Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie používateľského rozhrania z grafových dát

Bakalárska práca

2024
Silvia Bieliková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie používateľského rozhrania z grafových dát

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	doc. RNDr. Martin Homola, PhD.
Konzultant:	Mgr. Ján Kľuka, PhD.

Bratislava, 2024
Silvia Bieliková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Silvia Bieliková
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Generovanie používateľského rozhrania z grafových dát
User interface generation from graph-based data

Anotácia: Virtuálne výučbové prostredie course.matfyz.sk využíva plne sémantickú reprezentáciu dát uložených v grafovej databáze. Tieto údaje obsahujú bohaté informácie o kurzoch, kurzových udalostiach, úlohách, materiáloch, vzdelávacích cieľoch a témach (súborne „kontext štúdia“).

Cieľ: - Analyzovať požidavky používateľov na vizualizáciu a interakciu s objektami kontextu štúdia
- Navrhnúť a implementovať vhodné používateľské rozhranie pre tieto úlohy, ktoré bude automaticky generované na základe sémantického modelu dát

Literatúra:
1. Homola, M., Kubincová, Z., Urbánek, R., Kľuka, J., 2023. Tracking the Adaptive Learning Process with Topics Ontology. In: ISWC 2023, Sydney, Australia. Springer
2. Antoniou, G. and Van Harmelen, F., 2004. A semantic web primer. MIT press
3. Guarino, N., Oberle, D. and Staab, S., 2009. What is an ontology?. Handbook on ontologies, pp.1-17.

Vedúci: doc. RNDr. Martin Homola, PhD.
Konzultant: Mgr. Ján Kľuka, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.10.2023

Dátum schválenia: 10.10.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam ukážok

Úvod

Časť I

Súčasný stav

Kapitola 1

Sémantický web

Sémantický web môžeme zadefinovať ako súbor štandardov a technológií, ktoré spolu tvoria nadstavbu pre World Wide Web. Koncept sémantického webu nemá byť teda žiadnou náhradou aktuálneho webu, má ho jednoducho rozšíriť a zlepšiť. Táto myšlienka vznikla s účelom zefektívniť a zrýchliť prácu s dátami a predovšetkým kladie dôraz na presný význam dát a jeho zachovanie pri prenose. Chce teda docieliť, aby boli interpretácie rovnakých konceptov na webe naozaj identické a zrozumiteľné nielen pre používateľov, ale aj pre stroje, a aby tak boli autenticky zdieľateľné naprieč aplikáciami. Autorom tejto iniciatívy je Tim Berners-Lee, ktorý stojí aj za pôvodným World Wide Webom a už pri jeho vzniku v 80. rokoch mal zámer postaviť ho na hlavne sémantike. [1]

Ako je na tom sémantický web v súčasnosti? Vo svojom pomerne aktuálnom článku [2] analyzuje Hitzler jeho vývoj počas posledných 20 rokov, keďže pôvod myšlienky sémantického webu je spájaný s prvotným článkom Berners-Leeho a kol. [3] z 2001. Hitzler hodnotí, že cieľ sémantického webu ešte nebol úplne naplnený, ale táto myšlienka vytvorila základ pre množstvo užitočných aplikácií a datasetov využiteľných v praxi.

V nasledujúcich podkapitolách ponúkame prehľad niekoľkých dôležítých konceptov súvisiacich so sémantickým webom.

1.1 RDF

RDF, teda Resource Description Framework, je základná štruktúra na popisovanie dát pre účely sémantického webu. Pod anglickým pojmom resources si možno predstaviť akékoľvek zdroje - osoby, koncepty, dokumenty, teda všetko, čo sa dá popísať. Informácie o týchto zdrojoch sú obsiahnuté v trojiciach (angl. triplets), ktoré zjavne pozostávajú z podmetu, prísudku a predmetu, no v RDF kontexte sa správne nazývajú subjekt, predikát a objekt. Keďže o jednotlivých zdrojoch môže existovať viacero tvr-

dení, a tieto zdroje sa v nich v rámci trojicovej štruktúry môžu vyskytovať na rôznych pozíciach, tvrdenia spolu súvisia a tvoria graf. To, že informácie sú navzájom poprepájané v grafoch (a taktiež grafy môžu byť prepojené s inými grafmi), je najsilnejšia a najužitočnejšia vlastnosť RDF. Preto sa často v tejto súvislosti spomína pojem "linked data", teda prepojené dáta. [4]

V ukážke 1.1 vidíme príklad troch RDF trojíc, ich syntax je tu zatiaľ neformálna, konkrétne formáty si ukážeme až v 1.1.3

Ukážka 1.1: RDF trojice v neformálnej syntaxi

```
<Crime and punishment> <is> <a book>.
<Fyodor Dostoevsky> <is> <an author>.
<Crime and punishment> <is written by> <Fyodor Dostoevsky>.
```

1.1.1 IRI

Jednotlivé zdroje, o ktorých v RDF zapisujeme dáta, je potrebné jednoznačne identifikovať. Vďaka tomu ich potom vieme správne zdieľať a znovupoužívať, teda dodržiavame zásady a napĺňame ciele sémantického webu. Preto sa využíva IRI, International Resource Identifier. IRI je znakový reťazec, pozostáva z Unicode znakov, a slúži ako špecifické meno. Zásady niektorých z konkrétnych formátov (1.1.3) povoľujú aj relatívne IRI, vtedy je však nutné zadefinovať aj prefixy či bázové IRI, ktoré sa ku každému relatívnemu ešte predpoja. O každom identifikátore teda platí, že je buď absolútny, alebo sa z neho absolútny dá vytvoriť. Absolútne IRI okrem mena presne popisujú aj umiestnenie daného zdroja, dobrým príkladom takéhoto IRI je napríklad absolútna URL adresa. IRI sa môže vyskytovať na akejkoľvek pozícii v RDF trojici, teda smie identifikovať subjekt, predikát či objekt. [5]

1.1.2 Literály

Literálmi označujeme jednoduché hodnoty bežných dátových typov, teda napríklad celé, nezáporné či desatinné čísla, znakové reťazce, dátumy, atď. Keďže tieto uzly v grafe sú iba obyčajnými hodnotami a nereprezentujú žiadne zložitejšie koncepty s vlastnosťami, môžu byť v RDF trojiciach použité výhradne ako objekty, teda na treťom mieste.

Pre správne spracovanie je potrebné k literálom priradiť príslušný dátový typ, samostatne stojace literály budú vždy interpretované ako obyčajné znakové reťazce. V prípade označenia typom sa odporúča využívať dátové typy zadefinované v XML Schema, príkladom je `http://www.w3.org/2001/XMLSchema#decimal`. Ak pracujeme s literálom, ktorý je znakový reťazec, je možné k nemu navyše priradiť konkrétny jazyk [5].

V ukážke 1.2 v podkapitole o formátoch (1.1.3) sú zobrazené aj RDF trojice, ktoré na mieste objektu obsahujú typované literály.

1.1.3 Formáty

Konkrétna reprezentácia RDF trojíc záleží na zvolenom formáte, ktorých je niekoľko. Najjednoduchšia je štruktúra N-Triplets, združuje trojice vypísané za sebou v riadkoch oddelených bodkami. Formát Turte je s N-Triplets najpodobnejší, ponúka ale lepšiu čitateľnosť pre používateľa kvôli syntaktickému cukru. Aby dokument nebol zahltený dlhými IRI, môžeme použiť prefixy, ktoré zadefinujeme hneď na začiatku. Výhodou je tiež možnosť zoskupenia trojíc, ktoré sa odkazujú na rovnaký subjekt. Zaujímavým detailom je aj použitie jednoduchého `a` namiesto predikátu `rdf:type`.

V ukážke 1.2 vidno príklad Turtle syntaxe spolu so všetkými vyššie spomenutými metódami na sprehľadnenie.

Ukážka 1.2: Syntax Turtle

```
@prefix rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:fyodor-dostoevsky
    a ex:Author ;
    rdfs:label "Fyodor Dostoevsky"@en .

ex:crime-and-punishment
    a ex:Book ;
    rdfs:label "Crime and punishment"@en ;
    ex:writtenBy ex:fyodor-dostoevsky ;
    ex:hasNumberOfPages "720"^^xsd:nonNegativeInteger .
```

Ďalšie používané štruktúry majú spoločné to, že súvisia s inými už existujúcimi formátmi. Napríklad pre účely obohacovania webstránok metadátami je vhodné použiť syntax RDFa, ktorý umožňuje vložiť RDF dáta do existujúceho HTML kódu. RDF/XML formát prevzal štruktúru od XML a JSON-LD zase od JSON [4].

1.2 SPARQL

Na dopytovanie RDF štruktúrovaných dát sa používa jazyk SPARQL [6]. Rovnako ako napríklad pri relačných databázach s SQL dopytmi, aj pomocou SPARQL je možné dáta získavať, ukladať či mazať. UPDATE klauzulu SPARQL neponúka, dáta možno

aktualizovať kombináciou mazania a ukladania. Výsledky dopytov majú jeden zo štyroch formátov: XML, JSON, CSV, TSV. Okrem bežných dopytov existuje aj ASK dopyt. Odpoveďou naň je jednoduchá pravdivostná hodnota, teda áno alebo nie. CONSTRUCT dopyty zase z výsledkov vytvoria a vrátia novo vytvorený graf.

Príklad 1.3 reprezentuje SELECT dopyt, ktorý vráti všetky názvy kníh.

Ukážka 1.3: SPARQL dopyt, ktorý vráti všetky názvy kníh

```
PREFIX rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://www.example.org/>

SELECT ?bookLabel
WHERE {
  ?book a ex:Book ;
        rdfs:label ?bookLabel .
}
```

1.3 Ontológie

Ontológia má viacero definícií, ale jedna z najkonkrétnejších (spája viaceré predchádzajúce definície) je podľa Strudera v článku od Guarina [7] formulovaná ako "formálna explicitná špecifikácia zdieľanej konceptualizácie". Znie to pomerne zložito, no ide len o jasné a jednoznačné popísanie relevantných konceptov, ich vzájomných vzťahov a vlastností v zvolenej doméne, teda v časti sveta, ktorá je pre naše konkrétne účely zaujímavá. Táto časť sveta je očistená o všetky vlastnosti, ktoré v danej situácii nemajú žiaden prínos či význam, sústredíme sa teda len na naozaj podstatné jadro. Je to teda zadefinovaný slovník pojmov a množina axióm, ktoré k nemu prislúchajú. Dôležité je, že ontológie popisujú elementy, ktoré nepodliehajú dynamike, teda stav sveta sa môže zmeniť, ale ontológia zostane rovnaká. Zvyčajne je tvorba ontológie práca doménových expertov. Ontológie majú najväčší význam, ak sú verejne publikované, teda dostupné pre voľné použitie. Práve tak sa najlepšie napĺňa "zdieľaná" časť z definície ontológie, pretože opakovaným používaním sa upevňuje význam popísaných konceptov.

1.3.1 RDF Schema

RDFS, teda RDF Schema, je rozšírenie pre RDF, ktoré pomocou vopred definovaných zdrojov umožňuje popísať napríklad vlastné triedy či vlastnosti. V slovníku RDFS teda nájdeme napríklad `rdfs:Class`, `rdfs:subClassOf`, `rdfs:Property`, `rdfs:subPropertyOf` a ďalšie užitočné preddefinované zdroje vhodné napríklad na zostavenie triednej a vzťahovej hierarchie. Používanými sú aj `rdfs:domain` a `rdfs:range`, ktoré slúžia na

jednoduché obmedznie definičného oboru a oboru hodnôt predikátov. Pre účely dokumentácie je vhodné použiť `rdfs:label`, pre používateľa zrejmý a čitateľný názov, a `rdfs:comment`, podrobnejší popis zdroja. [8]

1.3.2 OWL 2

OWL 2 označuje Ontology Web Language, deklaratívny jazyk využívaný na reprezentáciu ontológií. OWL 2 možno tiež považovať za rozšírenie RDF (a tiež RDFS), ponúka však aj prostriedky na prácu so zložitejšími konceptami ako RDFS, umožňuje definovať hlavne rôzne užitočné obmedzenia. Predikáty, teda vlastnosti, sa v OWL delia na objektové (angl. object properties) a dátové (angl. datatype properties) podľa oboru hodnôt prislúchajúcich subjektov. [9] Základom pre OWL je deskripčná logika. [10]

Vyvodzovanie

Súbor znalostí popísaných pomocou OWL 2 možno ďalej spracúvať, zo zadefinovaných konceptov sa pomocou externého vyvodzovania dajú odvodiť nové informácie. Dáta tak vieme rozšíriť o trojice, ktoré napríklad v databáze aplikácie, ktorej jadro je ontológia, nemusia fyzicky existovať. Príkladom je dedičnosť tried, objekt je inštanciou nielen vymenovanej triedy, ale aj všetkých jej rodičovských tried.

Syntax

OWL 2 ontológie možno zapisovať pomocou rôznej syntaxe. RDF/XML syntax je základom, je všeobecne podporovaná, a keďže je to RDF syntax, vieme ju priamo previesť aj do Turtle formátu. Medzi ďalšie syntaxe patria Functional-Style, Manchester a OWL/RDF.

Ukážka 1.4 ponúka príklad veľmi jednoduchej ontológie, ktorá definuje triedy Book a Author a vzťahy isWrittenBy a hasNumberOfPages. príklad hasValue reštrikcie

Ukážka 1.4: Príklad jednoduchej ontológie

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://www.example.org/> .

ex:Author
    a owl:Class ;
    rdfs:label "Author"@en .

ex:Book
```

```
    a owl:Class ;
    rdfs:label "Book"@en .

ex:isWrittenBy
    a owl:ObjectProperty ;
    rdfs:label "is written by"@en ;
    rdfs:domain ex:Book ;
    rdfs:range ex:Author .

ex:hasNumberOfPages
    a owl:ObjectProperty ;
    rdfs:label "has number of pages"@en ;
    rdfs:domain ex:Book ;
    rdfs:range xsd:nonNegativeInteger .
```

Kapitola 2

Systém Courses

Courses je virtuálne výučbové prostredie vyvíjané na Fakulte matematiky, fyziky a informatiky Univerzity Komenského v Bratislave. Jadro tejto platformy tvorí bohatý ontologický model, ktorý predstavíme v nasledujúcej podkapitole. Dôležitým konceptom súvisiacim s vývojom Courses je proces adaptívneho učenia, označuje prispôsobovanie procesov individuálnym nárokom a požiadavkam jednotlivých študentov. Keďže Courses je určený na výučbu na fakulte a ponúka podporu pre rôzne prebiehajúce kurzy, je veľmi podstatné zvážiť okrem individuálneho aj časový aspekt výučby.

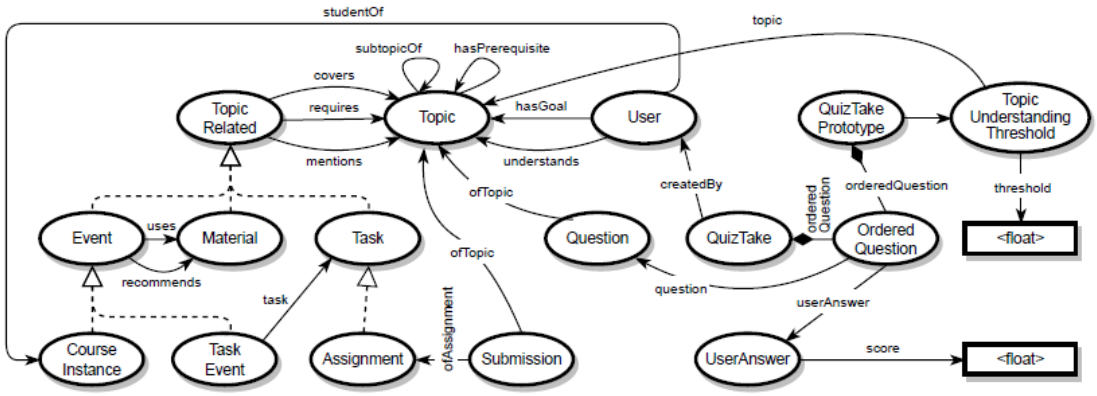
V prvom článku [11] Homola a kol. popisujú časovú zložku systému, jeho ontológiu a moduly, používateľské rozhranie, ktorého podstatnou časťou je časová os. Sústredia sa na termíny odovzdávania, blokové celky prednášok a cvičení, na časovú organizáciu kurzov počas semestra. V druhom článku [12] definujú kontext štúdia pozostávajúci z tém, materiálov, zadaní a ďalších prvkov, rozširujú ontológiu a viac rozvíjajú myšlienku prispôsobovania procesu učenia individuálnym potrebám.

Keďže táto práca je zameraná na generovanie používateľské rozhrania, viac sa budeme sústrediť na popis aktuálneho frontendu, no zhrnieme aj základy backendu a tiež komunikácie medzi týmito dvoma celkami. V nasledujúcich podkapitolách si popíšeme dôležité časti systému spolu s využívanými technológiami.

2.1 Ontologický model

Na pozadí Courses existuje bohatý ontologický model. Postupne sa rozširuje spolu s vývojom Courses. Model zahŕňa triedy ako `Course`, `CourseInstance`, `Agent`, `Result`, `Topic`, `Event` a mnohé ďalšie, popisuje vzťahy medzi nimi. Napríklad trieda `Course` definuje jednotlivé výučbové kurzy (predmety), `CourseInstance` reprezentuje konkrétny beh kurzu počas semestra.

Pre nás je najdôležitejšia trieda `Topic` a jej súvisiace vzťahy `subtopicOf` a `hasPrerequisite`, tieto tri elementy tvoria hierarchiu tém. Hierarchia je globálna, to zna-



Obr. 2.1: Zjednoduššená verzia ontológie Courses z článku Tracking the Adaptive Learning Process with Topics Ontology

mená, že témy a prerekvizity sú zdieľateľné medzi jednotlivými kurzami. Používateľ reprezentovaný triedou `User` má vlastnosť `understands`, ktorá odkazuje na témy, ktoré zvládol a teda im už rozumie. `Event`, `Material` a `Task` sú podtriedami triedy `TopicRelated`, pretože všetky sa môžu viazať na konkrétnu tému predikátmi `covers`, `requires` a `mentions`. Na obrázku 2.1 je vidieť zjednoduššený ontologický model Courses prezentovaný v článku [12].

Zároveň s touto bakalárskou prácou vznikajú aj ďalšie, ktoré sa venujú vývoju iných častí systému Courses, preto v tomto momente nie je možné pomocou už existujúcich zdrojov odprezentovať kompletný aktuálny ontologický model. Vždy aktuálna verzia systému je však dostupná na Githube v dvoch repozitároch [13] [14]. Jednotlivé časti systému sa vyvíjajú paralelne, a tak sa ontologické zmeny vykonávajú v nezávislých vetvách, neskôr sa zlučujú.

Ontológia systému Courses je uložená na backende v priečinku `src/model` v Javascripte vo forme objektov. Triedy zadefinované v `src/exporter` umožňujú export ontológie do rôznych formátov.

2.2 Databáza a získavanie dát

Courses uchováva svoje dáta v RDF databáze, využíva sa open-source verzia databázového servera Virtuoso. Databáza síce pracuje priamo so SPARQL dopytmi, no tie sú až posledným krokom pri práci s dátami, proces je zložitejší a pozostáva z viacerých postupných krokov. Jadrom je dopytovací jazyk GraphQL. V našom prípade ide o jeho implementáciu HyperGraphQL pre RDF dáta, no ešte konkrétnejšie systém Courses využíva rozšírenie UltraGraphQL. Pri spustení serveru sa vykoná extrakcia (pomocou tried definovaných v `src/exporter`), aby sa správne nastavila UltraGraphQL schéma. Podrobnejšie o týchto procesoch hovorí diplomová práca [15].

2.3 React frontend

Frontend aplikácie je implementovaný v Reacte [16], Javascript frameworku. Základnou stavebnou jednotkou frontendu je funkčný komponent, teda funkcia, ktorá vracia JSX. Ide o špeciálnu syntax, ktorá popisuje HTML v Javascripte bez nutnosti stáleho využívania znakových reťazcov. V priečinku `src/pages` sa nachádza jadro frontendu, každý modul má svoj priečinok a v ňom komponenty tvoriace užívateľské rozhranie. Komponenty sú pripravené tak, aby sa dali jednoducho využiť aj v iných moduloch.

2.4 Univerzálne používateľské rozhranie

V článku [12] je predstavný koncept univerzálneho grafového používateľského rozhrania. Proces jeho generovania pozostáva z niekoľkých častí: získať dáta a ich prepojenia z databázy, priradiť dátam štýlovanie na základe ich typov (tried), zadefinovať metódy interakcie na základe typu elementu, konkrétneho zobrazenia a používateľa, a nakoniec samotná vizualizácia pomocou vhodného vizualizačného nástroja. V článku je aj krátky príklad SPARQL dopytov, ktorý by mohli slúžiť na zobrazenie všetkých tém a ich prepojení a zároveň by zabezpečili rôzne štýlovanie pre témy, ktorým študent rozumie, a témy, ktoré ešte nemá zvládnuté. V ukážke 2.1 je vidno SPARQL dopyt, ktorý vráti nový graf pozostávajúci z tém, ktorým študent rozumie, týmto témam sa priradí trieda `CoveredTopic`. Táto trieda je iba príkladom v článku, v ontológii systému naozaj neexistuje. Výsledný graf tohto dopytu by sa potom operáciou disjunkcie zlúčil s grafom všetkých tém s prepojeniami, a tak by sme dostali témy, ktoré si so sebou nesú vlastné vizualizačné vlastnosti.

Ukážka 2.1: SPARQL dopyt pre univerzálne používateľské rozhranie

```
CONSTRUCT {
:t a :CoveredTopic
} WHERE {
:t a :Topic.
:courseXY :covers :t.
:studentYZ :understands :t
}
```

Kapitola 3

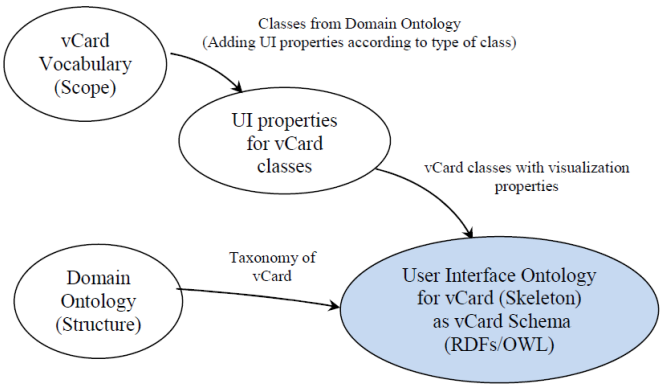
Grafové používateľské rozhrania

Keďže cieľom tejto práce je generovanie uživateľského rozhrania, ktoré bude priamo vychádzať zo sémantického modelu dát, v nasledujúcich podkapitolách zanalyzujeme relevantné existujúce riešenia podobných problémov a spôsoby vizualizácie grafových dát na základe ontológií.

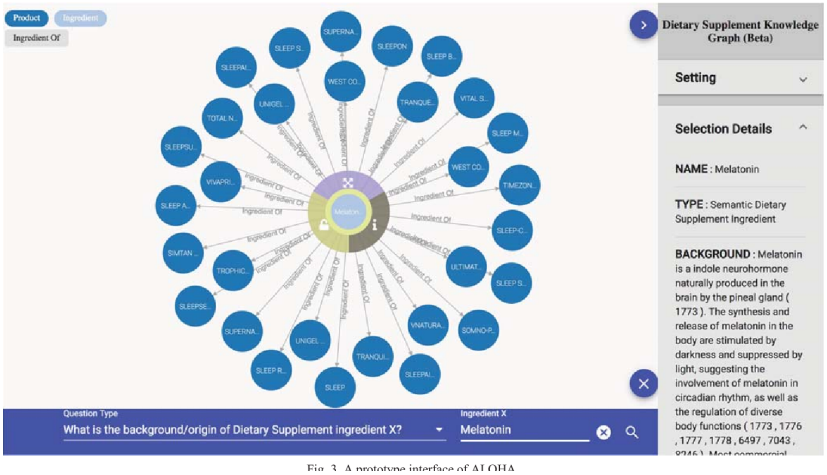
3.1 Používateľské rozhrania s ontologickým základom

Pre nás najrelevantnejší prínos k riešenému problému zachytil vo svojom článku Shahzad [17]. Hovorí o využití ontológií v rôznych vrstvách vývoja uživateľského rozhrania a popisuje proces mapovania ontológií priamo do grafického užívateľského rozhrania, aby tak boli zachované dôležité vlastnosti ontologických konceptov. Tento článok je však pomerne abstraktný, ponúka nám priestor na rozvoj pôvodnej myšlienky. Na obrázku 3.1 z pôvodného článku je vidno proces tvorby ontológie používateľského rozhrania (User Interface Ontology), pozostáva z obohatenia pôvodných tried doménovej ontológie o UI vlastnosti.

Autori článku [18] prezentujú svoju webovú aplikáciu ALOHA, ktorá slúži na vzde-



Obr. 3.1: Ontológia používateľského rozhrania podľa Shahzada



Obr. 3.2: Používateľské rozhranie v aplikácii ALOHA

lávanie sa o doplnkoch stravy. Jadrom je databáza vedomostí s ontologickým základom, iDISK, pre ktorú chceli vytvoriť používateľsky príjemného prieskumníka. Používateľ si vyberie jednu z možných otázok, do ktorej doplní konkrétny pojem, ktorý ho zaujíma, a výsledky vyhľadávania sa mu odprezentujú vizualizované v grafe. Toto používateľské rozhranie ponúka viaceré zaujímavé zlepšenia, napríklad po dvojkliku na konkrétny uzol sa graf rozrastie o informácie o danom uzle, ktoré boli dovtedy skryté, a farby jednotlivých uzlov reprezentujú triedy či vzťahy, ktorými sú uzly navzájom prepojené.

Relevantným projektom je aj automatické generovanie vizualizácií z dát na webe so základom v ontológiach [19]. Autori článku prezentujú svoj prototyp SemViz a popisujú, ako dokáže generovať vizualizácie hudobných rebríčkov. Toto riešenie ponúka vďaka využitiu teórie pravdepodobnosti omnoho silnejšiu automatizáciu, ako zaujíma nás, no zaujímavé je, že pracuje s dokopy troma ontológiami - doménovou, vizuálnou a tiež ontológiou, ktorá slúži na ich premostenie (semantic Bridging Ontology).

Články [17] a [19] majú spoločné to, že autori privedením ontológií do vizualizačného procesu dosahujú schopnosť vytvoriť rôzne druhy vizualizácií, stačí ich správne zadefinovať.

3.2 Vizuálizácie ontológií

Aktuálnym problémom síce nie je vizualizácia ontológií, ale konkrétnych dát, no pre požadované grafové užívateľské rozhranie vieme nabrať inšpiráciu aj z rôznych metód vizualizácie ontológií.

Dudáš a kol. [20] vo svojom článku popísali metódy a nástroje na vizualizáciu ontológií. Zaujímavá je časť o technikách na obohatenie vizualizácií, kde spomínajú napríklad približovanie, zameriavanie sa na konkrétny uzol či efekt rybieho oka (angl. fisheye distortion). Posledná z techník označuje priblíženie na vybratú entitu a oddialenie či

roznazanie okolitých, aby sa vizualizácia sprehľadnila, ale kontext zostal zachovaný.

Kapitola 4

Použité knižnice

4.1 Reactflow

Reactflow [21] je open-source knižnica na vytváranie interaktívnych užívateľských rozhraní z diagramov, konkrétne z uzlov a hrán, v Reacte. Existuje aj jej platená Pro verzia, neobsahuje nič navyše, ale umožňuje prístup k ďalším ukážkam a návodom. Pre edukačné účely je zadarmo. Preto sme aj pre tento projekt dostali možnosť využívať plnú verziu Reactflow.

Knižnica má základnú funkcionalitu už predpripravenú, približovanie, pohybovanie či označovanie uzlov je implementované, tieto a ďalšie interaktívne metódy možno vypnúť a zapnúť.

Ponúkané sú taktiež rôzne typy hrán a uzlov. Existujú `default`, `straight`, `step` a `smoothstep` hrany, je tiež možné vytvoriť si vlastné. To isté platí pre uzly, okrem predvolených je už zadefinovaný typ uzlu `group`. Uzly tohto typu môžu obsahovať ďalšie uzly a výhodou je, že poloha ich potomkov sa počíta relatívne k ich vlastnej. Predpokladá sa, že vývojár si vytvorí ďalšie typy uzlov podľa svojich konkrétnych potrieb. Je možné do nich vkladať rôzne tlačidlá, texty, či formulárové prvky, zadefinovať im vhodné štýlovanie a miesta, kde sa budú pripájať hrany.

4.2 React Flow Smart Edge

Reactflow Smart Edge [22] ponúka spôsob, ako v Reactflow vytvárať hrany, ktoré sa nepretínajú s uzlami. Knižnica ponúka už hotový nový typ pre hrany v troch verziách: `SmartBezierEdge`, `SmartStraightEdge` a `SmartStepEdge`, ktoré sa iba rovno importujú, sú pripravené na použitie. Dostupná je tiež funkcia `getSmartEdge`, ktorá po prijatí rovnakých props, ako sa posielajú do každej Custom Edge, vráti vhodnú SVG nekonfliktnú cestu (alebo vráti null, ak taká cesta neexistuje), výslednú hranu tak možno viac prispôsobiť konkrétnym potrebám. Do `getSmartEdge` sa môžu poslať

aj ďalšie nastavenia, napríklad hodnota `nodePadding`. Tá hovorí o tom, ako veľký má byť nedotknuteľný priestor, ktorý chceme zachovať okolo každého uzla.

4.3 Dagre.js

Dagre.js [23] slúži na vypočítanie polôh uzlov orientovaných grafov na strane klienta. Knižnica sa veľmi elegantne kombinuje s Reactflow, takže ešte pred vykreslením uzlov si môžeme dať vypočítať ich polohy. To zaručí metóda `layout()`. Voláme ju potom, ako sme do inštancie Dagre grafu vložili všetky uzly a hrany. Uzly potom získajú vypočítané hodnotu stredovej x a y súradnice.

4.4 rdflib.js

rdflib.js [24] je RDF knižnica pre Javascript. Ponúka nástroje na čítanie a zapisovanie rôznych RDF formátov. Dáta sa dajú prečítať zo správne formátovaného znakového reťazca alebo rovno z webu. Následne s nimi môžeme ďalej pracovať. Knižnica obsahuje napríklad tri užitočné metódy: `each()`, `any()` a `statementsMatching()`. Prvé dve dostanú ako argumenty dve hodnoty RDF trojice z troch, a vrátia také zdroje vo forme objektov, ktoré možno na tretie miesto dosadiť. Nedefinovanú hodnotu nazývajú ”wildcard”. Posledná metóda povoľuje aj viac wildcards, vráti pole všetkých trojíc, ktoré tomuto vzoru zodpovedajú.

Časť II

Riešenia

Kapitola 5

Návrh

5.1 Obohatenie dát o vizuálne vlastnosti

5.1.1 Vizuálna ontológia

5.1.2 Axiómy

5.1.3 Vyvodzovanie

5.2 Generovanie používateľského rozhrania

5.2.1 Prehľad technológií

5.2.2 Návrh vizualizácie

5.2.3 Interakcie

Kapitola 6

Implementácia

6.1 Používateľské rozhranie

6.1.1 Vizualizácia grafových uzlov

6.1.2 Rozloženie uzlov

6.1.3 Kontextové menu

6.1.4 Ďalšie interakcie

6.2 Vizualizácia dát

6.2.1 Parsery ontológií

6.2.2 Vyvodzovanie

6.2.3 Custom node

Záver

Literatúra

[1] Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004.

[2] Pascal Hitzler. A review of the semantic web field. *Communications of the ACM*, 64(2):76–83, 2021.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific american*, 284(5):34–43, 2001.

[4] Frank Manola, Eric Miller, Brian McBride, et al. Rdf primer. *W3C recommendation*, 10(1-107):6, 2004.

[5] Richard Cyganiak, David Hyland-Wood, and Markus Lanthaler. Rdf 1.1 concepts and abstract syntax. *W3C Proposed Recommendation*, 01 2014.

[6] World Wide Web Consortium et al. Sparql 1.1 overview. 2013.

[7] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? *Handbook on ontologies*, pages 1–17, 2009.

[8] Dan Brickley and R.V. Guha. Rdf schema 1.1. *W3C recommendation*, 2014.

[9] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, Sebastian Rudolph, et al. Owl 2 web ontology language primer. *W3C recommendation*, 27(1):123, 2009.

[10] Birte Glimm and Yevgeny Kazakov. Classical algorithms for reasoning and explanation in description logics. *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*, pages 1–64, 2019.

[11] Martin Homola, Ján Kl'uka, Zuzana Kubincová, Patrícia Marmanová, and Milan Cifra. Timing the adaptive learning process with events ontology. In *Advances in Web-Based Learning–ICWL 2019: 18th International Conference, Magdeburg, Germany, September 23–25, 2019, Proceedings 18*, pages 3–14. Springer, 2019.

[12] Martin Homola, Zuzana Kubincová, Rastislav Urbánek, and Ján Kl'uka. Tracking the adaptive learning process with topics ontology. In *International Conference on Web-Based Learning*, pages 155–165. Springer, 2023.

[13] matfyz sk. courses-frontend. https://github.com/matfyz-sk/courses-frontend.

[14] matfyz sk. courses-backend. https://github.com/matfyz-sk/courses-backend.

[15] Miroslav Baluch. Graph-based querying and mutations of linked data, 2023.

[16] React. https://react.dev/.

[17] Syed K Shahzad. Ontology-based user interface development: User experience elements pattern. *J. Univers. Comput. Sci.*, 17(7):1078–1088, 2011.

[18] Xing He, Rui Zhang, Rubina Rizvi, Jake Vasilakes, Xi Yang, Yi Guo, Zhe He, Mattia Prosperi, and Jiang Bian. Prototyping an interactive visualization of dietary supplement knowledge graph. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1649–1652, 2018.

[19] Owen Gilson, Nuno Silva, Phil W Grant, and Min Chen. From web data to visualization via ontology mapping. In *Computer graphics forum*, volume 27, pages 959–966. Wiley Online Library, 2008.

[20] Marek Dudáš, Steffen Lohmann, Vojtěch Svátek, and Dmitry Pavlov. Ontology visualization methods and tools: a survey of the state of the art. *The Knowledge Engineering Review*, 33:e10, 2018.

[21] xyflow. xyflow. https://github.com/xyflow/xyflow.

[22] Tiso Alvarez Puccinelli. react-flow-smart-edge. https://github.com/tisoap/react-flow-smart-edge.

[23] dagrejs. dagre. `https://github.com/dagrejs/dagre`.

[24] linkeddata. rdflib.js. https://github.com/linkeddata/rdflib.js.