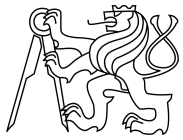
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název: Rozšíření systému Burza projektů
Student: Jakub Neburka
Vedoucí: Ing. Jiří Mlejnek
Studijní program: Informatika
Studijní obor: Softwarové inženýrství
Katedra: Katedra softwarového inženýrství
Platnost zadání: do konce letního semestru 2015/16

Pokyny pro vypracování

Analyzujte požadavky na rozšíření systému Burza projektů, který je vyvíjen na fakultě. Zaměřte se především na jeho využití pro nabídku rámcových témat závěrečných prací a na požadavky okolních systémů, které Burzu projektů využívají. Na základě analýzy požadavků navrhněte řešení, pokrývající tyto nové požadavky. Navržené řešení implementujte a otestujte. Proveďte integraci se systémem pro správu závěrečných prací, který je na fakultě využíván. Systém zdokumentujte a připravte pro produkční nasazení.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof.Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 5. ledna 2015

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

Bakalářská práce

Rozšíření systému Burza projektů

Jakub Neburka

Vedoucí práce: Ing. Jiří Mlejnek

16. května 2016

Poděkování

Děkuji vedoucímu této práce za trpělivost, užitečné rady a připomínky. Děkuji svému zaměstnavateli za to, že mi vždy vyšel vstříc ve studijních záležitostech. A především děkuji své ženě za nevyčerpatelnou psychickou podporu.

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 16. května 2016 .....

Odkaz na tuto práci

Neburka, Jakub. *Rozšíření systému Burza projektů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce je pokračováním vývoje systému Burza projektů, jehož účelem je nabízení projektů z různých zdrojů studentům. Během práce je dokončen vývoj základních funkcí systému a přidána podpora nabídky rámcových témat závěrečných prací a projektů portálu Spolupráce s průmyslem.

Práce popisuje analýzu požadavků, návrh řešení, postup implementace a testování. Výsledkem práce je serverová aplikace a webové uživatelské rozhraní Burzy projektů, několik dalších softwarových produktů a dokumentace.

Klíčová slova Burza projektů, nabídka projektů, semestrální práce, závěrečné práce, spolupráce s průmyslem, integrace systémů, Java, Spring Framework, Liferay Portal, JSF

Abstract

This thesis continues to develop the Projects Exchange system whose purpose is to offer projects from various sources to the students. Development of basic system functions is completed and the system is extended to offer the final theses topics and the projects from the Cooperation with Industry portal.

The work describes the process of requirements analysis, design, implementation and testing. The output of this work consists of the server application, the web user interface, the support tools and the documentation.

Keywords Projects exchange, projects offer, semestral projects, final theses, cooperation with industry, systems integration, Java, Spring Framework, Liferay Portal, JSF

Obsah

Seznam obrázků

Úvod

Tato práce se zabývá pokračováním vývoje systému Burza projektů. Tento systém je vyvíjen na FIT ČVUT. Jeho účelem je poskytnout studentům nabídku projektů z různých zdrojů na jednom místě. V rámci této práce bude dokončen vývoj základních funkčností systému a bude provedeno rozšíření systému o nabídku projektů vybraných okolních systémů.

Představení problému

V následujících odstavcích bude přiblížen problém, který má systém Burza projektů řešit.

Při studiu na FIT ČVUT pracují studenti na různých typech projektů, které jsou jim nabízeny a zadávány z různých zdrojů. Nabídka projektů je roztroušená a nepřehledná. V některých případech není nabízení projektů automatizováno. Následuje výčet a stručný popis vybraných zdrojů projektů:

- *Semestrální práce.* V některých předmětech jsou studentům zadávány semestrální práce. Téma semestrální práce může být zadáno učitelem, nebo si jej student vymyslí sám a nechá schválit. Většina operací při zadávání prací probíhá prostřednictvím osobní komunikace, emailu, ruční editací webových stránek předmětu. atd.
- *Softwarové projekty.* V předmětech Softwarový týmový projekt 1 a 2 řeší studenti projekty v kolektivu. Učitelé projekty vypisují a dohlížejí na to, aby každý projekt měl správný počet vhodných řešitelů. Zadávání softwarových projektů také zatím není automatizováno.
- *Průmyslové projekty.* Portál Spolupráce s průmyslem (SSP) je fakultní systém, do kterého projekty zadávají a vhodné řešitele vybírají průmysloví partneři fakulty. Jeho cílem je „umožnit efektivní a všestranně prospěšnou spolupráci průmyslových partnerů (společností) s univerzitním pracovištěm, zejména pak s jeho studenty“ [1].

- *Závěrečné práce.* Bakalářské a diplomové práce jsou zadávány a vedeny v systému Závěrečné práce (dále systém ZP), který spravuje Centrum znalostního managementu (CZM) – samostatná organizační jednotka ČVUT FEL.

Cílem vývoje Burzy projektů je sjednotit nabídku projektů z různých zdrojů, poskytnout tuto nabídku studentům přehledně na jednom místě a umožnit vypisování a nabízení nových projektů pro řešení v předmětech.

Historie vývoje Burzy projektů

V následujících odstavcích bude shrnuta historie vývoje systému Burza projektů a z ní pramenící stav systému při převzetí na začátku této práce. Ze stavu systému bude vycházet definice cílů práce.

Vývoj systému Burza projektů byl započat kolektivem studentů kombinovaného studia v předmětu Softwarový týmový projekt 1 (BI-SP1). Základní požadavky na systém vycházely z potřeb nabízení projektů tohoto předmětu. Další požadavky pocházely z portálu SSP. Výstupem práce byla původní analýza, původní návrh a první softwarový prototyp, který byl zamítnutý (společně s návrhem).

V dalším semestru práce na systému pokračovala. Bylo nutné upřesnit výstupy analýzy, přepracovat návrh a začít implementovat od začátku. Z časových důvodů byla práce na systému rozdělena na dvě části: vývoj serverové části a vývoj uživatelského rozhraní.

Ve vývoji serverové části pokračovala obměněná skupina studentů v předmětu Softwarový týmový projekt 2. Během vývoje došlo ke zjednodušení a upřesnění analýzy. Odchylky byly zdokumentovány formou poznámek. Výstupem práce byl prototyp serverové aplikace. Ta splňovala většinu požadavků předmětu BI-SP1 a umožňovala základní nabídku semestrálních prací a softwarových projektů. Aplikace nebyla řádně otestována a nebyla nasazena.

Vytvoření uživatelského rozhraní systému bylo ve druhém semestru vývoje systému svěřeno týmu vývojářů portálu SSP. Protože se do vývoje promítly odlišné představy o fungování systému a komunikace mezi týmy nebyla ideální, nebylo výsledné uživatelské rozhraní kompatibilní se serverovou aplikací. Z uživatelského rozhraní byla implementována jen část.

Motivace této práce

Po dvou semestrech vývoje, který předcházel této práci, zůstal systém Burza projektů v rozpracovaném stavu a hrozilo, že nebude dokončen. Motivací pro tuto práci byla snaha zúročit předchozí práci na systému, pokračovat v jeho vývoji, dostat systém do konzistentního zdokumentovaného stavu a přiblížit ho reálnému nasazení.

Cíle práce

Formální zadání této práce vycházelo z optimistických předpokladů o počátečním stavu systému Burza projektů. Hlavním cílem definovaným v zadání práce je rozšířit systém o nabídku projektů z okolních systémů, především systému ZP. Ve fázi zahájení práce se ukázalo, že před samotným rozšířením bude nutné dokončit vývoj původního systému. Proto bude tato práce mít následující hlavní cíle:

1. *Dokončit vývoj základních funkcí serverové části systému.* Půjde především o řešení důležitých připomínek k implementaci, vytvoření automatických testů, odstranění chyb, zlepšení podpory uživatelského rozhraní a odstranění překážek pro vývoj rozšíření.

2. *Převzít a dokončit vývoj uživatelského rozhraní.* Bude nutné nastudovat a pochopit použité technologie a postupy. Bude zajištěna kompatibilita se serverovou částí. Uživatelské rozhraní bude rozšířeno, aby pokrývalo všechny případy užití.

3. *Rozšířit systém o nabídku projektů z okolních systémů.* Bude provedena analýza možností rozšíření o nabídku projektů z okolních systémů, především systému ZP a portálu SSP. Bude navrženo a implementováno vhodné řešení.

4. *Zdokumentovat systém a připravit na předání.* Systém bude otestován, zdokumentován a připraven na předání k nasazení nebo dalšímu vývoji.

Struktura práce

Struktura tohoto textu sleduje „vodopádový“ přístup k vývoji softwaru.

První tři kapitoly se zabývají analýzou problému. V první kapitole je analyzován stav systému Burza projektů na začátku této práce. Ve druhé kapitole jsou analyzovány okolní systémy, jejichž nabídku projektů bude Burza projektů integrovat. Ve třetí kapitole jsou shrnuty závěry analýzy a definovány požadavky a případy užití.

Následující dvě kapitoly popisují návrh řešení. Zatímco čtvrtá kapitola se soustředí na návrh dokončení základních funkcí systému, pátá kapitola se zabývá návrhem rozšíření.

V šesté kapitole textu je popsán postup implementace. Sedmá kapitola se věnuje testování a dokumentaci. Výsledky práce jsou shrnuty v závěru.

Kapitola 1

Analýza Burzy projektů

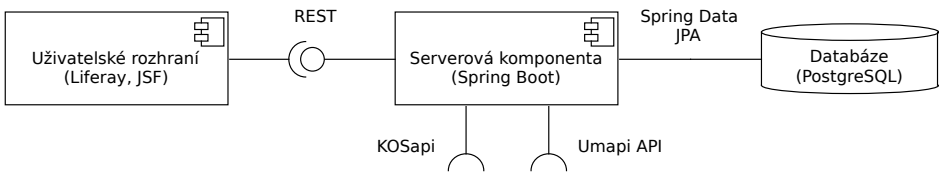
Tato práce se zabývá rozšířením systému Burza projektů. Před rozšířením systému je dobré znát jeho aktuální stav. Tato kapitola se bude zabývat stavem systému Burza projektů na začátku této práce. Budou popsány především principy fungování systému a stav implementace.

Popis stavu implementace bude základem pro požadavky na dokončení základních funkcí systému a odstranění překážek pro vývoj rozšíření. Popis fungování poslouží jako výchozí bod při návrhu rozšíření.

Burza projektů neodpovídá zcela původní analýze systému, odlišnosti nebyly řádně zdokumentovány a často se není kam odkázat. Zdrojem informací pro tuto kapitolu budou: původní analýza, poznámky vzniklé při předchozím vývoji, inspekce kódu, posbírané připomínky a v případě uživatelského rozhraní i komunikace s vývojáři.

1.1 Struktura systému

Burza projektů byla navržena jako systém s trojvrstvou (angl. 3-tier) fyzickou architekturou [2]. Každá vrstva je realizována jednou softwarovou komponentou. Komponenty jsou zachyceny v diagramu na obrázku 1.1. Následuje jejich



Obrázek 1.1: Diagram softwarových komponent Burzy projektů.

popis:

- *Databáze.* Funkcí databáze je perzistence (trvalé uložení) dat. Jako databáze Burzy projektů slouží objektově relační databázový systém PostgreSQL [3]. Kromě konfigurace nevyžaduje databáze žádný vývoj. Díky volbě technologií serverová komponenty, není nutné vytvářet SQL skripty.

- *Serverová komponenta.* Serverová komponenta tvoří aplikační vrstvu systému, tedy provádí hlavní výpočty a operace systému.

  Komponenta je vyvíjena za pomoci Spring Frameworku [4] a pro správu databáze používá jednu z jeho knihoven (Spring Data JPA). Pro účely uživatelského rozhraní (a případných dalších klientských aplikací) vystavuje serverová komponenta vlastní rozhraní s architekturou REST (Representational state transfer).

  Kromě databáze jsou dalším zdrojem dat pro Burzu projektů fakultní informační systémy KOS a Usermap, které serverová komponenta využívá pomocí jejich rozhraní KOSapi [5] a Umapi API [6].

- *Uživatelské rozhraní.* Účelem uživatelského rozhraní systému je zobrazování dat a interakce s uživatelem. Uživatelské rozhraní Burzy projektů je určeno pro nasazení na fakultní portálové řešení. Z uživatelského hlediska je to webová aplikace.

Předchozí vývoj serverové komponenty a uživatelského rozhraní Burzy projektů probíhal odděleně, což systému neprospělo. Na začátku této práce nebyly tyto komponenty kompatibilní. Nebylo proto možné použít Burzu projektů prostřednictvím uživatelského rozhraní a analyzovat ji jako celek. Obě vyvíjené komponenty proto budou v dalších částech kapitoly podrobněji analyzovány zvlášť.

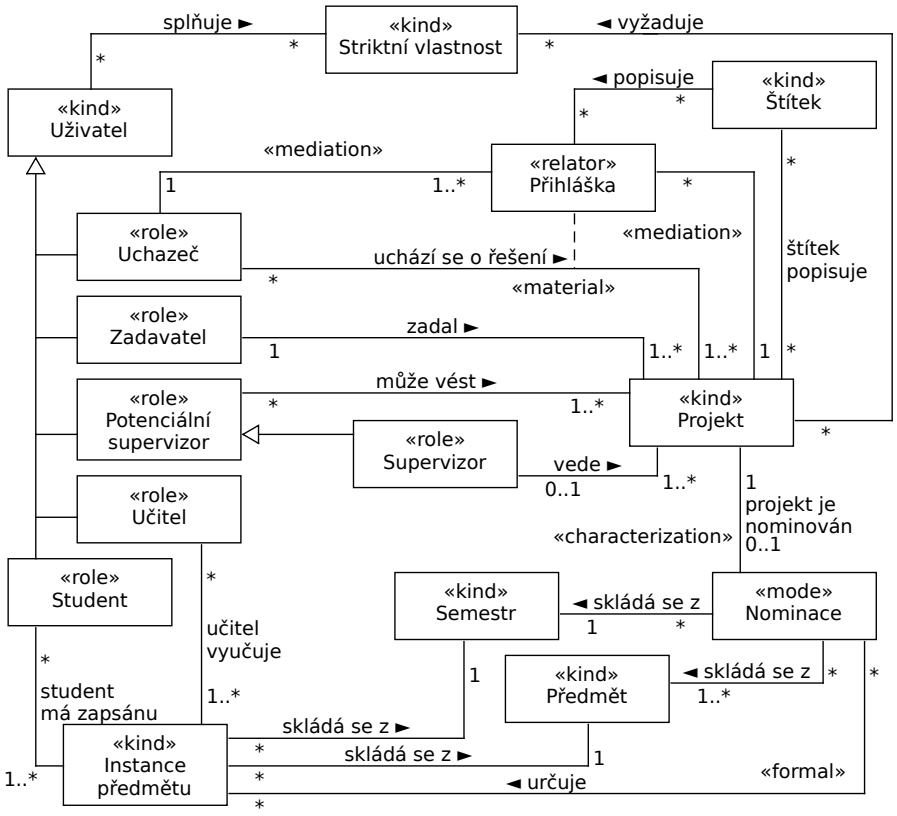
1.2 Serverová komponenta

Serverová komponenta Burzy projektů určuje hlavní principy fungování systému. Její logická architektura je třívrstvá [7]. Komponenta definuje doménové objekty, vykonává doménovou logiku (operace nad doménovými objekty) a poskytuje RESTové rozhraní.

V následujících sekcích textu bude serverová komponenta blíže popsána. Nejdříve budou popsány principy fungování, dále budou rozebrány použité technologie a na závěr bude analyzován stav implementace.

1.2.1 Popis domény

Na obrázku 1.2 je zachycen konceptuální model domény Burzy projektů. Pro modelování byl použit jazyk OntoUML [8], který je vhodný pro detailní zachycení vztahů entit a rolí [9]. Některé prvky modelu odpovídají objektům



Obrázek 1.2: Konceptuální model domény Burzy projektů (OntoUML).

z fakultních informačních systémů, které Burza projektů využívá. Jejich detailní popis je možné dohledat v dokumentacích jejich rozhraní [5, 6]. Následuje popis domény.

Základními prvky domény Burzy projektů jsou projekt a uživatel. *Projekt* je hlavní entita spravovaná Burzou projektů. Sdružuje zadání problému, který by měl být řešen, a další doplňkové informace. *Uživatel* Burzy projektů odpovídá osobě vedené v systému KOS.

Uživatelé mohou s projekty manipulovat podle toho, jakou mají vůči nim roli. *Zadavatelem* projektu se stane uživatel, který ho vytvoří. Další role uživatele vhledem k projektu jsou odvozeny od rolí uživatele (osoby) v systému KOS.

V systému KOS může být uživatel *studentem* nebo *učitelem* různých instancí předmětů. *Instance předmětu* je dvojice (semestr, předmět), která určuje, že daný předmět byl, je nebo bude v daném semestru vyučován. Projekt

může mít přiřazenu nominaci. *Nominace* se skládá z jednoho semestru a jednoho či více předmětů. Nominace projektu svými prvky určuje několik instancí předmětů, ve kterých bude projekt řešen.

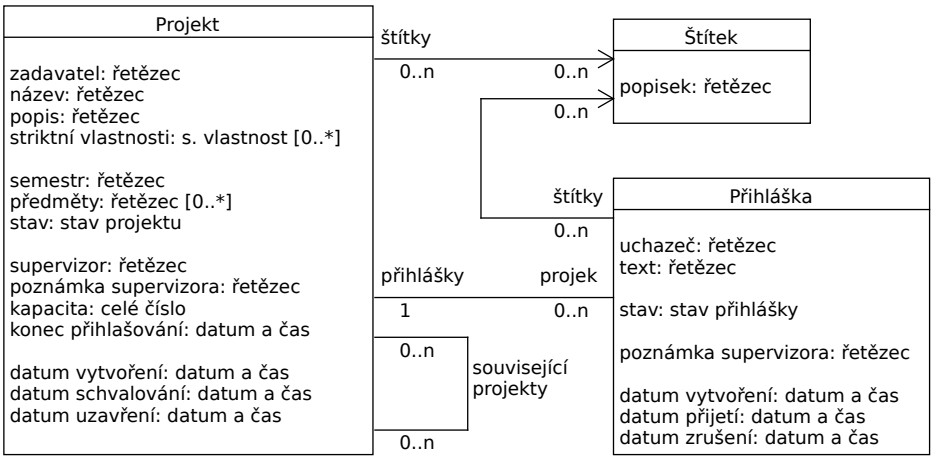
Potenciálním supervizorem projektu je uživatel, který je učitelem všech instancí předmětů, které jsou určeny nominací daného projektu. *Supervizor* projektu je jeden konkrétní potenciální supervizor, který je projektu přiřazen jako zodpovědná osoba.

Uchazečem o řešení projektu může být uživatel, který je studentem alespoň jedné instance předmětu určené nominací projektu. Uchazeči vytváří na projekty přihlášky. *Přihláška* je entita spravovaná Burzou projektů. Pomocí přihlášek jsou formovány kolektivy řešitelů projektů.

K omezení přístupu uživatelů k entitám Burzy projektů se používají *striktní vlastnosti*. Uživatel má přístup k projektům a přihláškám, které se k nim vztahují, pouze pokud splňuje všechny striktní vlastnosti, které projekty vyžadují. Jedinou v Burze projektů doposud použitou striktní vlastností je požadavek podepsané dohody o mlčenlivosti (tzv. NDA) [10]. Zda uživatel splňuje striktní vlastnost, je odvozeno z rolí uživatele v systému Usermap.

Pro kategorizaci projektů a přihlášek se používají textové *štítky*.

1.2.2 Spravované entity



Obrázek 1.3: Diagram doménových tříd Burzy projektů.

Serverová komponenta Burzy projektů spravuje doménové třídy (Entity Classes [11]), které realizují hlavní entity doménového modelu. Tyto třídy jsou zachyceny v diagramu na obrázku 1.3. Ostatní data (objekty domény) jsou získávána z fakultních informačních systémů. Spravované doménové třídy

jsou automaticky mapovány na tabulky databáze a jejich serializovaná (textová) podoba je předávána pomocí RESTového rozhraní. Následuje popis jednotlivých tříd.

Projekt

Projekt je třída, která zapouzdřuje zadání projektu a další informace potřebné pro správu projektu.

Každému projektu je při vytvoření nastaven atribut zadavatel na fakultní přihlašovací jméno (tzv. login) uživatele, který jej vytvořil. Hodnotu tohoto atributu není možné později měnit. Atributy název, popis, striktní vlastnosti, štítky a související projekty tvoří *zadání projektu*. Název a popis musí být vyplněny (obsahovat neprázdnou hodnotu). Striktní vlastnosti jsou množinou identifikátorů striktních vlastností.

Atributy semestr a předměty určují *nominaci* projektu. Projekt má *přiřazenu nominaci*, pokud jsou oba atributy vyplněny. Pro identifikaci semestrů a předmětů jsou použity jejich kódy (identifikátory v systému KOS). Stav projektu je určen pomocí atributu stav v kombinaci s některými dalšími atributy (viz dále). Atribut stav může nabývat hodnot: nový, schválený, zamítnutý a uzavřený.

Atributy supervizor, poznámka supervizora, kapacita a konec přihlašování budou nazývány *supervizorské atributy*. Poznámka supervizora slouží např. k odůvodnění zamítnutí projektu. Kapacita určuje horní mez pro počet řešitelů. Atribut konec přihlašování, pokud je nastaven, určuje datum a čas, do kdy mohou být na projekt vytvářeny přihlášky.

Dále projekt obsahuje *časové značky*, které jsou automaticky aktualizovány při operacích vytvoření, schválení či zamítnutí a uzavření projektu.

Přihláška

Třída přihlášky realizuje vztah projektu a uchazeče o jeho řešení. Přihlášce jsou při vytvoření nastaveny atributy projekt a uchazeč a jejich hodnoty se dále nemění. Text přihlášky musí být vyplněn. Štítky přihlášky musí být podmnožinou štítků projektu, ke kterému se přihláška vztahuje. Stav přihlášky je určen stejně pojmenovaným atributem, který může nabývat hodnot: nová, přijatá, odmítnutá, zrušená. Poznámka supervizora slouží k odůvodnění přijetí či odmítnutí přihlášky. Přihláška obsahuje *časové značky* pro operace vytvoření, přijetí či odmítnutí a zrušení.

Štítek

Štítky slouží pro kategorizaci projektů a přihlášek pomocí krátkých textových popisků. Mohou být použity pro filtrování kolekcí entit.

1.2.3 Stavy entit

Stavy, ve kterých se mohou nacházet entity spravované Burzou projektů, a jejich významy budou popsány v následujících sekcích textu. Diagram možných přechodů mezi stavy je v příloze A.

Stavy projektu

Stav projektu je odvozený od hodnoty atributu stav, který odpovídá základním stavům, a dalších atributů doménové třídy projektu. Možné stavy projektu jsou:

- *Nový.* Ve stavu nový se nachází nově vytvořený projekt. Nový projekt slouží pro definici zadání problému a předpokládá se, že mu bude později přiřazena nominace.
- *Nominovaný.* Pokud je novému projektu přiřazena nominace, nachází se ve stavu nominovaný a čeká na schválení. Přiřazení nominace projektu vyjadřuje navržení projektu pro řešení v instancích předmětů, které určuje daná nominace.
- *Schválený.* Projekt, který byl nominovaný, může být schválený. Schválení projektu potvrzuje, že projekt může být řešen v instancích předmětů určených jeho nominací.
- *Zamítnutý.* Nominovaný projekt může být také zamítnutý. Tento stav vyjadřuje, že projekt není vhodné řešit (pro danou nominaci).
- *Nabízený.* Projekt je nabízený, pokud byl schválený, byla mu nastavena nenulová kapacita a bylo mu nastaveno datum konce přihlašování, které je vzhledem k aktuálnímu okamžiku v budoucnosti. Na nabízený projekt je možné vytvářet přihlášky.
- *Uzavřený.* Schválený projekt lze uzavřít. Tento stav vyjadřuje, že projekt již nebude dále upravován. Projekt by měl být uzavřený po skončení jeho řešení.

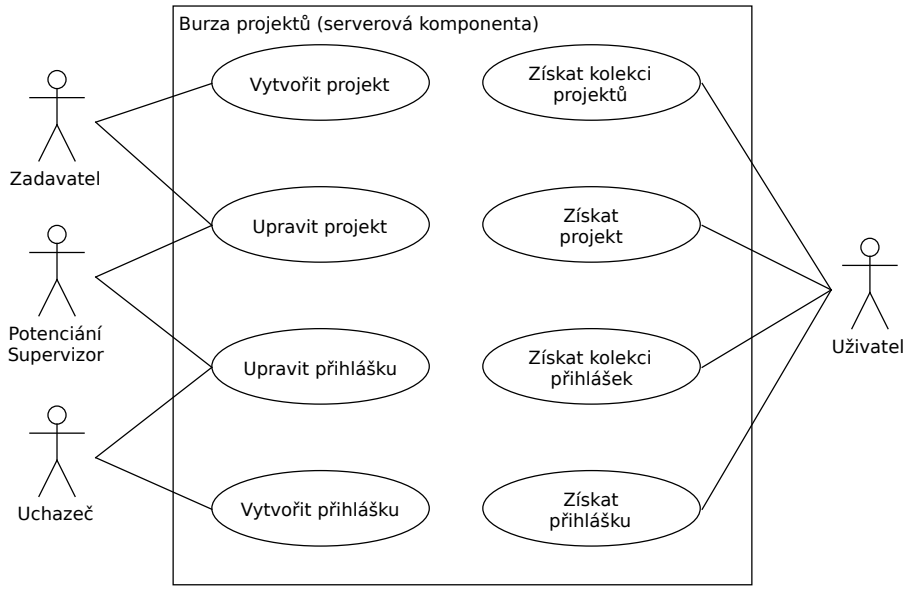
Stavy přihlášky

Stav přihlášky odpovídá hodnotě atributu stav doménové třídy přihlášky. Přihláška může být v těchto stavech:

- *Nová.* Nově vytvořená přihláška je ve stavu nová. Tato přihláška čeká na přijetí či odmítnutí.
- *Přijatá.* Pokud je uchazeč o řešení projektu přijat mezi řešitele, je jeho přihláška přijatá.

- *Odmítnutá.* Pokud je uchazeč shledán nevhodným pro řešení projektu, je jeho přihláška odmítnutá.
- *Zrušená.* Zrušená přihláška vyjadřuje, že uchazeč ztratil zájem o řešení projektu.

1.2.4 Operace s entitami

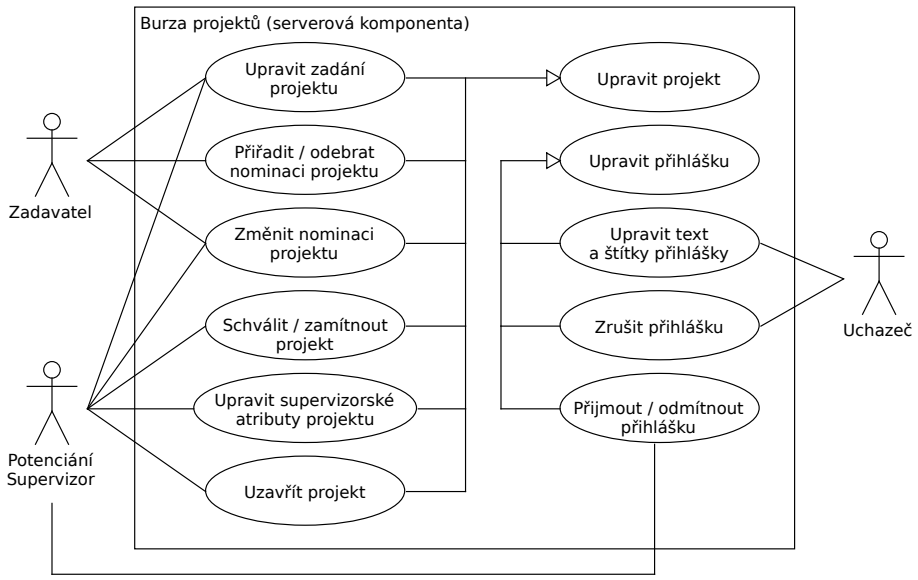


Obrázek 1.4: Případy užití serverové komponenty Burzy projektů.

Serverová komponenta Burzy projektů umožňuje uživatelům vykonávat operace se spravovanými entitami podle toho, jakou roli uživatelé vůči entitám mají. Uživatelé mohou provádět operace s projektem a přihláškami, které se k němu vztahují, pouze pokud splňují všechny striktní vlastnosti, které projekt vyžaduje.

Na obrázku 1.4 jsou formou případů užití zachyceny podporované operace. Jedná se o případy užití serverové komponenty prostřednictvím REST rozhraní, nikoli o případy užití celého systému. Speciální případy úprav projektu a přihlášky jsou znázorněny v diagramu na obrázku 1.5. Aktéři v diagramech odpovídají rolím uživatele definovaným v popisu domény (str. 6). Následuje popis operací:

- *Vytvořit projekt.* Libovolný uživatel může vytvořit nový projekt a stát se tak jeho zadavatelem. Poté může projekt upravovat (viz dále).



Obrázek 1.5: Speciální případy užití (úpravy entit) serverové komponenty Burzy projektů.

- *Upravit projekt.* Existující projekty lze upravovat, nejsou-li uzavřeny. Zadavatel může projekt upravovat pouze, pokud je nový nebo nominovaný. Může upravovat zadání a přiřadit, měnit nebo odebrat nominaci.

  Jestliže zadavatel přiřadí (změní) projektu nominaci, může se stát jeho potenciálním supervizorem. Pokud se jím nestane, musí být alespoň studentem všech instancí předmětů, které nominace projektu určuje, a pro projekt musí existovat alespoň jeden potenciální supervizor.

  Pokud je projekt nominovaný, může ho libovolný potenciální supervizor schválit nebo zamítnout. Zamítnutý projekt může později schválit. Schválenému projektu může upravovat zadání, měnit nominaci, supervizorské atributy a může ho uzavřít.

  Když potenciální supervizor změní nominaci projektu, musí on i případný supervizor projektu zůstat stále potenciálním supervizorem daného projektu.

  Pokud libovolný uživatel přiřadí (změní) projektu striktní vlastnosti, musí tyto vlastnosti splňovat. Pokud má projekt přiřazeného supervizora, musí je splňovat i on.

  Je-li projektu upravena nominace nebo striktní vlastnosti, jsou všechny dříve vytvořené přihlášky na tento projekt, které by po úpravě nemohly

vzniknout (uchazeč nesplňuje striktní vlastnosti nebo není dále uchazečem (vhledem k nominaci)), automaticky zamítnuty.

- *Vytvořit přihlášku.* Je-li projekt nabízený, mohou k němu vytvářet přihlášky uchazeči. Pro jeden projekt může mít každý uchazeč maximálně jednu přihlášku ve stavu nová nebo schválená. Při vytvoření přihlášky vyplní uchazeč text a štítky.

- *Upravit přihlášku.* Uchazeč může upravovat text a štítky své přihlášky, která je ve stavu nová. Novou nebo přijatou přihlášku může zrušit.

  Přihlášku ve správném stavu může také upravovat potenciální supervizor projektu, ke kterému se přihláška vztahuje. Pokud je přihláška nová, může ji přijmout nebo odmítnout. Přijatou přihlášku může odmítnout a naopak. Při úpravách může vyplnit poznámku supervizora.

- *Získat kolekci projektů (přihlášek).* Libovolný uživatel může získávat entity spravované Burzou projektů v kolekcích, které je možné stránkovat, filtrovat a řadit podle různých kritérií.

- *Získat projekt (přihlášku).* Entity, které může uživatel získat v kolekcích, může také získat jednotlivě.

Pro úplnost byly do přílohy A umístěny diagramy procesů zpracování spravovaných entit a stavové diagramy entit.

1.2.5 Použité technologie

Serverová komponenta Burzy projektů byla vyvíjena v jazyce Java za pomoci projektu *Spring Boot* [12]. Ten umožňuje rychlý vývoj webových služeb založených na *Spring Frameworku* [4]. Spring Boot provádí autokonfiguraci většiny použitých knihoven. Software založený na Spring Boot je možné buď spustit jako samostatnou aplikaci s vestavěným webovým serverem, nebo je možné ho nasadit na externí server jako WAR (Web Archive) soubor [13].

Stěžejní princip použitý v aplikacích založených na Spring Frameworku je *Inversion of Control* (IoC) . Spring Framework definuje strukturu nazvanou *IoC kontejner*. Na základě konfigurace je IoC kontejner naplněn instancemi tříd (tzv. bean třídy). Každá třída má definované závislosti na jiných třídách, se kterými spolupracuje. Propojení tříd řeší IoC kontejner. Tento princip propojení tříd je někdy také označován jako *dependency injection* [14].

Serverová komponenta Burzy projektů spravuje databázi pomocí projektu *Spring Data JPA* [15]. Tento modul provádí objektově relační mapování (ORM) doménových tříd na tabulky databáze, automaticky vytváří schéma databáze a usnadňuje dotazování. Dotazování je možné implementovat více způsoby [16].

Rozhraní serverové komponenty pro klientské aplikace má architekturu REST a využívá základní metody protokolu HTTP (Hypertext Transfer Protocol) pro vykonávání operací nad spravovanými entitami (`POST` pro vytvoření entity, `PUT` pro upravení entity, `GET` pro získání kolekce entit nebo jedné entity).

Díky použití projektu Spring Boot při vývoji serverové komponenty je možné se při implementaci zaměřit na vývoj důležitých částí aplikace. Zdrojové kódy serverové komponenty obsahují tyto hlavní části: soubor konfigurace, definice doménových tříd (mapují se na tabulky databáze), definice repozitářů (bean třídy, které umožňují dotazování), definice kontrolerů (bean třídy obsluhující REST rozhraní) a pomocné třídy.

1.2.6 Stav implementace

Serverová komponenta Burzy projektů byla na začátku této práce ve stádiu prototypu a nebyla řádně otestována. Při vývoji prototypu nebyl kladen dostatečný důraz na dobrý návrh tříd, výběr některých technologií a přehlednost kódu. V této části text budou popsány nejzávažnější nedostatky původní implementace, které je nutné nebo vhodné odstranit před realizací rozšíření systému. Zdrojem informací byly: inspekce kódu, sběr připomínek a manuální testování.

Nutný refaktoring velké části kódu

Nejzávažnějším problémem původní implementace je požadavek [17] na refaktoring (změnu struktury) důležitých částí kódu. Požadavek kritizuje původní třídu *Domain*, která nemá jasně vymezenou zodpovědnost, sdružuje operace, které by měly být rozděleny mezi více tříd, a její kód je složitý a nepřehledný. Pro třídu také neexistují automatické testy.

Protože by původní špatný návrh komplikoval implementaci rozšíření systému, bude nutné refaktoring provést. Nabízí se dvě možnosti: buď vytvořit nejdříve testy a poté provádět refaktoring po krocích s průběžným testováním, nebo kód nejdříve kompletně přepsat a následně doplnit testy. Psaní testů pro špatně navrženou třídu by bylo obtížné a pravděpodobně by bylo nutné podle testů ladit starý nepřehledný kód. Proto bude jednodušší zvolit druhou variantu.

Neexistence automatických testů

Původní implementace neobsahuje téměř žádné automatické testy. Po refaktoringu bude tedy nutné pokrýt vše novými testy.

Pro jednotkové testy v jazyce Java je běžné používat framework JUnit. Je to řešení pro psaní testů izolovaných tříd. Spring Boot aplikace ale vyžaduje také integrační testy, které testují spolupráci objektů IoC kontejneru. Pro

tento účel je možné zkombinovat JUnit s knihovnami, jako jsou například jMock, EasyMock nebo Mockito.

Jednodušším řešením je použít komplexnější nástroj, který sám umožňuje jak jednotkové, tak integrační testování. Takovým je například framework Spock [18] – testovací a specifikační framework pro jazyky Java a Groovy. Testy psané v jazyce Groovy s pomocí frameworku Spock mají následující výhody:

- Jazyk Groovy dovoluje kompaktnější syntaxi než Java, což může vést ke kratšímu a přehlednějšímu kódu.
- Spock podporoje důležité techniky pro testování spolupráce komponent programu, především „mocking“ a „stubing“. Ty umožňují při testování spolupráce objektů nahradit některé z objektů zástupnými objekty s předefinovaným chováním [19].
- Spock má výbornou podporu parametrizovaných a daty řízených testů. Díky tomu je možné jednoduše vykonávat podmínky testů pro různé parametry nebo pomocí většího objemu dat (např. tabulkou hodnot) [20].
- Spring Boot podporuje testování pomocí Spock frameworku [21].

Nevhodná volba technologií pro dotazování

Pro dynamické generování dotazů na kolekce entit byla při předchozím vývoji použita metoda vytváření predikátů pomocí *Specifications* [16]. Tato metoda vyžaduje psaní relativně dlouhého kódu. To se snaží implementace obejít generickými metodami. Výsledek je velmi nepřehledný a těžkopádný.

Další nevýhodou zvoleného řešení se ukázala být nutnost použít vnořenou třídu pro každý atribut doménových tříd, který je kolekcí. To dělá kód ještě více složitým. V opačném případě nastane chyba při aplikování vytvořených dotazů. Specifikace jsou navíc použité bez metamodelu, což může vést k běhovým chybám.

Jako vstup pro vytváření dotazů jsou použity různé parametry dotazů na RESTové rozhraní. Každý parametr je nejdříve zpracován zvlášť a vzniklé podmínky jsou následně spojeny logickou konjunkcí. Není tedy možné vytvářet podmínky, které by byly složitěji větvené pomocí logických spojek.

Zmíněné nedostatky by mohly být odstraněny použitím projektu Querydsl [22] pro dynamické generování dotazů a knihovny *rsql-parser* [23] pro syntaktickou analýzu složitějších podmínek zadaných uživatelem pomocí jednoho URL parametru, jak tomu je například v případě KOSapi [24].

Špatný návrh KOSapi klienta

Třída původní implementace s názvem *KosapiClientService*, která se stará o dotazy do KOSapi, není navržena ideálně. Neodstiňuje od detailů systému KOS (např. studentských a učitelských rolí), některé její veřejné metody mají nevhodné návratové typy a používá více dotazů do KOSapi, než je nutné, což vede k časovým prodlevám.

Chyby v definicích atributů doménových tříd

Doménové třídy mají nedostatečně specifikované typy některých atributů a automatický převod na relační databázový model vytváří sloupce s nevhodnými doménovými typy. Například popis projektu nemůže být delší než 255 znaků.

1.3 Uživatelské rozhraní

Uživatelské rozhraní Burzy projektů má sloužit jako prostředník mezi uživatelem a serverovou komponentou systému. Jeho účelem je prezentovat data a poskytnout uživateli grafické ovládací prvky. V následujícím textu bude popsán stav uživatelského rozhraní na začátku této práce. Budou analyzovány použité technologie, popsán stav implementace a rozebrány možnosti navázání na předchozí vývoj.

1.3.1 Použité technologie

Uživatelské rozhraní Burzy projektů bylo vyvíjeno pro nasazení na fakultní portálové řešení postavené na platformě Liferay [25]. Portálové řešení obecně umožňuje zobrazovat uživateli zároveň více grafických rozhraní (tzv. portletů) k více různým systémům v jeden okamžik na jedné obrazovce [7].

Uživatelské rozhraní Burzy projektů je vyvíjeno v jazyce Java a používá technologii JSF (Java Server Faces). To je jedna z realizací architektury MVC (Model-View-Controller), která definuje tyto druhy komponent: Model je komponenta, která spravuje doménové objekty. View slouží pro prezentaci dat uživateli. Controller je komponenta obsluhující akce uživatele [2, 7].

Uživatelské rozhraní Burzy projektů je zobrazováno Liferay portálem jako část webové stránky. Výstupem technologie JSF pro webový prohlížeč je HTML (HyperText Markup Language) kód s kaskádovými styly (CSS) a skripty v jazyce Javascript. Vzhled a chování uživatelského rozhraní ovlivňují také styly a skripty používané okolními portlety. Část fakultního portálu, pro kterou je určeno uživatelské rozhraní Burzy projektů, používá styly a skripty grafického tématu portálu SSP.

Uživatelské rozhraní také využívá části Spring Frameworku.

1.3.2 Stav implementace

Uživatelské rozhraní bylo přebíráno ve stavu nekompatibilním se serverovou komponentou systému. Přesto, že obě komponenty byly vyvíjeny s pomocí verzovacího systému, nepodařilo se dohledat verze obou komponent kompatibilní natolik, aby uživatelské rozhraní „nepadalo“ a zobrazovalo nějaký grafický výstup.

K uživatelskému rozhraní nebyla poskytnuta žádná programátorská dokumentace a kód nebyl komentovaný. Veškeré informace byly čerpány z inspekce kódu a z komunikace s vývojáři. Intenzita komunikace vyžadovaná pro pochopení kódu brzy přerostla míru únosnou oběma stranám.

V rámci této práce byl učiněn pokus o postupné upravování uživatelského rozhraní do stavu kompatibilního se serverovou komponentou. Tento pokus bohužel neuspěl pro příliš mnoho nekompatibilit a nejasností v kódu.

Byly identifikovány dva hlavní zdroje nekompatibilit uživatelského rozhraní se serverovou komponentou. Zaprvé je to fakt, že uživatelské rozhraní bylo vyvíjeno podle odlišných předpokladů (např. obrazovka pro editaci projektu obsahuje pouze možnost přiřadit projektu předměty, ale nedovoluje přiřadit semestr). Druhým zdrojem nekompatibilit je zbytečná a rozdílná duplikace aplikační logiky systému na straně uživatelského rozhraní.

1.3.3 Postup dalšího vývoje

Protože pokus o postupné upravování uživatelského rozhraní do funkční podoby neuspěl a jedním z původních požadavků na systém Burza projektů bylo uživatelské rozhraní, které je možné nasadit na fakultní Liferay portál, jeví se jako nejvhodnější postup pro další vývoj následující.

Bude vytvořeno nové uživatelské rozhraní jako aplikace pro Liferay portál s použitím původní implementace jako technologického základu s postupným přidáváním funkcí. Tento přístup umožní postupné seznamování se s použitými technologiemi a zužitkování hodnotných části původní implementace, jako jsou například konfigurační soubory pro Liferay, JSF a Spring Framework nebo různé ovládací prvky.

Původní implementace uživatelského rozhraní definuje tři uživatelské obrazovky, které měly sloužit pro zobrazení seznamu projektů, zobrazení detailu projektu a vytvoření nebo editaci projektu. Toto rozvržení obrazovek bude zachováno.

Dále bude v rámci refaktoringu serverové části Burzy projektů vytvořena knihovna obsahující datové typy a operace doménové logiky, které je vhodné sdílet mezi komponentami systému. Tím budou odstraněny hlavní zdroje nekompatibilit.

Kapitola 2

Analýza okolních systémů

Aby bylo možné analyzovat požadavky na rozšíření systému Burza projektů o nabídku projektů okolních systémů, bylo nutné nejdříve těmto systémům porozumět a shromáždit všechny potřebné informace.

Burza projektů bude nabízet projekty ze systému Závěrečné práce (systém ZP) a portálu Spolupráce s průmyslem (SSP). Tyto systémy budou v této kapitole postupně analyzovány. Informace byly získávány z veřejné dokumentace systémů a z komunikace s vývojáři.

Na závěr kapitoly bude proveden rozbor možností integrace systémů.

2.1 Systém Závěrečné práce

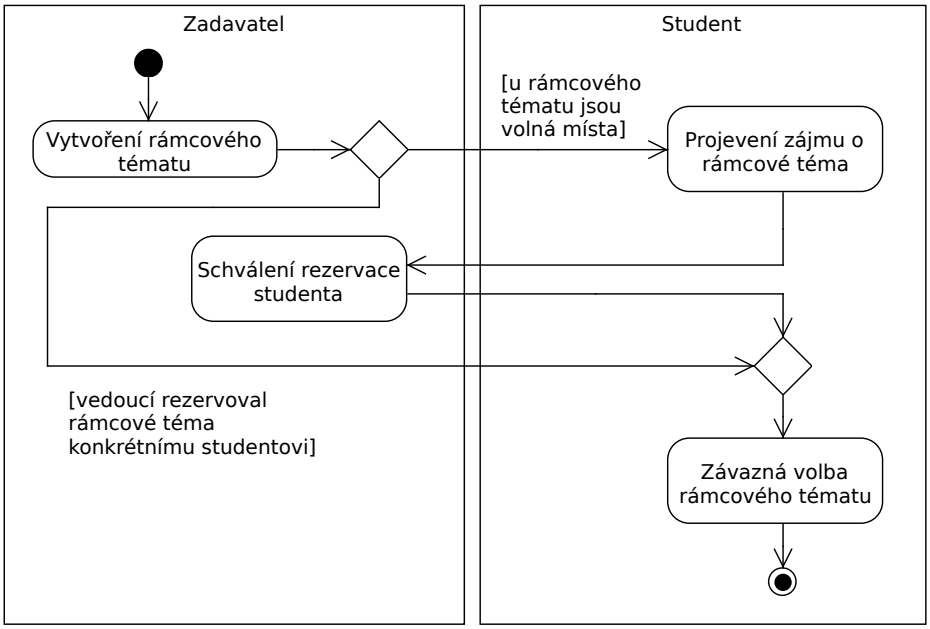
Systém ZP slouží pro zadávání a schvalování rámcových témat bakalářských a diplomových prací na FIT ČVUT. Jeho fungování je dáno směrnicemi děkana [26], které jsou převedeny na procesy vykonávané procesním portálem IBM BPM, který provozuje a spravuje Centrum znalostního managementu (CZM).

2.1.1 Proces zadávání rámcových témat

Na obrázku 2.1 je diagram procesu nabízení rámcových témat závěrečných prací (dále jen rámcová témata). Proces začíná vypsáním rámcového tématu zadavatelem. Tím může být akademický pracovník ČVUT, doktorand ČVUT nebo externí spolupracovník FIT. Prakticky může rámcové téma zadat libovolný člověk, který má v systému Usermap přiřazenu odpovídající roli.

Každý student si může rezervovat více rámcových témat. Rezervaci schválí nebo zamítne zadavatel. Zadavatel má také možnost rezervovat rámcové téma konkrétnímu studentovi. Student si může jedno ze svých schválených rámcových témat zvolit závazně.

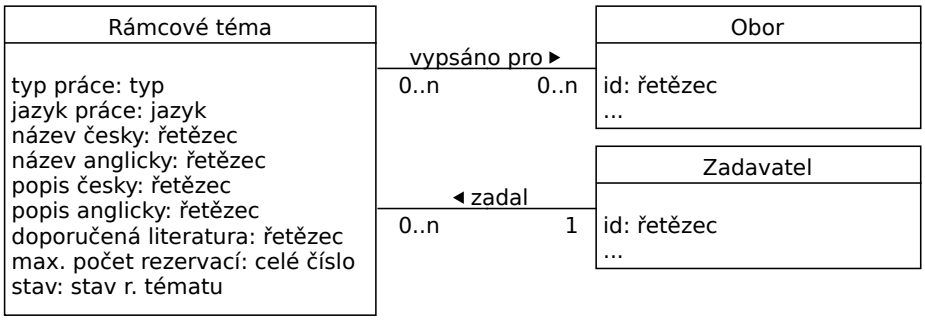
Potom, co si student závazně zvolí rámcové téma závěrečné práce, vznikne v systému zadavateli úkol vytvořit pro studenta konkrétní téma závěrečné práce. To je pak předáno do schvalovacího procesu.



Obrázek 2.1: Proces nabízení rámcových témat závěrečných prací.

Převzato z [27].

2.1.2 Doménový model rámcového tématu



Obrázek 2.2: Doménový model pro rámcové téma závěrečné práce.

V diagramu na obrázku 2.2 je znázorněna důležitá část doménového modelu systému ZP (rámcové téma a k němu se vztahující entity). Atribut typ

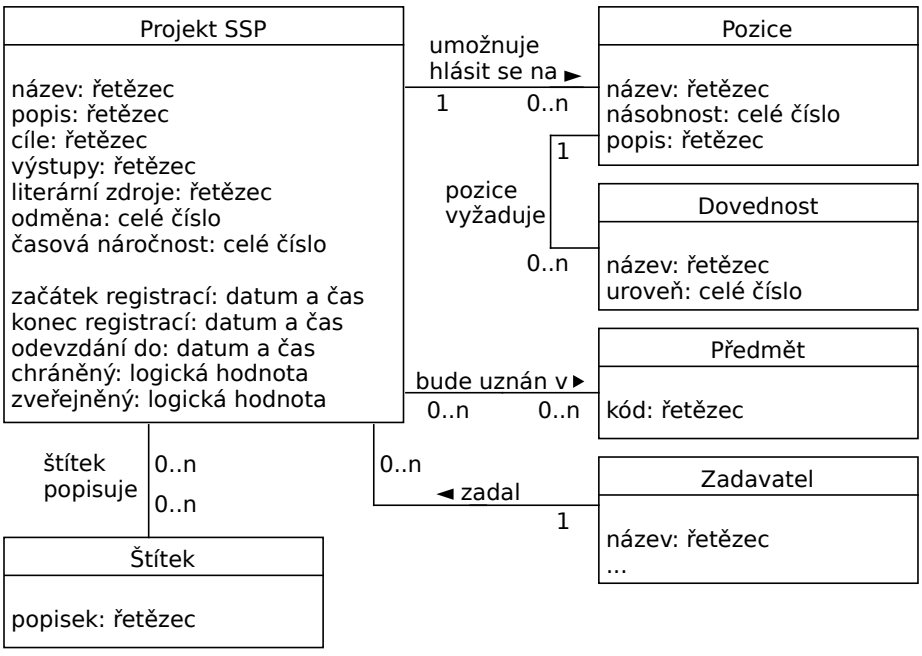
práce může nabývat hodnot: bakalářská nebo diplomová. Možné hodnoty jazyka práce jsou: český, slovenský, anglický a libovolný. Název a popis mohou být vyplněny jak v českém, tak anglickém jazyce. Může být vyjmenována doporučená literatura.

Každé rámcové téma má zadán maximální počet možných rezervací a může (ale nemusí) mít přiřazen jeden nebo více studijních oborů. Ty určují, kterým studentům se bude rámcové téma zobrazovat. Rámcové téma se vztahuje k jednomu zadavateli a nese informaci o svém stavu. Nejdůležitější pro účely nabízení je stav *nabízeno*.

2.2 Portál Spolupráce s průmyslem

Portál SSP je komplexní systém vyvíjený na FIT ČVUT. Je provozován na fakultním Liferay portálu. SSP umožňuje spolupráci fakulty s průmyslovými partnery. Ti zadávají do portálu projekty, které mohou řešit studenti za peněžní odměnu. Projekty také mohou být studentům uznávány jako semestrální práce.

2.2.1 Doménový model projektu SSP



Obrázek 2.3: Doménový model pro projekt SSP.

Na obrázku 2.3 je diagram doménového modelu projektu SSP a k němu se vztahujících entit.

Každý projekt SSP má název a textový popis, který je rozdělen mezi více položek různého významu: popis, cíle, výstupy a literární zdroje. Pro projekt je specifikována výše peněžní odměny v korunách a odhadovaná časová náročnost v hodinách. Projekt má určeno datum a čas začátku a konce registrací a odevzdání řešení. Projekt může být označen jako chráněný (viditelný pouze pro uživatele s podepsaným NDA). Projekt, který má být nabízen studentům, má nastaven příznak zveřejněný.

Projekty SSP jsou kategorizovány pomocí textových štítků. Pro každý projekt je určeno, na jaké pozice mají studenti možnost se hlásit. Každá pozice má název, násobnost a mohou jí být přiřazeny požadované dovednosti. Dovednost má název a číselné vyjádření úrovně od 1 do 5 (nejlepší).

Projektům SSP mohou být přiřazeny předměty, ve kterých bude jejich řešení uznáno jako semestrální práce. Projekt má zadavatele, který může být pro účely nabízení identifikován názvem průmyslového partnera.

2.2.2 Zadávání projektů SSP

Zadavateli projektů SSP jsou průmysloví partneři fakulty (tzv. sponzoři), kteří mají v systému vytvořený profil. Zadavatel kromě správy projektu také určuje, na jaké pozice v projektu se mohou studenti hlásit, a vybírá z přihlášených studentů vhodné řešitele.

Studenti se po zaregistrování v systému mohou hlásit na zveřejněné projekty. Studentský profil obsahuje informace o dovednostech a jejich úrovni. Tyto informace buď zadá student sám jako své subjektivní hodnocení, nebo jsou automaticky počítány ze studijních výsledků. Popis dovedností studentů je porovnáván s požadovanými dovednostmi rolí projektů a slouží zadavateli k výběru vhodných řešitelů z přihlášených studentů.

Učitelé mají v portálu SSP velmi omezenou roli. Po vytvoření profilu v systému mohou schvalovat projekty pro předměty, které vyučují, a určovat, za jakých podmínek řešení projektů v těchto předmětech uznají.

2.3 Možnosti integrace systémů

V této sekci textu bude proveden rozbor možností integrace Burzy projektů a analyzovaných okolních systémů. Nejdříve budou popsány obecné možnosti integrace, následně bude zvoleno vhodné řešení.

2.3.1 Obecné možnosti integrace

Integrace systémů může být provedena na úrovni prezentační vrstvy, na datové úrovni nebo na úrovni služeb [7].

Integrace na úrovni prezentační vrstvy může být realizována společným uživatelským rozhraním nebo portálovým řešením. Tato možnost má řadu nevýhod. Při prezentování projektů z různých zdrojů by vznikal problém se stránkováním dat a s různorodostí datových struktur. Data by bylo nutné upravovat při každém načítání a nebylo by možné dělat trvalé lokální úpravy.

Na datové úrovni je integrace systémů realizována sdíleným přístupem do databáze. Výměna dat je v tomto případě rychlá a data systémů jsou vždy synchronizovaná. Je ale nutné zajistit, aby schéma databáze vyhovovalo všem systémům, které do ní přistupují. Systémy musí data zamykat. Je nutné znát implementační detaily integrovaného systému a ty nemusí být veřejné nebo se mohou časem měnit. Může dojít ke zbytečné duplikaci aplikační logiky.

Preferovaným způsobem integrace systémů bývá integrace na úrovni služeb. Služby zprostředkovávají předem dohodnutá pevně daná zdokumentovaná rozhraní. Komunikace může být realizována například formou zasílání zpráv nebo vzdáleným voláním procedur (RPC). Nejčastěji se pro tyto účely používá protokol SOAP [28] (Simple Object Access Protocol) nebo architektura REST [29].

2.3.2 Zvolený způsob integrace

S vývojáři obou integrovaných systémů byla na základě jejich požadavků dohodnuta integrace systémů na úrovni služeb s následujícím postupem:

1. Budou navržena rozhraní pro nové webové služby integrovaných systémů. Služba systému ZP bude komunikovat protokolem SOAP, služba portálu SSP bude mít rozhraní s architekturou REST.
2. Budou vytvořeny zástupné webové služby s navrženými rozhraními.
3. Burza projektů provede integraci zástupných webových služeb.
4. (volitelně) Budou vytvořeny webové služby s dohodnutými rozhraními na straně integrovaných systémů.

Body 1 až 3 budou součástí této práce. Čtvrtý bod bude proveden později, pokud bude zájem ze strany fakulty pro reálné nasazení Burzy projektů.

Kapitola 3

Analýza požadavků

Na základě analýzy Burzy projektů a analýzy okolních systémů bude v této kapitole nejdříve upřesněn rozsah této práce a následně budou specifikovány požadavky na výsledný systém a jeho zamýšlené případy užití.

3.1 Upřesnění rozsahu práce

Z analýzy systémů v předchozích kapitolách je patrné, že práci na Burze projektů lze rozdělit na dvě části. Nejdříve bude nutné dokončit vývoj základního systému a následně bude systém rozšířen. Rozsah práce těchto dvou bodů bude upřesněn v následujících dvou sekcích textu.

3.1.1 Dokončení základního systému

Systém Burza projektů byl na začátku této práce přebírán v rozpracovaném stavu. Další vývoj serverové komponenty systému vyžaduje provedení refaktoringu kódu, přerozdělení zodpovědností tříd, pokrytí kódu automatickými testy a odstranění dalších nedostatků implementace. Uživatelské rozhraní bude od základů přepracováno s využitím původní implementace jako vzoru s postupným přidáváním funkcí.

Objem práce nutné pro dokončení základního systému bude značný. Řešení dalších požadavků a připomínek k základnímu systému bude proto ponecháno pro další vývoj po skončení této práce.

3.1.2 Rozsah rozšíření

Systém Burza projektů bude po rozšíření umožňovat nabízení rámcových témat závěrečných prací a projektů SSP společně s ostatními projekty Burzy projektů se stejnými možnostmi filtrování, řazení a stránkování.

Rámcových témat je mnoho a nemá smysl je v Burze projektů nabízet všechna. Zda bude rámcové téma nabízeno by měl mít možnost zvolit jeho

zadavatel. Měla by být nabízena jen rámcová témata nabízená v systému ZP k rezervování.

Projektů, které jsou v jeden okamžik nabízeny v SSP k přihlašování, bývá v řádech desítek a je možné je v Burze projektů nabízet automaticky všechny. Samozřejmě je nutné dodržet nabízení chráněných projektů SSP jen osobám s podepsaným NDA.

Burza projektů nebude kromě nabídky projektů SSP a rámcových témat systému ZP integrovat žádné další funkce těchto systémů (např. přihlašování na projekty SSP nebo rezervaci rámcových témat), a to především z následujících důvodů:

- Procesy okolních systémů se mohou měnit (např. se změnami fakultních směrnic).
- Vznikal by problém se synchronizací dat.
- Vznikal by problém s notifikacemi. Nebylo by jasné, který systém je má generovat a do kterého systému mají odkazovat.
- Byla by vyžadována součinnost druhé strany, kterou není možné zajistit.
- Integrace dalších funkcí by nepřinesla žádnou hodnotu navíc.

Pro operace s nabízenými projekty SSP a rámcovými tématy budou uživatelé Burzy projektů přesměrováni do odpovídajících systémů.

Aby při nabízení projektů nemusela být data načítána z více různých systémů, budou nabízené projekty SSP a rámcová témata do Burzy projektů importovány. Synchronizace dat mezi systémy bude probíhat na straně Burzy projektů manuální nebo automatickou aktualizací projektů vzniklých importem.

3.2 Požadavky

Jako závěr z předchozí analýzy budou v následujících bodech popsány požadavky na výsledný systém. Požadavky jsou obvyklým způsobem číslované a rozdělené na funkční (identifikátor začíná písmenem F) a nefunkční (identifikátor začíná písmenem N).

N1 *Refaktoring backendu.* Budou přerozděleny zodpovědnosti tříd serverové komponenty systému a přepracována jejich rozhraní tak, aby byla usnadněna tvorba automatických testů a implementace rozšíření. Operace a datové typy, které je vhodné pro zajištění kompatibility sdílet mezi komponentami systému, budou vyčleněny do samostatné knihovny. Bude přepracována implementace dotazování pro usnadnění vytváření nových dotazů pro účely uživatelského rozhraní.

N2 *Automatické testy.* Pro všechny důležité části kódu budou napsány jednotkové a integrační testy. Serverová komponenta systému bude otestována systémovými testy prostřednictvím REST rozhraní.

F1 *Manuální Import (aktualizace) rámcových témat.* Burza projektů umožní importovat rámcová témata závěrečných prací. Importovat bude možné pouze rámcová témata určená k nabízení studentům k rezervaci. Rámcová témata budou importována jejich zadavateli. Zadavatel bude moci zvolit, která jím nabízená rámcová témata budou importována.

Z importovaných rámcových témat vzniknou v Burze projektů nové projekty. Každý z nich bude mít vazbu na rámcové téma, ze kterého vznikl (zdrojové rámcové téma). Podle něho budou projektu automaticky nastaveny vhodné atributy (název, popis, předměty). Pokud bude importováno opakovaně stejné rámcové téma, bude pouze aktualizován z něho dříve vzniklý projekt.

F2 *Upravení importovaných rámcových témat.* Projekty vniklé importem rámcových témat bude moci zadavatel jejich zdrojových rámcových témat upravovat. Úprava bude mít smysl pouze pro atributy, které nebudou automaticky aktualizované (F3) (např. štítky nebo striktní vlastnosti).

F3 *Automatická aktualizace importovaných rámcových témat.* Burza projektů bude pravidelně (např. jednou denně) automaticky aktualizovat projekty vzniklé importem rámcových témat, jejichž aktualizace nebyla zakázána (viz F4). Při aktualizaci projektu budou nastaveny jeho atributy podle zdrojového rámcového tématu jako při importu. Pokud bude při aktualizaci projektu zjištěno, že jeho zdrojové rámcové téma již není v systému ZP nabízeno, bude pro projekt zakázáno další nabízení (viz F6) a aktualizace v Burze projektů.

F4 *Zakázání nabízení a aktualizace importovaných rámcových témat.* Zadavatel zdrojového rámcového tématu projektu, který vnikl importem tohoto rámcového tématu, bude moci zakázat další nabízení (viz F6) a aktualizaci (viz F3) daného projektu.

F5 *Automatický import a aktualizace projektů SSP.* Burza projektů bude periodicky (např. jednou denně) provádět automatický import všech projektů nabízených studentům v portálu SSP. Z nově nabízených projektů SSP vzniknou v Burze projektů nové projekty s vazbami na zdrojové projekty v SSP.

Při opakovaném importu stejných projektů SSP budou pouze aktualizovány dříve vniklé projekty v Burze projektů. Pokud bude při aktualizaci zjištěno, že nějaký dříve importovaný projekt již není v systému SSP nabízen, bude pro tento projekt zakázáno nabízení (viz F6) i v Burze projektů.

Projektům vzniklým importem budou podle zdrojových projektů SSP nastaveny vhodným způsobem atributy (název, popis, štítky, předměty, atd.).

F6 *Nabízení projektů vzniklých importem.* Importované projekty budou uživatelům zobrazovány společně s ostatními projekty Burzy projektů. Importované projekty, jejichž nabízení nebylo zakázáno (viz F3, F4 a F5), budou zobrazovány mezi ostatními nabízenými projekty. Importované projekty budou v nabídce explicitně označeny (např. jako „externí“) a bude u nich zobrazen odkaz na zdrojový projekt (rámcové téma) v systému jeho původu.

N3 *Rozhraní pro systém ZP.* Se systémem ZP bude Burza projektů komunikovat přes nově navrženou webovou službu s protokolem SOAP.

N4 *Rozhraní pro portál SSP.* S portálem SSP bude Burza projektů komunikovat pomocí nově navrženého rozhraní s architekturou REST.

3.3 Případy užití

Případy užití definují, jakým způsobem (v jakých krocích) dosáhne uživatel nějakého cíle (např. importu rámcového tématu). Případy užití jsou realizovány interakcí uživatele s obrazovkami uživatelského rozhraní.

Při dalším vývoji uživatelského rozhraní Burzy projektů bude zachováno rozložení obrazovek původní implementace. Uživatelské rozhraní nabídne následující tři základní obrazovky:

- *Zobrazení seznamu projektů.* Toto bude úvodní obrazovka. Obrazovka umožní zobrazení seznamu projektů s možnostmi filtrování, řazení a stránkování. Uživatel bude moci vyhledávat projekty podle různých kritérií. Pro každý projekt v seznamu budou zobrazeny nejdůležitější informace.

- *Zobrazení detailu projektu.* Po zvolení projektu v seznamu projektů, bude moci uživatel zobrazit obrazovku s jeho detailem. Detail projektu nabídne kompletní informace o projektu. Obrazovka také zobrazí seznam přihlášek na projekt a bude je zde možné spravovat.

  Pokud bude zobrazen detail importovaného projektu, bude zobrazen odkaz na jeho zdrojový projekt (rámcové téma).

  Z této obrazovky se bude možné vrátit zpět na seznam projektů.

- *Formulář pro úpravy projektu.* Tato obrazovka nabídne formulář s položkami odpovídajícími atributům projektu. Prostřednictvím formuláře bude možné projekty upravovat nebo zadávat jejich počáteční hodnoty při vytvoření.

Tuto obrazovku bude možné vyvolat buď zvolením možnosti „vytvořit projekt“ na úvodní obrazovce, nebo zvolením možnosti „upravit projekt“ na obrazovce detailu projektu.

Z obrazovky se bude možné také vrátit buď na detail projektu, nebo na seznam projektů. Při úspěšném dokončení úprav projektu, bude zobrazen jeho detail.

V rámci rozšíření systému bude přidána následující obrazovka, která pokryje požadavky na manuální import a aktualizaci rámcových témat a ovládání jejich nabízení (F1, F4):

- *Správa nabízených rámcových témat.* Uživatel, který může zadávat rámcová témata v systému ZP, bude moci po zvolení možnosti „nabízet rámcová témata“ na úvodní obrazovce zobrazit obrazovku realizující operace s nabízenými rámcovými tématy.

  Tato obrazovka uživateli zobrazí seznam rámcových témat, která zadal a jsou nabízená v systému ZP. V seznamu bude moci vyhledávat podle názvu.

  U rámcových témat, která ještě nebyla importována, bude nabídnuta možnost „importovat“. Tato témata budou importována a nabízena. U rámcových témat, která již byla importována, bude možné podle jejich stavu zvolit „aktualizovat“ nebo „nenabízet“. První možnost aktualizuje atributy importovaného projektu a ten bude nabízený. Druhá možnost bude dostupná pro nabízený projekt a zakáže jeho nabízení a aktualizaci.

  Po potvrzení změn se uživatel vrátí na seznam projektů.

Kapitola 4

Návrh dokončení systému

V této kapitole bude popsán návrh dokončení vývoje základních funkcí systému Burza projektů. Návrh je rozdělený na dvě části. Nejdříve bude popsán návrh přerozdělení zodpovědností tříd. Ve druhé části kapitoly bude vysvětlen postup návrhu obrazovek uživatelského rozhraní.

4.1 Přerozdělení zodpovědností tříd

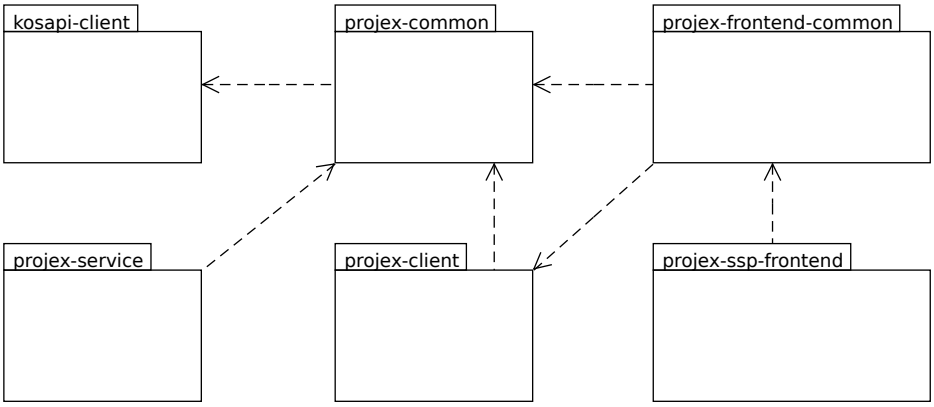
Návrh přerozdělení zodpovědností tříd bude základem pro refaktoring kódu. Jeho cílem je odstranit nedostatky návrhu serverové komponenty, usnadnit tvorbu automatických testů, zajistit kompatibilitu komponent systému a usnadnit další vývoj.

Pro rozdělování zodpovědností třídám byly použity návrhové vzory GRASP (General Responsibility Assignment Software Patterns) [2, 7]. Základní vzory jsou tyto:

- *Informační expert.* Tento vzor přiřazuje zodpovědnost třídě, která má informace potřebné ke splnění zodpovědnosti. Třída nemusí pro splnění zodpovědnosti získávat data využíváním jiných tříd a je tak podpořen vzor nízké provázanosti (viz. dále).

- *Nízká provázanost.* Na základě tohoto vzoru jsou přiřazeny zodpovědnosti třídám tak, aby každá třída byla co nejméně závislá na ostatních (aby byly třídy co nejméně provázané). Tento vzor přispívá k jednoduchosti kódu a umožňuje snadnější testování interakce tříd.

- *Vysoká soudržnost.* Použití tohoto vzoru spočívá v rozdělení zodpovědností tak, aby každá třída byla zaměřena na řešení jednoho úkolu (tedy její zodpovědnosti měly vysokou soudržnost). Třídy s jasně definovaným zaměřením se lépe spravují a testují.



Obrázek 4.1: Diagram návrhu balíčků Burzy projektů.

Společně s přerozdělením zodpovědností mezi třídy byly přerozděleny i třídy mezi balíčky. Návrh balíčků je zachycen v diagramu na obrázku 4.1. Balíčky budou detailněji popsány v následujících sekcích textu.

4.1.1 Balíček kosapi-client

Balíček *kosapi-client* byl navržen jako univerzální knihovna pro získávání dat ze služeb KOSapi a Umapi. Vznikl oddělením a zobecněním odpovídajícího kódu serverové komponenty systému.

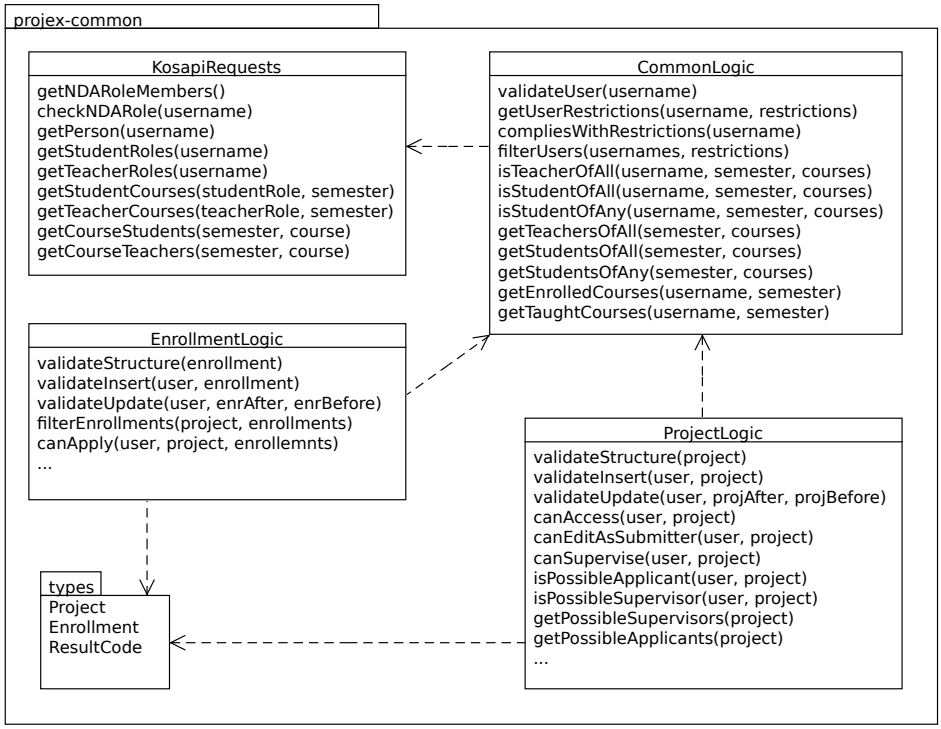
Balíček obsahuje tyto prvky:

- *Klientské třídy.* Nejdůležitější součástí balíčku jsou třídy *KosapiClient* a *UmapiClient*, z nichž každá realizuje dotazy do jednoho z fakultních informačních systémů.
- *DTO třídy.* Balíček obsahuje třídy pro reprezentaci dat získaných z fakultních informačních systémů. Pro KOS je to např. třída pro uložení osoby, studenta, instance předmětu, atd.
- *Pomocné třídy.* Návrh balíčku počítá s dalšími třídami pro zapouzdření pomocných funkcí a konstant. Budou sloužit například pro sestavování dotazů nebo extrakci dat z DTO objektů.

Vytvoření tohoto balíčku bude mít dva přínosy. Zaprvé budou od kódu Burzy projektů odděleny, zapouzdřeny a samostatně otestovány obecné mechanizmy pro práci se službami KOSapi a Umapi. Zadruhé bude umožněno znovupoužití této knihovny i v jiných aplikacích. Tím bude navíc pokryt požadavek [30] na wiki stránkách KOSapi.

4.1.2 Balíček projex-common

Do balíčku *projex-common* byla umístěna logika a datové typy, které je vhodné sdílet mezi komponentami systému. Účelem balíčku je předejít duplikaci aplikační logiky a zajistit kompatibilitu komponent. Detailní strukturu balíčku zachycuje diagram na obrázku 4.2. Následuje popis prvků návrhu:



Obrázek 4.2: Návrhové třídy balíčku projex-common.

- *KosapiRequests* je třída realizující konkrétní dotazy do služeb KOSapi a Umapi. Za tímto účelem vhodně využívá tříd z balíčku *kosapi-client*. Díky této třídě je možné například získat seznam uživatelů s NDA, otestovat, zda konkrétní uživatel splňuje NDA, získat studentské nebo učitelské role uživatele, získat seznam předmětů pro studentskou nebo učitelskou roli v daném semestru nebo získat seznam studentských a učitelských rolí pro danou instanci předmětu.

- *CommonLogic* je třída, která zapouzdřuje základní logiku systému a odstiňuje od dotazování se do okolních systémů a detailů s tím spojených (například studentských a učitelských rolí KOSu). Zodpovědnostmi

třídy jsou: validace uživatele, získání striktních vlastností uživatele, testování uživatele vůči striktním vlastnostem, filtrování uživatelů množinou striktních vlastností, testování uživatele na roli vůči nominaci, získání všech uživatelů s danou rolí vzhledem k nominaci a získání předmětů zapsaných nebo vyučovaných uživatelem v daném semestru.

- *types* je podbalíček obsahující datové typy pro reprezentaci základních entit systému a třídu *ResultCode*, která zapouzdřuje konstanty pro chyby validačních metod. Třídy *Project* a *Enrollment* odpovídají datovým typům předávaným mezi komponentami systému. Nad těmito třídami je vystavěna další aplikační logika.

- *ProjectLogic* je třída, prostřednictvím které je možné vykonávat základní aplikační logiku nad datovou strukturou projektu. Tato třída validuje strukturu projektu, validuje změny projektu, poskytuje informace o rolích a právech uživatele vzhledem k projektu (zda může projekt zobrazovat, editovat jako zadavatel nebo potenciální supervizor, atd.) a také poskytuje seznamy potenciálních supervizorů a možných uchazečů.

- *EnrollmentLogic* je třída realizující aplikační logiku nad přihláškami. Validuje strukturu přihlášky, validuje vytvoření a úpravu přihlášky, filtruje přihlášky v případě změn v projektu (striktních vlastností nebo nominace) a testuje, zda může daný uživatel vytvořit přihlášku na konkrétní projekt.

Třídy balíčku, které realizují doménovou logiku vznikly extrakcí logiky z třídy *Domain* původní implementace. Implementace tohoto balíčku bude obsahovat většinu částí původní implementace serverové komponenty systému, které nejsou svázány s obsluhou databáze nebo obsluhou požadavků REST rozhraní.

4.1.3 Další balíčky

Další balíčky budou popsány stručně. Části původní serverové komponenty systému, které nebyly přemístěny do balíčků popsaných v předchozím textu, budou zachovány v balíčku *projex-service*. Struktura implementace balíčku bude odpovídat Spring Boot aplikaci.

Návrh balíčku *projex-client* obsahuje jedinou třídu *ProjexClient*, kterou je možné použít pro dotazování se na REST rozhraní serverové komponenty. Balíček bude možné použít libovolnou klientskou aplikací a také bude použit pro systémové testy serverové komponenty.

Balíček *frontend-common* byl navržen pro umožnění znovupoužitelnosti kódu uživatelského rozhraní, který není spjatý se zvolenou platformou (Liferay, JSF, ...). Balíček *frontend-ssp-client* bude obsahovat přepracované uživatelské rozhraní.

4.2 Návrh obrazovek uživatelského rozhraní

Pro uživatelské rozhraní Burzy projektů bylo nutné navrhnout obrazovky, které budou realizovat případy užití systému. Uživatelské rozhraní bude zobrazováno jako portlet fakultním portálovým řešením. Část portálu, kde bude uživatelské rozhraní umístěno, používá grafické téma navržené pro SSP. Vývojáři SPP poskytli pro účely této práce zdrojové kódy grafického návrhu tohoto tématu.

Protože Burza projektů řeší podobný problém jako portál SSP, bylo možné pro uživatelské rozhraní použít mnoho prvků grafického návrhu SSP. Zbytek prvků bude implementován tak, aby byl vzhled uživatelského rozhraní jednotný a zapadal do vzhledu portálu.

Výsledek návrhu obrazovek uživatelského rozhraní bude možné najít v uživatelské příručce (příloha A).

Kapitola 5

Návrh rozšíření nabídky projektů

Návrh rozšíření systému o nabídku projektů okolních systémů, kterým se zabývá tato kapitola, je rozdělen na dvě části. První část se zabývá importem a aktualizací projektů okolních systémů obecně, druhá část se věnuje specifikům importu a nabídky projektů SSP a rámcových témat závěrečných prací.

5.1 Obecný návrh importu projektů

Tato část textu popisuje obecný návrh importu a aktualizace projektů okolních systémů. Návrh poskytne teoretickou možnost nabízet v Burze projektů projekty libovolného systému. Návrh je rozdělen na návrh rozšíření datové struktury projektu, návrh úprav doménové logiky a návrh obecného algoritmu pro import a aktualizaci projektů.

5.1.1 Nové datové položky projektu

Do datové struktury projektu budou přidány následující položky:

- *Zdroj projektu.* Tento atribut bude výčtového typu a bude určovat, jaký je původ projektu (Burza projektů, Systém ZP, portál SSP, v budoucnu případně další). Za importované budou považovány projekty, jejichž zdrojem nebude Burza projektů.
- *Externí ID.* Tato položka bude identifikátorem importovaného projektu v systému jeho původu.
- *Externí URL.* Importovaný projekt bude mít také vyplněn webový odkaz na zdrojový projekt v systému původu.

5.1.2 Úpravy doménové logiky

Budou provedeny následující změny a rozšíření doménové logiky:

- Nově vytvořený importovaný projekt bude ve stavu *nový* a budou mu nastaveny výše popsané rozšiřující atributy. Nastavení dalších atributů bude závislé na konkrétním druhu importovaného projektu.

- Importované projekty nebudou mít povinně přiřazeného zadavatele. Projekt bez zadavatele nebude moci upravovat žádný uživatel. Importovanému projektu bude nastaven zadavatel, pokud to pro konkrétní druh projektu bude dávat smysl (např. při importu rámcového tématu bude za zadavatele projektu dosazen zadavatel rámcového tématu)

- Importované projekty nebude možné nominovat. Tedy nebudou pro ně existovat potenciální supervizoři, nebude je možné schvalovat, atd.

- Zadavateli projektu bude nově povoleno projekt uzavřít. Uzavření importovaného projektu bude sloužit jako příznak, že projekt nebude dále automaticky aktualizován a nebude nabízen.

- Importovaný projekt, který nebude uzavřen (bude ve stavu nový) bude zobrazován mezi nabízenými projekty.

5.1.3 Algoritmus importu a aktualizace projektů

Obecný algoritmus, který bude sloužit k importu projektů a aktualizaci importovaných projektů, je popsán pseudokódem jako Algoritmus 1 na straně 39.

Vstupem algoritmu jsou dvě kolekce projektů. První, označená jako *internalProjects*, obsahuje projekty dříve importované do Burzy projektů, které jsou určeny k aktualizaci. Druhá kolekce, pojmenovaná *externalProjects*, je tvořena projekty, které jsou určeny pro import nebo aktualizaci dříve importovaných projektů. Obě vstupní kolekce mohou být prázdné. Výstupem algoritmu je kolekce obsahující nově importované a aktualizované projekty.

Algoritmus pracuje na principu „slévání“ uspořádaných množin projektů. Obě vstupní kolekce jsou nejdříve uspořádány podle zdroje a externího identifikátoru a poté jsou postupně porovnávány jejich prvky.

První větev porovnávání je vykonána, pokud pro nějaký dříve importovaný projekt neexistuje projekt určený k importu se stejným zdrojem a externím ID. V takovém případě se předpokládá, že importovaný projekt není v systému svého původu dále nabízen, a proto je v Burze projektů uzavřen (není dále nabízen ani aktualizován).

Druhá možnost nastává, pokud pro nějaký projekt určený k importu neexistuje v Burze projektů importovaný projekt se stejným zdrojem a externím ID. Potom je v Burze projektů vytvořen nový projekt.

Zbývající případ porovnání je proveden tehdy, když jsou ve vstupních kolekcích nalezeny projekty se stejným zdrojem a externím ID. V tom případě je dříve importovaný projekt aktualizován.

Algoritmus 1 Import a aktualizace projektů

Vstup: Kolekce interních projektů *internalProject* a kolekce kolekce projektů pro import *externalProjects*.
Výstup: Kolekce projektů pro update databáze.

```
result ← ∅
internalIterator ← iterator(sortBySourceAndExtId(internalProjects))
externalIterator ← iterator(sortBySourceAndId(externalProjects))
projectToUpdate ← NULL
sourceProject ← NULL
while hasNext(internalIterator) or hasNext(externalIterator)
or projectToUpdate ≠ NULL or sourceProject ≠ NULL do
  if projectToUpdate = NULL and hasNext(internalIterator) then
    projectToUpdate ← next(internalIterator)
  end if
  if sourceProject = NULL and hasNext(externalIterator) then
    sourceProject ← next(externalIterator)
  end if
  if sourceProject = NULL
  or 0 > compareBySourceAndExtId(projectToUpdate, sourceProject)
  then
    closeProject(projectToUpdate)
    add(result, projectToUpdate)
    projectToUpdate ← NULL
  else if projectToUpdate = NULL
  or 0 < compareBySourceAndExtId(projectToUpdate, sourceProject)
  then
    add(result, createProject(sourceProject))
    sourceProject ← NULL
  else
    updateProject(projectToUpdate, sourceProject)
    add(result, projectToUpdate)
    projectToUpdate ← NULL
    sourceProject ← NULL
  end if
end while
```

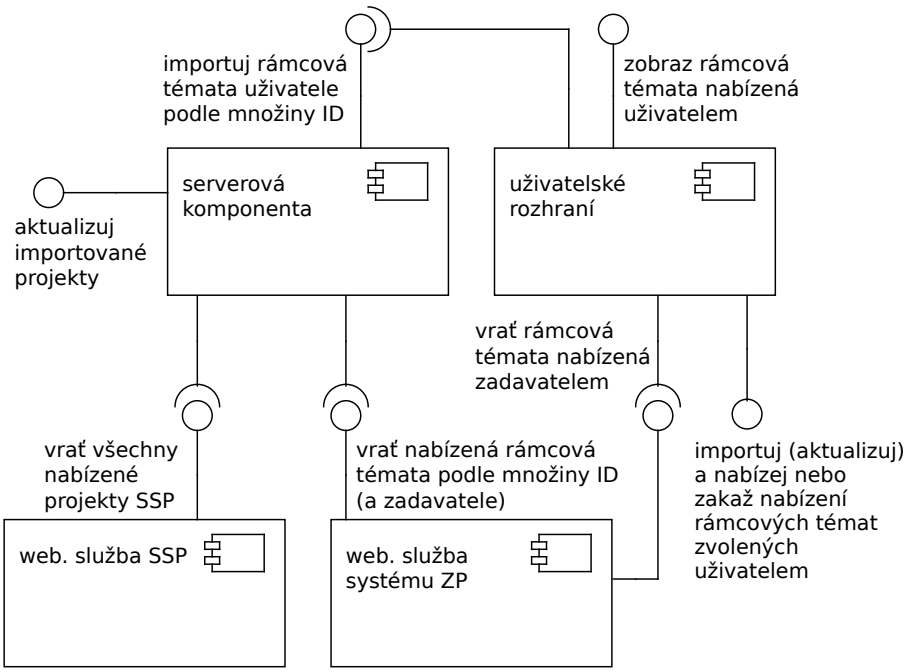
Při vytvoření nebo aktualizaci importovaného projektu jsou nejdříve nastaveny základní atributy (stav, zdroj, externí ID, externí URL, datum vytvoření/uzavření) a následně jsou nastaveny další atributy podle konkrétního typu externího projektu.

5.2 Nabídka rámcových témat a projektů SSP

V této části textu budou popsány jednotlivé kroky návrhu importu rámcových témat a projektů SSP a jejich nabídky v Burze projektů. Návrh vhodným způsobem využije obecný návrh importu projektů.

Pro systém ZP a portál SSP byly podle požadavků navrženy nové webové služby. V dalším textu bude nejdříve popsán návrh rozhraní nových služeb a také návrh nových rozhraní původních komponent Burzy projektů. Následovat bude popis interakce komponent při realizaci importu a nabídky projektů. Nakonec budou popsány datové struktury pro výměnu projektů mezi systémy.

5.2.1 Návrh rozhraní komponent



Obrázek 5.1: Návrh nových rozhraní komponent pro realizaci importu a nabízení projektů.

Na obrázku 5.1 je diagram zachycující návrh nových rozhraní všech komponent podílejících se na importu, aktualizaci a nabízení projektů SSP a rámcových témat. Následuje popis rozhraní.

Uživatelské rozhraní Burzy projektů poskytne uživateli tyto nové možnosti:

- *Zobraz rámcová témata nabízená uživatelem.* Toto rozhraní umožní uživateli zobrazit seznam rámcových témat nabízených v systému ZP, pro které je daný uživatel zadavatelem.
- *Importuj (aktualizuj) a nabízej nebo zakaž nabízení rámcových témat zvolených uživatelem.* Uživatel bude moci pomocí tohoto rozhraní zahájit import (aktualizaci) a nabízení zvolených rámcových témat a také zakázat nabízení jiných zvolených rámcových témat.

Serverová komponenta Burzy projektů nabídne dvě nová rozhraní:

- *Importuj rámcová témata uživatele podle množiny ID.* Rozhraní umožní provést import rámcových témat zadaných a nabízených uživatelem v systému ZP, jejichž identifikátor náleží dané množině.
- *Aktualizuj importované projekty.* Toto rozhraní umožní vykonání aktualizace dříve importovaných rámcových témat, aktualizace dříve importovaných projektů SSP a import nových projektů SSP. Rozhraní bude periodicky automaticky voláno.

Webová služba systému ZP bude poskytovat následující rozhraní:

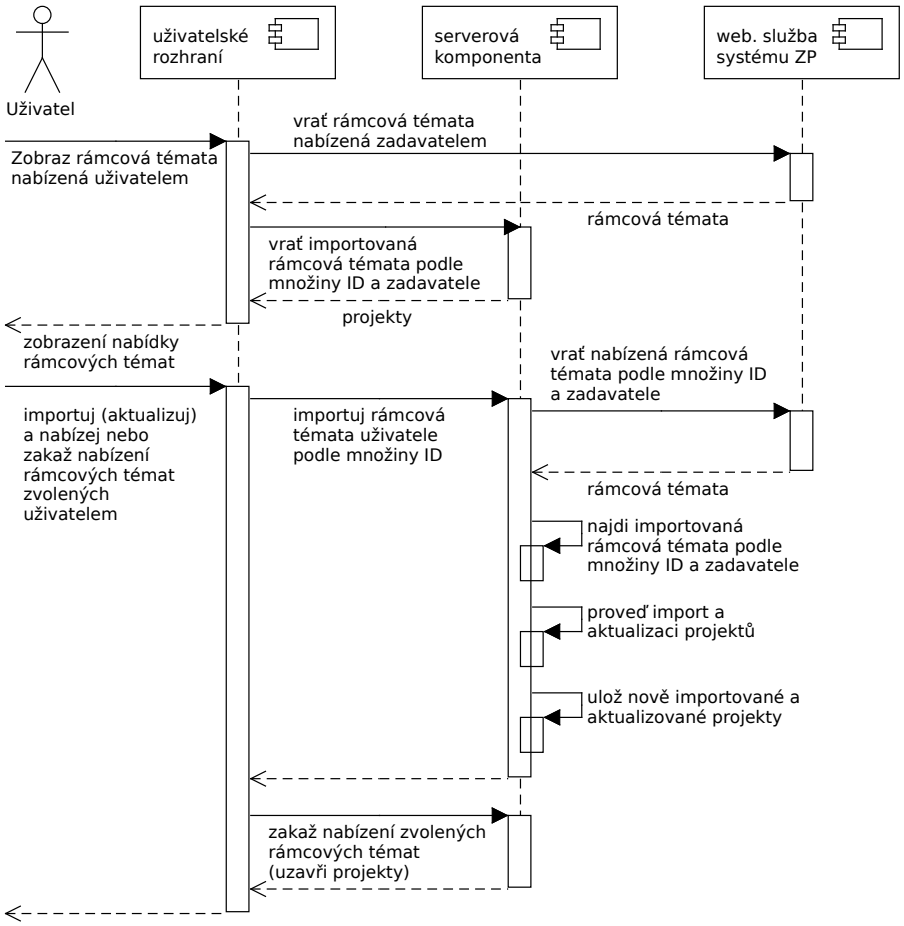
- *Vrať rámcová témata nabízená zadavatelem.* Toto rozhraní bude sloužit pro získání všech rámcových témat, která jsou nabízena v systému ZP daným zadavatelem.
- *Vrať nabízená rámcová témata podle množiny ID (a zadavatele).* Návratovou hodnotou volání tohoto rozhraní bude kolekce rámcových témat, která jsou nabízená v systému ZP a mají identifikátor ze zadané množiny. Volitelně bude možné požadovat pouze témata konkrétního zadavatele.

Webová služba portálu SSP nabídne pouze jedno rozhraní nazvané *vrať všechny nabízené projekty SSP*. Toto rozhraní vrátí všechny aktuálně nabízené projekty SSP.

5.2.2 Interakce komponent v čase

V této sekci textu bude popsána interakce mezi komponentami pomocí rozhraní navržených v předchozí části textu. Nejdříve bude uvedena sekvence volání rozhraní při realizaci správy nabízených rámcových témat. Poté bude popsána posloupnost volání při aktualizaci importovaných projektů.

Správa nabízených rámcových témat



Obrázek 5.2: Sekvenční diagram pro správu nabízených rámcových témat.

Na obrázku 5.2 je sekvenční diagram spolupráce komponent při správě nabízených rámcových témat.

Uživatel iniciuje celý proces tak, že vznese požadavek na zobrazení seznamu jím nabízených rámcových témat. Na tento seznam se uživatelské rozhraní doptá webové služby systému ZP. Serverová komponenta poskytne seznam těch rámcových témat z předchozího dotazu, která již byla dříve importována (Pro tuto akci není nutné přidávat rozhraní, jde o dotaz na kolekci projektů).

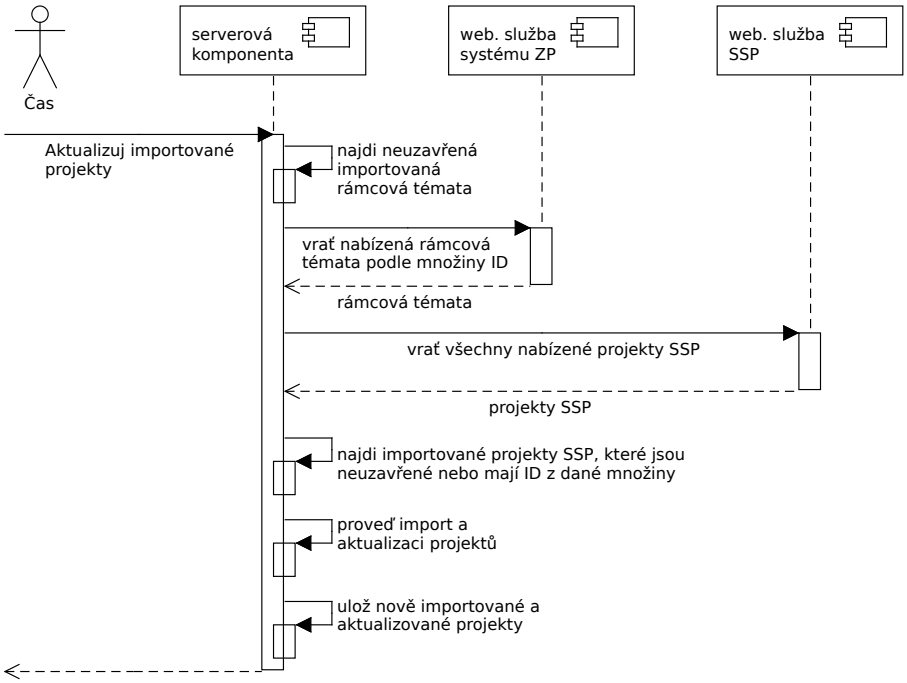
Uživateli je zobrazen seznam rámcových témat. Pro každé rámcové téma může uživatel zvolit, zda bude proveden import (nebo aktualizace, pokud již bylo importováno) a bude nabízeno, nebo zda bude zakázáno jeho nabízení

(pokud je nabízeno). Po potvrzení volby jsou uživatelskému rozhraní předány identifikátory zvolených rámcových témat. Uživatelské rozhraní deleguje obě akce (import i zákaz nabízení) na serverovou komponentu.

Pro realizaci importu si serverová komponenta nejdříve vyžádá od webové služby systému ZP seznam rámcových témat zadaných a nabízených daným uživatelem podle zvolených identifikátorů. Dále nalezne v databázi všechna dříve importovaná rámcová témata podle stejných parametrů. Získané kolekce jsou předloženy jako argumenty obecnému algoritmu pro import a aktualizaci projektů a výsledek je uložen do databáze.

Zákaz nabízení je realizován uzavřením zvolených importovaných projektů stejně jako uzavření běžného projektu (pro tuto akci opět není nutné přidávat rozhraní).

Aktualizace importovaných projektů



Obrázek 5.3: Sekvenční diagram aktualizace importovaných projektů.

Aktualizace importovaných projektů spojuje několik operací: Provádí aktualizaci dříve importovaných rámcových témat, aktualizuje importované projekty SSP a provádí import nově nabízených projektů SSP. Na obrázku 5.3 je sekvenční diagram celého procesu.

Operace je spouštěna automaticky vždy po uplynutí určitého času a je vykonávána serverovou komponentou systému. Ta nejdříve shromáždí vstupy pro algoritmus importu a aktualizace, poté provede tento algoritmus a výsledek použije pro aktualizaci databáze.

Vstupy pro algoritmus jsou získány následovně: Nejdříve jsou v databázi nalezena všechna importovaná rámcová témata, pro která nebyla zakázána aktualizace (nebyla uzavřena). Dále jsou z webové služby systému ZP získána nabízená rámcová témata, která odpovídají rámcovým tématům získaným předchozím dotazem.

Z webové služby SSP jsou získány všechny aktuálně nabízené projekty SSP. Poté jsou v databázi nalezeny všechny importované projekty SSP, které nebyly uzavřeny nebo mají identifikátor shodný s některým projektem SSP získaným předchozím dotazem.

Sjednocení nalezených dříve importovaných rámcových témat a projektů SSP je použito jako první parametr, sjednocení rámcových témat a projektů SSP získaných z okolních služeb jako druhý parametr algoritmu pro import a aktualizaci.

5.2.3 Datové struktury externích projektů

Další částí návrhu webových služeb okolních systémů je návrh datových struktur pro návratové hodnoty dotazů.

Tyto datové struktury jsou znázorněny v diagramu na obrázku 5.4. Společným základem je struktura *Externí projekt*, která obsahuje nutné atributy pro algoritmus importu a aktualizace.

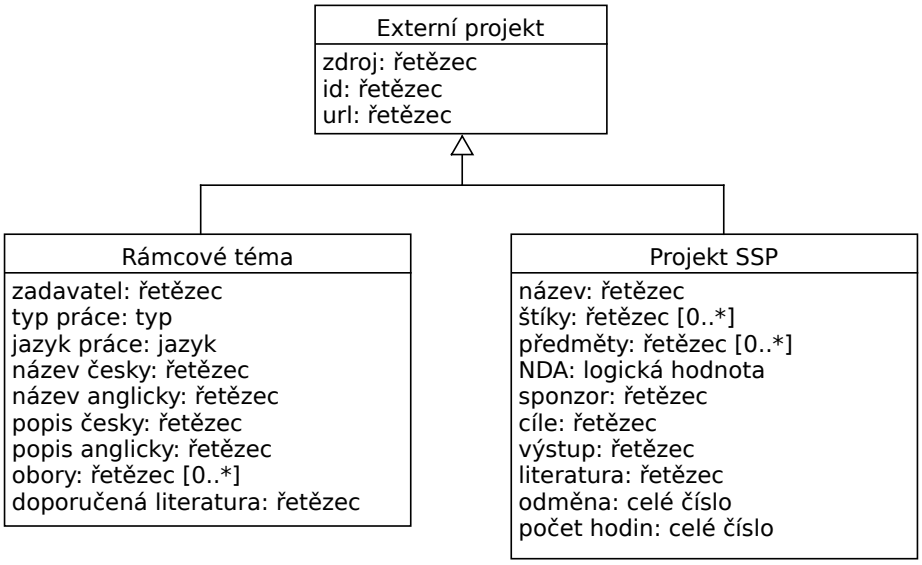
Pro každý z okolních systémů, jehož nabídku projektů bude Burza projektů integrovat, bude ze struktury externího projektu odvozena konkrétní datová struktura specifická pro daný systém. Podle atributů této struktury budou nastaveny atributy projektu vzniklého importem.

Využití datových struktur rámcového tématu a projektu SSP pro generování atributů projektu vzniklého jejich importem bude popsáno v následujících sekcích textu.

Datová struktura rámcového tématu

Projektu Burzy projektů, který vznikne importem rámcového tématu, bude nastaven zadavatel podle zadavatele rámcového tématu. Dále bude nastaven předmět podle typu práce. Bakalářské práce jsou řešeny v předmětu BI-BAP, diplomové práce v předmětu MI-DIP.

Podle atributu jazyk práce bude zvolen jazyk názvu a popisu rámcového tématu, který bude použit pro nastavení odpovídajících atributů projektu. Další atributy rámcového tématu budou použity pro doplnění popisu projektu.



Obrázek 5.4: Návrh datových struktur externích projektů.

Datová struktura projektu SSP

Podle názvu, štítků a předmětů projektu SSP budou nastaveny odpovídající atributy projektu Burzy projektů. Příznak NDA bude rozhodovat, zda bude projektu Burzy projektů nastavena stejně pojmenovaná striktní vlastnost. Z dalších atributů projektu SSP bude sestaven popis projektu Burzy projektů.

Poznámka k popisu projektu

Je zřejmé, že popis projektu vzniklého importem bude sestaven z atributů importovaného projektu, které nelze namapovat na jiné atributy. Proto by mělo uživatelské rozhraní Burzy projektů umožňovat zobrazování strukturovaného popisu projektu.

Pro tento účel by mohl být použit například formát Markdown [31]. Prostý text v tomto formátu je snadno čitelný a lze ho jednoduše převádět do HTML pomocí různých knihoven a nástrojů.

Kapitola 6

Implementace

Obsahem této kapitoly je popis implementační fáze práce. Nejprve budou uvedeny ukázky vybraných použitých implementačních postupů. Dále bude vyjmenován použitý software. Na závěr bude popsána struktura softwarových projektů, které jsou produktem implementace.

6.1 Implementační postupy

V následujících sekcích textu budou popsány vybrané implementační postupy použité během této práce. U postupů budou uvedeny příklady ze zdrojových kódů. Příklady budou zjednodušené a nedůležité části budou nahrazeny výpustkou (třemi tečkami).

6.1.1 Využití IoC kontejneru

Pro usnadnění testování interakce tříd, byl použit IoC kontejner Spring Frameworku způsobem, který bude demonstrován na konkrétních návrhových třídách (viz str. 33).

Například třída *KosapiRequests* je jedna z mnoha tříd (tzv. bean třídy), které jsou určeny k tomu, aby jejich jediná instance byla za běhu programu vložena do IoC kontejneru a mohla být využívána ostatními třídami.

K takové třídě bylo nejdříve vytvořeno rozhraní definující metody, které bude třída poskytovat.

```
public interface KosapiRequests {
        Set<String> getNDARoleMembers();
        ...
}
```

Dále byla vytvořena implementace tohoto rozhraní. Implementační třída byla označena anotací, která zaručí, že kdykoli bude nalezena mechanismem

automatického vyhledávání bean tříd (component scan), bude vytvořena její instance a přidána do IoC kontejneru.

```
@Service
public class KosapiRequestsService
  implements KosapiRequests {
        Set<String> getNDARoleMembers() {
                ...
        }
        ...
}
```

V implementaci návrhové třídy *CommonLogic* je možné automaticky získat instanci třídy *KosapiRequestsService* pomocí anotace atributu. Inicializaci atributu zajistí IoC kontejner.

```
@Service
public class CommonLogicService implements CommonLogic {
    @Autowired KosapiRequests kosapiRequests;
    ...
}
```

V automatickém testu třídy *CommonLogicService* je potom vynechána třída *KosapiRequestsService* z mechanizmu component scan a místo ní je dosazena její náhrada s předefinovaným chováním. Tato náhrada je vytvořena pomocí třídy `Stub` testovacího frameworku Spock a přidána do kontejneru pomocí anotace `@Bean`.

```
@Configuration
@ContextConfiguration(classes = [
  CommonLogicTest.class, CommonLogicService.class])
class CommonLogicTest extends Specification {
    @Bean KosapiRequests kosapiRequests() {
        return Stub(KosapiRequests) {
            getNDARoleMembers() >> ["A", "B"] as Set
            ...
            }
    }
    ...
}
```

6.1.2 Daty řízené testy

Při implementaci testů byly využity možnosti daty řízených testů frameworku Spock [20]. V dalším příkladu bude uveden test aplikační logiky, která ověřuje, zda uživatel splňuje vyžadované striktní vlastnosti projektu.

```
def "can access"() {
    expect:
    logic.canAccess(submitter, new Project(
      restrictions: restrictions)) == result
    where:
    submitter | restrictions | result
    "~"       | []          | true
    "~"       | [NDA]       | false
    "N"       | [NDA]       | true
}
```

6.1.3 Dotazování na kolekce entit

Dotazování na kolekce entit prostřednictvím REST rozhraní s možností filtrování podle různých kritérií a složitějších logických výrazů bylo implementováno s pomocí dvou knihoven. Syntaktickou analýzu dotazů obstarává knihovna `rsql-parser` [23], pro vytváření podmínek vyhledávání je použito Querydsl [22].

Byl implementován tzv. visitor (návrhový vzor [32]), který postupně navštíví (zpracuje) uzly syntaktického stromu vytvořeného třídou `Parser` knihovny `rsql-parser` a na základě v nich obsažených informací sestaví pomocí prvků knihovny Querydsl výsledný dotaz.

Následuje ukázka kódu implementace třídy. Třída je definována jako abstraktní, detaily jsou dodefinovány v odvozených třídách.

```
public abstract class RsqlToQuerydslConverter extends
  NoArgRSQLVisitorAdapter<BooleanExpression> {
        ...
    private final RSQLParser parser;
    @Override
    public BooleanExpression visit(AndNode node) {
        return BooleanExpression.allOf(
          processChildren(node.getChildren()));
    }
    @Override
    public BooleanExpression visit(OrNode node) {
        return BooleanExpression.anyOf(
          processChildren(node.getChildren()));
    }
    @Override
    public BooleanExpression visit(ComparisonNode node) {
        return processOperation(node.getSelector(),
          node.getOperator(), node.getArguments());
    }
```

```
    protected BooleanExpression processQuery(String query) {
        return isNullOrBlank(query) ? null
          : parser.parse(query).accept(this);
    }
    abstract protected BooleanExpression processOperation(
      String selector, ComparisonOperator operator,
      List<String> arguments);
    ...
}
```

6.1.4 Realizace obrazovek

Základ obrazovek uživatelského rozhraní byl implementován pomocí technologie JSF v kombinaci s prvky Spring Frameworku. Detaily vzhledu a chování byly doladěny kaskádovými styly a pomocí Javascriptu.

V implementaci obrazovek je oddělena prezentační vrstva od modelu. Model je realizován bean třídami IoC kontejneru. Například model pro obrazovku seznamu projektů vypadá následovně:

```
@Controller
@Scope("view")
public class ProjectList {
        ...
        public boolean canImport() {
                ...
        }
    ...
}
```

View tvoří XHTML soubory se speciálními tagy. Ve view je možné využívat atributy a funkce modelu pomocí jazyka JSF Expression Language [33]. Následuje příklad části view pro seznam projektů. V příkladu je tag generující odkaz, který je zobrazen pouze tehdy, může-li uživatel importovat rámcová témata (rozhodne model).

```
<f:view ...>
...
        <h:commandLink
          rendered="#{projectList.canImport()}" ...>
                ...
        </h:commandLink>
...
</f:view>
```

6.1.5 Implementace webových služeb

Zástupné webové služby pro systém ZP a portál SSP byly implementovány jako jediná Spring Boot aplikace. Při vývoji byly využity zkušenosti získané vývojem serverové komponenty Burzy projektů. Aplikace udržuje databázi v paměti a poskytuje REST a SOAP rozhraní.

Pro implementaci SOAP rozhraní byl použit postup [34], který umožňuje vygenerovat pomocné třídy pro implementaci tohoto rozhraní na základě popisu služby v souboru ve formátu XML Schema [35]. Tento soubor popisuje přenášené datové typy a metody rozhraní. WSDL soubor popisující službu je generován dynamicky za běhu aplikace.

XML schéma pro SOAP službu portálu SSP a soubor popisující REST rozhraní systému ZP ve formátu RAML [36] jsou na přiloženém médiu v adresáři s doplňkovou dokumentací (viz příloha G).

6.2 Použité nástroje

Všechny hlavní softwarové produkty této práce byly vyvíjeny v jazyce Java a pro jejich vývoj byly použity odpovídající nástroje. Následuje výčet těch nejdůležitějších:

- *Maven* [37]. Tento nástroj byl použit pro správu softwarových projektů. Maven se stará o stažení závislostí (knihoven) z repozitářů, překlad tříd, automatické testování, sestavení výsledných souborů (JAR nebo WAR), lokální instalaci nebo případné nahrání výsledného produktu do vzdáleného repozitáře (tzv. deploy artefaktu).

- *Git* [38]. Git je jedna z možných realizací distribuovaného verzovacího systému. Umožňuje spravovat verze softwaru, větvit vývoj, atd. Pro vývoj byla také použita školní instance projektu GitLab [39], což je webová nadstavba nad vzdálenými Git repozitáři. Odkazy na softwarové projekty, které jsou výsledkem práce, je možné nalézt v příloze E.

- *IntelliJ IDEA* [40]. Tento program je Java IDE (integrated development environment), tedy prostředí pro vývoj Java aplikací. Toto IDE má mimo jiné vestavěnou podporu nástroje Maven, Spring Frameworku nebo JSF.

- *Chromium* [41]. Pro ladění uživatelského rozhraní byly použity vestavěné vývojové nástroje webového prohlížeče Chromium.

6.3 Struktura softwarových projektů

Výsledkem této práce je několik samostatných softwarových projektů. Projekty je možné najít v adresáři se zdrojovými kódy implementace na přiloženém médiu (viz příloha G). Následuje výčet projektů a popis jejich struktury:

- *kosapi-client.* Tento projekt realizuje klientskou knihovnu pro dotazování se do informačních systémů KOS a Usermap.
- *projex-service-modules.* Tento projekt obsahuje implementaci serverové komponenty systému Burza projektů a několika dalších produktů. Skládá se z těchto částí:
  - *projex-common.* Tento modul (označení pro podprojekty Maven projektu) realizuje knihovnu sdílenou mezi komponentami systému.
  - *projex-service.* Modul je implementací serverové komponenty. Výsledkem sestavení tohoto modulu je Spring Boot aplikace.
  - *projex-client.* Modul je knihovna, kterou mohou využít klientské aplikace pro dotazování se na REST rozhraní serverové komponenty.
  - *projex-data-generator.* Tento modul je aplikace, která využívá knihovnu *projex-client* pro generování projektů a přihlášek.
  - *shell-client.* Pro účely testování REST rozhraní byl vytvořen jednoduchý klientský shell skript.
- *projex-frontend-modules.* Tento projekt obsahuje moduly, které realizují uživatelské rozhraní Burzy projektů:
  - *projex-frontend-common.* Modul je knihovnou, kterou je možné sdílet mezi různými implementacemi uživatelského rozhraní.
  - *projex-ssp-frontend.* Tento modul je implementací uživatelského rozhraní systému jako portletu pro Liferay portál.
- *projex-stubs.* V tomto projektu je obsažena implementace zástupných webových služeb pro systém ZP a portál SSP. Obsahuje tyto části:
  - *projex-stub-schema.* Modul obsahuje XML schéma pro SOAP službu portálu SSP. Výsledkem překladu projektu je knihovna obsahující třídy vygenerované pomocí schématu.
  - *projex-stub-services.* Modul je implementací zástupných webových služeb systému ZP a portálu SSP. Jeho sestavením vznikne Spring Boot aplikace.
  - *projex-stub-clients.* Tento modul realizuje knihovnu, která obsahuje klientské třídy pro webové služby realizované modulem *projex-stub-services*.
  - *shell-client.* Pro účely testování zástupných webových služeb a manipulaci s rámcovými tématy a projekty SSP, které jsou službami nabízeny, byl vytvořen klientský shell skript.

Kapitola 7

Testování a dokumentace

Aby mohly být softwarové produkty této práce předány, musely být otestovány a zdokumentovány. Zmíněným činnostem se věnuje tato kapitola. Postupně budou popsány provedené automatické testy, vytvořené testovací nástroje a testy uživatelského rozhraní. V závěru kapitoly bude shrnuta vytvořená dokumentace.

Testování softwaru může prokázat, že software obsahuje chyby, nemůže ale prokázat, že je software bez chyb [7]. Proto bylo záměrem testování pokrýt testy co největší množství důležitého kódu, otestovat důležité třídy (především třídy realizující doménovou logiku), jejich metody a jejich interakci a otestovat případy užití výsledného softwaru pomocí všech rozhraní.

7.1 Automatické testy

Automatické testy slouží především k průběžnému automatickému testování částí softwaru při vývoji. K jejich implementaci byl použit projekt Spock Framework, jehož výhody byly rozebrány v analýze (str. 15).

Implementace testů jsou v softwarových projektech, které jsou produktem této práce, umístěny v podadresáři `src/test/groovy` a je možné je spustit a ověřit jejich výsledek pomocí nástroje Maven následujícím příkazem vykonaným v adresáři projektu:

```
mvn clean test
```

Následuje výčet druhů provedených testů.

7.1.1 Jednotkové testy

Jednotkové testy se používají pro automatické testování izolovaných tříd. V implementaci Burzy projektů jsou jimi pokryty důležité pomocné třídy. Například třída *RsqlQueryBuilder* z projektu *kosapi-client*, která je použita pro usnadnění vytváření RSQL dotazů.

7.1.2 Integrační testy

Testy interakce tříd patří mezi integrační testy. Byla otestována především interakce tříd realizujících doménovou logiku systému Burza projektů. Využité postupy byly popsány v kapitole implementace (str. 47).

Bylo také otestováno dotazování do databáze systému Burza projektů nad databází v paměti. Každý test byl prováděn jako samostatná databázová transakce (z pohledu změn databáze izolovaně od ostatních testů) končící příkazem *rollback* (zrušením transakce).

7.1.3 Systémové testy

Serverová komponenta systému Burza projektů byla otestována jako tzv. „Black Box“ (jako černá skříňka) prostřednictvím REST rozhraní. Stejně tak byla otestována rozhraní zástupných webových služeb pro systém ZP a portál SSP.

7.2 Testovací nástroje

Pro další testování byly vytvořeny pomocné nástroje. Pro každou službu (serverovou komponentu a zástupné webové služby systému ZP a portálu SSP) byl vytvořen klientský shell skript. Skripty byly použity pro manuální testování REST a SOAP rozhraní služeb.

S využitím klientské knihovny serverové komponenty Burzy projektů byl vytvořen generátor dat, který je možné použít pro naplnění Burzy projektů větším objemem testovacích dat přes REST rozhraní. Dotazy generované tímto nástrojem otestovaly REST rozhraní, generovaná data byla využita především pro testování prezentace dat uživatelským rozhraním.

Popis nástrojů je možné nalézt v příloze D.

7.3 Testování uživatelského rozhraní

Uživatelské rozhraní bylo testováno pouze manuálně autorem práce. Byly testovány různé průchody obrazovkami za účelem realizace případů užití.

7.4 Dokumentace

K softwarovým produktům této práce byla vytvořena následující dokumentace:

- *Základní dokumentace.* Jako základní zdroj informací nechť slouží text této práce.
- *Příručka pro zprovoznění.* Byl vytvořen návod pro zprovoznění systému, který je možné najít v příloze B.

- *Popis uživatelského rozhraní.* Obrazovky uživatelského rozhraní a jejich ovládání bylo popsáno v příloze C.
- *Popis nástrojů.* K vytvořeným pomocným nástrojům byla vytvořena samostatná dokumentace. Nachází se v příloze D.
- *Specifikace rozhraní služeb.* Popisy rozhraní všech služeb byly umístěny na přiložené médium do adresáře s doplňkovou dokumentací (viz příloha G). Popisy REST rozhraní jsou ve formátu RAML, SOAP rozhraní je popsáno formátem XML Schema.

Závěr

Na začátku práce byl analyzován stav systému Burza projektů. Byla zjištěna nekompatibilita komponent a identifikovány závažné nedostatky implementace. Na základě těchto informací byly definovány požadavky na dokončení základních funkcí systému.

Bylo navrženo a implementováno řešení. Byl přepracován původní špatný návrh serverové části systému, odstraněny překážky pro tvorbu testů a vývoj rozšíření a zlepšena podpora uživatelského rozhraní. Implementace byla pokryta automatickými testy.

Dále byly nastudovány potřebné technologie a přepracováno uživatelské rozhraní tak, aby byla zajištěna kompatibilita se serverovou komponentou a bylo vhodným způsobem využíváno její rozhraní. Obrazovky uživatelského rozhraní byly upraveny a rozšířeny (přidány ovládací prvky atd.).

Poté, co byla odstraněna většina nedostatků původní implementace systému, byly analyzovány požadavky na rozšíření systému. Analýza byla na základě zadání práce a původního záměru systému zaměřena na rozšíření nabídky projektů o rámcová témata systému ZP a projekty portálu SSP.

Na základě analýzy těchto systémů a požadavků jejich vývojářů byl navržen způsob integrace. Byly navrženy a implementovány zástupné webové služby pro oba systémy a byla provedena integrace těchto služeb.

Systém Burza projektů byl zdokumentován a připraven na předání. Před nasazením systému bude nutné dokončit integraci nabídky projektů okolních systémů na straně těchto systémů (implementovat reálné služby místo zástupných). Také bude nutné pokračovat ve vývoji uživatelského rozhraní, neboť z časových důvodů nebyly implementovány všechny jeho zamýšlené funkce (viz příloha C).

Vedlejším produktem této práce je Java knihovna pro získávání dat z fakultních informačních systémů KOS a Usermap.

Literatura

[1] CODEVELS. *Portál spolupráce s průmyslem: SSP Form* [online]. Praha: FIT ČVUT, 2014 [cit. 28. 11. 2015]. Dostupný z: `https://ssp.test.fit.cvut.cz/`

[2] LARMAN, Craig. *Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Second edition. Foreword by Philippe Kruchten. New Jersey: Prentice Hall, [2002]. Dostupné také z: `https://www.utdallas.edu/~chung/SP/applying-uml-and-patterns.pdf`

[3] THE POSTGRESQL GLOBAL DEVELOPEMENT GROUP. *PostgreSQL* [software]. [přístup 2. 2. 2016]. Dostupné z: `http://www.postgresql.org/`

[4] PIVOTAL SOFTWARE. *Spring Framework* [software]. [přístup 2. 2. 2016]. Dostupné z: `https://projects.spring.io/spring-framework/`

[5] JIRŮTKA, Jakub. *KOSapi: Hlavní stránka* [online]. Praha: FIT ČVUT, aktualizováno 2015-01-22 16:27 [cit. 2. 2. 2016]. Dostupné z: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`

[6] *Umapi API documentation version v1* [online]. Praha: FIT ČVUT, [cit. 2. 2. 2016]. Dostupné z: `https://kosapi.fit.cvut.cz/usermap/doc/rest-api-v1.html`

[7] MLEJNEK, Jiří. *Přednášky k předmětu Softwarové inženýrství 1*. Praha: FIT ČVUT, Letní semestr 2014/2015.

[8] GUIZZARDI, Giancarlo. *Ontological Foundations for Structural Conceptual Models*. Enschede, The Netherlands: Telematica Instituut, 2005. ISBN 90-75176-81-3. Dostupné také z: `http://www.inf.ufes.br/~gguizzardi/OFSCM.pdf`

[9] BUCHTELA, David. *Přednášky k předmětu Objektové modelování*. Praha: FIT ČVUT, Zimní semestr 2013/2014.

[10] FIT ČVUT – ODDĚLENÍ PRO ROZVOJ. *Dohoda o mlčenlivosti a souhlasu s podmínkami používání portálu SSP-FIT* [online]. Praha: FIT ČVUT, [cit. 27. 4. 2016]. Dostupné z: https://wiki.cvut.cz/confluence/download/attachments/14057588/NDA-SSP.pdf

[11] *The Java EE 5 Tutorial: Entities* [online]. Oracle and/or its affiliates, 2010 [cit. 2. 2. 2016]. Dostupné z: http://docs.oracle.com/javaee/5/tutorial/doc/bnbqa.html

[12] PIVOTAL SOFTWARE. *Spring Boot* [software]. [přístup 2. 2. 2016]. Dostupné z: http://projects.spring.io/spring-boot/

[13] WEBB, Phillip, et al. Build tool plugins: Spring Boot Maven plugin. In: *Spring Boot Reference Guide* [online]. [cit. 2. 2. 2016]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/html/build-tool-plugins-maven-plugin.html

[14] JOHNSON, Rod, et al. Core Technologies: The IoC container. In: *Spring Framework Reference Documentation* [online]. [cit. 2. 2. 2016]. Dostupné z: http://docs.spring.io/autorepo/docs/spring/3.2.x/spring-framework-reference/html/beans.html

[15] PIVOTAL SOFTWARE. *Spring Data JPA* [software]. [přístup 2. 2. 2016]. Dostupné z: http://projects.spring.io/spring-data-jpa/

[16] GIERKE Oliver. *Advanced Spring Data JPA - Specifications and Querydsl* [online]. april 26, 2011 [cit. 27. 4. 2016]. Dostupné z: https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl

[17] JIRŮTKA, Jakub. Issue #7: Nutný refactoring třídy Domain. In: *GitLab: Probur / ProBur Sources* [online]. [cit. 17. 1. 2016]. Dostupné z: https://gitlab.fit.cvut.cz/probur/probur/issues/7

[18] THE SPOCK FRAMEWORK TEAM. *Spock Framework* [software]. [přístup 27. 4. 2016]. Dostupné z: https://github.com/spockframework/spock

[19] NIEDERWIESER, Peter. Interaction Based Testing. In: *Spock Framework Reference Documentation* [online]. Last updated 2015-03-02 11:42:41 UTC [cit. 27. 4. 2016]. Dostupné z: http://spockframework.github.io/spock/docs/1.0/interaction_based_testing.html

[20] NIEDERWIESER, Peter. Data Driven Testing. In: *Spock Framework Reference Documentation* [online]. Last updated 2015-03-02 11:42:41 UTC [cit. 27. 4. 2016]. Dostupné z: http://spockframework.github.io/spock/docs/1.0/data_driven_testing.html

[21] WEBB, Phillip, et al. Spring Boot features: Testing. In: *Spring Boot Reference Guide* [online]. [cit. 27. 4. 2016]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html

[22] THE QUERYDSL TEAM. *Querydsl* [software]. [přístup 27. 4. 2016]. Dostupné z: http://www.querydsl.com/

[23] JIRŮTKA, Jakub. *RSQL / FIQL parser* [software]. [přístup 27. 4. 2016]. Dostupné z: https://github.com/jirutka/rsql-parser

[24] JIRŮTKA, Jakub. *KOSapi: Vyhledávání* [online]. Praha: FIT ČVUT, aktualizováno 2013-04-21 16:05 [cit. 2. 2. 2016]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/Query

[25] LIFERAY INC. *Liferay Portal* [software]. [přístup 27. 4. 2016]. Dostupné z: https://web.liferay.com/community/liferay-projects/liferay-portal/overview

[26] TVRDÍK, Pavel. *Směrnice děkana FIT ČVUT č. 14/2015 pro závěrečné práce a státní závěrečné zkoušky na Fakultě informačních technologií Českého vysokého učení technického v Praze* [online]. Praha: FIT ČVUT, 16. 11. 2015 [cit. 28. 11. 2015]. Dostupné z: https://fit.cvut.cz/sites/default/files/STO/SmerniceDekana%20SZZ%2014-2015_3.pdf

[27] 1 - Vytvoření, rezervace a schvalování ZP. In: *PROCESNÍ PORTÁL* [online]. [cit. 28. 11. 2015]. Dostupné z: https://www.fel.cvut.cz/procesy/element.jsf?qprid=413295872&model=16713

[28] GUDGIN, Martin, et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)* [online]. W3C, 27 April 2007 [cit. 27. 4. 2016]. Dostupné z: https://www.w3.org/TR/soap12/

[29] MASSÉ, Mark. *REST API Design Rulebook*. Sebastopol: O'Reilly Media, Inc., 2012. ISBN: 978-1-449-31050-9.

[30] JIRŮTKA, Jakub. *KOSapi: Plány do budoucna* [online]. Praha: FIT ČVUT, aktualizováno 2014-02-26 21:26 [cit. 2. 2. 2016]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/Plans

[31] GRUBER, John. *Markdown* [online]. [cit. 27. 4. 2016]. Dostupné z: http://daringfireball.net/projects/markdown/

[32] PECINOVSKÝ, Rudolf. *Návrhové vzory*. Dotisk 1. vydání. Brno: Computer Press a.s., 2013. ISBN 978-80-251-1582-4.

[33] *The Java EE 6 Tutorial: Chapter 6 Expression Language* [online]. Oracle and/or its affiliates, 2010 [cit. 2. 2. 2016]. Dostupné z: `http://docs.oracle.com/javaee/6/tutorial/doc/gjddd.html`

[34] PIVOTAL SOFTWARE. *Getting Started · Producing a SOAP web service* [online]. [cit. 27. 4. 2016]. Dostupné z: `https://spring.io/guides/gs/producing-web-service/`

[35] XML SCHEMA WORKING GROUP. *XML Schema* [online]. [cit. 27. 4. 2016]. Dostupné z: `https://www.w3.org/XML/Schema`

[36] *RAML^{TM} Version 0.8: RESTful API Modeling Language* [online]. [cit. 27. 4. 2016]. Dostupné z: `https://github.com/raml-org/raml-spec/blob/master/raml-0.8.md`

[37] THE APACHE SOFTWARE FOUNDATION. *Maven* [software]. [přístup 27. 4. 2016]. Dostupné z: `https://maven.apache.org/download.cgi`

[38] SOFTWARE FREEDOM CONSERVANCY. *Git* [software]. [přístup 27. 4. 2016]. Dostupné z: `https://git-scm.com/downloads`

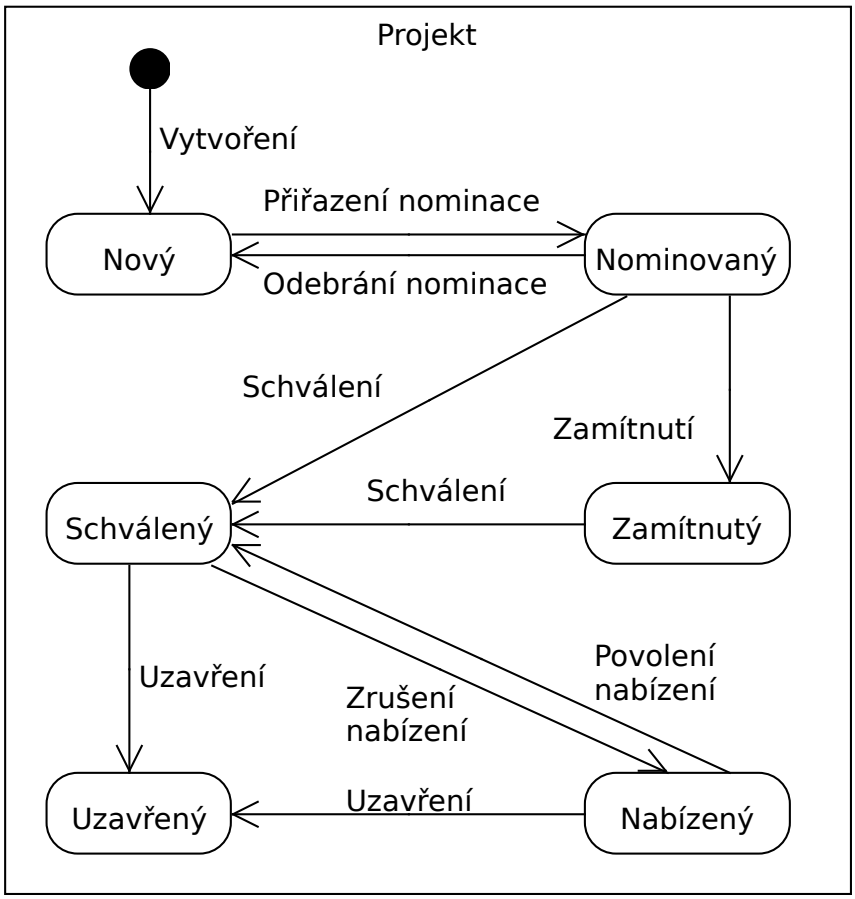
[39] GITLAB INC. *GitLab* [software]. [přístup 27. 4. 2016]. Dostupné z: `https://about.gitlab.com/downloads/`

[40] JET BRAINS. *IntelliJ IDEA* [software]. [přístup 27. 4. 2016]. Dostupné z: `https://www.jetbrains.com/idea/`

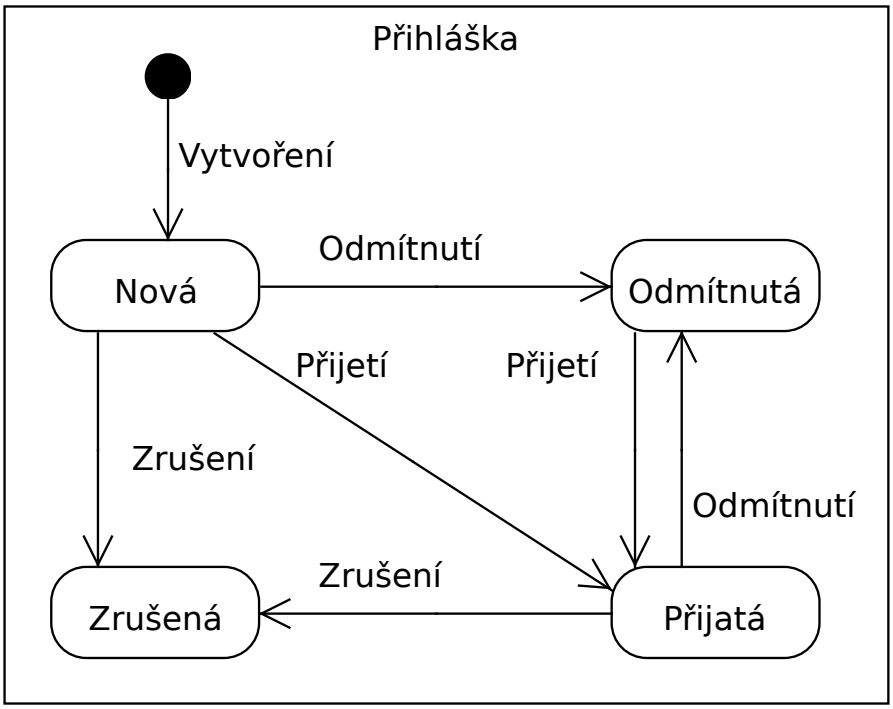
[41] GOOGLE. *Chromium* [software]. [přístup 27. 4. 2016]. Dostupné z: `https://www.chromium.org/getting-involved/download-chromium`

Příloha **A**

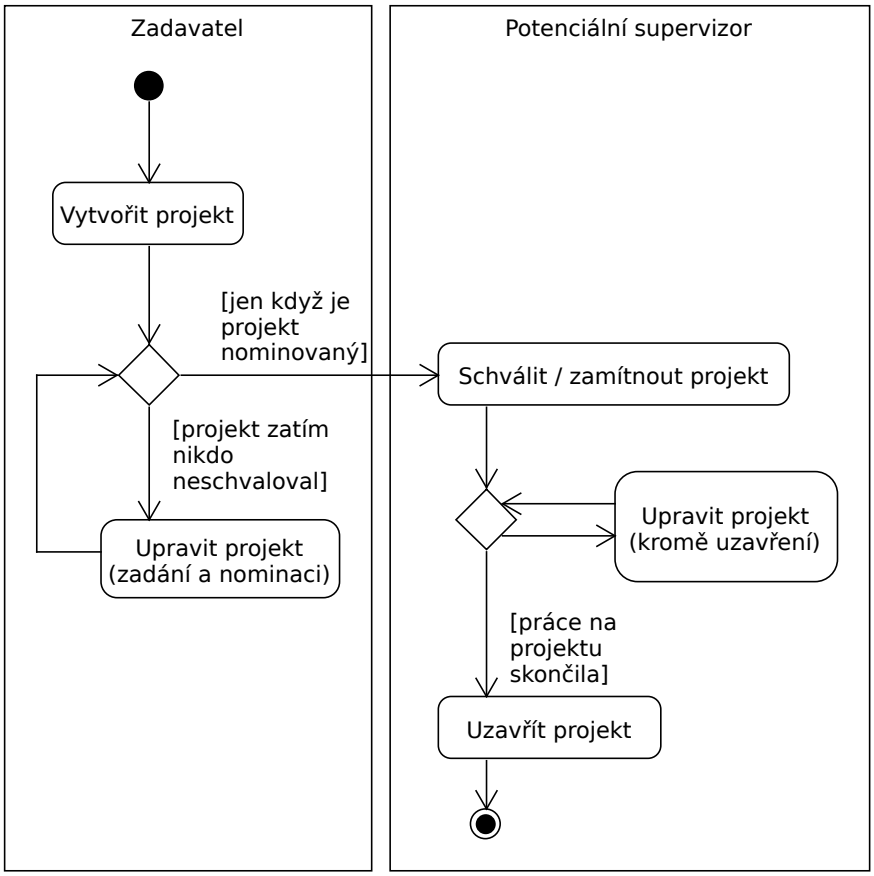
Doplňky textu



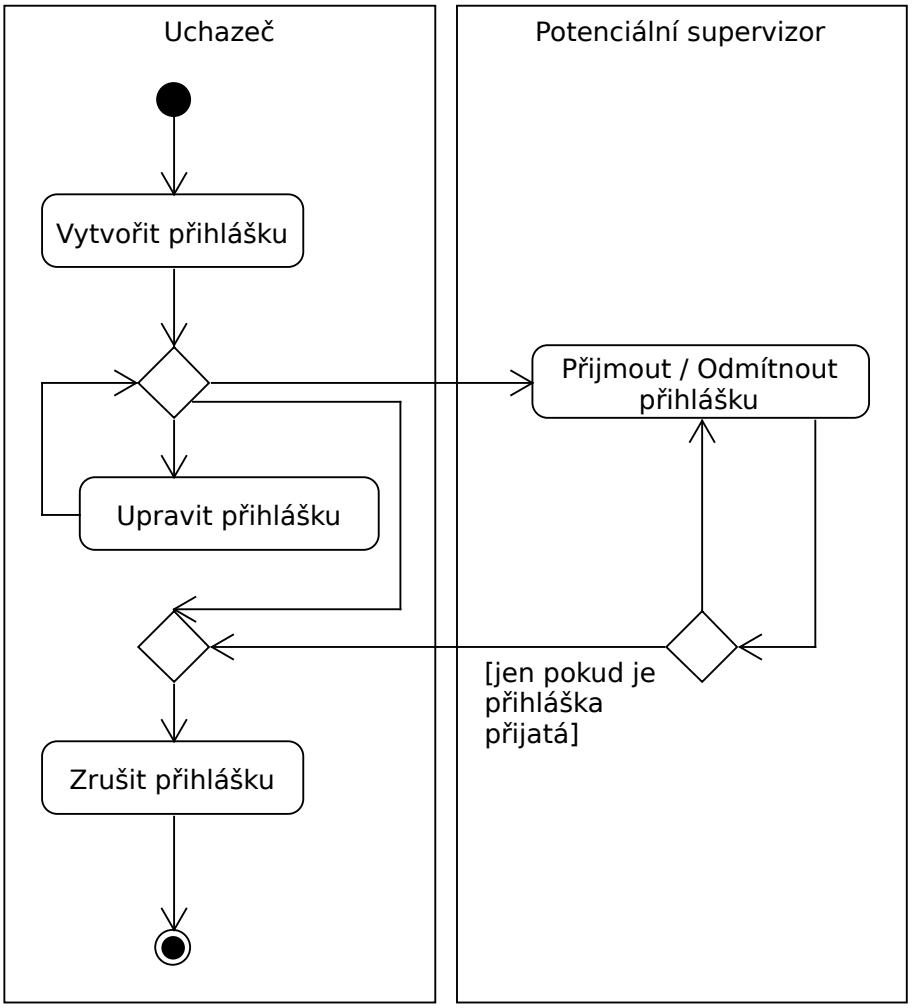
Obrázek A.1: Stavový diagram projektu Burzy projektů.



Obrázek A.2: Stavový diagram přihlášky Burzy projektů.



Obrázek A.3: Proces zpracování projektu v Burze projektů.



Obrázek A.4: Proces zpracování přihlášky v Burze projektů.

Příloha B

Příručka pro zprovoznění

Tato příloha obsahuje pokyny pro zprovoznění softwarových produktů této práce za účelem ověření jejich funkčnosti. Postupně bude popsán postup zprovoznění následujících části:

- Aplikace realizující zástupné webové služby systému ZP a portálu SSP.
- Serverové aplikace Burzy projektů.
- Uživatelského rozhraní Burzy projektů.

Příručka předpokládá zprovoznění všech částí na jednom počítači s operačním systémem Linux (testováno na Ubuntu 14.04.4 LTS 64b). Dále je vyžadováno internetové připojení (pro komunikaci s fakultními informačními systémy) a nainstalovaný software Java Runtime Environment verze 7 (testováno s OpenJDK a verzí Javy 1.7.0_101).

Pro zprovoznění serverové aplikace Burzy projektů je dále nutné mít přístup k PostgreSQL databázi, kterou může aplikace využít pro uložení svých dat. Tato databáze nemusí být žádným způsobem inicializována. Odkaz na manuál zprovoznění systému PostgreSQL a vytvoření databáze na systému Ubuntu je možné nalézt v příloze E.

Uživatelské rozhraní bylo vyvíjeno pro nasazení na portál Liferay verze 6.1.2-ce-ga3 s vestavěným Tomcat serverem. Odkaz pro stažení tohoto softwaru je opět v příloze E.

Všechny soubory potřebné pro zprovoznění softwarových produktů této práce, jsou umístěny v jediném adresáři na přiloženém médiu (viz obsah CD v příloze G). Příkazy uvedené v této příručce spouštějte ve zmíněném adresáři.

V dalším testu jsou předpokládány čtenářovy základní znalosti linuxové příkazové řádky.

Zprovoznění služeb

Pro spuštění aplikace realizující zástupné služby systému ZP a portálu SSP použijte následující příkaz:

```
java -jar stub-web-services.jar --server.port=10001
```

Služby budou dostupné na portu 10001. V případě potřeby změňte hodnotu parametru. Aplikaci je možné naplnit testovacími daty tímto příkazem (případně změňte hodnotu proměnné port):

```
port=10001 ./swsvc-cli.sh update
```

Pro upravení testovacích dat a kompletní manuál použitého nástroje čtěte přílohu D.

Serverovou aplikaci Burzy projektů lze spustit následujícím příkazem:

```
java -jar projex-service.jar \
  --projex.import_cron='* */5 * * * *' \
  --server.port=8088 \
  --stub_services.base_uri=http://localhost:10001 \
  --spring.datasource.url=jdbc:postgresql://localhost/projex \
  --spring.datasource.username=postgres \
  --spring.datasource.password=postgres
```

Prvním parametrem (za jar souborem) je nastavena perioda automatického importu a aktualizace projektů okolních systémů (zástupných služeb). Hodnota má formát linuxového programu cron. Druhý parametr nastaví port, na kterém bude služba dostupná. Hodnota třetího parametru odpovídá adrese (vřetně portu), na které jsou dostupné zástupné služby.

Poslední tři parametry příkazu nastavují připojení k databázi. Příkaz předpokládá lokální databázi PostgreSQL, která má název „projex“ a přihlašovací jméno a heslo „postgres“.

REST rozhraní Burzy projektů může být otestováno nástrojem popsaným v příloze D.

Pro testování funkčnosti uživatelského rozhraní Burzy projektů (postup zprovoznění bude uveden později) je vhodné automaticky vygenerovat větší množství projektů a přihlášek. K tomuto účelu může být použita následující posloupnost příkazů (jejich vykonání může trvat delší dobu):

```
java -jar projex-data-generator.jar \
  --semesters=B151,B152 \
  --courses=BI-PA1,BI-PA2,BI-PSI,BI-APS,BI-BEZ,BI-OSY \
  --nominationOnly \
  --nominationOut=nomination.json
```

```
java -jar projex-data-generator.jar \
  --projex.base_uri=http://localhost:8088 \
  --nominationIn=nomination.json \
  --projectsCount=100 --maxEnrollments=20
```

První příkaz generuje konfigurační soubor pro druhý příkaz, který vykonává dotazy na REST rozhraní Burzy projektů. Pro hlubší porozumění použitému nástroji čtěte přílohu D. Za povšimnutí stojí první argument (za jar souborem) druhého příkazu, který nastavuje adresu (včetně portu), na které je dostupná serverová aplikace Burzy projektů.

Zprovoznění uživatelského rozhraní

Pro zprovoznění uživatelského rozhraní Burzy projektů proveďte následující kroky přesně v pořadí, ve kterém jsou zapsány:

1. Stáhněte software Liferay Portal verze 6.1.2-ce-ga3 s vestavěným Tomcat serverem pomocí odkazu v příloze E a rozbalte obsah staženého archivu. Text dále předpokládá, že proměnná `$liferay` obsahuje absolutní cestu k adresáři, který vznikl po rozbalení archivu.

2. V souboru `$liferay/tomcat-7.0.40/conf/catalina.properties` nastavte tuto proměnnou:

   ```
   shared.loader=${catalina.base}/conf/portlet-conf
   ```

   Zkopírujte konfiguraci portletu uživatelského rozhraní těmito příkazy:

   ```
   mkdir $liferay/tomcat-7.0.40/conf/portlet-conf
   cp projex-ssp-frontend.properties \
     $liferay/tomcat-7.0.40/conf/portlet-conf/
   ```

3. Spusťte Liferay portál následujícím příkazem. Před spuštěním je dobré se ujistit, že na počítači není žádnou službou využíván port 8080.

   ```
   $liferay/tomcat-7.0.40/bin/catalina.sh run
   ```

   Vyčkejte, dokud se portál nespustí a neotevře se okno webového prohlížeče. Při prvním spuštění je nutné projít v prohlížeči základní konfigurací portálu. Při této konfiguraci je nastaven název portálu, jazyk zobrazovaných stránek (zvolte češtinu) a jméno, příjmení a email uživatele. Dále je nutné odsouhlasit licenci, nastavit heslo, kontrolní otázku a odpověď. Poté se uživatel dostane na hlavní stránku portálu.

4. V tento okamžik je vhodné provést nasazení (deploy) grafického tématu SSP a portletu Burzy projektů:

```
cp fit-ssp-theme.war \
  projex-ssp-frontend.war \
  $liferay/deploy/
```

Po provedení tohoto příkazu vyčkejte, dokud se nezastaví výpis v terminálu, kde byl spuštěn Liferay portál.

5. Na hlavní stránce Liferay portálu (kde skončil bod 3) zvolte vpravo nahoře možnost „Přejít na“ a „Ovládací panel“. V levé části obrazovky se zobrazí menu. Zde v sekci „Portál“ klikněte na „Weby“. Nad seznamem webů, který se zobrazí v hlavní části stránky, zvolte „Přidat“ a „Intranet Site“. Nový web pojmenujte „intranet“ a stiskněte tlačítko „Uložit v pravé části obrazovky“.

   V menu u levého okraje obrazovky v sekci, která má v nadpisu rozbalovací menu, ve kterém by nyní měla být zvolena hodnota „intranet“ (pokud není zvolena, zvolte ji), klikněte na „Stránky“ a v horní části stránky, která se zobrazí v hlavní části okna, zvolte záložku „Soukromé stránky“. V liště pod zvolenou záložkou najděte a stiskněte tlačítko „Import“.

   V zobrazeném dialogu stiskněte „Browse...“ a vyberte soubor `liferay-intranet.lar`, který se nachází v adresáři s ostatními soubory pro zprovoznění systému. Volbu potvrďte tlačítkem „Import“.

6. Na stránce zcela vpravo nahoře klikněte na jméno uživatele. Zobrazí se detail účtu. Zde změňte uživatelské jméno na fakultní login osoby, jejímž jménem chcete testovat uživatelské rozhraní. Nový údaj potvrďte stiskem tlačítka „Uložit“. Pro uplatnění změny je nutné se odhlásit a znovu přihlásit do Liferay portálu. Tento bod opakujte, kdykoli budete chtít změnit uživatele Burzy projektů.

7. Zadejte do adresního řádku prohlížeče adresu `http://localhost:8080/group/intranet/projex/project-list`. Zobrazí se úvodní obrazovka uživatelského rozhraní Burzy projektů. Přejděte ve čtení k popisu uživatelského rozhraní v příloze C.

Příloha C

Popis uživatelského rozhraní

V této příloze bude popsáno uživatelské rozhraní Burzy projektů. Postupně bude vysvětlena funkce jednotlivých obrazovek a možnosti přechodů mezi nimi.

Upozornění: Implementace uživatelského rozhraní není kompletní. Pro využití všech funkcí Burzy projektů použijte její REST rozhraní. Nástroj pro usnadnění použití REST rozhraní je popsán v příloze D.

Seznam projektů

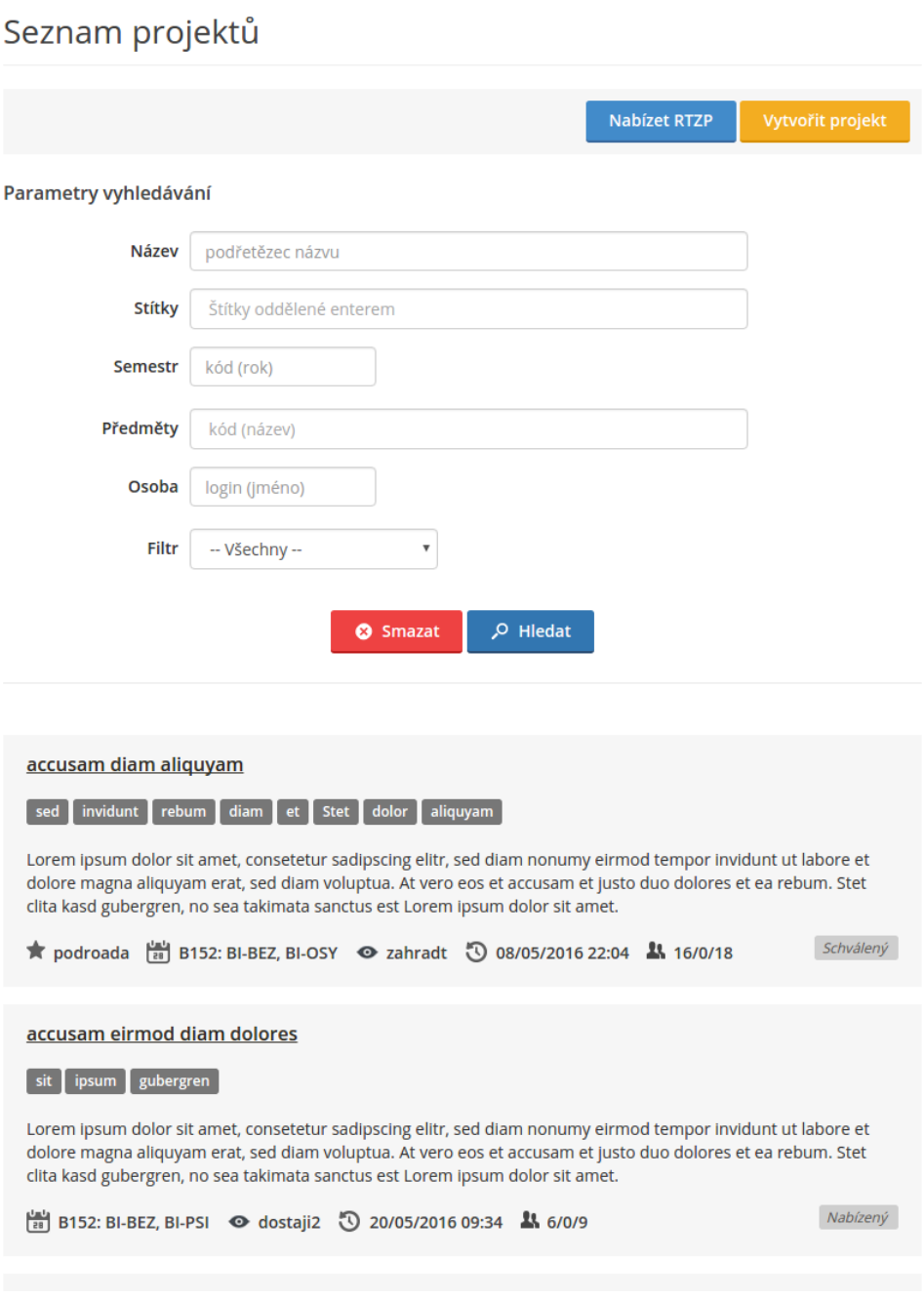
Obrazovka seznamu projektů je úvodní obrazovkou uživatelského rozhraní. Její primární funkcí je zobrazení seznamu projektů a umožnění jeho filtrování podle různých kritérií. Snímek obrazovky je na obrázku C.1. Obrazovka bude postupně popsána shora dolů.

Hlavička

Horní okraj obrazovky tvoří nadpis „Seznam projektů“ a pod ním se nachází oblast tlačítek. Tlačítko „Nabízet RTZP“ otevře modální okno pro správu nabízení rámcových témat (viz dále). Toto tlačítko se zobrazí pouze uživateli, který může v systému ZP nabízet rámcová témata. Tlačítko „Vytvořit projekt“ slouží pro přechod na obrazovku vytvoření projektu (viz dále).

Formulář pro vyhledávání

Formulář pro vyhledávání projektů je uvozen nadpisem „Parametry vyhledávání“, pokračuje jednotlivými parametry vyhledávání a končí tlačítky „Smazat“ a „Hledat“.



Obrázek C.1: Snímek obrazovky seznamu projektů.

Parametry vyhledávání jsou tyto:

- *Název.* Parametr určuje podřetězec názvu hledaných projektů (nezáleží na velikosti písmen).
- *Štítky.* Hodnotou parametru může být seznam štítků, z nichž alespoň jedním musí být hledané projekty označeny.
- *Semestr.* Filtruje projekty podle zvoleného semestru.
- *Předměty.* Parametr umožňuje zadat předměty, z nichž alespoň jeden musí být obsažen v nominaci hledaných předmětů.
- *Osoba.* Je-li tento parametr vyplněn, budou zobrazeny pouze předměty, jejichž zadavatel, supervizor, nebo uchazeč má fakultní login odpovídající hodnotě parametru.
- *Filtr.* Z nabídky hodnot parametru je možné vybrat stav, ve kterém se musí nacházet hledané projekty.

Pro některé parametry vyhledávání je implementováno tzv. našeptávání. V průběhu vyplňování hodnoty parametru jsou zobrazovány návrhy doplnění. Pro vyvolání našeptávání je možné zadat jiný typ hodnoty, než vyžaduje parametr. Například u osoby je možné začít psát jméno místo loginu a po zvolení položky našeptávání je doplněna správná hodnota. Zadávané štítky je nutné oddělit stiskem klávesy `enter`.

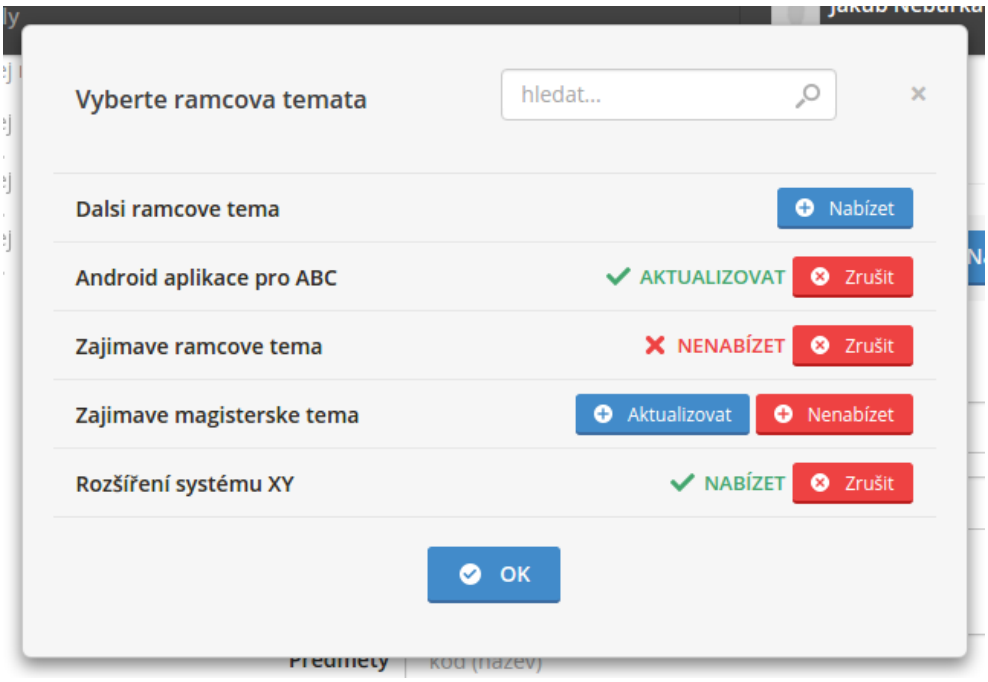
Seznam projektů (viz dále) uživatel zobrazí nebo aktualizuje stiskem tlačítka „Hledat“. Pokud jsou vyplněny nějaké parametry vyhledávání, uplatní se při filtraci zobrazeného seznamu. Pro smazání parametrů vyhledávání a seznamu projektů může uživatel použít tlačítko „Smazat“.

Seznam projektů

Poslední část obrazovky tvoří samotný seznam projektů. Seznam je zobrazovaný po částech. Pokud je možné načíst další projekty, je na konci seznamu zobrazeno tlačítko „Další“ (není na obrázku). V seznamu jsou projekty seřazeny podle názvu.

Pro každý projekt v seznamu jsou zobrazeny položky: název, štítky, úvodní část popisu (první odstavec do max. délky 1000 znaků), zadavatel, nominace, supervizor, datum konce přihlašování, kapacita (formát: řešitelé / nové přihlášky / kapacita projektu) a stav. Zadavatel je zobrazen, pokud se liší od supervizora. Nominace a supervizorské atributy jsou zobrazeny jen tehdy, jsou-li u projektu vyplněny.

Po kliknutí na název projektu se uživatel dostane na obrazovku s detailem projektu (viz dále).



Obrázek C.2: Snímek modálního okna pro správu nabídky rámcových témat.

Správa nabídky rámcovách témat

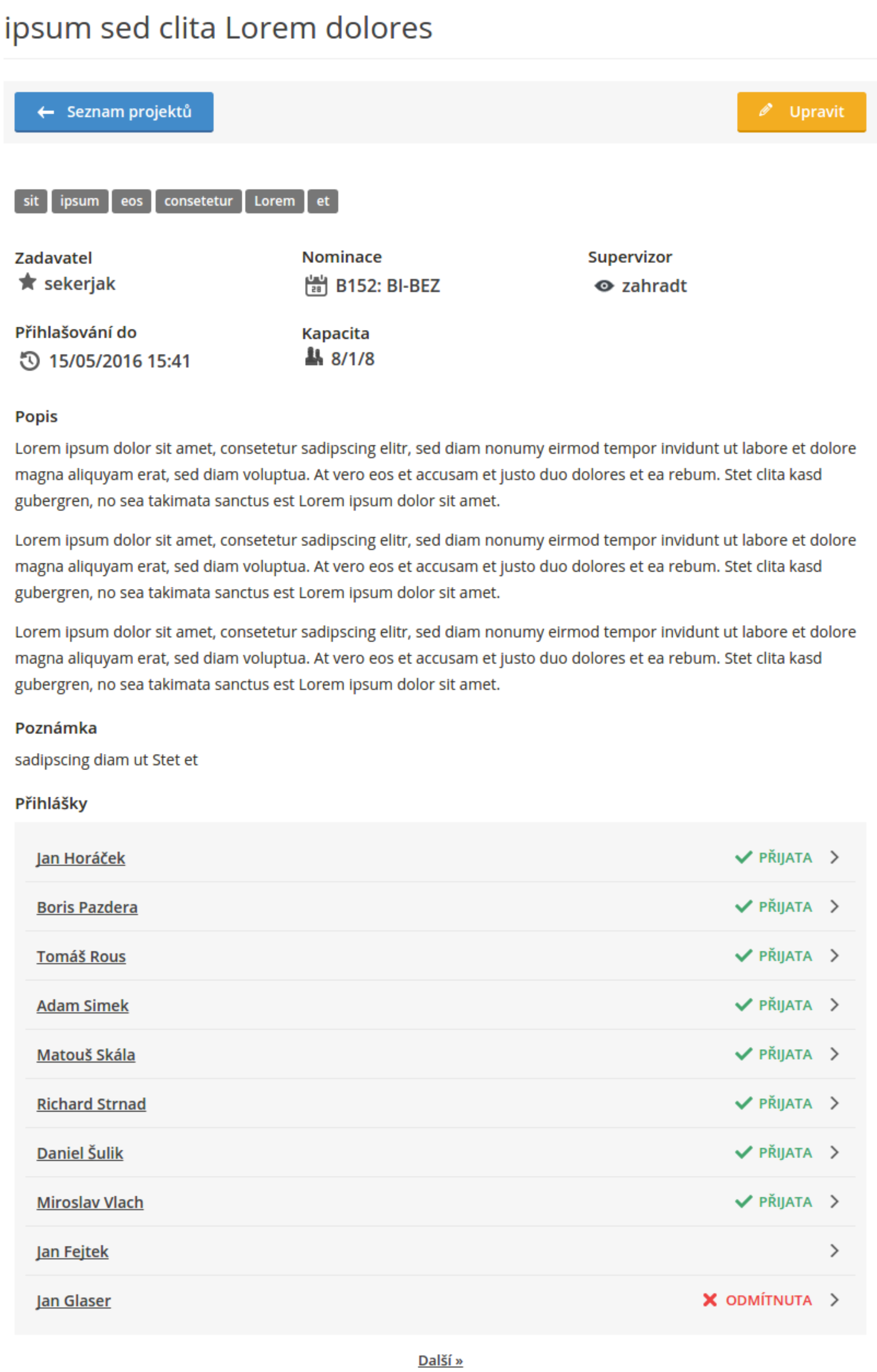
Po stisknutí tlačítka „Nabízet RTZP“ na obrazovce seznamu projektů se zobrazí modální okno pro správu nabízených rámcových témat. Snímek tohoto okna je na obrázku C.2.

Okno zobrazuje seznam rámcových témat zadaných a nabízených uživatelem v systému ZP. Pro každé rámcové téma jsou podle jeho stavu zobrazena tlačítka s možnými akcemi. Nenabízené rámcové téma je možné začít nabízet. Nabízené rámcové téma je možné aktualizovat, nebo přestat nabízet. Každou zvolenou akci lze zrušit. Zvolené akce může uživatel vykonat stiskem tlačítka „OK“ pod seznamem rámcových témat. Tlačítko zavře okno.

V hlavičce okna je textový vstup pro filtraci rámcových témat podřetězcem názvu (velikost písmen je ignorována). V pravém horním rohu okna se nachází křížek pro zavření okna bez provedení akcí.

Detail projektu

Úkolem obrazovky detailu projektu je zobrazovat detailní informace o konkrétním projektu. Zobrazuje také přihlášky na projekt. Obrazovku je možné vyvolat kliknutím na název projektu na obrazovce seznamu projektů. Snímek

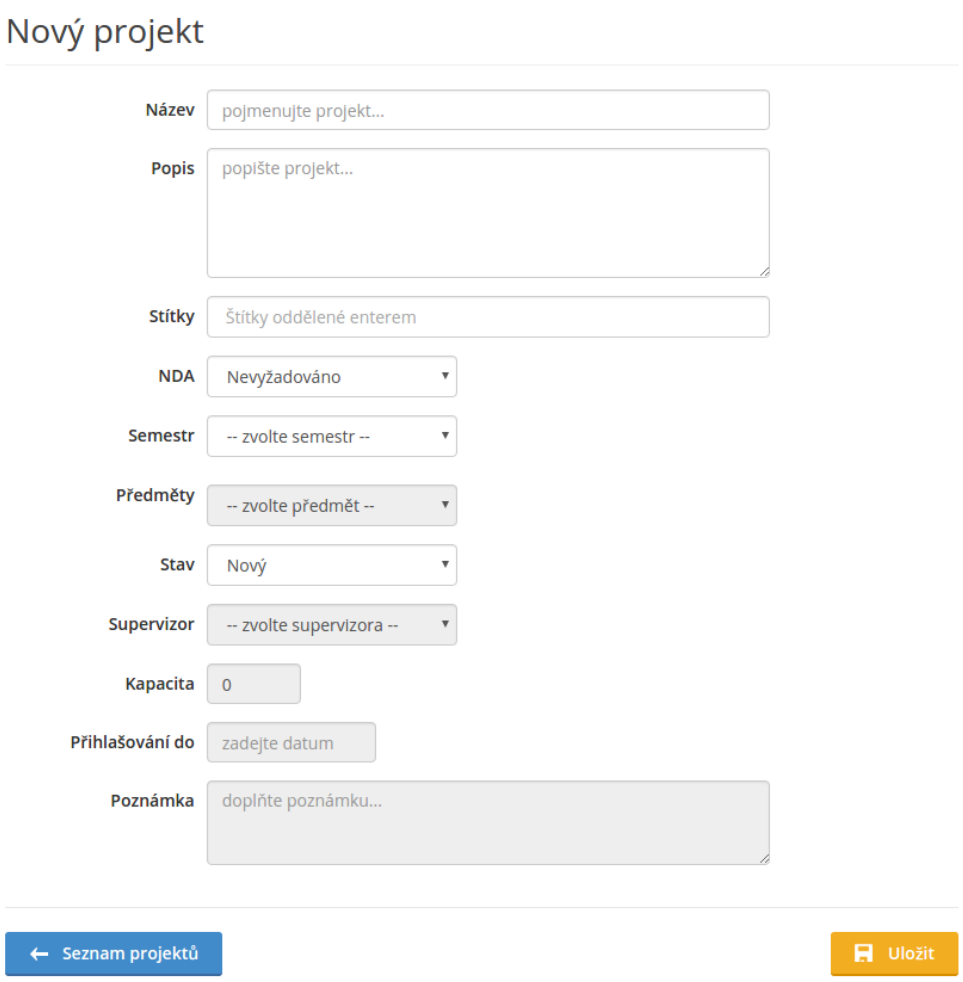


Obrázek C.3: Snímek okna detailu projektu.

obrazovky detailu projektu je na obrázku C.3. Obrazovka se shora dolů skládá z těchto částí: název, oblast tlačítek, další atributy projektu a seznam přihlášek. Význam částí je zřejmý ze snímku a popisků.

Oblast tlačítek zobrazuje tlačítko pro návrat na seznam projektů a v případě, že uživatel může projekt upravovat, také tlačítko pro přechod na obrazovku úpravy projektu. Pokud je zobrazen detail nabízeného projektu okolního systému, je zobrazeno tlačítko pro přechod na stránku projektu v odpovídajícím systému.

Vytvoření a úprava projektu



Obrázek C.4: Snímek okna vytvoření a úpravy projektu.

Obrazovka sloužící pro vytváření a úpravy projektů je zachycena na snímku na obrázku C.4. Hlavní částí obrazovky je formulář s položkami projektu. Nadpis stránky určuje akci, která je vykonávána. Obsahuje buď výraz „Nový projekt“, nebo „Úprava projektu“.

Na obrazovku je možné se dostat buď za účelem vytvoření projektu z obrazovky seznamu projektů, nebo za účelem úpravy projektu z obrazovky detailu projektu. Ve druhém případě jsou položky formuláře předvyplněny aktuálními hodnotami položek projektu.

Pod formulářem projektu jsou tlačítka pro návrat na seznam projektů a uložení hodnot položek projektu. V případě, že je projekt upravován, je zobrazeno také tlačítko pro návrat na detail projektu.

Některé z položek projektu lze vyplnit až v momentě, kdy to dává podle pravidel systému smysl nebo má k tomu uživatel potřebná oprávnění (roli vzhledem k projektu).

Po stisku tlačítka „Uložit“ je provedena validace úprav položek projektu a v případě úspěchu jsou změny uloženy a uživatel je přesměrován na detail upraveného projektu. V případě chyb při validaci je zobrazena chyba.

Příloha D

Popis nástrojů

Tato příloha obsahuje manuál k použití pomocných nástrojů, které byly vytvořeny v průběhu této práce. Jsou to nástroje k obsluze rozhraní vyvíjených služeb (REST, SOAP) a nástroj pro generování dat Burzy projektů. Nástroje jsou umístěny na přiloženém médiu v adresáři se soubory pro zprovoznění systému (viz příloha G).

Shell klient pro Burzu projektů

Shell klient pro Burzu projektů je skript, pomocí kterého je možné ovládat systém Burza projektů přes REST rozhraní. Využívá programy curl a python. Pokud je skript spuštěn bez parametrů, vypíše tuto nápovědu:

```
USAGE:
  ./projex-cli.sh getProjects key=value ...
  ./projex-cli.sh getProject id
  ./projex-cli.sh getEnrollments key=value ...
  ./projex-cli.sh getEnrollment id
  ./projex-cli.sh getTags query
  ./projex-cli.sh postProject <file.json
  ./projex-cli.sh putProject id <file.json
  ./projex-cli.sh postEnrollment <file.json
  ./projex-cli.sh putEnrollment id <file.json
  ./projex-cli.sh importTopics id ...

  env. variables: hostname, port, user
```

Operace, kterou skript vykoná, je určena prvním parametrem skriptu. Ten odpovídá názvu požadavku na REST rozhraní. Za prvním parametrem skriptu mohou následovat parametry operace. Proměnné prostředí, které nástroj využívá, jsou vyjmenovány v posledním řádku výpisu.

Dotazům na získání kolekce projektů nebo přihlášek je možné předat URL parametry dotazu jako parametry operace. Parametr `id` je identifikátor ob-

jektu, který je získávaný nebo upravovaný. Parametrem operace získání štítků je hledaný podřetězec. Parametry operace importu rámcových témat jsou jejich identifikátory.

Operace získávání objektů vypisují obdržená data formátovaně (čitelně) na standardní výstup. Operace vytvoření a úprav objektů očekávají posílaná data na standardním vstupu. Objekty jsou předávány ve formátu JSON. Kompletní specifikace REST rozhraní Burzy projektů je na přiloženém médiu v adresáři s doplňkovou dokumentací (viz příloha G). Příklady struktury projektu a přihlášky jsou v podadresáři `templates` adresáře, kde se nachází popisovaný skript.

Následuje příklad použití skriptu pro operaci získání prvních dvaceti nabízených rámcových témat srovnaných podle názvu vykonanou uživatelem s fakultním loginem `neburjak`:

```
user=neburjak ./projex-cli.sh getProjects \
  query="source==ZP;state==CREATED" \
  page=1 size=20 sort=title
```

Shell klient pro zástupné služby

Shell klient pro zástupné služby je skript, který usnadňuje použití SOAP rozhraní zástupné služby systému ZP a REST rozhraní zástupné služby portálu SSP. Skript využívá programy curl, xmllint a python. Pokud je skript spuštěn bez parametrů, vypíše tuto nápovědu:

```
USAGE:
  ./swsvc-cli.sh update
  ./swsvc-cli.sh getProjects
  ./swsvc-cli.sh getTopics
  ./swsvc-cli.sh getTopicsByAuthor author [ids...]

  env. variables: hostname, port
```

Operace, kterou skript provede, je určena prvním parametrem jeho volání. Operace `update` aktualizuje kolekci rámcových témat a projektů SSP, které budou zástupné služby nabízet. Kolekci rámcových témat sestaví z obsahů souborů v adresáři `topics`, kolekci projektů SSP z obsahů souborů v adresáři `ssp_projects`. Oba adresáře musí být v adresáři, kde je skript spouštěn.

Operace `getProjects` získá všechny nabízené projekty SSP ze zástupné služby portálu SSP. Operace `getTopics` vrátí všechna rámcová témata nabízená zástupnou službou systému ZP. Pomocí operace `getTopicsByAuthor` je možné získat jen nabízená rámcová témata pro konkrétního zadavatele a volitelně je možné požadovat pouze rámcová témata se zadanými identifikátory.

Generátor dat Burzy projektů

Nástroj pro generování dat Burzy projektů byl vytvořen především za účelem otestování REST rozhraní Burzy projektů větším množstvím požadavků a vygenerování většího objemu dat pro testování jejich prezentace v uživatelském rozhraní. Schéma použití nástroje je následující:

```
java -jar projex-data-generator.jar \
  [ --parametr=hodnota | --priznak ] ...
```

Nástroj se tedy spouští jako JAR soubor programem `java` a jeho chování je možné ovlivnit několika volitelnými parametry a příznaky (parametry bez hodnoty), které budou postupně vysvětleny.

Běh programu má následující dvě fáze:

1. Nejdříve jsou z fakultních informačních služeb shromážděny informace o tom, kteří uživatelé mají nějakou relevantní roli vůči všem nominacím, které je možné nakombinovat z hodnot parametrů `semesters` a `courses` (hodnoty parametrů jsou identifikátory semestrů a předmětů oddělené čárkou).

   Výstup této fáze je možné uložit do souboru, jehož jméno je zadáno parametrem `nominationOut`. Běh programu je možné ukončit po této fázi přidáním příznaku `nominationOnly`.

2. Ve druhé fázi běhu jsou na základě výstupu první fáze vygenerována data a provedeny dotazy na REST rozhraní Burzy projektů. První fázi lze přeskočit a její výstup nahradit vstupem ze souboru parametrem `nominationIn`, jehož hodnota obsahuje jméno vstupního souboru. Parametrem `projex.base_uri` je možné nastavit adresu (včetně portu) služby REST rozhraní Burzy projektů.

Ukázková použití nástroje jsou v příručce zprovoznění v příloze B. Následuje výčet dalších parametrů druhé fáze běhu nástroje bez dalšího popisu. Parametry slouží pro jemnější nastavení generování dat (např. k určení pravděpodobnosti, že bude nominovaný projekt schválen).

```
--projectsCount, --maxEnrollments, --minDeadline,
--maxDeadline, --minCapacity, --maxCapacity,
--studentSubmitterRatio, --nominatedProjectsRatio,
--supervisedProjectsRatio, --approvedProjectsRatio,
--openedProjectsRatio, --closedProjectsRatio,
--rejectedEnrollmentsRatio, --cancelledEnrollmentsRatio
```

Příloha E

Odkazy

Následující odkazy byly platné 12.5.2016.

GitLab: `http://gitlab.fit.cvut.cz`

- **kosapi-client:** `https://gitlab.fit.cvut.cz/neburjak/kosapi-client`
- **projex-stub-services:** `https://gitlab.fit.cvut.cz/neburjak/projex-stub-services`
- **projex-service-modules:** `https://gitlab.fit.cvut.cz/probur/probur`
- **projex-frontend-modules:** `https://gitlab.fit.cvut.cz/neburjak/projex-ssp-frontend`

Liferay Portal download: `https://sourceforge.net/projects/lportal/files/Liferay%20Portal/6.1.2%20GA3/liferay-portal-tomcat-6.1.2-ce-ga3-20130816114619181.zip/download`

Ubuntu help - PostgreSQL: `https://help.ubuntu.com/community/PostgreSQL`

Příloha F

Seznam použitých zkratek

API application programming interface

CSS cascading style sheets

CZM Centrum znalostního managementu

GRASP general responsibility assignment software patterns

HTML hypertext markup language

HTTP hypertext transfer protocol

IDE integrated development environment

IoC inversion of control

JAR Java archive

JPA Java Persistence API

JSF JavaServer Faces

MVC model-view-controller

NDA non-disclosure agreement (dohoda o mlčenlivosti)

ORM objektově relační mapování

RAML RESTful API modeling language

REST representational state transfer

RPC remote procedure call

RSQL REST query language

SOAP simple object access protocol

SQL structured query language

SSP spolupráce s průmyslem

URL uniform resource locator

WAR web archive

WSDL web service definition language

XHTML extensible hypertext markup language

XML extensible markup language

ZP závěrečné práce

Příloha G

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `dist` ..... adresář se soubory pro zprovoznění systému
- `doc` ..... adresář s doplňkovou dokumentací
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `zadani.pdf` ..... zadání práce ve formátu PDF