpravách je možné vytvořit novou databázi a naplnit ji připravenými Uzly a Vztahy.

7.2 Prototyp

V této sekci je definována funkcionalita celého prototypu. Na detailním modelu použití 7.1 jsou specifikovány požadavky na prototyp a následně jsou popsány individuálně.

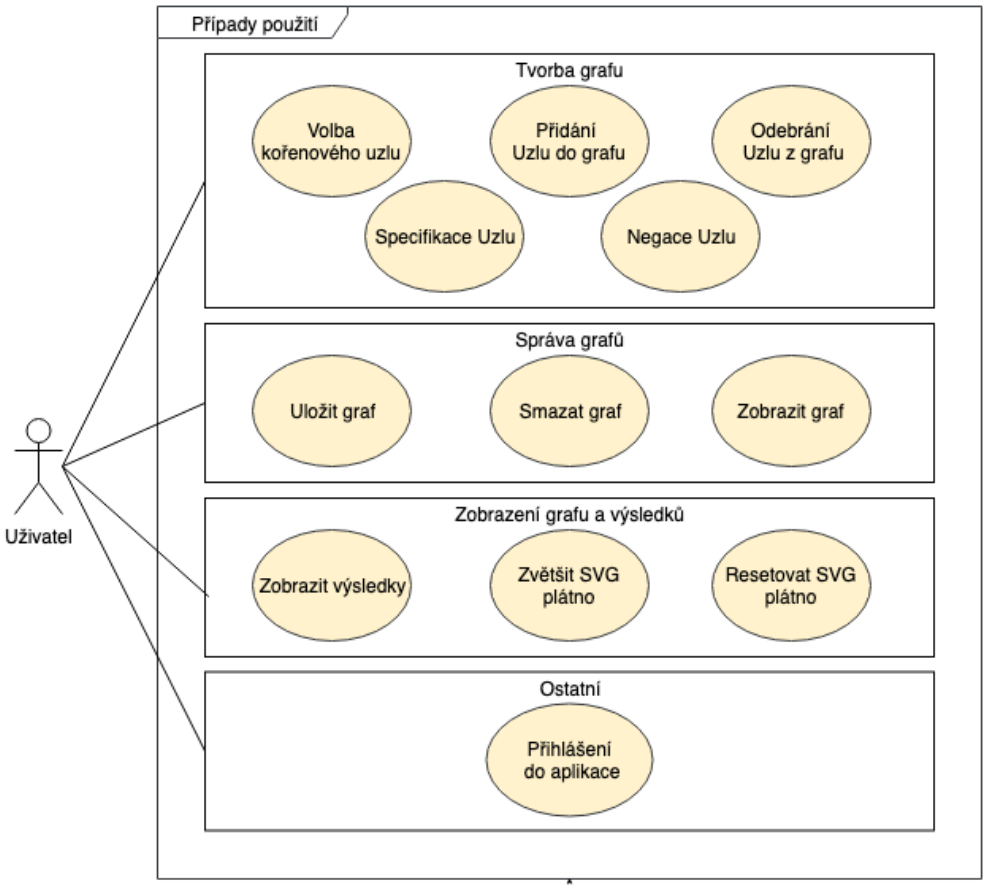
7.2.1 Tvorba grafu

V této sekci jsou uvedeny požadavky, které se týkají samotné konstrukce dotazovacího grafu, popsané v návrhu jazyka MBINL 4. Uživatel má k dispozici několik grafických prvků. Prvním je Seznam, který obsahuje všechny Objekty modelu MBI, které jsou hierarchicky uspořádané. Druhým prvkem je SVG Plátno, které obsahuje zvolený kořenový uzel a posléze konstruovaný dotazovací graf v podobě hlavních grafických prvků, Uzlů a Hran.

7.2.1.1 Volba kořenového Uzlu

Uživatel ze seznamu Objektů MBI zvolí požadovaný kořenový uzel. Zvolený uzel poté nahradí stávající kořenový uzel na Plátně. Pokud byl na plátně již zkonstruován nějaký graf, je přepsán. Výchozím kořenovým uzlem je Objekt Úloha.

[6] Dostupný z: https://github.com/neo4j-contrib/neomodel



Obrázek 7.1: Případy použití

7.2.1.2 Přidání Uzlu do grafu

Uživatel může po kliknutí na libovolný Uzel zvolit jeden ze souvisejících Objektů a přidá nový Uzel zvoleného Objektu. Mezi tyto dva uzly je automaticky přidána odpovídající Hrana.

7.2.1.3 Odebrání Uzlu z grafu

Uživatel může po kliknutí na libovolný Uzel (kromě kořenového uzlu) zvolit, že chce daný Uzel odstranit. Pokud se jedná o okrajový Uzel, je odebrán pouze daný Uzel. Pokud se jedná o vnitřní uzel, jsou současně s daným Uzlem odebrány veškeré související uzly směrem od kořene.

7.2.1.4 Specifikace Uzlu

Uživatel může po kliknutí na libovolný Uzel přiřadit Uzlu specifickou instanci odpovídajícího Objektu MBI. Pokud se uživatel rozhodne přidat více než jednu

instanci, musí dále specifikovat v jakém logickém vztahu (AND, nebo OR) se budou přiřazené instance nacházet vzhledem k souvisejícím Uzlům.

7.2.1.5 Negace Uzlu

Uživatel může po kliknutí na libovolný Uzel (kromě kořenového uzlu) zvolit, že má daný Uzel představovat negaci odpovídajícího Objektu MBI. Pokud Uzel nemá přiřazené atributy, související Uzly budou omezeny pouze na takové instance, které nemají s odpovídajícím Objektem MBI navázaný žádný vztah. Pokud už má přiřazené atributy, související Uzly budou omezeny pouze na takové instance, které nemají navázaný vztah s danými instancemi odpovídajícího Objektu MBI.

7.2.2 Správa grafů

V této sekci jsou uvedeny požadavky týkající se správy dotazovacích grafů v aplikaci. Uživatel má možnost ukládat zkonstruované dotazovací grafy a dále s nimi pracovat.

7.2.2.1 Uložit graf

Graf se uloží do databáze ve formátu JSON, který je použit k rekonstrukci grafu na SVG plátně. Každý uložený graf musí mít v rámci aplikace unikátní název.

7.2.2.2 Smazat graf

Smaže graf z celé aplikace.

7.2.2.3 Zobrazit graf

Uživatel může kliknutím na libovolný uložený graf jej zobrazit na SVG Plátně. touto akcí je přepsán jakýkoliv dotazovací graf, který byl předtím na SVG plátně konstruován.

7.2.3 Zobrazení grafu a výsledků

V této sekci jsou uvedeny požadavky, které se týkají se možností zobrazení dotazovacího grafu a jeho výsledků.

7.2.3.1 Zobrazit výsledky

Na základě zkonstruovaného dotazovacího grafu je v aplikaci zobrazen seznam instancí Objektů zvoleného kořenového Uzlu a jejich atributy. Momentálně jsou zobrazovány pouze atributy Název a Kód.

7.2.3.2 Zvětšit SVG plátno

Zvětší SVG Plátno pro konstrukci dotazovacích grafů na celou obrazovku.

7.2.3.3 Resetovat SVG plátno

Smaže aktuálně konstruovaný dotazovací graf a ponechá na plátně pouze kořenový uzel.

7.2.4 Ostatní

V této sekci jsou uvedeny požadavky týkající se ostatní funkcionality aplikace.

7.2.4.1 Přihlášení do aplikace

Na úvodní stránce aplikace umožní uživateli přihlásit se pod uživatelským jménem a heslem nastaveným při prvním spuštění uživatelského rozhraní Neo4j.

7.3 Architektura

Prototyp je pro jednoduchost navržen jako webová single-page aplikace, které typicky využívají server pouze jako úložiště dat [7]. Prototyp tedy nebude implementovat plnohodnotný back-end aplikace, který by byl nutný z hlediska zabezpečení.

Prototyp lze nasadit na webový server Apache, kde bude dostupný přes webový prohlížeč. K zobrazení dat uživatelům se bude využívat značkovací jazyk HTML. Pro vizuální stránku aplikace jsou použity kaskádové styly (CSS) a framework Materialize CSS.

Pro responzivní chování prototypu je použit JavaScript, konkrétně framework Angular (kapitola 8.1) a pro zobrazení výsledků dotazů JavaScriptová knihovna D3 (kapitola 8.2) a knihovna popoto.js (kapitola 8.3)

7.3.1 Vrstvy aplikace

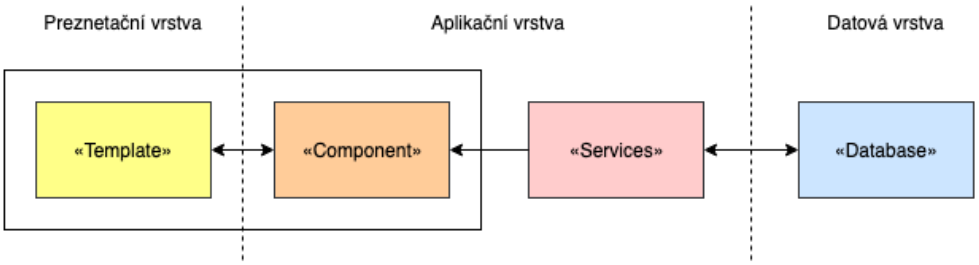
Struktura tříd aplikace tvoří dohromady tři vrstvy:

- **HTML Templates**: Šablony pro vizualizaci dat uživateli
- **Components**: Komponenty obsahují aplikační data a logiku
- **Services**: Jako komponenty,ale využívány na více místech aplikace
- **Databáze**: Slouží k ukládání a získávání perzitentních dat

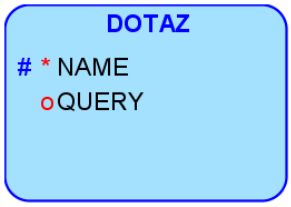
Více jsou jednotlivé komponenty popsány v kapitole 8.1).

7.3.1.1 Prezentační vrstva

Jedná se o vrstvu, která zobrazuje data uživateli a v rámci architektury je reprezentována HTML šablonou. K HTML šabloně také patří CSS soubor,



Obrázek 7.2: Architektura



Obrázek 7.3: Entita Dotaz

ve kterém jsou nadefinovány související styly, tedy způsob jakým budou data zobrazena (barva, velikost, ohraničení, umístění...).

7.3.1.2 Aplikační vrstva

Tato vrstva obsahuje samotnou logiku aplikace. Jsou do ní zařazeny Komponenty (Components) a Služby (Services) frameworku Angular. Komponenty komunikují s Prezentační vrstvou (předávají data) a pomocí Služeb i s Datovou vrstvou.

7.3.1.3 Datová vrstva

Datová vrstva slouží jako perzitentní úložiště dat pro aplikaci. Data ukládají do grafové databáze Neo4j a přistupuje se k nim pomocí Služeb Aplikační vrstvy.

7.3.2 Nové databázové entity

V kapitole 3 bylo revidováno datové schéma pro základní entity pro tvorbu grafu: Uzel (Node) a Hrana (Relationship). Jelikož prototyp bude podporovat ukládání grafů, je potřeba vytvořit další entitu typu Node pro uložení těchto dat. Datový model entity Dotaz lze vidět na obrázku 7.3.

7.3.2.1 Dotaz

Schéma Dotaz slouží pro ukládání jednotlivých grafů. Schéma obsahuje dva atributy:

- **NAME**: Obsahuje unikátní jméno grafu.
- **QUERY**: Obsahuje graf uložený ve formátu JSON, převedený do podoby řetězce.

Kapitola 8

Zvolené technologie

V této kapitole jsou posány technologie zvolené k implementaci prototypu dotazovacího nástroje a důvod k jejich volbě. Také jsou zde uvedeny alternativní technologie, které byly také zvažovány, ale nakonec nezvoleny.

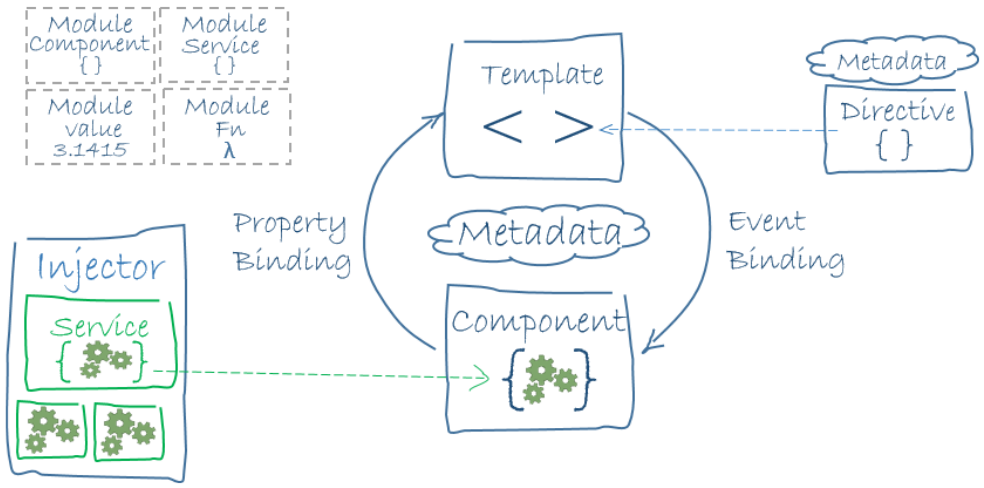
8.1 Angular

Angular je javascriptový framework určený pro tvorbu webových single-page aplikací [8]. Framework vznikl poprvé roce 2010 jako AngularJS a vytvořili jej Misko Hevery a Adam Abrons. Současná iterace, označovaná pouze jako Angular, nebo „Angular 2+“ [9] byla vytvořena v roce 2016 jako kompletní přepis celého frameworku. V současné jsou obě verze paralelně spravovány společností Google [10]. Pro implementaci prototypu byla zvolena iterace Angular 2+. Základní strukturu aplikace tvoří šablony (Templates). Jsou tvořeny HTML stránkami, rozšířené o nové syntaktické prvky jazyka Angularu. Šablony zobrazují uživateli data, která jsou zpracována pomocí Komponent (Components). Každá šablona má svoji vlastní komponentu, která se stará o logickou funkcionalitu dané části aplikace.

Konkrétními prvky jazyka Angularu se zabývají následující sekce, které vycházejí ze zdroje [11]. Souhrnné schéma celé architektury frameworku Angular lze vidět na obrázku 8.1.

8.1.1 Moduly

Angular definuje takzvané *NgModuly*. Pod modulem si lze představit kontejner, který obaluje ostatní prvky Angularu (šablony, komponenty, direktivy, ...), které spolu navzájem souvisí. Aplikace je rozdělena na moduly podle jednotlivých funkčních částí. Moduly mohou exportovat funkcionalitu pro užití ostatními moduly a naopak importovat exportovanou funkcionalitu jiných modulů. Každá Angular aplikace obsahuje alespoň jeden *root modul*, který se stará o spuštění celé aplikace.



Obrázek 8.1: Architektura Angular [11]

8.1.2 Komponenty

Komponenty (Components) představují základní stavební blok frameworku. Jedná se o programovou třídu či objekt, na kterou je navázaná šablona. Společně s šablonami komponenty dělí webovou stránku na samostatné celky, takzvané *Views*. Komponenta obsahuje datovou a logickou funkcionalitu asociovanou s navázanou šablonou.

8.1.3 Šablony

Šablony (Templates) jsou zodpovědné za zobrazování dat uživateli. Společně s komponentami šablony dělí webovou stránku na samostatné celky, takzvané *Views*. Šablony definují strukturu stránky pomocí HTML elementů rozšířených o direktivy frameworku.

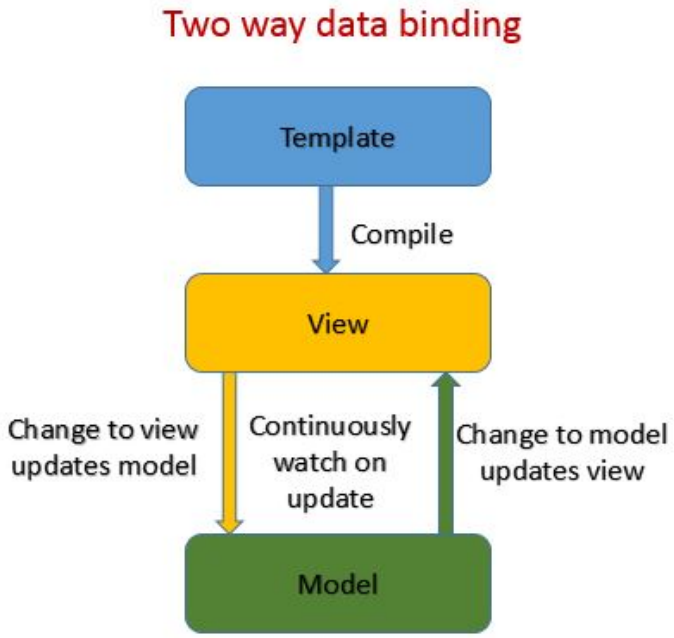
8.1.4 Direktivy

Direktivy (Directives) rozšiřují základní HTML elementy v šablonách a přidávají jim speciální chování definované frameworkem. Například zobrazení nebo skrytí elementu, provedení akce po kliknutí nebo nastavení určité CSS třídy na základě proměnné z Controlleru.

8.1.5 Two-way data-binding

Tento koncept propojení dat znamená, že když dojde ke změně dat v HTML DOM[7] šablony, změní se i související data samotné komponenty. A naopak

[7]Document Object Model - strom elementů vytvořený na základě HTML stránky



Obrázek 8.2: Two-way data-binding [12]

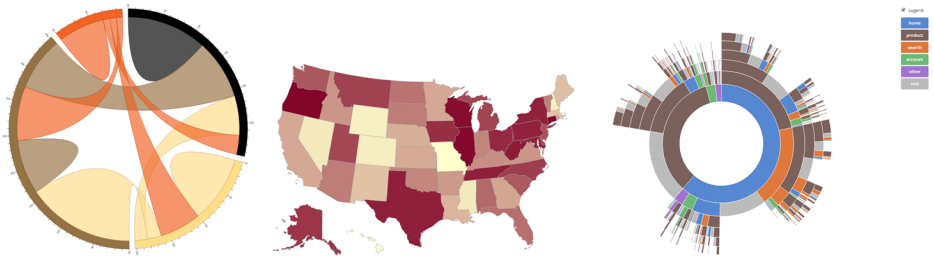
jakákoliv změna dat v komponentě, se projeví v HTML DOM šablony. Princip tohoto konceptu je vidět na obrázku 8.2.

8.1.6 Služby

Služby (Services) poskytují datovou a logickou funkcionalitu, která není vázaná na konkrétní *View*. Mohou tak být použity ve více částech aplikace.

8.1.7 Směrování

Směrování (Routing) umožňuje ve frameworku přepínání mezi jednotlivými stránkami webové aplikace. Angular používá pro směrování NgModul *Router*. Tento modul mapuje URL cesty na *Views* místo plnohodnotných stránek. Při přechodu mezi stránkami *Router* zachytí požadavek webového prohlížeče a zobrazí místo nové stránky odpovídající *View*. Pomocí tohoto způsobu směrování je zaručeno chování SPA (Single Page Application). Nedochází tedy k opakovanému načítání celého obsahu stránky, ale obsah je načten pouze při prvním zobrazení.



Obrázek 8.3: Příklady vizualizace pomocí D3.js[15][16][17]

8.2 D3.js

Tato kapitola vychází ze zdrojů [13] a [14].

D3.js je JavaScriptová knihovna, která poskytuje nástroje pro vizualizaci uživatelských dat. Autorem knihovny je Mike Bostock. Jedná se o open-source projekt, který je dostupný na GitHubu[8] pod licencí licencí BSD. Lze ji tedy volně šířit pouze s uvedením autora. Knihovna umožňuje vykreslování nejrůznějších typů diagramů, grafů, map a sítí. S vykresleným výsledkem je možné dále pracovat - přeskupovat, přibližovat, animovat, atd.

D3 je zkratka pro Data-Driven Documents, kde Data představují poskytované vstupní informace a Dokument představuje cokoliv, co může být zobrazeno na webové stránce. Knihovna D3 tyto prvky propojuje dohromady a vytváří z dat dokumenty. D3.js tedy pracuje především s daty, které dostane od uživatele a na jejich základě vytvoří nějaký prvek na stránce (např. tabulka nebo graf). Některé ze způsobů jak knihovnu využít lze vidět na obrázku 8.3. Další jsou dostupné z GitHubu projektu.

Základní funkce knihovny spočívá v načtení dat, které se připojí na vybraný prvek na webové stránce. Tento prvek je dále transformován (namapován) na konkrétní formu vizualizace. Například je-li výsledným zobrazením sloupcový graf, data s větší hodnotou mohou mít větší sloupec nebo například sytější barvu. Vizualizace se dále ještě může měnit a to na základě změny dat nebo podnětu od uživatele. Pro vykreslování prvků je použito SVG[9].

8.3 popoto.js

Tato kapitola vychází ze zdrojů [18] a [19]

Popoto.js je JavaScriptová knihovna, založená na knihovně D3.js, která započala vývoj v roce 2018 a je dostupná pod licencí GPLv3. Účelem této knihovny je tvorba rozhraní pro vizuální dotazování nad grafovou databází

[8] https://github.com/d3/d3
[9] Scalable Vector Graphics

Neo4j. Knihovna umožňuje přizpůsobit zobrazování výsledků dotazů a grafický vzhled celého rozhraní. Poskytuje také možnost převést vytvořený grafický dotaz do podoby v jazyce Cypher.

8.4 Neo4j

„*Yesterday's breakthrough applications were driven by big data – tomorrow's breakthrough applications will be driven by connected data.*“[20]

Výše zmíněný citát popisuje současný stav, kdy největší roli nenesou data, ale právě propojení mezi nimi, na což se zaměřuje grafová databáze Neo4j.

Grafové databáze slouží jako alternativa k relačním databázím a patří do skupiny tzv. *NoSQL*[10] databází. *NoSQL*, protože jejich data nejsou uložena v relační databázi a pro dotazování v nich není použit jazyk SQL. Grafové databáze nejsou pevně vázány principy *ACID*[11] a na úkor konzistence tak poskytují rychlejší výkon[21].

Jak bylo zmíněno výše, grafové databáze nepoužívají pro ukládání dat tabulky relačních databází, ale data jsou ukládána v podobě grafu (uzlů a hran). Vzhledem k této struktuře dat se grafové databáze využívají zejména pro vztahové problémy, kde jde záměrem pozorovat souvislosti mezi entitami (uzly). Relační databáze naopak řeší spíše problémy agregační[22].

Neo4j je jednou z nejpopulárnějších grafových databází[20]. Community verze určená k nekomerčnímu použití je dostupná pod licencí GPL.

Pro implementaci prototypu je použita Community verze 3.5.12. Pro komunikaci mezi JavaScriptem a databází byla použita knihovna neo4j-driver[12].

8.5 Alternativní technologie

V této kapitole jsou zmíněny některé alternativní technologie, které také řeší danou problematiku. Blíže jsou popsány JavaScriptové knihovny a nástroje pro vizualizaci grafů.

8.5.1 AngularJS

Jak bylo zmíněno v předchozí části8.1, Angular je JavaScriptový framework určený pro tvorbu webových single page aplikací. Specificky AngularJS je framework Angular verze 1, který měl značně odlišnou strukturu od verze Angular 2+. V této verzi frameworku byl implementován dotazovací nástroj jazyka MBIQL[3].

[10] Not only SQL
[11] Atomicity, Consistency, Isolation, Durability
[12] https://neo4j.com/developer/javascript/

8.5.2 React

React je JavaScriptová knihovna, která slouží pro tvorbu uživatelských rozhraní. Projekt byl interně vyvíjen Facebookem a v roce 2013 open-sourcován [23]. Podobně jako Angular,React také řeší problém překreslování celého HTML DOM umožňuje vykreslovat pouze jeho nezbytné části. Výhodou tohoto frameworku je nezávislost na DOM, ale pouze na svých komponentách[24].

8.5.3 Sigmajs

Informace v této sekci byly čerpány z [25].

Sigmajs je open-sourcová JavaScriptová knihovna, která se zabývá se vizualizací grafů. Podobně jako D3.js umí vynutit specifické uspořádání uzlů a hran, které se načítají pomocí formátu JSON. Uživatel může interaktivně procházet graf a pomocí funkce zoom se zaměřit jen na určitou část grafu. Funkce jsou podobné knihovně D3.js.

8.5.4 Cytoscape

Tato sekce čerpá z [26].

Cytoscape je open-sourcová platforma pro vizualizaci komplexních sítí dat, původně určená pro práci s biologickými daty. Cytoscape je možné použít jako samostatnou aplikaci, ale dostupná je i jako JavaScriptová knihovna Cytoscape.js, kterou lze importovat do projektu.

Po importu se vytvoří konfigurační soubor ve formátu JSON, ve kterém nastaví vlastnosti grafu a HTML element, na kterém bude graf vykreslen. Po vytvoření grafu je možné jej dále upravovat, například přidávat další uzly. Příklad vizualizace lze vidět na obrázku 8.4.

8.6 Odůvodnění volby technologií

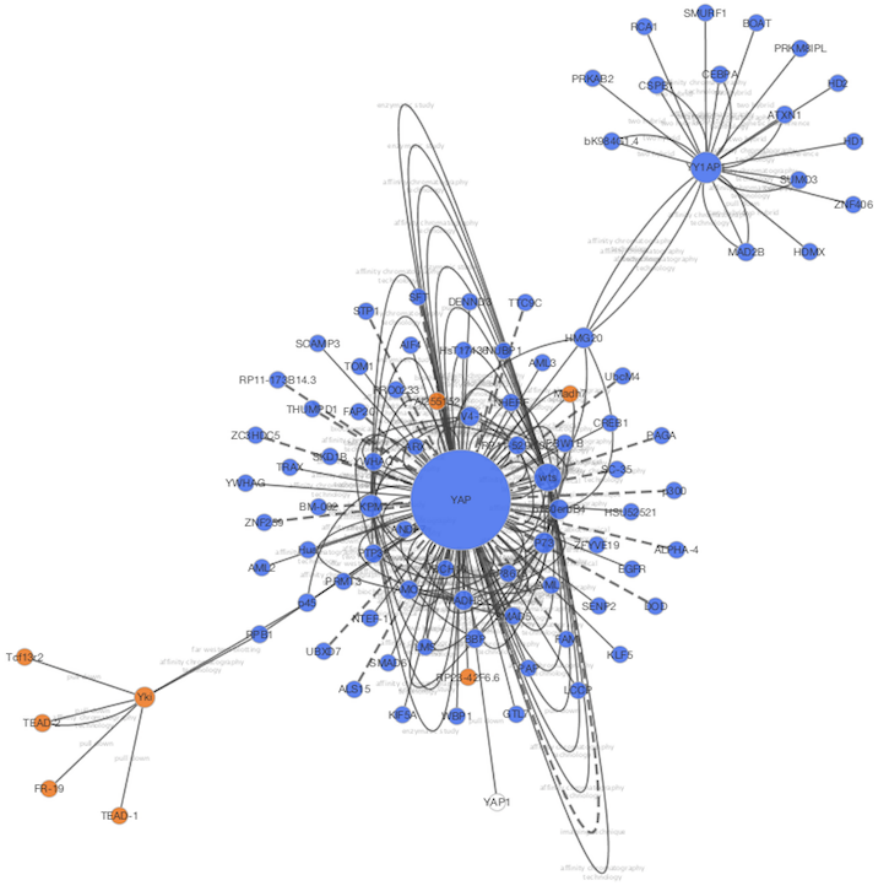
8.6.1 Angular

Jelikož aplikace pracuje s JavaScriptem, bylo vhodné zvolit nějaký JavaScriptový framework, pro usnadnění vývoje, namísto selektování a vkládání HTML elementů ze samotného JavaScriptu.

Angular 2+ byl zvolen, jelikož jeho funkce a možnosti jsou pro aplikaci dostačující, a z výše zmíněných frameworků působí jako nejjednodušší na uchopení bez předchozích zkušeností, které autorovi scházejí.

8.6.2 D3.js

Pro vizualizaci dotazovacího grafu bylo na výběr z mnoha možností. Avšak pouze knihovna D3.js se jeví jako dostačující pro tvorbu interaktivního, měnitelného grafu, na kterém je založen návrh jazyka MBINL. Protože se také



Obrázek 8.4: Vizualizace grafu: Cytoscape[27]

jedná o open-source knihovnu, je možné ji v případě nutnosti dále rozšiřovat a přizpůsobit.

8.6.3 popoto.js

Jako další nástroj pro vizualizaci byla zvolena knihovna popoto.js, která je na knihovně D3.js postavena. Na rozdíl od široko zaměřené knihovny D3.js je popoto.js specificky zaměřena na tvorbu rozhraní pro grafické dotazování nad grafovou databází Neo4j. Jeví proto jako ideální kandidát pro implementaci prototypu, který má především demonstrovat princip dotazování pomocí jazyka MBINL.

8.6.4 Neo4j

Pro ukládání dat dat byla zvolena databáze Neo4j. Jedná se o databázi použitou i v předešlých pracích. Neo4j je navíc leaderem na poli grafových databázích, proto není důvod tuto technologii měnit.

Kapitola 9

Realizace prototypu

Tato kapitola se zabývá implementací prototypu. Nejdříve je popsán proces vytvoření databáze Neo4j a import dat do databáze. Poté je popsán princip tvorby dotazovacího grafu a následně způsoby zobrazování výsledků a správy dotazovacích grafů. Na závěr je popsáno přihlašování uživatelů do systému.

9.1 Import dat do databáze Neo4j

Než bude realizována samotná aplikace je potřeba zpracovat vstupní data, nad kterými se v aplikaci provádí dotazování. Existují momentálně dva přístupy. V původní práci [1] byla data z báze MBI obdržena ve formátu XML a poté transformována do formátu CSV pomocí XSLT transformace. Tento proces již není zapotřebí, protože data dodaná z MBI jsou již ve formátu CSV v podobě tří souborů. Jeden soubor obsahuje Uzly, druhý obsahuje Hrany a třetí soubor obsahuje všechny typy Uzlů a Hran, které mapuje na jejich názvy v přirozeném jazyce. Druhý přístup, použitý v práci [3], data zpracoval a importoval pomocí vlastního programu v jazyce Java. Pro import dat v této práci byly vytvořeny skripty v jazyce Python s využitím ovladače neomodel pro komunikaci s databází Neo4j.

Nejprve je nutné připravit soubory CSV k importu do databáze.

Seznam Uzlů je již kompletní, a není tak potřeba jej dále měnit. Je uložen v souboru *finalNode.csv*.

Dále je zpracován seznam Hran. Ten obsahuje pouze Hrany instanční (typ II). Je tedy nutné vytvořit hrany hierarchické (typ IH). Pro tento účel je využit soubor Uzlů, který obsahuje u relevantních uzlů atribut SUPERIOR_OBJECT, který odkazuje na jejich nadřazený Objekt v modelu MBI. Pomocí atributů OBJECT_ID a SUPERIOR_OBJECT jsou vytvořeny hierarchické hrany. Hrany jsou uloženy do souboru *finalRelation.csv*.

Poslední soubor *finalMeta.csv* mapující atribut OBJECT_TYPE pro všechny typy Uzlů a Hran na přirozený jazyk je již také kompletní.

Příprava souborů je dokončena, následně je soubor Uzlů převeden do podoby Cypher dotazů a vložen do importovacího skriptu. Obdobně je zpracován soubor Hran.

Následně se importovací skript připojí pomocí ovladače neomodel k databázi Neo4j. Nejprve zkusí smazat Uzly i Hrany, které by mohly v databázi již existovat. Poté vykoná samotný import, nejprve Uzlů, poté všech Hran.

9.2 Konstrukce dotazovacího grafu

Tvorba grafu využívá několika grafických prvků. První je seznam všech Objektů modelu MBI, který je hierarchicky strukturován dle datového modelu 1.1. Druhým prvkem je SVG plátno, na kterém uživatel interaguje s dotazovacím grafem, který je tvořen hlavními grafickými prvky, Uzly a Hranami.

Některé aspekty dotazování se v implementaci prototypu liší od původního návrhu jazyka MBINL 4 a jeho navržené implementace 6. Je tomu tak kvůli využití prvků z knihovny popoto.js, popsané v předchozí kapitole 8.3. některé tyto prvky se jevily jako potenciálně lepší alternativa k původnímu návrhu. Díky těmto změnám se každý Uzel nyní skládá ze dvou částí:

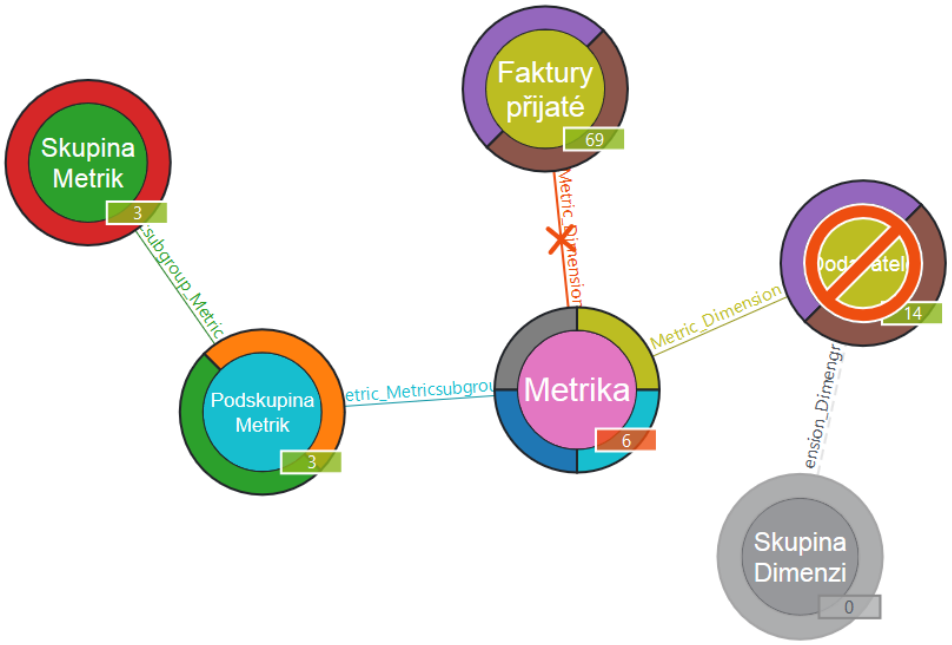
- **Tělo uzlu**: Představuje samotný Objekt, který uzel reprezentuje.
- **Okraj uzlu**: Může být rozdělen na několik částí. Představuje související Objekty MBI, se kterými má daný Objekt navázaný Vztah.

Další změnou je interakce s Hranami, které byly v původním návrhu zcela pasivním prvkem. Vliv těchto změn je popsán u jednotlivých operací.

V hierarchickém Seznamu tedy uživatel zvolí požadovaný *kořenový uzel*. Tento uzel představuje typ Objektu v databázi, kterým se uživatel chce zabývat. Ve výchozím stavu je jako kořenový uzel zvolen Objekt *Úloha* protože představuje středobod celého datového modelu. 1.1

Na Plátně je zobrazen dotazovací graf, nejprve pouze v podobě kořenového uzlu. Interakcí s kořenovým uzlem a posléze dalšími Uzly a Hranami na Plátně může uživatel konstruovat dotazovací graf, který tak postupně specifikuje, kterých instancí může kořenový uzel nabývat. Interagovat s grafem lze pomocí několika základních operací, jejichž chování se dále liší dle kontextu operace.

- Kliknutí na tělo uzlu
- Kliknutí na okraj uzlu
- Kliknutí na hranu spojující dva uzly



Obrázek 9.1: Příklad grafu z MBI v prototypu

9.2.1 Kliknutí na tělo uzlu

Ve výchozím stavu, kliknutí na tělo uzlu slouží ke *Specifikaci Uzlu*. Uživateli se zobrazí seznam všech instancí Objektu, kterých může Uzel nabývat v současném dotazovacím grafu. Zvolená instance je přiřazena danému Uzlu a ostatní Uzly jsou aktualizovány aby jejich potenciální instance byly pouze ty, které mají vztah ke zvolené instanci.

Seznam instancí má podobu instančních Uzlů, které obklopují původní Uzel. Pokud je instancí více, než se vejde instančních Uzlů kolem Uzlu původního, je seznam „stránkován“. Jednotlivé „stránky“ seznamu lze procházet pomocí šipek na těle původního Uzlu. Tato funkcionalita vychází z knihovny popoto.js a liší se od původního návrhu, který seznam instancí zobrazoval jako grafický prvek Seznam s možností vyhledávání. Odůvodněním pro tuto změnu je fakt, že tento přístup neodvádí pozornost uživatele od dotazovacího grafu k sekundárním grafickým prvkům mimo Plátno.

9.2.1.1 Kliknutí na uzel s již přiřazenou instancí

Na *Specifikovaný Uzel* lze opět kliknout. Tato operace slouží jako operátor *OR*. Potenciální instance souvisejících Uzlů tak mohou být pouze takové, které mají Vztah s jednou z přiřazených instancí daného Uzlu.

9.2.1.2 Kliknutí na uzel pravým tlačítkem

Kontextové kliknutí na uzel (pravým tlačítkem myši), které obvykle zobrazuje kontextovou nabídku, slouží k odstranění již přiřazených instancí z daného Uzlu. Odstraňování probíhá individuálně v opačném pořadí, než ve kterém byly instance přiřazeny.

9.2.1.3 Ctrl + Kliknutí na uzel

Kliknutí na uzel s přidržením tlačítka *Control* (Ctrl) slouží jako *Negace Uzlu*, tedy operátor *NOT*.

Kliknutí na Uzel bez přiřazených instancí provede negaci celého typu Objektu. Potenciální instance souvisejících Uzlů tedy mohou být pouze takové, které nemají vztah s žádnou instancí daného Objektu. Tato operace také provede negaci všech Uzlů navazujících na znegovaný Uzel směrem od kořenového uzlu.

Kliknutí na Uzel s přiřazenými instancemi provede negaci daných instancí Objektu. Potenciální instance souvisejících Uzlů tedy mohou být pouze takové, které nemají vztah s přiřazenými instancemi Objektu daného Uzlu.

9.2.2 Kliknutí na okraj uzlu

Kliknutí na okraj uzlu, slouží k *Přidání Uzlu do grafu*. Okraj Uzlu je rozdělen na části podle počtu souvisejících typů Objektů MBI. Kliknutím na odpovídající část okraje je do grafu přidán Uzel odpovídajícího typu Objektu. Přidaný uzel je poté možno *Specifikovat*, *Negovat* nebo dále přidávat související uzly.

Každý typ Objektu lze přidat do grafu jako Uzel více než jednou. Tato operace slouží jako operátor *AND*. Pokud jsou takto přidaným uzlům poté přiřazeny instance *Specifikací Uzlu*, potenciální instance souvisejících Uzlů mohou být pouze takové, které mají Vztah se všemi takto přiřazenými instancemi.

Tato operace využívá funkcionality knihovny popoto.js a umožňuje přidávat Uzly do grafu pouhým kliknutím na odpovídající část *okraje* Uzlu. V původním návrhu jazyka bylo nutné kliknout na Uzel a zvolit ze seznamu odpovídající typ Uzlu, který byl posléze přidán.

Jedná se také o alternativu k logickému operátoru AND. V původním návrhu jazyka byl tento operátor součástí stejné operace jako operátor OR 9.2.1.1. Při přiřazení více instancí k Uzlu bylo tedy nutné zvolit v jakém logickém vztahu se budou nacházet. Nebylo tak možné operátory efektivně kombinovat. Rozdělení logických operátorů do dvou různých operací to však umožňuje.

9.2.3 Kliknutí na hranu

Kliknutí na Hranu spojující dva Uzly provede *Odebrání Uzlu z grafu*. Odebrání Uzlů probíhá vždy směrem od kořenového uzlu. Tedy Uzel, který je na Hraně

vzdálenější o kořenového uzlu je z grafu odstraněn. Pokud na daný Uzel není navázán žádný další související Uzel, je odstraněn pouze tento Uzel. Pokud jsou na daný Uzel navázány další související Uzly, jsou všechny takové Uzly také odstraněny.

Tato operace využívá funkcionality knihovny popoto.js. Jelikož Hrany nemají v návrhu přiřazeny žádné další operace, je možné Uzly odstraňovat pouhým kliknutím na související Hranu. V původním návrhu jazyka bylo nutné kliknout na Uzel a zvolit, že má být z grafu odstraněn.

9.3 Možnosti zobrazení grafu

V této sekci jsou uvedeny operace, které upravují zobrazení dotazovacího grafu a Plátna.

9.3.0.1 Zvětšení SVG plátna

Kliknutím na ikonu v nabídce v horní části Plátna je možné Plátno převést na mód celé obrazovky.

9.3.0.2 Resetování SVG Plátna

Kliknutím na ikonu v nabídce v horní části Plátna je možné celé Plátno uvést do výchozího stavu. To znamená, že dosavadní konstruovaný graf je smazán a na Plátně je zobrazen pouze zvolený kořenový uzel.

9.3.0.3 Vycentrování SVG plátna

Kliknutím na ikonu v nabídce v horní části Plátna je možné Plátno „vycentrovat“. Kořenový uzel je tak umístěn doprostřed Plátna bez ohledu na tvar dotazovacího grafu.

9.4 Zobrazení výsledků

Výsledky momentálně zkonstruovaného grafu jsou zobrazeny ve spodní části aplikace a jsou průběžně aktualizovány během konstrukce grafu.

9.4.0.1 Zobrazení Cypher dotazu

První část obsahuje dotaz v jazyce Cypher, který odpovídá právě zkonstruovanému dotazovacímu grafu. Tato funkcionalita pochází z knihovny popoto.js a v původním návrhu nebyla obsažena.

9.4.0.2 Zobrazení vásledků grafu

Druhá část výsledků obsahuje seznam všech potenciálních instancí kořenového uzlu v daném dotazovacím grafu. Každá položka obsahuje název instance a její atributy. Dále všechny položky obsahují odkaz na detail instance daného Objektu na Portálu MBI.

9.5 Správa grafů

V této sekci jsou popsány operace pro správy vytvořených dotazovacích grafů.

9.5.1 Ukládání grafů

Kliknutím na ikonu v nabídce v horní části Plátna je možné uložit do aplikace právě konstruovaný dotazovací graf. Graf je po uložení zobrazen v Seznamu v pravé části aplikace.

9.5.2 Zobrazení grafů

Kliknutím na položku v Seznamu uložených grafů je možné zobrazit zvolený graf. Obsah Plátna je poté smazán a zvolený dotazovací graf je vykreslen na plátně.

9.5.3 Zobrazení grafů

Kliknutím na ikonu odstranění v Seznamu uložených grafů je možné odstranit zvolený uložený graf z aplikace.

9.6 Autentizace uživatelů

Pro přihlášení uživatele jsou využity přihlašovací údaje do databáze Neo4j. Při potvrzení uživatelského jména i hesla je proveden pokus o vytvoření spojení do databáze. V případě úspěchu je uživatel přihlášen do systému. V případě neúspěchu je vyzván autentizaci opakovat.

Kapitola 10

Ověření prototypu

V této kapitole jsou provedeny testy, které ověřují funkcionalitu prototypu. Nejprve zkontrolována správnost importu dat do databáze. Dále proběhlo ověření vzorových dotazů z kapitoly 4.2. Nakonec je na prototypu proveden uživatelský test.

10.1 Import dat do databáze

Kontrola importu dat do databáze slouží k ověření, zda nebyla žádná importovaná data ztracena. Je porovnán počet importovaných uzlů a hran s počtem skutečných dat obsažených v databázi. Výsledky obdržené při importu dat by tak měly odpovídat výsledkům Cypher dotazů na dané prvky v databázi.

```
Total of 2123 Nodes created;
Total of 14805 Relationships created;

MATCH (n) RETURN count(n) 2123
MATCH ()-[r]->() RETURN count(r) 14805
```

Z výsledků importu a Cypher dotazů lze vidět, že počet všech importovaných prvků odpovídá prvkům v databázi.

10.2 Vzorové dotazy

V této kapitole jsou na prototypu ověřeny vzorové dotazy z kapitoly 4.2. Při konstrukci dotazovacího grafu je pro každý dotaz odsimulováno chování uživatele. Následně jsou zkontrolovány zobrazené výsledky s uloženými vstupními daty z databáze.

1. *Zobraz instanci Objektu Úloha, která má název: Analýza datových zdrojů.*

Objekt Úloha je již zvolen jako výchozí kořenový uzel. Volbu kořenového uzlu může uživatel vynechat. Poté kliknutím na tělo Uzlu typu Úloha uživatel zobrazí dostupné instance tohoto Uzlu. Z těchto instancí zvolí instanci s názvem „Analýza datových zdrojů“.

Výsledkem je jeden záznam v seznamu výsledků typu Úloha s názvem: Analýza datových zdrojů a kódem U201A. *(Správně)*

Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (uloha:‘Uloha‘)
WHERE (uloha.gid = "TASK-43")
RETURN uloha
```

2. *Zobraz všechny Metody, které mají vztah k Úloze: Analýzy výnosů z IT služeb.*

   Uživatel nejprve zvolí ze seznamu Objektů jako kořenový uzel Objekt Metoda. Poté klikne na odpovídající část okraje kořenového uzlu a zvolí vztah [Task_Method]. Do grafu je přidán Uzel typu Úloha. Poté kliknutím na tělo Uzlu typu Úloha uživatel zobrazí dostupné instance tohoto Uzlu. Z těchto instancí zvolí instanci s názvem „Analýzy výnosů z IT služeb“.

   Výsledkem je 10 záznamů v seznamu výsledků typu Metoda. *(Správně)*

   Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (metoda:‘Metoda‘),
(metoda:‘Metoda‘)-[:‘TASK_METHOD‘]-(uloha:‘Uloha‘)
WHERE (uloha.gid = "TASK-62")
RETURN metoda
```

3. *Zobraz Skupiny úloh, které nespadají do Domény: „Strategické řízení IT“.*

   Uživatel nejprve zvolí ze seznamu jako kořenový uzel Objekt typu Skupina úloh. Poté klikne na odpovídající část okraje kořenového uzlu a zvolí vztah [Taskgroup_Domain]. Do grafu je přidán Uzel typu Doména. Poté kliknutím na tělo uzlu typu Doména uživatel zobrazí dostupné instance tohoto uzlu. Z těchto instancí zvolí instanci s názvem „Strategické řízení IT“. Nakonec uživatel provede negaci Uzlu Doména pomocí Ctrl + kliknutí na tělo Uzlu.

   Výsledkem je 57 záznamů v seznamu výsledků typu Skupina Úloh. *(Správně)*

   Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (skupinauloh:‘Skupina uloh‘),
(skupinauloh:‘Skupina uloh‘)
-[:‘TASKSGROUP_DOMAIN‘]-(domena:‘Domena‘)
WHERE NOT (domena.gid = "DOMAIN-1")
RETURN skupinauloh
```

4. *Najdi pouze Faktory, které nejsou využity u žádného Scénáře.* Uživatel nejprve zvolí ze seznamu jako kořenový Uzel Objekt typu Faktor. Uzel typu Scénář není přímo propojený s uzlem typu Faktor, jsou propojeny přes typ Úloha. Uživatel poté tedy klikne na odpovídající část okraje kořenového uzlu a zvolí vztah [Task_Factor]. Do grafu je přidán Uzel typu Úloha. Poté kliknutím na odpovídající část okraje Uzlu typu Úloha a volbou vztahu [Task_Scenario] uživatel přidá do grafu Uzel typu Scénář. Nakonec uživatel provede negaci Uzlu Scénář pomocí Ctrl + kliknutí na tělo Uzlu.

   Výsledkem je 60 záznamů v seznamu výsledků typu Faktor. *(Správně)*

   Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (faktor:‘Faktor‘)
WHERE (NOT exists((faktor)
-[:‘Task_Factor‘]-
(:‘Uloha‘)
-[:‘Task_Scenario‘]-
(:‘Scenar‘)))
RETURN faktor
```

5. *Zjisti, do jaké Skupiny metrik patří Metrika: Finanční náklady na bezpečnost IT služeb.*

   Uživatel nejprve zvolí ze seznamu jako kořenový uzel Objekt typu Skupina metrik. Uzel typu Metrika není přímo propojený se Skupinou metrik, jsou propojeny přes uzel typu Podskupina metrik. Uživatel poté tedy klikne na okraj uzlu a zvolí vztah typu [Metricsubgroup_Metridcroup]. Do grafu je přidán uzel typu Podskupina metrik. Poté kliknutím na okraj tohoto uzlu a volbou vztahu [Metric_Metricsubgroup] přidá do grafu uzel typu Metrika. Nakonec kliknutím na tělo uzlu zobrazí dostupné instance tohoto uzlu. Z těchto instancí zvolí instanci s názvem „Finanční náklady na bezpečnost IT služeb“.

   Výsledkem je jeden záznam v seznamu výsledků typu Skupina metrik: Metriky řízení IT služeb. *(Správně)*

   Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (skupinametrik:‘Skupina metrik‘),
(podskupinametrik:‘Podskupina metrik‘)
-[:‘METRIC_METRICSUBGROUP‘]-(metrika:‘Metrika‘),
(skupinametrik:‘Skupina metrik‘)
-[:‘METRICSUBGROUP_METRICGROUP‘]-
(podskupinametrik:‘Podskupina metrik‘)
WHERE (metrika.gid = "METRIC-226")
RETURN skupinametrik
```

6. *Zobraz kompletní hierarchii dvou Rolí: „Návrhář databází“ a „Auditor podnikové informatiky“ (kód: „R108“). K nim zobraz také všechny Úlohy, se kterými jsou vázány.*

   U tohoto dotazu by teoreticky bylo možné dotaz začít konstruovat z několika potenciálních kořenových uzlů. Jako nejužitečnější se však jeví jako kořenový uzel ponechat Objekt Úloha, protože všechny ostatní Uzly budou reprezentovat specifické instance. *Volbu kořenového uzlu* může uživatel tedy vynechat.

   Uživatel klikne na odpovídající část okraje kořenového uzlu a zvolí vztah [Task_Role]. Do grafu je přidán Uzel typu Role. Následně uživatel klikne na odpovídající část okraje Uzlu typu Role a zvolí vztah [Role_Rolegroup]. Do grafu je přidán Uzel typu Skupina Rolí. Poté, kliknutím na tělo Uzlu typu Role, uživatel zobrazí dostupné instance tohoto Uzlu. Z těchto instancí zvolí instanci s názvem „Analýza datových zdrojů“. Obdobným způsobem uživatel nad stejným Uzlem zvolí instanci s názvem „Analýza datových zdrojů“ pro vyjádření logického vztahu OR mezi zvolenými instancemi.

   Výsledkem je 11 záznamů typu Úloha. *(Správně)*

   Dotaz v jazyce Cypher, který uzly páruje podle jejich (GLOBAL_ID) pak vypadá následovně:

```
MATCH (uloha:‘Uloha‘),
(uloha:‘Uloha‘)-[:‘Task_Role‘]-(role:‘Role‘),
(role:‘Role‘)
-[:‘Role_Rolegroup‘]-(skupinaroli:‘Skupina Roli‘)
WHERE (role.gid IN ["ROLE-21", "ROLE-34"])
RETURN uloha
```

10.3 Uživatelské testování

Pro testování s uživateli byl připraven následující scénář. Uživateli je nejprve představena aplikace a její základní funkce. Testovací scénář začíná na uvodní přihlašovací stránce aplikace.

1. Přihlašte se do aplikace pomocí uživatelského jména *neo4j* a hesla *pass*.
2. Ze seznamu Objektů v levé části aplikace zvolte Objekt *Role*.
3. Najděte všechny *Role*, které souvisí s *Úlohou* s názvem „Evidence dopravy“ a zároveň nesouvisí s Úlohami „Analýza IT obchodních partnerů“ nebo „Analýza datových zdrojů“.
   a) Klikněte na okraj uzlu *Role*, který je nadepsán *Task_Role Uloha*.
   b) Klikněte na tělo nově přidaného uzlu *Uloha*.
   c) Pomocí šipek na uzlu *Uloha* najděte instanci úlohy „Evidence dopravy“.
   d) Přidejte další uzel *Uloha*.
   e) Klikněte na nově přidaný uzel *Uloha* a najděte instanci úlohy „Analýza IT obchodních partnerů“.
   f) Opět klikněte na tento uzel a najděte instanci úlohy „Analýza datových zdrojů“.
   g) Pomocí Ctrl + klik na tento uzel proveďte negaci tohoto uzlu.
   h) zobrazte si výsledky ve spodní části aplikace.
4. Dotaz uložte kliknutím na ikonu uložení v nabídce v horní části aplikace.
5. Resetujte graf kliknutím na ikonu reset v nabídce v horní části aplikace.
6. Najděte *Metriky*, které nesouvisí s *Dimenzí* „Dodavatelé“ a zároveň souvisí s *Dimenzemi* „Aplikace“ nebo „Konkurence“.
7. Zjistěte, do jaké *Skupiny Metrik* patří *Metrika* s názvem „Počet analyzovaných konkurentů“.
8. Dotaz uložte.
9. Resetujte graf.
10. Vyberte v seznamu dotazů v pravé části obrazovky dotaz s názvem *saved-query-0* a zobrazte jej.
11. Odhlašte se z aplikace.

Kapitola 11

Zhodnocení

V této kapitole je provedeno zhodnocení testů, a poté celého návrhu jazyka MBINL z hlediska možných přínosů a nákladů spojených s plnohodnotnou realizací návrhu. Nakonec jsou zhodnoceny možnosti budoucí práce a rozvoje v oblasti dané problematiky.

11.1 Zhodnocení testů

V rámci ověření prototypu byly provedeny testy bez uživatelů a následně i testy uživatelské. Nejprve byla otestována správnost importovaných dat do grafové databáze Neo4j. Poté byly otestovány vzorové dotazy a jejich správná konstrukce pomocí dotazovacího grafu. Uvedeny jsou také výsledné Cyhper dotazy, které byly následně ověřeny přímo přes rozhraní Neo4j a porovnány s výsledky v prototypu.

Jako poslední byl prototyp ověřen v rámci uživatelského testování dle scénáře popsaného v předchozí kapitole 10.3. Celková reakce uživatelů na prototyp byla pozitivní, nicméně z testování vyplynulo několik požadavků na zlepšení:

- Grafický seznam instančních Uzlů pro přiřazení instance může být s rostoucím množstvím instancí nepřehledný a zdlouhavý na procházení. Jako možné řešení se jeví implementace vyhledávání v seznamu, čímž by se procházení zjednodušilo. Alternativně lze použít původní návrh jazyka, který využíval grafický prvek Seznam zobrazující všechny instance Objektu.

- Někteří uživatele poukázali na omezené možnosti přiřazování instancí Uzlům z hlediska dostupných atributů. V rámci návrhu je možné přiřazovat pouze celé instance na základě názvu. Bylo by tedy nutné navrhnout způsob pro přiřazování instancí na základě libovolného atributu, nebo přiřazování jednotlivých atributů přímo.

11.2 Přínos jazyka MBINL

Hlavním účelem a tedy také zamýšleným přínosem návrhu jazyka MBINL je efektivní navigace v bázi MBI a intuitivní vizualizace vztahů mezi objekty této báze. Jedna metoda intuitivní vizualizace vztahů již byla realizována v rámci návrhu jazyka MBIQL [2][3], bylo jí však docíleno na úkor složitosti dotazovacího rozhraní. Jazyk MBINL tak byl navržen s cílem vizualizovat bázi MBI bez značného navýšení složitosti dotazování pro uživatele. Místo statické vizualizace po vyhodnocení nějakého dotazu byl navržen způsob dotazování, kde samotná vizualizace je dotazovacím rozhraním. Uživatel tak může interagovat s uzly a hranami grafové reprezentace MBI a přímo tvořit dotazovací graf.

11.3 Náklady spojené s realizací

V průběhu vývoje a testování byly identifikovány určité nedostatky návrhu, implementace a zvolených technologií, které bude nezbytné zohlednit při případné, plnohodnotné realizaci návrhu jazyka MBINL.

Některé takové nedostatky vyplynuly z uživatelského testování 11.1. Jako další, hlavní nedostatek se projevila knihovna popoto.js 8.3 použitá v rámci prototypu. Z hlediska funkcionality se jedná o robustní nástroj pro tvorbu grafických dotazovacích rozhraní. Problém však nastává ve struktuře kódu, který trpí vysokou provázaností jednotlivých komponent. Je tak obtížné se v kódu orientovat a rozšiřovat jej pro přizpůsobení existující a definování vlastní funkcionality. V případné realizaci by tak bylo vhodné knihovnu použít jako referenci pro návrh některých struktur a funkcionalit, nikoliv ji však používat přímo.

11.4 Budoucí práce

Ačkoliv návrh jazyka MBINL splnil vytyčený cíl efektivní a intuitivní navigace v bázi MBI, existují vždy oblasti pro další rozvoj problematiky. Částečným nedostatkem přístupů jako jsou jazyky MBIQL a MBINL je jejich samotné zaměření na vztahy místo obsahu. V případě báze MBI jsou žádoucí oba aspekty pro utvoření celistvého pohledu na řešené problémy. Jednou z oblastí dalšího rozvoje by tedy mohl být způsob jak efektivně zkombinovat aspekty grafových a dokumentových databází a zajistit tak intuitivní navigaci mezi objekty MBI a zároveň dostatečnou míru informací o jednotlivých objektech.

Závěr

Cílem diplomové práce bylo navrhnout alternativní způsob pro navigaci v bázi MBI a vytvořit prototyp, který demonstruje princip tohoto způsobu navigace.

V první kapitole (viz 1) byl popsán současný stav portálu MBI a jeho struktura.

Ve druhé kapitole (viz 2) byly analyzovány dosavadní přístupy k navigaci v bázi MBI a vizualizaci vztahů mezi Objekty báze. Nakonec byl nastíněn nový, alternativní přístup k navigaci.

Ve třetí kapitole (viz 3) byl analyzován současný datový model pro grafovou databázi MBI a zhodnoceno zda je nutné v modelu provést změny pro účely nového způsobu navigace.

Ve čtvrté kapitole (viz 4) byl popsán návrh nového navigačního jazyka MBINL nad datovým modelem MBI.

V páté kapitole (viz 5) byl nově navržený jazyk srovnán s předchozím návrhem jazyka MBIQL.

V šesté kapitole (viz 6) byl popsán návrh implementace dotazovacího nástroje pro navigační jazyk.

V sedmé kapitole (viz 7) byl uveden návrh jednoduchého prototypu na základě návrhu implementace, který demonstruje princip dotazování v navigačním jazyce.

V osmé kapitole (viz 8) byly popsány technologie zvolené k implementaci prototypu dotazovacího nástroje. Zejména tedy framework Angular, knihovny D3.js a popoto.js a databáze Neo4j. Zmíněny byly také některé alternativní technologie.

Ve deváté kapitole (viz 9) byla realizována implementace prototypu na základě předchozího návrhu a zvolených technologií.

V desáté kapitole (viz 10) byl prototyp otestován z pohledu správnosti dat a z hlediska uživatelské použitelnosti.

V poslední kapitole (viz 11) byl celý návrh navigačního jazyka zhodnocen společně s přínosy a náklady spojenými s možnou realizací.

Za osobní přínos lze pokládat seznámení se s frameworkem Angular a knihovnami D3.js a popoto.js, díky kterým bylo možné návrh nového jazyka ve zjednodušeném prototypu realizovat.

Literatura

[1] Stránský, V.: *Vizualizace vztahů v informacní bázi MBI.* Bakalářská práce, České vysoké ucení technické v Praze, Fakulta informačních technologií, 2014, [Cit. 24.3. 2019].

[2] Batík, O.: *Dotazovací jazyk pro vztahy MBI objektu.* Bakalářská práce, České vysoké ucení technické v Praze, Fakulta informačních technologií, 2015, [Cit. 24.3. 2019].

[3] Stránský, V.: *Návrh a implementace software pro interaktivní práci s objekty MBI pomocí jazyka MBIQL.* Diplomová práce, České vysoké ucení technické v Praze, Fakulta informačních technologií, 2017, [Cit. 24.3. 2019].

[4] Pour, J.: MBI [online]. [Cit. 24.3. 2019]. Dostupné z: `https://mbi.vse.cz`

[5] MBI: *MBI, Management Byznys Informatiky - Koncepce a návod k použití [online].* [Cit. 24.3. 2019]. Dostupné z: `http://mbi.vse.cz/mbi/file/help.pptx`

[6] Voríšek, J. k., J.; Pour: *Management podnikové informatiky.* Professional Publishing, první vydání vydání, 2012, ISBN 978-80-7431-102-34, [Cit. 5.4. 2019].

[7] Jahoda, B.: Single page application [online]. 2015, [Cit. 10.5. 2019]. Dostupné z: `http://jecas.cz/spa`

[8] Botelho, L.: Learn Angular, The Essentials [online]. 2017, [Cit. 10.5. 2019]. Dostupné z: `http://www.learn-angular.org/lessons/theessentials`

[9] Manjunath, M.: AngularJS and Angular 2+: a Detailed Comparison [online]. 2015, [Cit. 11.5. 2019]. Dostupné z: `https://www.sitepoint.com/angularjs-vs-angular/`

[10] TutorialsPoint: What is AngularJS? [online]. 2017, [Cit. 10.5. 2019]. Dostupné z: https://www.tutorialspoint.com/angularjs/angularjs_overview.htm

[11] Google: Angular Docs, Architecture overview [online]. [Cit. 11.5. 2019]. Dostupné z: https://angular.io/guide/architecture

[12] Elegance, J.: Two way data binding vs Traditional Approach in AngularJS [online]. Listopad 2015, [cit. 5.6. 2019]. Dostupné z: https://javaelegance.blogspot.com/2015/11/two-way-data-binding-angularjs-example.html

[13] Bostock, M.: Data-Driven Documents [online]. 2017, [Cit. 15.6. 2019]. Dostupné z: https://d3js.org/

[14] Murray, S.: *Interactive Data Visualization for the Web: An Introduction to Designing with D3*. O'Reilly Media, druhé vydání, Srpen 2017, ISBN 978-1491921289, [Cit. 15.6. 2019].

[15] Bostock, M.: Chord Diagram [online]. 2017, [Cit. 15.6. 2019]. Dostupné z: https://bl.ocks.org/mbostock/4062006

[16] Pasha: US State Map [online]. 2017, [Cit. 15.6. 2019]. Dostupné z: http://bl.ocks.org/NPashaP/a74faf20b492ad377312

[17] Rodden, K.: Sequences sunburst [online]. 2017, [Cit. 15.6. 2019]. Dostupné z: https://bl.ocks.org/kerryrodden/7090426

[18] Neo4j, I.: Graph Visualization Tools [online]. [Cit. 1.7. 2019]. Dostupné z: https://neo4j.com/developer/tools-graph-visualization/

[19] Interactive, N.: Popoto Github [online]. [Cit. 20.7. 2019]. Dostupné z: https://github.com/Nhogs/popoto/wiki

[20] Neo4j, I.: The Internet-Scale Graph Platform [online]. [Cit. 20.7. 2019]. Dostupné z: https://neo4j.com/product/

[21] Ramba, J.: Grafová terminologie a dostupné technologie [online]. Říjen 2013, [Cit. 29.7. 2019]. Dostupné z: https://www.zdrojak.cz/clanky/grafova-terminologie-a-dostupne-technologie//

[22] Holý, J.: Grafové databáze a neo4j [video] [online]. 2012, [Cit. 29.7. 2019]. Dostupné z: https://www.youtube.com/watch?v=lLuqIcYvWUQ

[23] Mikšu, V.: React - Úvod [online]. 2016, [Cit. 29.7. 2019]. Dostupné z: https://www.dzejes.cz/react-uvod.html

[24] Dostál, A.: Vývoj webových aplikací: React a Angular 2 [online]. Září 2016, [Cit. 29.7. 2019]. Dostupné z: http://www.aspectworks.com/2016/09/vyvoj-aplikaci-react-angular-2/

[25] Jacomy, A.: SigmaJS [online]. 2016, [Cit. 10.8. 2019]. Dostupné z: https://github.com/jacomyal/sigma.js/wiki

[26] Consortium, T. C.: What is Cytoscape? [online]. 2017, [Cit. 24.8. 2019]. Dostupné z: http://www.cytoscape.org/what_is_cytoscape.html

[27] Consortium, T. C.: Cytoscape User Manual [online]. 2017, [Cit. 24.8. 2019]. Dostupné z: http://manual.cytoscape.org/en/stable/Cytoscape.js_and_Cytoscape.html

Příloha **A**

Seznam použitých zkratek

ACID Atomicity, Consistency, Isolation, Durability

CSS Cascading Style Sheets

CSV Comma Separated Value

DB Database

DOM Document Object Model

HTML HyperText Markup Language

IH Instance-Hierarchy

II Instance-Instance

IM Instance-Model

IT Information Technology

JSON JavaScript Object Notation

MBI Management of Business Informatics

MBINL Management of Business Informatics Navigation Language

MBIQL Management of Business Informatics Query Language

NoSQL Not Only Structured Query Language

SPA Single Page Application

SVG Scalable Vector Graphics

XML eXtensible Markup Language

Příloha B

Konfigurační a instalační příručka

B.1 Systémové požadavky

- Python 3 (testováno na verzi 3.7.4) s moduly *unidecode* a *neomodel*
- Doporučený operační systém Windows 7 nebo vyšší

B.2 Instalace prototypu

1. Stáhnout a nainstalovat databázi Neo4j (testována verze 3.5.12) z `https://neo4j.com/`
2. Spustit Neo4j, vytvořit novou databázi a poté ji spustit.
3. Importovat data do databáze pomocí skriptů process-csv.py a data-import.py ve složce src/import/:

   ```
   python ./process-csv.py
   python ./data-import.py
   ```

4. Stáhnout a nainstalovat server Apache, dostupný například zde `http://www.wampserver.com/en/` jako součást WAMP stacku.
5. Umístit složku prototype/dist/mbi-visual do složky www/ serveru Apache.
6. Otevřít URL adresu aplikace: http://localhost/mbi-visual.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `import` ..... skripty a data pro import do databáze Neo4j
  - `prototype` ..... zdrojové kódy prototypu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Neumann_Marek_2020.pdf` ..... text práce ve formátu PDF