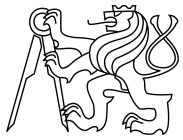
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Návrh a implementace integrace služeb do projektu EBIE
Student:	Cyril Černý
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

EBIE (Extended Business Inteligence Encyclopedia) je specializovaný portál pro zobrazení metadat a byznys kontextu nad datovým skladem.

Cílem této práce je integrace EBIE, tiketovacího systému Jira a IDM (Identity Manager).

1. Seznamte se se stávající informační architekturou EBIE.
2. Navrhněte a implementujte integraci EBIE a tiketovacího systému Jira - tj. umožněte z EBIE: vytvoření tiketu (žádosti o sestavu), zobrazení stavu řešení tiketu, přímý přístup k příslušnému tiketu do systému Jira a uzavření tiketu.
3. Seznamte se s koncepcí technických a byznys rolí v centrálním IDM ČVUT a fakultním IDM FIT.
4. V souladu s touto koncepcí navrhněte systém technických a byznys rolí pro práci s EBIE.
5. Navrhněte technickou realizaci přístupu k datům na základě rolí v portálu EBIE.
6. Proveďte prototypovou implementaci, prototyp otestujte, zdokumentujte a zhodnoťte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 11. listopadu 2015

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

Bakalářská práce

Návrh a implementace integrace služeb do projektu EBIE

Cyril Černý

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

12. května 2016

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Valentovi, PhD. za jeho trpělivost a vždy positivní náladu při vedení této práce.

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 12. května 2016 .....

Odkaz na tuto práci

Černý, Cyril. *Návrh a implementace integrace služeb do projektu EBIE*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Práce se zabývá řešením integrace nově vznikajícího portálu EBIE s poskytovatelem identit a ticketovacím systémem. K přihlašování je využit autentizační server Zuul. K integraci s ticketovacím systémem se používá JIRA REST API. Výsledkem je demo aplikace demonstrující navrhované řešení.

Klíčová slova Integrace, REST, JIRA REST API, Autentizace, Role, Ruby on Rails

Abstract

This thesis is about integration of new portal EBIE with identity provider and ticket system. For user sign-in is used authentication server Zuul. Integration with ticket system is done by JIRA REST API. The result of this thesis is demo application showing proposed solution.

Keywords Integration, REST, JIRA REST API, Autentization, Roles, Ruby on Rails

Obsah

Seznam obrázků

Úvod

V dnešní době větší organizace používají více informačních systémů. Ke sjednocení dat z několika zdrojů tak nasadí datový sklad. Bez kvalitní a jednotné dokumentace datových entit a sestav by však stav datového skladu nebyl udržitelný a budoucí vývoj by stál příliš úsilí. Na ČVUT takový popis chyběl a proto na ČVUT FIT vniká portál pro popis entit, procesů a sestav z datového skladu, který rovněž vzniká na FIT.

K usnadnění práce s portálem je potřeba jej integrovat s existujícími ČVUT systémy. K snazšímu zadávání požadavků na nové sestavy je nutné propojit portál a ticketovací systém JIRA, na kterém je provozován ČVUT Helpdesk. Protože data nemohou být veřejně dostupná, je vyžadována integrace s poskytovatelem identit k autentizaci a autorizaci uživatelů z ČVUT. Integrace s těmito službami zajistí uživateli pohodlnější používaní katalogu. Tato práce se zabývá integrací nově vznikajícího portálu EBIE s poskytovatelem identit a ticketovacím systémem.

Ze začátku budou určeny cíle práce a představíme si základní strukturu EBIE. Následovat bude stručné shrnutí souvisejících technologii. Po úvodních kapitolách je práce rozdělena na dvě hlavní části.

První část se zabývá integrací s ticketovacím systémem JIRA. V úvodu bude provedena analýza požadavků a možných řešení. Na základě této analýzy navrhneme řešení a implementujeme jej.

Druhá část se zabývá integrací s poskytovatelem identit na ČVUT. Po prvotní analýze požadavků se podíváme na strukturu ČVUT rolí. Navážeme využitím těchto rolí v EBIE a možnostmi přístupu k údajům o uživateli. Nakonec navrhneme řešení a demonstrujeme jeho implementaci.

Ke konci práce bude popsáno otestování výsledné demo aplikace a vysloven závěr.

Kapitola 1

Cíl práce

Cílem práce je integrace portálu EBIE s ticketovacím systémem JIRA a manažerem identit na ČVUT. Pro splnění cíle je nutné analyzovat poskytované služby zmiňovaných systémů. Pokud je poskytnutí služby něčím podmíněno, je potřeba zajistit splnění těchto požadavků. Po analýze je možné vytvořit návrh přístupu k systémům a následně zpracovat dostupná data z nabízených služeb.

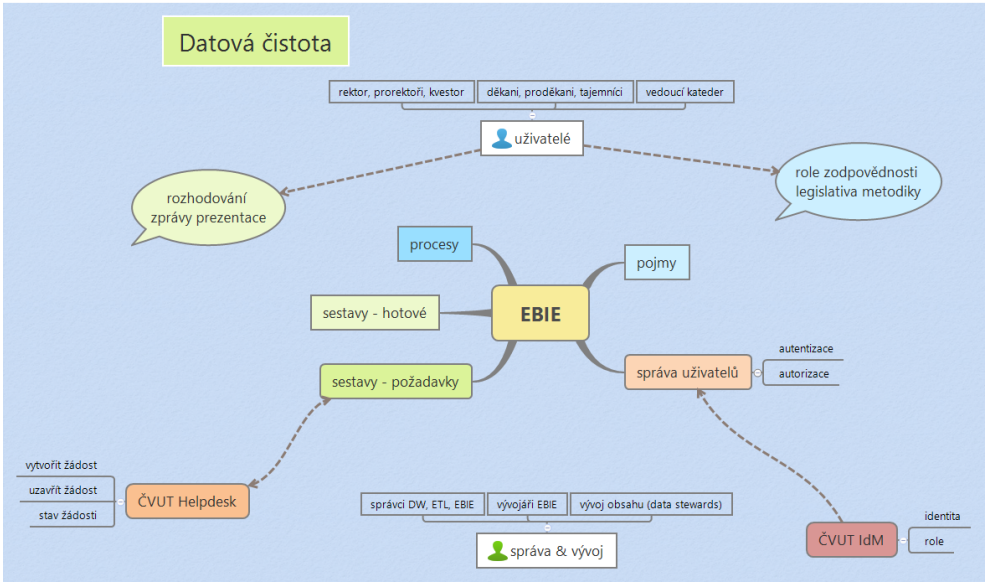
Pro integraci se systémem JIRA bude potřeba provést analýza poskytované API a zajistit přístup k ticketům v tomto systému. Uživatel bude mít možnost vytvořit, zjistit stav a uzavřít vlastní tickety přímo v portálu EBIE aniž by po něm bylo vyžadováno jakkoliv interagovat se systémem JIRA.

Při integraci s identity manažerem na ČVUT provedeme analýzu existující byznys rolí. Tyto role se namapují pro potřeby v portálu EBIE. Přístup do portálu EBIE bude zajištěn na základě navrženého mapování.

Kapitola 2

EBIE

BI encyklopedie je označení pro dokumentaci datových entit a procesů. Ve velkých organizacích bývá nasazeno několik samostatných systémů naráz, které spolu spolupracují. BI Encyklopedie slouží k udržení přehledu o datech a jejich souvislostech. EBIE (Extended Business Inteligence Encyclopedia) je portál vznikající v rámci projektu Datová čistota na ČVUT FIT. Jedná se o BI Encyklopedii rozšířenou o ontologickou a infologickou vrstvu. Cílem EBIE je nabídnout pohodlný přístup k popisu datových entit a sestav pro vedoucí pracovníky školy. Na obrázku 2.1 je naznačen koncept zamýšleného nasazeni EBIE v prostředí ČVUT. Cílem této práce je integrace se systémy ČVUT Helpdesk a ČVUT IdM, jak je vyznačeno ve spodní části obrázku.



Obrázek 2.1: Koncept architektury EBIE

Kapitola 3

Popis technologií

V této kapitole si představíme technologie, se kterými budeme pracovat. Bude se jednat o již využité jazyky při vývoji portálu EBIE, nebo důležité technologie pro integraci aplikací.

3.1 Ruby

Ruby[1] je objektově orientovaný skriptovací jazyk inspirovaný jazyky jako Python, Perl, Smalltalk, Ada nebo Lisp. V Ruby je vše objekt. Těmto objektům lze posílat zprávy na které odpovídají. Neexistují zde primitivní typy. Třídy nejsou uzavřené a lze je i za běhu upravit. Přechod k Ruby není pro lidi se zkušenostmi s objektovým programováním obtížný. Správa paměti je realizována přes tzv. „garbage collector“, který automaticky uvolňuje nepoužívanou paměť.

3.1.1 Gemy

Při vývoji v Ruby se naskytne potřeba volat nejen standardní knihovní funkce. Jako v jiných jazycích je i v Ruby možné přidávat knihovny. V Ruby se tyto knihovny jmenují „gemy“. Gemy se spravují přes příkazovou řádku nástrojem `gem`. Pro instalaci nového gemu tak stačí zavolat `gem install`. V kódu pak knihovnu použijeme příkazem `require`.

Bundler je správce gemů pro Ruby. Instaluje se příkazem `gem install bundler`. Umožňuje vytvořit ve složce projektu soubor `gemfile`, ve kterém specifikujeme potřebné závislosti. Při zavolání příkazu `bundle install` ve projektu se zkontrolují všechny zmíněné gemy. Pokud nějaký chybí, automaticky se stáhne a nainstaluje.

3.1.2 Ruby on Rails

Ruby on Rails[2] (RoR) je Model-View-Controller(MVC) framework pro vývoj webových aplikací. Framework je distribuován ve formě gemu. Ke správě závislostí používá nástroj `bundler`. RoR podporuje rychlý vývoj aplikací pomocí zabudovaných příkazů, které generují kód pro obvyklé úkony. Po instalaci RoR (`gem install rails`) je možné vytvořit novou aplikaci přes `rails new newapp`, kde `newapp` je jméno nové aplikace. Generátor automaticky připraví základní strukturu webové aplikace v RoR. Popíšeme si klíčové části.

Model Reprezentuje data, se kterými se pracuje. Definuje objekty v aplikaci a stará se o jejich persistenci. V RoR je pro persistenci modelu implicitně použitý ORM framework `Active Record`.

View Určuje vzhled prezentovaného obsahu. V RoR aplikaci je možné využit speciální syntaxe pro vkládání kusů kódu Ruby do HTML. Tyto soubory pak mají koncovku `html.erb`.

Controller Zpracovává požadavky přicházející na server. Pracuje s modelem a posílá data do `view` k zobrazení.

Ruby on Rails se stará o směrování všech požadavků na základě definic v souboru `routes.rb` do controllerů. Vývojáři se tak mohou soustředit na vývoj vlastní aplikace.

3.2 REST

REST(representional state transfer) je metoda komunikace mezi aplikacemi. Webové služby, které používají REST pro vystavení API se nazývají „RESTful APIs“. Ke komunikaci se využívá HTTP protokol. K volání zdrojů je potřeba adresa (URI, Unified Resource Identifier) a HTTP metoda (GET, POST a další). Pokud je potřeba předat data, posílají se většinou ve formátu JSON. Záleží však na serveru jaký formát přijmá. Nejčastější HTTP metody (POST, GET, PUT, DELETE) se dají namapovat na CRUD(Create, Read, Update, Delete) operace.

3.3 OAuth

OAuth je otevřený protokol pro zabezpečení webových služeb. Momentálně se používá OAuth 2.0 podle RFC 6749[3]. Protože OAuth 2.0 není kompatibilní se staršími verzemi, někteří poskytovatelé služeb stále používají normu OAuth 1.0a definovanou v RFC 5849[4]. OAuth 2.0 umožňuje aplikaci třetí strany limitovaný přístup ke službě v zastoupení uživatele. Aplikace třetí strany nemá přístup k přihlašovacím údajům uživatele. K přístupu se používá tzv. `access token`, který má krátkou životnost. Pokud aplikace bude chtít obnovit platnost

tohoto tokenu, použije k tomu `refresh token`. Uživatel při prvním přihlášení do aplikace třetí strany autorizuje aplikaci u autorizačního serveru. Autorizační server může uživateli popsat, která oprávnění aplikace vyžaduje a podle toho se může uživatel rozhodnout. Uživatel může kdykoliv odebrat oprávnění k přístupu. Aplikace tak bude muset znova zažádat o autorizaci.

OAuth verze 1.0a a 2.0 mezi sebou nejsou kompatibilní. Jejich hlavní odlišnost je v nutnosti podepisovat požadavky (OAuth 2.0 nevyžaduje podepisovat) a v životnosti (platnosti) tokenů. U OAuth 2.0 je doporučená platnost tokenu v řádech minut až hodin na rozdíl od nekonečné životnosti u verze 1.0a. Prošlý token tak musí aplikace obnovit pomocí `refresh token`.

OAuth autorizaci k webovým API používají firmy jako Google, Facebook nebo Twitter. Například přihlášení pomocí Facebook, které je na mnoha webových stránkách, využívá právě OAuth k autentizaci a zjištění informací o uživateli.

3.3.1 OAuth 2.0 Autorizační server Zuul

Na ČVUT FIT je vyvíjen OAuth 2.0 autorizační server (OAAS) Zuul, splňující RFC 6749[3]. OAAS Zuul je open-source a lze jej nalézt na GitHub[5]. Na ČVUT FIT je nasazen a používá se k autentizaci u těchto API:

- KOSapi
- Sirius
- Usermap API
- VVVSapi

Umožňuje vytvářet v prostředí ČVUT nové aplikace, které k autentizaci uživatelů využijí jejich existující ČVUT účty. Otevírá se tím cesta vzniku užitečných aplikací, které využívají zmíněné API. Jako příklad může být nové rozhraní pro zobrazení rozvrhů `Fittable` nebo online cvičebnice MARAST (MAtematika RAdoSTně).

Kapitola 4

Integrace se systémem JIRA

JIRA[6] je komerční ticketovací systém od společnosti Atlassian. Ticketovací systémy se používají k evidenci chyb, problémů a úkolů při vývoji software. Po dokončení vývoje a nasazení se dá využít jako helpdesk, kam lidé posílají problémy s používaným software. Na ČVUT se JIRA používá především k řízení a podporu projektů, které nemusí být nutně softwarové.

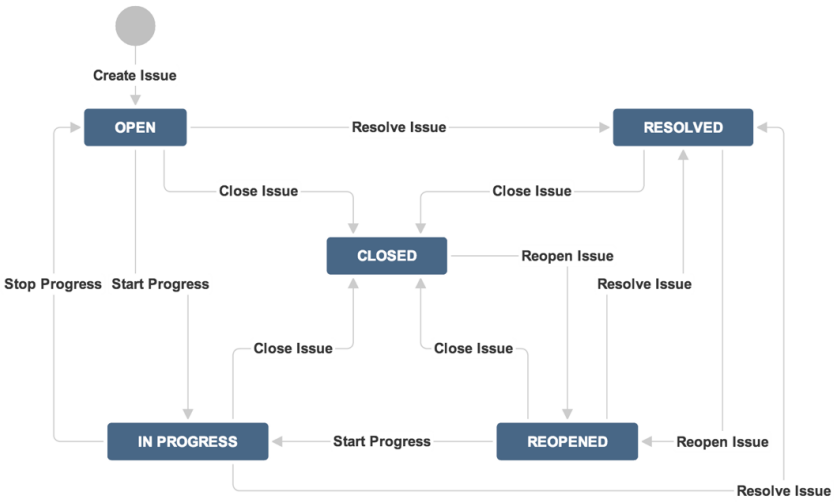
Ticketovací systém běžně umožňuje vytvořit ticket který má svůj životní cyklus. Tento ticket může reprezentovat například bug, problém, připomínku nebo úkol. Po vytvoření ticketu je v otevřeném stavu a může být někomu přiřazen (například kompetentní osobě) k vyřízení. V závislosti na definovaném postupu pak tento ticket přechází přes různé stavy až do koncového stavu. Koncových stavů může být více kde koncový stav indikuje důvod uzavření (vyřešeno nebo zamítnuto). Životní cyklus ticketu je znázorněn obrázkem 4.1.

V této kapitole bude rozebrán první ze dvou hlavních úkolů práce. Postupně rozebere analýzu zadání a rozebere možnosti pro řešení. Následně navrhne a zvolí přistup a demonstruje implementaci.

4.1 Analýza požadavků

V datových skladech je velké množství informací a není dobré je vystavit všechny v surové podobě. Uživatelé by byly nuceni si vytvářet složité filtry a zabralo by jím mnoho času než by se dostali k podstatné informaci. Proto se budou vytvářet tzv. sestavy. Díky sestavám si uživatelé budou moci prohlížet data v interaktivní podobě reprezentující nějakou konkrétní problematiku (počet přijatých studentů, studijní výsledky, úspěšnost předmětů). Je mnoho potenciálních sestav k vytvoření, ale bylo by neefektivní jich vytvořit příliš velké množství, protože by některé mohly být nevyužité. Aby byl investován čas pouze do užitečných sestav, budou si moci uživatelé zažádat o vytvoření nové podle jejich požadavků.

Diagram užití 4.2 znázorňuje požadavky. V portálu EBIE bude uživateli umožněno vytvořit a spravovat žádost k vytvoření nové sestavy. Tato žádost



Obrázek 4.1: Implicitní životní cyklus issue v systému JIRA. Převzato z webu Atlassian[7].

bude odeslána do ticketovacího systému JIRA (ČVUT Helpdesk). Aby uživatel nebyl zatěžován přihlašovaním se do jiného systému, bude celý proces podání žádosti spravován přímo v portálu EBIE.

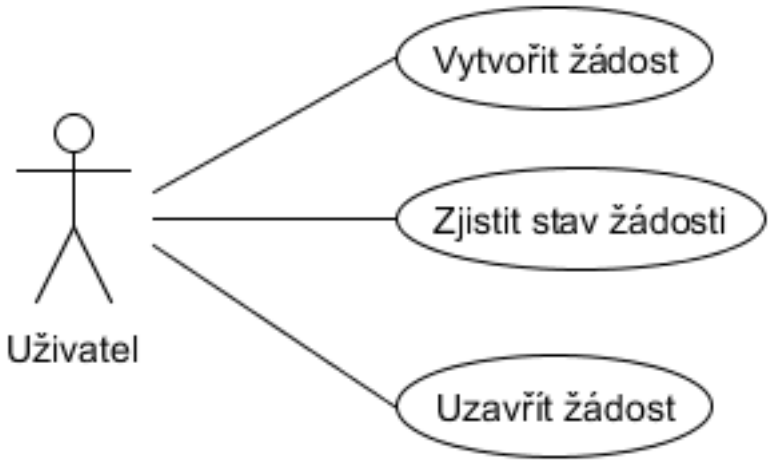
Úkolem je vyřešit komunikaci mezi EBIE a JIRA tak, aby uživatel přihlášený do EBIE mohl vytvářet a uzavírat tickety v JIRA aniž by opustil portál EBIE.

4.2 JIRA REST API

JIRA vystavuje REST API pro přístup k projektům a ticketům. Pojďme se nyní podívat, zda JIRA REST API[8] nabídne dostatečné nástroje pro správu ticketů z prostředí EBIE.

4.2.1 Autentizace přes REST API

Pro používaní systému JIRA přes webové rozhraní je vyžadováno, aby uživatel byl přihlášen. Pro REST operace tomu není jinak. Před samotným používáním REST API je tedy třeba se autentizovat. JIRA nabízí několik možností. Preferovaná metoda je OAuth (v1) a HTTP Basic (s nutností SSL/TLS). Další možností jsou HTTP Cookies.



Obrázek 4.2: Případy užití pro uživatele

4.2.1.1 HTTP Basic

Tato metoda je velmi přímá. V hlavičce každého požadavku se odešlou přihlašovací údaje (jméno a heslo) v base64 kódování. Protože jsou citlivé údaje pouze zakódovány a ne zašifrovány, je nutné použít SSL/TLS, při kterém se komunikace zabezpečí. Dotaz v nástroji curl s využitím základní autorizace může vypadat například takto:

```
curl –D– –X GET –H "Authorization: Basic
    dXNlcm5hbWU6cGFzc3dvcmQ=" –H "Content–Type:
    application/json" "https://helpdesk.cvut.cz/jira/rest
    /api/latest/myself"
```

4.2.1.2 Cookie autentizace

Tato metoda je založena na vytvoření relace (session) přes REST API a následně posílání session cookie v hlavičkách požadavků. Pro vytvoření relace klient odešle POST na endpoint `/rest/auth/1/session` se svými přihlašovacími údaji (v JSON). Jako odpověď přijde JSON se session cookie a ostatními informacemi, které momentálně nejsou podstatné. Klient si uloží přijatou cookie (v případě webového prohlížeče se nastaví automaticky v důsledku hlavičky „Set-Cookie“ v odpovědi) a pro pro následné volání REST API bude nastavovat v hlavičce `cookie:JSESSIONID={uložená hodnota}`. Stejně jako v předchozí metodě se zde posílají citlivé informace a je tedy vhodné opět využít zabezpečeného kanálu. JIRA navíc umožňuje zjistit, zda je klient přihlášen a odhlásit klienta při použíti GET a DELETE metod.

```
curl -D- -X POST -H "Content-Type: application/json" -d
    '{"username":"user", "password":"pass"}' "https://
    helpdesk.cvut.cz/jira/rest/auth/1/session"
```

V odpovědi přijde mimo jiného jméno a hodnota autentizační cookie:

```
{
  "session": {
    "name": "JSESSIONID",
    "value": "12345678901234567890"
  },
  ...
}
```

Jméno cookie je vždy `JSESSIONID`. Po získání cookie můžeme začít používat REST API.

```
curl -D- -X GET -H "cookie: JSESSIONID
    =12345678901234567890" -H "Content-Type: application/
    json" "https://helpdesk.cvut.cz/jira/rest/api/latest/
    myself"
```

4.2.1.3 OAuth

JIRA podporuje protokol OAuth 1.0a (RFC 5849[4]). OAuth umožnuje přistupovat k API, kde se autorizovaná aplikace jeví jako uživatel, který je v JIRA přihlášen. Bohužel Atlassian omezil tuto metodu pouze na komunikaci mezi svými produkty, které sdílejí uživatele[9]. Pro ostatní aplikace však dovoluje vytvořit „aplikační účet“, se kterým se cizí aplikace bude autentizovat. Tento speciální účet musí mít dostatečná privilegia k operacím, které budou potřeba. OAuth v systému JIRA nahrazuje starší (velmi podobnou) metodu pojmenovanou „Trusted Applications“[9], ve které si aplikace vyměnily certifikáty a poté podepisovaly požadavky svým klíčem. OAuth 1.0a funguje podobně a nejspíš protože je normou RFC, nahradil původní metodu.

Jak by se tedy dalo postupovat při zvolení využití OAuth autentizace? V JIRA se nejprve vytvoří speciální aplikační účet pro EBIE. Následně je potřeba v JIRA vytvořit tzv. aplikační link. Toto vyžaduje několik údajů. První je „Consumer Key“, který slouží jako identifikace konzumenta (připojované aplikace). Formát klíče je libovolný a určuje si ho konzument. Může to být čitelný text nebo náhodný řetězec. Dále je potřeba od EBIE získat RSA veřejný klíč a vložit jeho obsah do JIRA. Nakonec je možné vyplnit „Consumer Callback“ který se bude volat po úspěšném povolení k přístupu.

Nyní EBIE a JIRA provedou tzv. „OAuth dance“. Nejdříve EBIE zažádá o `request token`, který si nechá uživatelem autorizovat. Po potvrzení získá EBIE verifikační kód. Ten opět využije k získání `access token` od systému JIRA. Pro využití tohoto tokenu v REST API je poté potřeba podepisovat

požadavky s tímto tokenem. GET požadavek může například vypadat následovně:

```
POST /rest/api/latest/myself HTTP/1.1
Host: localhost:8080
Authorization: OAuth realm="http://localhost:8080/",
  oauth_consumer_key="3214567keycons15",
  oauth_token="randomtoken12345",
  oauth_nonce="asdf1168er999412",
  oauth_timestamp="1459013340",
  oauth_signature_method="RSA-SHA1",
  oauth_version="1.0",
  oauth_signature="VkT2RxJKx4IHZ2kFBmYKQRvjyTs="
```

Poslední řádek je RSA-SHA1 podpis celého požadavku bez posledního řádku.

4.2.2 REST prostředky

JIRA RESP API[8] umožňuje plně přistupovat k datům autentizovaného uživatele a to včetně admin operaci. Z předchozí sekce a požadavku neopustit portál EBIE vyplývá, že je potřeba vytvořit speciální účet pro EBIE, který musí mít dostatečná práva pro následující operace. To ale znamená, že vlastníkem všech ticketů bude EBIE a ne koncový uživatel. Následuje stručný popis základních operací nad REST API poskytované systémem JIRA.

4.2.2.1 POST /rest/api/2/issue

Vytvoří ticket z JSON dat v požadavku. JIRA umožňuje vyplnit téměř všechny atributy. Toho se dá využít pro obejití problému vlastníka ticketu. Sice vlastnictví zůstane na EBIE, můžeme ale upravit atribut „reporter“. K tomu abychom atribut mohli nastavit potřebuje speciální účet oprávnění nastavovat „reporter“ atribut. Dále je nutné znát přihlašovací jméno uživatele v JIRA. To je ale jednotné ve všech systémech na ČVUT. A protože EBIE bude využívat stejné databáze uživatelů k přihlašování jako JIRA, máme k dispozici jeho uživatelské jméno. Díky tomu bude koncový uživatel upozorňován na změny v ticketu. Takto například může vypadat JSON nového ticketu:

```
{
  "fields": {
    "project": {
      "id": "10000"
    },
    "summary": "Something is wrong",
    "issuetype": {
      "id": "10000"
    },
```

```
    "assignee": {
      "name": "someone"
    },
    "reporter": {
      "name": "cernycyr"
    },
    "priority": {
      "id": "20000"
    },
    "labels": [
      "bugfix"
     ],
    "versions": [
      {
        "id": "10000"
      }
    ],
    "environment": "environment",
    "description": "description",
    "duedate": "2016-03-11"
  }
}
```

V odpovědi přijde ID ticketu a link na něj v REST API.

4.2.2.2 GET /rest/api/2/issue/{issueIdOrKey}

Vráti detaily o požadovaném ticketu. V požadavku je možné parametrizovat zobrazené atributy a tím snížit množství přenášených dat.

4.2.2.3 PUT /rest/api/2/issue/{issueIdOrKey}

Upraví detaily ticketu. Požadavek ve formátu JSON je velmi podobný jako při vytváření ticketu.

4.2.2.4 POST /rest/api/2/issue/{issueIdOrKey}/transitions

Změní stav ticketu přes zadaný přechod. Při změně stavu je možné současně měnit atributy a proto se ve své podstatě jedná o glorifikovanou editaci. Data jsou opět velmi podobná jako při vytvoření a editaci s rozdílem povinného zadaní identifikace přechodu.

4.2.2.5 POST/GET /rest/api/2/search

Vyhledávání ticketů pomocí JQL[10] (JIRA Query Language). Je možné pro dotazování používat jak GET, tak POST metody. Rozdíl je pouze v umístění

parametrů. Stejně jako při zobrazení detailů jediného ticketu je možné určit, které atributy chceme vrátit. Pokud očekáváme příliš mnoho výsledků, můžeme nastavit parametry „startAt“ a „maxResults“ pro postupný průchod nalezených ticketů. Takto může vypadat JSON dotazu na tickety v projektu Datové Čistoty které nahlásil uživatel „cernycyr“.

```
{
  "jql":"project=FIT_DATCIS and reporter=cernycyr",
  "startAt": 0,
  "maxResults": 15,
  "fields": [
    "summary"
  ]
}
```

4.2.2.6 JSON ticketu

Když JIRA vrací ticket bez omezení, vrátí velké množství informací, které nejsou většinou v daný moment potřebné. Pokud však zažádáme pouze o podstatné informace, můžeme mnohem kratší a přehlednější odpověď. To se může hodit pro vrácení seznamu ticketů zadaných koncovým uživatelem. Následuje ukázka možné odpovědi když chceme zobrazit pouze „summary“.

```
{
  "expand": "editmeta,renderedFields,transitions,
     changelog,operations",
  "id": "12345",
  "self": "https://localhost:8080/rest/api/latest/issue
     /12345",
  "key": "TEST-42",
  "fields": {
    "summary": "Byl jsem tady. Manik"
  }
}
```

4.3 Vytvoření ticketu přes e-mail

Je vhodné se zmínit i o možnosti zakládat tickety posláním e-mailu na určenou adresu. JIRA podporuje automatické vytváření ticketů z přijatých zpráv[11]. Na portálu by tedy stačilo umístit e-mail určený k řešení požadavků od uživatelů EBIE. Jedná se tak o snadné řešení. Tato metoda však nesplňuje požadavky neopustit portál k vytvoření žádosti a zjištění stavu vytvořených žádostí přímo v portálu. Jedná se tak o nouzové řešení, pokud přistup k JIRA REST API selže a nebude ho možné použít.

4.4 Návrh řešení

Ke komunikaci se systémem JIRA bylo zvoleno rozebírané REST API, protože starší protokoly SOAP a XML-RPC již nejsou podporovány firmou Atlassian [12]. K systému JIRA se bude EBIE přihlašovat speciálně vytvořeným aplikačním účtem. Současně bude potřeba vytvořit projekt v systému JIRA. Aplikační účet EBIE musí mít kontrolu nad novým projektem. Podstatné oprávnění je možnost vytvořit ticket v projektu s upraveným zadavatelem. Autentizace bude probíhat přes HTTP Basic. OAuth protokol by byl vhodnější variantou, ale z důvodu technických komplikací jej není možné momentálně použít.

Pro volání JIRA REST API byl zvolen Ruby gem `Jira-ruby`[13], který vytváří rozhraní pro komunikaci. Zvládá jak HTTP Basic autentizaci tak i OAuth pro budoucí implementaci. Stačí jej tedy integrovat do projektu EBIE a nakonfigurovat.

4.5 Implementace

Gem `Jira-Ruby` obaluje funkcionalitu HTTP klienta a vytváří rozhraní pro volání JIRA REST API. Pro přidání `Jira-Ruby` do projektu stačí do `Gemfile` projektu doplnit řádek

```
gem 'jira-ruby', :require => 'jira'
```

Vytvoření nového JIRA klienta vypadá následovně:

```
def get_jira_client
 options = {
   :username => "username",
   :password => "passpass",
   :site     => 'http://localhost:8080',
   :context_path => '',
   :auth_type => :basic,
   :read_timeout => 120,
   :use_ssl => false
 }
 @jira_client ||= JIRA::Client.new(options)
end
```

`Get_tickets` nalezne všechny tickety v projektu „TST“ které nahlásil přihlášený uživatel.

```
def get_tickets
  username = current_user.uid
  @issues = @jira_client.Issue.jql("PROJECT = \"TST\"
      and reporter = \"#{username}\"")
end
```

`Create_ticket` vytvoří nový ticket. V požadavku se nastaví `reporter` na přihlášeného uživatele. Atribut `issuetype` je identifikátor typu ticketu. V této ukázce se jedná o „bug“.

```
def create_ticket
  text = params['text']
  username = current_user.uid
  issue = @jira_client.Issue.build
  issue.save({"fields"=>{
    "summary"=>"#{text}",
    "project"=>{"key"=>"TST"},
    "issuetype"=>{"id"=>"10004"},
    "reporter"=>{"name" => "#{username}"}
  }})
end
```

`Close_ticket` pouze uzavře ticket přes určený přechod. V ukázce přechod 31 přechází z „Open“ do „Closed“.

```
def close_ticket
  id = params[:id]

  issue = @jira_client.Issue.find(id)
  transition = issue.transitions.build
  transition.save("transition" => {"id" => 31})
end
```

Kapitola 5

Integrace s ČVUT IdM

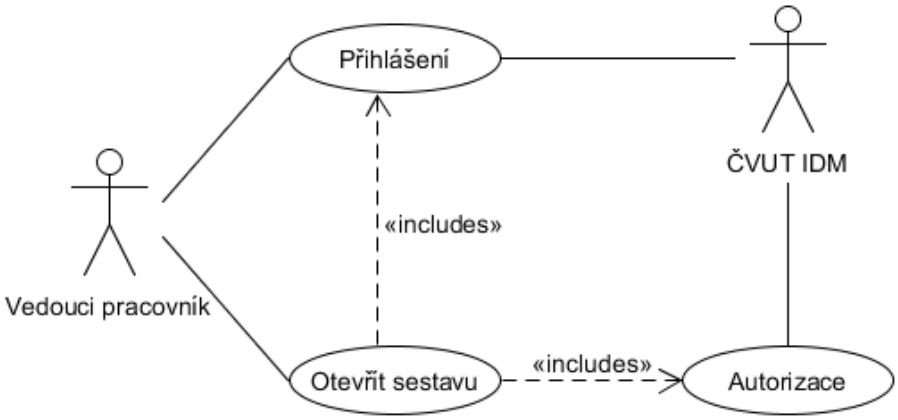
Velké organizace mohou provozovat více autonomních systémů ke kterým zaměstnanci přistupují. Pokud každý z těchto systému udržuje vlastní databázi uživatelů, je zde velká duplikace dat. Také vzniká problém se synchronizací. Pokud by byl potřeba vytvořit nový účet, je nutné jej vytvořit ve všech systémech. Podobné je to při mazání již neaktivních účtů. Neboť systémy mají nezávislé databáze účtů, jejich uživatelé se musí přihlašovat do každého systému zvlášť. Proto velké organizace nasazují tzv. Identity manažery (IdM). IdM sada nástrojů a pravidel pro správu uživatelů, jejich identit a rolí[14]. Spolupracující systémy tak mohou sdílet uživatelskou základnu. Při využiti SSO (Single sign on) pak stačí, aby se uživatel přihlásil pouze jednou a bude mít přístup do všech připojených systémů.

V této kapitole bude rozebrán druhý ze dvou hlavních úkolů práce. Postupně rozebere analýzu zadání a rozebere možnosti pro řešení. Následně navrhne a zvolí přistup a demonstruje implementaci.

5.1 Analýza požadavků

EBIE bude určena pro vedoucí pracovníky školy, fakult, kateder a ústavů. Poskytne jim informace pro rozhodování a řízení při konání jejich funkce. Protože EBIE bude fungovat na úrovni ČVUT, může od existujícího IdM získat informace o uživatelích a jejich rolích. Portál pouze poskytuje data a nenabízí žádnou možnost je upravovat. Není tedy třeba řešit, zda uživatel smí data měnit. Informace dostupné v portálu EBIE nejsou určeny pro všechny a je potřeba rozpoznat oprávněné uživatele ke čtení dostupných dat.

Cílem této části je zanalyzovat již existující role na ČVUT a na jejich základě vytvořit interní role pro využíti v portálu EBIE. Na základě vytvořených interních rolí následně navrhnout způsob autorizace přístupu k datům v portálu EBIE. Požadované případy užití jsou znázorněny v diagramu 5.1.



Obrázek 5.1: Případy užití IdM

5.2 Analýza rolí

ČVUT IdM v současné době obsahuje přes 20.000 rolí. Tyto role jsou rozděleny na byznys a technické role. Byznys role určují pozici v organizační struktuře ČVUT (např. děkan, vedoucí, zaměstnanec). Technické role jsou používány v připojených subsystémech (např. KOS).[14]

Protože nás zajímá pozice osoby v rámci ČVUT, zajímají nás primárně byznys role. Byznys role nám postačují, protože pomocí nich dokážeme rozpoznat vedoucí pracovníky. Pokud by data měla být přístupná například garantům předmětů, bude potřeba využít technické role z vhodného subsystému. V případě nutnosti je možné využít tzv. předmětové role[15].

5.2.1 Struktura role

Role v ČVUT IdM jsou textové řetězce s následujícím formátem

`{Typ role}-{Organizační jednotka}-{Název role}`

Typ role Byznys (`B`), nebo technická (`T`) role. U technických rolí se ještě upřesní k jakému subsystému patří (`T-KOS`).

Organizační jednotka Pěticiferný kód určující organizační jednotku a pracoviště. Speciální kód je `00000`, který označuje celé ČVUT. Pro ostatní značení lze rozdělit číslo na 2 části. První 2 cifry určují hlavní organizační jednotku. Mezi ně patří:

- Fakulty (11-18)
- Ústavy (31-37)

- Rektorát ČVUT (51)
- Výpočetní a informační centrum (81)
- Nakladatelství ČVUT (82)
- Ústřední knihovna (83)
- Inovacentrum (84)
- Správa účelových zařízení (91)

Zbytek kódu určuje konkrétní pracoviště. Pokud následují pouze nuly (např. `18000`), jedná se o roli vztaženou na organizační jednotku bez závislosti na pracoviště. V případě fakult kódy 100-199 zpravidla značí katedry. V těchto 3 cifrách mají fakulty volnost a i přestože je zde náznak konvence, ne všechny ji plně dodržují. Jako příklad může být (ne)existence děkanátu.[16]

Název role Jméno role. Možné příklady: `VEDOUCI`, `STUDENT`.

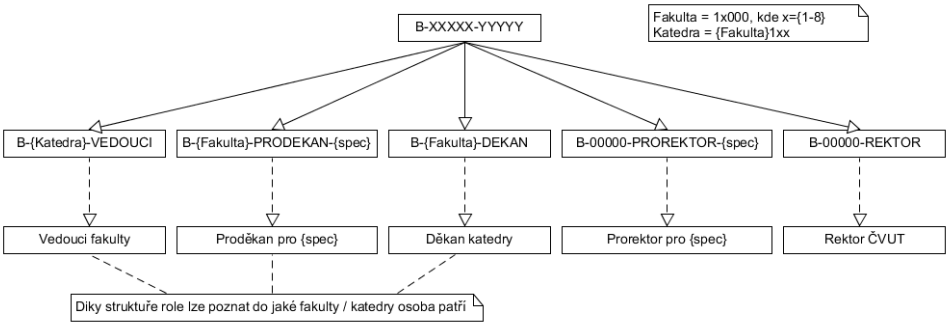
Pro lepší ilustraci si ukážeme reálnou roli `B-18102-ZAMESTNANEC-HPP-AKADEMICKY`. Z této byznys role můžeme vyčíst, že se jedná o zaměstnance s hlavním pracovním poměrem na katedře softwarového inženýrství na FIT. Pokud bychom neměli k dispozici seznam kateder na fakultách, víme alespoň ze daná osoba pracuje na nějaké katedře FIT, protože označení pracoviště začíná na `181`. Struktura rolí některých vedoucích pracovníků je znázorněna obrázkem 5.2.

Role v ČVUT IdM jsou odvozované. Co to znamená? Vezměme si studenta na FIT (`B-18000-STUDENT`). Tento student automaticky dostane od IdM roli studenta ČVUT (`B-00000-STUDENT`). Tyto vlastnosti nám umožňují se ptát jen na ty úrovně, které nás zajímají. Pokud nás zajímá pouze zda je daná osoba zaměstnancem a nepotřebujeme vědět kde, můžeme se ptát na přítomnost role `B-00000-ZAMESTNANEC`. Pokud bychom chtěli zaměstnance konkrétní fakulty, použijeme např kód 18000 (FIT). Odvozené role nám tak usnadní hledání osob s konkrétní rolí.

5.2.2 Předmětové role

Usermap API k existujícím byznys rolím přidává tzv. předmětové role. Ty slouží k identifikaci vztahu mezi osobu a předmětem. Předmětové role mají podobnou strukturu jako byznys role. Přidává vlastní typ role označený `P` a místo čísla v organizační jednotce je využit kód předmětu, který je na celém ČVUT unikátní. Tyto informace Usermap API čerpá z technických KOS rolí.[15]

Pro ilustraci si ukážeme roli `P-BI-DBS-UCITEL-CVICICI`. Na první pohled se jedná o předmětovou roli. Jméno role (`UCITEL-CVICICI`) jasně vyjadřuje, že osoba je cvičící v předmětu `BI-DBS`.



Obrázek 5.2: Ilustrace struktury rolí

5.2.3 Vedoucí pracovišť

Vedoucí nemusí být pouze vedoucí katedry. Tato role označuje vedoucího jakéhokoliv oddělení. Jakmile má nějaké pracoviště vedoucího pracovníka, bude mít roli vedoucího. Obecně můžeme hledat vedoucí oddělení regulárním výrazem `^B-\d{5}-VEDOUCI$`. Pokud se ale omezíme pouze na akademické pracovníky a budeme chtít nalézt vedoucí kateder, bude potřeba následující výraz `^B-1[1-8]1\d{2}-VEDOUCI$`. Co přesně hledáme? Osobu s rolí `VEDOUCI` na jakékoliv fakultě a katedře.

Jak je to v případě děkana jako „vedoucího fakulty“? Můžeme hledat vedoucího fakulty (`^B-1[1-8]000-VEDOUCI$`)? Např. na FIT je vedoucím děkanátu (přesněji pracoviště `18911` - děkan, sekretariát a podatelna). Díky tomu má děkan FIT roli `B-18911-VEDOUCI`. Bohužel vytvoření děkanátu, nebo podobného pracoviště není pravidlem. Fakulta strojní nemá vytvořené pracoviště děkanátu a proto jejich děkan nebude mít roli `VEDOUCI`. Sice má roli vedoucího, ale na úrovni vedoucího katedry, což taktéž nemusí být pravidlem. Využití role `VEDOUCI` není tedy univerzální pokud pouze měníme organizační jednotku.

ČVUT IdM k identifikaci výše postavených pracovníků používá jména jejich funkcí. V listu agregovaných rolí[17] nalezneme `DEKAN` i `REKTOR`. V případě proděkanů a prorektorů pro určené oblasti existují specifické role pro danou oblast, kterou mají na starosti (např. `PRODEKAN-VYUKA`, `PROREKTOR-ROZVOJ`).

Na ČVUT nejsou pouze fakulty. Součástí ČVUT jsou např. ústavy, správa účelových zařízení, výpočetní a informační centrum, ústřední knihovna nebo rektorát. Tyto součásti mají také své „vedoucí“. „Vedoucí“ ústavu ale není děkan, jak je tomu u fakulty, nýbrž ředitel. Proto v ČVUT IdM existuje role `REDITEL`. Vedoucí pozice opět mají roli `VEDOUCI` a lze je hledat obdobně jako vedoucí kateder.

5.3 Použití ČVUT IdM rolí v EBIE

Portál EBIE poskytuje data pro vedoucí pracovníky ČVUT. Tyto pracovníky lze rozpoznat podle přidělených rolí jak bylo rozvedeno v předchozí kapitole. Role osob využijeme k autorizaci přístupu k datům v EBIE. První úroveň autorizace je přistup do EBIE. Do EBIE chceme pustit osoby, které mají alespoň jednu z kvalifikovaných rolí. Kvalifikované role se dají shrnout následovně:

- Rektor
- Kvestor
- Prorektor
- Dekan
- Tajemník
- Proděkan
- Vedoucí (katedry)
- Ředitel

Pro tyto role je portál určen. Není ale žádoucí všem vpuštěným osobám zobrazit všechna data. Proto je potřeba jemnější úroveň přístupu k datům.

Dostupná data se mohou týkat například jedné fakulty nebo ústavu. Vedoucí katedry by měl mít přístup k sestavám o jeho fakultě, ale už ne k sestávám určených pro Ústřední knihovnu. Je tedy potřeba identifikovat pracoviště vedoucího pracovníka. K tomu slouží pětimístný kód každé byznys role v ČVUT IdM. Díky struktuře kódu pracoviště jsme schopni určit jak celou fakultu, tak jednotlivá oddělení v rámci fakulty. Pokud k sestavě budeme chtít pustit pouze vedoucího katedry softwarového inženýrství, tak tam pustíme pouze osobu s rolí `B-18102-VEDOUCI`. Pokud však bude sestava určena pro kohokoliv (kvalifikovaného) z FIT, musí osoba mít přidělenou roli vyhovující regulárnímu výrazu `B-18\d{3}-(VEDOUCI|DEKAN|PRODEKAN.*)`.

5.3.1 Výjimky a vnitřní role

Mohou nastat případy, kdy bude potřeba zajistit přistup lidem, kteří nejsou vedoucími pracovníky a tudíž nemají kvalifikovanou roli. Mezi takové lidi mohou patřit vývojáři, sekretářky nebo i studenti, kteří píší práci, která vyžaduje data dostupná v EBIE.

Tento problém řeší seznam pro zvláštní přistup - whitelist. Whitelist je seznam lidí, kteří mohou být vpuštěni do portálu bez nutnosti mít jednu ze zmiňovaných rolí. V seznamu je uvedeno uživatelské jméno a přidělaná role. Přidělená role se může shodovat s již existující ČVUT IdM rolí. V tom případě

bude uživatel mít přistup tam, kam ostatní vlastníci dané role. Tato metoda může být použita například pro vpuštění sekretářek vedoucích pracovníků.

V případě autorizování studentů pro přístup k jedné sestavě, kterou potřebují ke své práci, je role vedoucího příliš široká. Student by mohl prohlížet všechny sestavy určené pro vedoucího. Pro tyto případy budou sloužit vnitřní role. Vnitřní role mohou být dočasné i dlouhodobé. Pro konkrétní osoby se tak vytvoří role, která je oprávní přistupovat pouze k některým sestavám. Student tak dostane dočasně roli `EBIE-Student`, která umožní přistup k určené sestavě. Formát vnitřních rolí je libovolný. Jméno role by však mělo vystihovat koho nebo co reprezentuje.

Následující příklady ilustrují vnitřní role pro vpuštění nevedoucích osob do portálu

- EBIE-Developer
- EBIE-Guest
- EBIE-Demo
- EBIE-Student-cernycyr

Role `EBIE-Guest` a `EBIE-Demo` jsou určeny pro ukázku podoby a funkcionality EBIE nevedoucím pracovníkům, případně i lidem mimo ČVUT. K takové ukázce by se museli najít nebo vytvořit ukázkové sestavy určené k demonstraci. `EBIE-Student-cernycyr` je vzorová role pro určitého studenta, který dostane přístup pouze k některým sestavám.

5.4 Získání informací o uživateli

Nyní vyvstává otázka, jak se dostat k informacím o uživatelích a jejich rolích. Pochopitelně tyto informace nejsou veřejně dostupné. Hodilo by se, kdybychom měli přístup k systému USERMAP, ve kterém jsou uživatelé a jejich role evidovány. K tomu na ČVUT FIT vzniklo „Usermap API“[15].

5.4.1 Usermap API

Usermap API agreguje údaje o identitách lidí z ČVUT. Přes REST poskytuje základní informace jako jméno, uživatelské jméno a byznys role. Usermap API je zabezpečená protokolem OAuth 2.0 a využívá fakultní autorizační server. V dokumentaci[18] nalezneme, jak se dostaneme k potřebným informacím. Pro naše potřeby se hodí tyto 2 endpointy.

5.4.1.1 GET /people/{username}

Tento endpoint vrátí informace o uživateli v JSON formátu. Příklad je převzat z dokumentace[18]

```
{
  "username": "flynn",
  "personalNumber": 123456,
  "firstName": "Kevin",
  "lastName": "Flynn",
  "fullName": "Ing. Kevin Flynn, CSc.",
  "emails": [
    "flynn@fit.cvut.cz",
    "kevin.flynn@fit.cvut.cz"
  ],
  "preferredEmail": "kevin.flynn@fit.cvut.cz",
  "departments": [
    {
      "code": 18927,
      "nameCs": "Oddělení pro rozvoj",
      "nameEn": "Office of Development"
    }
  ],
  "rooms": [
    "TH:A-1324"
  ],
  "phones": [
    "420224355555"
  ],
  "roles": [
    "B-00000-ZAMESTNANEC-HPP",
    "B-00000-ZAMESTNANEC-NEAKADEMICKY",
    "B-00000-ZAMESTNANEC",
    "B-18000-ZAMESTNANEC-HPP",
    "B-18000-ZAMESTNANEC-NEAKADEMICKY",
    "B-18000-ZAMESTNANEC",
    "B-18927-ZAMESTNANEC-HPP-NEAKADEMICKY"
  ]
}
```


Z příkladu lze vidět, že vrácené údaje obsahují pro nás důležité role.

5.4.1.2 HEAD /people/{username}/roles

Pokud nás bude zajímat, zda konkrétní osoba má nějakou roli, je možné využít tento endpoint. V parametrech dotazu se uvedou požadované role a server vrátí status 200, pokud osoba splňuje podmínku. V opačném případě bude vrácen status 404.

```
curl -D- -X HEAD -H "Authorization: Bearer 0c45ba50
   -46c7-452c-99b1-a8443c9bd757" "https://kosapi.fit.
   cvut.cz/usermap/v1/people/cernycyr/roles?roles=B
   -18000-STUDENT
```

V tomto příkladě se ptáme serveru, zda uživatel cernycyr má roli `STUDENT` na FIT. V odpovědi se vrátí pouze hlavička s návratovým kódem.

5.4.2 Autentizace uživatele

Jak již bylo zmíněno, k přístupu k Usermap API je potřeba se autentizovat. Autentizace probíhá přes OAuth 2.0 autorizační server (OAAS) Zuul[5], který vzniká na ČVUT FIT. V moment, kdy se do EBIE uživatel přihlásí, budeme vědět jeho uživatelské jméno, které můžeme využít k dotazu na Usermap API k získání jeho rolí. Dostaneme tak všechny informace potřebné k řízení přístupu.

Ve stručnosti popíšeme, jak probíhá autentizace a následná autorizace uživatele EBIE.

1. Uživatel navštíví portál EBIE a bude vyzván k přihlášení.

2. Uživatel je přesměrován na OAAS, kde zadá své přihlašovací údaje. To mimo jiné znamená, že portál EBIE nemusí udržovat databázi uživatelů a nemá přístup k heslům uživatelů.

3. Zuul OAAS se zeptá uživatele, zda autorizuje EBIE k přístupu k Usermap informacím v jeho zastoupení. Uživatel má možnost tuto žádost zamítnout, nebudě však do EBIE vpuštěn.

4. Pokud je žádost autorizována, je uživatel přesměrován zpátky na portál. V parametrech přesměrování EBIE dostane `authorization code`.

5. Nyní EBIE odešle žádost o `access token` a `refresh token` na OAAS Zuul. Ten, pokud je poslaný požadavek v pořádku, vrátí požadované tokeny.

6. Uživatel je v tuto chvíli autentizován, protože víme, že patří do akademické obce ČVUT a úspěšně se přihlásil. Nyní je potřeba ověřit, zda může být vpuštěn do portálu

7. EBIE odešle požadavek na podrobnosti o uživateli na Usermap API. K autorizaci použije `access token`, který byl získán v bodě 5.

8. Na základě vrácených rolí je uživatel úspěšně vpuštěn do EBIE, nebo odmítnut z důvodu nedostatečného oprávnění.

5.4.3 Shibboleth a SSO

Shibboleth[19] je open-source systém pro ověřování uživatelů[20]. Většina ČVUT systému využívá k autentizaci právě nasazený Shibboleth. Nabízí možnost se jednou přihlásit a přistupovat k dalším systémům na ČVUT bez nutnosti se znova přihlašovat (single sign-on). Shibboleth se skládá z poskytovatele identit (např. LDAP) a poskytovatele služeb (např. web server). K integraci Shibboleth do EBIE by bylo potřeba na server EBIE doinstalovat přídavný software pro zprostředkování komunikace mezi EBIE a poskytovatelem identit.

5.5 Návrh řešení

EBIE je určena pro vedoucí pracovníky školy. V analýze bylo rozebráno, které role reprezentují tyto osoby. Protože mají ČVUT IdM role jasně určenou strukturu a nesou informaci o nositeli (vztah k pracovišti), můžeme přejmout vybrané role z ČVUT IDM. S pomocí regulárních výrazů je možné filtrovat pracovníky na úrovni pracovišť i vedoucích pozic. Pro výjimky bude vytvořen tzv. whitelist. Whitelist bude obsahovat 2 informace. Uživatelské jméno a seznam přidělených rolí. Tyto role mohou být stejné jako převzaté role. Vedle převzatých rolí budou definovatelné vnitřní role. Tyto role jsou pouze v rámci EBIE a nemají žádnou souvislost s rolemi ČVUT IdM. Tyto role mohou být použity k autorizaci přístupu k omezenému množství dat. Návrh na formát vnitřní role je `EBIE-{jméno_role}`. Tyto role budou následně přidány do seznamu s převzatými rolemi aby sloužili k řízení přístupu. V jednotlivých sestavách pak bude definováno, které role k nim mohou přistupovat.

Pro získání rolí přihlašovaného uživatele a následnou autorizaci přístupu je nutné nejdříve autentizovat daného uživatele. K tomu využijeme OAAS Zuul. Důvod výběru OAAS Zuul nad Shibboleth je snadné nasazení řešení a možnost jej využít nejen k autentizaci, ale i k přístupu k REST API poskytující informace i uživatelích. S ohledem na uživatele OAAS Zuul podporuje SSO Shibboleth a uživatel tak bude moci využít již existující sezení.

Pro komunikaci se Zuul OAAS bylo zvoleno několik spolupracujících gemů. EBIE již využívá gem `devise`[21] pro správu uživatelů a jejich autentizaci. K této skutečnosti byl brán zřetel a výběr tedy závisel na možnosti integrace s `devise`. K autentizaci uživatele přes OAAS Zuul byl zvolen gem `omniauth-oauth2`[22], který obaluje OAuth2 klienta pro Ruby `oauth2`[23]. `Omniauth-oauth2` je šablona pro vytvoření strategie pro další gem: `omniauth` [24]. `Omniauth` je knihovna, která umožňuje využití více autentizačních zdrojů a také dovoluje flexibilně vytvářet nové strategie pro autentizaci. `devise` a `omniauth` jsou vzájemně propojitelné, aby `devise` spravoval lokální uživatele a udržoval jejich sezení a `omniauth` autentizoval uživatele.

Pro komunikaci s Usermap API může být využit jakýkoliv HTTP klient. Protože `oauth2` gem obaluje klienta `faraday`[25], je možné využít `oauth2`.

5.6 Implementace

Cílem této části bude demonstrovat implementaci autentizace a autorizace uživatele v EBIE.

Podpora OAuth 2.0 je klíčová pro autentizaci uživatele a následné volání REST API chráněné OAAS Zuul. Použijeme `omniauth-oauth2`[22] šablonu pro vytvoření komunikace s OAAS Zuul. Pro přidání vybraných gemů do projektu stačí přidat do `Gemfile` tyto řádky.

```
gem 'omniauth', '>= 1.3.1'
gem 'omniauth-oauth2', '~> 1.3.1'
```

V konfiguraci pro `devise` nastavíme `omniauth`. V souboru `devise.rb` přidáme řádek.

```
config.omniauth :zuul, ENV["OAUTH2_APP_ID"],
 ENV["OAUTH2_APP_SECRET"]
```

Kde `OAUTH2_APP_ID` a `OAUTH2_APP_SECRET` jsou údaje získané při registraci aplikace u autorizačního serveru.

5.6.1 Vytvoření strategie

Nyní vytvoříme strategii pro komunikaci s OAAS Zuul. Protože používáme šablonu, stačí doplnit podstatné informace pro připojení k OAAS. Nejprve si vytvoříme soubor se jménem strategie a vytvoříme třídu, která dědí strategii `OAuth2`.

```
module OmniAuth
  module Strategies
    class Zuul < OmniAuth::Strategies::OAuth2
```

Nastavíme endpointy pro získání autorizačního kódu a přístupového tokenu. Tyto adresy lze získat na wiki oddělení pro rozvoj[26].

```
option :client_options, {
  :site          => "https://auth.fit.cvut.cz",
  :authorize_url => 'oauth/authorize',
  :token_url     => 'oauth/token'
}
```

OAAS Zuul vyžaduje pro získání tokenů HTTP Basic autentizaci. Protože použitá šablona implicitně nepoužívá HTTP Basic autentizaci, předefinujeme metodu `build_access_token`, která `access token` získává.

```
def build_access_token
 options.token_params.merge!(:headers =>
   {'Authorization' => basic_auth_header })
 super
```

```
end

def basic_auth_header
  "Basic " + Base64.strict_encode64(
    "#{options[:client_id]}:
      #{options[:client_secret]}")
end
```

Nyní se naše strategie dokáže připojit k OAAS Zuul a autentizovat uživatele. Nemá však příliš informaci o uživateli.

```
uid{raw_info['username']}

def raw_info
  @raw_info ||=
    access_token.get(USER_INFO_URL).parsed
end

def usermap_info
  @usermap_info ||=
    access_token.get("#{USERMAP_URL}/#{uid}").parsed
end
```

Tyto 2 metody zajistí všechny potřebné informace o uživateli. Jsou zde volány metody objektu `access_token`. Tento objekt je vytvořen použitou knihovnou a umožňuje snadno posílat požadavky obsahující získaný `access_token`. Metoda `raw_info` dostane od OAAS Zuul JSON s informacemi o uživateli, kterému token patří. Podstatná informace je jeho uživatelské jméno. Uživatelské jméno se následně použije k volání Usermap API v metodě `usermap_ info`. V kódu se používají následující konstanty.

```
USER_INFO_URL =
  'https://auth.fit.cvut.cz/oauth/userinfo'
USERMAP_URL =
  'https://kosapi.fit.cvut.cz/usermap/v1/people/'
```

K tomu aby vytvořená strategie předala informace do prostředí Omniauth, je třeba dodefinovat `info` a `extra`, které ponesou informace k dalšímu zpracování v aplikaci.

```
info do
{
  :username => raw_info['username'],
  :email => raw_info['email']
}
end
```

```
extra do
{
  'raw_info' => raw_info,
  'usermap_info' => usermap_info,
  'access_token' => access_token
}
end
```

Tímto je strategie dokončena. Nyní máme připravenou autentizaci uživatele pomoci OAuth 2.0. K vlastnímu přihlášení uživatele do portálu je však potřeba pokračovat v implementaci.

5.6.2 Callback controller

Poslední fáze provádění OmniAuth strategii se jmenuje „callback phase“. V této fázi se posílají získané údaje ze strategie ke zpracování do controlleru. První krok je nastavit controller ke zpracování callback operací v souboru `routes.rb`.

```
devise_for :users,
  :controllers =>
    { :omniauth_callbacks => "omniauth_callbacks" }
```

`Devise` vytvoří cesty pro třídu `User` a nastaví `omniauth_callbacks` jako OmniAuth callback controller. Tento nově vytvořený controller tedy bude zpracovávat informace z vytvořené strategie.

Nový controller dědí z `devise` třídy `OmniauthCallbacksController`. Nyní definujeme metodu, která se zavolá při použití strategie `zuul`. Jméno metody se musí shodovat z jménem strategie.

```
def zuul
  omniHash = request.env["omniauth.auth"]
  if allow_access? omniHash.extra.usermap_info.roles
     or whitelisted token.uid
    @user = User.find_for_zuul(omniHash)
    if @user.persisted?
      sign_in_and_redirect @user, :event =>
        :authentication
    else
      redirect_to new_user_session_path, :alert => "
         Something is wrong"
    end
  else
    redirect_to new_user_session_path, :alert => "You
       are not authorized to access EBIE"
  end
end
```

V OmniAuth autorizačním hash je momentálně přihlašovaný uživatel. Víme, že tento uživatel je autentizovaný přes Zuul OAAS a známe jeho identitu. Nejprve se otestuje, zda má povolený přístup do EBIE metodou `allow_access?`. K povolení musí mít přidělenou jednu z autorizovaných roli popsaných v analýze. V této metodě se testují role na shodu s regulárními výrazy. Zjednodušená metoda, která by pustila pouze vedoucí kateder by mohla vypadat takto:

```
def allow_access? roles
  roles.each do |role|
     return true if /^B-1[1-8]1\d{2}-VEDOUCI$/.match
        role
  end
end
```

Pokud uživatel nemá potřebnou roli, zkontroluje se, zda není uveden ve whitelistu. `Whitelisted` pouze kontroluje, zda je uživatelské jméno v seznamu. Metoda `User.find_for_zuul` je pomocná metoda, která zajistí, že uživatel bude validní `ActiveRedord` entita potřebná pro správnou funkcionalitu `devise`. Metoda `sign_in_and_redirect` řekne `devise`, že má uživatele přihlásit, tedy vytvořit sezení a spravovat jej.

5.6.3 Třída User

Třída `User` je `ActiveRedord` třídou. `ActiveRecord` je ORM(Object Relational Mapping) framework, který `devise` využívá ke správě uživatelů. K nastavení `devise` modulů a hlavně modulu `omniauthable` stačí přidat následující řádek

```
devise :omniauthable, :omniauth_providers => [:zuul]
```

Pokud jsou již nějaké moduly aktivovány, pouze přidáme potřebné moduly. Tím řekneme `devise`, že uživatelé se budou přihlašovat pouze přes `OmniAuth` a že má používat strategii `zuul`, kterou jsme vytvořili.

5.6.4 Základní autorizace

K autorizaci k jednotlivým sestavám se dají vytvořit pomocné metody, které generují regulární výrazy na základě parametrů.

```
def authorized? access_role, workplace=nil
  roles = session[:user_roles]
  if !workplace
    regString = /#{access_role}/
  else
    regString = /^B-#{workplace}-#{access_role}$/
  end

  roles.each do |role|
```

```
    return true if regString.match role
  end

  return false
end

def vedouci? fakulta=nil, katedra=nil
  if !fakulta then fakulta = "[1-8]" end
  if !katedra then katedra = "1\\d{2}" end
  return true if authorized? ("VEDOUCI",
    "1#{fakulta}#{katedra}")
  return true if dekan? fakulta
end
```

Zde `authorized?` příjme buďto celý regulární výraz, nebo jméno role a pracoviště, ze kterým vytvoří dotaz na byznys roli. Metoda `vedouci?` je pomocná metoda, která autorizuje vedoucí kateder. Současně umožní přístup i děkanovi stejné fakulty a rektorovi.

Kapitola 6

Ověření

Pro ověření navrhovaných řešení byla vytvořena v Ruby on Rails prototypová demo aplikace. Protože se jedná o demo integrace s jinými aplikacemi, je potřeba je mít k dispozici. Pro otestování systému JIRA byla využita vlastní instalace na lokálním počítači s omezenu zkušební licencí. K přístupu k ČVUT IdM je potřeba mít `client ID` a `client secret` od OAAS Zuul. Pro účely ověření byla v OAAS Zuul vytvořena aplikace `Demo App`. Aplikace demonstruje následující činnosti

- Integrace s ČVUT IdM
  - Přihlášení ČVUT účtem (spojení s OAAS Zuul)
  - Vypsání rolí přihlášeného uživatele (získání rolí z Usermap API)
  - Základní autorizace ke zdrojům (využití rolí)
- Integrace s JIRA
  - Vypsání ticketů založených přihlášeným uživatelem
  - Vytvoření ticketu s vlastním nadpisem
  - Uzavření ticketu
  - Přejití do systému JIRA k detailům o ticketu

Závěr

Cílem práce bylo navrhnout integraci nově vznikajícího portálu EBIE s ticketovacím systémem JIRA a manažerem identit na ČVUT. Po návrhu bylo úkolem vytvořit funkční prototyp řešení. Tyto cíle byly splněny následovně.

Byla navržena metoda komunikace s JIRA REST API. Tím je umožněno portálu EBIE spravovat požadavky uživatelů, aniž by uživatelé museli pracovat s webovým rozhraním ČVUT Helpesk, nebo psát email. Tento návrh byl implementován v prototypové aplikaci.

Návrh integrace s ČVUT IdM využil již existující struktury rolí. Pro autentizaci a získání rolí uživatele je použit OAAS Zuul a Usermap API. V prototypové aplikaci je možné se přihlásit ČVUT účtem. Do aplikace jsou vpuštěni pouze vedoucí pracovníci školy a předdefinovaní lidé v seznamu výjimek (whitelist). Základní autorizace je demonstrována několika odkazy, které se zobrazí pouze určeným rolím.

Implementovaná aplikace není určena k nasazení a pouze demonstruje navrhovaná řešení. Slouží k názorné ukázce implementace integrace s OAAS Zuul, Usermap API a systémem JIRA.

Literatura

[1] Ruby. [Online; accessed 29-April-2016]. Dostupné z: https://www.ruby-lang.org/en/

[2] Ruby on Rails. [Online; accessed 29-April-2016]. Dostupné z: http://rubyonrails.org/

[3] Hardt, D.: The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor, October 2012. Dostupné z: https://www.rfc-editor.org/rfc/rfc6749.txt

[4] Hammer-Lahav, E.: The OAuth 1.0 Protocol. RFC 5849, RFC Editor, April 2010. Dostupné z: https://www.rfc-editor.org/rfc/rfc5849.txt

[5] Jirůtka, J.: zuul-oaas. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/cvut/zuul-oaas

[6] Atlassian: JIRA Software. [Online; accessed 29-April-2016]. Dostupné z: https://www.atlassian.com/software/jira

[7] Atlassian: What is workflow. [Online; accessed 29-April-2016]. Dostupné z: https://confluence.atlassian.com/jira/what-is-workflow-185729618.html

[8] Atlassian: JIRA REST API Reference. [Online; accessed 29-April-2016]. Dostupné z: https://docs.atlassian.com/jira/REST/latest

[9] Atlassian: OAuth security for application links. [Online; accessed 29-April-2016]. Dostupné z: https://confluence.atlassian.com/display/APPLINKS/OAuth+security+for+application+links

[10] Atlassian: Advanced Searching. [Online; accessed 29-April-2016]. Dostupné z: https://confluence.atlassian.com/jira/advanced-searching-179442050.html

[11] Atlassian: Creating Issues and Comments from Email. [Online; accessed 29-April-2016]. Dostupné z: https://confluence.atlassian.com/jira/creating-issues-and-comments-from-email-185729464.html

[12] Atlassian: SOAP and XML-RPC API Deprecation Notice. [Online; accessed 29-April-2016]. Dostupné z: https://developer.atlassian.com/jiradev/latest-updates/soap-and-xml-rpc-api-deprecation-notice

[13] sumoheavy: jira-ruby. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/sumoheavy/jira-ruby

[14] ČVUT: IdM. [Online; accessed 29-April-2016]. Dostupné z: http://intranet.cvut.cz/informace-pro-zamestnance/is/role/idm

[15] Jirůtka, J.: Usermap API. [Online; accessed 29-April-2016]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/usermap-api

[16] ČVUT: USERMAP. [Online; accessed 29-April-2016]. Dostupné z: https://usermap.cvut.cz

[17] Jirůtka, J.: Role v IdM ČVUT. [Online; accessed 29-April-2016]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/role-v-idm-cvut

[18] Jirůtka, J.: Umapi API documentation. [Online; accessed 29-April-2016]. Dostupné z: https://kosapi.fit.cvut.cz/usermap/doc/rest-api-v1.html

[19] Jirůtka, J.: How Shibboleth Works: Basic Concepts. [Online; accessed 29-April-2016]. Dostupné z: https://shibboleth.net/about/basic.html

[20] ČVUT: Shibboleth. [Online; accessed 29-April-2016]. Dostupné z: https://www.civ.cvut.cz/info/info.php?id=207

[21] plataformatec: devise. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/plataformatec/devise

[22] intridea: omniauth-oauth2. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/intridea/omniauth-oauth2

[23] intridea: oauth2. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/intridea/oauth2

[24] intridea: omniauth. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/intridea/omniauth

[25] lostisland: faraday. [Online; accessed 29-April-2016]. Dostupné z: https://github.com/lostisland/faraday

[26] Jirůtka, J.: OAuth 2.0. [Online; accessed 29-April-2016]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/oauth2

Příloha **A**

Seznam použitých zkratek

(E)BIE (Extended) Business Inteligence Encyclopedia

API Application Programming Interface

REST Representational State Transfer

HTML HyperText Markup Language

MVC Model-View-Controller

RoR Ruby on Rails

ORM Object-relational mapping

URI Uniform Resource Identifier

OAAS OAuth 2.0 Authorization Server

JQL JIRA Query Language

JSON JavaScript Object Notation

Příloha B

Demo Aplikace

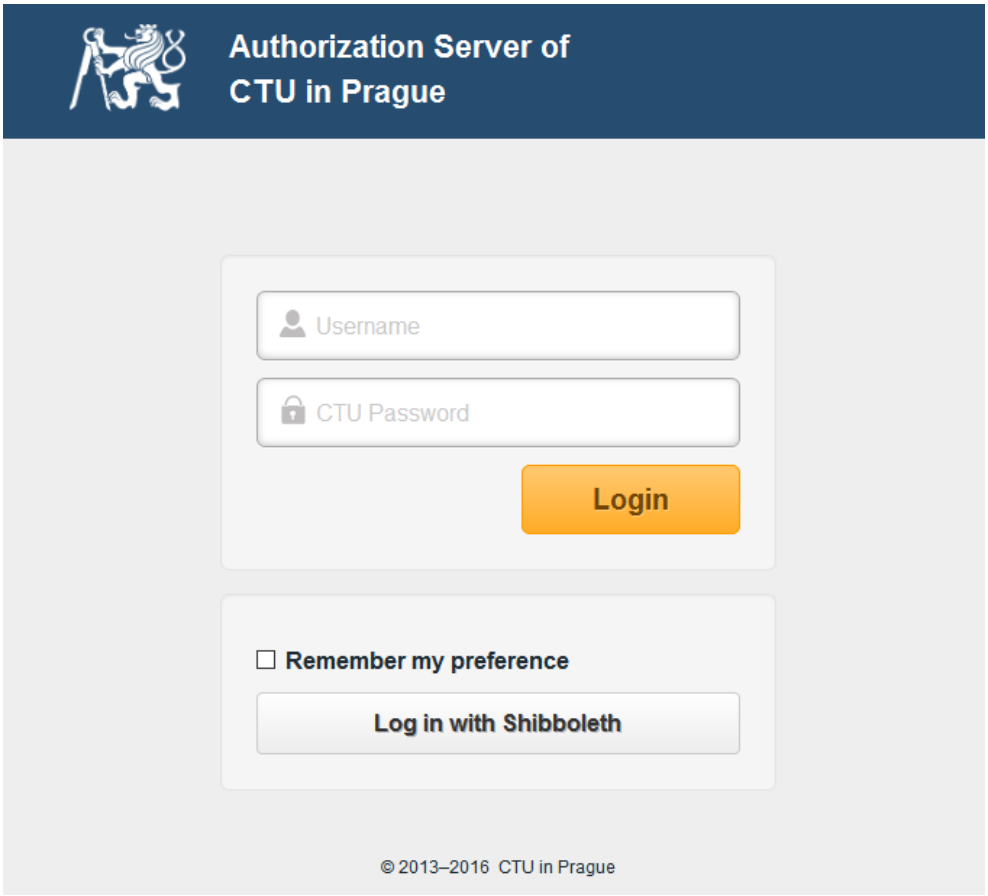
Ukázka průchodu ukázkovou aplikaci. Je zde ukázáno přihlášení uživatele, úvodní obrazovka rolí a správa ticketů. U rolí je také demonstrováno jak ovlivňují zobrazovaný obsah.

Při prvním spuštění aplikace bude uživateli nabídnuto se přihlásit (obrázek B.1). Po kliknutí na odkaz budete přesměrování na OAAS Zuul kde proběhne autentizace (ilustrováno obrázkem B.2).

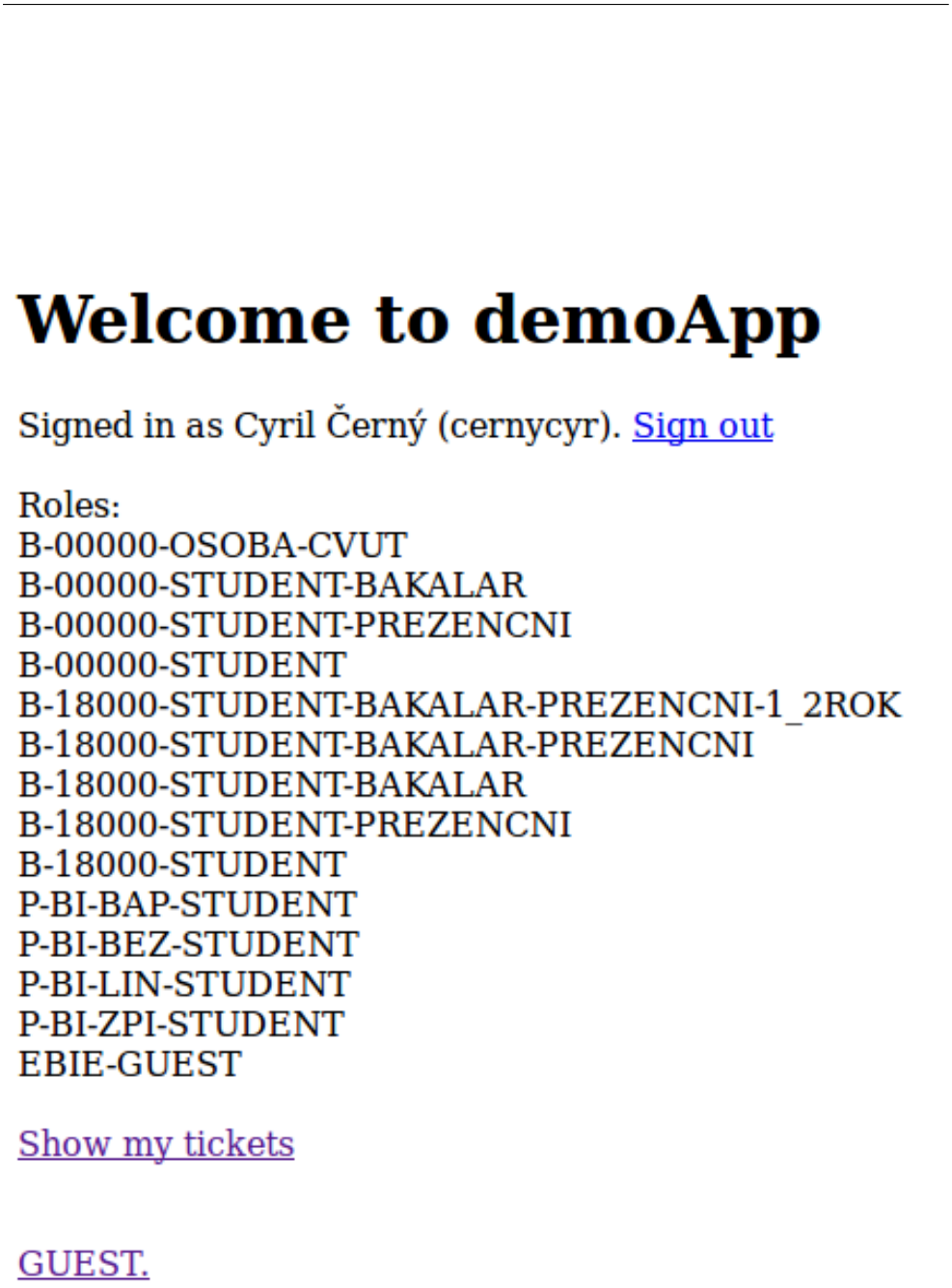
Po úspěšném přihlášení se zobrazí hlavní stránka (obrázek B.3). Vypíše přihlášeného uživatele jeho role získané z Usermap API. K těmto rolím jsou přidané i role z určených výjimek (whitelist). Pro otestování jsem se musel přihlašovat vlastním jménem a proto přidělení vedoucích rolí bylo realizováno přes whitelist. Na obrázku B.3 má uživatel pouze roli `EBIE-GUEST` a proto se mu zobrazí pouze odkaz pro návštěvníka. Obrázek B.4 ilustruje rozšíření obsahu po přidání další role. Zobrazení obsahu nemusí záviset pouze na roli. Pokud změníme roli uživatele na vedoucího katedry z FIT, otevře se mu další obsah (ilustrováno obrázkem B.5).

Na obrázku B.6 je seznam ticketů vytvořených uživatelem. Při kliknutí na „New ticket“ bude přesměrován na jednoduchou obrazovku se zadáním jména ticketu (obrázek B.7). Po odeslání se v seznamu objeví jeho nový ticket (obrázek B.8). Po kliknutí na „Go to Jira“ bude přesměrován na portál systému JIRA, kde se mu zobrazí vytvořený ticket (obrázek B.9).

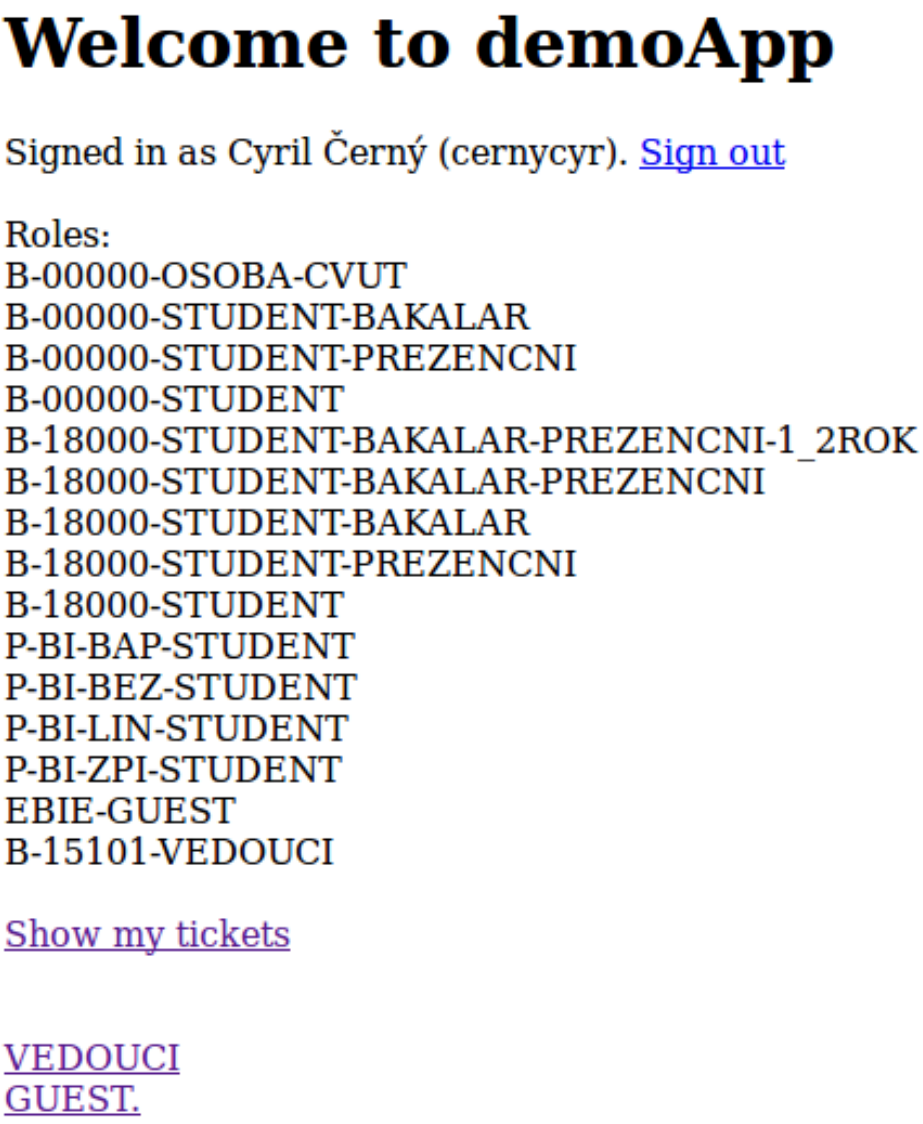
Obrázek B.1: Při prvním spuštění budete vybídnuti k přihlášení.



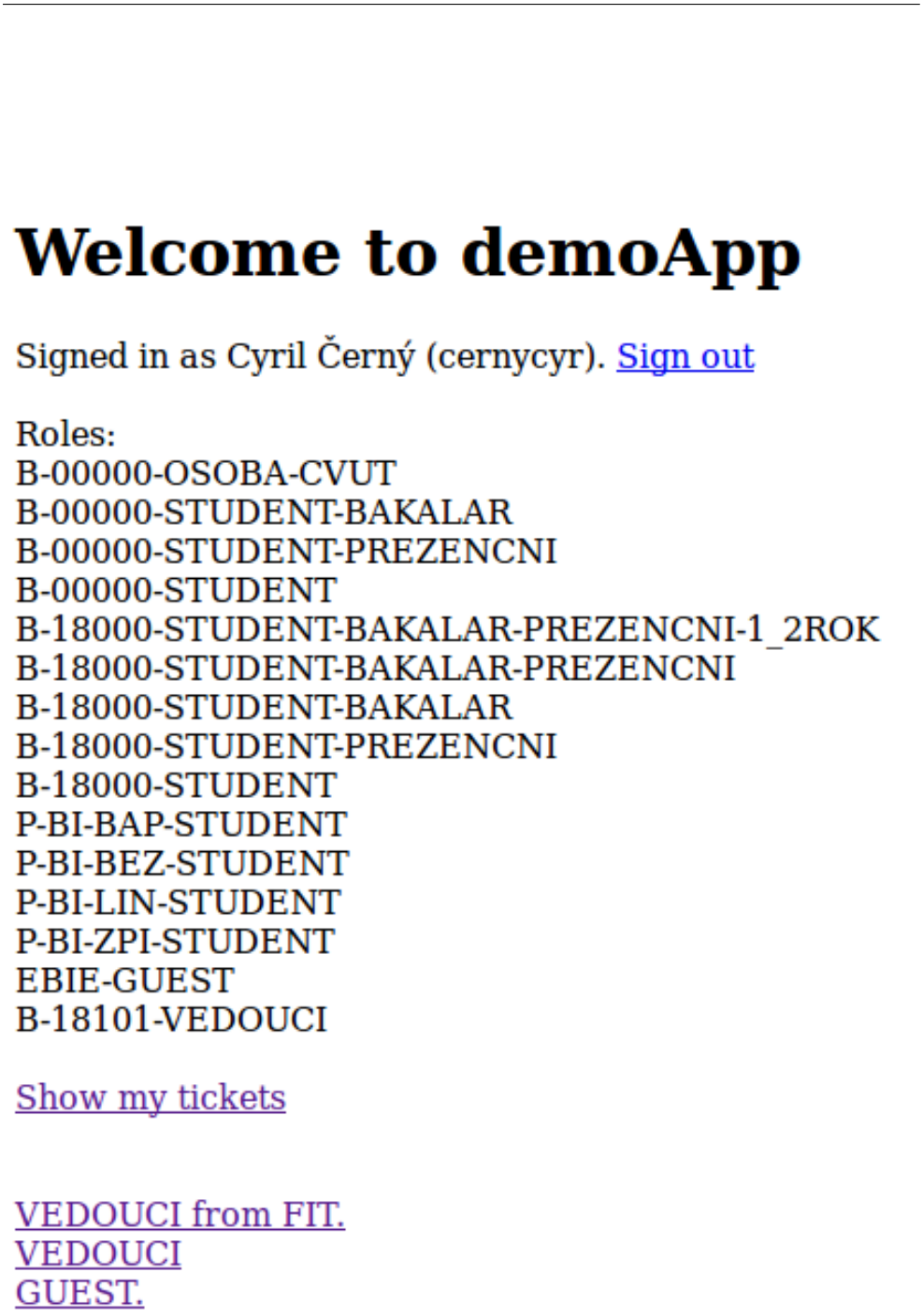
Obrázek B.2: Obrazovka přihlášení pomocí OAAS Zuul



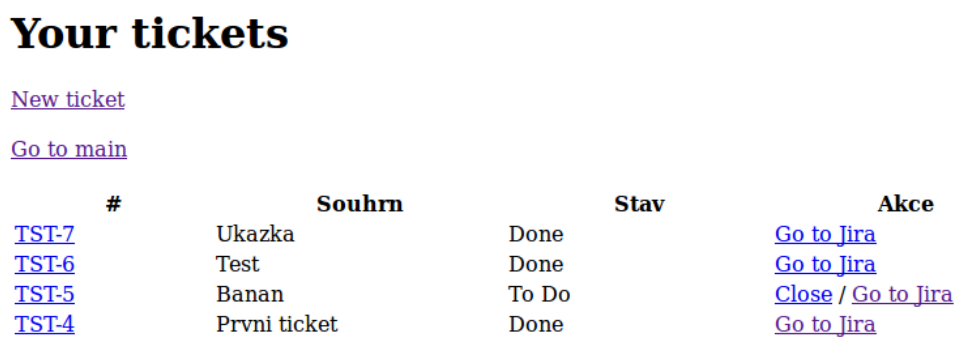
Obrázek B.3: Po přihlášení aplikace vypíše role přihlášeného uživatele



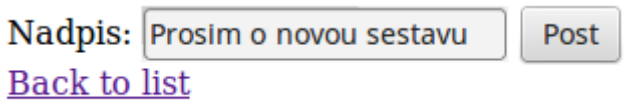
Obrázek B.4: Pokud uživateli přidáme roli vedoucího katedry, otevře se mu více možností



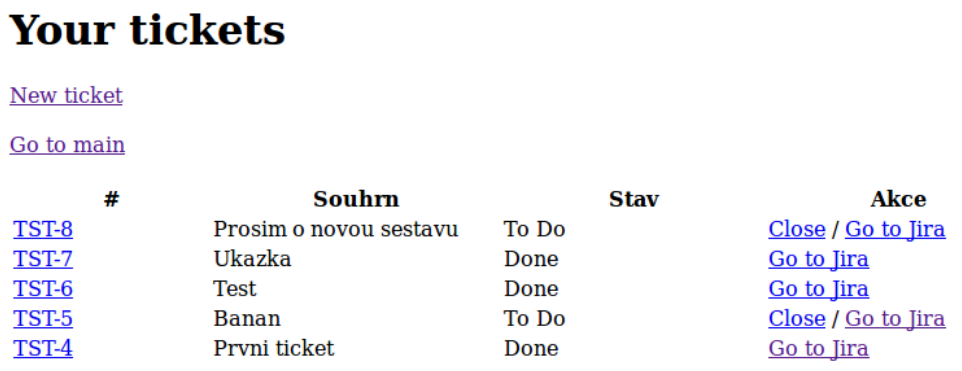
Obrázek B.5: Některý obsah závisí nejen na roli. Může záležet i na pracovišti



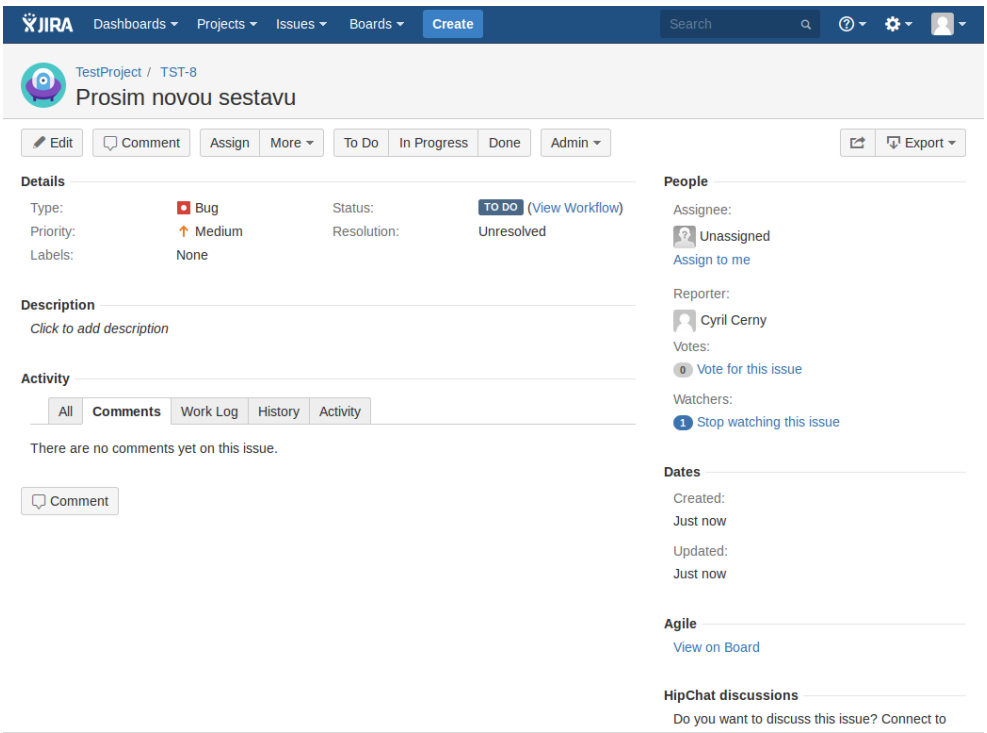
Obrázek B.6: Seznam ticketů



Obrázek B.7: Jednoduché vytvoření nového ticketu



Obrázek B.8: Nový ticket v seznamu



Obrázek B.9: Vytvořený ticket v systému JIRA

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF