Sem vložte zadání Vaší práce.

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

Magisterská práce

Průvodce studenta FIT ČVUT

Bc. Jan Molnár

Vedoucí práce: Ing. Michal Havryluk

13. května 2012

Poděkování

V první řadě bych rád poděkoval vedoucímu práce, inženýru Havrylukovi, za veškerou pomoc, kterou mi při tvorbě diplomové práce poskytl. V této souvislosti bych chtěl poděkovat i docentu Vitvarovi, že mi pomohl vytvořit zadání. Velké díky patří Kristýně Řehákové za podporu a provedení jazykové korektury práce. V neposlední řadě bych chtěl poděkovat rodičům za možnost ničím nerušeného studia a tvorby diplomové práce. Zvláštní dík patří testerům aplikace, autorům svobodných technologií, na kterých je práce založena, a autorům svobodných nástrojů, ve kterých byla vytvořena.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl 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 výplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České výsoké učení technické v Praze má právo na uzavření licenční smlouvý o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 13. května 2012 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Jan Molnár. *Průvodce studenta FIT ČVUT: Magisterská práce.* Praha: ČVUT v Praze, Fakulta informačních technologií, 2012.

Abstract

Master's thesis is focused on domain of information needed for studying at CTU, Faculty of Information Technology and on the students' access to this information. **TODO** Zrevidovat a dopřeložit abstrakty.

Keywords Guide, CTU FIT, Ontology, Mobile application.

Abstrakt

Diplomová práce se zabývá doménou informací potřebných pro studium na Fakultě informačních technologií ČVUT a přístupem studentů k těmto informacím. Cílem práce bylo vytvořit ontologii reprezentující danou doménu a demonstrovat její užití na mobilní aplikaci sloužící jako průvodce studenta. Dosaženo toho bylo serverovou aplikací implementovanou na *Node.js* shromažďující informace a poskytující je prostřednictvím SPARQL *endpointu* mobilní aplikaci implementované v *HTML5*. Práce se zaměřila na analýzu, návrh, realizaci i testování vzniklých aplikací.

Klíčová slova Průvodce, ČVUT FIT, ontologie, mobilní aplikace.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Když jsem se v květnu roku 2011 začal zabývat diplomovou prací, neměl jsem v tom, co budu dělat, vůbec jasno. Jedním z mála tehdejších cílů bylo vytvoření aplikace, ze které bude mít co největší počet lidí užitek. V rámci bakalářské práce jsem se zabýval doménou navigace studentů po Fakultě elektrotechnické ČVUT [8], proto jsem, mezi jinými tématy, zvažoval i pokračování v obdobné doméně na Fakultě informačních technologií – jedná se o oblast, ve které lze studentům přinést mnoho užitečného.

Nejprve jsem zkontaktoval pana docenta Vitvara, kterého téma oslovilo a začali jsme společně tvořit zadání diplomové práce. Dohodli jsme se na struktuře práce, použitých technologiích a některých dalších detailech. Mezi požadavky se objevila tvorba mobilní aplikace pro přístup k informacím, těmi se pan docent nezabývá, proto mi doporučil další pokračování práce pod panem inženýrem Havrylukem, jehož jsou primární doménou. Této nabídky jsem využil.

Shodná doména bakalářské a diplomové práce se potvrdila být prospěšnou – ačkoliv jsem z bakalářské práce v diplomové mimo získaného přehledu o doméně nakonec nic jiného nevyužil, právě ten se ukázal být jako velmi přínosný – od začátku jsem měl přehled o problematických místech a těch, které tehdy nebylo možné realizovat, a mohl se na ně zaměřit a vyřešit tentokrát podstatně lépe. Bakalářská práce se zabývala pouze navigací do určitého bodu, diplomová jde podstatně více do hloubky i do šířky a přináší celého průvodce postaveného na solidních základech.

KAPITOLA 1

Popis problému, specifikace cíle

Tato kapitola se zabývá problematikou zpracovávanou diplomovou prací, kontextem, do kterého je zasazena, rozvržením práce a končí rešerší již zpracovaných obdobných témat.

1.1 Popis řešeného problému

Ačkoliv máme možnost čerpat z mnoha zdrojů, ne vždy nalezneme to, co hledáme. Důvodů je samozřejmě mnoho: Známe velmi omezenou množinu zdrojů, neumíme se v nich zorientovat, nedokážeme je pro jejich komplexitu zpracovat, nevíme, které zdroje jsou ověřené, aktuální... Nic není samozřejmě ztraceno – může se tu objevit někdo, kdo je zmapuje, zagreguje, přetřídí, ohodnotí a pod jednotným rozhraním poskytne. Přesně tento úkol – umožnit studentům snadný přístup k informacím – je cílem diplomové práce.

Fakulta informačních technologií Českého vysokého učení technického v Praze, stejně tak, jako většina ostatních obdobných institucí, nabízí studentům nepřeberné množství informací. Některé se nacházejí na webové stránce fakulty (`http://fit.cvut.cz/`), univerzity (`http://www.cvut.cz/`), jiné na serverech edux (`https://edux.fit.cvut.cz/`) a komponenta studia (`https://kos.cvut.cz/`), další pak pod Správou účelových zařízení (`http://agata.suz.cvut.cz/jidelnicky/`) nebo třeba na webu Centra informačních a poradenských služeb ČVUT (`http://www.cips.cvut.cz/`).

Zorientovat se ve zdrojích nějakou dobu trvá, v těch nejdůležitějších, zpravidla ne markantní, přesto existuje mnoho zdrojů, používaných méně často, na které nemusí potencionální uživatel vůbec narazit – jedním z nich je třeba seznam akcí ČVUT (`http://akce.cvut.cz/`). Velmi by proto pomohl nějaký průvodce, který by dostupné informace seskupil a koherentně poskytl.

1.2 Popis uživatele a jeho potřeb

Práce se zaměřuje především na studenty Fakulty informačních technologií Českého vysokého učení technického v Praze, přesto z ní mohou mít užitek i ostatní zainteresované strany – vyučující, neakademičtí pracovníci, zájemci o studium, návštěvníci... Kdokoliv pohybující se po dejvickém kampusu nebo hledající nějakou informaci vztaženou k fakultě.

Cílová skupina uživatelů chce zpravidla všechno a hned, nezabývá se proto rozsáhlými předpisy / návody a k informacím se buď dostane během krátké doby z nějakého vhodného zdroje, nebo vůbec. Vhodným zdrojem může být i to, co by pro většinovou populaci vhodné nebylo – studenti Fakulty informačních technologií mají nadprůměrně rozsáhlé znalosti v oblasti informačních technologií, proto může být vhodným zdrojem i komputerizované řešení.

K předchozí potřebě se váží i nároky na použitelnost výsledného řešení – ačkoliv jsou studenti informačních technologií na ledacos zvyklí a zvládnou tak využívat i nestandardní nástroje, mají na druhou stranu profesionální odpor k vyloženě špatným řešením. Na přístupnost není třeba brát u této cílové skupiny zřetel.

Studenti zjišťují informace často až na poslední chvíli – například umístění učebny, ve které mají výuku, zjišťují až několik minut před jejím zahájením. V té době nemají možnost využít plnohodnotný počítač, ale musejí se spolehnout na omezené mobilní zařízení, které zpravidla mívají u sebe.

1.3 Vymezení cílů diplomové práce

Cílem diplomové práce je vytvořit průvodce sloužícího studentům jako jednotné rozhraní pro přístup k nejčastěji používaným / nejdůležitějším informacím a usnadnit jim tak určitou – nestudijní – oblast studentského života. Práci je vhodné rozdělit do dvou částí – mobilní aplikaci, kterou budou studenti využívat, a serverovou aplikaci, která bude sloužit jako univerzální zdroj dat, i pro mobilní aplikaci.[1]

Před započetím implementace je nutné přibližně určit oblasti a rozsahy informací obsažených v průvodci. Následně je potřeba k požadovaným informacím nalézt vhodné zdroje. Neexistující zdroje je třeba nahradit simulovanými zdroji. Přichází na řadu analýza, návrh a realizace nejprve serverové, poté mobilní aplikace. Testování je vhodné provádět po celou dobu.

Mobilní aplikace by měla být dostupná velkému počtu studentů, měla by tedy být multiplatformní a fungovat (jak plyne z označení) i na mobilních

[1] Pojmy *mobilní* a *serverová aplikace* se vyskytují napříč celou prací, až na výjimky všude reprezentují aplikaci sloužící studentům jako průvodce – *mobilní aplikace* a aplikaci shromažďující data z nestrukturovaných zdrojů a poskytujících je unifikovaným způsobem – *serverová aplikace*. V některých případech pojem *serverová aplikace* značí pro nenalezení lepšího označení pouze jednu z komponent tohoto dříve ustanoveného pojmu, význam je z kontextu zřejmý.

zařízeních. Využití aplikace se předpokládá až v následujících letech, není tedy třeba podchytávat stará a méně schopná zařízení – s jejich podporou se již nepočítá. Specifickým zdrojem poskytovaných informací by měla být navigace.

Serverová aplikace by měla být přístupná pouze administrátorům a uživatelům by sloužilo čistě jen vhodné rozhraní pro získávání požadovaných informací. Informace mají být v systému uložené dle, na systému nezávislého, vhodného modelu.

Práce nemá za cíl plně obsáhnout všechny zpracovávané domény, tím je kvalitní zpracování základu každé domény, které pak lze dle již hotových částí doplnit do celého, pro praktické nasazení potřebného, rozsahu. Postačuje tedy vytvoření částí odpovídajících reprezentativnímu vzorku.

1.4 Popis struktury DP ve vztahu k vytyčeným cílům

Diplomová práce má být členěna do tří hlavních částí:

- Analýza s návrhem.
- Realizace.
- Testování.

Chronologické uspořádání částí je o trochu komplikovanější – začíná se analýzou, která přechází v návrh, po dokončení se vše realizuje, po celou dobu se průběžně testuje. Po ukončení této iterace se začíná nová, se širším záběrem. Tento postup se opakuje až do vytvoření aplikace.

Ve fázi analýzy a návrhu dochází nejprve k průzkumu zpracovávané domény, vytvoření uživatelských scénářů a vytyčení požadavků na vytvářenou aplikaci, později je navržena struktura aplikace, navržena bezpečnost a jsou voleny použité technologie.

Realizace by měla být pouhým mechanickým převedením navrženého systému do reálného světa, bohužel se v praxi vyskytují neočekávané problémy, které je třeba během realizace vyřešit. Vzhledem k akademickému účelu této práce je vhodné uvést obě řešení – nakonec implementované i to zamýšlené ideální.

Testování je třeba provádět na mnoha úrovních – od kontroly zdrojového kódu, přes integrační testování, testování systémové integrace, zátěžové testování, až po testování použitelnosti.

1.5 Serverová aplikace

Systém, který se skrývá pod pojmem *serverová aplikace* (viz vysvětlení v poznámce 1 na straně 20), je velmi specifický a proto pravděpodobně nemá

odpovídající obdoby. Jeho kruciální část, model reprezentující informace potřebné pro průvodce, ale obdob má velmi mnoho, proto byla rešerše prováděna hlavně v této oblasti.

1.5.1 Rešeršní zpracování existujících ontologií

Doposud bylo vytvořeno veliké množství univerzitních ontologií – jedná se totiž o oblíbené téma domácích úkolů předmětů zabývajících se sémantickým webem, z toho ale plyne i druhá častá vlastnost těchto ontologií – nebývají příliš propracované a odzkoušené. Kvalitních ontologií je tedy pouze malý zlomek.

1.5.1.1 Univ-Bench

Univ-Bench (`http://swat.cse.lehigh.edu/projects/lubm/`) je, pravděpodobně, nejznámější ontologií modelující univerzitu – jedná se totiž o ontologii využitou v The Lehigh University Benchmark (LUBM) – nástroji pro srovnávací testování úložišť sémantických dat [5]. Ontologie se snaží být co nejrealističtější, vzhledem ke svému účelu se ale nemůže dále rozvíjet. Autorem je Zhengxiang Pan z Lehigh University. Aktuální verze pochází z roku 2004.

1.5.1.2 Academic Institution Internal Structure Ontology

Academic Institution Internal Structure Ontology (AIISO) (`http://purl.org/vocab/aiiso/schema`) je, pravděpodobně, nejpropracovanější ontologií modelující univerzitu. Ve spojení s Participation (`http://purl.org/vocab/participation/schema`), Friend of a friend (FOAF) (`http://xmlns.com/foaf/0.1/`) a AIISO Roles (`http://purl.org/vocab/aiiso-roles/schema`) popisuje téměř celou doménu univerzity [12]. Ontologie je uvolněna pod licencí Creative Commons (attribute). Autory jsou Rob Styles a Nadeem Shabir z Talis Information Ltd. Aktuální verze pochází z roku 2008.

1.5.1.3 Higher Education Reference Ontology

Higher Education Reference Ontology (HERO) (`http://sourceforge.net/projects/heronto/`) je další ontologií mající za cíl pokrýt celou doménu jakékoliv univerzity. Ontologie je uvolněna pod Adaptive Public License. Autorkou je Leila Zemmouchi-Ghomari. Aktuální betaverze, aktivně vyvíjená, pochází z roku 2012 [18].

1.5.1.4 University Ontology

University Ontology (`http://purl.org/weso/uni`) je univerzitní ontologií vycházející z AIISO, FOAF a An organization ontology (Org). Je uvolněna pod licencí Creative Commons (attribute). Autorem je Jose Emilio Labra Gayo z

výzkumné skupiny WESO na Universidad de Oviedo. Aktuální verze pochází z roku 2011 [4].

1.5.1.5 University Ontology

University Ontology (`http://www.cs.umd.edu/projects/plus/SHOE/onts/univ1.0.html`) náleží do projektu Simple HTML Ontology Extensions (SHOE), který má poněkud širší záběr. Autorem je Jeff Heflin z Lehigh University (SHOE, jako celek, ale patří pod katedru informatiky na University of Maryland). Aktuální verze pochází z roku 2000 a v současné době už není udržovaná [6].

1.6 Mobilní aplikace

Na poli mobilních aplikací sloužících jako průvodce studenta je konkurence podstatně větší, než tomu bylo u serverové. Mnoho jich je dokonce napojených na externí zdroj dat, žádná z objevených ale nepracuje přímo se sémantickým úložištěm.

1.6.1 Rešeršní zpracování existujících mobilních aplikací

Práce zpracovávanou doménou navazuje na bakalářskou práci *Mobilní navigační systém pro FEL ČVUT* [8], ta ale není jediným zdrojem, který byl prozkoumán. Ostatní uvedené aplikace zpravidla cílí jinými směry, proto jim nebude věnováno tolik prostoru.

Jako důsledek relativně nedávného prudkého vzestupu počtu chytrých telefonů se za uplynulé roky objevilo mnoho aplikací zabývajících se podobnými tématy, jako tato práce.[2] Ne vždy se jedná o průvodce studentů po univerzitě, problém ale řeší obdobný. Nabídka jejich funkcí bývá vzájemně podobná, proto zde uvedu pouze výběr.

V první části této sekce se budu věnovat pracím obdobným této, později se budou objevovat práce stále více odlišné, přesto ale v některých ohledech inspirativní.

1.6.1.1 Mobilní navigační systém pro FEL ČVUT

Bakalářská práce *Mobilní navigační systém pro FEL ČVUT* [8] je zaměřená především na multiplatformnost a použitelnost. Bylo tedy nutné v některých oblastech (včetně funkcionality) podstoupit určité kompromisy. Aplikace vyhledá místnost na základě zadaného označení a přehledným plánem k ní uživatele navede. Označením může být oficiální název, přezdívka, jméno sídlícího pracovníka, či jiný identifikátor. Dále aplikace vizualizuje body zájmu, jako

[2] Zpracování těchto aplikací bývá ovšem jiné, jsou zaměřené na užitek, zatímco sofistikovanost zpracování zpravidla nehraje roli.

jsou občerstvení, výtahy, toalety... Práce poskytuje užitečného pomocníka, po čistě programové stránce ale moc zajímavá není.[3]

1.6.1.2 Průvodce ČVUT

Nyní již zastaralá mobilní aplikace *Průvodce ČVUT FEL* (`http://lr.czechian.net/j2me/`[4]) vytvořená v J2ME poskytovala vyhledávání místností s následným textovým popisem cesty, případným minimalistickým plánkem a dopravním spojením z frekventovaných lokalit.

1.6.1.3 ČVUT navigátor

Projekt *ČVUT navigátor* (`http://navigator.fit.cvut.cz/`) je tématem několika akademických prací. Vypadá slibně, ale ani v době dokončování této práce není známo více, než cíle vytyčené v době založení projektu.

1.6.1.4 Ostatní průvodci ČVUT

Mezi další, v souvislosti s touto prací méně významné, průvodce patří tištěný *Průvodce prváka po ČVUT* [2] nebo například specializovaný *Průvodce prváka po koleji Podolí* (`http://pruvodce.pod.cvut.cz/`).

1.6.1.5 Vlastní průvodci

Mobilní průvodci univerzitami, tak jako se děje i v jiných oblastech mobilních aplikací, se v poslední době začínají velmi rozmáhat. Velké množství univerzit nabízí svým studentům na vlastní půdě vytvořené aplikace, ať již studenty, nebo výpočetními centry. Tyto aplikace bývají šité na míru potřebám univerzit, na druhou stranu ale nemusí být tolik odladěné, jako to bývá u větších projektů. Nabídka funkcionalit sahá od minimalistických až po velmi rozsáhlé. S podporovanými platformami je to podobné – od jedné až po univerzální aplikace.

1.6.1.6 CampusM

Univerzity jako Imperial College London, London School of Economics, University College London, Trinity College Dublin a pár desítek dalších využívají komerční aplikace z dílen společnosti *campusM* (`http://www.campusm.com/`).

[3] Cílem této práce je naopak vytvoření robustní ontologie a samotná mobilní aplikace slouží pouze jako ukázka implementace za využití moderních mobilních technologií, takže ke kompromisům dojde v rozdílných oblastech. Návaznost na bakalářskou práci se bude projevovat převážně v poučení se z předchozích chyb ve shodné zpracovávané doméně.

[4] Aplikace *Průvodce ČVUT FEL* byla, pravděpodobně kvůli neaktualizovanému obsahu a celkovému nedokončení, stažena a nyní již není veřejně dostupná. Nachází se na přiloženém Compact Disc (CD) v adresáři *data/Průvodce ČVUT FEL*.

Řešení je nabízeno pro různé mobilní platformy a nabízí následující funkcionality:

- Vyhledání cesty.
- Lokalizace přátel.
- Novinky a události.
- Upozornění studentů.
- Integrace se systémem knihovny.
- Integrace s univerzitním adresářem.
- Přístup k rozvrhům.
- Informace o předmětech.

1.6.1.7 Guidebook

Guidebook (`http://guidebook.com/`) je typickým představitelem aplikace pro podporu určité události v širším slova smyslu – může sem spadat i průvodce školou. Ve webovém rozhraní pořadatel událost vytvoří a upraví podle specifických potřeb, na základě toho jsou vytvořeny aplikace pro různé mobilní platformy a ty pak jsou nabízeny návštěvníkům dané události. *guidebook* nabízí obdobné funkcionality výše uvedeným a je kompletně zdarma pro omezený počet uživatelů. Aplikace je pro diplomovou práci inspirativní záběrem a profesionalitou zpracování.

1.6.1.8 Ostatní, neuniverzitní, průvodci

V této široce zaměřené podpodsekci zmíním hlavně průvodce z jednotlivých akcí – ti totiž představují nejmohutnější podmnožinu. V poslední době se stává, hlavně na technicky zaměřených odborných konferencích,[5] zvykem poskytnout účastníkům mobilní aplikaci disponující informacemi o konferenci – navozuje to profesionální dojem, přináší více možností (ankety, upomínky...) a jsou tu mimo jiné i taková pozitiva, jako možnost aktualizace v případě nenadálé události. Další mobilní průvodci se vyskytují například na hudebních festivalech nebo po městech.

[5]Na těchto konferencích má většina účastníků u sebe zařízení schopné s touto aplikací pracovat.

1.6.1.8.1 Příklady funkcionalit: Žádná z nalezených aplikací nenabízí všechny níže uvedené funkcionality, poměrně velká část se k tomu ale blíží.

- Registrace účastníků.
- Zobrazování harmonogramu.
- Sestavení vlastního programu s upomínkami.
- Zobrazování plánů.
- Zobrazování informací o přednášejících, vystavovatelích, účinkujících...
- Zobrazování doplňujících materiálů (slidy a podrobnosti k přednáškám, texty písní u koncertů...).
- Interakce se sociálními sítěmi.
- Získávání zpětné odezvy (otázky přednášejícím, ankety...).
- Vytvoření prostoru pro reklamu sponzorů.

1.6.1.8.2 Příklady aplikací: Po úvaze jsem se rozhodl odkázat pouze na několik reprezentantů – aplikací je velmi mnoho a vzhledem k podobnosti by nemělo přínos uvádět všechny. Jako ukázku předkládám jednu svobodnou a několik komerčních aplikací:

iosched (`http://code.google.com/p/iosched/`) je svobodnou aplikací vyvíjenou pro konferenci Google I/O. Nabízí všechny výše uvedené funkcionality vyjma: registrace, sestavování vlastního programu a prostoru pro reklamu, což má svá opodstatnění. Aplikace je napsána v Javě a je určena pro běh v mobilních telefonech se systémem Android. *iosched* je pro diplomovou práci inspirativní záběrem zpracování a možností poučení se z otevřeného zdrojového kódu.

Google Maps Floor Plans (`http://maps.google.com/help/maps/floorplans/`) je dalším, v tomto případě již nesvobodným, nástrojem společnosti Google poskytujícím veřejnosti možnost tvorby plánů vnitřních prostor budov. Tyto plány budou v budoucnu – *Google Maps Floor Plans* je stále ve vývoji – zaneseny do mapových podkladů *Google Maps for Android*. Aplikace je tedy pro diplomovou práci inspirativní hlavně v oblasti navigace uvnitř budov.

KAPITOLA 2

Analýza a návrh implementace

Analýza s návrhem probíhaly nezávisle na dvou místech – po krátké analýze celku a určení vzájemných komunikačních rozhraní byly provedeny analýza s návrhem serverové aplikace a až po jejich dokončení začaly analýza s návrhem mobilní aplikace. V průběhu času jsem se k analýze a návrhu ještě několikrát vrátil a vylepšil / rozšířil to, co se ne vždy na poprvé povedlo.

Analýza se skládala z identifikace potřeb uživatelů, vytvoření případů užití, ze kterých byly odvozeny funkční požadavky, ke kterým byly doplněny nefunkční požadavky. Poté byl vytvořen návrh řešení, zapracován návrh bezpečnosti a celý koncept byl podložen návrhem nasazení. Nakonec byly voleny technologie.

2.1 Serverová aplikace

Začal jsem pracovat nejprve na serverové aplikaci – jednak na ní měla být ta mobilní závislá (sama na žádné jiné části práce závislá není) a také má užší vazbu ke zpracovávané doméně.

2.1.1 Potřeby uživatelů

Tato část se věnuje potřebám uživatelů serverové aplikace, zejména uživatelů rozhraní, přes které poskytuje informace. Potřeb je mnoho – proto zde uvedu pouze reprezentativní vzorek.

2.1.1.1 Potřeba zjistit informaci

Potřeba zjistit informaci je základní potřebou, kvůli které se mají uživatelé obracet na systém. Tato potřeba může vzniknout z velikého množství příčin – od hledání informací spjatých s průchodem studiem, až například po zjišťování jídelního lístku v menze.

2.1.1.2 Potřeba ověřit informaci

Často je cílová skupina uživatelů v situaci, kdy má požadovanou informaci, není si jí ale jistá a potřebuje ji ověřit / vyvrátit. Bez jistoty pravdivosti informace je její hodnota podstatně nižší.

2.1.1.3 Potřeba ověřených informací

Velmi podstatnou potřebou je potřeba ověřených informací. Ačkoliv je název velmi podobný té předchozí, jedná se o dvě diametrálně odlišné potřeby, předchozí se bez naplnění této neobejde. Jakmile jsou informace poskytovány, je nutné, aby byly ověřené – pravdivé.

2.1.1.4 Potřeba aktuálních informací

Potřeba aktuálních informací opět souvisí s již zmíněnými potřebami – sebepravdivější jídelní lístek z předchozího dne mi při výběru dnešního oběda bohužel moc nepomůže.

2.1.1.5 Potřeba kompletních informací

Někdy nestačí jen pravdivé a aktuální informace, naopak vědomí záruky pravdivosti a aktuálnosti může i uškodit – pokud totiž informace nejsou kompletní, může dojít k jejich misinterpretaci a pod vlivem záruk nedojde ke kritickému posouzení. Proto je zde i potřeba kompletních informací.

2.1.1.6 Potřeba nalézt, co nehledám

Na první pohled tato potřeba nedává smysl, v praxi se ale často vyskytují informace, které bychom měli znát a o jejich existenci nemáme ani ponětí. Je proto důležité ve vhodné chvíli sdělit i nepožadované informace, které jsou s velkou pravděpodobností pro adresáta důležité.

2.1.2 Případy užití

Případy užití (*use cases*) jsou takové sekvence kroků, které reprezentují danou činnost prováděnou aktérem na systému a vedou od jejího zahájení až po naplnění.

V serverové aplikaci byli identifikováni celkem čtyři hlavní aktéři:

- Uživatel – student / softwarová komponenta (*mobilní aplikace*).
- Správce – administrativní pracovník fakulty.
- Plánovač – softwarová komponenta.
- Zdroj – server poskytující informace.

Roli správce by šlo dále dělit na specifičtější role, vzhledem k velmi nízkému rozsahu činností každé z nich (zpravidla jedna činnost), je vše zahrnuto pod tímto souhrnným názvem. Pod plánovačem je zase zahrnut aktér čas.

Uživatel se pouze dotazuje nad systémem a zjišťuje informace. Správce manuálně upravuje informace v systému, může vynutit aktualizaci a jinak spravuje systém. Plánovač automaticky obsluhuje aktualizace obsažených informací. Zdroj na vyžádání poskytne dané informace.

2.1.2.1 UC-S-1: Vyžádání informací

Aktéři:

- Uživatel.

Systém:

- Serverová aplikace.

Vstupní podmínky:

- Uživatel zná jazyk dotazovacího rozhraní systému.
- Uživatel má přístup k dotazovacímu rozhraní systému.

Hlavní scénář:

- Uživatel vytvoří dotaz.
- Uživatel odešle dotaz na rozhraní systému.
- Systém přijme dotaz.
- Systém zpracuje dotaz – nalezne požadované informace.
- Systém odešle odpověď.
- Uživatel přijme odpověď.
- Uživatel zpracuje odpověď.

Alternativní scénáře:

- V případě neplatného dotazu je navráceno chybové hlášení.
- V případě chyby serveru je navráceno chybové hlášení.

Poznámky:

- V případě nenalezení informace je navrácena prázdná množina.
- Pokud je uživatelem softwarová komponenta, student ji obsluhující nemusí znát jazyk dotazovacího rozhraní – může s komponentou komunikovat jiným způsobem a ta si požadavky do požadovaného jazyka převádět.

2.1.2.2 UC-S-2: Aktualizace informací

Aktéři:

- Plánovač.
- Zdroj.

Systém:

- Serverová aplikace.

Vstupní podmínky:

- Plánovač je řádně nastaven a spuštěn.
- Nastal čas aktualizace určitých informací.

Hlavní scénář:

- Plánovač zadá systému požadavek na aktualizaci určitých informací.
- Systém přijme požadavek.
- Systém dá zdroji požadavek na informace.
- Zdroj zpracuje požadavek a informace vrátí.
- Systém přijme informace.
- Systém zpracuje informace.
- Systém odešle plánovači potvrzení.

Alternativní scénáře:

- V případě chyby zdroje je navráceno chybové hlášení.
- V případě chyby zpracování informací je navráceno chybové hlášení.

Poznámky:

- Případné chyby jsou zaznamenány do logu.
- Aktualizaci může obdobným způsobem vynutit i správce.
- Zdrojů určitých informací může být i více a vzájemně se mohou doplňovat / ověřovat.

2.1.2.3 UC-S-3: Správa informací

Aktéři:

- Správce.

Systém:

- Serverová aplikace.

Vstupní podmínky:

- Správce má přístup do systému.
- Správce zná jazyk rozhraní správy systému.

Hlavní scénář:

- Správce vytvoří dotaz.
- Správce se připojí do systému.
- Správce odešle dotaz.
- Systém přijme dotaz.
- Systém zpracuje dotaz.
- Systém odešle odpověď.
- Správce přijme odpověď.
- Správce se odpojí ze systému.
- Správce zpracuje odpověď.

Poznámky:

- Případné chyby jsou zaznamenány do logu.

2.1.3 Požadavky

2.1.3.1 Funkční požadavky

Funkčními požadavky (*functional requirements*) jsou myšleny takové požadavky, které jsou zaměřeny na jednotlivé funkcionality systému. Ne všechny požadavky budou nakonec implementovány, pouze jejich reprezentativní vzorek, dle kterého bude možné systém o ty ostatní rozšířit. Pro serverovou aplikaci mezi ně patří především:

1. Aplikace bude čerpat informace o názvech budov z předstíraného adaptéru.[6]

2. Aplikace bude čerpat informace o umístění budov z předstíraného adaptéru.[7]

3. Aplikace bude čerpat informace o názvech místností z předstíraného adaptéru.[8]

4. Aplikace bude čerpat informace o umístění místností z předstíraného adaptéru.[9]

5. Aplikace bude čerpat seznam vyučujících z KOSapi (`http://kosapi.fit.cvut.cz/`).[10]

6. Aplikace bude čerpat seznam studentů z KOSapi.[11]

7. Aplikace bude čerpat seznam neakademických pracovníků z předstíraného adaptéru.

8. Aplikace bude čerpat informace o rozvrzích místností z KOSapi.[12]

9. Aplikace bude čerpat informace o rozvrzích vyučujících z KOSapi.[13]

[6] Vynecháno. Implementováno na straně *mobilní aplikace*.
[7] Vynecháno. Implementováno na straně *mobilní aplikace*.
[8] Vynecháno. Implementováno na straně *mobilní aplikace*.
[9] Vynecháno. Implementováno na straně *mobilní aplikace*.
[10] Nahrazen zdroj za předstíraný adaptér – žádný seznam vyučujících obsahující identifikátory (uživatelská jména) nebyl nalezen.
[11] Nahrazen zdroj za předstíraný adaptér – z důvodů konzistence se zdrojem vyučujících.
[12] Vynecháno – projektu KOSapi nebyl dosud umožněn přístup do komponenty studia. Implementováno na straně *mobilní aplikace*.
[13] Vynecháno – projektu KOSapi nebyl dosud umožněn přístup do komponenty studia. Implementováno na straně *mobilní aplikace*.

10. Aplikace bude čerpat informace o kancelářích vyučujících z Usermap ČVUT (`http://usermap.cvut.cz/`).

11. Aplikace bude čerpat informace o kontaktech vyučujících z Usermap ČVUT.

12. Aplikace bude čerpat informace o rozvrzích studentů z KOSapi.[14]

13. Aplikace bude čerpat informace o kontaktech studentů z Usermap ČVUT.

14. Aplikace bude čerpat informace o kancelářích neakademických pracovníků z Usermap ČVUT.

15. Aplikace bude čerpat informace o kontaktech neakademických pracovníků z Usermap ČVUT.

16. Aplikace bude čerpat informace o akcích ČVUT z Kalendáře akcí ČVUT (`http://akce.cvut.cz/`).

17. Aplikace bude čerpat informace o harmonogramu akademického roku z předstíraného adaptéru (na základě `http://fit.cvut.cz/`).[15]

18. Aplikace bude čerpat informace o datech MAR budovy T9.[16]

19. Aplikace bude čerpat informace o datech MAR kolejí Orlík.[17]

20. Aplikace bude čerpat informace o datech MAR Masarykovy koleje.[18]

21. Aplikace bude čerpat informace o jídelníčcích menz z Jídelníčků menz (`http://agata.suz.cvut.cz/jidelnicky/`).

22. Aplikace bude čerpat informace o otvírací době menz z Jídelníčků menz.[19]

23. Aplikace bude čerpat informace o otvírací době NTK z Národní technické knihovny (`http://www.techlib.cz/`).[20]

24. Aplikace bude čerpat informace o otvírací době Vydavatelství průkazů (`http://ke.customer.decent.cz/a021/mon/wc-mon.php?co=2`).[21]

[14] Nahrazen zdroj za libovolný zdroj ve formátu iCalendar – projektu KOSapi nebyl dosud umožněn přístup do komponenty studia.

[15] Vynecháno. Implementováno na straně *mobilní aplikace*.

[16] Jednání o udělení přístupu nedopadla – vyskytly se blíže nespecifikované technické komplikace na straně technickoprovozních služeb Fakulty architektury.

[17] Vynecháno. Implementováno na straně *mobilní aplikace*.

[18] Vynecháno – data poskytována pouze pod licencí, kterou nebylo možné naplnit.

[19] Vynecháno. Implementováno na straně *mobilní aplikace*.

[20] Vynecháno. Implementováno na straně *mobilní aplikace*.

[21] Vynecháno. Implementováno na straně *mobilní aplikace*.

25. Aplikace bude čerpat informace o počtu lidí ve frontě ve Vydavatelství průkazů (`http://ke.customer.decent.cz/a021/mon/`).[22]

26. Vytvořte / nalezněte / sestavte z dílčích částí ontologii reprezentující výše uvedené informace.

27. Informace získané z jednotlivých zdrojů převádějte dle výše požadované ontologie pro další zpracování do RDF.

28. Aplikace umožní automatické aktualizace informací.

29. Aplikace umožní manuální aktualizace informací.

30. Aplikace umožní manuální úpravy informací.

31. Aplikace bude ukládat sémantické informace do databáze.

32. Aplikace umožní vykonávat dotazy nad uloženými daty.

2.1.3.2 Nefunkční požadavky

Nefunkčními požadavky (*non-functional requirements*) jsou myšleny takové požadavky, které jsou zaměřeny na dílo jako celek, nikoliv na jednotlivé funkcionality. Pro serverovou aplikaci mezi ně patří především:

1. Aplikaci naprogramujte v jazyce JavaScript.

2. Pro běh serveru využijte Node.js.

3. Pro dotazování využijte dotazovací jazyk SPARQL.

4. Sémantická data ukládejte do takového úložiště, aby nad ním bylo možné vykonávat dotazy přímo (případně přes nezávislého prostředníka) a nebylo nutné přistupovat přes vlastní aplikaci.

5. Práce nemusí nutně obsáhnout celou zpracovávanou doménu, je ale žádané, aby kvalitně zpracovala hlavní scénáře využití a nebránila dalšímu rozšiřování.

6. Zveřejněte aplikaci a její data pod svobodnými licencemi z rodin GNU, Creative Commons a kompatibilními, aby byl umožněn další rozvoj nezávislý na původním autorovi.

[22] Vynecháno.

2.1.4 Návrh ontologie

Tvorbě ontologie bylo věnováno velmi dlouhé období – neměl jsem s tímto oborem předchozí zkušenosti a vytvoření kvalitní ontologie považuji jako nutný předpoklad pro následné pokračování – na špatných základech nelze vybudovat kvalitní systém.

Zpočátku jsem vytvářel ontologii od základů vlastní, s přibývajícími zkušenostmi jsem si ale uvědomil, že to nebyl dobrý nápad – ve světě sémantického webu je jednou z nejdůležitějších priorit vzájemná interoperabilita – s každou novou ontologií je tak vzájemné mapování informací čím dál tím těžší. V té době jsem veškerou svou dosavadní práci zahodil a začal znovu.

Nejprve jsem se vrátil k výsledkům rešeršního zpracování ontologií a vybral z nich tu, která se mi zdála po v mezičase nabitých zkušenostech nejlepší – AIISO. Tato ontologie je již předurčená k využití s ontologiemi Participation, FOAF a AIISO Roles. Ontologie jsou to již zaběhlé, takže problém s interoperabilitou pozbyl platnosti.

Dalším bodem zvratu bylo zjištění, jak složité může být nalezení styčných bodů mezi obecnou ontologií a konkrétní situací. Bylo třeba v některých případech nalézt další vhodné ontologie reprezentující doposud nepokryté oblasti, případně již použité ontologie rozšířit (zpravidla je použita licence Creative Commons, takže je rozšiřování možné). Příkladem může být reprezentace akcí ČVUT – žádná z použitých ontologií nemodeluje události. Abych se držel kurzu ve směru interoperability, bylo třeba nalézt vhodnou ontologii. Nabízelo se jich několik, nakonec byla použita Linking Open Descriptions of Events (LODE), právě kvůli svému zastřešení obdobných ontologií.

Následuje seznam zvolených ontologií, pro specifické případy se ale počítá s jeho dalším rozšiřováním:

- AIISO (`http://purl.org/vocab/aiiso/schema`).
- Participation (`http://purl.org/vocab/participation/schema`).
- FOAF (`http://xmlns.com/foaf/0.1/`).
- AIISO Roles (`http://purl.org/vocab/aiiso-roles/schema`).
- LODE (`http://linkedevents.org/ontology/`).
- OWL-Time (`http://www.w3.org/TR/owl-time/`).

2.1.5 Návrh architektury

Nejprve byl vytvořen návrh architektury (*architecture design*) aplikace, který byl později rozpracován do návrhu nasazení (*deployment design*).

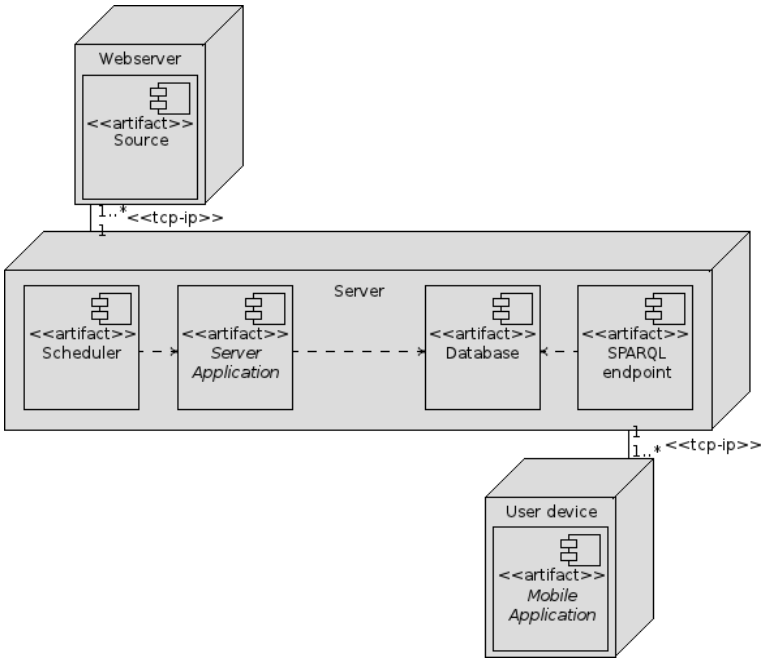
Identifikovány byly následující základní struktury systému:

- *Serverová aplikace.*

- Plánovač.
- Databáze.
- SPARQL endpoint.

Pod pojmem *serverová aplikace* se skrývá aplikace, která získává data přímo ze zdrojů, z nich vybírá informace, ty strukturuje a ukládá do databáze. Plánovač je reprezentován aplikací, která ve vhodné časy vyžádá na serveru aktualizaci informací. SPARQL endpoint je aplikace, která na požadavek v dotazovacím jazyku vrátí odpovídající data z databáze. Databáze je úložištěm informací.

Nasazení bylo nakonec navrženo, jak je vidět na obrázku 2.1, umístěním všech prvků serverové aplikace na jediný server, který získává informace z prostředí Internetu a zároveň poskytuje informace uživatelům.



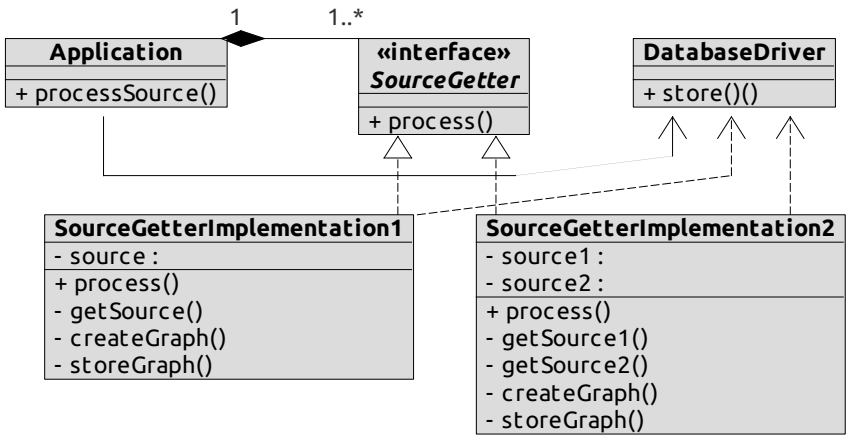
Obrázek 2.1: Diagram nasazení serverové aplikace

2.1.6 Návrh řešení

Návrh (*design*) byl prováděn po rozvržení architektury a za přispění volby použitých technologií.

Pro tři ze čtyř dříve identifikovaných struktur systému jsem nalezl vhodné hotové řešení (viz část 2.1.8 na straně 39) – plánovač je tvořen cronem, jako databáze je použita MongoDB a SPARQL endpoint poskytuje rdfstore. Zbývá tedy doimplementovat pouze *serverovou aplikaci*.

Na obrázku 2.2 je znázorněn třídní diagram (*class diagram*) reprezentující strukturu serverové aplikace. Metoda `process()` má za úkol stažení dat z patřičných zdrojů, jejich zpracování, převedení do sémantických grafů a uložení do databáze. V návrhu byl použit návrhový vzor strategie (*strategy*).



Obrázek 2.2: Třídní diagram serverové aplikace

2.1.7 Návrh bezpečnosti

Po zběžném návrhu aplikace byly modelovány hrozby. Nejprve jsem aplikaci dekomponoval na důvěryhodné a nedůvěryhodné komponenty systému a určil mezi nimi hranice důvěry. Poté jsem identifikoval hrozby. Nakonec byly identifikovány zranitelnosti. Vymodelované hrozby byly následně snižovány.

2.1.7.1 Dekompozice

Byly identifikovány následující komponenty (obdobné s těmi z návrhu architektury):

- *Serverová aplikace.*
- Plánovač.
- Databáze.
- SPARQL endpoint.
- Uživatel.
- Zdroj informací.

Plně důvěřovat můžu pouze serverové aplikaci a plánovači – mají to být vlastní jednoduché aplikace. Dále lze do určité míry důvěřovat databázi a SPARQL

endpointu – aplikace budou provozovány pod vlastní správou. Zdroje informací a uživatelské aplikace jsou naopak nedůvěryhodné – jsou plně v režii třetích stran. Hranice důvěry jsou nicméně mezi všemi komponentami.

2.1.7.2 Identifikace hrozeb

U komponenty *serverová aplikace* byly identifikovány následující hrozby:

- Podvržení nepravdivých dat.
- Přetížení komponenty velkým počtem požadavků.

U komponenty *databáze* byly identifikovány následující hrozby:

- Přetížení komponenty velkým počtem požadavků.
- Podvržení nepravdivých dat.

U komponenty *SPARQL endpoint* byly identifikovány následující hrozby:

- Přetížení komponenty velkým počtem požadavků.
- Podvržení identity uživatele.
- Únik informací.

U komponenty *plánovač* nebyly identifikovány, vzhledem k jednoúčelovému zaměření, žádné hrozby.

2.1.7.3 Identifikace zranitelností

Všechny výše uvedené hrozby mohou vyústit ve zranitelnosti, nejvyšší prioritu mají ty nejpravděpodobnější – na hranicích důvěry směrem k uživatelům a zdrojům dat. S hrozbou můžeme naložit následovně:

- Ponechat v programu.
- Odstranit problém.
- Odstranit hrozbu.

Preferovaná je zpravidla poslední možnost.

2.1.7.4 Důsledky

- Je třeba důsledně ošetřovat vstupy (aby nedošlo k uložení škodlivých dat do databáze, aby na SPARQL endpointu nebyly prováděny jiné operace, než dotazovací...).
- Je třeba ve výchozím nastavení vše zakázat a k potřebným úkonům udělit výjimky (pouze dotazy nad SPARQL endpointem, přístup jen k některým zdrojům ze serverové aplikace...).
- Je třeba zamezit odepření služeb (přetížením SPARQL endpointu, zahlcením databáze...).

2.1.8 Použité technologie

Mnoho použitých technologií bylo dáno zadáním práce, byl mi tak do značné míry usnadněn jejich výběr. Na druhou stranu nejsou některé z technologií příliš ověřené, zdokumentované a neexistují k nim příklady, takže by často bývala práce s jinými snazší.

Celá serverová aplikace je napsána v JavaScriptu, zprovozněna na Node.js, datovým úložištěm se stalo MongoDB.

2.1.8.1 JavaScript

Využití JavaScriptu bylo, pro svoje pravděpodobné vzrůstající nasazení v budoucnosti a své specifické vlastnosti, dáno jako nefunkční požadavek.

2.1.8.2 Node.js

Node.js (`http://nodejs.org/`) je platforma postavená nad běhovým prostředím JavaScriptu z webového prohlížeče Google Chrome. Slouží ke tvorbě rychlých, škálovatelných síťových aplikací. Node.js využívá událostmi řízený neblokovací I/O model, který ho činí nenáročným a efektivním, vhodným pro datově náročné *real-timové* aplikace běžící v distribuovaném prostředí [14].

Použití Node.js bylo vynuceno nefunkčním požadavkem, jedná se ale o velmi cennou zkušenost – je novou technologií a ačkoliv je jeho dokumentace na slušné úrovni, není zatím vytvořena dostatečně velká báze příkladů a nejlepších praktik.[23]

2.1.8.3 MongoDB

MongoDB (`http://www.mongodb.org/`) je škálovatelnou, vysoce výkonnou, otevřenou NoSQL databází napsanou v C++ [13].

[23] K dobrým praktikám jsem se tedy musel postupně propracovat a při té příležitosti se toho hodně naučil. Navíc, zdá se, se budou nabité vědomosti v budoucnu hodit.

Databázi jsem volil z dalších sémantických / NoSQL databází (zaměření dáno požadavkem na použití nových inovativních technologií), zvažovány byly mimo jiné i CouchDB (`http://couchdb.apache.org/`), Neo4j (`http://neo4j.org/`) a Openlink Virtuoso (`http://virtuoso.openlinksw.com/`).

Poslední jmenovaná databáze byla na základě předchozích zkušeností vybrána nejdříve, později se ale ukázalo, že jsou její správa i obsluha pro konkrétní použití těžkopádné, takže jsem nakonec přešel k podstatně vhodnější databázi, MongoDB, kterou lze navíc snadno využít v kombinaci s JavaScriptovým RDF úložištěm rdfstore (viz část 2.1.9.6).

2.1.8.4 Cron

Cron je démonem vykonávajícím naplánované úlohy. Název neoznačuje konkrétní implementaci, ale používá se pro označení programu splňujícího určité specifikace (jak je u Unixových nástrojů zvykem). Byla vybrána implementace, jejímž autorem je Paul Vixie – jedná se o verzi využitou například v GNU/Linuxu a BSD (odkud pochází).

Cron byl vybrán pro svou vhodnost pro daný účel – plánovat, a pro svoji prověřenost.

Celé prostředí serverové aplikace je umístěno v serverové edici GNU/linuxové distribuce Ubuntu (`http://www.ubuntu.com/`) a ta je pro snadnou přenositelnost nainstalována do kontejneru virtuálního stroje virtualizačního řešení VirtualBox (`http://www.virtualbox.org/`). Technologie byly vybrány na základě výborných předchozích zkušeností.

2.1.9 Použité knihovny

Mimo standardního JavaScriptu lze v Node.js využívat i specifická rozšíření a množství modulů třetích stran. Z těch jsem po zvážení z různých příčin využil:

2.1.9.1 Node-date-utils

Node-date-utils (`https://github.com/JerrySievert/node-date-utils`) je mikro frameworkem doplňujícím funkcionalitu JavaScriptového objektu Date. Nástroj byl vybrán z potřeby snadno formátovat čas.

Zvažováno bylo i použití obdobného nástroje node-dateformat (`https://github.com/felixge/node-dateformat`), ten je ale svou nabídkou funkcionalit pro projekt méně vhodný. Oba moduly jsou stále ve vývoji.

2.1.9.2 Express

Express (`http://expressjs.com/`) je renomovaným frameworkem pro tvorbu webových aplikací. Je založený na middleware frameworku connect (`http:`

`//www.senchalabs.org/connect/`).

Express byl vybrán jako ověřené řešení pro vytvoření struktury webové aplikace – nejenže v sobě zahrnuje mnoho dobrých praktik, ke kterým bych jen stěží došel vlastním zkoumáním, ale přináší zároveň i markantní zjednodušení struktury kódu aplikace.

2.1.9.3 Node-icalendar

Node-icalendar (`https://github.com/tritech/node-icalendar`) je knihovnou pro práci s formátem iCalendar. Je stále ve vývoji, nicméně množina poskytovaných funkcionalit je pro tuto práci dostatečná.

Zvažovány byly i další knihovny, například node-ical (`https://github.com/peterbraden/node-ical`), ty ale neimplementují dostatek funkcionality potřebné pro tvorbu aplikace.

2.1.9.4 Node-iconv

Node-iconv (`https://github.com/bnoordhuis/node-iconv`) je knihovnou pro obsluhu systémového příkazu iconv sloužícího k převádění kódování. Knihovna byla vybrána pro bezpečnou implementaci převodu kódování.

2.1.9.5 Node-jquery

Node-jquery (`https://github.com/coolaj86/node-jquery`) je *wrapperem* umožňujícím využít renomovanou JavaScriptovou knihovnu jQuery (`http://jquery.com/`) v prostředí Node.js.

jQuery bylo vybráno pro usnadnění zpracování externích zdrojů dat – nabízí prověřené způsoby selekce, traverzování a manipulace s DOM.

Node-jquery interně využívá modul jsdom (`https://github.com/tmpvar/jsdom`), který mu poskytuje JavaScriptovou implementaci DOM.

2.1.9.6 Rdfstore-js

Rdfstore-js (`https://github.com/antoniogarrote/rdfstore-js`) je JavaScriptovou implementací RDF grafového úložiště s podporou pro SPARQL. Data mohou být (a v implementaci DP jsou) persistentně ukládána prostřednictvím Node.js modulu-ovladače node-mongodb (`https://github.com/christkv/node-mongodb-native`) do NoSQL databáze MongoDB (`http://www.mongodb.org/`). Modul je stále ve vývoji.

Právě pro implementaci manipulace s RDF, grafové úložiště a podporu pro SPARQL byl rdfstore-js vybrán jako základ pro SPARQL endpoint a důležitá součást *serverové aplikace*.

2.1.10 Použité nástroje

Pro tvorbu serverové aplikace byly vybrány následující nástroje:

2.1.10.1 Kate

Pro implementaci serverové aplikace, stejně jako pro mnoho dalších činností, byl vybrán textový editor Kate (`http://kate-editor.org/`) z prostředí GNU/Linuxového desktopového prostředí KDE. Kate nabízí zvýrazňování syntaxe všech v diplomové práci použitých formátů, poskytuje velmi užitečnou práci s regulárními výrazy, umožňuje pracovat s mnoha soubory najednou a, co je asi nejpodstatnější, plně se integruje do (mnou používaného) KDE, takže ním například lze prostřednictvím KDE Input/Output (KIO) transparentně pracovat se vzdálenými soubory. Kate neumožňuje automatické doplňování kódu, nabízí ale již použitá slova kdekoliv v dokumentu, což je vlastnost v mnoha ohledech srovnatelná. Kate byl zvolen i na základě velice kladných zkušeností pro vytvoření zdrojového kódu a konfiguračních souborů.

2.1.10.2 Kompare

Pro případ potřeby důkladného porovnání souborů byl vybrán Kompare (`http://www.caffeinated.me.uk/kompare/`). Nástroj umožňuje velmi přehledně porovnat, mimo jednotlivých souborů, i celé jejich adresáře. Kompare neporovnává soubory sám, ale je nadstavbou nad konzolovou aplikací diff či obdobnou. Mimo již popsaných vlastností byl Kompare vybrán i pro svou integraci s KDE.

2.1.10.3 Webové prohlížeče

Pro průzkum zdrojů a jejich struktur byly zvoleny internetové prohlížeče a jejich rozšíření:

- Mozilla Firefox (`http://www.mozilla.org/firefox`) + Firebug (`http://getfirebug.com/`).
- Chromium (`http://chromium.org/`).

2.1.10.4 Protégé

Pro průzkum a práci s ontologiemi byl zvolen editor Protégé (`http://protege.stanford.edu/`). Rozhodnutí bylo učiněno pro nabídku funkcionalit a na základě předchozích zkušeností.

2.1.10.5 Curl

Konzolový nástroj curl z projektu cURL (`http://curl.haxx.se/`) byl zvolen pro snadnou kontrolu cizích zdrojů i vlastní vytvářené aplikace.

2.1.10.6 Git

Pro správu verzí práce byl zvolen nástroj Git (`http://git\discretionary{-}{}{}scm.com/`), čímž má být zajištěné pohodlné zálohování, jak ve smyslu možnosti návratu k některé z minulých verzí, tak i ve smyslu vzdáleného duplikování na serveru GitHub (`http://github.com/`). Pozitivním vedlejším efektem tohoto přístupu je možnost sledování vývoje prostřednictvím veřejného repozitáře.

2.2 Mobilní aplikace

Mobilní aplikace má sloužit studentům jako průvodce a měla by spolupracovat se serverovou aplikací. Vzhledem k této závislosti je třeba ji realizovat až po té serverové, ovšem analýza s návrhem by měly probíhat kvůli vzájemné spolupráci v některých fázích souběžně.

2.2.1 Potřeby uživatelů

Následující část se věnuje identifikaci základních potřeb potenciálních uživatelů aplikace, které by jim mohla / měla naplnit – na jejich základě totiž budou hledány uživatelské scénáře a požadavky.

2.2.1.1 Potřeba informací

Pod tuto potřebu shrnuji všechny potřeby nalezené u serverové aplikace (viz část 2.1.1 na straně 27). Jsou to:

- Potřeba zjistit informaci.
- Potřeba ověřit informaci.
- Potřeba ověřených informací.
- Potřeba aktuálních informací.
- Potřeba kompletních informací.
- Potřeba nalézt, co nehledám.

Zejména naplnění poslední potřeby – předložit informaci, o kterou člověk sice nepožádal, ale udělal by to, kdyby ho to bylo napadlo – je od průvodce vítané.

2.2.1.2 Potřeba uživatelské přívětivosti

Vzhledem k předpokládanému využití mobilní aplikace podstatně méně pokročilými uživateli, oproti serverové, je vhodné, aby tomu byla uzpůsobena. Cílová skupina navíc zpravidla není ochotná číst dlouhé návody a je tak třeba, aby byla aplikace použitelná sama o sobě.

2.2.1.3 Potřeba určit cílovou pozici

Jedna z nejdůležitějších potřeb, které by měl průvodce naplnit, je potřeba určit cílovou pozici. Může to být například:

- Budova, učebna, kancelář, knihovna...
- Menza, hospoda, občerstvení, nápojový automat...
- Pitná voda, toalety, sprchy...
- Výtah, schodiště, nájezd pro kočárek...
- Vchod, únikový východ, hasící přístroj...
- Bankomat, zastávka MHD, parkoviště...
- Informace, vrátnice, šatna, údržba...

Ačkoliv jsou v seznamu uvedené pozice rozděleny do určitých logických celků, je důležité si povšimnout, že existují i jiná jejich, v některých ohledech důležitější, uspořádání:

- Dle jednoznačnosti místa – místnost s jasně daným názvem je oproti libovolnému schodišti jediná.
- Dle jednoznačnosti identifikátoru – stejná místnost může být vyhledána na základě různých kritérií (název, následující výuka, popis).
- Dle frekvence hledání – přednáškové místnosti jsou vyhledávány podstatně častěji, než hasící přístroje.
- Dle kritéria priority – vzdálenější menza může mít vyšší prioritu na základě lepší kuchyně, zatímco sebelepší nápojový automat může mít kvůli své větší vzdálenosti prioritu nižší.

Pro úplnost ještě uvádím seznam informací vedoucích k určení cílové pozice:

- Označení místnosti, budovy... (T9:349, TK...).
- Název místnosti, budovy... (Gočár, Národní technická knihovna...).
- Neoficiální název místnosti, budovy... (Bouračka, Modrá menza...).
- Jméno, pokud hledáme kancelář vyučujícího nebo neakademického pracovníka.
- Rozvrh, pokud hledáme místo další výuky nebo zkoušky.
- Zeměpisné souřadnice.

- Poloha všech míst, které přicházejí v úvahu (toalety, bankomaty...) – hledající si vybere.
- Předchozí bod a navíc přibližná aktuální poloha – lze určit nejbližší.

Přehledy rozšiřují poznatky získané při tvorbě bakalářské práce [8].

2.2.1.4 Potřeba určit aktuální pozici

Zvláště, pokud máme mapu nebo jsme v jiné situaci, kdy víme, kam jít, vystává potřeba určit aktuální pozici – bez té totiž nelze určit směr, vzdálenost, a v případě, že hledáme nejbližší místo daného typu, ani cíl cesty.

2.2.1.5 Potřeba nezávislosti

Cílová skupina má díky specifickému životnímu stylu silnou potřebu nezávislosti v mnoha oblastech – pokud jim má být poskytnutá aplikace užitečná, ani ta je nesmí příliš omezovat – je proto vhodné, aby se jednalo o aplikaci multiplatformní, mobilní a aby umožňovala offline provoz, jinak hrozí nemožnost jejího využití v situacích jako je cesta do školy nebo přítomnost na některém z mnoha signálem nepokrytých míst budovy T9.

2.2.2 Případy užití

U případů užití (*use cases*) mobilní aplikace byli identifikováni jediní dva aktéři:

- Uživatel – student.
- Server – SPARQL endpoint.

Dále byly identifikovány tři hlavní případy užití:

- Získání určité (textové) informace.
- Vyhledání (mapové) pozice.
- Vykonání SPARQL dotazu.

2.2.2.1 UC-M-1: Získání jídelníčku

Aktéři:

- Uživatel.
- Server.

Systém:

- Mobilní aplikace.

Vstupní podmínky:

- Uživatel se nachází na patřičném místě systému.
- Systém má připojení k serveru.

Hlavní scénář:

- Uživatel nalezne akci získání jídelníčku.
- Uživatel dá požadavek na získání jídelníčku.
- Systém odešle požadavek na server.
- Server zpracuje požadavek a odešle odpověď.
- Systém přijme odpověď
- Systém zpracuje odpověď.
- Systém vizualizuje jídelníček.

Alternativní scénáře:

- V případě chyby připojení je navráceno chybové hlášení.
- V případě neexistence jídelníčku je navráceno oznámení.

Poznámky:

- Místo získávání jídelníčku může být prováděna libovolná obdobná činnost.
- Pro některé obdobné informace nemusí být vyžadováno připojení k serveru, záleží na době jejich platnosti.

2.2.2.2 UC-M-2: Vyhledání občerstvení

Aktéři:

- Uživatel.

Systém:

- Mobilní aplikace.

Vstupní podmínky:

- Uživatel se nachází na patřičném místě systému.

Hlavní scénář:

- Uživatel zadá do vyhledávacího pole požadavek na občerstvení.
- Uživatel potvrdí vyhledávání.
- Systém vyhledá všechna občerstvení.
- Systém vizualizuje nalezená občerstvení.
- Uživatel zadá požadavek na zobrazení aktuální polohy.
- Systém určí uživatelovu polohu.
- Systém vizualizuje polohu.
- Uživatel si dle poskytnutých informací vybere občerstvení.

Alternativní scénáře:

- V případě neplatného dotazu je navráceno chybové hlášení.
- V případě chyby určování polohy je navráceno chybové hlášení.
- V případě výskytu mimo mapu je navráceno chybové hlášení.

Poznámky:

- Místo vyhledávání občerstvení mohou být vyhledávány i jiné pozice.
- U některých typů dotazů není nutné znát aktuální polohu.
- U některých typů dotazů je navrácena jediná pozice.

2.2.2.3 UC-M-3: Vykonání SPARQL dotazu

Aktéři:

- Uživatel.
- Server.

Systém:

- Mobilní aplikace.

Vstupní podmínky:

- Uživatel se nachází na patřičném místě systému.
- Uživatel zná dotazovací jazyk SPARQL.
- Systém má připojení k serveru.

Hlavní scénář:

- Uživatel vytvoří dotaz.
- Uživatel potvrdí dotaz.
- Systém odešle dotaz na server.
- Server zpracuje požadavek a odešle odpověď.
- Systém přijme odpověď.
- Systém zpracuje odpověď.
- Systém vizualizuje odpověď.

Alternativní scénáře:

- V případě neplatného dotazu je navráceno chybové hlášení.
- V případě chyby připojení je navráceno chybové hlášení.
- V případě chyby serveru je navráceno chybové hlášení.

Poznámky:

- V případě nenalezení informace je navrácena prázdná množina.

2.2.3 Požadavky

2.2.3.1 Funkční požadavky

Funkčními požadavky (*functional requirements*) jsou myšleny takové požadavky, které jsou zaměřeny na jednotlivé funkcionality. Pro mobilní aplikaci mezi ně patří především:

1. Aplikace umožní zadat vlastní SPARQL dotaz. Ten se odešle na server, kde se zpracuje, a výsledek se navrátí uživateli.
2. Aplikace bude získávat data předpřipravenými SPARQL dotazy.
3. Aplikace dokáže na mapě vizualizovat budovy.
4. Aplikace dokáže na mapě vizualizovat místnosti.
5. Aplikace dokáže na mapě vizualizovat body zájmu.
6. Aplikace umožní určení aktuální pozice a její předání dalšímu zpracování.
7. Aplikace dokáže na mapě vizualizovat polohu uživatele.

8. Aplikace bude poskytovat nápovědu pro snazší použití.

9. Aplikace bude postavená nad daty reprezentovanými výše uvedenou ontologií.

Pro očekávaný malý rozsah sémantické databáze bude aplikace získávat data přímo ze zdrojů na úkor předpřipravených SPARQL dotazů – na datech databáze by jich mnoho vytvořit nešlo a aplikace by tak nepřinášela mnoho užitku. Prostor pro předpřipravené SPARQL dotazy je nicméně třeba vytvořit.

2.2.3.2 Nefunkční požadavky

Nefunkčními požadavky (*non-functional requirements*) jsou myšleny takové požadavky, které jsou zaměřeny na dílo jako celek, nikoliv na jednotlivé funkcionality. Pro mobilní aplikaci mezi ně patří především:

1. Aplikaci vytvořte v HTML5, aby byla na současných, obzvláště mobilních, zařízeních multiplatformní.

2. Aplikace bude funkční v prohlížečích na nejnovějších verzích OS Android, iOS a Symbian.

3. Aplikace bude funkční v majoritních desktopových prohlížečích.

4. Při vývoji se zaměřte na použitelnost, aplikaci by měl být schopný, po krátkém zaškolení, používat jakýkoliv běžný student Fakulty informačních technologií.

5. Snažte se nabízet alternativy k zadávání dlouhých textů, aby byla usnadněna obsluha uživatelům mobilních zařízení bez hardwarových klávesnic.

6. Práce nemusí nutně obsáhnout celou zpracovávanou doménu, je ale žádané, aby kvalitně zpracovala hlavní scénáře využití a nebránila dalšímu rozšiřování.

7. Zveřejněte aplikaci a její data pod svobodnými licencemi z rodin GNU, Creative Commons a kompatibilními, aby byl umožněn další rozvoj nezávislý na původním autorovi.

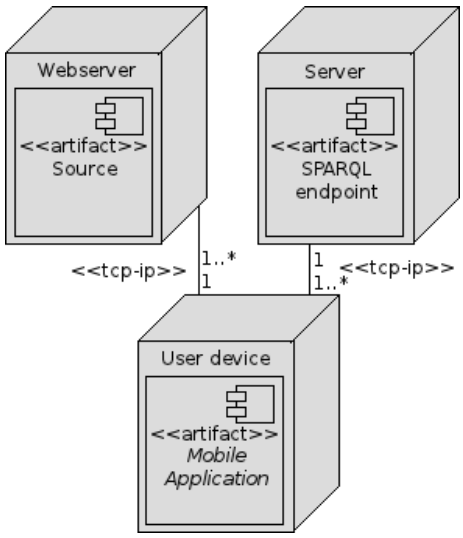
8. Aplikace umožní využití offline.

2.2.4 Návrh architektury

I v případě mobilní aplikace byl nejprve vytvořen návrh architektury (*architecture design*) aplikace a ten byl později rozpracován do návrhu nasazení (*deployment design*). Náročná práce to ale nebyla – systém je jednolitý.

Nasazení mobilní aplikace bylo navrženo tak, jak je vidět na obrázku 2.3. Aplikace komunikuje se SPARQL endpointem serverové aplikace, v případě

neposkytování daných informací aplikace komunikuje přímo se zdrojem informací.



Obrázek 2.3: Diagram nasazení mobilní aplikace

2.2.5 Návrh řešení

Návrh (*design*) byl prováděn po rozvržení architektury, za přispění volby použitých technologií a reflektovány do něj byly i požadavky na uživatelské rozhraní.

Systém byl rozdělen do čtyř modulů sloužících jako průvodce, navigace, SPARQL konzole a nápověda. Každý z modulů je víceméně nezávislý na těch ostatních, jen nástroje pro dotazování se SPARQL endpointu byly vyčleněny pro možnost využití ve všech modulech.

Na obrázku 2.4 je znázorněn třídní diagram (*class diagram*) mobilní aplikace.

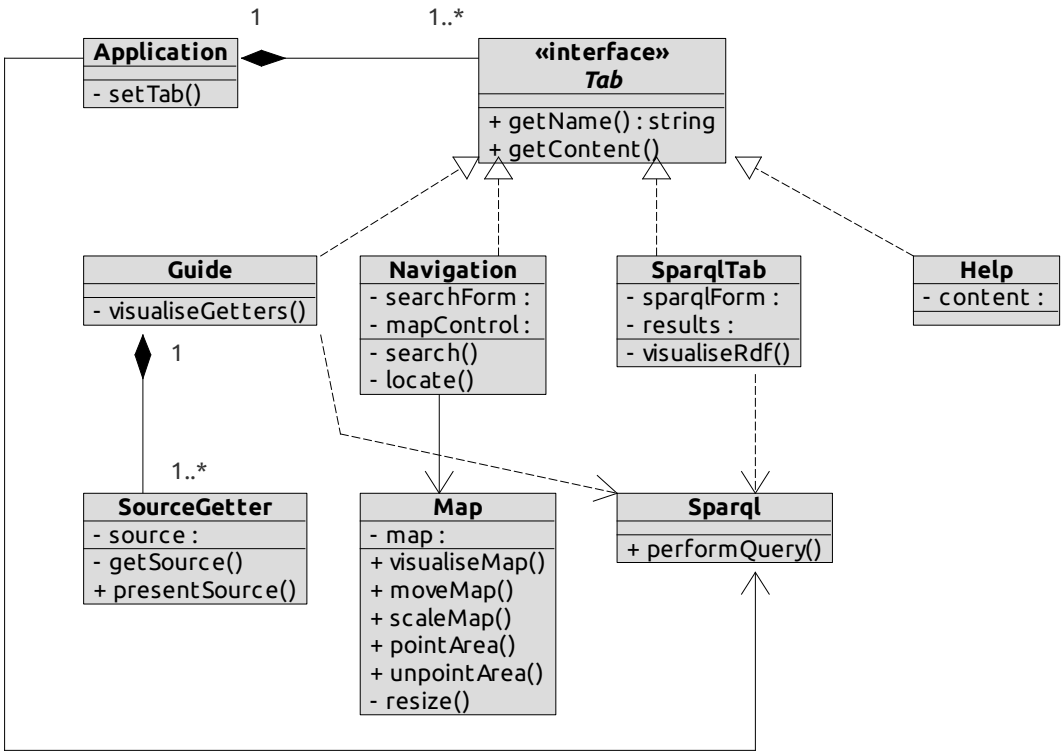
2.2.6 Návrh bezpečnosti

Návrh aplikace je poměrně přímočarý a není ho třeba nějak složitě vysvětlovat, co se ale týče bezpečnosti, je to podstatně horší – v prováděných činnostech se vyskytuje několik skrytých hrozeb, které je třeba identifikovat a odstranit z nich plynoucí zranitelnosti.

2.2.6.1 Dekompozice

Byly identifikovány následující komponenty:

- Mobilní aplikace.



Obrázek 2.4: Třídní diagram mobilní aplikace

- SPARQL endpoint.
- Zdroje informací.
- Uživatel.

Plná důvěra může být pouze v rámci mobilní aplikace – tam, kde nemůže být žádný zásah zvenčí. Dále lze do určité míry důvěřovat SPARQL endpointu – bude provozován pod vlastní správou. Zdroje informací a vstupy uživatele jsou naopak nedůvěryhodné – jsou plně v režii třetích stran. Hranice důvěry jsou přesto mezi všemi komponentami.

2.2.6.2 Identifikace hrozeb

Hrozby je třeba identifikovat pouze na komponentě *mobilní aplikace*, ostatní komponenty jsou v tuto chvíli irelevantní:

- Podvržení nepravdivých dat.
- Podvržení škodlivých dat.

2.2.6.3 Identifikace zranitelností

Obě výše uvedené hrozby mohou vyústit ve zranitelnosti. S podvržením nepravdivých dat se bohužel nevypořádáme - data jsou čerpána z oficiálních zdrojů a v případě, že někdo kompromituje je, nám žádné z opatření nepomůže. Zranitelnosti plynoucí z druhé hrozby omezit ale lze. Mohou jimi být:

- Injektáž škodlivého skriptu ze zdroje – vzhledem k použitým technologiím, JavaScriptu.
- Injektáž škodlivého značkovacího kódu ze zdroje – neuzavřené HTML elementy, identifikátory duplicitní k již existujícím, destruktivní formátování...
- Neošetřený uživatelský vstup – neplatný regulární výraz...

2.2.6.4 Důsledky

- Je třeba ve výchozím nastavení zakázat všechny cizí vstupy a potřebným udělit výjimky.
- Je třeba důsledně ošetřovat vstupy.
- Je třeba zamezit injektování cizích JavaScript souborů.
- Je třeba ve výchozím nastavení zakázat všechny HTML elementy cizích zdrojů a potřebným udělit výjimky.
- Je třeba ve výchozím nastavení zakázat všechny HTML atributy cizích zdrojů a potřebným udělit výjimky.
- Je třeba ve výchozím nastavení zakázat všechny hodnoty HTML atributů cizích zdrojů a potřebným udělit výjimky.

2.2.7 Návrh uživatelského rozhraní

Na základě požadavků byl, ve spojitosti s návrhem aplikace, vytvářen návrh uživatelského rozhraní (*user interface design*). Návrh probíhal hlavně za využití *low fidelity* prototypů, na těch bylo uživatelské rozhraní odzkoušeno a podle nich pak bylo uživatelské rozhraní aplikace realizováno. Tvorba samostatných *high fidelity* prototypů byla svým způsobem vynechána – použité technologie jsou natolik flexibilní, že byla za podobným účelem využita samotná finální aplikace.

2.2.7.1 Low fidelity prototypy

Pod pojmem *low fidelity prototypy* se skrývají jednoduché, velmi snadno vytvořitelné a editovatelné prototypy uživatelského rozhraní aplikace, které mimo jiné ověřují vhodnost rozvržení ovládacích komponent a uživatelských procesů zamýšlené aplikace.

Pomocí prototypů byly navrženy jednotlivé kontexty aplikace:

- Průvodce.
- Navigace.
- SPARQL konzole.
- Nápověda.

Podrobněji zmíním pouze navigaci – je ze zpracovávaných kontextů nejsložitější, ostatní obsahují převážně shodné prvky.

Obrázek 2.5: Low fidelity prototyp mobilní navigace

Na obrázku 2.5 je znázorněn *low fidelity* prototyp mapové části mobilní aplikace. V horní části je vidět menu pro pohyb v různých kontextech aplikace, to je v každém kontextu shodné. V horní části specifické pro navigaci se nachází vyhledávací formulář navigace. Do jediného vyhledávacího pole by mělo být možné psát libovolné hledané fráze a aplikace by si s nimi měla poradit. Vyhledávací pole nabízí našeptávač, čímž je usnadněno nalezení výsledku

těm, kteří nevědí, co přesně hledají, i těm, kteří mají psaní komplikované třeba vlivem dotykového displeje. Ve zbytku prostoru se nachází mapa s ovládacími prvky pro:

- Posun.
- Škálování.
- Zobrazení aktuální polohy.

Tyto akce mají být implementovány i pomocí gest přímo na mapě, pro obsluhu s tím neseznámenými uživateli je ale nutné grafické ovládací prvky zachovat. Za povšimnutí také stojí ikony umístěné na mapě – je třeba vhodně reprezentovat uživatelovu polohu, polohu hledaného místa a body zájmu.

2.2.8 Použité technologie

Níže uvádím soupis použitých technologií s odůvodněním, proč byly vybrány a v jakém kontextu použity.

2.2.8.1 HTML5

Většina technologií tvořících mobilní aplikaci by se dala zahrnout do pojmu HTML5 (v širším smyslu významu). Pomocí JavaScriptu se manipuluje s HTML strukturou, která je prezentována za pomocí CSS, klíčová část aplikace – mapa – má být tvořena SVG (o tom ještě dále).

Technologie byly vybrány pro jejich výbornou portovatelnost – téměř všechna dnešní zařízení je implementují podobně (nebo alespoň mají takové výkonnostní parametry, že na nich lze provozovat přidanou vrstvu abstrahující od skutečné implementace).

2.2.8.2 SVG

Mapa použitá v aplikaci je implementována prostřednictvím SVG. Rozhodnutí bylo učiněno na základě identifikace a porovnání možných technologií:

- Statické obrázky s předgenerovanými proměnnými částmi.
- Statické obrázky s dynamicky pozicovanými proměnnými částmi.
- Dynamicky kreslené obrázky na *canvas*.
- Statické SVG s dynamickými úpravami.

První zmíněné řešení bylo zavrženo kvůli nízké flexibilitě – není možné dopředu vygenerovat všechny varianty obrázků, které by pokrývaly možné

situace – i kdyby to možné bylo, množství obrázků by bylo velmi vysoké[24] a pro omezenou paměťovou kapacitu cílových zařízení nevhodné.

Druhý přístup, dynamické pozicování, byl zavržen z důvodů optimalizací rozvržení prvků mobilními prohlížeči – není zaručené, že se prvek (například ukazatel nalezené místnosti) skutečně umístí, kam má.

Nová vlastnost HTML5, kreslení prostřednictvím JavaScriptu na *canvas* element, byla, na rozdíl od předchozích možností, zvažována až do finálního rozhodnutí – ačkoliv se jedná o řešení čistě v režii již používaných technologií (JavaScript, HTML), jeho využití jsem z několika důvodů zavrhl:

- Neumožňuje nakreslený obrázek nijak upravit (například posunout ukazatel na mapě), ale musí se vždy překreslit, což je náročné na zdroje.
- Kvůli principu dynamického generovaní obrázků JavaScriptem nelze kostru připravit pomocí běžných grafických editorů.
- Subjektivně mi přijde práce s mapou jako SVG obrázkem reprezentovaným XML dokumentem majícím DOM přehlednější, než s méně strukturovatelným JavaScriptem.

Reprezentaci mapy jsem tedy svěřil SVG z následujících důvodů:

- Snadné vytvoření mapy prostřednictvím běžného grafického editoru.
- Možnost znovupoužití mapy standardního grafického formátu i v jiných projektech.
- Práce s mapou (například vyznačování pozic) se provádí pouhou úpravou jejího DOM.
- I po úpravách se stále jedná o týž objekt – není nutné ho překreslovat.

Nutno podotknout, že jsou implementace kreslení na *canvas* a SVG na různých mobilních zařízeních na různých úrovních, ani jedna ovšem nemá výrazně navrch [1] [3].

2.2.8.3 PHP

Z důvodů popsaných v části 3.2.1.7 na straně 66 bylo nutné vytvořit podpůrnou serverovou aplikaci – proxy. Po úvaze jsem se rozhodl pro využití jazyka PHP, který je podporovaný jako jediný nástroj pro tvorbu dynamického obsahu na fakultním webovém serveru `http://webdev.fit.cvut.cz/`. Tuto kombinaci, vzhledem k její podpoře ze strany fakulty a určení práce pro fakultní nasazení, považuji i přes jinde využitý JavaScript za vhodnou.

[24] Pro pokrytí všech situací by byly třeba zvlášť obrázky s vyznačením kterékoliv místnosti, kteréhokoliv bodu zájmu, kterékoliv výchozí pozice... Vzhledem k rozsáhlosti mapy by tak počet obrázků byl enormní.

2.2.9 Použité knihovny

V práci byla, po úvaze, nakonec využita i práce třetích stran – svobodné knihovny *jQuery* a *HTML Purifier*. Jejich krátký popis a odůvodnění následují:

2.2.9.1 JQuery

Stále více projektů využívá renomovanou JavaScriptovou knihovnu jQuery (`http://jquery.com/`). Osobně z toho moc nadšený zpravidla nejsem, často se nadužívá – velikost každé stránky se tak zvýší řádově minimálně o sto kilobytů, což není zanedbatelné[25] – na druhou stranu se knihovna ve velkém počtu případů *cachuje*. Po důkladném zvážení jsem se jQuery ale nakonec rozhodl využít i v případě mobilní aplikace – aplikace má být určena pro offline použití, takže se předpokládá už tak rozsáhlá aplikační logika, velikost jQuery tak bude zanedbatelná a knihovna může být dobrým základem stabilní multiplatformní aplikace.

Mimo abstrakce od různých implementací specifikací webovými prohlížeči nabízí jQuery selekci, traverzování a manipulaci s DOM.

2.2.9.2 HTML Purifier

Po několika vlastních implementacích nástroje na pročištění a ošetření HTML vstupů pocházejících od třetích stran jsem sáhl k implementaci již prověřené – oblast ošetření aplikace před škodlivými vstupy má z hlediska bezpečnosti vysokou prioritu a neustále se objevující nové hrozby nakonec vedly k tomu, že jsem se rozhodl nespoléhat na své vlastní implementace a využít *know-how* tvůrců svobodné PHP knihovny HTML Purifier (`http://htmlpurifier.org/`) – ta odstraní případná nebezpečí za využití pokročilého auditovaného *whitelistu* a v případě potřeby navíc zajistí validitu kódu. Knihovna je využita v rámci proxy zmíněné v části 3.2.1.7 na straně 66.

2.2.10 Použité nástroje

Pro tvorbu aplikace byly použity převážně následující nástroje. Majorita jich je uvolněných pod svobodnými licencemi, zpravidla GNU GPL nebo kompatibilními. Jedinou výjimkou jsou některé mobilní prohlížeče, které byly použity k testování aplikace.

2.2.10.1 Kate

Textový editor Kate (`http://kate-editor.org/`) byl, mimo serverové (viz část 2.1.10.1 na straně 42), vybrán i pro tvorbu mobilní aplikace. Nástroj

[25] V průzkumu [10] věnovanému maximální velikosti stránky je přesně tato velikost uvedená jako maximální přijatelná pro celou stránku. Průzkum je sice patnáct let starý, rychlosti připojení k některým webovým serverům ale za tu dobu nevzrostly.

byl zvolen na základě již dříve zmíněných důvodů pro napsání kódu v JavaScriptu, HyperText Markup Language (HTML), Cascading Style Sheets (CSS) i úpravy mapy v Scalable Vector Graphics (SVG).

2.2.10.2 Inkscape

Grafické podklady práce byly vytvořeny v editoru vektorové grafiky Inkscape (`http://www.inkscape.org/`). Nástroj implementuje téměř kompletní specifikaci formátu SVG verze 1.1, je multiplatformní a svobodný. Inkscape byl vybrán kvůli předchozím výborným zkušenostem s prací v něm a celkové vhodnosti pro tvorbu mapových podkladů.

2.2.10.3 Kompare

Nástroj byl zvolen ze stejných důvodů, jak tomu bylo u serverové aplikace (viz část 2.1.10.2 na straně 2.1.10.2).

2.2.10.4 Webové prohlížeče

Pro správné odladění aplikací, průzkum možností a testování v průběhu vývoje byly vybrány následující prohlížeče. Některé kvůli jejich významnému zastoupení mezi uživateli, jiné pro odladění aplikace i v jiných implementacích použitých technologií.

- Mozilla Firefox (`http://www.mozilla.org/firefox`).
- Chromium (`http://chromium.org/`) – základ Google Chrome.
- Prohlížeč platformy Android (`http://android.com/`).
- Opera Mobile (`http://www.opera.com/mobile`).

Další prohlížeče byly použity po hrubém dokončení aplikace k odladění zbývajících nedostatků z oblasti multiplatformnosti (viz část 4.2.1 na straně 72).

2.2.10.5 Balsamiq Mockups

Pro tvorbu *low fidelity* prototypů byl využit nástroj Balsamiq Mockups (`http://www.balsamiq.com/`) ve webové edici – je sice mírně omezená, ale použití je zdarma. Nástroj byl vybrán pro množství poskytovaných možností a pro výborné předchozí zkušenosti.

2.2.10.6 Git

Nástroj byl zvolen ze stejných důvodů, jak tomu bylo u serverové aplikace (viz část 2.1.10.6 na straně 2.1.10.6).

KAPITOLA 3

Realizace

Během implementace (*implementation*) se objevila celá řada zádrhelů, z nichž mnoho souviselo s doposud ne zcela zdokumentovanými a odladěnými technologiemi, přesto pro mne byla realizace velmi přínosná – neustálým hledáním různých řešení a alternativ jsem nutně narazil na mnoho dalších zajímavých postřehů a nápadů. Kapitola se zabývá řešenými problémy, komplikacemi a celá je protkána implementací bezpečnosti.

3.1 Serverová aplikace

Když se zpětně ohlížím za implementací serverové aplikace, náročná objektivně vůbec nebyla, osobně jsem ale žádnou z používaných technologií / knihoven... v té době neznal, takže mi náročnost přišla enormně vysoká. Rozsah popisu realizace serverové aplikace je tak krátký – mnoho objektivně zajímavých míst se v ní nevyskytlo.

Realizováno bylo pouze zpracování čtyř rozličných typů zdrojů – implementace ostatních by byla dlouhotrvajícím opakováním stejných postupů a práci by moc neobohatila.

3.1.1 Řešené problémy

V době, kdy byl vytvořený návrh, bylo vytvoření aplikace už poměrně snadné. Popíšu tedy jen ve stručnosti několik základních částí řešení.

3.1.1.1 Zpracování zdroje

Proces zpracování zdroje je poměrně přímočarý. Nejprve je zdroj po jednotlivých *chuncích* stažen, následně sestaven:

```
var http = require('http'); //načtení modulu
```

```
var request = http.get({
    host: 'akce.cvut.cz',
    path: '/index.php'
});
request.on('response', function (r) {
    var data = '';
    r.on('data', function (chunk) {
        data += chunk;
    });
    r.on('end', function () {
        //zpracování dat
    });
});
```

Poté je pro snadnou a spolehlivou manipulaci vytvořen model dokumentu daného formátu, se kterým se pak pracuje, například pro HTML nebo XML:

```
var $ = require('jquery');
var jqo = $(data);
jqo.find('#content span.location').text();
```

Pro iCalendar:

```
var icalendar = require('icalendar');
var ical = icalendar.parse_calendar(data);
ical.events()[0].properties.LOCATION;
```

Získaná data jsou pak ukládána do grafového úložiště:

```
var rdfstore = require('rdfstore');

var store = rdfstore.create(config, function (store) {
    var graph = store.rdf.createGraph();

    store.rdf.setPrefix(
        "lode", "http://linkedevents.org/ontology/");

    for (var e in ical.events()) {
        var event = store.rdf.createNamedNode(
            ical.events()[e].properties.URL.value);

        graph.add(
            store.rdf.createTriple(
                event,
                store.rdf.createNamedNode(
```

```
                    store.rdf.resolve("rdf:type")),
                store.rdf.createNamedNode(
                    store.rdf.resolve("lode:Event"))
            )
        );

        //další informace zpracovávány obdobně
    }

    store.insert(graph, function () {
        console.log("Graph has been inserted.")
    });
});
```

Zdrojový kód je v tomto případě dobře čitelný, žádné další komentáře tedy asi nepotřebuje. Za povšimnutí pouze stojí asynchronní charakter celé aplikace – nikde se neblokují zdroje zbytečně a nemůže se stát, že by nějaká událost předběhla jinou.

3.1.1.2 Zdroje uživatelských jmen

Nepodařilo se mi nalézt vhodný zdroj uživatelských jmen vyučujících a neakademických pracovníků. Zatímco uživatelská jména studentů jsou zveřejněna na `https://timetable.fit.cvut.cz/public/cz/studenti/`, na obdobné stránce věnované učitelům, `https://timetable.fit.cvut.cz/public/cz/ucitele/`, jsou pouze tituly, křestní jména a příjmení – tedy nic, co by bylo vhodným identifikátorem. Zdroj uživatelských jmen neakademických pracovníků se mi nepodařilo nalézt žádný. Vzhledem k plánovanému budoucímu napojení aplikace na KOSapi (`http://kosapi.fit.cvut.cz/`), až získá přístup do KOS, byly jako dočasné řešení zvoleny soubory obsahující jména a uživatelská jména všech osob.

3.1.1.3 Routování

Aplikace je serverem[26] a obsluhuje se HTTP požadavky, je tak dosaženo větší univerzálnosti. Například pro aktualizaci osob spjatých s fakultou je třeba odeslat požadavek `GET` na URL `http://localhost:1337/osoby/`, na které je server spuštěný. Zpracování takového požadavku je vyřízeno následujícím kódem ze *spouštěcího* souboru aplikace:

```
var express = require('express'); //načtení frameworku express
```

[26]Pozor, zde je důležité vzít v potaz vymezení pojmů z poznámky 1 na straně 20 – aplikace, o které je řeč, je *serverovou aplikací* v užším slova smyslu – není nazývána *serverovou aplikací* proto, že by nutně musela být serverem, ale proto, že je jednou z komponent serverového řešení. Že je tedy také serverem je pouze shoda náhod.

```
var app = express.createServer(); //vytvoření serveru

app.get('/osoby/:login?', function (req, res) {
    require('./sources/usermap.js').process(req.params.login);
    res.send("OK: Source enqueued for processing.");
});

//další zdroje zpracovávány obdobně

app.get('*', function (req, res) { //neodchycené cesty
    res.send("Error: Service not found.", 404);
});

app.listen(1337); //spuštění aplikace na portu 1337
```

V těchto několika řádcích je vytvořený celý server zpracovávající příchozí požadavky. Na pátém řádku je řečeno, že cesta směřující do *segmentu* osoby bude touto metodou odchycena. Odchycen bude i další, nepovinný, segment, který bude uložený do proměnné `login`. Je tedy možné volat i URL `http://localhost:1337/osoby/molnaja2/`, aniž by došlo k odchycení dvanáctým řádkem, který klientovi vrátí patřičnou chybu.

Uvnitř metody je na prvním řádku vyžadováno zavolání metody `process()` z uvedeného souboru s parametrem `login`. V této metodě je na základě přítomnosti parametru rozhodnuto, jestli se mají zpracovat všechny osoby nebo jen jedna konkrétní. Na druhém řádku je klientovi odesláno oznámení o úspěšném přijetí požadavku.

3.1.1.4 Nastavení časovače

Vzhledem k výše uvedené obsluze *serverové aplikace* HTTP požadavky je možné časovač reprezentovaný cronem jednoduše nastavit, například:

```
# aktualizace jidelnicku kazdy den 3 minuty po zverejneni
3 9,10,16,17 * * * user curl localhost:1337/jidelnicky/\
               2>&1 1> jidelnicky.all.log 2> jidelnicky.err.log

# aktualizace osob kazdy den 10 minut po druhe hodine
10 2 * * * user curl localhost:1337/osoby/\
               2>&1 1> osoby.all.log 2> osoby.err.log

# aktualizace akci 10 minut po kazde sude hodine
10 */2 * * * user curl localhost:1337/akce/\
               2>&1 1> akce.all.log 2> akce.err.log
```

```
# aktualizace rozvrhu kazdy den 10 minut po druhe hodine
10 2 * * * user curl localhost:1337/rozvrh/\
               2>&1 1> rozvrh.all.log 2> rozvrh.err.log
# aktualizace rozvrhu v dobe tvorby 10 minut po kazde hodine
10 * * 6,9,10,1,2 * user curl localhost:1337/rozvrh/\
               2>&1 1> rozvrh.all.log 2> rozvrh.err.log
```

Prvních pět parametrů udává čas vykonání akce (minuta, hodina, den měsíce, měsíc, den týdne), šestý uživatele, zbytek je vykonání příkazu – odeslání požadavku a *zalogování* výsledku.

3.1.2 Komplikace

Během implementace serverové aplikace vyvstalo několik komplikací, které bylo třeba vyřešit:

3.1.2.1 Převod kódování

Node.js vznikl až po prosazení UTF-8, dá se tedy předpokládat, že bude toto kódování podporovat. Předpoklad je to samozřejmě, i dle dokumentace [17], správný; horší je to ale, dle stejného zdroje, s ostatními kódováními. Samotný Node.js může data v nich převádět do *bufferu*, který pak lze následně, například pomocí systémového programu iconv volaného prostřednictvím knihovny node-iconv, převést do správného kódování.

Problém vyvstává u vysokoúrovňových nástrojů, které například dokáží jednoduše stáhnout cizí internetovou stránku a předpřipravit ji k následnému zpracování – pokud je u těchto nástrojů vrácen třeba DOM, v případě, že pocházela stránka ze zdroje, který využíval s Node.js nekompatibilní kódování, řekněme Windows-1250, převod kódování už nelze zpětně ovlivnit. Při zpracovávání některých stránek jsem tak musel zůstat u nízkoúrovňových nástrojů.

3.1.2.2 Špatně formované XML

Při práci se sémantickým úložištěm rdfstore-js jsem narazil na špatně formované XML. Chybu jsem identifikoval – jednalo se o sekvenci `<literal>>` v některých případech generovaného XML, nalezl chybu v kódu, nahlásil a odeslal *patch*. V současné době je chyba již odstraněna [9].

3.2 Mobilní aplikace

Pod pojmem *mobilní aplikace* se skrývá aplikace pevně nezávislá na výše uvedené *serverové*. Jejím hlavním cílem je sloužit studentovi FIT ČVUT jako průvodce, zároveň má ale plnit i účel ukázky využití SPARQL endpointu vytvořeného v *serverové* části.

3.2.1 Řešené problémy

Implementace mobilní aplikace přinesla mnoho menších problémů, které bylo třeba vyřešit. Mezi nejzajímavější patří:

3.2.1.1 Vyhledávání v navigaci

Aby bylo vyhledávání pro uživatele co nejvíce přívětivé, je třeba mu věnovat hodně pozornosti. Změnou vyhledávací fráze (stiskem klávesy, stiskem vyhledávacího tlačítka, odesláním formuláře, změnou formuláře) je vyvoláno hledání, kde jsou nejprve z vyhledávané fráze ořezány (počáteční a koncové) bílé znaky a následně je převedena do dvou regulárních výrazů – první je tvořený frází tak, jak je zadána, druhý je normalizovaný. Normalizací je odstraněna diakritika, interpunkce a zbylé bílé znaky. Nyní už jsou prohledána všechna místa z databáze, nejprve na přítomnost prvního zmiňovaného regulárního výrazu, poté se normalizují a testují na druhý regulární výraz. Ve vyhledávání nezáleží na velikosti písmen.

V průběhu celého procesu jsou zpracovávány krajní situace a řešena otázka uživatelské přívětivosti – v případě chyby v regulárním výrazu je uživateli nabídnuta oprava, jsou vypisována hlášení o průběhu vyhledávání a dle počtu nalezených výsledků jsou tyto prezentovány – jediný výsledek je rovnou zobrazen, více výsledků se vypíše a uživatel má možnost si dále z nich vybrat, případně pokračovat v zadávání hledané fráze.

3.2.1.2 Mapa v navigaci

Mapa v navigaci je tvořena SVG obrázkem, se kterým je manipulováno JavaScriptem. Z mapy je zobrazována pouze výseč, která je navíc dle potřeby škálována – to nutně vede k mnoha různým algoritmům:

- Změna velikosti mapy v případě změny velikosti okna (minimalizací prohlížeče nebo třeba překlopením mobilního zařízení) – velikost mapy se totiž automaticky nastavuje na velikost obrazovky, aby nebyla zbytečně velká a přesto ji mohl mít uživatel zobrazenou na celém displeji.

- Posun zobrazované výseče po vyžádání uživatelem nebo na nalezený objekt. Je nutné brát v potaz aktuální škálování mapy.

- Škálování mapy vyžádané uživatelem nebo dle nalezeného objektu. Je nutné brát v potaz aktuální zobrazovanou výseč.

- Zjištění pozice vykonaného gesta – je odvozována na základě pozice obalového elementu, zjišťování pozice z mapy dávalo vlivem různých implementací jiné výsledky napříč prohlížeči.

- Převedení zeměpisných souřadnic na místo na mapě – je realizováno prostým odvozením od souřadnic rohů mapy, zakřivení země je zanedbáno, pro dané účely tento přístup dostačuje.

V mapových podkladech jsou z důvodu velkého rozsahu dopodrobna na úrovni místností rozpracovány pouze první a třetí podlaží budovy T9 a na úrovni budov pouze dejvický kampus.

3.2.1.3 Rozdílné DOM u HTML a SVG

SVG je, co se týče návrhu DOM, komplikovanější, než HTML – zatímco u HTML stačí k hodnotě atributu v DOM přímo přistoupit, v SVG se musí zpravidla ještě o dvě úrovně níže, tedy třeba místo k `.width` se musí přistupovat k `.width.baseVal.value`. Při implementaci to bylo třeba brát v potaz.

3.2.1.4 Malá velikost displeje

Ačkoliv je toto omezení u mobilních zařízení zřejmé, nic to nemění na faktu, že se s ním musí neustále počítat a je třeba vybírat vhodné komponenty a ty vhodně rozvrhovat. Malá velikost displeje byla omezením i z opačného hlediska – prsty jsou u dotykových telefonů, vzhledem k velikosti displeje, poměrně velké a ovládací prvky tak musí být dostatečně rozměrné, což konzumuje další místo. Mým cílem proto bylo vizualizovat jen to nejnutnější.

3.2.1.5 Různé typy polohovacích zařízení

Různá zařízení poskytují různé ovládací prvky využívající různé principy, aplikaci je ale potřeba přizpůsobit pro všechny najednou. Jiné je ovládání pro myš (touchpad, trackpad...), jiné pro klávesy a jiné pro dotyková zařízení – právě s těmi je problém – v rámci použitých technologií ještě nebyl vytvořen žádný standard pro zpracování zpracování gest, proto bylo nutné umístit nad mapu výrazné ovládací prvky.

3.2.1.6 Dynamické načítání lokálních souborů

Z bezpečnostních důvodů došlo v minulosti v prohlížečích k zamezení načítání jakýchkoliv souborů, mimo JavaScriptu, z jiných zdrojů, než ze kterých stránka pochází. To v některých případech[27] přerostlo v zamezení dynamického načítání jakýchkoliv lokálních souborů – jejich zdroj nemá žádnou hodnotu (`null`), takže je nelze načíst. Aby byla zachována možnost offline využití aplikace na větší skupině zařízení, byla aplikace přepsána na jediné možné řešení – explicitní uvedení všech potenciálně vkládaných souborů, což značně omezilo modularitu aplikace.

[27] Příkladem je Google Chrome a jeho základ Chromium [11].

Například v případě SVG je široká nabídka volby – lze vložit JavaScriptem do elementů `img` (`<img src="map.svg" />`), `embed` (`<embed src="map.svg" />`), `object` (`<object type="image/svg+xml" data="map.svg" />`), `iframe` (`<iframe src="map.svg" />`), pokud je pak umožněna manipulace s obsahem pomocí JavaScriptu, tak jen velice neohrabaně. Další možností je vložit SVG přímo do HTML kódu – v případě dynamicky generovaného kódu by bylo vhodné celý soubor načíst a na místo vložit, jenže zde narážíme na problém z předchozího odstavce – některé prohlížeče to neumožní. Zbývá tedy poslední řešení – celý soubor převést do JavaScriptového řetězce, uložit do proměnné a vložit z ní. Jedině tak zle vložit do stránky SVG soubor, aby s ním pak bylo snadné manipulovat prostřednictvím jediného DOM.

3.2.1.7 Zpracování Cross-Origin zdrojů

Aplikace potřebuje získávat a zpracovávat cizí zdroje – aby poskytovala aktuální informace, nevystačí si jen s lokálním úložištěm, musí přímo k jejich původci. To je ale zpravidla v doméně použitých technologií komplikované – webové prohlížeče z bezpečnostních důvodů nepovolují k cizím zdrojům přístup (výjimkou je například JavaScript – více dále), takže je třeba danou situaci řešit.

SPARQL endpoint ze serverové aplikace, tedy hlavní předpokládaný zdroj informací, máme plně ve své režii, takže na něm můžeme povolit Cross-origin resource sharing (CORS), tedy technologii umožňující serveru zpřístupnit své zdroje i z jiné domény. Jediný problém je u špatné podpory některých prohlížečů.

Z již dříve popsaných důvodů, tedy malého rozsahu sémantické databáze, ale finální aplikace podporuje mimo přístupu k samotnému SPARQL endpointu i přímý přístup k původním zdrojům. Zde vyvstává problém – na cizích zdrojích CORS povolit nemůžeme, je tedy třeba najít jiné řešení. Tím je JSON with padding (JSONP), tedy postup umožňující získat přes JavaScriptový soubor (ten má výjimku) data ve formátu JSON obalená funkcí námi zvoleného jména, která je po stažení zdroje v prohlížeči vykonána, a tím jsou aplikaci předána data ke zpracování.

JSONP sám o sobě ale nic nevyřeší – je k němu stejně tak potřeba mít uzpůsobený server. Tady nám pomůže jednoduchá *proxy*, které zadáme požadavek na zdroj, ona ho získá a odešle nazpátek. I zde by nakonec nemuselo být JSONP potřeba, proxy zase bude v naší režii (v případě existence jiné vhodné, v cizí), a šlo by pak použít CORS; protože se ale jedná o dočasné řešení v době, kdy ještě některé menšinové prohlížeče CORS nepodporují, rozhodl jsem se si vyzkoušet implementaci další technologie.

Z veřejně k použití dostupných proxy lze využít například server Yahoo! využívaný pro *Yahoo! Query Language* (`http://developer.yahoo.com/yql/`) – na něm je založeno jQuery rozšíření cross-domain-ajax (`https://github.com/padolsey/jQuery-Plugins/`), dále je možné na vlastním ser-

veru použít nějaké hotové řešení, například *AJAX Cross Domain* (`http://www.ajax-cross-domain.com/`), nebo si vše od základu vytvořit sám.

Rozhodl jsem se pro třetí možnost – veřejně dostupná řešení nenabízejí potřebné možnosti konfigurace, mezi již hotovými řešeními se mi nepodařilo nalézt vhodné pro konkrétní použití, takže jsem si vytvořil proxy vlastní.[28]

Proxy je požádána o zprostředkování zdroje, na základě *whitelistu* je rozhodnuto o jeho poskytnutí – z bezpečnostních důvodů je zamezen přístup ke všem mimo nutně potřebných serverů. Zdroj proxy stáhne, převede na přednastavené kódování (v rámci JavaScriptu nejsou převody snadné, dojde tak k výraznému usnadnění mobilní aplikaci) a před odesláním odpovědi je zdroj pročištěn – za využití knihovny *HTML Purifier* je zdroj zvalidněn a je z něj odstraněno vše, co není explicitně dalším *whitelistem* povoleno – skripty, obrázky, rámce, další zdroje, potenciálně škodlivé atributy a elementy... V případě chyby je uživateli (mobilní aplikaci) navráceno příslušné chybové hlášení.

3.2.1.8 Uživatelské rozhraní

Tvorba uživatelského rozhraní mobilní aplikace nepřinesla, mimo tvorby mapy a zpracování cizích zdrojů, žádné zvlášť komplikované úkony. Tvorba mapy byla již popsána. Zpracování cizích zdrojů bylo vyžadováno pro jejich různorodost – každý poskytoval informace v jiném formátu, transformoval jsem je tedy do obdobnějších reprezentací vhodných pro mobilní zařízení. Pro ilustraci přikládám obrázek 3.1 znázorňující uživatelské rozhraní mobilního průvodce.

Obrázek 3.1: Snímek obrazovky aplikace v prohlížeči platformy Android

[28]Byla tedy přidána další serverová aplikace do již tak terminologicky složité situace. Z tohoto důvodu se u zmíněného serveru držím označení *proxy*.

3.2.2 Komplikace

Během implementace mobilní aplikace se objevily i komplikace, mimo očekávaných, již zmíněných v předchozí části, mezi ně patřilo:

3.2.2.1 Chyby prohlížečů v implementaci specifikací

Při práci jsem několikrát narazil na situaci, kdy určitá vlastnost v jednom prohlížeči fungovala, ve druhém nikoliv, a přesto bylo vše implementované správně podle specifikace. V takovém případě pak bývala chyba na straně prohlížeče, který určitou specifikaci špatně implementoval. Situace nastávala převážně u hodně specifických případů. Příkladem může být chybějící podpora pro nastavení transformace SVG elementu JavaScriptem (nutné pro práci s mapou) v prohlížeči Google Chrome [7], tento problém ale lze obejít vlastním sestavením hodnoty celého atributu a přímým vložením.

3.2.2.2 Implementace SVG

Implementace SVG je v desktopových prohlížečích na velmi dobré úrovni již nějakou dobu,[29] v nedávné minulosti se ale podpora značně zlepšila i na poli mobilních zařízení [1].

U mobilních zařízení nepřekvapuje chybějící implementace některých neklíčových možností SVG. V této oblasti v současné době rozsáhlému nasazení nepřeje SVG nepodporující hojně využívaný nativní webový prohlížeč Androidu verze 2; od verze 3 je podpora zavedena. Problém lze obejít využitím jiného dostupného prohlížeče.

Překvapením byla pomalá, ačkoliv jinak velmi rozsáhlá, implementace SVG v prohlížeči Mozilla Firefox – po některých operacích, v mém případě to byla třeba změna hodnot atributu `viewBox` SVG elementu (posunutí mapy), prohlížeč vyžaduje překreslení celého obrázku, což je velmi zdlouhavé a zabraňuje to plynulému posunu mapy z aktuální na novou pozici, byť by to bylo vhodné pro lepší uživatelovo zorientování se. Tento nedostatek je známý již delší dobu [15].

Nalezené problémy byla snaha obejít, ne vždy se to ale bohužel podařilo.

3.2.2.3 Neexistující DNS server

Po nasazení proxy na server `http://webdev.fit.cvut.cz/` byl zjištěn nepříjemný problém – ačkoliv přes proxy nasazené na jiné testovací servery bylo připojení z mobilní aplikace promptní, přes server webdev trvalo pokaždé přibližně třicet a půl vteřiny. Pravidelnost takového zvláštního času mě vedla ke snaze problém identifikovat. Nakonec jsem z výpisů komunikace zjistil, že je problém ve webdevem využívaných DNS serverech – trvalo jim vždy půl

[29] Posledním významným desktopovým prohlížečem podporujícím SVG se stal s dlouhým odstupem verzí 9.0 Internet Explorer [1], předtím pro něj ale byly dostupné vhodné pluginy.

minuty, než převedly URL na IP adresu. Problém jsem nahlásil IT oddělení FIT a za pár dní byl odstraněn – mezi DNS servery byl nastavený záložní, provozovaný třetí stranou, ten byl ale bez upozornění zrušen.

KAPITOLA 4

Testování

Testování (*testing*) provázelo vzniklé aplikace po celou dobu vývoje – od evaluece požadavků na počátku, až po akceptační testy na konci. Testování probíhala, byť v jinou dobu, obdobně u serverové i mobilní aplikace, pro větší přehlednost jejich popis tedy sloučím.

4.1 Funkční testování

Funkční testování (*functional testing*) reprezentuje skupinu různorodých testů, které ověřují, zda byly splněny všechny funkční požadavky na systém. Jeho výsledky jsou tedy zpravidla velmi kategorické. Nevyužil jsem proto formálních metodik – jejich závěry by měly být shodné s těmi mými založenými na prosté důkladné kontrole projektu.

4.1.1 Testování programátorem

Testování programátorem zahrnuje širokou skupinu testů prováděných přímo programátorem v průběhu vývoje. Patří mezi ně jednotkové testování (*unit testing*), revize kódu (*code review*), testování mezních hodnot (*boundary testing*)...

Každou přidávanou jednotku jsem testoval mezními hodnotami. V průběhu vývoje jsem opakovaně revidoval v předchozích iteracích napsaný kód a v případě potřeby ho komentoval a refaktoroval pro větší kvalitu (lepší čitelnost, deduplikakace, zjednodušení...). Tato činnost sloužila k eliminaci lokálních chyb v kódu a přinášela pozitivní výsledky. Po změně vždy proběhlo nutné otestování.

4.1.2 Integrační testování

Integrační testování (*integration testing*) bylo prováděno u všech vyvíjených aplikací. Po přidání nové funkcionality byla vždy příslušná část aplikace důkladně otestována, aby bylo zamezeno situacím, kdy nově zanesené úpravy poškodí již existující funkcionality aplikace – takto vzniklé a neošetřené chyby totiž bývají velmi záludné – vývojář otestoval všechny části, nenalezl v nich žádný problém, takže je považuje za korektní, možnost chyb daných integrací ho nenapadne a při pozdějším výskytu problému pak hledá řešení na špatných místech, protože skutečné místo považuje za již důkladně otestované.

4.1.3 Akceptační testování

Akceptačního testování (*acceptance testing*) ověřuje, zda byly do požadované míry naplněny všechny požadavky a po jeho úspěšném dokončení je systém zpravidla předán.

Akceptační testování bylo prováděno bez formální metodiky vedoucím práce, inženýrem Havrylukem, a mnou, jako osobou mající největší zájem na úspěšném naplnění všech požadavků. Systém byl akceptován.

4.2 Nefunkční testování

Nefunkční testování (*non-functional testing*) zahrnuje skupinu testů ověřujících naplnění nefunkčních požadavků na systém. Zde se z principu očekávají výsledky více vágní, než v případě funkčního testování, proto jsem také věnoval podstatně více času na provedení formalizovanějších testů. Jeho výsledky by měly korelovat s kvalitou vytvořeného produktu.

4.2.1 Testování kompatibility

Testování kompatibility (*compatibility testing*) je proces ověřující správný chod systému v různých prostředích – zařízeních, operačních systémech, běhových prostředích... Serverová aplikace je nasazena na konkrétním serveru a s častým přemisťováním se nepočítá, věnovat se proto budu v této sekci pouze mobilní aplikaci, která je na tom přesně opačně – bude nasazena na zařízeních uživatelů, která jsou vzájemně dosti odlišná.

4.2.1.1 Charakteristika testu

Testování bylo v omezeném počtu prohlížečů prováděno po celou dobu vývoje, ve zbylých až v době finalizace. Testovalo se v prohlížečích:

- Mozilla Firefox (`http://www.mozilla.org/firefox/`).

- Chromium (`http://chromium.org/`).[30]
- Rekonq (`http://rekonq.kde.org/`).
- Konqueror (`http://www.konqueror.org/`).
- Internet Explorer (`http://ie.microsoft.com/`).
- Prohlížeč platformy Android (`http://android.com/`).[31]
- Opera Mobile (`http://www.opera.com/mobile/`).
- Opera Mini (`http://www.opera.com/mini/`).
- Internet Explorer Mobile (`http://www.microsoft.com/windowsphone/en-us/features/default.aspx#internet-explorer`).

Mimo výše uvedených navíc nabídl pan inženýr Havryluk, vedoucí práce, otestování v následujících prohlížečích:

- Google Chrome for Android (`https://www.google.com/intl/en/chrome/android/`).
- Prohlížeč platformy Android (`http://android.com/`).[32]

4.2.1.2 Průběh a shrnutí testu

V následující části jsou popsány nalezené problémy jednotlivých prohlížečů, pokud není uvedené jinak, aktuálních verzí:

Mozilla Firefox

Mozilla Firefox po pozměnění vynucuje překreslení SVG, které je navíc poměrně pomalé, snažil jsem se proto algoritmy upravit tak, aby SVG zbytečně nepozměňovaly. Jinak vše funguje, jak má.

Chromium

Nelze nastavit transformaci SVG elementu, problém ale lze obejít méně elegantním způsobem. Vše funguje, jak má.

[30] Chromium je základem prohlížeče Google Chrome, dá se tedy předpokládat, že ten bude aplikaci podporovat na stejné úrovni.
[31] Verze 2 a 4.
[32] Verze 3.

Rekonq

Prohlížeč nepodporuje geolokační API, takže nelze určit polohu uživalete. Prohlížeč také výchozím fontem zobrazuje velmi úzké bílé znaky, takže nemusí být tolik přehledné formátování například ve SPARQL konzoli. Jinak vše funguje, jak má.

Konqueror

Prohlížeč ve výchozím *renderovacím* jádru, KHTML, nepodporuje přímé vkládání SVG do HTML, nefunguje v něm tedy navigace – uživatel je na toto omezení upozorněn. Jinak vše funguje, jak má.

Internet Explorer

Do verze 9 neměl Internet Explorer bez přidaných rozšíření dostatečnou podporu SVG a nepodporoval tak navigaci; od verze 9, včetně, je vše v pořádku. V tomto prohlížeči jsem také narazil na chybu oproti specifikaci ECMAScriptu:[33] není zde povoleno pojmenovat vlastnost objektu rezervovaným slovem `default`, což by mělo být zakázané pouze na místech označených jako *Identifiers*, nikoliv tedy jako *Identifier Names* [16], přejmenování proměnné zde problém vyřešilo. Jinak vše funguje, jak má.

Opera

V Opeře se při vyhledávání v rámci navigace nezobrazovaly objekty uvnitř budov, nejprve jsem z toho velmi chybně vinil špatnou implementaci průhlednosti,[34] po dlouhém zkoumání jsem ale zjistil, že Opera nedokáže v některých případech vlivem odlišného DOM SVG pracovat přímo se třídami SVG elementů, ale musí se k jejich hodnotám v DOM přistupovat skrze delší cestu – většina ostatních prohlížečů si s tímto problémem poradila. Problematické algoritmy byly přepracovány a problém odstraněn. Vše funguje, jak má.

Prohlížeč platformy Android

Do verze 2, včetně, nepodporuje prohlížeč SVG – tudíž nefunguje navigace, to se ale ve verzích 3 a 4 změnilo a podpora celé aplikace je tak bezproblémová. Vše funguje, jak má.

[33] Jazyk JavaScript je z právních důvodů standardizován pod názvem ECMAScript, v ptáci se držím více zažitého názvu.

[34] Nejprve jsem se s tímto problémem setkal u Opera Mobile, kde by to bylo pochopitelné.

Opera Mobile

V Opeře Mobile se vyskytoval stejný problém, jaký byl odhalen u *desktopové* verze – nezobrazovaly se objekty uvnitř budov. Odstraněním problému v *desktopové* verzi byl odstraněn i zde. Vše funguje, jak má.

Opera Mini

Prohlížeč Opera Mini nepodporuje přímé vkládání SVG do HTML, nefunguje v něm tedy navigace – uživatel je na toto omezení upozorněn. Také tento prohlížeč nedokáže zobrazit tabulátor v textových polích, například ve SPARQL konzoli – byly tedy nahrazeny za mezery. Jinak vše funguje, jak má.

Internet Explorer Mobile

Testována byla pouze verze 6, v té ještě nebyla přidána podpora SVG, která je již v aktuální verzi přítomná – v testované verzi tedy nefunguje navigace. Také se zde projevovala, jako v *desktopové* verzi, již opravená, chybná implementace omezení rezervovaných slov. Jinak vše funguje, jak má.

Google Chrome for Android

Google Chrome for Android je teprve v betaverzi a některé věci tak ještě nejsou úplně odladěné. V aplikaci se projevila chyba, jinak velmi užitečné, funkce pro automatické přibližování v reakci na stisknutí odkazu – uživateli by tak mělo být usnadněno jeho finální potvrzení, aniž by byl omylem aktivován jiný odkaz v blízkosti zamýšleného, v tomto případě je ale pouze zvětšováno a odkaz nelze vyvolat. Zdá se, že jinak vše funguje, jak má.

4.2.2 Výkonnostní testování

Výkonnostní testování (*performance testing*) se zabývá, jak již název napovídá, testováním odezvy a stability systému za určitého zatížení. Lze ho dělit na zátěžové testování (*load testing*), *stress* testy[35], testování výdrže (*endurance testing*), testování konfigurací (*configuration testing*)...

Výkonnostní testování má smysl provádět jen na větších systémech dostupných většímu počtu potencionálních uživatelů – což je v případě této práce pouze SPARQL endpoint *serverové aplikace*.

Pro testování SPARQL endpointů jsou zpravidla využívány tyto *benchmarky*:

- The Lehigh University Benchmark (LUBM) (`http://swat.cse.lehigh.edu/projects/lubm/`)
- SP²Bench Download Section (SP²Bench) (`http://dbis.informatik.uni-freiburg.de/?project=SP2B`)

[35] Český ekvivalent anglického *stress* se v tomto případě pravděpodobně nevyužívá.

- Berlin SPARQL Benchmark (BSBM)
  (`http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/`)

První jmenovaný *benchmark* pochází z Lehigh University, poslední verze je přibližně z roku 2005 a testování probíhá v doméně univerzity, tedy přesně v té, kterou potřebujeme. Pro potřeby tohoto *benchmarku* vznikla ontologie zkoumaná v části 1.5.1.1 na straně 22.

Druhý jmenovaný pochází z Universität Freiburg, poslední verze je z roku 2009 a testování probíhá v doméně počítačové bibliografie, která je doméně našeho systému alespoň vzdáleně blízká.

Poslední jmenovaný pochází z Freie Universität Berlin, aktuální verze je z roku 2011 a testování probíhá v doméně elektronického obchodního systému.

Po hrubém prozkoumání jmenovaných *benchmarků* jsem se rozhodl vydat jinou cestou – vytvořit si vlastní jednoduchý test zkoumající pouze chování SPARQL endpointu v zatížení o několik řádů větším, než maximálním předpokládaném. Výše uvedené *benchmarky* mi tedy nakonec posloužily pouze jako zdroj inspirace, přesto je zde nechávám případným zájemcům pro pozdější podrobnější přezkoumání.

4.2.2.1 Charakteristika testu

K testování byl použit jednoduchý konzolový nástroj ApacheBench (`http://httpd.apache.org/docs/2.4/programs/ab.html`), jehož prostřednictvím byl SPARQL endpoint vystavován kontinuální paralelní zátěži. Tento nástroj stačil pouze po nějakou dobu – v případě extrémních testů se projevilo jeho historické omezení – umožňuje práci pouze s protokolem HTTP 1.0, zatímco Node.js, nad kterým je postavený SPARQL endpoint, naopak využívá perzistentní připojení HTTP 1.1, takže v ApacheBench dojde po nějaké době k vypršení jednoho z několika vnitřních časových limitů a následnému ukončení.

SPARQL endpoint byl pro potřeby testování umístěn na virtuálním stroji řešení VirtualBox (`http://www.virtualbox.org/`) o jednom procesoru frekvence 2 GHz, 2 GB operační paměti a dostatečně rychlém síťovém připojení k hostitelskému počítači, ze kterého byly testy spouštěny.

Testy byly prováděny na bázi dat obsahující všechna data dostupná prostřednictvím implementovaných obsluhovačů zdrojů, nebylo prováděno *cachování* a dotazovalo se ukázkovým příkladem uvedeným ve SPARQL konzoli mobilní aplikace (výběr všech jmen a k nim příslušných emailových adres):

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
    ?person a foaf:Person.
    ?person foaf:name ?name.
    ?person foaf:mbox ?email.
}
```

Charakteristika	1c/1000n[a]	10c/1000n[a]	100c/1000n[a]
Doba testu [s]	97.000	92.860	92.767
Dokončené požadavky	1000	1000	1000
Chybné požadavky	0	0	0
Požadavky/s	10.31	10.77	10.78
Doba požadavku[b][ms]	97.00	928.60	9276.69
Doba požadavku[c][ms]	97.00	92.86	92.77
Rychlost přenosu [KB/s]	140.40	146.66	146.81
Nejkr. doba požadavku [ms]	75	630	7556
Prům. doba požadavku [ms]	97	926	9246
Medián doby požadavku [ms]	87	908	9231
Nejd. doba požadavku [ms]	483	1470	10902

[a] Počet souběžných požadavků najednou v celkovém počtu.
[b] Průměrná doba požadavku z pohledu uživatele.
[c] Průměrná doba požadavku z pohledu serveru.

Tabulka 4.1: Průběh výkonnostního testování

4.2.2.2 Průběh testu

Test byl prováděn příkazem `ab -n 1000 -c 100 -H "Accept: application /json" -e benchmark http://server:8080/sparql?query=PREFIX+f...`, kde parametr `n` značí počet požadavků, `c` počet souběžných požadavků, `H` hlavičku požadavku (jinak by server navrátil data ve formátu XML, místo využívaného JSON), `e` soubor, do kterého se zapíšou výsledky a zbytek příkazu tvoří zakódovaný SPARQL požadavek.

Testování bylo provedeno se třemi různými počty souběžných požadavků (1, 10, 100) vždy z celkového počtu 1000 (navýšení na 10000 už nemělo vliv).

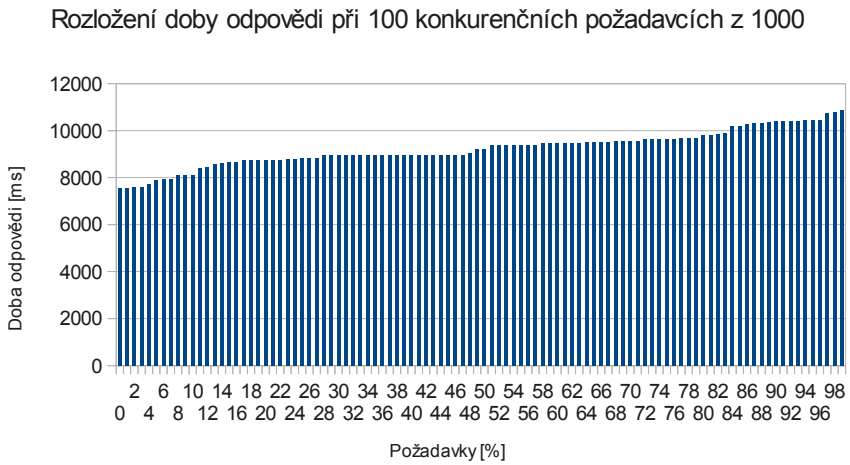
Výsledky jednotlivých měření jsou zaznamenány v tabulce 4.1. Na obrázku 4.1 je znázorněn graf procentuálního rozložení doby odpovědi pro 100 souběžných požadavků z celkového počtu 1000 – ostatní průběhy byly velmi podobné.[36]

V průběhu testu nebyl pozorován znatelný nárůst spotřeby operační paměti, na druhou stranu systém vytížil přidělený procesor na maximum.

4.2.2.3 Shrnutí testu

Bylo ověřeno, že je možné provozovat systém bez jakýchkoliv obav, bude dostačovat pro předpokládaný provoz. Bohužel se mi nepodařilo zjistit, kolik souběžných požadavků by muselo být na server odesláno, aby vznikly vážné

[36] Na přiloženém CD se nacházejí ostatní grafy a další podkladové materiály tohoto testu.



Obrázek 4.1: Graf rozložení doby odpovědi

komplikace, z naměřených výsledků je ale patrné, že k tomu, mimo případy útoků, nikdy za stávajícího stavu nedojde.

V případě desíti požadavků za vteřinu, tedy s odezvou pod jednu vteřinu, uživatelé stále nepocítí žádné výkonnostní problémy, v případě sta požadavků za vteřinu, s odezvou pod deset vteřin, by dle [10] stále měli být schopní se soustředit na řešenou úlohu. Velmi pozitivní je nulová chybovost požadavků i při vysoké zátěži.

Pokud by za nějakých nepředpokládaných okolností nastala situace vyžadující navýšení výkonu, lze postupovat jednou z následujících cest:

- Přesunout systém z virtuálního stroje do plnohodnotného prostředí.
- Začít *cachovat* odpovědi (téměř každá může být uložena a, při shodném dotazu v krátkém časovém horizontu, navrácena).
- Vhodněji indexovat databázi.
- Optimalizovat SPARQL endpoint.
- Distribuovat zátěž.

Předchozí cesty lze samozřejmě kombinovat a jejich výčet rozhodně není kompletní.

4.2.3 Testování použitelnosti

Na použitelnost (*usability*) byl kladen důraz hlavně v případě mobilní aplikace, u serverové jsem se naopak použitelností příliš nezabýval – nemá obsluhu,

případně zaškoleného správce. Testování použitelnosti (*usability testing*) bylo prováděno po celou dobu vývoje, formalizovaně ale až těsně před jeho dokončením, aby byla finální aplikace k uživatelům co nejpřívětivější.

4.2.3.1 Charakteristika testu

Testování probíhalo na čtyřech osobách podle následujícího scénáře: Účastníkovi byla předložena mobilní aplikace a byl mu ponechán čas na zběžné seznámení. V případě, že měl účastník vlastní vhodné zařízení, testování probíhalo na něm, jinak mu bylo poskytnuto jiné včetně pomoci s ovládáním, aby nebyly výsledky testu zbytečně zkreslovány. Poté byly účastníkovi zadány následující čtyři úkoly pokrývající předpokládané využití aplikace:

UT1 Vyhledejte pozici budovy T9. Úloha sleduje orientaci v základní struktuře aplikace.

UT2 Vyhledejte jídelníček menzy Masarykova kolej. Úloha sleduje schopnost využít průvodce.

UT3 Nacházíte se u vchodu to budovy T9, najděte nejbližší občerstvení. Úloha sleduje pokročilou obsluhu navigace.

UT4 Zjistěte si podrobnosti libovolného problému prostřednictvím SPARQL dotazu. Úloha sleduje schopnost využití SPARQL konzole.

Po celou dobu byla osoba pozorována a po dokončení úkolů s ní byla provedena evaluace. Testování přineslo velký počet podnětů, které byly následně realizovány.

4.2.3.2 Charakteristika účastníků

Testování použitelnosti bylo provedeno na čtyřech účastnících charakterizovaných v tabulce 4.2.

4.2.3.3 Průběh testu

Osoby v tabulce 4.2 charakterizující účastníky testování jsou seřazeny chronologicky. Mezi jednotlivými testy byly ponechány krátké rozestupy, ve kterých jsem analyzoval jejich průběh a snažil se odstranit alespoň některé nalezené problémy – jednotlivé průběhy tedy mezi sebou nelze příliš porovnávat.

P01

UT1 Nejprve se snaží nalézt cestu k budově v sekci *Průvodce*, po projití si ale uvědomuje, že by měla hledat v sekci *Navigace* – zpočátku ji považovala za mapu stránek. Poté hledaný objekt vyhledává posunem mapy.

Osoba	Věk	Pohlaví	Povolání	Zkušenosti s počítačem, s mobilním zařízením
P01	25	žena	výzkum	pokročilá uživatelka, mírně pokročilá
P02	23	žena	studentka	pokročilá uživatelka, bez zkušeností
P03	21	muž	student	mírně pokročilý uživatel, bez zkušeností
P04	21	žena	studentka	občasná uživatelka, pokročilá

Tabulka 4.2: Charakteristika účastníků testování použitelnosti

UT2 Nalézá bez sebemenších problémů.

UT3 Snaží se nalézt občerstvení na mapě, nenalézá, přechází do sekce *Průvodce*, opět nenalézá vrací se do *Navigace*, ve vyhledávání zadává slovo *bufet* a nalézá, pouze chvíli nevidí na mapě – velmi malé a nevycentrované.

UT4 Nezná syntaxi dotazovacího jazyka SPARQL, prohlíží si ukázkový dotaz a vykonává ho.

Poznámky Při pohybu na rozsáhlé úvodní stránce (nechala zobrazit příliš mnoho informací) se ztrácí v obsahu, nedaří se jí se jednoduše dostat nahoru k ovládání. Nečte nápovědu. Nevyužívá zpočátku vyhledávání u mapy.

P02

UT1 Zpočátku ji nenapadá hledat v sekci *Navigace*, vzápětí se tam ale přesouvá a budovu pohybem po mapě nalézá.

UT2 Nalézá bez sebemenších problémů.

UT3 Přechází do navigace, prochází mapou, občerstvení nenachází, zkouší ho proto vyhledat a nyní ho už nachází.

UT4 Nalézá záložku SPARQL a vykonává ukázkový dotaz, více se o SPARQL nezajímá.

Poznámky Nečte nápovědu. Nevyužívá vyhledávání u mapy.

P03

UT1 Ihned vstupuje do sekce *Navigace*, mapou ale prochází manuálně, tlačítky – ovládání gesty nezaregistruje. Nalézá.

UT2 Nalézá bez sebemenších problémů.

UT3 Přechází mezi sekcemi *Navigace* a *Průvodce*, nakonec používá vyhledávání, ve kterém nejprve zadává slovo *vobčerstvení*, nakonec ale uznává, že použití slova *občerstvení* bude mít větší šanci na úspěch a nalézá.

UT4 Vykonává předpřipravený dotaz a dále se o SPARQL nezajímá.

P04

UT1 Sekci *Navigace* nalézá okamžitě, budovu vyhledává, není si ale hned jistá, co je je výsledkem – v mapě se nezobrazilo překrytí okolí vyhledaného místa.

UT2 Nalézá bez sebemenších problémů.

UT3 Občerstvení nalézá ihned, dále zkouší vyhledat slova *bufet* a *jídlo* a na obě také nalézá výsledky.

UT4 Vykonává předpřipravený dotaz a dále se o SPARQL nezajímá.

4.2.3.4 Shrnutí testu

Testování použitelnosti přineslo mnoho konstruktivních podnětů k vylepšení aplikace, naprostá většina se jich týkala navigace. Patří mezi ně především:

Motivovat k přečtení nápovědy

Je vhodné motivovat uživatele k přečtení krátké nápovědy vysvětlující základní principy práce s aplikací. Všem účastníkům testování by to velmi prospělo. Může tak být učiněno například na webové stránce nabízející aplikaci ke stažení.

Našeptávání

V průběhu testů použitelnosti byl na základě zpětné vazby od uživatelů realizován podstatně lepší vyhledávací algoritmus s našeptáváním.

Zvětšení vstupního pole navigace

Vstupní pole vyhledávacího pole navigace bylo zvětšeno na základě dvou postřehů vyvstalých při testování použitelnosti: Pole bylo málo výrazné, takže uživatelé vyhledávání přehlíželi a bylo poměrně malé, takže se do něj na dotykových zařízeních špatně trefovalo prstem.

Odřezání bílých znaků z okolí dotazu

V průběhu testování byl několikrát nevědomky zadán k vyhledání výraz, který předcházel nebo následoval bílý znak, tudíž neodpovídal tomu v databázi, takže nebylo nic nalezeno. Problém jsem se rozhodl řešit odstraněním takových znaků.

Přesun na nalezené objekty

Zpočátku uživatelé nezaregistrovávali nalezení hledaného objektu – pouhé zobrazení šipky někde na mapě nebylo dostatečné. Implementoval jsem proto algoritmus pro přesun na vyhledávané místo a následné přeškálování mapy tak, aby byl zobrazen hledaný objekt i jeho okolí.

Odebrání tlačítka Centrovat

Tlačítko pro vycentrování mapy se po přidání předchozí funkce ukázalo jako zbytečné – uživatelé se už nemuseli při neúspěšném hledání nalezeného objektu vracet na původní místo, aby to zkusili od začátku, ale vyhledávaný objekt jim byl automaticky vycentrován.

Nápověda k regulárním výrazům

Původně v nápovědě nebyla žádná zmínka o regulárních výrazech a jejich užití, ukázalo se to jako potřebné, proto byla přidána.

4.2.4 Srovnávací testování

Srovnávací testování (*comparative testing*) proběhlo pouze velmi zběžně – neexistuje pravděpodobně totiž žádný jiný systém mající podobně specifické cíle. Pro potřeby testování jsem proto systém rozdělil na jednotlivé části tak, jak byly vyvíjeny – serverovou a mobilní aplikaci. V případě serverové aplikace se mi nepodařilo objevit žádný konkurenční systém, na poli mobilních aplikací jich několik doménou obdobných již je, přesto je každá zaměřena na něco jiného.

Porovnávány byly aplikace Průvodce ČVUT FEL (`http://lr.czechian.net/j2me/`) a Studentův průvodce po fakultě (`http://webdev.felk.cvut.cz/~molnaja2/`). Je vhodné podotknout, že data první ze jmenovaných aplikací dávno pozbyla na platnosti a pro druhou aplikaci podklady pokrývající

celou doménu ani nebyly vytvořeny. Za porovnání by určitě stály i vznikající aplikace z projektu ČVUT navigátor (`http://navigator.fit.cvut.cz/`), ty ale stále nejsou zveřejněny.

Vzhledem k výše uvedeným důvodům nebyla pro srovnání stanovena řádná metrika, ale je ponecháno na interpretovi, aby si z nalezených rozdílů vyvodil pro své potřeby patřičné závěry. Výsledky srovnání se nacházejí v tabulce 4.3.

Charakteristika	Průvodce ČVUT FEL	Studentův průvodce po fakultě	Průvodce FIT ČVUT
Technologie	J2ME	XHTML, ECMAScript…	HTML5
Platformy	Všechny s Java ME	Všechny s webovým prohlížečem	Všechny s lepším webovým prohlížečem
Stav vývoje	Předčasně ukončený 2006	Ukončený 2010	Ukončený 2010
Podklady	Jiné[a]	Jiné, malé[b]	Dostatečné[c]
Velikost (KB)	30	1900[d]	340[d]
Licence	Freeware	GNU/GPLv3	GNU/GPLv3
Průvodce	Navigace, dopravní spojení	Navigace	Navigace, studium, menzy, kontakty…
Objekty navigace	Budovy, místnosti	Místnosti, body zájmu[e]	Budovy, místnosti, body zájmu[e]
Vyhledávání objektů	Označení místnosti	Označení místnosti, staré označení, zažitý název, jméno vyučujícího	Označení místnosti/budovy, neoficiální označení, body zájmu
Možnosti navigace	Popis, obrys budovy	Plán podlaží, RegEx[f]	Plán kampusu/podlaží, aktuální pozice, RegEx[f]
Další vlastnosti		Nápověda	Nápověda, SPARQL konzole

[a] Neaktuální. Budovy FEL ČVUT – T2, Z2, Z4, KN:E…
[b] Pouze vzorek – první a druhé podlaží na Karlově náměstí.
[c] Rozsáhlý vzorek – dejvický kampus nahrubo a první a třetí podlaží budovy T9 dopodrobna.
[d] Velikost je brána z aktuálního stavu, s kompletními podklady naroste.
[e] Občerstvení, toalety, výtahy…
[f] Regulární výraz.

Tabulka 4.3: Srovnání s ostatními mobilními aplikacemi

Závěr

TODO Zhodnocení splnění cílů DP.

TODO Doporučení pro další pokračování. více zdrojů čerpat data pouze z endpointu podrobnější mapy

TODO Shrnutí výsledků a vlastního přínosu. Jedním z nejdůležitějších poznatků, které mi diplomová práce dala, je nepodceňovat rizika plynoucí z nových, neznámých nebo jen důkladně nezdokumentovaných technologií. Toto je pravděpodobně můj poslední veliký projekt založený pouze na pro mě nových Neznamená to samozřejmě, že o využití takových technologií přestanu uvažovat, to v žádnem případě ne, pouze o nich budu uvažovat podstatně důkladněji.

Literatura

[1] Deveria, A.: When can I use.... 2012, http://caniuse.com/, stav z 1. 5. 2012.

[2] *Průvodce prváka po ČVUT*. IAESTE ČVUT Praha, 2011.

[3] Firtman, M.: Mobile HTML5. 2012, http://mobilehtml5.org/, stav z 1. 5. 2012.

[4] Gayo, J. E. L.: University Ontology. 2011, http://purl.org/weso/uni, stav z 10. 3. 2012.

[5] Guo, Y.: Lehigh University Benchmark. http://swat.cse.lehigh.edu/projects/lubm/, stav z 1. 5. 2012.

[6] Hendler, J.; Heflin, J.; Luke, S.: SHOE. 2001, http://www.cs.umd.edu/projects/plus/SHOE/, stav z 1. 5. 2012.

[7] Laag, L.: SVG transform attribute does not work when manipulated from JavaScript. Září 2010, http://code.google.com/p/chromium/issues/detail?id=55010, stav z 1. 5. 2012.

[8] Molnár, J.: *Mobilní navigační systém pro FEL ČVUT*. České vysoké učení technické v Praze, Fakulta elektrotechnická, Květen 2010.

[9] Molnár, J.; Garrote, A.: rdfstore-js – Issue #26. 2012, https://github.com/antoniogarrote/rdfstore-js/issues/26, stav z 7. 5. 2012.

[10] Nielsen, J.: Size Limits for Web Pages. 1997, http://www.useit.com/alertbox/sizelimits.html, stav z 7. 5. 2012.

[11] Schuh, J.: Allow a directory tree to be treated as a single origin. Červen 2010, http://code.google.com/p/chromium/issues/detail?id=47416, stav z 7. 5. 2012.

[12] Styles, R.; Shabir, N.: Academic Institution Internal Structure Ontology (AIISO). 2008,
http://purl.org/vocab/aiiso/schema, stav z 1. 5. 2012.

[13] MongoDB. 2012,
http://www.mongodb.org/, stav z 5. 5. 2012.

[14] Node.js. 2012,
http://nodejs.org/, stav z 5. 5. 2012.

[15] Bugzilla. 2012,
https://bugzilla.mozilla.org/.

[16] ECMAScript Language Specification. Červen 2011,
http://www.ecma-international.org/publications/standards/Ecma-262.htm, stav z 7. 5. 2012.

[17] Node.js Manual & Documentation. 2012,
http://nodejs.org/api/all.html#all_buffer_1, stav z 8. 5. 2012.

[18] Zemmouchi-Ghomari, L.: Higher Education Reference Ontology (HERO). 2012,
http://sourceforge.net/projects/heronto/, stav z 1. 5. 2012.

PŘÍLOHA A

Seznam použitých zkratek

AIISO Academic Institution Internal Structure Ontology.

API application programming interface.

BSBM Berlin SPARQL Benchmark.

BSD Berkeley Software Distribution.

CD Compact Disc.

CORS Cross-origin resource sharing.

CSS Cascading Style Sheets.

cURL Client for URLs.

ČVUT České vysoké učení technické v Praze.

DNS Domain Name System.

DOM Document Object Model.

DP diplomová práce.

FEL Fakulta elektrotechnická.

FIT Fakulta informačních technologií.

FOAF Friend of a friend.

GNU GNU's Not Unix!.

GPL General Public License.

HERO Higher Education Reference Ontology.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

IP Internet Protocol.

IT informační technologie.

J2ME Java 2 Platform, Micro Edition.

JSON JavaScript Object Notation.

JSONP JSON with padding.

Kate KDE Advanced Text Editor.

KDE K Desktop Environment.

KIO KDE Input/Output.

KOS komponenta studia.

LODE Linking Open Descriptions of Events.

LUBM The Lehigh University Benchmark.

MAR měření a regulace.

MHD městská hromadná doprava.

NTK Národní technická knihovna.

Org An organization ontology.

OS operační systém.

OWL-Time An OWL Ontology of Time.

PHP Hypertext Preprocessor.

RDF Resource Description Framework.

SHOE Simple HTML Ontology Extensions.

SP²Bench SP²Bench Download Section.

SPARQL SPARQL Protocol and RDF Query Language.

SVG Scalable Vector Graphics.

URL Uniform Resource Locator.

UTF Unicode Transformation Format.

XML Extensible Markup Language.

PŘÍLOHA B

Instalační a uživatelská příručka

TODO spuštění - příkazy

B.1 Serverová aplikace

B.2 Mobilní aplikace

PŘÍLOHA C

Obsah přiloženého CD

Přiložené CD je strukturováno:

- `/`
  - `index.html` ..... popis obsahu CD
  - `text` ..... text diplomové práce
    - `DP_Molnár_Jan_2012.pdf` .. text diplomové práce ve formátu PDF
  - `mobil` ..... zdrojové kódy mobilní aplikace
    - `index.html` ..... spouštěcí soubor mobilní aplikace
  - `proxy` ..... zdrojové kódy proxy mobilní aplikace
    - `index.php` ..... spouštěcí soubor proxy mobilní aplikace
  - `server` ..... zdrojové kódy serverové aplikace
    - `index.js` ..... spouštěcí soubor serverové aplikace
  - `navrh` ..... dokumenty vzniklé při návrhu
    - `diagramy` ..... diagramy obou aplikací
    - `prototypy` ..... prototypy mobilní aplikace
  - `testovani` ..... dokumenty vzniklé při testování
    - `vykonnostni` ..... dokumenty vzniklé při výkonnostním testování
  - `podklady` ..... podklady diplomové práce
    - `mapy` ..... mapy vytvořené pro potřeby práce
    - `ontologie` ..... použité ontologie
  - `jine` ..... další soubory spjaté s prací
    - `obrazovky` ..... ofocené obrazovky mobilní aplikace
    - `pruvodce CVUT FEL` ..... aplikace Průvodce ČVUT FEL

PŘÍLOHA D

TODO

2 Doporučení pro další pokračování.

3 Shrnutí výsledků a vlastního přínosu.

4 spuštění - příkazy

1 Zhodnocení splnění cílů DP.

0 Zrevidovat a dopřeložit abstrakty.

(Shodné položky jsou vynechány.)