Sem vložte zadání Vaší práce.

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

Bakalářská práce

Analýza a kompletace webové aplikace RAT

Vedoucí práce: Ing. Jiří Hunka

11. května 2015

Poděkování

Mé poděkování patří panu Ing. Jiřímu Hunkovi za odborné vedení a cenné rady které mi pomohli při zpracování této práce.
Poděkování patří také Ing. Ondřeji Kochovi, Ing. Jakubu Koudelkovi a Janu Dvořákovi za poskytnuté rady a konzultace ohledně problematiky nasazení aplikace, vývoje aplikace a problematiky překladačů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Brabenec, Miroslav. *Analýza a kompletace webové aplikace RAT*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cílem této práce je zrevidovat stav školní webové aplikace RAT. Na základě analýzy aplikace jsou navrženy úpravy a opravy, které mají zlepšit použitelnost aplikace. V textu je řešena problematika stavu aplikace a problematika kvality jejího návrhu, implementace a nasazení. Klíčové a problémy a vylepšení jsou realizována a výsledná aplikace je nasazena na serveru.

Klíčová slova překladač relační algebry, webová aplikace, Java, Python, překladač

Abstract

The aim of this thesis is to revise the state of the school web application RAT. Designed adjustments and corrections that have improved usability are based on the application analysis. The text focuses on the issue of the state of application and the quality of its design, implementation and deployment. The key issues and enhancements are implemented and final application is deployed on the server.

Keywords relational algebra translator, web application, Java, Python, compiler

Obsah

Seznam obrázků

Seznam tabulek

Úvod

RAT[1] je školní webová aplikace dostupná na adrese rat.fit.cvut.cz. Tato aplikace byla vytvořena za účelem podpory výuky relační algebry a je hojně využívána v rámci výuky na Fakultě informačních technologií ČVUT. Za 2 roky jejího fungování se stala vítanou pomůckou jak studentů tak učitelů.

Vývoj a rozvoj aplikace probíhá v rámci předmětů BI-SP1[2] a BI-SP2[3] tzn. jedná se o aplikaci vyvíjenou studenty a po 2 semestrech většinou dochází ke kompletní obměně týmu lidí, kteří na aplikaci pracují. Hlavním účelem těchto předmětů je to, aby si studenti prakticky vyzkoušeli práci v týmu a aby si prakticky vyzkoušeli analýzu, návrh, prototypovou realizaci a iterativní vývojový proces na rozsáhlejším softwarovém systému[2][3]. Z těchto důvodů má bohužel aktuálně běžící verze systému (v2.2) mnoho nedostatků, které je třeba odstranit. Nutno ovšem dodat, že i aktuální verze je hojně využívána všichni ji berou jako velmi vítanou pomoc.

Ve své bakalářské práci se nejprve podívám na stav aplikace, provedu její analýzu, zjistím v jaké fázi vývoje jsou její jednotlivé komponenty, zhodnotím kvalitu použitého kódu a otestuji, jak dobře je nasazená. Ze zjištěných informací následně vyvodím závěry, které velmi ovlivní následující postup.

Druhým krokem bude oprava, věcí které působí v aplikaci problémy. Zaměřím se primárně na věci, které znemožňují používání aplikaci, bezpečnostní problémy a chyby v nasazení. Z vlastní zkušenosti vím, že aplikace padá a nechová se tak jak by měla. Bude zde jistě několik věcí k řešení. V závislosti na množství zbylého času se pokusím do aplikace implementovat nové technologie a další rozšíření její funkcionality.

[1] RAT je zkratka z **R**elation **A**lgebra **T**ranslator, relation algebra je pojem označující rozšířenou variantu booleovy algebry. Správný překlad výrazu relační algebra je relational algebra. Nejlépe by pak vystihl činost této aplikace pojem relational algebra compiler.

[2] BI-SP1 je předmět na FIT ČVUT, plný název předmětu je Softwarový týmový projekt 1

[3] BI-SP2 je předmět na FIT ČVUT, navazuje na BI-SP1, plný název předmětu je Softwarový týmový projekt 2

Výsledek své práce nasadím na server a podniknu všechny možné kroky k tomu, aby aplikace obsahovala všechna uživatelská data a přechod na ni byl maximálně bezproblémový.

Kapitola 1

Cíl práce

Primárním cílem mé práce je prozkoumat aktuální řešení RATu a pokusit se do něj zakomponovat nově vytvořený překladač, který by splňoval potřeby, které jsou aktuálně na aplikaci kladeny. Provedení analýzy je nutné zejména z důvodu toho, že se na vývoji vystřídaly 2 týmy vývojářů a dokumentace k projektu je velmi slabá a v mnoha oblastech dostupná pouze ve slovní podobě. Dalším důvodem je pak fakt, že k projektu momentálně neexistuje dokument, který by shrnoval jeho aktuální stav.

Novým požadavkem, který by měla aplikace splňovat, je možnost jejího využití v rámci projektu, který si klade za cíl umožnit automatizaci opravování testů a semestrálních prací v rámci výuky databázových předmětů. RAT by zde měl mít roli komponenty, která bude zajišťovat kontrolu a překlad dotazů v relační algebře. Aktuálně používaný překladač momentálně bohužel nesplňuje požadavky, které jsou potřeba. Po předchozích problémem s jeho rozšířením a používáním bylo přistoupeno k napsání nového překladače. Nová implementace překladače je do jisté míry hotova. Vzhledem k tomu, jak uspěchaně probíhal jeho vývoj a k téměř nulovému testování, je zde mnoho otazníků ohledně jeho kvalitě.

Nejprve bude tedy třeba zjistit v jakém stavu jsou jednotlivé části aplikace. Zaměřím se zde primárně na části, které by měli být dále využívány tzn. webovou aplikaci a novou implementaci překladače. Ze staré implementace překladače mě bude zajímat pouze rozhraní pomocí kterého komunikuje s webovou aplikací.

Na základě výsledků analýzy specifikuji místa, kde má aktuálně aplikace problémy a navrhnu způsob jejich řešení. Z analýzy bude také vycházet návrh aplikačního rozhraní, které by mělo umožnit propojení aplikace a nově vzniklého překladače. Toto rozhraní by mělo v budoucnu sloužit i pro snadné připojení dalších překladačů.

Většinu z navržených věcí se pokusím i implementovat a nasadit do produkční verze. Zde je bohužel otázkou složitost a časová náročnost jednotlivých úkonů, kterou vzhledem k nedostatečné dokumentaci jen velmi těžko odhado-

vat.

K výsledné aplikaci bych rád umožnil přístup skrz REST API [4] rozhraní s možností autorizace přes OAuth2[5], což by mělo zajistit jednoduší použití RATu v jiných projektech.

Všechny výstupy mé práce by měly řádně otestovány a zdokumentovány a aplikace obsahující změny by měla být nasazena na serveru.

[4]REST API je architektura aplikačního rozhraní pro distribuované prostředí, je zkratka **RE**presentational **S**tate **T**ransfer **A**pplication **P**rogramming **I**nterface

[5]OAuth2 je moderní autorizační protokol (resp. framework), který se stal de facto standardem pro zabezpečení RESTových webových služeb[4]

Kapitola 2

Analýza webové aplikace

Implementaci RATu lze rozdělit do dvou částí, webové aplikace a překladače.

2.1 Technologie

Tato část aplikace je implementována v jazyce Java. Konkrétně se jedná o kombinaci Javy EE 6 pro backend a Javy SE 6 pro business metody. Klasická Java je zde rozšířena o balíčky pro komunikacemi s databázemi, balíčky pro Scalu a balíčkem pro komunikaci s LDAP serverem. Pro zajištění modularity zde byla použita technika EJB[6]. Pro spuštění aplikace je potřeba aplikační server GlassFish ve verzi 3.1.2.2. Data jsou uložen a v PostgreSQL databázi verze 9.1.

Na serveru je aktuálně nasazena aplikace ve verzi 2.2. Existuje také verze 2.3, ve které došlo k úpravám překladače. Každá verze má bohužel problém s dotazy určitého typu a většinou o nasazení rozhoduje to která verze momentálně lépe vyhovuje potřebám studentů a kantorů.

2.1.1 Java SE 6

Java SE, též známá pod názvem Java Platform, Standard Edition, je široce používaná platforma pro vývoj portabilních aplikací pro nasazení na desktop a na server[5]. Java SE verze 6 (někdy též označována jako verze 1.6.0[6]) byla vydána dne 11. 12. 2006 a byla první verzí která nesla označení v podobě Java SE, které nahradilo starší J2SE[7]. Její podpora byla ukončena v únoru 2013, kdyby došlo k zastavení veřejně dostupných aktualizací. Poslední veřejně dostupnou aktualizací byl pak patch Java SE 6 Update 45 který vyšel 16. 4. 2013. Od tohoto data vydává Oracle updaty dostupné pouze pro zákazníky s rozšířenou verzí podpory[8]. Poslední dostupnou verzí Javy SE k dnešnímu datu je

[6]Enterprise Java Beans, zkráceně EJB, je architektura serverových komponent, která umožňuje modulární tvorbu enterprise aplikací

Java SE 8, jenž byla vydaná 18. 3. 2014. Aktuálně stále podporovaná je také Java SE 7 vydaná 28. 7. 2011.

2.1.2 Java EE 6

Java EE, též známá pod názvem Java Platform, Enterprise Edition (dříve označovaná jako J2EE), staví na základech Javy SE. Tuto verzi pak rozšiřuje o věci určené pro běh a vývoj velkých, škálovatelných, spolehlivých a zabezpečených síťových aplikací[9]. Java EE 6 byla vydána 10. 12. 2009. Poslední verze Javy EE je Java EE 7 vydaná 28. 5. 2013. Momentálně probíhají práce na Javě EE 8 jejíž příchod je očekáván ve 3. čtvrtletí roku 2016.

2.1.3 GlassFish 3.1.2.2

Jedná se multiplatformní open-source aplikační server pro Javu EE, jehož vývoj započala firma Sun Microsystems. Dnes financuje vývoj serveru firma Oracle a implementace serveru je dostupná ve dvou variantách GlassFish Server Open Source Edition a Oracle GlassFish Server, k němuž lze získat placenou podporu. GlassFish Server verze 3 podporuje Javu EE 6 a byl vydán spolu s ní tzn. 10. 12. 2009. Jeho poslední update vyšel 17. 7. 2012 a nese označení 3.1.2.2[10].

2.1.4 PostgreSQL 9.1

PostgreSQL, často zkráceně označován jako Postgres, je objektově-relační databázový systém (ORDBMS[7])[11]. Jedná se komunitou vyvíjený open-source multiplatformní projekt. Verze 9.1 byla vydaná 11. 9. 2011, poslední stabilně vydaná verze je 9.4.1 vydaná 5. 2. 2015[12].

2.2 Stav aplikace

2.2.1 Repositář

Aplikaci lze stáhnout z SVN[8] repositáře z adresy svn.project.fit.cvut.cz/RA/. Přístup k němu má každý, kdo je schopný se autorizovat vůči školnímu LDAP[9] serveru. Soubory s webovou aplikací se pak nalézají v podsložce RAT.

Repositář obsahuje několik verzí aplikace. Jedná se o klasickou hlavní verzi nacházející se v trunk větvi, k níž se vážou 2 tagované[10] verze (1.0 a 1.1). Dále

[7] ORDBMS je zkratka pro **O**bject-**R**elational **D**atabase **M**anagement **S**ystem

[8] SVN je systém pro správu a verzování zdrojových kódů, jeho plný název je Apache Subversion

[9] LDAP je definovaný protokol pro ukládání a přístup k datům na adresářovém serveru, zkratka znamená **L**ightweight **D**irectory **A**ccess **P**rotocol

[10] tag ve verzovacím nástroji SVN slouží pro označení významné verze aplikace

repositář obsahuje větev[11] s vývojovou verzí (označenou jako dev) a větev s testovací verzí (označenou jako test).

To, že vše nebude, tak jak by mělo být, naznačuje fakt, že na serveru je aktuálně nasazená verze 2.2 a je dostupná verze 2.3 (obě verze jsou dostupné pouze ve zkompilované podobě). Po vytvoření projektu v IDE s použitím souborů uložených v trunk[12] větvi jsem zjistil, že aplikace neobsahuje žádné z externích knihoven potřebných k jejímu zkompilování. Dále také u aplikace chybí zdrojové kódy některých tříd, a občas i celé balíčky zdrojových kódů. Verze s tagem 1.1 nese po načtení projektu označení vývojové verze. Verze s tagem 1.0 by podle označení měla být produkční verze. Novější verze se nacházejí ve větvích dev a test. Verze v obou větvích zde nesou označení 2.2. S největší pravděpodobností má tedy smysl zabývat se jen dev a test verzí.

Rozdíl mezi mezi dev a test verzí není nikde popsán. Po krátkém pročtení logu posledních cca 100 commitů z celkových 836 (letmé pročtení toho o jaké soubory se jednalo a občas detailnější pohled co se měnilo) jsem dospěl k závěru, že dev verze jednoho den byla prohlášena za verzi k testování. Do testovací verze byly doplněny automatické testy, testovací výpisy, atd. V aplikaci však docházelo k dalším změnám, které byly prováděny pouze v rámci testovací verze, z čehož usuzuji, že testovací verze je jakýsi hybrid mezi vývojovou a testovací verzí. Testovací verze nese stále označení 2.2, pravděpodobně to bude i většinová část verze 2.3 jenž je zkompilována na serveru. Informace o tom co by měla obsahovat verze 2.3 oproti věcem v repositáři nesoucí označení 2.2 jsem bohužel nikde nenašel.

V ideálním případě by bylo nejlepší podrobně historii commitů a zjistit co přesně se v projektu dělo. Vzhledem k tomu, že se jedná o commity za 2 semestry (2. tým vývojářů v rámci BI-SP1 a BI-SP2) a tomu, že je v aplikaci nutné provést určité změny, aby byla vůbec použitelná pro příští běh předmětu BI-DBS[13] v zimním semestru 2015/16, to bohužel není v mých časových možnostech.

Pro budoucí vývoj bude dle mého názoru třeba vytvořit jednu produkční verzi, která bude v repositáři tam kde by měla. Pro tuto verzi je pravděpodobně nejlepším adeptem testovací verze.

2.2.2 Dokumentace

Dokumentace aplikace je velmi špatná. Část informací ohledně celého projektu je dostupná na adrese trac.project.fit.cvut.cz/RA/wiki. Jedná se o wiki projektu. Na stránky se ovšem nelze jednoduše dostat, jelikož je potřeba, aby byl člověk přidán jako člen týmu. Přístupy k ní mi poskytl jeden z členů 2. týmu, který mi stáhl zajímavé části wiki do offline podoby. Je zde několik zajímavých

[11] větěv neboli branch slouží k vývoji jednotlivých funkcionalit izolovaně od ostatních

[12] trunk je hlavní vývojová větev, pro kterou by měla platit zásada, že zde musí vždy být fungující nebo aspoň přeložitelný kód

[13] BI-DBS je předmět na FIT ČVUT, plný název předmětu je Databázové systémy

```
//----------------------NO COMMENT--------------------i hate myself
    for that
public void setSampleQuery(List<RAQuery> l) {//procisteni bugu... ehm
    sampleQuery = new ArrayList<LocalQuery>();
    for (RAQuery r : l) {
        sampleQuery.add(new LocalQuery(r.getQuery(), r.getName(),
            r.getDescription(), r.getId()));
    }
}
```

Zdrojový kód 2.1: Ukázka jednoho z mála komentářů

informací k práci s GlassFish serverem, jinak je většina informací příliš strohá, než aby mohla posloužit ke zrychlení orientace v projektu. Informace ohledně nasazení na server jsou zastaralé, jelikož došlo přeinstalaci serveru z důvodu nutnosti přechodu na 64-bitový systém a došlo ke změnám v nasazení, které zde již nebyly doplněny.

Zdrojový kód je z větší části úplně bez komentářů. Nejlépe to dle mého shrnuje citace jednoho z komentářů, který je ve zdrojovém kódu.

2.3 Problémy

2.3.1 Nešifrovaná komunikace s LDAP serverem

Autorizační požadavky odesílané na školní LDAP server jsou posílány v nešifrované podobě. V době vzniku aplikace to bylo možné, od 9. 2. 2015 je tato možnost ovšem blokována. Důsledkem toho je, že momentálně není možné se k aplikaci přihlásit pro uživatele, kteří se autorizují vůči serverům FIT ČVUT.

Navrhované řešení: Úprava aplikace, tak aby komunikovala s LDAP serverem s použitím zabezpečení pomocí protokolu SSL[14]. Další alternativou je implementace autorizace pomocí technologie Shibboleth SSO[15], kterou by možná bylo dobré do budoucna do aplikace dodat.

[14]SSL je je protokol, resp. vrstva vložená mezi transportní a aplikační vrstvu, která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran, jedná se o zkratku **S**ecure **S**ockets **L**ayer

[15]Shibboleth SSO je opnesource projekt, který který poskytuje službu Single Sign-On neboli jednotné přihlášení

```
0000: 6c 6f 67 69 6e 5f 66 6f 72 6d 3d 6c 6f 67 69 6e login_form=login
0010: 5f 66 6f 72 6d 26 6c 6f 67 69 6e 5f 66 6f 72 6d _form&login_form
0020: 25 33 41 6c 6f 67 69 6e 54 79 70 65 3d 46 49 54 %3AloginType=FIT
0030: 26 6c 6f 67 69 6e 5f 66 6f 72 6d 25 33 41 6c 6f &login_form%3Alo
0040: 67 69 6e 3d 62 72 61 62 65 6d 69 33 25 34 30 66 gin=brabemi3%40f
0050: 69 74 2e 63 76 75 74 2e 63 7a 26 6c 6f 67 69 6e it.cvut.cz&login
0060: 5f 66 6f 72 6d 25 33 41 70 61 73 73 77 6f 72 64 _form%3Apassword
0070: 3d 70 72 69 73 6e 65 5f 74 61 6a 6e 65 5f 68 65 =prisne_tajne_he
0080: 73 6c 6f 26 6c 6f 67 69 6e 5f 66 6f 72 6d 25 33 slo&login_form%3
0090: 41 62 74 6e 2d 6c 6f 67 69 6e 3d 50 25 43 35 25 Abtn-login=P%C5%
00a0: 39 39 69 68 6c 25 43 33 25 41 31 73 69 74 26 6a 99ihl%C3%A1sit&j
00b0: 61 76 61 78 2e 66 61 63 65 73 2e 56 69 65 77 53 avax.faces.ViewS
00c0: 74 61 74 65 3d 2d 31 35 32 37 31 33 32 38 37 39 tate=-1527132879
00d0: 31 38 39 32 35 30 30 31 36 25 33 41 38 34 30 37 189250016%3A8407
00e0: 30 36 34 34 38 32 32 35 32 33 35 34 38 30 37 064482252354807
```

Výstup z programu 2.2: Dekódovaný packet odchycený programem Wireshark

2.3.2 Hesla přes HTTP

Komunikace se serverem probíhá celou dobu přes protokol HTTP[16]. Problém nastává při odesílání přihlašovacích údajů do aplikace, případně při odesílání hesla pro připojení vlastní databáze. Jelikož je komunikace nešifrovaná, tak v případě, že někdo odposlouchává provoz na síti (packet sniffing[17]), lze bez problému ze získaných dat vyčíst hesla. Ukázka toho co lze odchytit, například s použitím programu Wireshark[18], je níže (viz výstup z tohoto programu 2.2). Jedná se především o data na řádcích 0030 až 0070 (login=brabemi3 a password=prisne_tajne_heslo).

Navrhované řešení: Bude třeba nastavit, aby aplikace komunikovala přes HTTPS[19]. Lze to řešit změnou nastavení formulářů přes které dochází k odesílání hesel. Dále je zde možnost přepnout aplikaci, aby komunikovala pouze přes HTTPS. Obě varianty budou pravděpodobně vyžadovat změnu konfigurace webservru (Apache2, GlassFish).

2.3.3 Nelze editovat již vytvořené uživatelské databáze

Aplikace umožňuje přihlášeným uživatelům připojit se na databázi. Lze se připojit buď na nadefinované databáze, které jsou přístupné všem přihlášeným

[16] HTTP je základní protokol datové komunikace pro World Wide Web

[17] packet sniffing je činost, při které dochází k zachytávání komunikace v počítačové síti, zachycené packety je následně možné dekódovat

[18] Wireshark je protokolový analyzér a paketový sniffer

[19] HTTPS je nadstavba síťového protokolu HTTP, která šifruje přenášená data pomocí SSL nebo TLS

uživatelům (výukové databáze pro BI-DBS, ...) nebo je možné vytvořit si privátní připojení na vlastní databázi. Po připojení se rozšíří možnosti editoru dotazů o nabídku obsahující názvy tabulek a názvy sloupců každé tabulky. Přibude možnost přeložený dotaz vykonat nad danou databází a podívat se na data jaká vrací. Aplikace si také pamatuje historii dotazů a je zde možnost uložit si oblíbené dotazy s vlastním komentářem.

Problém nastává v okamžiku, kdy potřebujete pozměnit některý z parametrů připojení k databázi. Možnost editace zde není a je tedy nutné připojení smazat a vytvořit jej znovu. Historie a oblíbené dotazy se ovšem vztahují ke konkrétnímu připojení nelze je přenést na jiné připojení.

Navrhované řešení: Implementovat funkce a potřebnou logiku pro editaci připojení a vyřešit co dělat v případě, kdy uživatel zůstane připojen k nezměněnému připojení. Případně vytvořit import a export oblíbených dotazů a historie dotazů, který by mohl být využit i v jiných částech aplikace.

2.3.4 Problémy se zobrazením aplikace

V aplikaci se na mnoha místech objevují problémy se zobrazením. Jedním z nich je náhled historie dotazů. Zde se zobrazuje vždy jen maximálně prvních 15 znaků dotazu. V případě delších názvů tabulek je to pak trochu matoucí a výrazně to omezuje použitelnost tohoto prvku (viz obr. 2.1).

Lépe by mohlo také fungovat zobrazení oblíbených dotazů. Probíhá zde zkracování textů stejně jako u historie a navíc zde ještě dochází k vytečení položek mimo vymezenou oblast. Takto zobrazené prvky nejde moc dobře používat, působí rušivě a aplikace díky tom působí nedodělaně. Jelikož se do aplikace momentálně nelze přihlásit (viz problém 2.3.1 a 2.3.7), není možné zde dodat ukázku.

Při pokusu zobrazit stánky na zařízení s menším displejem občas k překreslení nedojde vůbec. Pokud k němu dojde je to spíše na škodu. Stane se totiž pouze to, že prvky obsahující historii a dostupné tabulky se rozšíří na šířku celé stránky a navzájem se překrývají s horní nabídkou (viz obr. 2.2).

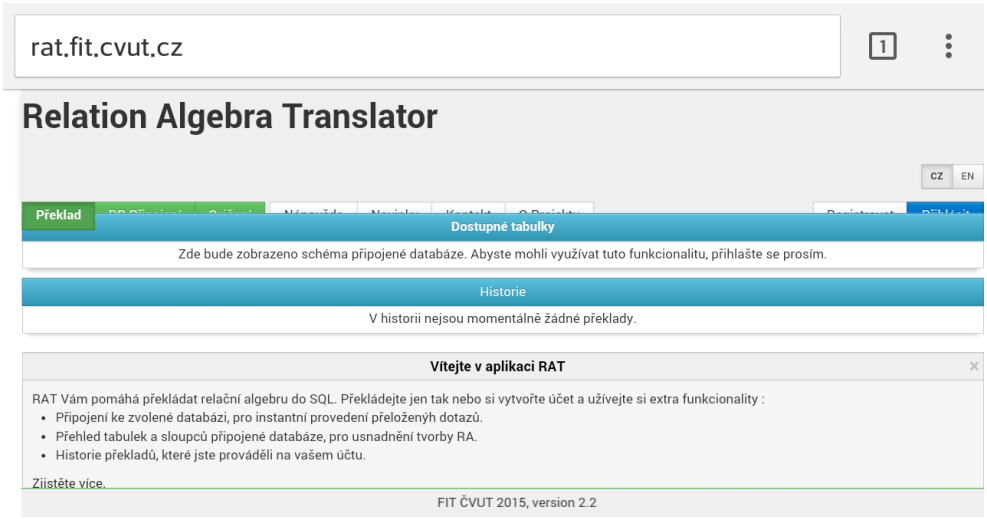
Navrhované řešení: Bude potřeba upravit CSS[20] u prvků, které špatně zobrazují. V případě problémů se zobrazením historie dotazů a oblíbených dotazů bude třeba zasáhnout i do kódu aplikace, která zřejmě omezuje délku zobrazeného náhledu dotazů na maximálně 15 znaků. Vzhledem k tomu, že aplikace využívá CSS knihovnu Bootstrap[21] v2.1.1, bude třeba zkontrolovat jestli není špatně nastavena a jestli zde nedochází ke kolizi CSS. Tato knihovna by měla být při správném použití schopna řešit většinu problémů. Možností je

[20] CSS je jazyk pro popis způsobu zobrazení dokumentů zapsaných ve značkovacích jazycích (HTML, XML, ...), zkratka znamená **C**ascading **S**tyle **S**heets

[21] Bootstrap je populární HTML, CSS, a Javascript framework pro tvorbu front-endu webových aplikací

Historie
(PACIENT !<* R
REZERVACE(den=
MISTNOST \ (OR
ORDINACNIHODIN
(ORDINACNIHODI
(ORDINACNIHODI
ORDINACNIHODIN

Obrázek 2.1: Zkrácené dotazy v prvku historie dotazů



Obrázek 2.2: Zobrazení na telefonu LG D605 (display 1280 x 720 px)

také přechod na novější verzi Bootstrapu (aktuálně je dostupná v3.3.2) nebo přechod na jinou CSS knihovnu. Možnost napsaní kompletně vlastních CSS beru jako poslední zcela nouzovou variantu.

2.3.5 Nelze načíst některé zdroje ze serveru

U aplikace se projevuje problém s dostupností některých zdrojů ze serveru. Občas se jeví nedostupné celé stránky s chybou 502[22]. Jindy se nepodaří načíst některé soubory s Javascriptem nebo CSS. V tomto případě se pak stránky zobrazují špatně a vypadají různě, vždy podle toho, která část je nedostupná.

Aplikace je na serveru nasazená tak, že veškeré HTTP a HTTPS požadavky jsou poslány nejprve na Apache2 HTTP Server, který tyto požadavky případně posílá dále na GlassFish Server, na kterém běží samotná aplikace. Komunikace mezi Apache2 serverm a GalssFish serverem je řešena pomocí protokolu AJP[23]. Na straně Apache2 komunikaci zařizuje komponenta s názvem mod_jk. Na straně GlassFish serveru, je pak pro tyto účely vytvořen speciální network listener.

Z analýzy logů vyplynulo, že problém je způsoben chybami v komunikaci přes protokol AJP. Dle informací se jedná o známý bug modulu grizzly-http-ajp, který je zodpovědný za komunikaci přes protokol AJP v GlassFish serveru[13].

Navrhované řešení: Problém lze řešit nasazením patche pro modul grizzly-http-ajp. Další variantou jak problém vyřešit, je změna způsobu komunikace mezi Apache2 a GlassFish serverem. Tuto komunikaci lze realizovat například pomocí komponenty mod_proxy na straně Apache2 a výchozích network listenerů na straně GlassFish serveru. Výchozí network listenery komunikují přes bežnější HTTP a HTTPS protokoly. Vzhledem k vyšší četnosti jejich používání (oproti protokolu AJP), lze očekávat jejich lepší odladěnost.

2.3.6 Nestabilita aplikace

Aplikace má problémy se stabilitou. Na serveru relativně často dochází k tomu, že aplikace neúměrně zatěžuje CPU. Jedná se o dlouhodobé vytížení vytížení CPU na úrovni 80% na jádro na všech jádrech. Vytížení s časem neopadá a je nutné restartovat webovou aplikaci. Tento jev se častěji projevuje ve dnech, kdy je k serveru připojeno nejvíce uživatelů (týden okolo před termíne odevzdání semestrální práce z BI-DBS). Vzhledem k tomu, že chybí jakékoliv použitelné logy, nelze říct o kolik přístupů se jedná. Vzhledem k tomu, že v posledním semestru mělo předmět BI-DBS zapsáno 353 studentů [14]. Pokud předpokládám, že traffic způsoboný jinými uživateli je zanedbatelný, server (CPU 4x 2,3 GHz, 8GB RAM) by měl počet uživatelů ustát.

[22] 502 Bad Gateway je stavový kód protokolu HTTP, který je stavový kód protokolu HTTP, který znamená, že při zpracovávání požadavku došlo ke blíže nespecifikované chybě

[23] AJP je binární protokol, slouží k předávání požadavků mezi webovým a aplikačním serverem, zkratka znamená **A**pache **J**Serv **P**rotocol

```
2015-02-15T11:59:33.626+0100|SEVERE|glassfish3.1.2|com.sun.grizzly.config.Gri
    ↪ zzlyServiceListener|_ThreadID=21;_ThreadName=Thread-2;| GRIZZLY0051:
    ↪ ProcessorTask exception.
java.lang.IllegalStateException: Invalid packet magic number: 4745 pos=0 last
    ↪ Valid=40 end=0
   at com.sun.grizzly.http.ajp.AjpInputBuffer.readAjpMessageHeader(AjpInputB
       ↪ uffer.java:90)
   at com.sun.grizzly.http.ajp.AjpProcessorTask.parseRequest(AjpProcessorTas
       ↪ k.java:107)
   at com.sun.grizzly.http.ProcessorTask.doProcess(ProcessorTask.java:717)
   at com.sun.grizzly.http.ProcessorTask.process(ProcessorTask.java:1056)
   at com.sun.grizzly.http.DefaultProtocolFilter.execute(DefaultProtocolFilt
       ↪ er.java:229)
   at com.sun.grizzly.DefaultProtocolChain.executeProtocolFilter(DefaultProt
       ↪ ocolChain.java:137)
   at com.sun.grizzly.DefaultProtocolChain.execute(DefaultProtocolChain.java
       ↪ :104)
   at com.sun.grizzly.DefaultProtocolChain.execute(DefaultProtocolChain.java
       ↪ :90)
   at com.sun.grizzly.http.HttpProtocolChain.execute(HttpProtocolChain.java:
       ↪ 79)
   at com.sun.grizzly.ProtocolChainContextTask.doCall(ProtocolChainContextTa
       ↪ sk.java:54)
   at com.sun.grizzly.SelectionKeyContextTask.call(SelectionKeyContextTask.j
       ↪ ava:59)
   at com.sun.grizzly.ContextTask.run(ContextTask.java:71)
   at com.sun.grizzly.util.AbstractThreadPool$Worker.doWork(AbstractThreadPo
       ↪ ol.java:532)
   at com.sun.grizzly.util.AbstractThreadPool$Worker.run(AbstractThreadPool.
       ↪ java:513)
   at java.lang.Thread.run(Thread.java:701)
```

Log 2.3: Log chyby komunikace pomocí protokolu APJ

Z logů lze vyčíst, že v momentech, kdy dochází k vytížení serveru se v aplikaci objevuje mnoho neodchycených výjimek. Většina z nich pochází z překladače a je v nich patrné, že v této čísti kódu dochází k velmi hlubokému zanořování funkcí v rámci jednoho objektu. Vzhledem k těmto informacím bych očekával, že nestabilita aplikace bude způsobena překladačem.

Navrhované řešení: Vzhledem k problémům s aktuálně používaným překladačem, bude dle mého nejlepší přestat tento překladač používat a místo něj použít nějaký nově vzniklý překladač. Zároveň by bylo dobré do budoucna evidovat obdobné problémy, aby bylo více dat pro ladění aplikace.

```
[#|2014-12-07T20:16:36.804+0100|WARNING|glassfish3.1.2|javax.enterprise.syste
    ↪ m.container.web.com.sun.enterprise.web ...
java.lang.AssertionError
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.helper$1(Simp
        ↪ leSQLFormatter.scala:326)
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.helper$1(Simp
        ↪ leSQLFormatter.scala:333)
... 25x same line
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.addEnd$1(Simp
        ↪ leSQLFormatter.scala:341)
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.addRAChunk(Si
        ↪ mpleSQLFormatter.scala:345)
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter$$anonfun$cz$c
        ↪ vut$fit$rat$engine$model$sql$impl$SimpleSQLFormatter$$doFromRec$1$
        ↪ 1.apply$mcV$sp(SimpleSQLFormatter.scala:205)
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.CHUNK(SimpleS
        ↪ QLFormatter.scala:267)
    at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.NEW(SimpleSQL
        ↪ Formatter.scala:262)
... 50x at cz.cvut.fit.rat.engine.model.sql.impl.SimpleSQLFormatter.*
    at cz.cvut.fit.rat.engine.business.impl.SimpleTranslator.doTranslate(Simp
        ↪ leTranslator.scala:64)
    at cz.cvut.fit.rat.engine.business.impl.SimpleTranslator.translateWithSch
        ↪ ema(SimpleTranslator.scala:39)
    at cz.cvut.fit.rat.weblogic.business.beans.TranslateManagedBean.doTransla
        ↪ te(TranslateManagedBean.java:227)
    at cz.cvut.fit.rat.weblogic.business.beans.TranslateManagedBean.translate
        ↪ AndExecute(TranslateManagedBean.java:119)
    at sun.reflect.GeneratedMethodAccessor8389.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMet hodAcces
        ↪ sorImpl.java:43)
... 100 more lines
```

Log 2.4: Jedna z chyb, které se objevují před pádem aplikace

2.3.7 Nefunkční odesílání mailů

V aktuálně nasazené verzi aplikace nefunguje odesílání mailů. Chyba se projevu při akcích jejichž součástí je odesílání pošty. Jedná se zejména o akce registrace nového uživatele a obnova zapomenutého hesla. V obou případech je výsledkem chybová stránka s chybou 500[24].

V kombinaci s nefunkční autorizací proti serverům FIT ČVUT je výsledkem, že do aplikace se aktuálně může přihlásit pouze uživatel s již vytvořeným lokálním účtem, který si pamatuje heslo. Většina uživatelů tedy momentálně nemá možnost využívat pokročilé funkce aplikace, které jsou dostupné až po přihlášení.

[24] 500 Internal Server Error je stavový kód protokolu HTTP, který znamená, že při zpracovávání požadavku došlo ke blíže nespecifikované chybě

Navrhované řešení: Pokusit se v dokumentaci nebo ve zdrojovém kódu najít zmínky o tom, jak má fungovat odesílání emailů a prověřit toto nastavení u nasazené verze.

Kapitola 3

Analýza překladače

3.1 Starý překladač

Nejstarší překladač je implementován v jazyce Scala a jedná se o překladač, který je aktuálně používán. Kód je rozdělen do 81 souborů a jedná se celkem o 4446 řádek. Překladač byl vyvíjen po dobu dvou let (LS 11/12, ZS 12/13, LS 13/13, ZS 13/14). Překladač podporuje překlad RA do OracleSQL, PostgreSQL a MySQL. Aplikace využívá pouze běžné knihovny Scaly (SortedSet, TreeSet, ...), jedná se tedy prakticky o plně vlastní implementaci. Dokumentace je dostupná pouze ve formě komentářů ve zdrojovém kódu.

V LS 13/14 bylo rozhodnuto, že bude vyvinut nový překladač, který by odstraňoval chyby v návrhu tohoto překladače. Při vývoji a testování tohoto překladače se totiž vyskytly následující problémy.

Problémy z rozšířením překladače při implementaci překladu do PostgreSQL a MySQL. Při snaze o toto rozšíření došlo k rozbití schopnosti kontroly existence tabulek a sloupců pro OracleSQL.

Za dobu používání se ukázalo, že překladač má nízkou výkonnost a také pravděpodobně zapříčiňuje nestabilitu aplikace (viz problém 2.3.6). Překladač má v sobě několik neodchycených výjimek. Z jejich výpisu je patrné, že při překladu mnohdy dochází k zanoření do hloubky 100-200 funkcí s maximální délce vstupního řetězce 300 znaků.

3.1.1 Propojení překladače a webové aplikace

Jazyk Scala je navržen tak, aby snadno pracoval na virtuálních strojích, největší pozornost pak byla kladena na virtuální stroj Java (JVM)[15]. Propojení v aplikaci tohoto využívá a je řešeno tak, že k webové aplikaci je překladač přikompilován. Při překladu pak dojde k vytvoření instance překladače nad kterou jsou volány příslušné funkce.

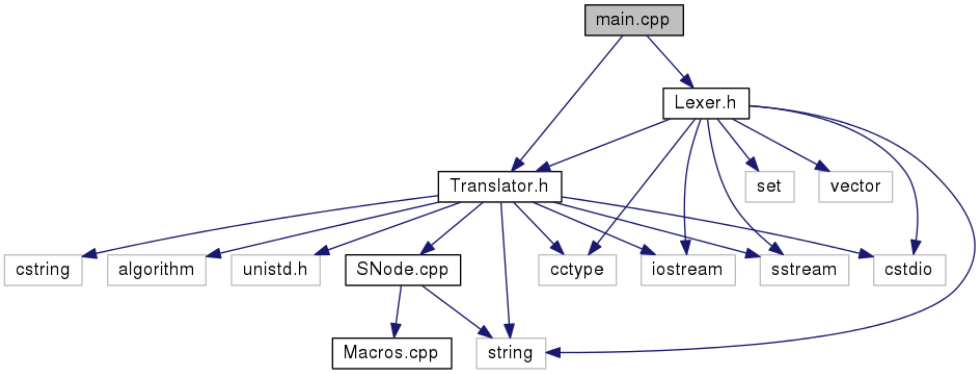
3.2 Nový překladač

Nová implementace překladače je napsaná v jazyku C++. Překladač byl vyvíjen poslední rok (LS 13/14 a ZS 14/15). Kód je rozdělen mezi 7 souborů a jedná se 2644 řádek kódu.

3.2.1 Struktura aplikace

Aplikace je rozdělena do dvou hlavních částí (viz obr. 3.1). První je třída Lexer, která se stará syntaktickou kontrolu vstupu ze kterého sestavuje abstraktní syntaktický strom[25]. Druhou částí je pak třída Translator, která překládá abstraktní syntaktický strom do SQL. Abstraktní syntaktický strom je tvořen instancemi třídy SNode (každá instance představuje jeden vrchol).

V aplikaci jsou využívány pouze standardní knihovny dostupné v jazyce C++[16].



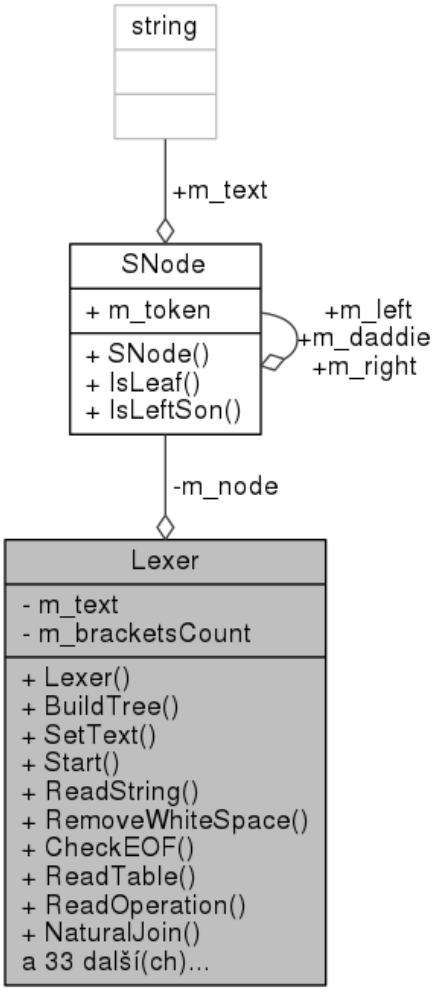
Obrázek 3.1: Graf závislostí

3.2.1.1 Lexer

Lexer tvoří většinu celkového kódu (1650 řádek z 2644). Autor se rozhodl si celý kód napsat sám bez použití jakékoli knihovny zaměřené na tuto problematiku.

Z pohledu OOP na mě působí tento kus kódu, jako by se autor nemohl rozhodnou jestli bude programovat objektově nebo ne. Definice a implementace třídy jsou rozděleny do hlavičkového souboru a souboru s implementací jednotlivých metod. Autor bohužel nepoužil žádnou enkapsulaci a všechny metody objektu jsou `public`. Mezi těmito metodami jsou například funkce pro změnu vnitřních stavů při lexikální analýze, které by dle mého měli být rozhodně `private` nebo `protected` (viz obr. 3.2).

[25]abstraktní syntaktický strom je stromová struktura, která reprezentuje syntaktickou strukturu nějakého „zdrojového kódu“



Obrázek 3.2: Diagram tříd Lexer

Při pohledu do souboru s implementací metod objektu zaujme, že je zde dalších 12 metod, které nejsou uvedeny v hlavičkovém souboru. Tyto metody se nevztahují k objektu.

Způsob implementace lexikálního a syntaktického analyzátoru také není zrovna bezchybný. Jednotlivé funkce provádí mnohdy provádí naráz zároveň tokenizaci a tvorbu AST. Při pohledu na kód tak není patrné, jaká je definice jednotlivých tokenů, ale ani jak vypadají pravidla gramatiky. Autor se v dokumentaci ohledně tohoto tématu odkazuje na dokument, v němž je vše popsáno. Daný dokument však součástí dokumentace není. Čitelnost kódu dále komplikuje časté používání `goto`. Zajímalo by mě, co autora vedlo k tak častému použití tohoto příkazu, jelikož většina materiálů o programování v C++ to silně nedoporučuje[17][18][19].

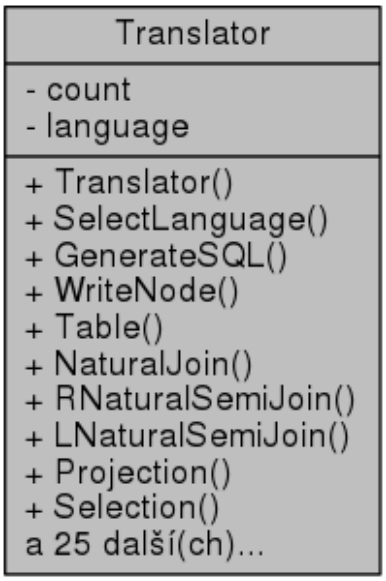
Detekce chyb je řešena špatně a to vypsáním jedné ze dvou chybových hlášek a příkazem `exit(-1)`, aniž by se program staral o alokovanou paměť.

Kód obsahuje i další chyby, ale dle mého je zbytečné je zde zmiňovat.

3.2.1.2 Translator

Tato třída je napsána lépe. Opět zde není použita enkapsulace (viz obr. 3.3), ale v souboru s implementací jsou pouze funkce vztahující se k objektu, které jsou předtím popsány v hlavičkovém souboru. Kód mi přijde lépe čitelný a lépe algoritmicky navržený. Možná to je z důvodů, že kód prakticky pouze generuje text na základě stromu, což mi přijde jednodušší než předchozí část.

Momentálně tato třída překládá pouze „hloupě“ tzn. bez znalosti struktury databáze, ke které se dotazy vztahují. Přihlédneme-li k tomu, že překlad se znalostí struktury databáze má velký význam, bude do budoucna třeba tuto funkcionalitu implementovat.



Obrázek 3.3: Diagram tříd Translator

3.2.2 Dokumentace

Nová implementace překladače dobře zdokumentována co se týče komentářů zdrojového kódu. Z komentářů lze vygenerovat dokumentaci například pomocí Doxygenu. Zde zamrzí, že autor nevygeneroval dokumentaci sám, případně alespoň nedodal informace k tomu co se má generovat nebo nezanechal potřebné soubory pro Doxygen. Faktorem, který by mohl někoho zmást, může být to jsou všechny funkce `public`.

Součástí dokumentace bohužel nejsou žádné další materiály. Problém to znamená obzvlášť u lexikálního analyzátoru, kde lze velmi velmi obtížně v kódu dohledávat informace pro další vývoj aplikace.

3.2.3 Problémy

V této sekci zmíním další problémy této implementace. U problémů nebudu uvádět možnosti, jak daný problém vyřešit. Důvody tohoto kroku osvětluji v sekci 4.2.

3.2.3.1 Kompilace

Implementace neobsahuje nic jako makefile nebo soubor s projektem pro nějaké IDE. Pro zkompilování aplikace je tedy nutné si napsat vlastní makefile nebo si vytvořit projekt v nějakém IDE. Při kompilaci se následně objeví cca 100 řádků warningů (parametry `-Wall` a `-pedantic`).

3.2.3.2 Memory leaky

Aplikace korektně neuvolňuje alokovanou paměť. Nepodařilo se mi vymyslet vstup, na po který by aplikace validně uvolnila veškerou paměť. Množství míst kde byla neuvolněná paměť alokována a množství paměti se mění v závislosti na vstupu. Z výstupů z Valgrindu[26] je patrné, že většina memory leaků[27] pochází z lexeru (viz výstup z programu 3.1). Vzhledem k tomu jakám způsobem je lexer implementován (viz sekce 3.2.1.1) se velmi obtížně dohledává, jak k leakům došlo. Případná oprava je ze stejných důvodů také velmi náročná.

3.2.3.3 Podpora unicode

Podpora unicode[28] kódování je v překladači řešena dost chabým způsobem způsobem. Se vším se pracuje jako s charem a následná detekce je prováděna porovnáním s integerem, který odpovídá doplnění charu na integer při zachování znaménka (např. `c == 0xfffffffc2`). V kódu se vyskytuje také funkce na detekci českých znaků, která je schopna detekovat 12 českých znaků které nejsou v ASCII[29]. Těchto znaků je samozřejmě více. Důvody vyřazení například ’Ú’ z české abecedy autor v dokumentaci bohužel nezmínil.

[26] Valgrind je program pro unixové systémy, který pomáhá při ladění a profilování programů

[27] Memory leak je situace, kdy program alokuje operační paměť, kterou není schopen následně uvolnit. V exttrémních případech může dojít k alokování veškeré dostupné operační paměti.

[28] Unicode je tabulka znaků všech existujících abeced

[29] ASCII je jedna z nepoužívanejších znakových sad se kterými se dnes lze setkat, zkraka znamená **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

```
~ valgrind --leak-check=full ./rat-engin "a*b" 2
...
HEAP SUMMARY:
    in use at exit: 172 bytes in 5 blocks
  total heap usage: 26 allocs, 21 frees, 1,460 bytes allocated

172 (40 direct, 132 indirect) bytes in 1 blocks are definitely lost in loss
    ↪ record 5 of 5
   at 0x4C2B0E0: operator new(unsigned long) (in /usr/lib/valgrind/vgpreloa
       ↪ d_memcheck-amd64-linux.so)
   by 0x408ADF: Lexer::CreateTableNode(std::string) (Lexer.cpp:1308)
   by 0x403BF4: Lexer::ReadTable() (Lexer.cpp:379)
   by 0x403A22: Lexer::Start() (Lexer.cpp:342)
   by 0x4039B4: Lexer::SetText(std::string) (Lexer.cpp:329)
   by 0x40C49F: main (main.cpp:39)

LEAK SUMMARY:
   definitely lost: 40 bytes in 1 blocks
   indirectly lost: 132 bytes in 4 blocks
     possibly lost: 0 bytes in 0 blocks
   still reachable: 0 bytes in 0 blocks
       suppressed: 0 bytes in 0 blocks
```

Výstup z programu 3.1: Výstup z programu Valgrind

Kapitola 4

Sumarizace aktuálního stavu

4.1 Webová aplikace

Z analýzy webové aplikace vyplynulo, že není v dobrém stavu, dokumentace k ní je neúplná a nasazení by chtělo doladit. Za prioritní momentálně považuji to, aby aplikace byla schopná fungovat v té verzi jaké je teď. Ladění aplikace a doplňování nové funkcionality by aplikaci jistě prospělo. V tento okamžik, kdy aplikace padá, nelze se do ní přihlásit a s každým přihlášením uživatel riskuje možnost velmi snadného odcizení hesla, jsou zde však palčivější problémy.

Jelikož poslední verze aplikace, která je označena jako stabilní v sobě nezahrnuje poslední rok ladění a vývoje, tak jsem po dohodě s vedoucím práce rozhodl, že budu vycházet z poslední verze. Jedná se o mix verze označené jako dev a verze označené jako test.

Priorita prací v této oblasti bude následující:

1. Úpravy v nasazení aplikace
2. Oprava částí potřebných k tomu, aby se aplikace dala použítat
3. Příprava aplikace na napojení nového překladače – aktuálně používaný překladač je pravděpodobně zodpovědný za nestabilitu aplikace
4. Analýza UI – analýzu nebylo doposud možné kvalitně provést, jelikož si aplikace nenačítala všechny zdroje ze serveru a nebylo možné se do ní přihlásit
5. Úpravy a ladění UI

4.2 Překladač

Aktuálně používanému překladači se ve své práci příliš nevěnuji. V rámci mé práce jsem krátce nahlédl do jeho kódu a prohlédl jsem si některé commity,

které se vázaly ke snaze jej upravit nebo rozšířit. Dále jsem také ještě dostal mnoho informací od vedoucího práce, který má s používáním a vývojem tohoto překladače zkušenosti.

Na základě těchto informací pak musím konstatovat, že co se týče návrhu je nová implementace překladače výrazně lepší. A pokud jde o budoucí vývoj tak plně souhlasím se snahou přejít na nový překladač. Z těchto důvodů jsem také věnoval novému překladači mnohem více pozornosti.

Z analýzy nového překladače je však patrné, že autor nedokázal návrh podpořit i kvalitou kódu. Tuto implementaci by asi bylo možné dostat do stavu, kdy by byla schopna nahradit aktuálně používaný překladač. Obávám se, ale že kvalita by nebyla dostatečná a v budoucnu by s ním mohli být problémy, které by opět vedly ke kompletnímu přepsání.

Pro dosažení potřebné kvality by bylo dle mého potřeba zcela nahradit nebo z velké části odladit, opravit a přepsat lexer. Dále by také bylo nutné promítnout tyto změny do translatoru, ve kterém by bylo nutné implementovat další funkcionalitu potřebnou pro použití překladače. Dle mého odhadu by to znamenalo zásah cca do 80 % kódu.

Z těchto důvodů jsem po konzultaci s vedoucím práce rozhodl, že nemá smysl nadále pokračovat v práci na nové implementaci překladače a že nejlepší bude navrhnout překladač komplet znovu. Do návrhu tohoto překladače by mělo bát promítnuto co nejvíc zkušeností z předchozích překladačů.

Pro překladač jsme definovali následující požadavky:

1. Snadná editovatelnost a rozšířitelnost
   - Lexikální pravidla
   - Syntaktická pravidla
   - Přidání nového dialektu SQL
   - Definice SQL k dané operaci v relační algebře
   - Vše ostatní, seznam nemůže být úplný
2. Podpora unicode kódovaní znaků
3. Překlad se znalostí databáze
4. Možnost napojení do aktuální webové aplikace
5. Podpora OracleSQL

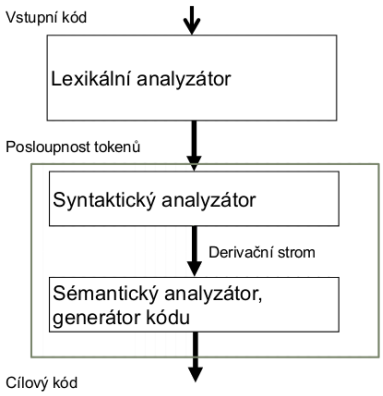
Kapitola 5

Návrh překladače

Problematice překladače se v mnoha ohledech věnuji jen velmi povrchně. K tomu sem přistoupil z několika důvodů. Prvním důvodem byl rozsah a složitost celé problematiky, na kterou by klidně mohla být vypracována samostatná bakalářská práce. Druhým důvodem je pak nedostatek času, jelikož psaní vlastního překladače vůbec nemělo být součástí této bakalářské práce.

5.1 Struktura překladače

Zjednodušeně lze říct, že překladač lze v našem případě chápat jako program, která transformuje kód napsaný v jednom programovacím jazyka do jiného programovacího jazyka[20]. Vezmeme-li v úvahu jednoduché překladače, tak lze říct, že mají 3 hlavní části. Těmito částmi jsou lexikální analyzátor, syntaktický analyzátor a sémantický analyzátor a generátor kód[1] (viz obr. 5.1).



Obrázek 5.1: Model velice jednoduchého překladače[1]

5.1.1 Lexikální analýza

Lexikální analýza je proces při kterém dochází k převodu vstupního řetězce na řetězec tokenů[21]. Lexikální analyzátor čte po znacích vstupní řetěze a pokud detekuje sekvenci znaků, která odpovídá nějakému nadefinovanému vzoru, je tato sekvence označena jako token. Lexikální analyzátor dále také provádí očistění vstupního řetězce od bílých znaků, komentářů, apod.

Sekvence tokenů, která je výstupem lexikálního analyzátoru, je pak předána k dalšímu zpracování do syntaktického analyzátoru.

5.1.2 Syntaktická analýzy

Syntaktická analýza je proces při kterém dochází ke kontrole, jestli posloupnost tokenů splňuje pravidla gramatiky daného jazyka[22]. Tuto kontrolu provádí analyzátor, tak že testuje, zda je možné vygenerovat vstupní sekvenci tokenů z počátečního symbolu za použití dané gramatiky. To lze provést pomocí dvou metod. První je metoda shora dolů, kdy syntaktický analyzátor začíná počátečním symbolem a snaží se jej převést na vstup. Druhou metodou je zdola nahoru, kdy syntaktický analyzátor začíná vstupem a snaží se jej převést na počáteční symbol[23].

5.1.3 Sémantický analyzátor

Sémantický analyzátor je část překladače, která je zodpovědná za analýzu významu jednotlivých operací. Dochází zde například k typové kontrole, konverzi datových typů nebo ke kontrole jestli jsou všechny proměnné již deklarovány.

5.1.4 Generátor kódu

Generátor kódu je komponenta, která se stará o vygenerování výsledného výstupního kódu. Během překladu je původní kód většinou prezentován formou pseudokódu, který by měl usnadnit předchozí kroky. Generátor pak převádí tento pseudokód do výstupního kódu.

5.2 Realizace překladače

Překladač jsem se rozhodl realizovat v jazyku Python. Pro lexikální analyzátor a syntaktický analyzátor jsem se rozhodl využít knihovnu PLY (Python Lex-Yacc). Na sémantický analyzátor a generátor kódu jsem zvažoval použití nějaké knihovny, která by se chovala jako SQL query builder. Bohužel se mi zde nepodařilo nalézt žádnou knihovnu, která by dostatečně pokrývala možnosti OracleSQL.

5.2.1 Python

Python je vyšší objektově orientovaný programovací jazyk, který vznikl v roce 1991[24]. Základní filozofie Pythonu je umožnit programátorům tvořit čitelnější kód, který mnohdy může být napsán na méně řádek než alternavní kód v Javě nebo C++[25]. Pro vývoj jsem zvolil ptyhon 2.7, jelikož se k němu dá najít více návodů a ukázkových kódů než pro novější verzi 3.x.

Pro překladač jsem Python vybral z několika důvodů. Prvním je podpora kódování UTF-8[30]. V základu Python ve verzi 2.x pracuje sice v kódování ASCII, ale pokud je správně nastaven ve všech okamžicích je správně zvoleno kódování, tak nemá problém pracovat plně v UTF-8[26].

Druhým důvodem byla vysoké množství dostupných knihoven z nejrůznějších oborů. U některých distribucí linuxu lze v repositářích nalézt pro tento programovací jazyk i přes několik tisíc balíčků[27]. Samostatné Python repositáře pak obsahují přes 50 tisíc balíčků[28]. Velké množství dostupných rozšíření dělá z tohoto programovacího jazyku velmi všestranný nástroj, který může být použit téměř na cokoliv.

5.2.2 Knihovna Python Lex-Yacc

Knihovna PLY je opensource knihiva šířená pod licencí BSD. Jedná se o implementaci lexikálního analyzátoru a syntaktického analyzátoru generovaného pomocí YACC[31]. Knihovna je plně implementovaná v Pythonu. Pro syntaktickou analýzu využívá syntaktocký analyzátor typu LR[32]. Knihovna byla původně navržena pro účely výuky. Její použití by mělo být přímočaré a jsou zde rozsáhlé možnosti kontroly chyb[29].

Knihovna vznikla v roce 2001 a aktuálně je dostupná ve verzi 3.4, která vyšla v roce 2011. Za dobu její existence se na jejím vývoji a ladění podílelo více než 30 lidí[29]. Knihovna by měla být dostupná na většině linuxových distribucí v podobě balíčku v repositářích[27][30][31]. Knihovna je hojně využívána a existuje mnoho projektu, které na ni staví[32].

5.3 Implementace

5.3.1 Lexikální analyzátor

Pro lexikální analýzu jsme využil knihovnu PLY. Knihovna umožňuje nadefinovat si libovolné tokeny. Každý token je identifikován textovým řetěz-

[30]UTF-8 je znaková sada, která je schopna pojmout všechny znaky formátu Unicode, zkratka znamená Universal Character Set Transformation Format 8-bit

[31]YACC je generátor syntaktických analyzátorů pro unixové operační systémy vyvinutý v 70. letech 20. století Stephenem C. Johnsonem, zkratka znamená Yet Another Compiler Compiler

[32]LR je typ snytaktického analyzátoru využívající metodu zdola nahoru, zkraka znamená Left-to-right, Rightmost derivation

cem. Specifikace tokenů může být provedena dvěma způsoby. Základní způsob umožňuje jen jednoduchou definici, kdy je token specifikován pouze pomocí regulárního výrazu. Druhá možnost je komplexnější a umožňuj při analýze tokenu zavolat téměř libovolný kód. Tato možnost také umožňuje přechod mezi stavy zásobníkového automatu, který je součástí lexikálního analyzátoru a je možné ho využít. Stavy mohou být dvojího druhu. Liší se podle toho jestli rozšiřují základní stav nebo mají exkluzivní pravidla. Definice tokenu může být specifická pro každý stav.

```
t_rename_RENAME = ur'->'

def t_lparen_RPARENBRC (t):
   ur'\)'
   t.lexer.level -=1
   if t.lexer.level == 0:
      t.type = 'RSELECTBRC'
      t.lexer.pop_state()
   return t
```

Zdrojový kód 5.1: Ukázka definice tokenů

Provádění úprav je pak velmi jednoduché. Pokud je třeba změnit definici tokenu stačí najít jeho stávající definici a upravit regulární výraz, případně upravit kód který je vykonáván při analýze tohoto tokenu. Existence exkluzivních stavů pak umožňuje dělat větší úpravy „lokálně“, takže je minimální možnost ovlivnit ostatní stavy.

Výstupem lexikálního analyzátoru je pak seznam rozpoznaných tokenů. Každý token má u sebe uložen typ, hodnotu původního řetězce, pozici počátečního znaku ve vstupním řetězci a číslo řádky vstupního řetězce (viz tab. 5.1). V případě potřeby knihovna by mělo bát možné do kterékoliv z těchto položek uložit libovolný běžný datový typ, který je součástí Pythonu.

Vstup:	user * account			
Výstup:	Typ tokenu	Hodnota	Číslo řádku	Pozice prvního znaku
1. token	TABLE	u'user'	1	0
2. token	NJOIN	u'*'	1	5
3. token	TABLE	u'account'	1	7

Tabulka 5.1: Ukázka výstupu lexikálního analyzátoru

5.3.2 Syntaktický analyzátor

Pro syntaktický analyzátor je využita YACC část knihovny PLY. Syntaktická analýza pak funguje velmi jednoduše. Analyzátor dostane jako parametr seznam tokenů a instanci lexikálního analyzátoru. Vstupní text, který pošleme do mu dáme na vstup, nejprve projde lexikálním analyzátorem. Výsledkem toho je seznam tokenů, na která jsou aplikována pravidla gramatiky a je rozhodnuto jestli byl výraz syntakticky správný nebo ne.

Pravidla gramatiky se definují jako funkce, kde je dané pravidlo je definováno jako string. V tělu funkce pak může být prováděn libovolná kód.

5.3.3 Sémantický analyzátor a generátor kódu

Vzhledem k tomu, problematika překladačů je velmi rozsáhlá a že cílem této práce neměl být návrh překladače byly tyto části přenechány na dodělání týmu v rámci předmětu BI-SP1.

5.3.4 Komunikační rozhraní

Jako komunikační rozhraní jsem zvolil REST API rozhraní s použitím dat ve formátu JSON. Tento způsob komunikace jsem zvolil z důvodu snadného napojení na libovolnou technologii, která podporuje komunikaci přes HTTP.

Obsah dat je velmi jednoduchý Klient posílá na server požadavek na překlad dotazu, který obsahuje dotaz v RA, případně informace o schématu databáze. Server klientovi odpovídá přeloženým dotazem nebo chybovou hláškou. Tyto požadavky formát JSON umožňuje přenášet a jeho struktura je čitelnější než XML což je důvod proč jsem ho zvolil.

Pro realizaci jsem zvolil Django REST framework. Jedná se webový framework napsaný v Pythonu, který umožňuje jednoduchou tvorbu REST API rozhraní[33]. Výsledná překladač se tedy chová jako webová aplikace, který je aktuálně dostupná pouze přes REST API rozhraní.

5.3.4.1 Autorizace

Autorizace pro komunikaci je řešena pomocí technologie OAuth2. Tato forma autorizace byla zvolena jelikož se jedná o moderní autorizační protokol (resp. framework), který se stal de facto standardem pro zabezpečení RESTových webových služeb. Dalším důvodem bylo to, že je tato technologie již na FIT ČVUT využívána[4].

Django REST framework má v sobě knihovny pro autorizaci přes OAuth2. Bohužel jsou tyto knihovny koncipované, tak že v sobě kombinují roli autorizačního serveru (authorization server) a serveru poskytujícího službu (resource server). Pro zasazení aplikace do školního prostředí je ovšem nutné mít implementaci, která umožňuje oddělení těchto rolí. Výsledná aplikace se pak chová

jako resource server a ověřuje token, který dostal od clienta proti centrálnímu školnímu autorizačnámu serveru.

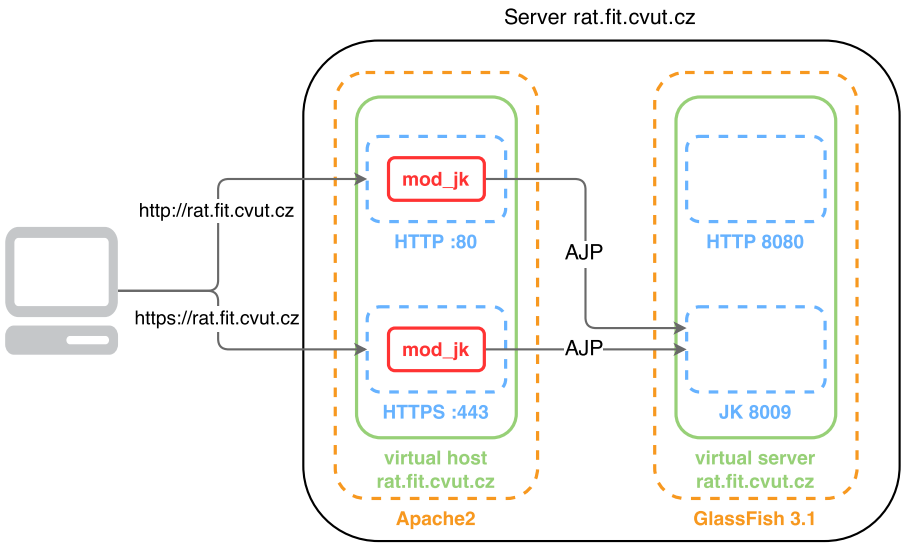
Aktuálně je autorizace nastavena tak, že stačí, aby došlo k potvrzení od autorizačního serveru, že je access token platný. Informace o povolených scopech pro daný access token, která je poskytnuta s informací o platnosti tokenu není využita.

Kapitola 6

Realizace změn a oprav

6.1 Úpravy nasazení aplikace

Jelikož v tato oblast nebyla takřka vůbec zdokumentována, tak jsem se rozhodl nejdříve vytvořit diagram toho, jak je aplikace aktuálně nasazena na serveru. Nasazení vypadá tak, že všechny požadavky jdou nejprve na Apache2. Pokud jsou požadavky směřovány na doménu rat.fit.cvut.cz nebo rat2.fit.cvut.cz jsou poslány na GlassFish ve kterém je nasazena aplikace RAT (viz obr. 6.1).



Obrázek 6.1: Diagram nasazení původní stav

6.1.1 Hesla přes HTTP

Problém se nacházel na komunikaci mezi klientem a serverem po portu 80. Pro jeho vyřešení bylo třeba upravit aplikaci tak, aby všechna hesla posílala zabezpečeně přes protokol HTTPS. Rozhodl jsem se zvolit variantu, kdy jde veškerá komunikace mezi klientem a serverem přes HTTPS.

Toho je docíleno s použitím komponenty mod_rewrite (komponenta pro Apache2). Ta zajišťuje to, že pokud přijde na server požadavek v protokolu HTTP, dojde automaticky k přesměrování komunikace na protokol HTTPS.

Pro zabezpečenou komunikaci je třeba OpenSSL certifikát. Pro tyto potřeby jsem zařídil certifikáty podepsaný certifikační autoritou TERENA[33]. Certifikát má platnost po dobu 3 let a je společný pro všechny domény, které aktuálně běží na serveru.

Zařízení certifikátu podepsaného důvěryhodnou certifikační autoritou jsem řešil proto, aby nedocházelo k zobrazení varování ve webovém prohlížeči při komunikaci se serverem, ke kterému dochází při použité self-signed certifikátu. Druhým důvodem bylo to, že v rámci opravy přihlašování bylo třeba použít technologii cvutID[34]. Pro její použití bylo doporučeno použít certifikát podepsaný důvěryhodnou certifikační autoritou.

Konfiguraci serveru jsem testoval pomocí testu na stránkách ssllabs.com. Z výsledku testu plyne (viz obr. 6.2), že mnou použitá konfigurace je odolná vůči všem objeveným slabinám SSL (Heartbleed[35], POODLE[36], CVE-2014-0224[37], ...), které jsou součástí testu. Dále je také možno sestavit zabezpečené spojení se všemi testovanými klienty s výjimkou IE6[38].

6.1.2 Nelze načíst některé zdroje ze serveru

Problém s načítáním zdrojů ze serveru byl způsoben v komunikaci přes protokol AJP mezi Apache2 a GlassFish serverem. Jednalo se o bug v implementaci JK connectoru na straně GlassFish serveru.

Jako řešení tohoto problému jsem zvolil změnu způsobu komunikace mezi Apache2 a GlassFish. Na straně Apache2 se o komunikaci stará komponenta

[33] TERENA je nezisková organizace, která slouží mimo jiné jako certifikační autorita pro digitální certifikáty pro výzkumné a vzdělávací instituce, zkratka znamená **T**rans-**E**uropean **R**esearch and **E**ducation **N**etworking **A**ssociation

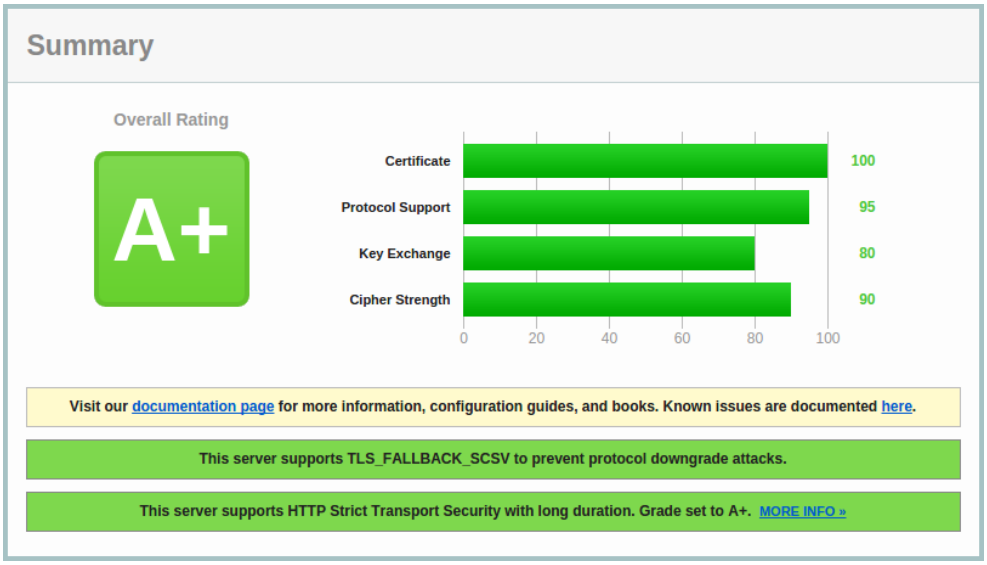
[34] cvutID je federace subsystémům IS ČVUT, ke kterým lze přistupovat pomocí autorizace přes Shibboleth SSO

[35] Heartbleed je bezpečnostní chyba kryptografické knihovny OpenSSL objevená v dubnu 2014

[36] POODLE je bezpečnostní chyba, která může být využita při man-in-the-middle útoku, kdy lze olsabit kominakci zabezpečenou pomocí TLS na komunikaci zabezpečenou pomocí slabšího SSL 3.0, zkratka znamená **P**adding **O**racle **O**n **D**owngraded **L**egacy **E**ncryption

[37] CVE-2014-0224 CCS Injection Vulnerability (CVE-2014-0224) je slabina v zabezpečení OpenSSL, kterou lze využít při man-in-the-middle útoku, kdy útočník může být schopen dešifrovat a upravovat síťový provoz

[38] Microsoft Internet Explorer 6 byl výchozí webový prohlížeč pro Windows XP and Windows Server 2003



Obrázek 6.2: Výsledek testu konfigurace SSL

mod_proxy. Ta je schopná požadavek předat na libovolnou IP adresu a port. Požadavek je dále předán ve stejném protokolu v jakém přišel. Znamená to, že na straně Glassfishe je použit HTTP connector.

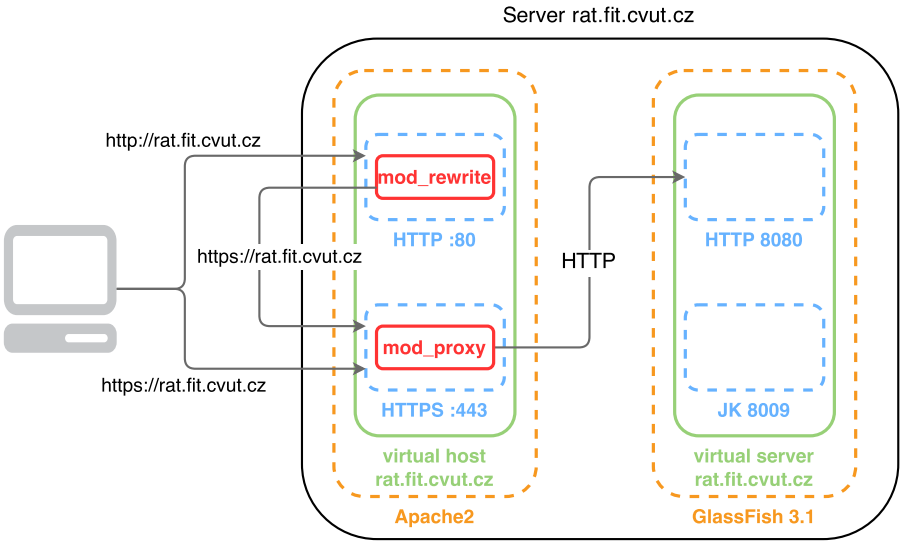
K tomuto způsobu komunikace jsem přistoupil ze dvou důvodů. Prvním byla možnost použít HTTP connector, který se se vytvoří automaticky při instalaci GlassFish serveru. Zjednoduší se tím konfigurace a také odpadá potřeba ručně patchovat instalaci serveru kvůli opravě bugu v JK connectoru. Druhým důvodem byla možnost použití komponenty mod_proxy na širší spektrum projektů oproti komponentě mod_jk. Mod_proxy splňoval potřeby naší aplikace.

6.1.3 Nefunkční odesílání mailů

V dokumentaci k původnímu projektu se mi podařilo najít poznámky ke konfiguraci nastavení odesílání emailů přes účet ve službě gmail. Tento postup jsem zreplikoval a doladil nastavení účtu na gmailu, tak aby bezpečnostní politika google nebránila odesílání emailů z aplikace. Do budoucna by bylo dobré pro tyto potřeby zařídit školní mailový účet.

6.1.4 Výsledek

V nasazení aplikace bylo provedeno několik změn. Všechny provedené změny jsem zanesl do diagramu nasazení. Aktuální stav lze vidět na obr. 6.3.



Obrázek 6.3: Diagram nasazení aktuální stav

6.2 Přihlašování

Při zkoumání možností autorizace vůči školním systémům jsem zjistil, že bude vhodné přihlédnout k politice s jakou ČVUT přistupuje k ochraně osobních údajů. Z dostupných údajů vyplynulo několik důležitých informací.

- Od 9. 2. 2015 je možná pouze šifrovaná komunikace s LDAP serverem – důvod proč momentálně nefunguje přihlášení
- Od 1. 5. 2015 bude zakázáno využívání služby LDAP pro autorizaci uživatelů (webových) aplikací
- Náhradou za ověřování pomocí LDAPu je Shibboleth SSO
- Preferovaný poskytovatel identit je cvutID
- Autorizace proti fakultním poskytovatelům identit je možná, ale nebude dlouhodobě podporován a jeho provoz bude v horizontu jednoho roku až dvou let ukončen

Z těchto bodů plyne to, že úprava LDAP autorizace na šifrovanou variantu je zbytečná. Bude tedy nutné upravit aplikaci, tak aby umožňovala autorizaci přes Shibboleth SSO. Pro Shibboleth SSO autorizaci by bylo vhodné využít doporučeného poskytovatele identit tzn. cvutID. Na serveru je aktuálně používán fakultní Shibboleth, který nebude možné použít a bude nutné jej

překonfigurovat pro cvutID. Dále také bude třeba koordinovat úpravy s projekty, které na serveru běží a využívají fakultní Shibboleth.

Pro nasazení a úpravu Shibbolethu jsem postupoval podle návodů vztahujícím se k této problematice, které jsou dostupné na wiki ČVUT.

6.2.1 Instalace OpenSSL certifikátů

Nejprve bylo nutné nahradit self-signed certifikát, který server doposud používal certifikátem podepsaným důvěryhodnou certifikační autoritou. Certifikát jsem si vyžádal u příslušné osoby a jeho instalaci jsem spojil s konfigurací HTTPS v rámci řešení problému s odesíláním hesel přes HTTP.

6.2.2 Konfigurace web serveru - Apache2

Pro používání Shibbolethu jsem se rozhodl použít modul, který je dostupný pro Apache2. Pro použití právě tohoto způsobu jsem se rozhodl z následujících důvodů.

- Vztahují se k němu návody na wiki ČVUT
- Tento způsob mi byl doporučen lidmi, kteří se starají o fakultní Shibboleth
- Na serveru je tento modul, již využíván
- Pro tento modul je dle mého názoru dostupné výrazně větší množství návodů, než je tomu u alternativ

Server je nakonfigurován tak, aby umožňoval přístup k některým url pouze autorizovaným uživatelům. Při požadavku na danou url je uživatel nejprve přesměrován na cvutID, po úspěšné autorizaci je pak schopen se dostat k dané url. Jedna z možností jak aplikace pozná, kdo k ní přistupuje, je přečtení HTTP hlaviček požadavku, které nesou potřebné informace.

6.2.3 Konfigurace SP

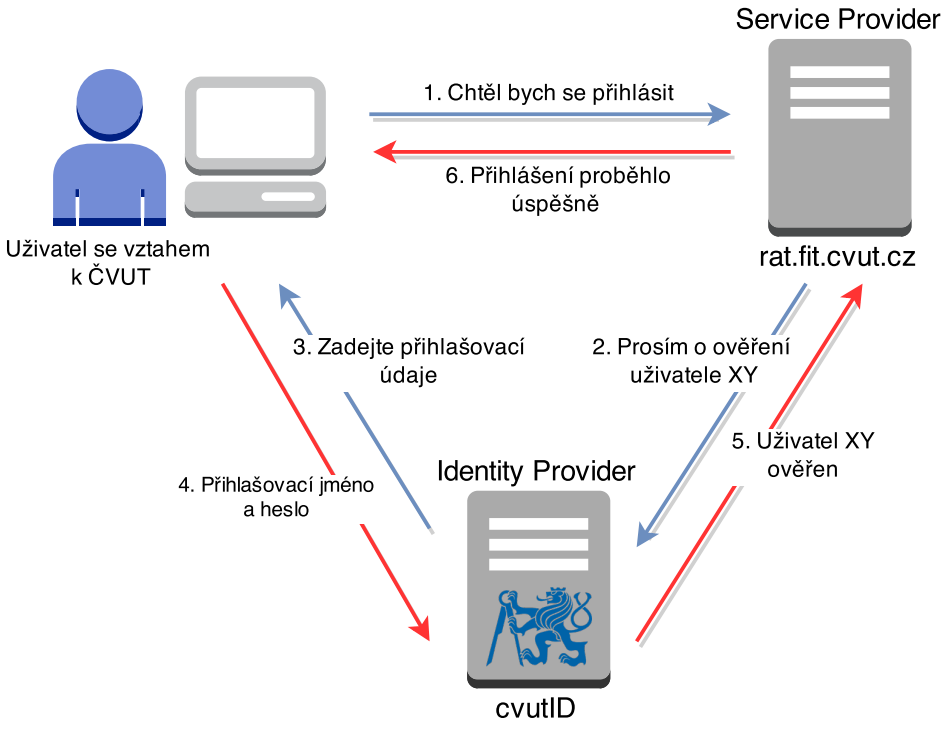
Při konfiguraci jsem vycházel z ukázkové konfigurace na wiki ČVUT. Tuto konfiguraci bylo třeba rozšířit o nastavení potřebná pro fungování jedné instance Shibbolethu pro více doménových jmen. Vzhledem nestandardní konfiguraci bylo třeba si nastudovat potřebné informace z oficiální dokumentace a následně doladit nastavení se správcem fakultního a školního Shibbolethu.

Jednalo se o nastavené prvku `NativeSPApplicationOverride`. Tento prvek slouží k nastavení instance Shibbolethu, aby si detekovala více doménových jmen. Zároveň také umožňuje kompletně podědit nastavení výchozí domény nebo si libovolnou část nastavení upravit pro potřeby dané domény.

Druhou změnou byla úprava konfigurace generátoru metedat. Ve výchozím nastavení jsou metadata generována pro každou doménu (endpoint) zvlášť. Bylo tedy potřeba nastavit generátor, aby správně detekoval všechny endpointy a generoval metedata pro všechny naráz. Druhou položkou, kterou bylo třeba nastavit, bylo mapování požadavků. Zde bylo třeba docílit toho, aby byla metadata pro všechny domény podepsána jako stejná doména. Jednotná metada jsou potřeba, aby identity provider nemusel mít registrovanou každou doménu zvlášť, když domény běží na stejném serveru.

6.2.4 Implementace cvutID do webové aplikace

Aplikace aktuálně podporuje „klasický“ způsob autorizace, tzn. aplikace dostane jméno a heslo a následně ho ověří vůči vlastní databázi nebo tyto údaje posílá k ověření na další server. Tento způsob autorizace je však velmi odlišný od toho, jak funguje autorizace v případě Shibbolethu.



Obrázek 6.4: Schéma autorizace pomocí Shibbolethu

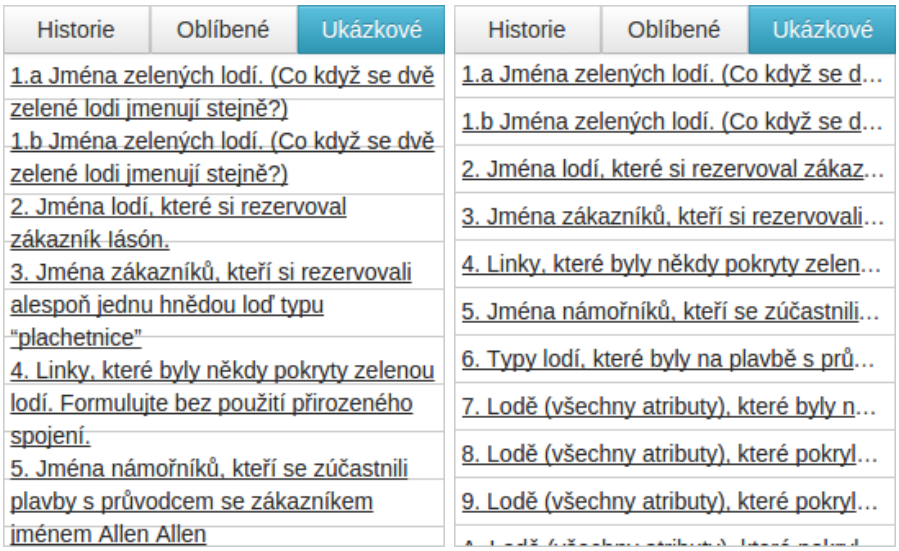
Shibboleth využívá princip jednotného přihlášení (Single Sign-On). Přihlášení probíhá tak, že se uživatel autorizuje pouze vůči jednomu serveru (identity provider). Přihlášení do dalších služeb (services provider) probíhá, tak že tyto

služby vždy požádají identity providera, aby uživatele autorizoval. Následná reakce services providera záleží na odpovědi identity providera (viz obr. 6.4).

Pro začlenění Shibboletu do aplikace bylo tedy nutné upravit strukturu webu, tak aby bylo možné vyhradit zabezpečenou část, do které bude povolen přístup pouze po úspěšné autorizaci přes cvutID. Do této části jsem následně umístil nově vytvořený kód, který se stará o autorizaci uživatelů. V tomto místě také dochází k registraci úspěšně autorizovaných uživatelů, kteří ještě nemají v aplikaci záznam.

6.3 Úpravy frontendu aplikace

V rámci úprav front-endu jsem se zaměřil na vyřešení problémů se zobrazením na PC. Zde byl problém pouze v prvku, který řeší zobrazení historie dotazů, ukázkových dotazů a oblíbených dotazů. Jednalo se o změny v CSS stylech a o změny v šabloně tohoto prvku. Výsledky změny jsou vidět na obrázku 6.6 a 6.5.



Obrázek 6.5: Rozdíl mezi starým a novým zobrazením ukázkových dotazů

Obrázek 6.6: Rozdíl mezi starým a novým zobrazením historie dotazů

Kapitola 7

Testování

Všechny provedené úpravy a nové kusy kódu jsem testoval. Cílem testování bylo ověřit jestli daná změna vyřešila daný problém, který měla řešit, a zároveň jestli nedošlo k změně chování aplikace. Ke změnám chování docházelo zejména z důvodu rozdílnosti verze aplikace, která byla nasazena na serveru, a verze aplikace v repositáři. Zdrojové kódy obsahovaly také podmínky, které byly vázány na konkrétní doménové jméno. Výsledkem toho bylo, že aplikace měnila své chování podle toho, kde byla spuštěna.

7.1 Hesla přes HTTP

Veškerá komunikace se serverem probíhá šifrovaně přes protokol HTTPS. Při přihlašování pomocí školních přihlašovacích údajů tyto údaje vůbec nejdou přes server. Probíhá zde ověřování přes cvutID. Důsledkem toho je, že při packet sniffingu lze zachytit pouze data, která jsou v zašifrované podobě. Odolnost vůči sniffingu jsem testoval zopakováním testu, kdy jsem odchytával packety pomocí programu Wireshark. Zachycený provoz však z principu věci nelze při tomto typu útoku dešifrovat.

Použitá konfigurace HTTPS na straně serveru by zároveň měla být odolná vůči slabinám OpenSSL, které lze využít k man-in-the-middle útokům. Toto nastavení jsem testoval pomocí testu na webu ssllabs.com.

7.2 Nešifrovaná komunikace s LDAP serverem

Možnost přihlášení přes LDAP jsem zrušil a nahradil ji přihlašováním přes cvutID. Tento způsob bude podle informací ČVUT do budoucna preferován a mělo by se tedy jednat o dlouhodobě nejlepší řešení. Tento způsob autorizace je také „důvěryhodnější“, jelikož aplikace jako taková nepřijde do kontaktu a s heslem uživatele.

Nový způsob přihlašování jsem testoval na oba způsoby jeho využití. Tím je jednak přihlášení nového uživatele, kdy je třeba zanést tohoto uživatele do systému. Druhou variantou je pak párování uživatelů s účty, které byly vytvořeny již dříve. Test jsem provedl s 5 uživateli. U všech fungovala jak registrace tak párování bez problémů.

7.3 Problémy se zobrazením aplikace

Tento problém jsem nevyhodnotil jako kritický, proto sem ho řešil pouze částečně. Opravy se týkaly komponenty zobrazující historii dotazů, oblíbené dotazy a ukázkové dotazy. Testování ukázalo, že provedené změny způsobily to, že komponenta plně využívá prostor který má. Zároveň byl také vyřešen problém s vytékáním textu mimo stanovenou oblast. Při testování se zde však objevil nový problém s chováním některých prvků na událost kliknutí. Nasazená verze měla tento problém vyřešený. Změny, které ho řešily, však nejsou zaneseny v repositáři.

7.4 Nelze načíst některé zdroje ze serveru

Problém byl vyřešen úpravami v nasazení aplikace na serveru. Od provedení těchto změn se již problém neopakoval. Bohužel zde nebyla možnost provést test v plné zátěži, které je server vystaven v době kolem odevzdávání semestrálních prací na BI-DBS. Jelikož se však jednalo o známý bug, toto řešení by mělo problém plně řešit.

7.5 Nefunkční odesílání mailů

Aplikace byla nastavena podle původní dokumentace. Několik změn navíc bylo provedeno pouze na straně bezpečnostních nastavení účtu na gmailu. Odesílání mailů je nyní funkční. Testování jsem provedl na několika testovacích účtech. Testoval jsem jak registraci nového uživatele s lokální heslem, tak obnovu hesla u již existujícího účtu.

7.6 Nestabilita aplikace

Aplikace zatím běží na původním překladači, který pravděpodobně způsobuje pády aplikace. V rámci oprav tohoto problému jsem navrhl nový překladač, implementoval některé jeho části a vytvořil jsem aplikační rozhraní pro komunikaci mezi překladačem a webovou aplikací. Komunikace je řešena pomocí REST API rozhraní na straně aplikace, které je autorizované přes školní OAuth2.

7.6.1 Překladač

První testovatelné verze překladače byla dokončena týden před termínem odevzdání BP. Na jeho otestování bohužel nebyl čas.

7.6.2 Aplikační rozhraní

Vytvořená aplikační rozhraní byla testována proti sobě. Úspěšně jsem otestoval sestavení přístupového tokenu na straně webové aplikace a jeho ověření na straně překladače. Dále sem také úspěšně otestoval komunikaci pomocí přes REST API rozhraní, kde jsou data uložena ve formátu JSON.

Kapitola 8

Aktuální stav

Aplikace s provedenými úpravami běží na adrese rat.fit.cvut.cz. V rámci úprav aplikace jsem úplně nebo částečně vyřešil následující problémy:

- Hesla přes HTTP
- Nešifrovaná komunikace s LDAP serverem
- Problémy se zobrazením aplikace
- Nelze načíst některé zdroje ze serveru
- Nefunkční odesílání mailů
- Nestabilita aplikace
  - Návrh překladače
  - Implementace lexikálního analyzátoru
  - Implementace syntaktického analyzátoru
  - Vytvoření komunikačního rozhraní

8.1 Webová aplikace

Webová aplikace s provedenými změnami je uložena na serveru v nově vytvořeném git repositáři. Všechny provedené změny byly zdokumentovány komentáři ve zdrojovém kódu. Aplikace má doladěné nasazení a do aktuální verze byla přenesena veškerá uživatelská data z předchozího provozu. Provedené změny se pozitivně projevily na funkčnosti aplikace. Nyní je aplikace ve stavu kdy je možné ji používat.

I přes doplnění dokumentace v nově upravených částech aplikace a vytvoření nového repositáře, je stav v tomto ohledu stále velmi špatný. Dokumentace

ve zbytku kódu je neúplná a některé třídy jsou dostupné pouze ve zkompilované podobě. Tento fakt silně komplikoval a zvyšoval časovou náročnost implementace nových částí aplikace.

S použitím poslední verze aplikace, která byla dostupná v repositáři, se v aplikaci objevilo několik dalších chyb. Tyto chyby byly v původně nasazené verzi aplikace vyřešeny. Změny v kódu, které je opravovaly, se bohužel do verze v repositáři nedostaly. Dále se zde také objevil problém, s chováním aplikace po nasazení na server. Zjistil jsem, že aplikace se chová jinak po nasazení na doménu rat.fit.cvut.cz, protože některé kusy kódu jsou vázané na konkrétní doménové jméno.

8.2 Překladač

Vzhledem ke stavu dostupných překladačů, jsem navrhl nový překladač. Sám jsem provedl část jeho implementace a to konkrétně implementaci lexikálního a syntaktického analyzátoru. Další část byla implementována studenty v rámci předmětu BI-SP1. Z jejich strany bohužel nebylo na překladači provedeno tolik práce kolik by bylo potřeba, aby mohl být překladač nasazen.

Momentálně je překladač ve stavu kdy dokáže překládat dotazy z relační algebry do OracleSQL. Tento překlad je schopen provádět jak se bez znalosti schématu databáze tak s ním. Bude třeba provést první testování a doladění komunikace mezi překladačem a webovou aplikací. V komunikaci je nutné dořešit zejména zpracování chybových hlášení.

Veškerá implementace byla zdokumentována a je uložena v git repositáři na serveru.

Závěr

Tato práce vyžadovala velmi široké spektrum znalostí. Bylo potřeba orientovat se v oblastech od administrace linuxového serveru, přes programování v několika jazycích, až po problematiku návrhu překladačů.

Již z analýzy vyplynulo, že aplikace nebyla v dobrém stavu. Webová aplikace byla velmi špatně zdokumentována a její zdrojové kódy jsou v repositáři, jenž byl veden chaoticky. Jakákoliv práce, která se týkala této části, byla z těchto důvodů velmi obtížná a zabrala výrazně více času než by měla. Vzhledem k tomu, že tato část aplikace 2 roky funguje, rozhodl jsem se ji v projektu ponechat a provádět v ní další vývoj.

Druhou částí projektu je překladač. Jeho aktuálně používaná verze dosluhuje a měla být nahrazena novou implementací. Vývoj této implementace byl ovšem velmi problematický a o její kvalitě bylo mnoho pochybností. Z analýzy nové implementace vyplynulo, že tyto pochybnosti byly na místě. Překladač nebyl dobře vymyšlen a zároveň i jeho implementace byla nekvalitní. I přesto byla tato verze lepší než aktuálně používaná implementace překladače. Vzhledem k jejímu stavu bylo rozhodnuto, že její použití by do budoucna znamenalo jen mnoho problémů a její další vývoj by byl jen ztrátou času.

V rámci práce na aplikaci jsem navrhl nový překladače, který má čitelný kód a je navržen, tak aby se dal snadno upravovat a rozšiřovat. Dále bylo opraveno nasazení aplikace a byly opraveny komponenty aplikace, které byly nutné k jejímu fungování. Poslední aktivitou pak byl návrh a vytvoření jednoduchého aplikačního rozhraní pro komunikaci mezi webovou aplikací a překladačem.

Výsledkem práce je aplikace aktuálně nasazená na doméně rat.fit.cvut.cz. Aplikace je díky provedeným změnám výrazně stabilnější, bezpečnější a je ve stavu, kdy ji lze používat. Aplikace je připravena na nasazení nového překladače, po jehož nasazení by měly být vyřešené všechny problémy se stabilitou. Při přechodu na novou verzi byla uživatelům přenesena všechna data. Poslední dotupná verze aplikace z repositáře, ve které byl prováděn vývoj, obsahovala další chyby. Tyto chyby byly v nasazených verzích, ze kterých jsem vycházel opravené. Aktuálně bežící aplikace tedy obsahuje několik chyb o kterých se

nezmiňuji. Žádná z těchto chyb však není kritická.

Stav aplikace je i přes provedené změny nadále špatný. Veškeré prováděné změny, díky chaoticky vedenému repositáři a špatné dokumentaci, zabraly výrazně více času než by měly. Aplikace je momentálně funkční, což by mělo poskytnout čas na to ji zrevidovat případně přepsat.

Jako největší přínos této práce beru to že přinesla ucelený pohled na aplikaci RAT. Dalším přínosem je překladač, který je navržen tak aby se dal dlouhodobě používat. Pokud se povede věnovat trochu času jeho testování a ladění má parametry na to, aby se dal do budoucna využívat a rozšiřovat dle potřeby. Jednoznačným přínosem je také to, že aplikaci je momentálně možné používat. Na začátku tohoto semestru byla aplikace mnohdy nedostupná. Pokud se povedlo ji načíst nebylo možné se přihlásit a tím pádem nešlo využívat valnou většinu funkcí aplikace.

Jako další možnosti vývoje vidím doladění a úpravy překladače. Druhou oblastí je rozvoj webové aplikace. Po mých zkušenostech však musím konstatovat, že by zde byla potřeba minimálně kompletní revize kódu. Osobně bych se po zkušenostech s realizací změn v implementaci klonil spíše k variantě začít na „čisté louce“ a implementovat vše znovu na základě zkušeností získaných z této práce.

Literatura

[1] doc. Ing. Jan Janoušek, P.: *Struktura překladače, úvod do předmětu [online]*. FIT ČVUT, [cit. 2015-04-04]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-PJP/_media/lectures/01/prednaska1.pdf

[2] FIT ČVUT: *BI-SP1 - Softwarový týmový projekt 1 [online]*. [cit. 2015-05-02]. Dostupné z: http://bk.fit.cvut.cz/cz/predmety/00/00/00/00/00/00/01/12/36/p1123606.html

[3] FIT ČVUT: *BI-SP2 - Softwarový týmový projekt 2 [online]*. [cit. 2015-05-02]. Dostupné z: http://bk.fit.cvut.cz/cz/predmety/00/00/00/00/00/00/01/12/56/p1125606.html

[4] Jirůtka, J.: *OAuth 2.0 [online]*. [cit. 2015-04-20]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/oauth2

[5] Oracle Corporation: *Java SE Overview - at a Glance [online]*. [cit. 2015-03-19]. Dostupné z: http://www.oracle.com/technetwork/java/javase/overview/index.html

[6] Java webnotes, Sun: *Version 6 [online]*. [cit. 2015-03-19]. Dostupné z: http://java.sun.com/javase/6/webnotes/version-6.html

[7] Oracle Corporation: *Java Naming [online]*. [cit. 2015-03-19]. Dostupné z: http://www.oracle.com/technetwork/java/javase/overview/javanaming-2227065.html

[8] Oracle Corporation: *Oracle Java SE Support Roadmap [online]*. 2013, [cit. 2015-03-19]. Dostupné z: http://www.oracle.com/technetwork/java/eol-135779.html

[9] Oracle Corporation: *Differences between Java EE and Java SE [online]*. [cit. 2015-03-19]. Dostupné z: http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html

[10] Clingan, J.: *GlassFish Server 3.1.2.2 Now Available [online]*. [cit. 2015-03-19]. Dostupné z: https://blogs.oracle.com/theaquarium/entry/glassfish_server_3_1_2

[11] The PostgreSQL Global Development Group: *What is PostgreSQL? [online]*. [cit. 2015-03-19]. Dostupné z: http://www.postgresql.org/docs/current/static/intro-whatis.html

[12] The PostgreSQL Global Development Group: *PostgreSQL 9.4.1, 9.3.6, 9.2.10, 9.1.15 & 9.0.19 Released [online]*. [cit. 2015-03-19]. Dostupné z: http://www.postgresql.org/about/news/1569/

[13] oleksiys: *Wrong magic number w/ ajp [online]*. [cit. 2015-03-19]. Dostupné z: https://java.net/jira/browse/GRIZZLY-1340

[14] FIT ČVUT: *BI-DBS-Databázové systémy - B141 [online]*. [cit. 2015-03-21]. Dostupné z: https://timetable.fit.cvut.cz/archive/B141/public/cz/predmety/11/22/p1122406.html

[15] Odersky, M.; Rompf, T.: Unifying functional and object-oriented programming with Scala. *Communications of the ACM*, ročník 57, č. 4, 2014: str. 76, doi:10.1145/2591013. Dostupné z: http://dl.acm.org/citation.cfm?doid=2580723.2591013

[16] The C++ Resources Network: *Standard C++ Library reference [online]*. [cit. 2015-03-27]. Dostupné z: http://www.cplusplus.com/reference/

[17] Tutorials Point: *C++ goto statement [online]*. [cit. 2015-03-27]. Dostupné z: http://www.tutorialspoint.com/cplusplus/cpp_goto_statement.htm

[18] Bílek, P.: *Podmínky a cykly - Skok goto [online]*. [cit. 2015-03-27]. Dostupné z: http://www.sallyx.org/sally/c/c12.php

[19] cppreference.com: *goto statement [online]*. [cit. 2015-03-27]. Dostupné z: http://en.cppreference.com/w/cpp/language/goto

[20] PC Magazine: *Definition of:compiler [online]*. [cit. 2015-04-04]. Dostupné z: http://www.pcmag.com/encyclopedia/term/40105/compiler

[21] Farrell, J. A.: *Anatomy of a Compiler [online]*. [cit. 2015-04-04]. Dostupné z: http://www.cs.man.ac.uk/~pjj/farrell/comp3.html

[22] Shao, Z.: *Compilers and interpreters [online]*. Yale University, [cit. 2015-04-04]. Dostupné z: http://flint.cs.yale.edu/cs421/lectureNotes/c04.pdf

[23] Grune, D.; Jacobs, C. J. H.: *Parsing Techniques - A Practical Guide*. Ellis Horwood, první vydání, 1990, ISBN 0 13 651431 6. Dostupné z: http://dickgrune.com/Books/PTAPG_1st_Edition/

[24] van Rossum, G.: *Python - zdrojové kódy [online]*. Python Foundation, [cit. 2015-04-05]. Dostupné z: http://svn.python.org/view/*checkout*/python/trunk/Misc/HISTORY

[25] Summerfield, M.: *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. Prentice Hall, 2007, ISBN 9780132354189. Dostupné z: http://www.cs.washington.edu/research/projects/urbansim/books/pyqt-book.pdf

[26] Python Software Foundation: *Unicode HOWTO [online]*. [cit. 2015-04-05]. Dostupné z: https://docs.python.org/2/howto/unicode.html

[27] Canonical Ltd.: *Software Packages in "vivid", Subsection python [online]*. [cit. 2015-04-05]. Dostupné z: http://packages.ubuntu.com/vivid/python/

[28] Python Software Foundation: *PyPI - the Python Package Index [online]*. [cit. 2015-04-05]. Dostupné z: https://pypi.python.org/pypi

[29] Beazley, D.: *PLY (Python Lex-Yacc) [online]*. [cit. 2015-04-06]. Dostupné z: http://www.dabeaz.com/ply/

[30] Software in the Public Interest, Inc.: *Lex and Yacc implementation for Python2 [online]*. [cit. 2015-04-06]. Dostupné z: https://packages.debian.org/stable/python/python-ply

[31] Red Hat: *python-ply [online]*. [cit. 2015-04-06]. Dostupné z: https://admin.fedoraproject.org/pkgdb/package/python-ply/

[32] Python Software Foundation: *ply 3.4 [online]*. [cit. 2015-04-06]. Dostupné z: https://pypi.python.org/pypi/ply

[33] Christie, T.: *Django REST framework [online]*. [cit. 2015-04-20]. Dostupné z: http://www.django-rest-framework.org/

Příloha A

Seznam použitých zkratek

RA Relační algebra

RAT Relation Algebra Translator

FIT Fakulta informačních technologií

ČVUT České vysoké učení technické v Praze

ZS Zimní semestr

LS Letní semestr

IDE Integrated Development Environment

OOP Objektově orientované programování

AST Abstraktní syntaktický strom

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `newRAT` ..... snapshot repositáře obsahujícího novou implementaci překladače
    - `RA` ..... snapshot repositáře z původního projektu RAT
    - `RAT` ..... snapshot repositáře obsahujícího nejnovější verzi webové aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `Brabemi3_BP_UTF-8.pdf` ..... text práce ve formátu PDF
  - `Brabemi3_zadani.pdf` ..... zadání práce ve formátu PDF