FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	CzechIdM: Modul pro správu licencí
Student:	Bc. Vladimír Kotýnek
Vedoucí:	Ing. Jiří Špaček
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je implementovat komplementární modul „Podpora správy licencí“ do open-source produktu CzechIdM, jehož pomocí bude možno řídit přidělování licencí v rámci automatizovaného životního cyklu identity v organizaci.

Implementován bude serverový modul s RESTful rozhraním a klientský modul s tímto rozhraním komunikující.

Úkoly:
1) Proveďte rešerši současných možností integrace licenčních serverů a IdM.
2) Navrhněte a realizujte serverový i klientský modul poskytující funkcionalitu správy licencí v produktu CzechIdM. Implementované řešení otestujte.
3) Popište standardní životní cyklus identity, jak je realizován produktem CzechIdM.
4) Navrhněte a realizujte integraci modulu do standardního životního cyklu identity v organizaci.
5) Zhotovte technickou dokumentaci relevantních částí implementace a dodejte instalační příručku s popisem konfiguračních voleb.
6) Zhodnoťte výsledky práce a proveďte diskuzi nad možnostmi budoucího rozšíření navrženého řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 7. ledna 2018

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

Diplomová práce

CzechIdM: Modul pro správu licencí

Bc. Vladimír Kotýnek

Vedoucí práce: Ing. Jiří Špaček

9. května 2018

Poděkování

Chtěl bych tímto poděkovat kolegům ze společnosti BCV solutions s. r. o. za navržení tohoto tématu a za cenné konzultace v průběhu jeho vypracování. Rovněž bych chtěl poděkovat panu Ing. Jiřímu Špačkovi za vedení mé práce a cenné rády. A v neposlední řadě bych chtěl poděkovat své rodině, která mě po celou dobu studia plně podporuje, a svým přátelům, kteří mi jsou duševní oporou.

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 9. května 2018 .....

Odkaz na tuto práci

Kotýnek, Vladimír. *CzechIdM: Modul pro správu licencí.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá tvorbou modulu pro správu licencí pro systém pro správu identit *CzechIdM*. Nejdříve je provedena analýza aplikace *CzechIdM*, a rešerše možností správy licencí pomocí tohoto programu. Poté je provedena analýza sebraných požadavků a navržena funkční a technická specifikace modulu. Nakonec je implementován první prototyp modulu pro správu licencí.

Klíčová slova Identity Management, CzechIdM, licence, správa oprávnění, správa licencí, Java, JavaScript, Spring, React, Redux, RESTful, IdM, oprávnění, modularita

Abstract

This thesis deals with the creation of a license management module for the identity management system *CzechIdM*. First, an analysis of the *CzechIdM* is performed, and options of license management using this program are researched. The requirements analysis is then performed and the functional and technical specifications of the module are proposed. Finally, the first prototype of the license management module is implemented.

Keywords Identity management, CzechIdM, license, privilege management, licence management, Java, JavaScript, Spring, React, Redux, RESTful, IdM, privileges, modularity

Obsah

Seznam obrázků

Úvod

Správa uživatelských identit

Správa identit a řízení přístupů – Identity & Access Management (IAM) – je bezpečností disciplína, která zajišťuje přístup těm správným osobám k těm správným zdrojům v ten správný čas a z toho správného důvodu.[1] Pro naplnění tohoto úkolu bývá v organizacích využíváno více technologií a systémů zároveň, specializovaných na konkrétní oblasti IAM. Typicky jsou to adresářové služby, např. *Microsoft Active Directory* (AD) nebo systémy implementující Lightweight Directory Access Protocol (LDAP), implementace Single sign-on (SSO) nebo Identity Management systém (IdM). Jedním z příkladů IdM je aplikace i *CzechIdM*.

CzechIdM je Open Source systém pro správu identit a uživatelských přístupů, vyvíjený společností BCV solutions s. r. o. od roku 2009. Jde již o druhou generaci tohoto systému; poslední stabilní verzí té první je *CzechIdM 6*. Od roku 2016 probíhá vývoj nové generace produktu, tentokrát již veřejně na službě *GitHub*. Tato nová generace se od té staré liší nejen použitými technologiemi, ale i architekturou. Ještě téhož toku byly publikovány první dvě stabilní verze *CzechIdM 7*. V době vzniku této práce je poslední stabilní verzí *CzechIdM 8.0.0*, s kódovým označením *Hematite*.

V organizacích, implementujících správu uživatelských účtů a oprávnění životním cyklem identity v IdM, bývá častým požadavkem rozšířit tuto automatizaci i na další entity. Takovými entitami mohou být např. uživatelské certifikáty, žádanky o vydání nějakého příslušenství (např. přístupových karet, mobilních telefonů nebo notebooků) nebo o přidělení, či zarezervování licence k softwaru. V rámci celého řešení IAM v organizaci je pak třeba najít způsob, jak tento požadavek splnit.

Procesy životního cyklu identity jsou často implementovány právě v IdM. Systém *CzechIdM* lze rozšiřovat pomocí modulů, aniž by byly nechtěně ovlivněny jeho ostatní funkce. Vytvoření specializovaných rozšiřujících modulů pro

jednotlivé agendy může být řešením těchto požadavků. Předmětem této práce bude vytvoření jednoho takového modulu, a sice modulu pro správu softwarových licencí.

Mou motivací k výběru tohoto tématu byl fakt, že jsem se několik let podílel na vývoji první generace systému *CzechIdM* i jeho nasazení u zákazníků společnosti BCV solutions s. r. o. Touto prací bych chtěl pomoci k rozšíření funkcí nové generace systému. Zejména jsem pak chtěl vytvořit něco, co bude v praxi používáno.

Cíl práce

Úkolem, kterým se tato práce zabývá, bylo navrhnout a implementovat rozšiřující modul pro systém *CzechIdM*, zajišťující správu softwarových licencí. Modul má zajišťovat přidělování a odebírání softwarových licencí v rámci životního cyklu identity v organizaci. V případě potřeby bude možné modul rozšířit tak, aby umožnil vytváření žádanek i o další příslušenství, nejen licence.

Prvním cílem bylo vytvořit funkční specifikaci a návrh modulu. Nejprve bylo třeba analyzovat sebrané byznys požadavky na novou funkčnost, poté analyzovat a popsat stávající funkce a možnosti *CzechIdM* a případných dalších již existujících rozšiřujících modulů, kterých půjde při návrhu využít.

Druhým cílem bylo implementovat jak serverovou, tak i klientskou část modulu. Ve výsledku šlo o samostatný serverový modul, implementovaný v jazyce *Java* a samostatný klientský modul, implementovaný v jazyce *JavaScript*. Implementaci pak bylo třeba podrobit uživatelskému testování a toto testování vyhodnotit.

Třetím cílem bylo vytvořit postup instalace modulu do *CzechIdM*. Mimo standardní dokumentace modulu, bylo třeba popsat, jak jej do aplikace nainstalovat, a zejména jak upravit konfiguraci aplikace i dalších objektů v aplikaci, aby bylo možno modul plnohodnotně používat. Na závěr pak byly diskutovány možnosti integrace s externími licenčními servery.

Struktura práce

V první části se tato práce zabývá funkční analýzou na základě sebraných požadavků na funkce modulu. Obsahuje nejdříve popis IAM a IdM jako jeho součásti, nástin řešené problematiky a popis *CzechIdM* a jeho vlastností, zejména modularity a událostmi řízené architektury. Následuje specifikace funkčních požadavků, uživatelských rolí a definice hlavních případů užití. Nakonec tato část obsahuje návrh integrace modulu se stávajícími funkcemi *CzechIdM* a jeho moduly a návrh uživatelského rozhraní v podobě „wireframe“ skic.

Druhá část práce se zabývá popisem použitých technologií, diskusí jejich výhod a popisem implementace. Nejprve je popsána serverová část aplikace,

volba použitých technologií v této části, členění a struktura projektu a nejdůležitější balíčky, třídy a rozhraní. Poté je popsána klientská část aplikace, volba použitých technologií v této části a struktura projektu. Nakonec jsou popsány možnosti a nápady, jak modul dále rozvíjet v budoucnu.

Třetí část práce se zabývá uživatelským testováním modulu. Je zde popsána metodika testování, průběh testování a zhodnocení jeho výsledků. Nakonec jsou v této kapitole popsána doporučení na změny, vyplývající z tohoto testování a rozhodnutí, které změny budou v rámci této práci zapracovány a které budou ponechány pro další možný rozvoj aplikace.

Čtvrtá část práce se zabývá dokumentací modulu. Nejprve obsahuje návod na sestavení a instalaci jednotlivých částí modulu do aplikace *CzechIdM*. Dále obsahuje popis konfigurace modulu nainstalovaného v *CzechIdM* a konfigurační změny, které je dále nutné pro používání modulu v aplikace provést. Tato část neobsahuje kompletní dokumentaci *JavaDoc*, ani dokumentaci RESTful API, které jsou automaticky vygenerovány.

Kapitola 1

Analýza a návrh

Tato kapitola se zabývá analýzou a návrhem modulu pro správu licencí. Je rozdělena do celkem osmi navazujících částí. Nejprve se zabývá popisem problematiky IAM a základních vlastností aplikace *CzechIdM*. Dále se pak věnuje sběru požadavků, rešerši možností řešení pomocí stávajících funkcí aplikace a z těchto poznatků vyplývající definici funkční a technické specifikace modulu. Poté popisuje architekturu a možnosti rozšíření rozšíření *CzechIdM* o doplňující moduly i její další integrační možnosti. Nakonec následuje návrh uživatelského rozhraní modulu a návrh datového modelu.

1.1 Popis problému

IAM se v podnikovém IT zabývá definováním a správou rolí a přístupových oprávnění jednotlivých uživatelů sítě a podmínkami, za kterých jsou tato oprávnění uživatelům přidělena (nebo zamítnuta). Hlavním cílem IAM systémů je zavedení jedné digitální identity, reprezentující vždy jednu reálnou osobu. Jakmile je tato digitální identita v organizaci zavedena, musí být udržována, aktualizována a monitorována v rámci celého svého životního cyklu i životního cyklu všech jejích uživatelských účtů.[2]

IAM systémy poskytují administrátorům nástroje a technologie ke změnám uživatelských rolí, auditování aktivit uživatelů, auditování změn atributů uživatelských účtů, vytváření reportů o těchto aktivitách a průběžnému vynucování politik společnosti. Tyto systémy jsou navrženy tak, aby poskytovaly prostředky pro správu přístupů uživatelů v rámci celé organizace a k zajišťování dodržování jak interních pravidel organizace, tak i legislativních nařízení. IdM systémy často poskytují nástroje pro správu uživatelských hesel, nástroje pro synchronizaci uživatelských účtů do spravovaných systémů, nástroje pro vynucování bezpečnostních politik, nástroje pro reportování a monitorování a zejména databázi identit. V organizacích, které pro automatizaci svých personálních procesů nepoužívají jiný workflow engine, však bývají nástroje IdM

využívány i mimo oblast informační bezpečnosti, a to k evidenci dalších zdrojů, které jsou s identitou spjaty.

Jedním ze zdrojů, jejichž přidělování uživatelům může být v organizacích řízeno pomocí IdM, jsou softwarové licence. Pro řadu informačních systémů a aplikací, používaných v organizaci, bývá k dispozici omezený počet uživatelských licencí, případně je omezen počet účtů, či uživatelských identit, které v aplikaci mohou najednou existovat. Při vytváření účtu v takovémto informačním systému nebo povolování přístupu k takovéto aplikaci je třeba brát omezení dostupných licencí v potaz. Z toho důvodu je vhodné, aby systém, který zajišťuje vytváření účtů a nastavení přístupů do těchto systémů a aplikací, uměl evidovat zároveň i licence a licenční omezení a uměl jim přizpůsobit své procesy.

Prvním krokem této práce byl sběr byznys požadavků a funkční analýza. V rámci nich bylo identifikováno několik hlavních oblastí, které má IdM v rámci správy licencí pokrýt. První a nejzásadnější z těchto požadavků je, aby bylo vytvoření účtu na systému podmíněno dostupností licence pro tento systém. Dalším důležitým požadavkem byla možnost žádat o licenci samostatně, bez návaznosti na existenci uživatelského účtu na koncovém systému. Požadavkem také je, aby bylo možno podmínit přidělení licence schválení nadřízeným nebo jinou oprávněnou osobou. Dále pak přidělit uživateli více, než jednu licenci. Z analýzy také vyplynulo, že zadávání dostupných licencí do systému musí být dostatečně jednoduché, aby jej zvládli i běžní uživatelé pouze s minimálním proškolením. Detailněji jsou funkční požadavky, uživatelské role a případy užití rozpracovány v kapitolách 1.3, 1.4 a 1.5.

1.1.1 Identity Manager CzechIdM

CzechIdM je systém pro správu identit, poskytující většinu z výše popsaných funkcí IAM. Jde o aplikaci, skládající se ze dvou oddělených částí. První z nich je serverová část *CzechIdM*, implementovaná v jazyce *Java* pomocí frameworku *Spring*. Tato serverová část aplikace potřebuje mít pro svou správnou funkci k dispozici databázi *PostgreSQL*, k níž se připojuje pomocí JDBC, přičemž je využito frameworku *Hibernate ORM*. Není-li k dispozici DBMS *PostgreSQL*, může *CzechIdM* používat interně zabudované DBMS *H2*, v takovém případě jsou však uložená data při restartu aplikace ztracena. V serverové části jsou prováděny veškeré operace s identitami, jejich atributy, kontrakty a rolemi. Veškeré funkce serverové části jsou dostupné přes JSON RESTful API. Součástí aplikace je i klientská část – webové rozhraní. Jde o samostatnou aplikaci, která se serverovou částí komunikuje pomocí jejího JSON RESTful API. Klientská aplikace je implementována v jazyce *JavaScript* pomocí knihoven *React* a *Redux*.

Jak je naznačeno výše, serverová část *CzechIdM* je nezávislá od klientské, která tvoří uživatelské rozhraní. Lze tedy v rámci jedné instalace *CzehcIdM* provozovat více klientských částí – uživatelských rozhraní, která se mohou

lišit dostupnými funkcemi, účelem použití, dostupností nebo grafikou. Tato vlastnost umožní de facto zpřístupnění některých modulů z pohledu koncového uživatele samostatně.

Pro napojení spravovaných i zdrojových systémů používá *CzechIdM* tzv. konektory. Pomocí těchto konektorů jsou jak zakládány, spravovány a mazány účty v napojených systémech, tak i synchronizována data ze zdrojových systémů. Stejně jako je tomu u jiných IdM je i i součástí *CzechIdM* interní identity connector server s názvem *ConnId*. Navíc existuje možnost současného využití externího, samostatně provozovaného identity connector serveru, podmínkou však je, že tento server musí používat stejnou verzi identity connector frameworku (ICF) *ConnId*, jako *CzechIdM*. Všechny konektory, které jsou v *CzechIdM* používány, rovněž musí implementovat rozhraní, vystavené ICF *ConnId*.

Účet na koncovém systému je uživateli založen na základě přidělení a schválení role. Zároveň je možno pomocí rolí uživateli přiřazovat i konkrétní oprávnění na daném systému. Role jsou uživateli přiděleny buďto automaticky na základě atributu identity, či úvazku, nebo manuálně na základě žádosti uživatele pomocí webového rozhraní *CzechIdM*. Z hlediska zakládání uživatelského účtu a správy oprávnění jsou oba způsoby přidělení role rovnocenné.

Přidělení role uživateli je v *CzechIdM* schvalováno, viz kapitola 1.2.1. Schvalování je implementováno pomocí tzv. schvalovacího Business Process Management (BPM) workflow. K implementaci workflow je využit *Activiti* workflow engine, používající k reprezentaci workflow standard Business Process Model and Notation (BPMN) 2.0, zabudovaný přímo do aplikace *CzechIdM*. Použití externího workflow enginu není aplikací *CzechIdM* podporováno. Další využívání *Activiti* workflow enginu není v *CzechIdM* omezeno pouze na role, mohou jej rovněž využívat další rozšiřující moduly.

1.2 Analýza

1.2.1 Životní cyklus identity v IdM

Identitou je v IdM rozuměna množina informací, které popisují jednu reálnou osobu. Některé informace, jako křestní jméno, příjmení uživatelské jméno nebo heslo jsou zásadní pro mnoho IT systémů, protože ty je zpracovávají nebo je např. používají k autentizaci a autorizaci uživatele. IdM zpracovává data o identitě, transformuje je a používá je při správně účtů v napojených systémech[3]

V systému *CzechIdM* je uživatel reprezentován entitou zvanou identita. Jednoduše řečeno, identitu lze popsat jako uživatele registrovaného v *CzechIdM* se všemi jeho atributy, jako např. křestní jméno, příjmení, telefonní číslo atd. Reprezentace identity je však složitější disciplína. Aby bylo možné zvládnout automatické procesy životního cyklu identity, nabízí *CzechIdM* další entity

s atributy, které se k identitě vztahují. Jsou to kontrakty, role a prvky stromového grafu, tvořící stromové struktury (např. organizační strukturu).[3]

Vztah identity v *CzechIdM* se společností nebo s organizací je reprezentován entitou zvanou kontrakt. Tuto entitu si lze představit jako pracovně-právní vztah (PPV), pracovní pozice, zápis do studijního programu, kontrakt s externím spolupracovníkem nebo jinou formální vazbu reálné osoby na organizaci. Identita může mít více kontraktů. Kontrakt potom může být ve vztahu s dalšími objekty v *CzechIdM* – již zmíněnými identitami, dále pak s rolemi nebo prvky stromové ho grafu, např. organizační struktury atd. Kontrakt tedy představuje hlavní vlastnost identity v *CzechIdM*, zajišťující její vztah k ostatním objektům v databázi (DB) systému.

Identity lze do *CzechIdM* synchronizovat ze zdrojového systému nebo je lze zakládat ručně. Zdrojovým systémem může pro *CzechIdM* být např. personální systém nebo datový výměník. Identity a jejich kontakty v tomto případě vznikají v rámci nakonfigurované pravidelně spouštěné automatické synchronizační úlohy. Ruční založení identity pak může být používáno např. při samostatné registraci uživatelů, či pro zakládání účtů externistům jejich garantem v organizaci. Při obou způsobech vzniku nové identity v *CzechIdM* jsou na tuto identitu aplikovány stejné automatické procesy životního cyklu.

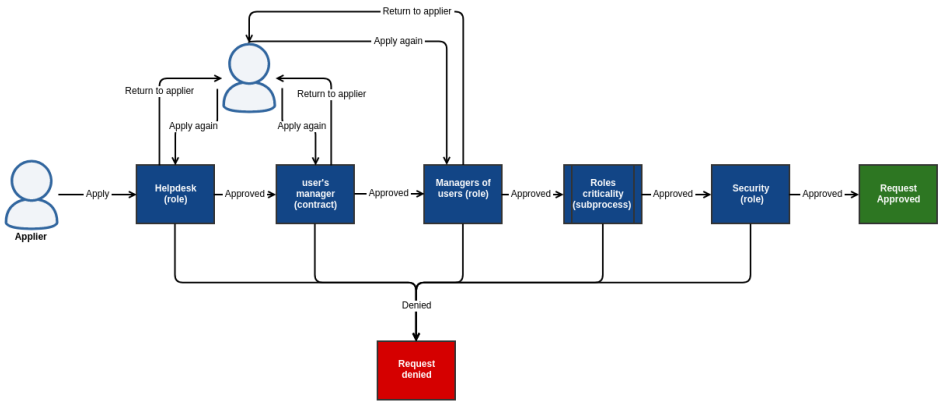
Automatickými procesy životního cyklu identity (Identity lifecycle process, ILP) je označována množina procesorů a spouštěčů, které automaticky reagují na konkrétní události – změnu stavu identity, jejích kontraktů a přidělených rolí. Základními procesy v *CzechIdM* jsou *Začátek kontraktu*, *Konec kontraktu*, *Přerušení kontraktu* (např. vynětí z evidenčního počtu u PPV) a *Změna organizačního zařazení*. Tato množina základních procesů v *CzechIdM* je také označována jako personální procesy nebo HR procesy.[4] Na jednotlivé události však lze navázat i další rozšiřující operace (procesory), které na ně reagují. Tyto operace mohou být dostupné pouze v případě aktivace některého rozšiřujícího modulu.

Kromě automatických procesů je identita spravována i manuálními zásahy přímo v *CzechIdM*. Příkladem takových zásahů ze škály základních funkcí systému bez rozšiřujících modulů je žádání o přidělení rolí. Role mohou být uživateli přiděleny skrze formulář „Žádost o změnu oprávnění“. Uživatel si ve formuláři zvolí, které role chce přidat nebo odebrat, může vyplnit ještě textový popis žádosti a odešle žádost ke zpracování. Pro zpracování této žádosti je v *CzechIdM* využíváno výchozí workflow.

Toto workflow lze konfigurovat pomocí aplikačních proměnných typu *Boolean* (nabývají hodnot `true`/`false`):

idm.sec.core.wf.approval.helpdesk.enabled – zapnutí/vypnutí schvalování celé žádosti pracovníkem podpory uživatelů

idm.sec.core.wf.approval.manager.enabled – zapnutí/vypnutí schvalování celé žádosti správcem uživatelů (administrátorem)



Obrázek 1.1: Proces schvalování žádosti o přidělení role [5]

idm.sec.core.wf.approval.usermanager.enabled – zapnutí/vypnutí schvalování celé žádosti nadřízeným uživatele, pro nějž je o roli žádáno

idm.sec.core.wf.approval.security.enabled – zapnutí/vypnutí schvalování celé žádosti pracovníkem oddělení bezpečnosti

Rolím v *CzechIdM* nastavuje administrátor tzv. kritičnost. Mezi schválením žádosti o změnu oprávnění vedoucím uživatele a schválením žádosti pracovníkem oddělení bezpečnosti je pro každou roli ze schvalované žádosti spuštěno samostatné schvalování dle její kritičnosti. V *CzechIdM* je definováno pět základních kritičností role, k nimž lze pomocí konfiguračních proměnných `idm.sec.core.wf.role.approval<1-5>` přiřadit název příslušného schvalovacího workflow. Pokud daná úroveň kritičnosti role nemá vyplněno žádné workflow, schvalování dle kritičnosti role je přeskočeno. V *CzechIdM* jsou k dispozici dvě standardní workflow pro schválení role dle její kritičnosti:

approve-role-by-guarantee – Roli schvaluje její byznys vlastník (garant).

approve-role-by-manager – Roli schvaluje vedoucí uživatele, vypočítaný z organizační struktury.

Ve schvalovacím procesu jsou použity jednotlivé role schvalujících uživatelů: pracovník podpory uživatelů, správce uživatelů a pracovník oddělení bezpečnosti. Tyto role nejsou v *CzechIdM* ve výchozím stavu definovány a je tedy třeba je po instalaci nastavit dle potřeb daného prostředí. Uživatelé jsou vyhledáni na základě toho, že mají jednotlivé role přiděleny v *CzechIdM*. Názvy (kódy) těchto rolí jsou definovány v konfiguračních proměnných:

idm.sec.core.wf.approval.helpdesk.role – pracovník podpory uživatelů

idm.sec.core.wf.approval.manager.role – správce uživatelů (administrátor)

idm.sec.core.wf.approval.security.role – pracovník oddělení bezpečnosti

1.2.2 Rešerše možností správy licencí v IdM

Na základě dohody s vedoucím práce byly analyzovány pouze stávající možnosti integrace *CzechIdM* s aplikacemi a systémy, v nichž mají být spravovány licence. Analýza dalších IdM systémů a jiných aplikací by svým rozsahem překračovala parametry závěrečné práce. Aplikace *CzechIdM* ve své první generaci, ani ve své současné druhé generaci nedisponuje žádnými standardními nástroji pro správu licencí. V současnosti lze absenci těchto nástrojů obejít v zásadě třemi způsoby. První možností je spravovat licence stejně jako uživatelské účty pomocí ICF konektorů a modulu *acc*. Druhou možností je implementace, či použití existujících klientských knihoven pro jednotlivé aplikace, které budou integrovány v procesorech událostí. Poslední možností je ruční správa pomocí modulu *vs* (Virtuální systémy). Ani jedno z těchto řešení není optimální, proto bylo rozhodnuto o implementaci modulu pro správu licencí.

Jak zmiňuje kapitola 1.1.1, *CzechIdM* používá pro integraci se spravovanými i zdrojovými systémy ICF *ConnId*. Ačkoliv je tento ICF určen primárně pro správu uživatelských účtů a skupiny, pokud to konektory podporují, lze je využít ke správě jakýchkoliv objektových tříd. Pro správu licencí je tedy možno implementovat *ConnId* konektor, podporující propisování („provisioning“) licencí. *CzechIdM* nativně podporuje „provisioning“ identit, rolí (skupin) a stromových struktur (organizačních jednotek). Z toho důvodu by bylo třeba navázat správu licencí na jeden z těchto typů objektů, přičemž jako nejvhodnější se jeví uživatelské identity. De facto se tedy IdM bude k licencím chovat stejně, jako by to byly uživatelské účty. Takovéto řešení bude plně funkční, nicméně z hlediska logického členění funkcí nebude správné.

V *CzechIdM* lze implementovat služby (procesory), které reagují na určité typy událostí (podrobnější popis v kapitole 1.5.20.2). Další možností, jak integrovat IdM s aplikací, v níž jsou spravovány licence, je implementace takovéto služby (procesoru), která bude reagovat např. na přidělení či odebrání role uživateli. Tato služba pak pomocí klientské knihovny (konektoru) provede přidělení licence uživateli ve spravované aplikaci. Toto řešení je z hlediska logiky správnější, než předchozí, jelikož nezneužívá pro správu licencí jiné typy objektů. Jeho velkou nevýhodou však je malá univerzálnost, nejzásadnějším nedostatkem tohoto přístupu potom je absence jakékoliv evidence přidělených licencí v IdM.

Modul *Virtuální systémy* je nástroj pro správu aplikací a systémů, které nejsou s *CzechIdM* integrovány, ale pro něž existují v IdM role. Pokud identita dostane přidělenu roli, garantující účet nebo nastavující oprávnění na některém z takto spravovaných systémů, IdM vygeneruje úkol správci tohoto sys-

tému, aby změnu realizoval. Obdobně pak při smazání účtu nebo deaktivaci identity je vytvoření úkol pro smazání nebo deaktivací takového účtu. Tyto systémy jsou tedy spravovány pouze virtuálně. Pro správu licencí na koncových systémech lze tento modul použít tak, že vznikne virtuální systém, jenž bude namísto účtu evidovat licence na daném systému, a pro tento systém vzniknou příslušné role. Samotné přidělení, či odebrání licence, pak bude provádět správce tohoto systému manuálně na základě úkolu v IdM. Toto řešení je sice nejjednodušší a umožňuje evidenci přidělených licencí, ale jednak nezajišťuje plnou automatizaci procesu, jednak rovněž zneužívá správu uživatelských účtů ke správě jiného typu objektu, není tedy správné ani z hlediska logického členění aplikace.

1.3 Funkční a nefunkční požadavky

Na základě schůzek s obchodníky, byznys analytiky a architektem ze společnosti BCV solutions s. r. o byl vytvořen popis požadovaných vlastností aplikace. Na základě tohoto popisu byly vytvořeny funkční a nefunkční požadavky modulu pro správu licencí pro *CzechIdM*. Tyto požadavky byly následně schváleny architektem a produktovým manažerem a rozděleny do tří fází implementace:

1. **Základní správa licencí:** F1.1.1, F1.1.2, F1.1.3.2, F1.1.4, F1.1.4.1, F1.1.4.2, F1.1.5, F1.1.6, F1.1.7, F1.1.7.1, F1.1.7.2, F1.2.1, F1.2.1, F1.2.2, F1.2.4, F1.2.6, F1.2.6.1, F1.2.6.1.1, F1.2.6.3, F1.2.6.3.1, F1.2.6.3.2, F1.2.7, F1.2.9, F1.3.1, F1.4, F2.1, F2.2, F2.3, F2.4, F3.1.3, F3.1.4, F3.1.5, F3.1.6[1], F3.2

2. **Správa licencí na základě role:** F1.1.3, F1.1.3.1, F1.2.1, F1.2.1.2, F1.2.3, F1.2.3.1, F1.2.3.2, F1.1.3.3, F1.2.8, F1.2.8.1, F1.2.8.1.1, F1.2.8.2, F1.2.8.2.1, F1.3.2, F1.4, F4, F4.1, F4.2, F4.3, F4.4, F4.7, F4.8

3. **Pokročilá správa licencí:** Zbývající funkční požadavky a zapracování připomínek a změnových požadavků z uživatelského testování.

Tato práce se zabývá implementací zejména první fáze, tedy základní správy licencí, a přípravy implementace fáze druhé.

1.3.1 Funkční požadavky

F1 Zadávání licencí do systému

F1.1 Lze vytvořit licenční sadu.

[1]Pouze základní ověření, zda má uživatel na právo licence přidělovat, v rámci této fáze nebude implementováno detailnější členění oprávnění

F1.1.1 Povinně vyplnit unikátní identifikátor (kód) licenční sady, uživatelský název licenční sady (např. *MS Office 365*).

F1.1.2 Volitelně vyplnit hodnotu licence, poznámku.

F1.1.3 Povinně vybrat z nabídky, zda lze o licenci žádat přímo, nebo je přidělována automaticky na základě role.

F1.1.3.1 Pokud je licence přidělována na základě role, povinně vybrat, zda blokovat přidělení role při vyčerpání licencí.

F1.1.3.2 Pokud lze o licenci žádat, povinně vybrat schvalovací workflow z nabídky.

F1.1.4 Povinně zadat dostupné licence.

F1.1.4.1 Lze zvolit, zda zadat pouze počet dostupných licencí - zobrazí se pole pro zadání celkového počtu.

F1.1.4.2 NEBO lze nahrát seznam dostupných licenčních čísel z textového (CSV) souboru, případně zadat ručně do textového pole.

F1.1.5 Lze vybrat, zda mohou být licence opětovně přiděleny.

F1.1.6 Zadanou licenci lze uložit, nebo zadání stornovat.

F1.1.7 Lze zadat byznys vlastníky licence.

F1.1.7.1 Byznys vlastníků může být 0 až N.

F1.1.7.2 Zadání byznys vlastníka není povinné.

F1.2 Lze upravit licenční sadu.

F1.2.1 Editovat lze název pro uživatele, hodnotu licence, poznámku, seznam byznys vlastníků licence a volbu, zda je umožněno opětovné přidělení licence.

F1.2.1.1 Pokud o licenci lze žádat, lze editovat volbu schvalovacího workflow.

F1.2.1.2 Pokud je licence přidělována na základě role, lze editovat volbu, zda vyčerpání volných licencí zablokuje přidělení role.

F1.2.2 Nelze editovat unikátní identifikátor (kód) licenční sady, ani výběr, zda je dostupný seznam licenčních čísel, nebo pouze počet licencí.

F1.2.3 Lze změnit, zda jde o licenci žádat, nebo je přidělována na základě role.

F1.2.3.1 Před změnou musí uživatel potvrdit, že chce změnu provést.

F1.2.3.2 Po změně je licence odebrána všem držitelům.

F1.2.4 Lze zobrazit seznam licencí – přidělených i dostupných.

F1.2.4.1 Pokud o licenci lze žádat, je umožněno licenci uživateli odebrat, nebo přidat.

F1.2.4.2 Položku lze odstranit ze seznamu (pokud ji uživatel nemá přidělenu na základě role).

F1.2.5 Nelze změnit výběr, zda je zadán pouze počet dostupných licencí, seznam dostupných licenčních čísel.

F1.2.6 Lze změnit počet/seznam dostupných licencí.

F1.2.6.1 Pokud IdM eviduje pouze počet licencí, lze změnit celkový počet dostupných licencí.

F1.2.6.1.1 Nelze zadat menší číslo, než je počet v daný okamžik přidělených licencí.

F1.2.6.2 Pokud IdM eviduje pouze počet licencí, lze ke stávajícímu počtu licencí přičíst nové.

F1.2.6.3 Pokud IdM eviduje seznam licenčních čísel, lze zadat nový seznam licenčních čísel.

F1.2.6.3.1 Lze nahrát z textového (CSV) souboru, nebo zadat ručně do textového pole.

F1.2.6.3.2 Pokud nový seznam neobsahuje licenční čísla, která jsou přidělena o licence lze žádat, jsou uživatelům tyto licence odebrány.

F1.2.6.3.3 Pokud nový seznam neobsahuje licenční čísla, která jsou přidělena o licence nelze žádat (jsou přidělovány na základě role), IdM zobrazí chybovou hlášku.

F1.2.6.4 Pokud IdM eviduje seznam licenčních čísel, lze rozšířit seznam licenčních čísel o nové položky.

F1.2.6.4.1 Lze nahrát z textového (CSV) souboru, nebo zadat ručně do textového pole.

F1.2.7 Bude zobrazen celkový počet licencí, počet přidělených licencí a počet nepřidělených licencí.

F1.2.8 Pokud je licence přiřazována na základě role, lze zobrazit seznam rolí, které ji přidělují.

F1.2.8.1 Lze odebrat roli ze seznamu.

F1.2.8.1.1 Všem držitelům role je licence odebrána.

F1.2.8.2 Lze přidat roli do seznamu.

F1.2.8.2.1 Všem držitelům role je licence přidělena.

F1.2.9 Editaci lze uložit, nebo stornovat.

F1.3 Lze odstranit licenční sadu.

F1.3.1 Po odstranění licenční sady jsou všechny licence, o které jde žádat, odebrány uživatelům.

F1.3.2 Po odstranění licenční sady jsou všechny licence, které jsou přidělovány rolí, odebrány od daných rolí a tím odebrány uživatelům.

F1.4 Přiřazení/odebrání licence roli a přiřazení/odebrání licence uživateli je řízeno standardním schvalovacím procesem v IdM.

F2 Lze zobrazit seznam licenčních sad.

F2.1 Je zobrazen název, popis, počet přidělených/celkový počet licencí sady.

F2.2 Lze přejít do editace konkrétní licenční sady.

F2.3 Lze odstranit konkrétní licenční sadu.

F2.4 Lze přejít do k vytvoření nové licenční sady.

F3 Žádání o licence přímo.

F3.1 Lze požádat o přidělení licence ze seznamu dostupných licencí v profilu uživatele.

F3.1.1 Lze zadat počet licencí, o který je žádáno.

F3.1.2 Nelze zadat větší počet, než kolik jich je momentálně volných.

F3.1.3 Je spuštěno schvalovací workflow, schvalující uživatelé mohou počet licencí měnit.

F3.1.4 Nelze schválit přidělení licence, pokud v ten okamžik není volných alespoň tolik licencí, o kolik uživatel žádá.

F3.1.5 Po schválení IdM automaticky přidělí licence - vybere licenční čísla ze seznamu, nebo z čítače odečte volné licence.

F3.1.6 Seznam licencí, o které lze žádat, bude vypočítán na základě oprávnění přihlášeného uživatele.

F3.2 Lze odebrat licenci z profilu uživatele.

F4 Přidělení licence na základě role.

F4.1 Lze vybrat, ze které sady role licence přiděluje.

F4.2 Jedna role může přidělovat licence z více licenčních sad.

F4.3 Licence z jedné licenční řady mohou být přidělovány více rolemi.

F4.4 Licence z licenční sady nemusí být přidělovány žádnou licenční sadou.

F4.5 Lze nastavit, kolik licencí role přiděluje.

F4.6 Přidělením licenční sady roli bude licence přidána všem jejím držitelům.

F4.7 Seznam licencí, které lze roli nastavit, bude vypočítán na základě oprávnění přihlášeného uživatele.

F4.8 Pokud jsou vyčerpány všechny licence z licenční sady, z níž role uživateli licence přiděluje, a zároveň má licence nastaveno, že blokuje přidělení role při vyčerpání licencí, roli nelze přidělit uživateli – přidělení role zůstane i po schválení aktivní.

1.3.2 Nefunkční požadavky

NF1 Architektura REST.

NF2 Implementace serverové části bude provedena v jazyce *Java* a technologiích, které využívá aplikace *CzechIdM*.

- NF2.1 Spring
- NF2.2 Hibernate ORM
- NF2.3 Swagger
- NF2.4 RESTEasy

NF3 Implementace klientské části bude proveden v jazyce *JavaScript* a technologiích, které využívá aplikace *CzechIdM*.

- NF3.1 React
- NF3.2 Redux

NF4 Pro schvalování přidělení licence bude použit interní workflow engine *Activiti*.

NF5 Modul bude distribuován pod stejnou licencí (*MIT License*), jako aplikace *CzechIdM*.

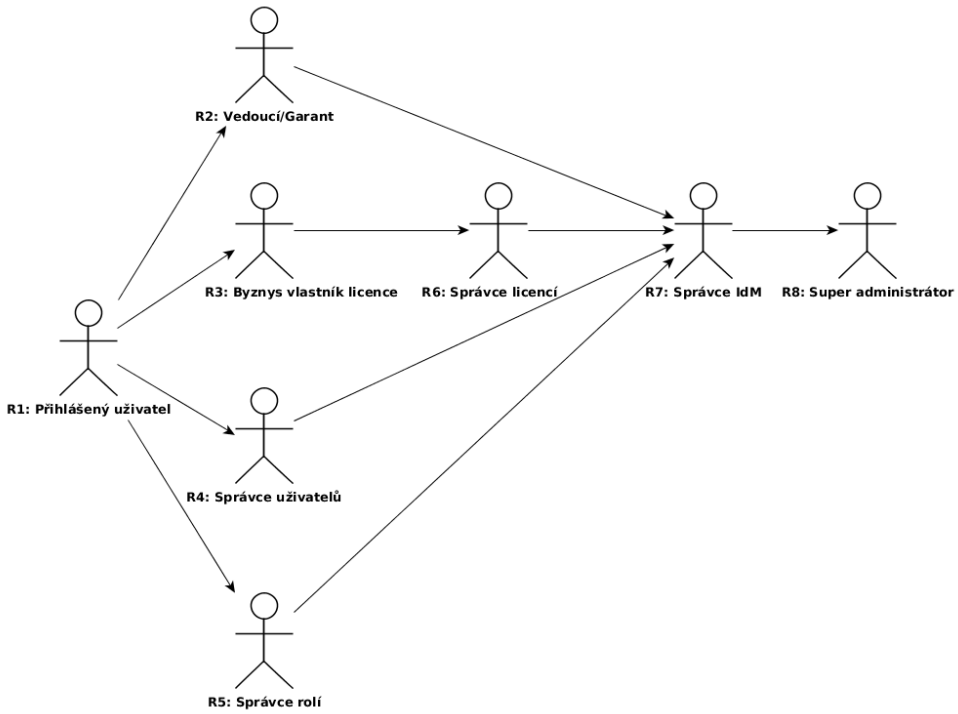
1.4 Uživatelské role

Uživatelské role v *CzechIdM* jsou vztaženy k funkčnostem modulu pro správu licencí. V této sekci není definováno, jaká oprávnění mají jednotlivé role vzhledem k ostatním funkcím systému nebo jeho ostatních rozšiřujících modulů. Ostatní oprávnění jsou zde pouze nastíněna pro lepší představu čtenáře – u takových není uveden identifikátor případu užití. Dále je předpokladem, že každý uživatel, který má v *CzechIdM* přístup k funkčnostem modulu pro správu licencí, má nejméně oprávnění pro přihlášení se do *CzechIdM*.

1.4.1 R1: Přihlášený uživatel

Předek: není

- Zobrazuje vlastní profil
- Žádá o přidělení role pro sebe (U17)
- Sám sobě odebírá přidělenou roli (U18)
- Žádá o přidělení licence pro sebe (U14)
- Sám sobě odebírá přidělenou licenci (U16)



Obrázek 1.2: Hierarchie uživatelských rolí v licenčním modulu

1.4.2 R2: Vedoucí/Garant

Předek: R1: Přihlášený uživatel

- Zobrazuje seznam podřízených
- Zobrazuje profil podřízeného
- Žádá o přidělení role pro svého podřízeného (U17)
- Odebírá přidělené role svým podřízeným (U18)
- Žádá o přidělení licence pro své podřízené (U14)
- Odebírá přidělené licence svým podřízeným (U16)
- Schvaluje přidělení licence svým podřízeným (U15)

1.4.3 R3: Byznys vlastník licence

Předek: R1: Přihlášený uživatel

- Schvaluje přidělení licence z vlastněné licenční sady uživatelům (U15)

- Zobrazuje licence z vlastněné licenční sady (U19)
- Přiděluje/odebírá licence z vlastněné licenční sady uživatelům (U6, U7)
- Aktualizuje seznam licencí z vlastněné licenční sady (U2, U3, U4)

1.4.4 R4: Správce uživatelů

Předek: R1: Přihlášený uživatel

- Zobrazuje seznam spravovaných uživatelů
- Zobrazuje profil spravovaného uživatele
- Žádá o přidělení role pro spravovaného uživatele (U17)
- Odebírá přidělené role spravovaným uživatelům (U18)
- Žádá o přidělení licence pro spravovaného uživatele (U14)
- Odebírá přidělené licence spravovaným uživatelům (U16)

1.4.5 R5: Správce rolí

Předek: R1: Přihlášený uživatel

- Zobrazuje seznam rolí
- Edituje role
- Odstraňuje role
- Nastavuje roli, které licence přiděluje (U12, U13)

1.4.6 R6: Správce licencí

Předek: R1: Přihlášený uživatel, R3: Byznys vlastník licence

- Zobrazuje všechny licenční sady (U19)
- Vytváří nové licenční sady (U1)
- Edituje stávající licenční sady (U2, U3, U4, U5, U6, U7, U8, U9, U10)
- Odstraňuje licenční sady (U11)

1.4.7 R7: Správce IdM

Předek: R3: Byznys vlastník licence, R4: Správce uživatelů, R5: Správce rolí, R6: Správce licencí

- Spravuje aplikaci *CzechIdM* a všechna data v ní

1.4.8 R8: Super administrátor

Předek: R7: Správce IdM

- Účet s nejvyššími právi v *CzechIdM*

1.5 Případy užití

Všechny testovací scénáře předpokládají, že je uživatel již přihlášen do *CzechIdM* pomocí uživatelského účtu s dostatečnými oprávněními.

1.5.1 U1: Vytvoření nové licenční sady

1.5.1.1 Cíl:

Vytvoření nové licenční sady správcem licencí (R6)

1.5.1.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel klikne na tlačítko „Nová sada“.
3. Systém zobrazí formulář pro založení licenční sady.
4. Uživatel vyplní povinné položky formuláře: kód licenční sady, uživatelský název licenční sady.
5. Uživatel vybere z nabídky, zda lze o licenci žádat přímo, nebo je přidělována automaticky na základě role.
6. Systém aktualizuje formulář dle volby.
7. Uživatel vyplní konfiguraci vybraného způsobu přidělení.
   a) Pro přidělení rolí: Uživatel zvolí, zda blokovat přidělení rolí při vyčerpání licencí.
   b) Pro přímé žádání: uživatel vybere z nabídky schvalovací workflow.
8. Uživatel vybere, zda chce zadat pouze počet dostupných licencí, nebo vložit seznam licenčních čísel.
9. Systém přegeneruje formulář dle volby uživatele.
10. Uživatel zadá dostupné licence.
    a) Pouze počet licencí: Uživatel vyplní do formulářového pole číslo větší, než 0.

b) Seznam licenčních čísel:

1. Uživatel může vyplnit seznam do textového pole – každé licenční číslo na novém řádku.
2. Uživatel může kliknout na tlačítko „Načíst ze souboru“.
3. Systém po kliknutí na tlačítko „Načíst ze souboru“ nahradí textové pole „drop zone“ elementem pro nahrání souboru.
4. Uživatel pomocí „drop zone“ nahraje soubor se seznamem licencí.
5. Systém seznam zpracuje a doplní jej do textového pole a nahradí jeho původní obsah.
6. Uživatel může obsah textového pole dále editovat.

11. Uživatel zvolí, zda mohou být licence opětovně přiděleny.
12. Uživatel volitelně vybere byznys vlastníka licence.
    12.1. Uživatel začne v textovém poli psát jméno, příjmení nebo uživatelské jméno garanta.
    12.2. Systém našeptává odpovídající uživatele.
    12.3. Uživatel vybere garante.
    12.4. Systém přidá garanta do seznamu.
    12.5. V případě více garantů uživatel postup opakuje.
13. Uživatel klikne na „Uložit a zavřít“.
14. Systém novou licenční sadu uloží a přesměruje uživatele na seznam licenčních sad.

1.5.1.3 Chybové stavy:

- V případě nevalidního souboru se seznamem licencí je uživatel na chybu upozorněn a soubor není zpracován.
- V případě nevyplnění povinné položky je uživatel upozorněn standardním upozorněním systému.

1.5.2 U2: Změna seznamu licencí ve stávající licenční sadě ručně

1.5.2.1 Cíl:

V seznamu dostupných licencí pomocí editačního formuláře přidat nebo odebrat licenční čísla.

1.5.2.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel změní počet dostupných licencí:
   a) pokud je zadán pouze počet dostupných licencích, tak
      1. Uživatel změní v textovém poli počet dostupných licencí a klikne na tlačítko „Potvrdit“.
      2. Systém zkontroluje, zda je nový počet licencí větší nebo roven počtu použitých licencí, pokud ano přidá nebo odebere volné licence pro aktualizaci počtu. Pokud ne, zobrazí uživateli chybovou zprávu, aby nejprve odebral některé licence uživatelům.

   b) pokud je zadán seznam dostupných licenčních čísel, tak
      1. Uživatel klikne na tlačítko „Upravit seznam“.
      2. Systém uživateli zobrazí textové pole se seznamem licenčních čísel.
      3. Uživatel upraví textový seznam licenčních čísel a klikne na tlačítko „Potvrdit“.
      4. Systém zkontroluje, zda nebyly odebrány licenční čísla, která jsou přidělena uživatelům a pokud ano, zareaguje dle toho:
         A. Pokud o licence lze žádat, systém zobrazí uživateli modální okno s dotazem, zda skutečně chce smazat tato licenční čísla a odebrat je tak uživatelům. Pokud toto uživatel potvrdí, systém licenční čísla odebere uživatelům a smaže je z licenční sady. Jinak tato čísla v seznamu ponechá.
         B. Pokud jsou licence přidělovány na základě role, zobrazí uživateli chybovou zprávu a dané číslo v seznamu ponechá.

      Pokud přiřazena uživatelům nejsou, odebere systém licenční čísla ze seznamu. Přidaná licenční čísla pak přidá do seznamu dostupných licencí.
7. Uživatel klikne na tlačítko „Uložit a zavřít“.
8. Systém novou licenční sadu uloží a přesměruje uživatele na seznam licenčních sad.

1.5.2.3 Chybové stavy:

- Uživatel zadá nižší počet dostupných licencí, než již bylo z dané sady přiděleno, v takovém případě je na stav upozorněn standardním upozorněním systému.
- Uživatel se snaží odebrat sériové číslo, které je přiděleno uživateli na základě role, v takovém případě je na stav upozorněn standardním upozorněním systému.

1.5.3 U3: Změna seznamu licencí ve stávající licenční sadě importem ze souboru – přepsání seznamu

1.5.3.1 Cíl:

Stávající seznam dostupných licencí v konkrétní licenční sadě přepsat novým seznamem, importovaným ze souboru v počítači uživatele.

1.5.3.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel klikne na tlačítko „Načíst nový seznam ze souboru“.
7. Systém uživateli zobrazí drop zónu pro nahrání souboru.
8. Uživatel nahraje nový CSV soubor se seznamem licenčních čísel.
9. Systém zkontroluje, zda nebyly odebrány licenční čísla, která jsou přidělena uživatelům a pokud ano, zareaguje dle toho:

   a) Pokud o licence lze žádat, systém zobrazí uživateli modální okno s dotazem, zda skutečně chce smazat tato licenční čísla a odebrat je tak uživatelům. Pokud toto uživatel potvrdí, systém licenční čísla odebere uživatelům a smaže je z licenční sady. Jinak tato čísla v seznamu ponechá.

   b) Pokud jsou licence přidělovány na základě role, zobrazí uživateli chybovou zprávu a dané číslo v seznamu ponechá.

   Pokud přiřazena uživatelům nejsou, odebere systém licenční čísla ze seznamu. Přidaná licenční čísla pak přidá do seznamu dostupných licencí.

1.5.3.3 Chybové stavy:

- V případě nevalidního souboru se seznamem licencí je uživatel na chybu upozorněn a soubor není zpracován.
- V případě nevyplnění povinné položky je uživatel upozorněn standardním upozorněním systému.
- Uživatel zadá nižší počet dostupných licencí, než již bylo z dané sady přiděleno, v takovém případě je na stav upozorněn standardním upozorněním systému.
- Uživatel se snaží odebrat sériové číslo, které je přiděleno uživateli na základě role, v takovém případě je na stav upozorněn standardním upozorněním systému.

1.5.4 U4: Změna seznamu licencí ve stávající licenční sadě importem ze souboru – rozšíření seznamu

1.5.4.1 Cíl:

Stávající seznam dostupných licencí v konkrétní licenční sadě rozšířit o nový seznam, importovaný ze souboru v počítači uživatele.

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel klikne na tlačítko „Rozšířit seznam ze souboru“.
7. Systém uživateli zobrazí drop zónu pro nahrání souboru.
8. Uživatel nahraje nový CSV soubor se seznamem licenčních čísel.
9. Systém zkontroluje, zda nebyly odebrány licenční čísla, která jsou přidělena uživatelům a pokud ano, zareaguje dle toho:
    a) Pokud o licence lze žádat, systém zobrazí uživateli modální okno s dotazem, zda skutečně chce smazat tato licenční čísla a odebrat je tak uživatelům. Pokud toto uživatel potvrdí, systém licenční čísla odebere uživatelům a smaže je z licenční sady. Jinak tato čísla v seznamu ponechá.

b) Pokud jsou licence přidělovány na základě role, zobrazí uživateli chybovou zprávu a dané číslo v seznamu ponechá.

Pokud přiřazena uživatelům nejsou, odebere systém licenční čísla ze seznamu. Přidaná licenční čísla pak přidá do seznamu dostupných licencí.

1.5.4.2 Chybové stavy:

- V případě nevalidního souboru se seznamem licencí je uživatel na chybu upozorněn a soubor není zpracován.
- V případě nevyplnění povinné položky je uživatel upozorněn standardním upozorněním systému.
- Uživatel zadá nižší počet dostupných licencí, než již bylo z dané sady přiděleno, v takovém případě je na stav upozorněn standardním upozorněním systému.
- Uživatel se snaží odebrat sériové číslo, které je přiděleno uživateli na základě role, v takovém případě je na stav upozorněn standardním upozorněním systému.

1.5.5 U5: Změna konfigurace stávající licenční sady

1.5.5.1 Cíl:

Existující licenční sadě změnit popis, to zda je přidělována na základě žádosti o licence, nebo na základě přidělení role a další atributy.

1.5.5.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel změní volbu, zda jde o licenci žádat přímo, nebo je přidělována na základě role.
7. Systém zobrazí uživateli varování, že touto změnou dojde k odebrání všech licencí uživatelům, kteří je mají přiděleny.
8. Uživatel potvrdí akci.

9. Uživatel klikne na tlačítko „Uložit a pokračovat“.
10. Systém uloží změny a odebere licence uživatelům nebo zruší jejich přidělení na základě role a tím je odebere uživatelům.
11. Uživatel může dále editovat licenční sadu.
12. Uživatel klikne na tlačítko „Uložit a zavřít“.
13. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.

1.5.5.3 Chybové stavy:

- Odebrání licence uživateli zachytí procesor jiného modulu, který na něj zareaguje a operaci zruší, v takovém případě systém neprovede změnu způsobu přidělování licencí a zobrazí uživateli chybové hlášení.

1.5.6 U6: Přidělení licence uživateli ze seznamu licencí v editaci licenční sady

1.5.6.1 Cíl:

Přidělit konkrétní licenční číslo uživateli z formuláře pro editaci licenční sady.

1.5.6.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel přidělí uživateli licenci.
   a) Pokud je evidován pouze počet dostupných licencí.
      1. Uživatel klikne na tlačítko „Přidat licenci uživateli“.
   b) Pokud je evidován seznam licenčních čísel.
      1. Uživatel v seznamu licenčních čísel zaškrtne to, které chce přidělit.
      2. Uživatel klikne na tlačítko „Přidat licenci uživateli“.
7. Uživatel klikne na tlačítko „Uložit a zavřít“.

8. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.
9. Je-li přidělení licence schvalováno, systém spustí schvalování přidělení licence.
10. Je-li přidělení licence schvalováno, oprávněný uživatel schválí přidělení licence.
11. Systém přidělí licenci uživateli.

1.5.6.3 Chybové stavy:

- Oprávněný uživatel zamítne přidělení licence, v takovém případě systém licenci nepřidělí.
- Mezi přidáním uživateli ve formuláři a uložení změn je licenční číslo přiděleno jinému uživateli, v takovém případě systém zobrazí chybové hlášení a licenci uživateli nepřidělí.

1.5.7 U7: Odebrání licence uživateli ze seznamu licencí v editaci licenční sady

1.5.7.1 Cíl:

Odebrání konkrétního licenčního čísla uživateli z formuláře pro editaci licenční sady.

1.5.7.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel v seznamu licencí přidělených uživateli vybere tu, kterou chce odebrat a klikne na tlačítko „Odebrat licenci“.
7. Uživatel klikne na tlačítko „Uložit a zavřít“.
8. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.
9. Systém odebere licenci uživateli.

1.5.7.3 Chybové stavy:

- Odebrání licence uživateli zachytí procesor jiného modulu, který na něj zareaguje a operaci zruší, v takovém případě systém neprovede změnu a zobrazí uživateli chybové hlášení.

1.5.8 U8: Přidání role, která přiděluje licence z licenční sady v editaci licenční sady

1.5.8.1 Cíl:

V editaci licenční sady přidat do seznamu rolí další, která přiřazuje licenci z této sady a nastavit počet licencí, které role přiřazuje.

1.5.8.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel klikne na tlačítko „Přidat roli“.
7. Systém zobrazí formulář pro výběr role.
8. Uživatel vybere roli.
9. Systém přidá roli do seznamu.
10. Uživatel klikne na tlačítko „Uložit a zavřít“.
11. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.

1.5.8.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.9 U9: Odebrání role, která přiděluje licence z licenční sady v editaci licenční sady

1.5.9.1 Cíl:

V editaci licenční sady odebrat ze seznamu roli některou z těch, které přiřazují licence z této sady.

1.5.9.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.
6. Uživatel ze seznamu rolí přidělujících licence vybere tu, kterou chce odebrat a klikne na tlačítko „Odebrat roli“.
7. Systém zobrazí varování, že odebráním role budou licence odebrány všem uživatelům s touto rolí.
8. Uživatel potvrdí odebrání role.
9. Systém přidá roli do seznamu.
10. Uživatel klikne na tlačítko „Uložit a zavřít“.
11. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.
12. Systém na pozadí provádí odebrání licencí uživatelům.

1.5.9.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.10 U10: Změna schvalovacího workflow licence

1.5.10.1 Cíl:

Změnit schvalovací workflow pro přidělení licence.

1.5.10.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel klikne na ikonu lupy nebo na název licenční sady.
5. Systém zobrazí formulář detailu licenční sady.

6. Uživatel vybere u položky „Schvalovací workflow“ ze seznamu schvalovacích workflow jednu z položek.
7. Uživatel klikne na tlačítko „Uložit a zavřít“.
8. Systém uloží změny a přesměruje uživatele na seznam dostupných licenčních sad.

1.5.10.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.11 U11: Odstranění stávající licenční sady

Cíl: Odstranit licenční sadu ze seznamu.

1.5.11.1 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Správa licencí“.
2. Uživatel vyhledá licenční sadu.
3. Systém na základě zadaného fragmentu názvu vyhledá licenční sadu.
4. Uživatel označí zaškrtnutím políčka licenční sadu.
5. Uživatel z nabídky hromadných akcí vybere možnost „Odstranit“.
6. Uživatel klikne na tlačítko „Provést akci“.
7. Systém zobrazí varování, zda chce uživatel skutečně smazat vybranou licenční sadu.
8. Uživatel potvrdí smazání licenční sady.
9. Systém odebere přidělené licence z dané sady uživatelům (dle způsobu přidělení), odstraní všechny dostupné licence a odstraní sadu.

1.5.11.2 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.12 U12: Přiřazení licenční sady roli z editace role

1.5.12.1 Cíl:

V rámci editace role správcem rolí nastavit této roli licenční sady, z nichž budou po přidělení role uživateli přiděleny i licence.

1.5.12.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Role“.
2. Uživatel vyhledá roli.
3. Systém na základě zadaného fragmentu názvu vyhledá roli.
4. Uživatel klikne na ikonu lupy nebo na název role.
5. Systém zobrazí formulář detailu role.
6. Uživatel klikne na záložku „Licence“.
7. Systém zobrazí záložku licence.
8. Uživatel klikne na tlačítko „Přidat licenci“.
9. Systém zobrazí uživateli pole s výběrem licenčních sad.
10. Uživatel vybere licenční sadu ze seznamu a klikne na tlačítko „Uložit a zavřít“.
11. Systém provede změny a přesměruje uživatele na seznam rolí.
12. Pokud již roli mají přidělenu uživatelé, systém jim přidělí příslušné licence.

1.5.12.3 Chybové stavy:

- V seznamu dostupných licencí není dostatek volných licencí pro přidělení uživatelům, v takovém případě je uživatel upozorněn standardní chybovou zprávou a licenční sada není roli přidělena.

1.5.13 U13: Odebrání licenční sady roli z editace role

.

1.5.13.1 Cíl:

V rámci editace role správcem rolí odebrat z nastavení této role licenční sady, z nichž přidělovala role při svém přidělení licence uživateli.

1.5.13.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne položku „Role“.
2. Uživatel vyhledá roli.
3. Systém na základě zadaného fragmentu názvu vyhledá roli.

4. Uživatel klikne na ikonu lupy nebo na název role.
5. Systém zobrazí formulář detailu role.
6. Uživatel klikne na záložku „Licence“.
7. Systém zobrazí záložku licence.
8. Uživatel klikne na tlačítko „Odebrat licenci“ u dané položky.
9. Systém zobrazí varování, že odebráním licence dojde k odebrání daných licencí u všech držitelů role..
10. Uživatel potvrdí akci a klikne na tlačítko „Uložit a zavřít“.
11. Systém provede změny a přesměruje uživatele na seznam rolí.
12. Na pozadí systém provede odebrání licencí uživatelům, kteří mají přidělenu danou roli.

1.5.13.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.14 U14: Žádost o přidělení licence

1.5.14.1 Cíl:

Uživateli požádá pro sebe, svého podřízeného nebo jiného spravovaného uživatele o přidělení licence z dostupných licenčních sad.

1.5.14.2 Hlavní scénář:

1. Uživatel přejde na profil uživatele a klikne na kartu „Přidělené licence“.
2. Uživatel klikne na tlačítko „Požádat o licenci“.
3. Systém zobrazí formulář pro podání žádosti o licenci.
4. Uživatel ze seznamu dostupných licencí vybere jednu, o kterou chce žádat, případně vyplní poznámku žádosti.
5. Uživatel klikne na tlačítko „Požádat“.
6. Systém spustí schvalovací workflow pro schválení přidělení žádosti.

1.5.14.3 Chybové stavy:

1.5.15 U15: Schválení přidělení licence

1.5.15.1 Cíl:

Poté, co uživatel požádal o přidělení licence, musí být tato žádost schválena nejprve jeho vedoucím (pokud není sám zároveň žadatelem), poté byznys vlastníkem licence (pokud není sám zároveň žadatelem).

1.5.15.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne na položku „Úkoly“.
2. Uživatel ze seznamu vybere daný úkol a klikne na ikonu lupy nebo na název úkolu.
3. Systém zobrazí detail žádosti o přidělení licence uživatel.
4. Uživatel klikne na tlačítky „Schválit“.
5. Systém schválí úkol a pokračuje ve workflow buďto vytvořením úkolu v dalším kole schvalování, nebo přidělením licence uživateli..
6. Systém přesměruje uživatele na seznam nevyřešených úkolů.

1.5.15.3 Chybové stavy:

- Uživatel zamítne přidělení licence, v takovém případě schvalovací proces končí a licence není přidělena.

1.5.16 U16: Odebrání licence uživateli

1.5.16.1 Cíl:

Uživatel odebere licenci sobě, svému podřízenému nebo jinému spravovanému uživateli.

1.5.16.2 Hlavní scénář:

1. Uživatel přejde na profil uživatele a klikne na kartu „Přidělené licence“.
2. Uživatel klikne na tlačítko „Požádat o licenci“.
3. Uživatel klikne na tlačítko „Odebrat licenci“.
4. Systém odebere ze seznamu danou položku.
5. Uživatel klikne na tlačítko „Uložit a zavřít“.
6. Systém spustí schvalovací workflow pro schválení přidělení žádosti a přesměruje uživatele na seznam uživatelů.

1.5.16.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.17 U17: Přiřazení licence uživateli na základě role

1.5.17.1 Cíl:

Uživatel zažádá pro sebe, svého podřízeného nebo jiného spravovaného uživatele o přidělení role, která licenci přiděluje.

1.5.17.2 Hlavní scénář:

1. Uživatel přejde na profil uživatele a klikne na kartu „Přiřazené role“.
2. Uživatel klikne na tlačítko „Změnit oprávnění“.
3. Uživatel v přehledu přidělených rolí klikne na tlačítko „Přidat“.
4. Systém zobrazí uživateli formulář pro výběr role.
5. Uživatel vybere z nabídky roli, která přiděluje licenci, a ostatní údaje (PPV, platnost přidělení od a do) a klikne na tlačítko „Nastavit“.
6. Systém zavře formulář pro výběr role a ve formuláři žádosti roli přidá.
7. Uživatel klikne na tlačítko „Podat žádost“.
8. Systém spustí schvalovací workflow žádosti o změnu oprávnění uživatele a přesměruje uživatele zpět na profil.
9. Oprávnění uživatelé schválí žádost v rámci celého procesu schvalování.
10. Systém přidá uživateli roli a touto rolí garantovanou licenci.

1.5.17.3 Chybové stavy:

- Oprávněný uživatel zamítne žádost o změnu oprávnění, v takovém případě systém roli nepřidělí, ani licenci.
- V licenční sadě není dostupná žádná volná licence a přidělení role je blokováno, v takovém případě.

1.5.18 U18: Odebrání licence uživateli na základě role

1.5.18.1 Cíl:

Uživatel sobě, svému podřízenému nebo jinému spravovanému uživateli odebere roli, která přiřazuje licence.

1.5.18.2 Hlavní scénář:

1. Uživatel přejde na profil uživatele a klikne na kartu „Přiřazené role“.
2. Uživatel klikne na tlačítko „Změnit oprávnění“.
3. Uživatel v nabídce odebere roli, která přiděluje licenci.
4. Uživatel klikne na tlačítko „Podat žádost“.
5. Systém spustí schvalovací workflow žádosti o změnu oprávnění uživatele a přesměruje uživatele zpět na profil.
6. Oprávnění uživatelé schválí žádost v rámci celého procesu schvalování.
7. Systém odebere uživateli roli a touto rolí garantovanou licenci.

1.5.18.3 Chybové stavy:

- Oprávněný uživatel zamítne žádost o změnu oprávnění (odebrání role), v takovém případě systém ponechá roli přidělenu a licenci neodebere.

1.5.19 U19: Zobrazení seznamu licenčních sad

1.5.19.1 Cíl:

Zobrazit uživateli seznam licenčních sad, na něž má oprávnění. V případě R3 pouze licenční sady, jichž je přihlášený uživatel byznys vlastníkem. V případě R6 všechny licenční sady v systému.

1.5.19.2 Hlavní scénář:

1. Uživatel v hlavní nabídce klikne na položku „Správa licencí“.
2. Systém mu seznam všech licenčních sad, na jejichž zobrazení má oprávnění.

1.5.19.3 Chybové stavy:

Tento scénář nemá žádné chybové stavy.

1.5.20 Architektura CzechIdM

1.5.20.1 Modularita CzechIdM

Jednou ze zásadních vlastností systému *CzechIdM* je jeho modularita. Dokonce i veškerá základní funkcionalita systému je rozdělena do samostatných modulů, které jsou z hlediska vývoje samostatnými projekty. Základními moduly *CzechIdM* jsou:

core Jádro aplikace; modul obsahuje základní služby, entity a RESTful rozhraní pro identity, role a stromové struktury (např. organizační stromy). Tento modul nelze deaktivovat.

acc Správa účtů; modul obsahuje služby, entity a RESTful rozhraní pro napojení systému, správu uživatelských účtů a dalších objektů na systémech. Obsahuje implementaci procesorů, starajících se o provisioning (propisování účtů na koncové systémy).

ic Konektor server a konektory; modul obsahuje identity connector server *ConnId*, třídy potřebné pro integraci *ConnId* v *CzechIdM* a všechny v systému dostupné *ConnId*-kompatibilní konektory pro napojování koncových systémů. Tento modul nelze deaktivovat.

example Vzor; modul slouží jako vzor pro vývojáře dalších rozšiřujících modulů, ve výchozím stavu je vypnutý.

vs Virtuální systémy; modul pro podporu správy oprávnění v systémech a aplikacích, které nejsou přímo připojeny k *CzechIdM*, ale požadované operace musí na základě úkolu v identity manageru vykonat jejich správce. Modul lze deaktivovat.

rpt Reporty; modul poskytuje služby, entity a RESTful rozhraní pro tvorbu reportů. Tento modul sám o sobě neobsahuje implementaci žádného reportu, nicméně poskytuje vývojářům doplňkových modulů veškeré nástroje pro tvorbu reportů specifických pro daný modul.

Součástí *CzechIdM* je i několik pomocných modulů, sloužících zejména jednoduššímu sestavení aplikace pomocí nástroje *Apache Maven*. Jsou to:

aggregator Slouží pouze jako agregátor projektů pro program *Apache Maven*, obsahuje moduly *app*, *parent* a *core*. Spuštěním *Maven* úkolu (*Maven Goal*[2]) na tomto projektu je zajištěno vykonání *Maven* úkolu stejného typu i na všech modulech.

app Webová aplikace *CzechIdM*; obsahuje moduly a koncové body RESTful rozhraní.

parent Rodičovský modul, na nějž mají závislost všechny moduly *CzechIdM*, včetně rozšiřujících.

gui Modul sloužící sestavení aplikace včetně klientské webové aplikace v jednom WAR souboru.

[2] *Maven Goal* reprezentuje specifický úkol v rámci sestavování aplikace programem *Apache Maven*, např. spuštění automatických testů.

Kromě těchto základních aplikačních modulů je k dispozici i řada dalších, doplňkových rozšiřujících modulů. Tyto moduly však nejsou dostupné bezplatně.

Architektura jednotlivých modulů odpovídá architektuře celé aplikace *CzechIdM*, i moduly jsou tedy rozděleny na serverovou a klientskou část. Mohou existovat moduly, které obsahují pouze serverovou část implementace a jelikož nevyžadují žádné úpravy uživatelského rozhraní, nemají žádnou samostatnou klientskou část. Obdobně mohou existovat speciální klientské moduly, které nemají vlastní implementaci serverové částí, nýbrž využívají již existujících RESTful přístupových bodů. Jednotlivé moduly mohou být zapnuty i vypnuty v uživatelském rozhraní *CzechIdM* bez nutnosti restartu.

Klientskou i serverovou část modulu lze navzájem provázat. Při aktivaci nebo deaktivaci serverové části modulu v GUI aplikace nebo pomocí konfigurační proměnné pak dojde zároveň k deaktivaci klientské části modulu. Tímto je zabráněno vzniku chybových stavů, kdy by zůstala aktivní pouze klientská část modulu a serverová by byla neaktivní.

1.5.20.2 Zpracování událostí v CzechIdM

Vedle modularity je další zásadní vlastností *CzechIdM* jeho událostmi řízená architektura (Event-Driven Architecture, EDA).[6] Mechanismus zpracování událostí umožňuje dle potřeby rozšířit nebo změnit základní chování aplikace pomocí modulů a změn konfigurace. Událost (*EntityEvent*) určitého typu (*EventType*) je publikována skrze službu *EntityEventManager* z různých míst aplikace (*hook*). Na jednu událost může reagovat více procesorů, rozšiřujících abstraktní třídu *AbstractEntityEventProcessor*, v různém pořadí (čím nižší číslo, tím dříve je procesor spuštěn). Procesory jsou ve výchozím nastavení *CzechIdM* spouštěny synchronně v jedné transakci. Pokud má více procesorů stejné pořadové číslo, budou spouštěny v náhodném pořadí – je dobrou praxí navrhnout a nastavit nekolizní pořadí. Namísto použití anotace `@Order` je třeba přetížit metodu `getOrder`. Předmětem (obsahem) události může být jakýkoliv serializovatelný (tj. implementující rozhraní *Serializable*) objekt, nicméně je preferován DTO (Data Transfer Object), který je potomkem abstraktní třídy *AbstractDto*. Událost bez kontextu nemůže existovat.

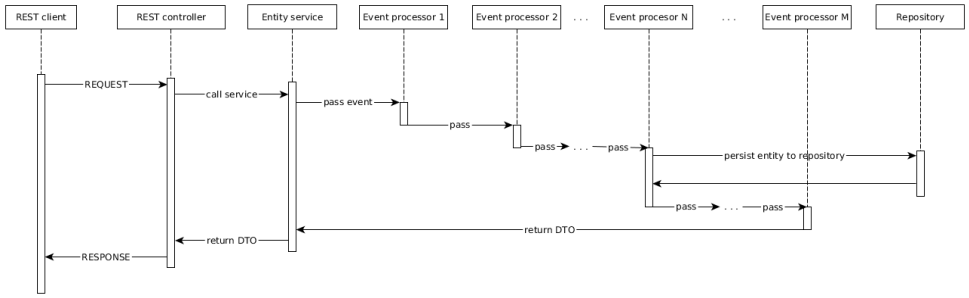
Životní cyklus události je následující:

1. Vznikne událost s daným kontextem

```
EntityEvent<IdmIdentityDto> event = new
    IdentityEvent(IdentityEventType.CREATE, createIdentity);
```

2. Událost je předána ke zpracování pomocí třídy *EntityEventManager*

```
EventContext<IdmIdentityDto> context =
    entityEventManager.process(event);
```



Obrázek 1.3: Životní cyklus události od požadavku REST klienta, přes zpracování kontrolerem a procesory až po uložení do repositáře.

3. Pokud je událost publikována a její předmět (obsah) je potomkem třídy *AbstractDto*, je do kontextu události vyplněn původní zdrojový DTO – obsahuje dříve uložené DTO a může být použit v procesorech např. ke zjištění modifikací. Pokud událost vytváří nový DTO, je do kontextu místo původního zdrojového DTO vyplněna hodnota `null`.

4. Návratová hodnota v kontextu události obsahuje výsledek všech procesorů na ni reagujících v pořadí definovaném pořadím procesorů. Procesor může událost označit jako ukončenou (closed) nebo pozastavenou (suspended), čímž dojde k přeskočení všech následujících procesorů. Pokud je přerušená událost pomocí třídy *EntityEventManager* opět publikována a pokud je zachován stejný kontext události, její zpracování bude pokračovat procesorem, který měl následovat po tom, jenž zpracování před tím přerušil. Pokud je zpracování události přerušeno, volaná metoda by měla na výstupu vrátit adekvátní stav `accepted`.

5. Poté, co událost projde procesorem, je inkrementován její čítač pořadí procesoru. Tento čítač je použit v případě přerušení a opětovném vyvolání události – zpracování události bude pokračovat procesorem s následujícím pořadovým číslem.

Modul pro správu licencí bude reagovat na události přidělení a odebrání role uživateli, dále pak na událost odstranění identity. V případě přidělení role uživateli bude muset zajistit přidělení licencí, které tato role přiděluje, v případě odebrání role uživateli bude muset zajistit odebrání licencí touto rolí přidělených. Při odstranění identity musí modul zajistit uvolnění/odebrání licencí odstraněnému uživateli. Jiných oblastí životního cyklu identity se modul dotýkat nebude.

Předmětem události může být i přidělení licence uživateli. Na tuto událost lze navázat specifický procesor, který zajistí integraci s externím licenčním serverem nebo systémem, kde je třeba licenci dále registrovat, či jinak zpracovat. K tomuto účelu se jako nejoptimálnější řešení jeví implementace spe-

cifického modulu, obsahujícího potřebné procesory, pro každý z napojených externích systému, s nimiž má být *CzechIdM* integrováno za účelem správy licencí. V rámci implementace modulu pro správu licencí je tedy nezbytné zajistit, aby při přidělení licence uživatel byla vyvolána událost, jíž bude tato licence předmětem (obsahem).

1.5.21 Možnosti rozšiřitelnosti CzechIdM

Jak již bylo zmíněno v předcházejících kapitolách, *CzechIdM* je modulární aplikace, jejíž sadu základních modulů lze libovolně rozšiřovat. Kromě doplňkových komerčních modulů je možné implementovat i vlastní moduly, v takovém případě je třeba brát v úvahu, že serverová a klientská část aplikace řeší modularitu odlišně. V podstatě jde o dvě samostatné aplikace komunikující spolu pomocí RESTful JSON API.

Modularita serverové části je zajištěna použitím frameworku *Spring Plugin*. Jednotlivé moduly lze přidávat nebo měnit buďto nakopírováním mezi knihovny aplikace do umístění `idm.war/WEB-INF/lib`, nebo přidáním přímé závislosti projektu v souboru `pom.xml` modulu *app* a následným sestavením aplikace a nasazením výsledného WAR souboru do *Java Servlet* kontejneru. Oba tyto přístupy jsou vzájemně ekvivalentní. Při startu aplikace jsou načteny a zaregistrovány deskriptory všech dostupných modulů. Tyto deskriptory implementují rozhraní *ModuleDescriptor* a obsahují metadata o jednotlivých modulech, zejména jsou to:

- unikátní identifikátor modulu
- verze modulu
- zda lze modul odebrat nebo deaktivovat
- oprávnění přidaná v rámci modulu
- popisné informace, jako je název, dodavatel, popis funkčnosti

Moduly dále mohou obsahovat databázové skripty. *CzechIdM* využívá nástroj *Flyway* pro vytváření a aktualizaci databázového schématu. Každý modul může obsahovat svou vlastní sadu skriptů, které budou aplikovány na databázové schéma při startu aplikace. Díky tomu moduly automaticky inicializují a aktualizují databázové schéma. Je požadováno, aby každý modul zasahoval pouze do své části schématu, tj. pouze do tabulek, které sám vytvořil a do žádných jiných nezasahoval.

Klientská část aplikace je sestavována nástrojem *Gulp*. V klientské části *CzechIdM* není použit žádný specializovaný nástroj pro podporu modularity, adresáře se zdrojovými kódy jednotlivých modulů, které jsou v sestavované klientské části požadovány, je třeba ručně přidat v adresáři projektu

do umístění `{project_home}/czechidm-app/czechidm-modules/` a do adresáře `{project_home}/node_modules/` přidat symbolické odkazy na tyto adresáře. Po sestavení klientské části je pak v umístění `{project_home}/dist` vytvořen balíček aplikace, ten je možno vzít a nasadit do libovolného webového serveru.

Většina modulů v serverové části *CzechIdM* se skládá ze tří *Maven projektů* (dále jen projektů). Prvním z nich je *idm-<kód modulu>*, který obsahuje pouze soubor `pom.xml`, definující závislosti na *parent* modul *CzechIdM* a případné další knihovny a frameworky modulu. Další částí je projekt *idm-<kód modulu>-api*, který obsahuje definice rozhraní, enumerací, DTO a filtrů a také deskriptor modulu. Třetím projektem je *idm-<kód modulu>-impl*, který obsahuje implementaci rozhraní, definovaných v rámci *idm-<kód modulu>-api*, veškerou další implementaci modulu včetně REST API, *Flyway* skripty, workflow a případné další komponenty.

CzechIdM poskytuje veškeré své funkce dostupné v GUI i skrze RESTful JSON API. Toto API lze využít z externích aplikací, které mohou jednak číst data z aplikace, jednak i aplikaci pomocí privilegovaného vzdáleně řídit. Dokumentace tohoto API vygenerovaná nástrojem *Swagger* je dostupná na adrese */idm/api*.

Rozšiřující modul *scim* je implementací SCIM 2.0 (System for Cross-domain Identity Management) serveru.[7] Specifikace SCIM 2.0 určuje, jak definovat RESTful JSON API, přístupné pomocí HTTP protokolu, pro práci s atributy uživatelů (identit) a skupin (rolí) a definuje jejich tři základní schémata: *User*, *EnterpriseUser* a *Group*.[8][9] Dále SCIM 2.0 definuje způsob, jak pomocí RESTful JSON API vystavit schéma atributů objektu i všechny dostupné objekty. Rozhraní SCIM 1.1 nebo 2.0 podporuje řada systémů z oblasti IAM (např. *Microsoft Active Direcotry*) i mimo tuto oblast.[10] Modul *scim* není standardní součástí *CzechIdM*, tato práce se mu tedy nebude dále věnovat, pouze je zmíněn jako jedna z dalších možností integrace.

1.6 Propojení se stávajícími funkcemi *CzechIdM*

1.6.1 Workflow engine

Součástí aplikace *CzechIdM* bez dalších rozšiřujících modulů jsou schvalovací procesy (workflow) pro schválení přiřazení role uživateli. Ke spouštění těchto workflow je používán interní workflow engine *Activiti BPM Platform*. Jak již jejich účel v aplikaci napovídá, tato workflow jsou uzpůsobena k použití v rámci žádosti o změnu oprávnění identity, nejsou tedy univerzálně využitelná v jiných funkcích aplikace. Z toho důvodu bude třeba implementovat nová, pro modul specifická workflow, která budou při žádosti o přidělení licence uživateli pomocí interního workflow enginu vykonávána. V případech, kdy bude přidělení licence navázáno na přidělení role uživateli, budou využita pro schválení přidělení role již existující základní workfow.

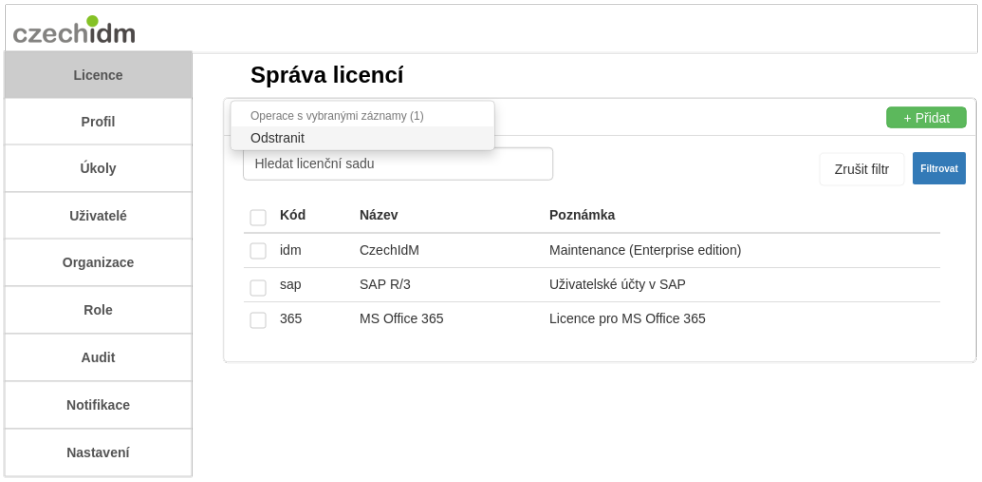
Activiti je workflow engine vyvíjený pod otevřenou licencí od roku 2010 *Activiti komunitou* a společností *Alfresco*. Slouží k zabudování podpory pro Business Process Management (BPM) do aplikace implementované v jazyce *Java*. *Activiti* podporuje otevřené standardy jako Business Process Model and Notation (BPNM) 2.0 a Decision Model and Notation (DMN) s otevřeným RESTful API pro procesy v organizaci. Definice procesů jsou ukládány v XML souborech s příponami `.bpmn20.xml`, jejich vizualizace v integrovaném vývojovém prostředí (Integrated development environment, IDE) má pak strukturu BPMN procesu. V aplikaci jsou uloženy definice procesů (workflow), z nichž je při spuštění vytvořena nová instance procesu, která je pak vykovávána. Pokud se tedy změní definice procesu ve chvíli, kdy některá instance procesu probíhá, není běh této instance změnou ovlivněn.

1.6.2 Auditování změn

Pro auditování změn na entitách v *CzechIdM* je využit modul *Hibernate Envers* z frameworku *Hibernate ORM*. Auditní záznamy jsou v databázi ukládány v tabulkách se stejným názvem, jako originální tabulky entity, pouze s příponou `_a`. Auditovány jsou všechny základní identity v *CzechIdM* s výjimkou těch, které být auditovány nesmí (např. entita *IdmPassword*, sloužící k ukládání hesel) nebo jejich auditování z hlediska logiky nedává smysl (např. entita *IdmAudit*, sloužící k auditování).

Jednotlivé tabulky auditních záznamů jsou propojeny skrze auditní entitu *IdmAudit*, nad kterou je postaveno standardní agenda s RESTful API, pomocí kterého lze auditní záznamy číst. Do *Hibernate Envers* je jako posluchač (listener) událostí zaregistrována třída *IdmAuditListener*, zajišťující vyplnění tabulek s historickými daty na základě všech událostí, jež *Hibernate Envers* vyhodnotí jako auditované. Dále je v *CzechIdM* implementována třída *AuditableEntityListener*, která zajišťuje zaznamenání metadat přímo k auditovaným entitám, jde např. o informace jako datum a čas vytvoření entity, datum a čas poslední modifikace entity, který uživatel entitu vytvořil, který uživatel entitu naposledy modifikoval atd. Třída *AuditableInterceptor* potom zajišťuje, aby údaje o datu a čase vytvoření entity a o tom, který uživatel entitu vytvořil, nebyly v budoucnu modifikovány.

V modulu *Hibernate Envers* je auditování změn na entitách zajištěno přidáním anotace `@Audited` před definici entity. V *CzechIdM* však tento postup není použit, namísto toho jsou anotací `@Audited` označeny jednotlivé atributy třídy, které mají být auditovány, viz následující ukázka. Takto budou anotovány rovněž atributy všech entit v modulu pro správu licencí. Tím bude zajištěno auditování změn a auditování přidělení licencí uživatelům.

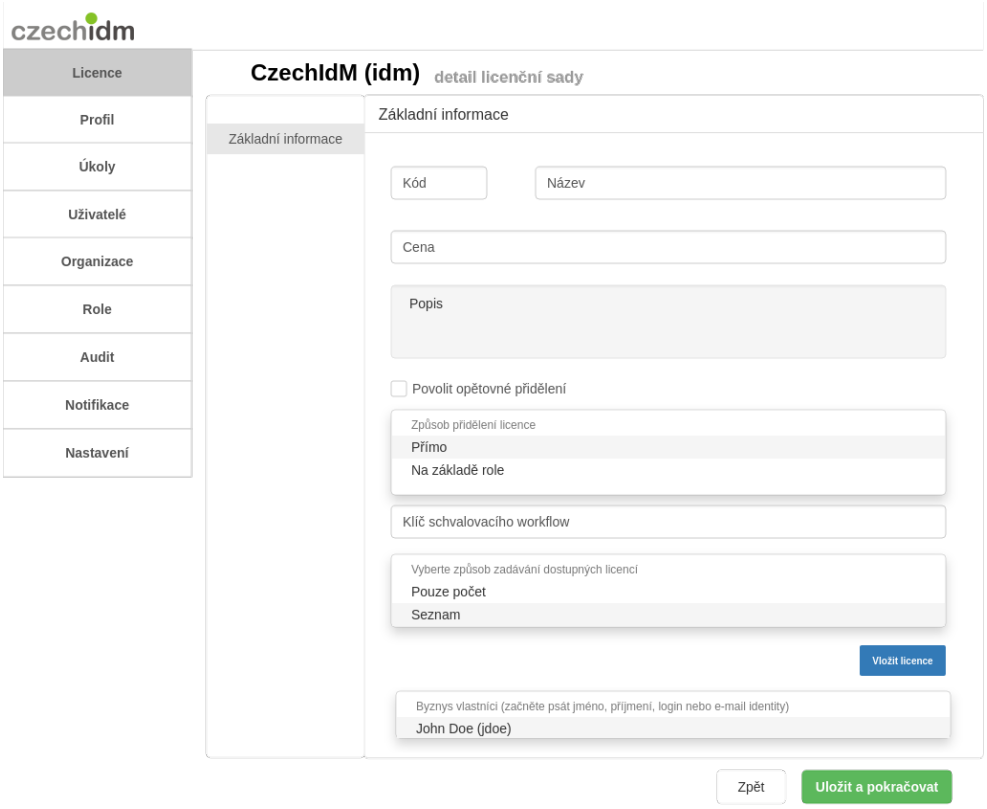


Obrázek 1.4: Stránka s přehledem licenčních sad z pohledu R3, R6, R7 a R8

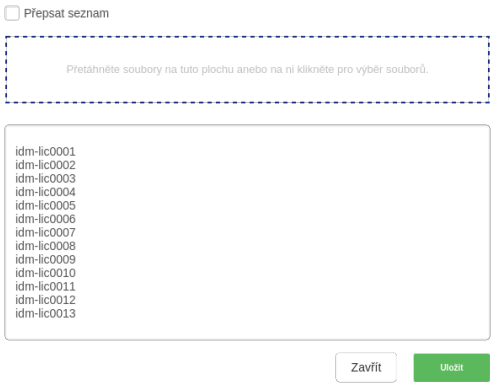
```
@Entity
@Table(name = "lic_license")
public class LicLicense extends AbstractEntity {

   private static final long serialVersionUID = 1L;

   @Audited
   @NotEmpty
   @Size(min = 1, max = DefaultFieldLengths.NAME)
   @Column(name = "serial_number", length = DefaultFieldLengths.NAME)
   private String serialNumber;
```

1.6.3 Reporty a přehledy

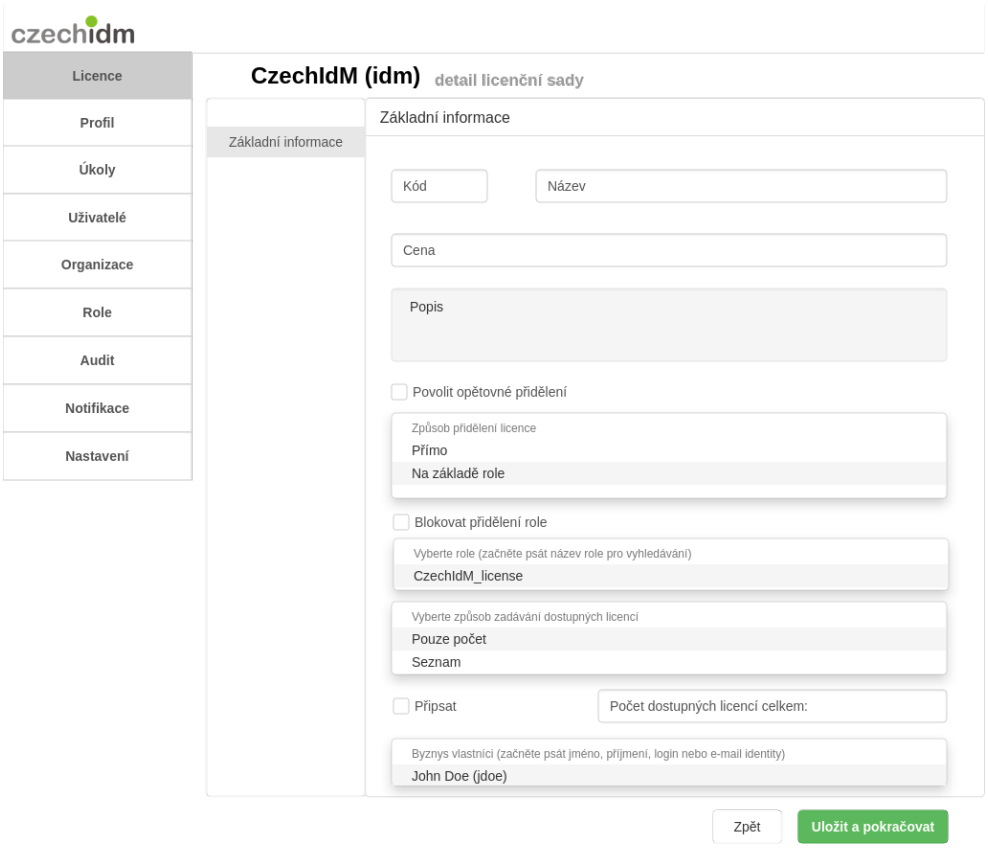
Jedním z identifikovaných požadavků na modul pro správu licencí je i generování reportů o licencích přidělených uživatelům. K tomuto účelu se jeví jako nejvhodnější využití modulu *rpt*, který je jedním ze základních modulů *CzechIdM*. Modul pro správu licencí bude obsahovat generátor reportu, který se po aktivaci modulu zobrazí v nabídce dostupných reportu na stránce „Reporty“. Tento generátor vytvoří report se seznamem uživatelů a licencí, které mají přiděleny. Oprávněný uživatel bude moci tento report vygenerovat a stáhnout.



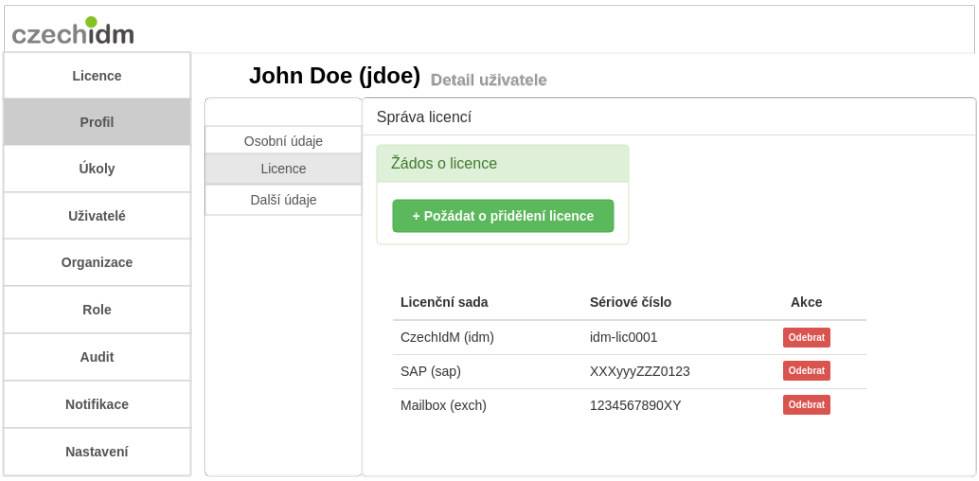
Obrázek 1.5: Detail (editace) licenční sady z pohledu R3, R7 a R8. Licence jsou přidělovány přímo, kódy (sériová čísla) licencí jsou zadávány pomocí seznamu.



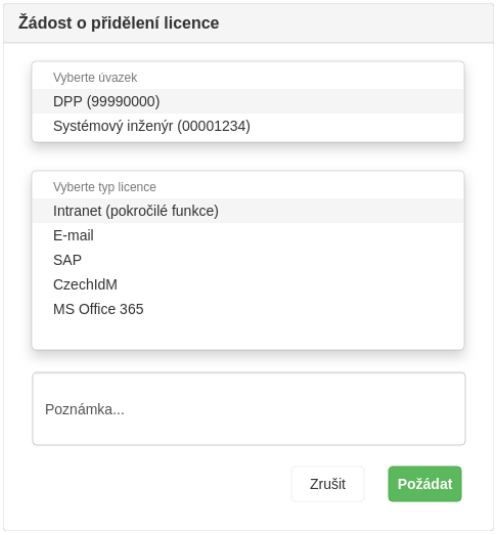
Obrázek 1.6: Formulář pro vyplnění seznamu licenčních čísel, nebo nahrání seznamu z CSV souboru z pohledu R3, R7 a R8.



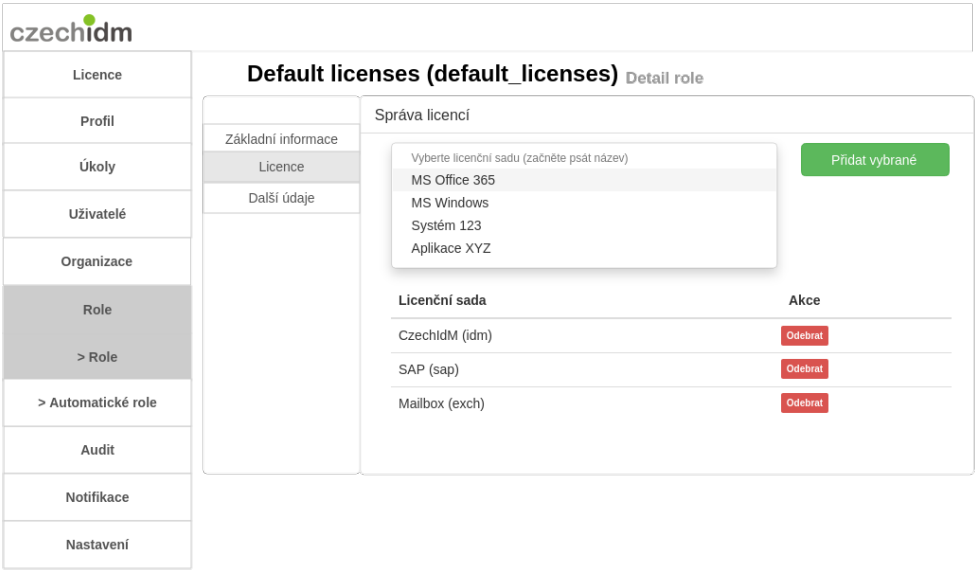
Obrázek 1.7: Detail (editace) licenční sady z pohledu R3, R7 a R8. Licence jsou přidělovány pomocí role, je zadán pouze počet dostupných licencí.



Obrázek 1.8: Karta „Správa licencí“ v profilu uživatele z pohledu R1, R2 a R4.



Obrázek 1.9: Formulář žádosti o přidělení licence uživateli z pohledu R1, R2 a R4..



Obrázek 1.10: Karta „Správa licencí“ v detailu role z pohledu R5.

1.7 Návrh uživatelského rozhraní

V rámci návrhu uživatelského rozhraní bylo analyzováno GUI základního jádra aplikace *CzechIdM* i existujících rozšiřujících modulů, které disponují vlastním GUI (některé mají pouze serverovou část, např. *sms*). Návrh uživatelského rozhraní zohledňuje konvence aplikace a počítá s maximálním využitím již existujících komponent.

1.7.1 Wireframe

Byl vytvořen wireframe model GUI modulu:

Seznam licencí – 1.4 představuje reprezentaci přehledu licenčních sad.

Detail licence – 1.5 a 1.7 představují formulář pro editaci licenční sady.

Vložení seznamu licencí – 1.6 představuje formulář pro vložení seznamu licencí, které licenční sada obsahuje.

Profil uživatele – 1.8 představuje záložku "Správa licencí"v profilu uživatele.

Žádost – 1.9 představuje formulář žádosti o přidělení licence z vybrané licenční sady.

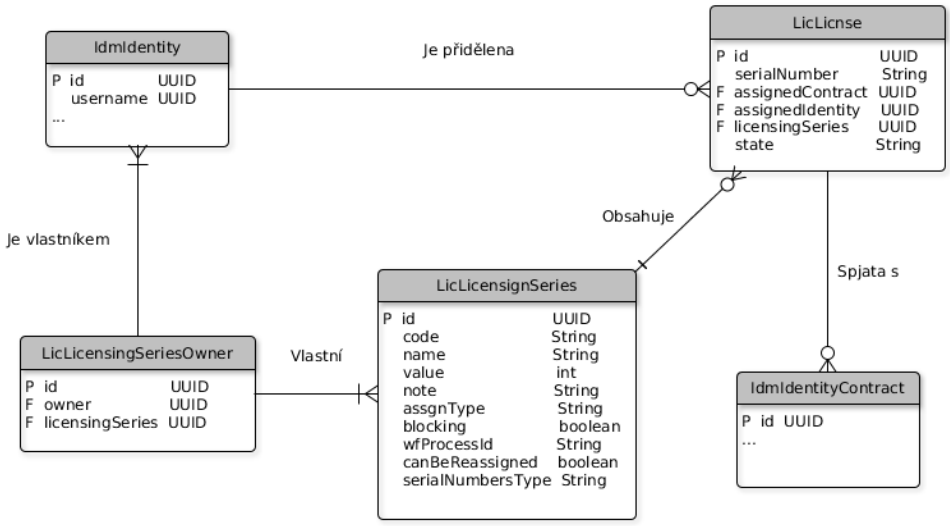
Detail role – 1.10 představuje záložku "Správa licencí"v detailu role, na základě jejíhož přidělení má být licence uživateli udělena.

1.8 Datový model

Entity, vytvořené v rámci této práce jsou prefixovány předponou *Lic-*. Důvodem je dodržení jmenné konvence *CzechIdM*. V datovém modelu (1.11) jsou doplněny rovněž entity z produktového jádra, s nimiž mají nové entity vazbu.

1.8.1 LicLicense

Entita *LicLicense* bude reprezentací jedné softwarové licence (např. jednoho sériového čísla). Bude udržovat informaci o tom, zda je dostupná (tedy lze přidělit uživateli), přidělena uživateli, nebo archivována (v minulosti byla přidělena uživateli a již nejde znovu přidělit), do které licenční sady patří a jaké je její sériové číslo. V případě licenčních sad, u nichž je evidován pouze počet dostupných licencí, bude licenční číslo generováno automaticky.



Obrázek 1.11: Datový model entit.

1.8.2 LicLicensingSeries

Entita *LicLicensingSeries* bude reprezentací jedné sady licencí. Jde o hlavní, a tedy i nejkomplexnější entitu datového modelu. Kromě základních popisných informací bude evidovat i referenci na technický název (identifikátor) schvalovacího worklow a svou konfiguraci: způsob přidělení, hodnotu, způsob zadání dostupných licencí a zda mohou být licence opětovně přiděleny.

1.8.3 LicLicensingSeriesOwner

LicLicensingSeriesOwner je pomocnou entitou, implementující M:N vztah mezi licenční sadou a jejím byznys vlastníkem.

1.8.4 Vazby na další entity

Entity z modulu pro správu licencí budou mít vazby i na další entity z modulu *core*, jmenovitě jde o:

IdmIdenitiy – reprezentace identity uživatele, který je byznys vlastníkem licenční sady. Tento vztah je reprezentován entitou *LicLicensingSeriesOwner*. Dále pak reprezentace identity, které je přidělena licence, tato informace je uložena v entitě *LicLicense*.

IdmIdentityContract – reprezentace úvazku, na nějž je uživateli přidělena licence, reprezentovaná entitou *LicLicense*.

Kapitola 2

Realizace

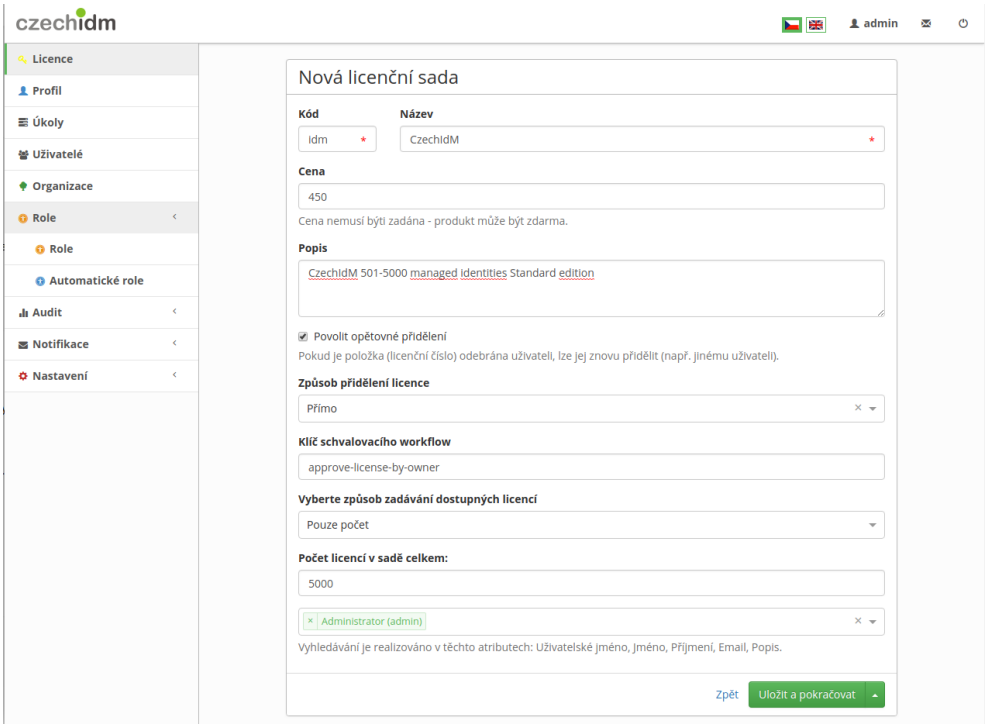
V této kapitole je popsána realizace modulu pro správu licencí. Kapitola je rozdělena na popis dvou částí – realizace serverové části modulu a realizace klientské části modulu. Obě části kapitoly mají obdobnou strukturu. Nejprve jsou popsány využité technologie, následuje popis implementace a popis způsobu automatického testování.

2.1 Serverová část

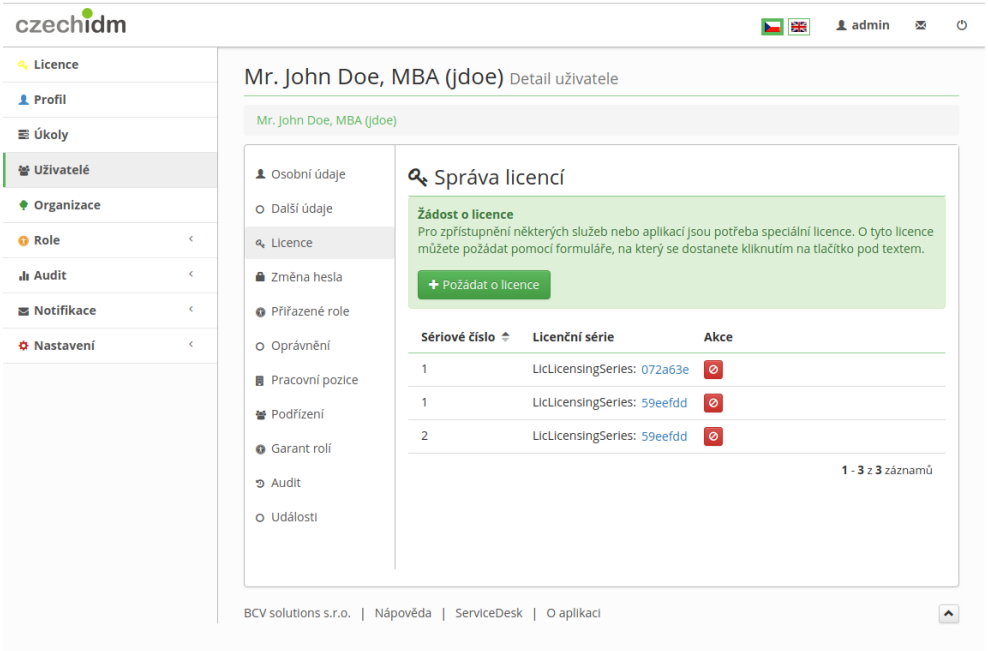
Serverová část modulu pro správu licencí je nezávislá na části klientské. Pro komunikaci s vnějším světem vystavuje RESTful JSON API. S ostatními moduly *CzechIdM* potom komunikuje pomocí jejich rozhraní, definovaných v `*-api` modulech. Z hlediska architektury modulů je doporučeno v ostatních modulech nepoužívat žádné třídy nebo jiné objekty z modulů `*-impl`, ale odkazovat se na ně pouze skrze rozhraní, definovaná v modulech `*-api`, u nichž je jako u jediných udržována zpětná kompatibilita.

2.1.1 Volba technologií

Volba technologií pro implementaci serverové části modulu je podmíněna architekturou a technologiemi použitými v aplikaci *CzechIdM*, pro kterou je modul implementován. Zejména jsou to framework *Spring*, zajišťující modularitu, framework *Hibernate ORM* jako implementace datové vrstvy a aplikace, RESTful JSON webové služby pro komunikaci s klientskou částí aplikace, implementované pomocí frameworku *RESTEasy* a jazyk *Java* verze 1.8. Pro testování jsou pak použity frameworky a nástroje *JUnit*, *Spring Testing* a *Mockito*. Pro běh aplikace je podporován a doporučen používat aplikační server *Apache Tomcat* verze 8 nebo vyšší, jediným podporovaným systémem řízení báze dat (Database Management System, DBMS) je *PostgreSQL* verze 9.3 a vyšší.



Obrázek 2.1: Screenshot: založení licenční sady



Obrázek 2.2: Screenshot: záložka s přehledem licencí v profilu uživatele

2.1.1.1 Spring Framework

Spring Framework je aplikační framework pro platformu *Java*, vyvíjený společností *Pivotal Software* od roku 2002. Tento framework obsahuje řadu užitečných funkcionalit, využívaných v aplikaci *CzehcIdM*. Jsou to:

Spring Boot je základní nástrojem pro tvorbu aplikací založených na frameworku *Spring*.

Spring Data je modul pro práci s relačními i NoSQL databázemi.[11] K datům může být přistupováno jednoduše pomocí vystaveného rozhraní, odpovídajícího daným konvencím. Rodina Spring Data obsahuje i řadu dalších modulů, pro tuto práci jsou relevantní zejména:

Spring Expression Language (SpEL) je výrazový jazyk (expression language) Syntaxe jazyka je podobná syntaxy Jednotného výrazového jazyka (Unified Expression Language, UEL), nabízí však další funkcionality,zejména šablonování textových řetězců.[12]

Spring Data JPA poskytuje podporu pro implementaci vrstvy pro přístup k datům srze Java Persistence API (JPA).

Sprind Data REST staví na *Spring Data* repositářích: analyzuje doménový model aplikace a vystaví HATEOAS (Hypermedia as the Engine of Application State) HTTP přístupové body (resource) pro agregáty obsažené v tomto modelu.[13]

Spring Security je framework pro autentizaci a řízení přístupu v aplikaci. Je de facto standardem pro aplikace založené na frameworku Spring.

Spring Plugin je framework umožňující snadné zajištění modularity aplikace. Framework nabízí pragmatičtější přístup, než např. framework *Open Services Gateway Initiative* (OSGi). Výhodou frameworku *Spring Plugin* vůči OSGi nebo implementacím architektury *Microservices* je jeho snadné použití, založené na definici rozhraní, jejich implementaci a registraci těchto implementací v aplikaci pro jejich použití. Nevýhodou je nutnost restartu aplikace (resp. aplikačního serveru) v případě, že je instalován nový zásuvný modul.

Spring Testing je framework, poskytující nástroje pro jednotkové testování a integrační testování.[14]

2.1.1.2 Querydsl

Querydsl je open-source framework, který umožňuje konstrukci typově bezpečných (type-safe) SQL-like dotazů nad řadou platforem včetně JPA, *MongoDB* a SQL v jazyce *Java*.[15] Podobně, jako JPA *Criteria* API používá procesor anotací Java 6 pro vytváření meta-modelu objektů, ale vytváří mnohem lépe

přístupné rozhraní. Další užitečná věc na tomto frameworku je že nejenže poskytuje podporu pro JPA, ale umožňuje i dotazování pomocí *Hibernate*, JDO, *Lucene*, JDBC a dokonce i prosté kolekce.[16]

2.1.1.3 JUnit

JUnit 5 je framework pro jednotkové testování v jazyce *Java 8*, skládající se z několika různých modlů a tří různých podprojektů:

JUnit Platform slouží jako základ pro spouštění testovacích frameworků na JVM. Také definuje *TestEngine* API pro vývoj testovacích frameworků, spouštěných na této platformě. Dále tato platforma poskytuje konzolový nástroj pro spouštění z příkazové řádky a umožňuje tak vytvoření modulů pro *Gradle* a *Apache Maven*.

JUnit Jupiter je kombinací nového programovacího modelu a rozšiřujícího modelu pro psaní testů a doplňků v JUnit 5. Projekt Jupiter poskytuje *TestEngine* rozhraní pro spouštění testů na dané platformě.

JUnit Vintage poskytuje rozhraní *TestEngine* pro spouštění testů pro *JUnit 3* a *JUnit 4* na dané platformě.

2.1.1.4 Mockito

Mockito je framework pro vytváření jednotkových testu v aplikacích implementovaných v jazyce *Java*, založený na použití návrhového vzoru *test spy*. Byl navržen pro intuitivní použití ve chvíli, kdy jednotkové testy potřebují tzv. „mocking“ – napodobení chování jiného objektu, na kterém jsou závislé. „Mocking“ spočívá ve vytvoření objektu, který svým chováním napodobuje chování originálního objektu, na kterém testovaná funkce závisí. Toto umožňuje testovat jednotlivé objekty izolovaně bez nutnosti vytváření instancí jiných objektů aplikace.

2.1.1.5 ORM

Objektově relační zobrazeni (Object-Relational Mapping, ORM) je programovací technika pro převod dat mezi systémy se vzájemně nekompatibilními modely reprezentace dat za použití jazyků pro objektově orientované programování. Většina moderních projektů pro vývoj podnikových aplikací využívá objektové technologie, jako jsou *Java* nebo *C#*, k vytváření aplikačního softwaru a relačních databází pro ukládání dat.[17] Hlavním cílem ORM je synchronizace mezi používanými objekty v aplikaci a jejich reprezentací v databázovém systému tak, aby byla zajištěna persistence dat.[18]

2.1.1.6 Hibernate ORM

Hibernate ORM je open-source framework distribuovaný pod otevřenou licencí, poskytující nástroje pro používání ORM v při vývoji aplikací v jazyce Java. Framework vyvíjí společnost *Red Hat*. Dále poskytuje implementaci JPA (Java Persistence API) specifikace, která je standardem pro ORM. Použití *Hibernate ORM* nevyžaduje po vývojáři velkou znalost jazyka SQL (Structured Query Language), jelikož implementuje práci s SQL a JDBC (Java Database Connectivity). Důležité je však porozumění principům datového modelování. *Hibernate ORM* se stará o mapování tříd v jazyce *Java* na databázové tabulky a mapování datových typů v jazyce *Java* na SQL datové typy. Navíc poskytuje nástroje dotazování se (načítání, vyhledávání) nad daty. *Hibernate ORM* je tedy vhodným řešením pro aplikace s objektově orientovaným doménovým modelem a byznys logikou implementovanou v aplikační vrstvě založené na technologii Java. *Hibernate ORM* také poskytuje odstínění specifických vlastností SQL databází různých dodavatelů.

2.1.1.7 REST

Representational State Transfer (REST) je koncept pro návrh distribuovaných systémů, navržený Royem Fieldingem v kapitole 5 jeho disertační práce *Architectural Styles and the Design of Network-based Software Architectures.*[19] Distribuovaná architektura v tomto smyslu znamená, že části programu běží na různých strojích a pro svojí komunikaci využívají síť.[20] Koncept REST není striktně svázán s *HyperText Transfer Protocol* (HTTP), ale je s ním nejčastěji spojován. Základními principy REST jsou:[21]

Výkon (Performance) – interakce komponent mohou být dominantním faktorem v uživatelem vnímaném výkonu a efektivitě sítě.

Škálovatelnost (Scalability) – podpora pro velké množství komponent a interakcí mezi nimi.

Jednoduchost (Simplicity jednotného rozhraní.

Modifikovatelnost (Modifiability) komponent v případě změn požadavků (i za běhu aplikace).

Viditelnost (Visibility) komunikace mezi komponentami pro agenty služeb.

Přenositelnost (Portability) komponent přesunutím části kódu programu spolu s daty.

Spolehlivost (Reliability) – odolnost vůči chybám na úrovni systému, dojde-li k chybě na úrovni komponenty, konektoru nebo dat.

Pro operace vytvoření, načtení, aktualizování a mazání (create, retrieve, update, delete – CRUD) lze využít HTTP metody:[21]

GET – Načtení informace. Dotaz pomocí metody GET musí být bezpečný a idempotentní, což znamená, že nehledě na tom, kolikrát bude metoda se stejnými parametry zavolána, výsledek bude vždy stejný. Metody mohou mít vedlejší efekty, ale uživatel je neočekává, tudíž nemohou být pro operaci na systému kritické. Dotazy mohou být také částečné nebo podmíněné.

Příklad načtení licence s ID hodnoty 1:

```
GET /licenses/1
```

POST – Dotaz pomocí metody POST na dané URI provede změnu zadané entity. Často je používán pro vytváření nových entit, ale lze ji využít i pro aktualizaci.

Příklad vložení nové licence:

```
POST /licenses
```

PUT – Dotaz pomocí metody PUT uloží entitu na zadanou URI. Metoda může vytvořit novou entitu nebo aktualizovat existující; je idempotentní. Idempotentnost je hlavním rozdílem mezi metodami PUT a POST – PUT přepíše existující entitu, pokud je metodě poskytnuta jen část elementů entity (např. atributů objektu), je zbytek nahrazen hodnotami `null`

Příklad modifikace licence s ID hodnoty 1:

```
PUT /licenses/1
```

PATCH – Dotaz pomocí metody PATCH aktualizuje pouze specifikovaná pole entity na zadané URI. Metoda je idempotentní.

Příklad aktualizace licence s ID hodnoty 1:

```
PATCH /licenses/1
```

DELETE – Dotaz pomocí metody DELETE odstraní zdroj na zadané URI, ale tento zdroj nemusí být odstraněn okamžitě. Může být odstraněn v rámci asynchronního dlouho běžícího úkolu.

Příklad odstranění licence s ID hodnoty 1:

```
DELETE /license/1
```

Aplikace, která používá v rámci svého REST API pouze HTTP metody GET, PUT, POST a DELETE, je označována jako *RESTful*.[22]

2.1.1.8 RESTEasy

RESTEasy je *JBoss* projekt, která poskytuje různé frameworky pro podporu vytváření RESTful Java aplikací. Je plně certifikovanou a přenositelnou implementací specifikace JAX-RS 2.1 a specifikace JCP, která poskytuje *Java* API pro RESTful webové služby nad HTTP protokolem. RESTEasy může být spuštěn v jakémkoliv Servlet kontejneru, ale pro zlepšení uživatelské zkušenosti a příjemnější prostředí je možná užší integrace s *WildFly* aplikačním serverem.[23]

2.1.1.9 Aplikační server

Apache Tomcat je open-source implementací technologií *Java Servlet*, *Java-Server Pages* (JSP), *Java Expression Language* a *Java WebSocket*, vyvíjenou organizací *Apache Software Foundation*. Skládá se z několika komponent:

Catalina – servlet kontejner, implementující specifikaci JSP.

Coyote – konektorová komponenta pro podporu protokolu HTTP 1.1 a webový server.

Jasper – JSP engine parsující JSP soubory a kompilující je do kódu jazyka Java jako servlety, které může spouštět *Catalina*.

Bylo testováno rovněž spouštění aplikace v aplikačním serveru z rodiny *JBoss*, konkrétně na verzích *WildFly 10.0.0*, *WildFly 10.1.0*, *WildFly 11.0.0* a *WildFly 12.0.0* s *CzechIdM 7.8.3*. Toto testování proběhlo úspěšně a aplikaci se podařilo spustit. S příchodem *CzechIdM 8.0.0*, na kterém probíhal vývoj modulu, však aplikace na těchto aplikačních serverech přestala fungovat. Předmětem této práce není zkoumání a zajištění přenositelnosti aplikace na jiné aplikační servery, než nativně podporovaný *Apache Tomcat*, proto tato problematika nebyla dále zkoumána.

2.1.2 Popis implementace

Implementace serverové části modulu je rozdělena do dvou samostatných projektů. Prvním z nich je *idm-lic-api*, obsahující definice rozhraní (*interface*) pro implementované třídy, enumerace (*enum*), popis modulu, implementace filtrů pro vyhledávání a implementace *Data Transfer Object* (DTO) tříd. Pomocí objektů v tomto projektu je udržována kompatibilita s jádrem aplikace a ostatními moduly. Druhým z nich je *idm-lic-impl*, obsahující implementaci rozhraní, definovaných v *idm-lic-api*, implementaci tříd, reprezentujících ORM entity a implementaci tříd (controller) RESTful API. Oba tyto moduly jsou nadefinovány jako moduly rodičovského projektu *idm-lic*, v němž jsou zároveň definovány závislosti na aplikaci, její další moduly a případné externí knihovny.

2.1.2.1 Modul idm-lic-api

Jak již bylo nastíněno výše, tento modul obsahuje třídy, rozhraní a enumerace, definuje rozhraní pro komunikaci rozšiřujícího modulu s jádrem aplikace a s ostatními moduly. Je členěn do následujících balíčků:

eu.bcvsolutions.idm.lic Balíček obsahuje rozhraní:

- **LicModuleDescription** – Jedná se o rozhraní pro „deskriptor“ modulu, definuje unikátní ID modulu v rámci celé aplikace a rozšiřuje rozhraní *ModuleDescriptor* z modulu *idm-core*.

eu.bcvsolutions.idm.lic.api.domain – Balíček obsahuje veškeré enumerace:

- **LicAssignType** – Způsob přidělení licence.
- **LicLicenseGroupPermission** – Definuje oprávnění pro práci s entitou *LicLicense*, implementuje rozhraní *GroupPermission* z modulu *idm-core*.
- **LicLicenseStateType** – Definuje stav přidělení licence.
- **LicLicensingSeries** – Definuje oprávnění pro práci s entitou *LicLicensingSeries*, implementuje rozhraní *GroupPermission* z modulu *idm-core*.
- **LicSerialNumbersType** – Definuje způsob zadávání licenčních čísel.

eu.bcvsolutions.idm.lic.api.dto Balíček obsahuje třídy DTO objektů pro práci s datovými entitami:

- **LicLicenseDto** – DTO reprezentující entitu *LicLicense*, obsahuje všechny její atributy, žádné jiné.
- **LicLicensingSeriesDto** – DTO reprezentující entitu *LicLicensingSeries*, obsahuje všechny její atributy, navíc je při vzniku doplněno o další popisné informace, jako je počet dostupných licencí, seznam licenčních čísel, počet přidělených licencí a celkový počet licencí. Při ukládání pak jsou vyhodnoceny změny těchto atributů a provedeny na ně navazující akce.
- **LicLicensingSeriesOwnerDto** – Objekt reprezentující vazbu mezi *LicLicensingSeries* a *IdmIdentity*.

Všechny tyto třídy rozšiřují abstraktní třídu *AbstractDto* z modulu *idm-core*. DTO objekty slouží jako reprezentace tříd *LicLicense*, *LiceLicensingSeries* a *LicLicensingSeriesOwner* v celé aplikaci. K převodu těchto DTO objektů na instance tříd datové vrstvy jsou implementovány speciální služby (service), viz dále.

eu.bcvsolutions.idm.lic.api.filter Balíček tříd, rozšiřujících *DataFilter* z modulu *idm-core*, sloužící k vyhledávání objektů v databázi:

LicLicenseFilter – Filtr pro vyhledávání jednotlivých licencí – entit *LicLicense*.

LicLicensingSeriesOwnerFilter – Filtr pro vyhledávání licenčních sad – entit *LicLicensingOwnerSeries*.

LicLicensingSeriesFilter – Filtr pro vyhledávání licenčních sad – entit *LicLicensingSeries*.

LicLicensingSeriesOwnerFilter – Filtr pro vyhledávání pomocných objektů, reprezentujících vazbu mezi licenční sadou (*LicLicensingSeries*) a jejím vlastníkem (*IdmIdentity*) – entit *LicLicensingSeriesOwner*.

eu.bcvsolutions.idm.lic.api.service Balíček obsahující definice rozhraní pro služby (service, servlety) pro práci s DTO licenčního modulu.

LicLicenseManager – Rozhraní pro servlet pro práci s *LicLicenseDTO*.

LicLicenseService – Rozhraní pro službu pro CRUD operace s *LicLicenseDto*, rozšiřují rozhraní *ReadWriteService*, *CodeableService* a *AuthorizableService* z modulu *idm-core*.

LicLicensingSeriesManager – Rozhraní pro servlet pro práci s *LicLicensingSeriesDto*, rozšiřují rozhraní *ReadWriteService*, *CodeableService* a *AuthorizableService* z modulu *idm-core*.

LicLicensingSeriesService – Rozhraní pro službu pro CRUD operace s *LicLicensingSeriesDto*.

LicLicensingSeriesOwnerManager – Rozhraní pro servlet pro práci s *LicLicensingSeriesOwnerDto*.

LicLicensingSeriesOwnerService – Rozhraní pro službu pro CRUD operace s *LicLicensingSeriesOwnerDto*, rozšiřují rozhraní *ReadWriteService*, *CodeableService* a *AuthorizableService* z modulu *idm-core*.

2.1.2.2 Modul idm-lic-impl

Jak již bylo nastíněno výše, tento modul obsahuje implementace potřebných rozhraní z modulu *idm-lic-api*. Zejména se jedná o implementace servletů služeb pro CRUD operace a manažerů pro DTO, dále o implementace entit, implementace jednotkových testů, implementace REST API a konfigurační soubory modulu. Je členěn do následující struktury balíčků.
Podadresář `src/main/java`**:**

eu.bcvsolutions.idm.lic Balíček obsahuje implementaci základních tříd balíčku:

DefaultLicenseModuleDescription – „Desktiptor“ modulu, nutný pro jeho instalaci do *CzechIdM*, implementuje rozhraní *LicModuleDescriptor* a rozšiřuje abstraktní třídu *PropertyModuleDescriptor* z *idm-core*.

LicenseModuleInitializer – Slouží k inicializaci modulu při jeho aktivaci, implementuje rozhraní *ApplicationListener<ContextRefreshedEvent>* z modulu *idm-core*.

eu.bcvsolutions.idm.lic.config.swagger Balíček pro konfiguraci nástroje *Swagger*:

LicenseSwaggerConfig – Třída obsahující konfiguraci pro nástroj *Swagger*, rozšiřuje abstraktní třídu *AbstractSwaggerConfig* z modulu *idm-core*.

eu.bcvsolutions.idm.lic.config.flyway Balíček pro konfiguraci nástroje *FlyWay*:

LicFlywayConfig – Třída obsahující konfiguraci pro nástroj *FlyWay*, rozšiřuje abstraktní třídu *AbstractFlywayConfiguration* z modulu *idm-core*.

eu.bcvsolutions.idm.lic.entity Balíček z třídami implementujícími entity datové vrstvy.

LicLicense – Třída rozšiřuje abstraktní třídu *AbstractEntity* z modulu *idm-core*. Jedná se o implementaci entity, jenž je pomocí *Hibernate ORM* ukládána v tabulce `lic_license`.

LicLicensingSeries – Třída rozšiřuje abstraktní třídu *AbstractEntity* z modulu *idm-core*. Jedná se o implementaci entity, jenž je pomocí *Hibernate ORM* ukládána v tabulce `lic_licensing_series`.

LicLicensingSeriesOwner – Třída rozšiřuje abstraktní třídu *AbstractEntity* z modulu *idm-core*. Jedná se o implementaci entity, jenž je pomocí *Hibernate ORM* ukládána v tabulce `lic_licensing_series_owner`.

eu.bcvsolutions.idm.lic.event Balíček obsahující procesory událostí modulu *lic*. Všechny procesory rozšiřují třídu *CoreEventProcessor* a implementují rozhraní *IdentityContractProcessor*.

ContractDeleteProcessor Procesor, reagující na smazání kontraktu (PPV) identity. Zajišťuje odebrání licencí, které měla identita na tento kontrakt přiděleny.

ContractEndProcessor Procesor, reagující na ukončení platnosti časově omezeného kontraktu (PPV) identity. Zajišťuje odebrání licencí, které měla identita na tento kontrakt přiděleny.

eu.bcvsolutions.idm.lic.repository Balíček rozhraní, rozšiřující rozhraní *AbstractEntityRepository* z balíčku *idm-core*, pro vyhledávání objektů v repositáři:

- **LicLicenseRepository** – Rozhraní pro vyhledávání objektů *LicLicense*.
- **LicLicensingSeriesRepository** – Rozhraní pro vyhledávání objektů *LicLicensingSeries*.
- **LicLicensingSeriesOwnerRepository** – Rozhraní pro vyhledávání objektů *LicLicensingSeriesOwner*.

eu.bcvsolutions.idm.lic.rest Balíček REST API:

- **LicLicenseController** – Implementace REST API pro objekt *LicLicenseDto*.
- **LicLicensingSeriesController** – Implementace REST API pro objekt *LicLicensingSeriesDto*.

eu.bcvsolutions.idm.lic.service Balíček implementací servletů pro práci s DTO:

- **LicLicenseController** – Implementace CRUD servletu pro objekt *LicLicenseDto*.
- **LicLicensingSeriesOwnerController** – Implementace CRUD servletu pro objekt *LicLicensingSeriesDto*.
- **LicLicensingSeriesController** – Implementace CRUD servletu pro objekt *LicLicensingSeriesOwnerDto*.

Podadresář `src/main/resources`:

***.properties** Soubory s konfiguračními proměnnými modulu.

eu.bcvsolutions.idm.sql Obsahuje *FlyWay* skripty pro inicializaci databáze při instalaci modulu.

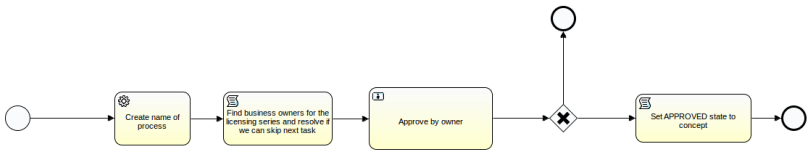
eu.bcvsolutions.idm.workflow V balíčku se nachází workflow *approveLicenseByBusinessOwner.bpmn20.xml* procesu schválení přidělení licence jejím byznys vlastníkem.

Podadresář `src/test/java`:

Obsahuje implementaci jednotkových testů.

Podadresář `src/test/resources`:

Obsahuje soubory s konfigurací jednotkových testů.



Obrázek 2.3: Workflow procesu *approveLicenseByBusinessOwner (approve-license-by-business-owner)* pro schválení přidělení licence jejím byznys vlastníkem.

2.1.2.3 Schvalovací proces

Bylo implementováno schvalovací workflow schválení přidělení licence 2.3 s unikátním identifikátorem *approve-license-by-business-owner*. Proces na vstupu dostane jako parametry identifikátor licenční sady, identifikátor kontraktu identity a uživatelské jméno identity. Následně vypočítá řešitele schvalovacího úkolu, kterými jsou všichni byznys vlastníci licenční sady, a vytvoří úkol. V případě schválení úkolu se pokusí přidělit licenci uživateli. Pokud není žádná licence dostupná, proces licenci nepřidělí a končí. Pokud je dostupná licence, proces ji přidělí uživateli. V případě zamítnutí proces končí a licence není přidělena.

2.1.3 Automatické testy

V projektu *idm-lic-impl* v podadresáři `srv/test` se nachází implementace jednotkových a integračních testů modulu. Testy pokrývají významné veřejné metody služeb, manažerů a „kontrolerů“, které byly v rámci implementace přetíženy. Testy nepokrývají metody, které v rámci implementace pouze volají metodu předka.

2.2 Klientská část

Klientská část modulu je závislá na té serverové, ačkoliv se jedná o samostatnou aplikaci. Zejména potřebuje RESTful JSON API serverové části modulu, s nímž komunikuje. Bez aktivní serverové části nelze modul používat.

2.2.1 Volba technologií

Volba technologií pro implementaci klientské části modulu je podmíněna architekturou a technologiemi použitými v aplikaci *CzechIdM*, pro kterou je modul implementován. Zejména jsou to knihovny *RactJS/Redux*, poskytující sadu nástrojů pro vytvoření webové aplikace v jazyce *JavaScript*. Zejména jsou využity k udržování informace o stavu aplikace, zpracování událostí a navigaci mezi stránkami. Dále pak testovací nástroje *Mocha* a *Chai* a balíčkovací systém *npm*. Nakonec pak nástroj *Gulp.js*, který slouží k sestavení celé aplikace. V rámci implementace modulu nebyl tento seznam technologií nikterak rozšířen.

2.2.1.1 React a Redux

React je knihovna v jazyce *JavaScript* pro tvorbu uživatelských rozhraní vyvíjená společnostmi *Facebook*, *Instagram* a vývojářskou komunitou. Umožňuje implementaci jednoduchých pohledů pro každý stav aplikace a při změně dat aktualizovat a překreslit právě danou komponentu. *React* je založen na komponentách, které řídí svůj stav a jejichž skládáním lze vytvořit komplexní uživatelské rozhraní (User Iterface, UI). Logika komponent je implementována v jazyce *JavaScript* namísto šablon, tudíž lze jednoduše zpracovávat složitější datové objekty a udržovat tak stav aplikace mimo *Document Object Model* (DOM).

Oproti jiným knihovnám, manipulujícím přímo s DOM, má *React* několik výhod. V první řadě usnadňuje práci vývojářům tím, že pomocí svého virtuálního DOM vytváří uživatelsky přívětivější rozhraní nad skutečným DOM. Dále umožňuje vývojářům deklarativně popisovat jejich uživatelské rozhraní a modelovat stav těchto rozhraní. Vývojář pouze popíše konečný stav rozhraní po provedení transakce pomocí funkce a *React* se na základě toho postará o aktualizaci UI ve chvíli přechodu do daného stavu. Nakonec je to relativně malá náročnost React na vstupní znalosti vývojáře. *React* disponuje velmi malým API, které si vývojář musí osvojit, jinak si vývojář vystačí se znalostmi jazyka *JavaScript*.[24]

Redux je open-source knihovna pro jazyk *JavaScript* řídící stav aplikace. Pomáhá s vývojem aplikací, které se chovají konzistentně v různých prostředích (klientech, serverech i nativně) a jsou snadno testovatelné. *Redux* lze používat společně s *React* nebo s jakoukoliv jinou knihovnou.

ReactTestUtils je knihovna v jazyce *JavaScript* z frameworku *React*. Slouží ke zjednodušení testování *React* komponent v libovolném testovacím frameworku.

2.2.1.2 Mocha

Mocha je famework pro testování v jazyce *JavaScript* spouštěný na *Node.js* i v prohlížeči. Sériově spouštěné *Mocha* testy umožňují flexibilní a přesná hlá-

šení při zjišťování nepodchycených výjimek v rámci testovacích scénářů.

2.2.1.3 Chai.js

Chai.js (Chai) je knihovna v jazyce *JavaScript*, poskytující „assertions“ pro libovolný testovací framework, poskytující podporu pro chováním řízený vývoj (Behavior-driven development, BDD) a testy řízený vývoj (Test-driven development, TDD) Je používána podobně, jako funkce `assert` v *Node.js*. Knihovna je spouštěna v *Node.js* nebo v prohlížeči.

2.2.1.4 npm

npm je správce balíčků pro vývojové prostředí jazyka *JavaScript*. Je výchozím správcem balíčků pro prostředí *Node.js*. Skládá se z klienta příkazové řádky (`npm`) a online databáze veřejných i placených balíčků (*npm registry*).

2.2.1.5 Gulp.js

Gulp.js je sada open-source nástrojů, postavená na technologiích *Node.js* a *npm*, používaná pro automatizaci časově náročných a opakujících se úkolů, jako např. minimalizace (Minification), zřetězování, vypínání cache (cache busting), jednotkové testování, „linting“ – analyzování kódu a označování programátorských, softwarových a stylistických chyb a podezřelých konstruktů –, optimalizaci atd., ve vývojovém procesu webových aplikací. Je používán k sestavování klientských webových aplikací.

2.2.2 Popis implementace

Byla implementována klientská aplikace *czechidm-lic* jako rozšiřující modul aplikace *czechidm-app*. Modul je členěn do stejné adresářové struktury, jako jiné moduly pro *CzechIdM* a skládá se z následujících částí:

src/components Adresář, slouží k umístění modulově specifických GUI komponent. Tento modul žádné takové komponenty neobsahuje

src/content Adresář, v němž jsou umístěny jednotlivé obsahové stránky.

src/content/licenses Adresář se stránkami pro vykreslení karty licencí v profilu uživatele

- **IdentityLicenses** Karta licencí uživatele, obsahuje *LicenseTable*.
- **LicenseTable** Tabulka pro zobrazení seznamu licencí, rovněž obsahuje formulář žádosti o přidělení licence.

src/content/licensingseries Adresář se stránkami pro vykreslování agendy licencí

LicensingSeries Stránka seznamu licencí, obsahuje *LicensingSeriesTable*, je definována v *routes.js*.

LicensingSerisContent Stránka pro vykreslení detailu licenční sady, obsahuje *LicensingSeriesDetail*.

LicensingSeriesDetail Formuláře pro editaci licenční sady, obsahuje všechny její atributy a všechny ovládací prvky formuláře a potřebné pomocné funkce.

LicensingSeriesRoute Obalová stránka pro detail licence, zajistí vykreslení všech svých potomků, je definována v *routes.js*.

LicensingSeriesTable Tabulka všech dostupných licenčních sad.

src/domain Adresář určený pro umístění enumerací. Tento modul žádné enumerace neobsahuje.

src/locales Obsahuje soubory *cs.json* a *en.json* s českou a anglickou lokalizací textací pro GUI.

src/redux Adresář obsahuje třídy pro práci s entitami (manažery).

LicenseManager Manažer pro práci s entitou *LicLicense* skrze *LicLicenseDto*.

LicensingSerisManager Manažer pro práci s entitou *LicLicensingSeries* skrze *LicLicensingSeriesDto*.

src/services Adresář obsahuje implementace tříd pro komunikaci s REST API serverové části.

LicenseManager Rozhraní pro komunikaci s REST API entity *LicLicense*.

LicensingSerisManager Rozhraní pro komunikaci s REST API entity *LicLicensingSeries*.

test Adresář obsahuje test lokalizací.

component-descriptor.js Slouží k popisu modulově specifických GUI komponent.

module-descriptor.js Obsahuje popis modulu, a jeho jednotlivých prvků. Zde jsou definovány položky v nabídkách, které modul doplní a relativní cesty stránek, na které je uživatel po kliknutí přesměrován.

package.json Popis modulu.

routes.js Obsahuje mapování jednotlivých relativních cest na stránky ze `src/content`.

2.2.3 Automatické testy

V rámci této práce byly implementovány pouze testy pro kontrolu lokalizací. Tento test je spouštěn pokaždé při sestavování aplikace a kontroluje, zda jsou lokalizace pro jednotlivé jazyky kompletní. Pokud ne, nedovolí aplikaci sestavit. Princip kontroly spočívá v tom, že speciální skript projde soubory s lokalizacemi a porovná seznamy jejich klíčů. V modulu je implementována funkce s definovanou cestou k lokalizačním souborům, nad kterými spustí test.

Kapitola 3

Uživatelské testování

Tato kapitola popisuje testování prvního prototypu modulu pro správu licencí. Nejdříve jsou popsány dostupné funkce prototypu před testováním a způsob testování. Následně jsou vyhodnoceny jeho výsledky.

3.1 Postup testování

Jelikož byl v okamžiku testování k dispozici pouze prototyp první fáze implementace modulu, byla testována pouze základní funkcionalita. Zároveň však byly diskutovány funkce, a to jak ty již dostupné, tak ty již plánované, přičemž závěry z těchto diskusí jsou také součástí výsledků testování. Testování probíhalo pomocí předem daných scénářů na počítači s nainstalovanou a spuštěnou aplikací *CzechIdM*.

Testování se účastnili dva testeři, které lze klasifikovat jako velice pokročilé znalce aplikace i znalce problematiky IAM. V důsledku toho nelze očekávat odhalení nedostatků, ke které by přišly testeři, kteří aplikaci neznají a nezabývají se problematikou IAM. Nicméně pro potřeby otestování prvního prototypu, který ještě dozná řady změn a rozšíření, než bude nasazen do produkčního provozu, bylo toto testování shledáno jako dostačující.

3.1.1 Testovací scénáře

T1 Založení nových licenčních sad: Cílem je založit nejméně jednu licenční sadu, která má zadán pouze počet dostupných licencí a nejméně jednu, která má zadán seznam licenčních čísel.

T2 Správce požádá o licenci pro uživatele: Cílem je jako správce licencí zažádat o přidělení licence pro některého z uživatelů.

T3 Schválení přidělení licence: Cílem je jako garant licence najít žádost o schválení přidělení licence a tuto schválit.

3.2 Vyhodnocení testování

Testeři během testování intuitivně postupovali scénáři a bez větších zaváhání se jim je podařilo projít. Následně ještě namátkově klikali v aplikaci a zkoušeli jednotlivé funkce. Výsledkem testování je následující výčet připomínek a požadavků:

1. Oprávnění uživatele nejsou dostatečně jemná. Pokud má uživatel oprávnění na čtení se skupiny oprávnění „Licence“, vidí zároveň záložku pro správu licencí a seznam všech licenčních sad včetně jejich detailů.
2. Chybové hlášky jsou nesrozumitelné.
3. Formulář pro editaci licenční sady není příliš přehledný.
4. Změnit pořadí pro položku „Licence“ v hlavní nabídce (nyní je první).
5. Změnit pořadí záložky „Licence“ v profilu uživatele (nyní je třetí), vhodnější umístění by bylo níže, než je záložka „Pracovní pozice“.
6. Zamyslet se, jak vyřešit situaci, kdy sice nemám k dispozici volnou licenci, ale přesto chci roli přidělit a účet na systému, byť v omezeném rozsahu, založit.

Kapitola 4

Dokumentace

V této kapitole je zdokumentován modul pro správu licencí. Je zde popsán postup sestavení serverové i klientské části aplikace. Dále pak instalace modulu do aplikace *CzechIdM* a dokumentace klíčových metod a koncových bodů REST API serverové části modulu.

4.1 Postup instalace serverové části

V tomto návodu modul nejdříve sestavíme pomocí nástroje *Apache Maven*. Poté modul nainstalujeme do již připraveného balíčku s aplikací *CzechIdM*. Další možností by bylo sestavení aplikace *CzechIdM* spolu s modulem najednou tak, že modul přidáme do závislostí v souboru aplikace. To však v současné době není doporučovaný postup. Obdobně by bylo možné si vytvořit nový projekt a v jeho souboru `pom.xml` závislosti definovat, nevýhodou by však opět byla nutnost ručně zajišťovat aktuálnost tohoto projektu při vydání nové verze aplikace nebo modulu. V návodu dále předpokládáme již existující instalaci *CzechIdM*.

4.1.1 Sestavení modulu

Jelikož modul pro správu licencí nebude volně dostupný, návod pro sestavení modulu předpokládá, že uživatel, který jej používá, má přístup do systému *Nexus* společnosti BCV solutions s. r. o a zde oprávnění ke čtení všech potřebných závislostí. V případě, že by uživatel tyto přístupy neměl, bude nejprve muset všechny tyto závislosti vyřešit, a to včetně stažení a sestavení aplikace *CzechIdM* v příslušné verzi.

1. Otevřete terminál a přejdete do domovského adresáře projektu *idm-lic*
2. `mvn clean install`

3. V adresáři `idm-lic-api/target` naleznete sestavený modul v souboru `idm-lic-api-8.0.0.jar`

4. V adresáři `idm-lic-impl/target` naleznete sestavený modul v souboru `idm-lic-impl-8.0.0.jar`

4.1.2 Instalace modulu do CzechIdM

Kroky 1, 3, 4 a 5 tohoto postupu není třeba provádět v případě, že bychom aplikaci *CzechIdM* sestavili s modulemx *idm-lic-api* a *idm-lic-impl* v závislostech projektu.

1. Z předešlého kroku máme sestaveny knihovny `idm-lic-api-8.0.0.jar` a `idm-lic-impl-8.0.0.jar` nebo máme tyto knihovny stažené ze systému *Nexus* společnosti BCV solutions s. r. o.

2. Zastavíme aplikační server *Apache Tomcat*: `service tomcat stop`

3. Otevřeme balíček aplikace *CzechIdM* `<idm.war>` v nástroji pro správu archivů nebo jej rozbalíme.

4. Do umístění `<idm.war>/WEB-INF/lib` zkopírujeme knihovny `idm-lic-api-8.0.0.jar` a `idm-lic-impl-8.0.0.jar`

5. Uložíme a nástroj pro správu archivů zavřeme, nebo balíček znovu zabalíme

6. Nahrajeme do aplikačního serveru *Apache Tomcat* upravený balíček aplikace `<idm.war>`.

7. Nastartujeme aplikační server *Apache Tomcat*: `service tomcat start`

4.2 Postup instalace klientské části

Návod pro sestavení klientské části aplikace s libovolným přídavným modulem je dostupný v online dokumentaci *CzechIdM*. Modul pro správu licencí je s tímto návodem plně kompatibilní.

4.2.1 Instalace modulu do CzechIdM

Sestavenou webovou aplikaci nasadíme do webového serveru na serveru, na němž ji chceme provozovat, nebo ji můžeme přidat do stejného souboru `<idm.war>`, jako serverovou část aplikace a provozovat obě části na stejném serveru:

1. Z předešlého kroku máme sestavenu klientskou část aplikace.

2. Zastavíme aplikační server *Apache Tomcat*: `service tomcat stop`

3. Otevřeme balíček aplikace *CzechIdM* `<idm.war>` v nástroji pro správu archivů nebo jej rozbalíme.

4. Do kořene archivu soubory webové aplikace

5. Uložíme a nástroj pro správu archivů zavřeme, nebo balíček znovu zabalíme

6. Nahrajeme do aplikačního serveru *Apache Tomcat* upravený balíček aplikace `<idm.war>`.

7. Nastartujeme aplikační server *Apache Tomcat*: `service tomcat start`

4.3 Další konfigurace v CzechIdM

Nyní je třeba modul aktivovat a nastavit příslušná oprávnění uživatelské roli a administrátorským rolím. Role s nejvyššími oprávněními (*SuperAdminRole*) je aktualizována automaticky.

1. Přihlásíme se do CzechIdM jako administrátor.

2. Přejdeme na stránku *Nastavení/Moduly*

3. Na záložce „Moduly (backend)“ vyhledáme modul s id *lic* a názvem „CzechIdM License Management Module“ a klikneme na tlačítko *Aktivovat* ve sloupečku *Akce*

4. Přejdeme na záložku *Role*

5. Vyhledáme roli s názvem „userRole“ (v závislosti na instalaci může být pojmenována i jinak, jedná se o základní roli všech uživatelů, definující výchozí oprávnění přihlášeného uživatele v aplikaci) a otevřeme její detail.

6. Přejdeme na záložku „Oprávnění“ a klikneme na tlačítko *Přidat*.

7. Typ agendy zvolíme „Licence“ a uložíme.

4.3.1 Integrace do životního cyklu identity

Všechny potřebné procesory, které reagují na události spojené se správou licencí, jsou při aktivaci modulu nastaveny automaticky, není tedy třeba nic konfigurovat. V případě, že chceme některý procesor vypnout, na stránce *Nastavení/Moduly* na záložce „Processory“ vyhledáme jeho název a na stránce *Konfigurace* pak vytvoříme novou konfigurační položku s klíčem stejným, jako název procesoru a hodnotou `false`. Pokud chceme procesor znovu aktivovat, konfigurační položku na téže stránce opět smažeme.

4.3.2 Možnosti integrace s dalšími moduly

V této fázi vývoje modulu pro správu licencí nejsou další možnosti integrace podporovány. Nicméně v rámci návrhu je s několika dalšími možnostmi počítáno. V první řadě je to možnost implementace procesoru, který bude reagovat na změny objektů *LicLicenseDto* a *LicLicensingSeriesDto*. V rámci takovýchto procesorů je pak možné implementovat integraci např. s externími licenčními servery nebo jinými aplikacemi, či moduly.

Dále se nabízí možnost implementovat generátory pro různé typy reportů pomocí API, které základní modul *rpt* poskytuje. Lze takto vytvořit např. generátor reportů s přehledem o tom, jaké licence má který uživatel přiděleny nebo jaká je celková hodnota licencí každého uživatele. Tyto reporty pak mohou využít jak byznys vlastníci reportů nebo auditoři ke kontrole, tak i aplikace, které si je pomocí REST API stáhnou kdykoliv budou potřebovat.

4.4 Popis RESTful API

Dokumentace RESTful API je automaticky generována pomocí nástroje *Swagger* na adrese *idm/swagger-ui.html*. V této kapitole jsou popsány pouze metody, které jsou v modulu přetíženy, nově definovány nebo slouží jako příklad. Třídy *LicLicenseController* a *LicLicensingSeriesController*, v nichž jsou metody RESTful API implementovány, jsou potomky abstraktní třídy *DefaultReadWriteDtoController*, která některé obecné metody implementuje, tyto metody zde nejsou popsány.

4.4.1 Metody třídy *LicLicensingSeriesController*

4.4.1.1 Vyhledávání

Metody pro vyhledávání licenčních sérií.

URL `/lic/licensing-series/search/quick`

Metody `GET`

Příklad dotazu:

```
GET /idm/api/v1/lic/licensing-series/search/quick?
size=10&page=0&sort=name,asc&text=123 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
```

```
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>
```

Příklad odpovědi:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 05 May 2018 00:01:39 GMT

{
 "_embedded" : {
   "licensingSeries" : [ {
     "id" : "57ecd93e-9d3d-4c23-b904-680cdf57acf8",
     "created" : "2018-05-04T08:26:42.181Z",
     "modified" : "2018-05-04T23:56:12.915Z",
     "creator" : "admin",
     "creatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
     "modifier" : "admin",
     "modifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
     "originalCreator" : "admin",
     "originalCreatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
     "originalModifier" : "admin",
     "originalModifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
     "code" : "123",
     "name" : "123",
     "value" : 123,
     "note" : "122",
     "assignType" : "DIRECTLY",
     "blocking" : false,
     "canBeReassigned" : true,
```

```
    "serialNumbersType" : "COUNT",
    "availableLicensesCount" : 0,
    "serialNumbers" : null,
    "wfProcessId" : "approve-license-by-owner",
    "owners" : null,
    "_trimmed" : true,
    "_embedded" : { }
  } ]
},
"_links" : {
  "self" : {
    "href" : "http://localhost:8080/idm/api/v1/lic/licensing-seri
es/search/quick?size=10&page=0&sort=name,asc"
  }
},
"page" : {
  "size" : 10,
  "totalElements" : 1,
  "totalPages" : 1,
  "number" : 0
}
}
```


4.4.1.2 Operace s jednou entitou

Metody slouží k základním operacím vytvoření, čtení, aktualizaci a smazání licenční série.

URL `/lic/licensing-series/:enetityId`

Metody `GET`, `POST`, `PUT`, `DELETE`

Příklad dotazu GET:

```
GET /idm/api/v1/lic/licensing-series/57ecd93e-9d3d-4c23-b904-680
cdf57acf8 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>;
```

Příklad odpovědi na GET:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 04 May 2018 23:52:17 GMT

{
 "id" : "57ecd93e-9d3d-4c23-b904-680cdf57acf8",
 "created" : "2018-05-04T08:26:42.181Z",
 "modified" : "2018-05-04T14:04:51.979Z",
 "creator" : "admin",
 "creatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "modifier" : "admin",
 "modifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "originalCreator" : "admin",
 "originalCreatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "originalModifier" : "admin",
 "originalModifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "code" : "123",
 "name" : "123",
 "value" : 123,
 "note" : "122",
 "assignType" : "DIRECTLY",
 "blocking" : false,
 "canBeReassigned" : true,
 "serialNumbersType" : "COUNT",
 "availableLicensesCount" : 22,
 "serialNumbers" : null,
 "wfProcessId" : "approve-license-by-owner",
 "owners" : [ "86dbd041-1169-4071-abfc-57eb22608196" ],
 "_trimmed" : false,
```

```
 "_embedded" : { },
 "_links" : {
   "self" : {
     "href" : "http://localhost:8080/idm/api/v1/lic/licensing-ser
ies/57ecd93e-9d3d-4c23-b904-680cdf57acf8"
   }
 }
}
```

Příklad dotazu PUT:

```
PUT /idm/api/v1/lic/licensing-series/57ecd93e-9d3d-4c23-b904-680cdf
57acf8 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 867
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>
{
 "id":"57ecd93e-9d3d-4c23-b904-680cdf57acf8",
 "code":"123",
 "name":"123",
 "value":123,
 "note":"122",
 "assignType":"DIRECTLY",
 "blocking":false,
 "canBeReassigned":true,
 "serialNumbersType":"COUNT",
 "availableLicensesCount":22,
 "serialNumbers":null,
 "wfProcessId":"approve-license-by-owner",
 "owners":[
  "86dbd041-1169-4071-abfc-57eb22608196"
  ],
 "_trimmed":false,
 "_embedded":{},
 "_links":{
  "self":{
```

```
   "href":"http://localhost:8080/idm/api/v1/lic/licensing-s
eries/57ecd93e-9d3d-4c23-b904-680cdf57acf8"
  }
 }
}
```

Příklad odpovědi na PUT:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 04 May 2018 23:52:17 GMT

{
 "id" : "57ecd93e-9d3d-4c23-b904-680cdf57acf8",
 "created" : "2018-05-04T08:26:42.181Z",
 "modified" : "2018-05-04T14:04:51.979Z",
 "creator" : "admin",
 "creatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "modifier" : "admin",
 "modifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "originalCreator" : "admin",
 "originalCreatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "originalModifier" : "admin",
 "originalModifierId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "code" : "123",
 "name" : "123",
 "value" : 123,
 "note" : "122",
 "assignType" : "DIRECTLY",
 "blocking" : false,
 "canBeReassigned" : true,
```

```
 "serialNumbersType" : "COUNT",
 "availableLicensesCount" : 22,
 "serialNumbers" : null,
 "wfProcessId" : "approve-license-by-owner",
 "owners" : [ "86dbd041-1169-4071-abfc-57eb22608196" ],
 "_trimmed" : false,
 "_embedded" : { },
 "_links" : {
   "self" : {
     "href" : "http://localhost:8080/idm/api/v1/lic/licensing-ser
ies/57ecd93e-9d3d-4c23-b904-680cdf57acf8"
   }
 }
}
```


Příklad dotazu POST:

```
POST /idm/api/v1/lic/licensing-series HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 279
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>

{
 "assignType":"DIRECTLY",
 "availableLicensesCount":"5000",
 "blocking":null,
 "canBeReassigned":true,
 "code":"idm",
 "name":"czechidm",
 "note":"Standard",
 "owners":[
  "86dbd041-1169-4071-abfc-57eb22608196"
 ],
 "serialNumbersType":"COUNT",
 "value":"123",
 "wfProcessId":"lic-license-approve-owner"
}
```

Příklad odpovědi na POST:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 04 May 2018 23:59:59 GMT

{
 "id" : "10f232da-81c2-4f0c-a2de-f3fcac3d4334",
 "created" : "2018-05-04T23:58:38.724Z",
 "modified" : null,
 "creator" : "admin",
 "creatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "modifier" : null,
 "modifierId" : null,
 "originalCreator" : "admin",
 "originalCreatorId" : "86dbd041-1169-4071-abfc-57eb22608196",
 "originalModifier" : null,
 "originalModifierId" : null,
 "code" : "idm",
 "name" : "czechidm",
 "value" : 123,
 "note" : "Standard",
 "assignType" : "DIRECTLY",
 "blocking" : false,
 "canBeReassigned" : true,
 "serialNumbersType" : "COUNT",
 "availableLicensesCount" : 22,
 "serialNumbers" : null,
 "wfProcessId" : "lic-license-approve-owner",
 "owners" : [ "86dbd041-1169-4071-abfc-57eb22608196" ],
 "_trimmed" : false,
 "_embedded" : { },
```

```
 "_links" : {
   "self" : {
     "href" : "http://localhost:8080/idm/api/v1/lic/licensing-s
eries/10f232da-81c2-4f0c-a2de-f3fcac3d4334"
   }
 }
}
```

Příklad dotazu DELETE:

```
DELETE /idm/api/v1/lic/licensing-series/10f232da-81c2-4f0c-a2de-
f3fcac3d4334 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST:<<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>
```

Příklad odpovědi na DELETE:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Date: Sat, 05 May 2018 00:01:39 GMT
```

4.4.1.3 Žádost

Metoda pro vytvoření žádosti o licence

URL `/lic/licensing-series/request`

Metody `PUT`

Příklad dotazu:

```
GET /idm/api/v1/lic/licensing-series/request
Host: localhost:8080
Connection: keep-alive
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>
```

Příklad odpovědi:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 05 May 2018 00:01:39 GMT
```

4.4.2 Metody třídy *LicLicenseController*

4.4.2.1 Odebrání licence

Metoda pro odebrání licence uživateli.

URL `/lic/license/:entityId/unassign`

Metody `PUT`

Příklad dotazu:

```
GET /idm/api/v1/lic/license/10f232da-81c2-4f0c-a2de-f3fcac3d4334
/unassign
Host: localhost:8080
Connection: keep-alive
Accept: application/hal+json
Origin: http://localhost:3000
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: cs-CZ,cs;q=0.9
Cookie: JSESSIONID=<<hodnota>>
```

Příklad odpovědi:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Expose-Headers: CIDMST
Access-Control-Allow-Credentials: true
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
CIDMST: <<hodnota>>
Content-Type: application/hal+json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 05 May 2018 00:01:39 GMT
```

Závěr

Tato práce se zabývala analýzou a návrhem modulu pro správu licencí pro aplikaci *CzechIdM* a implementací jeho prototypu. V první části se věnovala popisu problematiky IAM, popisu aplikace *CzechIdM* a sběru požadavků na funkce modulu. Poté byly požadavky analyzovány a byla vytvořena podrobná funkční a technická specifikace a tato specifikace, která byla následně připomínkována a nakonec schválena architektem a produktovým manažerem *CzechIdM* ze společnosti BCV solutions s. r. o, společnosti vyvíjející aplikaci *CzechIdM*, která byla zadavatelem práce. Jednotlivé požadavky pak byly rozděleny do tří samostatných fází implementace.

Následovala fáze technického návrhu řešení. Zde bylo třeba se vypořádat s poměrně živelným vývojem druhé generace *CzechIdM*, pro kterou byl modul vyvíjen, a rychlé zastarávání a nebo absence některých částí dokumentace, zejména té programátorské. V teoretické části tedy byly popsány technologie, používané v *CzechIdM* a architektura aplikace. V rámci této fáze bylo rovněž navrženo uživatelské rozhraní modulu a jeho datový model. V další části se práce zabývala implementací první verze prototypu modulu, jeho testováním a dokumentací. Nakonec byly diskutovány i další možnosti integrace modulu s dalšími částmi *CzechIdM* i externími aplikacemi.

Podařilo se vytvořit komplexní popis funkčních požadavků modulu a jeho technický návrh. Dále se podařilo popsat architekturu aplikace *CzechIdM* nad rámec jejího popisu v oficiální dokumentaci. Podařilo se rovněž vytvořit první prototyp modulu. Jeho funkce z velké části pokrývají požadavky na první fázi vývoje, která byla cílem této práce.

Hlavními nedostatky modulu oproti požadavkům na první fázi, je malé pokrytí jednotkovými a integračními testy a absence komplexnějšího výpočtu oprávnění uživatele. Dále pak nedostatky na úrovni přívětivosti uživatelského rozhraní, tedy klientské části modulu. Další připomínky k funkci modulu jsou výstupem testování a zejména výstupem průběžných konzultací s vývojáři a konzultanty z BCV solutions s. r. o.

Touto prací vývoj modulu nekončí. Dalším krokem ve vývoji bude dokončení první verze modulu, zejména pak výpočet oprávnění a přidání možnosti zpracovávat změny v licenčních sadách a přidělení licence uživateli pomocí procesorů, která na ně reagují. Následovat pak bude implementace funkčních požadavků z druhé fáze vývoje, tedy přidělování licencí na základě přidělení role. Tím se zvýší míra automatizace celého procesu. Poté bude třeba zrevidovat funkční požadavky, doplnit je o ty nové, které budou výstupem dalšího testování a stanovit priority jejich implementace. Jednou z možností dalšího směřování rozvoje modulu je jeho zobecnění na správu jakéhokoliv majetku, který je v rámci životního cyklu identity v organizaci uživateli přidělen.

Literatura

[1] Identity Management - Access Management - Gartner Research [online článek]. Mar 2017, [cit. 2018-05-03]. Dostupné z: https://blogs.gartner.com/it-glossary/identity-and-access-management-iam/

[2] MARTIN, W. J. K., James A.: What is identity management? IAM definition, uses, and solutions [online článek]. Jan 2018, [cit. 2018-05-03]. Dostupné z: https://www.csoonline.com/article/2120384/identity-management/what-is-identity-management-iam-definition-uses-and-solutions.html

[3] Identities (users) [online dokumentace]. [cit. 2018-05-03]. Dostupné z: https://wiki.czechidm.com/8.0/documentation/identities

[4] HR Processes [online dokumentace]. [cit. 2018-05-03]. Dostupné z: https://wiki.czechidm.com/8.0/documentation/hr_processes

[5] BCV solutions s.r.o.: Role change workflow [online obrázek]. 2017, [cit. 2018-05-03]. Dostupné z: https://wiki.czechidm.com/_media/devel/documentation/roles_approval_process.png

[6] Events - processing of eventseventarchitectureconfiguration [online dokumentace]. [cit. 2018-05-03]. Dostupné z: https://wiki.czechidm.com/8.0/documentation/architecture/dev/events

[7] HUNT, P.; KHASNABISH, B.; NADALIN, A.; aj.: System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements. RFC 7642, RFC Editor, September 2015. Dostupné z: https://www.rfc-editor.org/rfc/rfc7642.txt

[8] HUNT, P.; GRIZZLE, K.; ANSARI, M.; aj.: System for Cross-domain Identity Management: Protocol. RFC 7644, RFC Editor, September 2015. Dostupné z: https://www.rfc-editor.org/rfc/rfc7644.txt

[9] HUNT, P.; GRIZZLE, K.; WAHLSTROEM, E.; aj.: System for Cross-domain Identity Management: Core Schema. RFC 7643, RFC Editor, September 2015. Dostupné z: https://www.rfc-editor.org/rfc/rfc7643.txt

[10] SCIM 2.0 Implementations [online stránka]. [cit. 2018-05-03]. Dostupné z: http://www.simplecloud.info/#Implementations2

[11] GIERKE, O.; DARIMONT, T.; STROBL, C.; aj.: Spring Data JPA [online dokumentace]. Dostupné z: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.multiple-modules

[12] Core Technologies [online dokumentace]. [cit. 2018-05-03]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#expressions

[13] Spring Data REST [online stránka]. [cit. 2018-05-03]. Dostupné z: https://projects.spring.io/spring-data-rest/

[14] Testing [online dokumentace]. [cit. 2018-05-03]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/testing.html

[15] Querydsl [online repositář]. [cit. 2018-05-03]. Dostupné z: https://github.com/querydsl/querydsl

[16] GIERKE, O.: Advanced Spring Data JPA - Specifications and Querydsl [online článek]. Apr 2011, [cit. 2018-05-03]. Dostupné z: https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl/

[17] Mapping Objects to Relational Databases: O/R Mapping In Detail [online článek]. [cit. 2018-05-03]. Dostupné z: http://www.agiledata.org/essays/mappingObjects.html

[18] FOWLER, M.: OrmHate [online článek]. [cit. 2018-05-03]. Dostupné z: https://www.martinfowler.com/bliki/OrmHate.html

[19] Fielding, R. T.: *Architectural styles and the design of network-based software architectures*. Dizertační práce, 2000.

[20] PICHLÍK, R.: A REST [online článek]. Oct 2007, [cit. 2018-05-03]. Dostupné z: https://dagblog.cz/a-rest-c5156313d79e

[21] Understanding: REST [online článek]. [cit. 2018-05-03]. Dostupné z: https://spring.io/understanding/REST

[22] ROUSE, M.; HANNAN, E.; WILSON, S.: What is RESTful API? - Definition from WhatIs.com [online článek]. [cit. 2018-05-03]. Dostupné z: https://searchmicroservices.techtarget.com/definition/RESTful-API

[23] RESTEasy - JBoss Community [online článek]. [cit. 2018-05-03]. Dostupné z: http://resteasy.jboss.org/

[24] BUNA, S.: Yes, React is taking over front-end development. The question is why. [online článek]. Mar 2017, [cit. 2018-05-03]. Dostupné z: https://medium.freecodecamp.org/yes-react-is-taking-over-front-end-development-the-question-is-why-40837af8ab76

Příloha **A**

Seznam použitých zkratek

AD Microsoft Active Directory

AM Access Manager

API Application Programming Interface

BDD Behavior-driven development

BPM Business Process Management

BPMN Business Process Model and Notation

CSV Comma-separated values

DB Databáze

DMN Decision Model and Notation

DOM Document Object Model

DTO Data transfer object

EDA Event-driven architecture

GUI Graphical user interface

HATEOAS Hypermedia as the Engine of Application State

HTTP Hypertext trasnfer protocol

IAM Identity & Access management

ICF Identity Connectors Framework

IdM Identity Management

ILP Identity lifecycle processes

IT Information technology

JDBC Java Satabase Connectivity

JDO Java Data Objects

JPA Java Persistance API

JSP JavaServer Pages

JVM Java Virutal Machine

LDAP Lightweight Directory Access Protocol

MS Microsoft

ORM Object-relational mapping

PPV Pracovně-právní vztah

REST Representational state transfer

SpEL Spring Expression Language

SSO Single sing-on

SQL Structured Query Language

TDD Test-driven development

UEL Unified Expression Language

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

WAR Web Application Resource

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `dist` ..... adresář se spustitelnou formou implementace
  - `idm.war` ..... sestavená aplikace *CzechIdM* s nainstalovaným modulem
  - `idm-lic-api.jar` ..... sestavená knihovna API
  - `idm-lic-impl.jar` ..... sestavená knihovna implementace
  - `client` ..... adresář se sestavenou aplikací klientské části modulu
- `src` ..... adresář se zdrojovými soubory práce
  - `impl` ..... zdrojové kódy implementace
    - `backend` ..... zdrojové kódy serverové části
      - `idm-lic`
        - `idm-lic-api` ..... implementace API modulu
        - `idm-lic-impl` ..... implementace modulu
    - `frontend` ..... Zdrojové kódy klientské části
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF