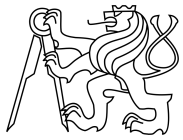
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	ACS-Manager: aplikace pro správu pravidel přístupu do místností
Student:	Michal Pěch
Vedoucí:	Ing. Tomáš Kadlec
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je vytvořit webovou aplikaci acs-manager, která bude poskytovat uživatelské rozhraní pro správu pravidel vstupu pro acs-controller. Ta byla vyvinuta OICT FIT ČVUT.

* Pravidla (ACE) mohou být vztažena ke dveřím a/nebo skupinám dveří, identitám a/nebo skupinám identit.
* Aplikace bude poskytovat RESTful API pro dotazování se na pravidla pro kombinaci dveře/identita.
* V aplikaci bude možné zadat oprávněné uživatele, kteří provádějí nastavení přístupů a umožní práva dále delegovat.
* Aplikace bude fungovat v režimu
(1) samostatném s lokálním úložištěm sdíleném s aplikací card-manager,
(2) propojeném s IdM (Identity Management), identity budou uloženy pouze v IdM, ACE se přes IdM propisují do LDAP.

Zjistěte požadavky na aplikaci, proveďte analýzu. Navrhněte webovou aplikaci a její API. Implementujte aplikaci a otestujte (jednotkové, funkční a integrační testy - směrem k IdM).
Práce se řídí dokumentem Pravidla a zásady projektů FIT.

Seznam odborné literatury

* Pravidla a zásady projektů FIT v aktuální verzi - https://goo.gl/tSb5b5
* Další literaturu dodá vedoucí práce v průběhu řešení dle potřeby.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 5. února 2017

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

Bakalářská práce

ACS-Manager: aplikace pro správu pravidel přístupu do místností

Michal Pěch

Vedoucí práce: Ing. Tomáš Kadlec

15. května 2017

Poděkování

Chtěl bych touto cestou poděkovat vedoucímu práce Ing. Tomášovi Kadlecovi a oponentovi Ing. Jiřímu Špačkovi za cenné rady, konzultace a vstřícné jednání při realizaci této bakalářské práce. Dále bych chtěl poděkovat rodině a kamarádům, kteří mě při studiu podporovali. Zejména pak Martinovi, Markétě a Lubošovi.

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.

V Praze dne 15. května 2017 .....

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Pěch, Michal. *ACS-Manager: aplikace pro správu pravidel přístupu do místností*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨`https://gitlab.com/michalpechnet/bakalarska-prace`⟩.

Abstrakt

Tato bakalářská práce si dává za cíl vytvořit aplikaci pro správu pravidel vstupu do místností, díky kterým bude možné provádět autorizaci osob. Jejím obsahem je analýza a návrh řešení, na základě kterého je vytvořena implementace. Výstupem je webová aplikace.

Klíčová slova správa pravidel přístupu do místností, identifikační karta, webová aplikace, Symfony framework, PHP

Abstract

Main goal of this bachelor's thesis is to create an application for managing room access control which will be able to authorize persons to access room. The content of this thesis is the analysis and proposed solution on which implementation is created. The end result is a web application.

Keywords managing room access control, identity card, web application, Symfony framework, PHP

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Vysoké školy disponují vysokým počtem prostor, do kterých potřebují umožnit vstup osobám z akademické obce i mimo ni. Vstup je často podmíněn na základě předem definovaných pravidel. Běžným řešením je řízení přístupu na základě identifikační karty. Jednotlivé dveře mají své čtecí zařízení, pomocí kterého je možné provést identifikaci osoby, na jejímž základě je povolen, nebo zamítnut přístup. Toto řešení používá i České vysoké učení technické v Praze (ČVUT). Fakulta informačních technologií (FIT) však disponuje specializovanými prostory, které nefungují v tzv. běžném provozu a je do nich potřeba řídit přístup explicitně.

Motivace k vytvoření práce

V srpnu 2016 byly na FIT zabezpečeny konferenční místnosti čtečkami RFID karet. Dříve bylo nutné k přístupu do konferenčních místností zapůjčit žadateli klíč. S přibývajícím zájmem o konferenční místnost se však předávání klíčů žadatelům a jejich následný návrat stával více problematický a tento systém správy přístupu do konferenčních místností přestal být únosný.

Podobně byly zabezpečeny některé neveřejné místnosti fakulty (např. serverovny). Rovněž byly čtečky nainstalovány ke vstupům do studentských respirií v budově TH:A. Tak bylo učiněno na základě podnětů a stížností studentů o častém využití respirií studenty jiných fakult.

Řešení, v podobě instalace čteček u místností zmíněných v úvodu, bylo na žádost FIT implementováno Oddělením ICT FIT. Systém, jehož implementace se zdařila v krátkém časovém úseku, výrazně usnadnil agendu správy přístupu do místností. Samotná správa pravidel přístupu však byla dočasně řešena konfiguračním souborem aplikace ovládající zámky dveří, jež není uživatelsky přívětivá.

Cíl práce

Cílem této bakalářské práce je vytvořit webovou aplikaci ACS-Manager (ACS-M), která poskytne uživatelské rozhraní pro správu pravidel přístupu. Obsahem práce je analýza současného řešení a požadavků na aplikaci, návrh její architektury, na základě kterého je vytvořena implementace.

Struktura práce

Práce se řídí dokumentem Pravidla a zásady projektů FIT (Pravidla)[1], jež popisuje preferovaný způsob řešení projektů v rámci FIT. S ohledem na povahu práce není dle Pravidel plně dodržena její doporučená struktura. Tato bakalářská práce je strukturována do pěti logických celků.

V první části pod názvem Analytická fáze jsem se zaměřil na analýzu současného stavu. Dále jsem provedl kvalitativní průzkum, z nějž vyplynuly požadavky na funkce a kvality aplikace. Tyto požadavky poté byly vyhodnoceny vzhledem k současnému stavu řešení.

Obsahem druhé části s názvem Případy užití je popis interakce uživatele s aplikací ACS-M. V rámci něj definuji případy užití aplikace a jednotlivé aktéry, již v nich vystupují.

Ve třetí části s názvem Návrhová fáze je samotný návrh aplikace. V rámci něj jsem stanovil datový model, aplikační logiku, rozhraní aplikace a také vytvořil návrh uživatelského prostředí za pomoci wireframe diagramů.

Obsahem čtvrté části s názvem Implementační fáze je implementační návrh aplikace. V této kapitole popisuji použité technologie, architekturu aplikace a podstatné části její implementace.

V poslední části dokumentu pod názvem Testování aplikace se zabývám způsoby testování aplikace. Popisuje zde různé metody testování a jejich vyhodnocení.

Kapitola 1

Analytická fáze

Tato část práce se zabývá analýzou zadaného problému. Prozkoumal jsem v ní současný stav řešení. Dále jsem provedl kvalitativní průzkum se zadavatelem práce a dalšími zainteresovanými osobami.

Na základě tohoto průzkumu vznikly požadavky na funkce a kvality aplikace. Na závěr jsem tyto získané požadavky vyhodnotil vzhledem k současnému řešení.

1.1 Průzkum současného řešení

Přístup do specializovaných místností je v současném stavu ovládán aplikací ACS-Controller (ACS-C)[2]. Ta je napsána v jazyce Python[3] a byla vyvinuta Oddělením ICT FIT ČVUT. ACS-C napřímo ovládá čtecí zařízení a zámky jednotlivých vstupů.

ACS-C funguje následovně. Při přiložení identifikační karty ke čtecímu zařízení si aplikace načte číslo karty. Toto číslo následně odešle na fakultní LDAP server (případně se využije cache ACS-C), z nějž získá osobu, které tato karta patří.

S takto získanou osobou se poté provede autorizace ke vstupu do místnosti. Autorizace může mít časově omezenou platnost a je trojího typu:

1. Osoba je ke vstupu autorizována na základě úspěšné identifikace.
2. Vstup je umožněn pouze definovaným osobám.
3. Vstup je umožněn na základě členství osoby ve skupině LDAP.

1.2 Kvalitativní průzkum

Kvalitativní průzkum vznikl na základě rozhovorů. Během nich vznikly hypotézy uvedené níže, které shrnují získané poznatky o dané problematice.

Rozhovory jsem provedl s následujícími osobami:

Ing. Tomáš Kadlec je vedoucím Oddělení ICT FIT ČVUT. Inženýr Kadlec je autorem aplikace ACS-C a má celý systém přístupů na starosti. Rovněž je vedoucím této bakalářské práce.

Ing. Jiří Špaček je správcem fakultního Identity Management (IdM) systému. To je systém, v němž jsou uloženy informace o osobách a jejich rolích, kartách a místnostech. Inženýr Špaček je zároveň oponentem této bakalářské práce.

Ing. Martin Bílý je fakultním správcem systému K4. Tento systém řidí vstupy do místností v rámci celého ČVUT.

Aplikace ACS-M potřebuje pro svou činnost data o vstupech a osobách a jejich rolích. Tato data lze mít uložena lokálně a nebo mohou být poskytnuta nějakou službou. Takové službě se typicky říká Identity Management (IdM).

ACS-M je primárně vyvíjeno pro Oddělení ICT FIT. Protože se však dle úvodního prohlášení jedná o svobodně šiřitelný software, aplikace bude navržena tak, aby ji bylo možné použít i mimo FIT ČVUT.

Z tohoto důvodu bude ACS-M umět fungovat ve dvou režimech provozu. První z nich je lokální, ve kterém aplikace používá lokální datové uložiště. Druhý je pak IdM režim, ve kterém aplikace plně využívá uložiště fakultní IdM služby. Režim aplikace se nakonfiguruje při jejím nasazení.

Aplikace ACS-M není jediná, jež potřebuje data o osobách a jejich rolích v obou režimech provozu. Současně s ní vzniká aplikace Card-Manager (C-M), která pro potřeby FIT spravuje karty osob. C-M podléhá stejné licenci a má na práci s osobami identické požadavky jako ACS-M.

Protože jsou obě dvě aplikace vyvíjeny v rámci Oddělení ICT, je vytvořen balík Identity, jenž poskytuje funkce pro práci s osobami a jejich rolemi. Obě dvě aplikace je možné nasadit společně. V lokálním režimu využijí sdílenou databázi, v IdM režimu používají jako datové uložiště fakultní IdM.

Osoby a jejich role můžeme shrnout pod pojmem identity. S identitami pracuje stejnojmenný balík Identity a způsob jejich použití je v obou režimech provozu aplikace stejný.

Dveře reprezentují jednotlivé vstupy, ve skutečnosti se jedná o dvojici čtečka - zámek, a mohou být organizovány do skupin. Pro přístup do dveří je ACS-M dotázáno vždy na konkrétní osobu a vstup.

V IdM režimu je reprezentace dveří odlišná. ČVUT nepracuje na úrovni datového modelu se vstupy jako takovými, ale rovnou s celými místnostmi. Tyto místnosti je rovněž možné organizovat do skupin. Ponechání dveří v lokálním režimu je však žádoucí z důvodu zachování co největší univerzálnosti aplikace.

Vstupy jsou podmíněny pravidly, které může vytvářet správce pravidel. Správce pravidel je osoba, jež získá oprávnění ke správě pravidel pro konkrétní dveře, respektive jejich skupinu. Toto oprávnění udílí administrátor aplikace a může mít časovou platnost od - do. Správce pravidel smí volně svá oprávnění delegovat na jinou osobu.

Jednotlivá pravidla mají určena právě jeden vstup, respektive jejich skupinu. Mohou být přiřazena jedné či více identit. Každé pravidlo může mít příznak exkluzivity, který ruší všechna neexkluzivní pravidla v daný čas pro dané dveře/skupinu dveří. Volitelně mohou mít časovou platnost od - do.

Pravidla ani správcovská oprávnění nelze editovat. V lokálním režimu je možné je smazat, v režimu IdM se prohlásí za neplatná. Veškerá aktivita aplikace je logována.

Aplikace ACS-M bude poskytovat uživatelské rozhraní pro správu pravidel, oprávnění a dveří. Ve výpisech těchto artefaktů bude možné filtrovat dle jejich atributů.

Dále poskytne RESTful API pro aplikaci ACS-C za účelem ověření oprávnění přístupu pro danou osobu do daného vstupu. Komunikace mezi oběma aplikacemi bude zabezpečena. V lokálním režimu je zabezpečení řešeno formou HTTP basic, v IdM režimu je použit protokol OAuth 2.0[4].

1.3 Kvantitativní průzkum

Vzhledem k povaze práce a jejímu cílení na velice úzký okruh uživatelů je shledáno provedení kvantitativního průzkumu jako nerelavantní. Tohoto závěru bylo dosaženo na základě rozhovorů s vedoucím práce Ing. Tomášem Kadlecem, který jej schválil.

1.4 Požadavky na kvality

Požadavky na kvality aplikace ACS-M vychází z dokumentu Pravidla a zásady projektů FIT [1]. Tento dokument specifikuje preferovaný způsob řešení projektů v rámci FIT. Všechna doporučení však nebyla dodržena.

Aplikace plně nerespektuje požadavky WCAG 2.0 AA[5] s ohledem na způsob používání aplikace a cílení aplikace na specifický okruh uživatelů. Aplikace vyžaduje použití javascriptu v prohlížeči pro zajištění plné funkcionality.

1.5 Požadavky na funkce

Tato kapitola popisuje požadavky na funkce vycházející z kvalitativního průzkumu. Jednotlivé požadavky jsou kategorizovány dle úrovně nezbytnosti. Pokud úroveň není definována, je brána jako [MUST].

Požadavky z kategorie [MUST] jsou nezbytnou součástí aplikace ACS-M. [SHOULD] jsou požadavky, které by v ACS-M měly být, avšak aplikace na jejich implementaci není funkčně závislá. [MAY] značí požadavky, které mohou v budoucnu sloužit k rozšíření aplikace ACS-M.

Vzhledem k tomu, že požadavky na funkce se mohou lišit v závislosti na použitém režimu provozu aplikace, jsou označeny příznaky. Příznak [IDM] značí požadavky IdM režimu, [LOKÁLNÍ] označuje požadavky lokálního režimu. V případě, že ani jeden z těchto příznaků není uveden, týká se požadavek na funkci obou režimů provozu.

1. Aplikace podporuje dva režimy provozu:
   a) lokální,
   b) IdM.
2. Pravidla pro přístup:
   a) vytvoření pravidla,
   b) zneplatněné/smazání pravidla,
   c) získání pravidel a jejich kontrola pro dané dveře a identitu.
3. Správce pravidel:
   a) přidělení oprávnění pro správu pravidel,
   b) odebrání oprávnění,
   c) [SHOULD] delegace oprávnění na jinou identitu a její odebrání.
4. Dveře:
   a) získání dveří,
   b) organizace dveří do skupin a jejich následné použití v pravidlech pro přístup a oprávnění pro správce pravidel,
   c) přidání dveří, resp. skupin,
   d) úprava dveří, resp. skupiny,
   e) odebrání dveří, resp. skupiny,
   f) [LOKÁLNÍ][MAY] import dveří, resp. skupiny.
5. Pohledy získaných pravidel, delegací a dveří podporují filtrování.

1.6 Vyhodnocení průzkumu současného řešení

Správa pravidel řízena skrze aplikace ACS-C má jistá omezení. Jedním z nich je možnost vytvářet pravidla pouze skrze konfigurační soubor. Tento přístup není uživatelsky přívětivý a vyžaduje jeho nutné, opakované nasazení s každou jeho změnou a následný restart aplikace. Každá čtečka má však svůj vlastní proces aplikace, jenž ji ovládá, a tedy restart je nutné provést u všech těchto procesů.

Vytváření pravidel skrze konfigurační soubor však má i další úskalí. Pravidla nelze vytvářet pro skupiny dveří, respektive pro skupiny osob, ale vždy pouze pro konkrétní vstup do místnosti a konkrétní osobu. Pro povolení přístupu do většího počtu místností pro větší skupinu osob je tak nutné vytvářet celou sadu pravidel, jež se velmi těžce udržuje konzistentní. Dalším omezením je nemožnost vytvořit pravidlo pro dočasné omezení přístupu. Omezení je možné docílit pouze zásahem správce a zkrácením platnosti pravidel.

Významným omezením je pak samotná správa tohoto konfiguračního souboru. Vzhledem k tomu, že v tomto přístupu ke správě pravidel není možné jakkoliv omezit práva pro jejich správu, je určen pouze jeden správce. Ten musí znát přesnou syntax souboru a je zde poměrně velké riziko snadného zanesení chyby.

Konfigurační soubor sice prochází základní kontrolou syntaxe, ta však nedokáže pokrýt výskyt všech chyb, a tak jeho nasazení může způsobit až nefunkčnost aplikace. Konfigurační soubor je navíc verzován skrze fakultní službu GitLab, která vyžaduje alespoň základní znalosti pro práci v příkazové řádce s verzovacím systémem Git.

Kapitola 2

Případy užití

V této kapitole popisuji interakci uživatelele s aplikací ACS-M. Případy užití vychází z předchozí části práce (kapitola 1. Analytická fáze) a je možné je rozdělit do funkčních celků.

Tyto funkční celky jsou reprezentovány jednotlivými případy užití a definovány v podkapitole 2.2 - Případy užití. Průchody se mohou lišit dle režimu provozu aplikace a jsou vytvořeny na základě podkapitol 1.5 - Požadavky na funkce a 1.4 - Požadavky na kvality.

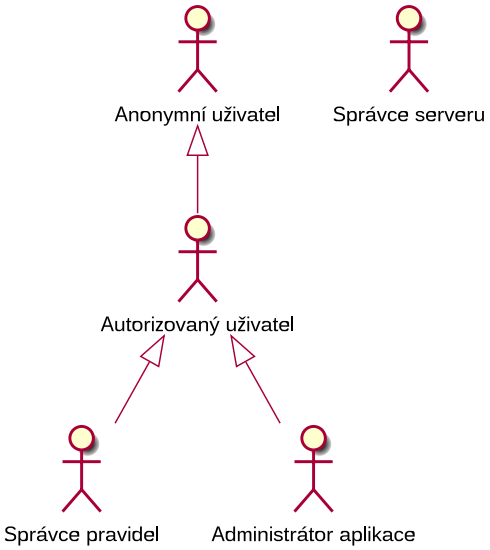
Každý z těchto případů užití má tzv. aktéra. Aktér je uživatel aplikace, jenž v ní vykonává nějakou činnost. Aktéři jsou kategorizováni do několika typů, kde se každý z nich liší dle oprávnění požadovaných v jednotlivých částech aplikace.

2.1 Aktéři

Tato podkapitola popisuje aktéry, již vystupují v případech užití aplikace ACS-M. Aktéři jsou rozděleni dle jejich oprávnění v jednotlivých částech aplikace. ACS-M má pět druhů aktérů, jejichž struktura je zobrazena na obrázku 2.1 - Aktéři ACS-M.

Prvním z aktérů je anonymní uživatel, který může vykonat pouze přihlášení do aplikace. Jeho potomkem je autorizovaný uživatel, jenž si může zobrazit přehledy pravidel, oprávnění, identit a dveří. Autorizovaný uživatel je předkem dvou aktérů, v jejichž závislosti se liší data zobrazená v jednotlivých přehledech.

Prvním potomkem autorizovaného uživatele je správce pravidel, jehož činnosti v aplikaci souvisí s agendou pravidel přístupu. Druhým je pak administrátor aplikace, který je správcem oprávnění a jednotlivých dveří. Posledním aktérem je správce serveru, na kterém aplikace běží. Ten vykonává svou činnost v příkazové řádce aplikace.



Obrázek 2.1: Aktéři aplikace ACS-M

2.2 Případy užití

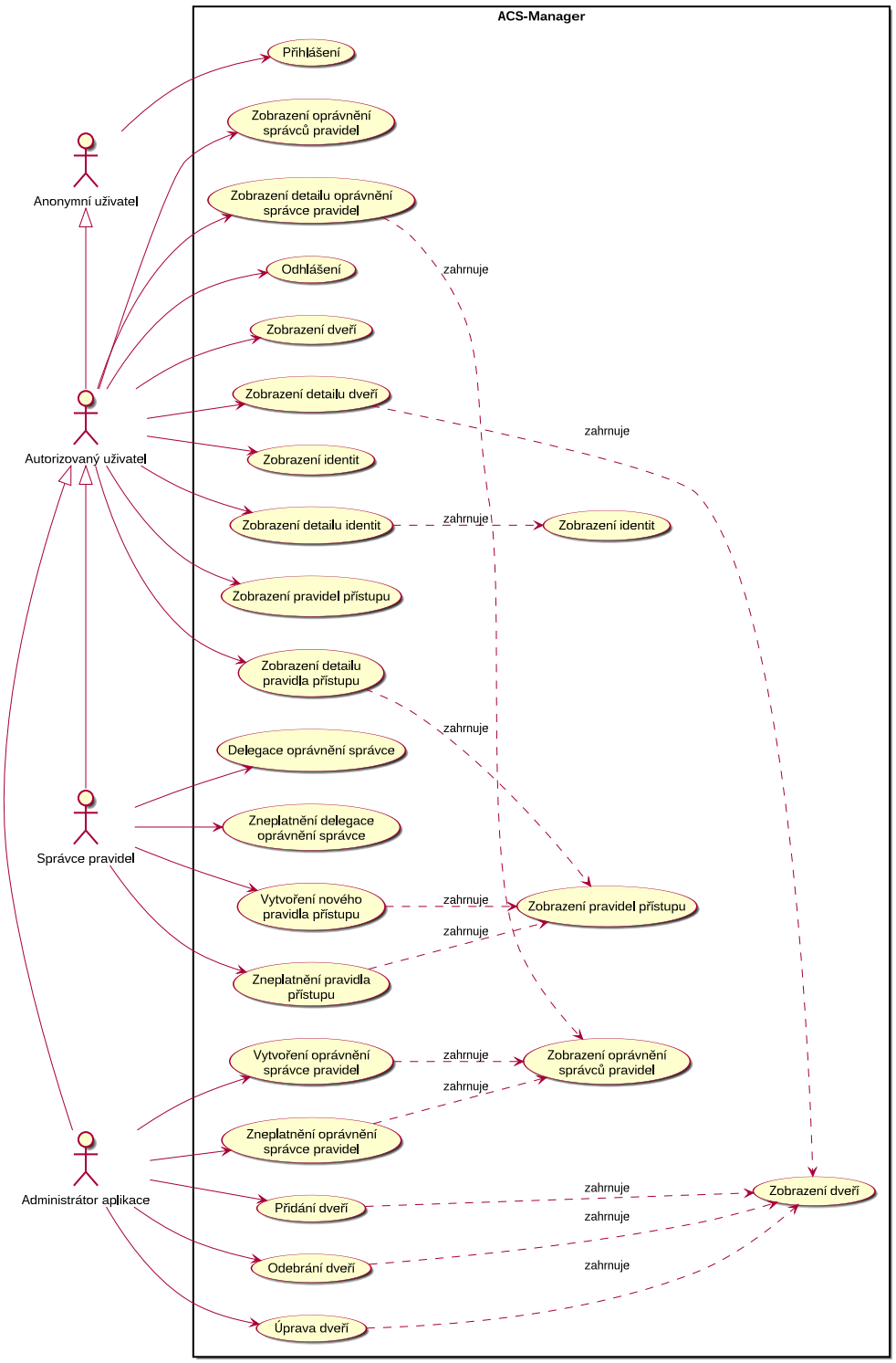
Jednotlivé případy užití popisují interakci uživatele s aplikací ACS-M. Každý z průchodů má svého aktéra respektive aktéry a scénář bodově popisující interakci uživatele s aplikací. Jeden případ užití může být rovněž součástí jiného.

Případy užití se liší dle použitého režimu aplikace. Z tohoto důvodu jsou rozděleny do tří samostatných podkapitol. V první z nich jsou popsány průchody, jež je možné vykonávat nezávisle na zvoleném režimu aplikace. Zbylé dva se pak liší dle použitého režimu aplikace.

Průchody značím prefixem UCx.y, kde x značí aktéra a y jednotlivé případy užití. Dle aktérů je tedy můžeme rozdělit do pěti částí. Předpokladem všech případů užití, vyjma UC1.x a U5.x, je přihlášení do aplikace ACS-M.

V první části UC1.y se nachází pouze jeden průchod pro přihlášení uživatele. V části druhé označené UC2.y jsou průchody, které vykonává autorizovaný uživatel. Průchody UC3.y vykonává správce pravidel. Čtvrtá část označená UC4.y reprezentuje případy užití, jež vykonává administrátor ACS-M. Aktérem průchodů značených UC5.y je správce serveru.

Výsledná zobrazená data se u případů užití UC2.y pro autorizovaného uživatele mohou lišit v závislosti na odvozené roli od toho uživatele. Administrátorovi aplikace jsou vždy zobrazena všechna data. V případě, že se jedná o správce pravidel, mohou být tato data ještě filtrována. Tato skutečnost je případně uvedena v popisu případu užití.



Obrázek 2.2: Společné případy užití aplikace ACS-M

2.2.1 Společné případy užití

Společné případy užití zahrnují průchody, jež se mohou vykonávat v obou režimech ACS-M. Tyto případy užití pokrývají její hlavní funkčnost. Pro přehlednost případů užití jsou skupiny dveří v IdM režimu uvedeny rovněž jako dveře. Přehled společných případů užití je znázorněný na obrázku 2.2.

UC1.1 - Přihlášení

Tento případ užití popisuje proces přihlášení uživatele do aplikace ACS-M.

Aktér: Anonymní uživatel

Scénář:

1. Systém zobrazí formulář pro přihlášení.
2. Uživatel vyplní uživatelské jméno a heslo a klikne na ovládací prvek pro přihlášení.
3. Systém požadavek zpracuje a zobrazí úvodní obrazovku.

UC2.1 - Odhlášení

Tento případ užití popisuje proces odhlášení uživatele.

Aktér: Anonymní uživatel

Scénář:

1. Uživatel klikne na tlačítko „Odhlásit se".
2. Systém odhlásí uživatele.

UC2.2 - Zobrazení dveří

V tomto případu užití si chce uživatel zobrazit seznam všech dveří. Výsledný pohled se liší dle práv uživatele. Správci pravidel se zobrazí pouze dveře (a případně jejich potomci), ke kterým má oprávnění spravovat pravidla.

Aktér: Autorizovaný uživatel

Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Dveře".
3. Systém zobrazí existující dveře.

UC2.3 - Zobrazení detailu dveří

V tomto případu užití si chce uživatel zobrazit detail dveří.

Aktér: Autorizovaný uživatel

Zahrnuje: UC2.2 - Zobrazení dveří

Scénář:

1. Uživatel provede UC2.2 - Zobrazení dveří.
2. Uživatel si vybere konkrétní dveře a klikne na možnost „Detail“.
3. Systém zobrazí detail dveří.

UC2.4 - Zobrazení identit

V tomto případu užití si chce uživatel zobrazit seznam identit. Identitou může být buď osoba, nebo role. Princip zobrazení seznamu je pro oba dva typy identit stejný.

Aktér: Autorizovaný uživatel

Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Osoby“, respektive „Role“.
3. Systém zobrazí všechny osoby, respektive role.

UC2.5 - Zobrazení detailu identity

V tomto případu užití si chce uživatel zobrazit detail identity (osoby respektive role). Detail identity dále zobrazuje pravidla, jež má daná identita přiřazena. V případě, že se jedná o osobu a aktér je administrátor aplikace, jsou zobrazena také oprávnění dané osoby a případně pravidla, jež vytvořila.

Aktér: Autorizovaný uživatel

Zahrnuje: UC2.4 - Zobrazení identit

Scénář:

1. Uživatel provede UC2.4 - Zobrazení identit.
2. Uživatel si vybere osobu, respektive roli a klikne na možnost „Detail“.
3. Systém zobrazí detail osoby, respektive role.

UC2.6 - Zobrazení pravidel přístupu

V tomto případu užití si chce uživatel zobrazit seznam existujících pravidel. Výsledný pohled se liší dle práv uživatele. Správci pravidel se zobrazí pouze pravidla, jež má oprávnění spravovat.

Aktér: Autorizovaný uživatel

Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Pravidla“.
3. Systém zobrazí existující pravidla.

UC2.7 - Zobrazení detailu pravidla přístupu

V tomto případu užití si chce uživatel zobrazit detail pravidla přístupu.

Aktér: Autorizovaný uživatel

Zahrnuje: UC2.6 - Zobrazení pravidel přístupu

Scénář:

1. Uživatel provede UC2.6 - Zobrazení pravidel přístupu.
2. Uživatel si vybere pravidlo a klikne na možnost „Detail“.
3. Systém zobrazí detail pravidla.

UC2.8 - Zobrazení oprávnění správců pravidel

V tomto případu užití si chce uživatel zobrazit seznam oprávnění správců pravidel. Výsledný pohled se liší dle práv uživatele. Správci pravidel se zobrazí pouze oprávnění, jichž je vlastníkem nebo která delegoval.

Aktér: Autorizovaný uživatel

Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Oprávnění“.
3. Systém zobrazí oprávnění správce pravidel.

UC2.9 - Zobrazení detailu oprávnění správce pravidel

V tomto případu užití si chce uživatel zobrazit detail oprávnění správců pravidel.

Aktér: Autorizovaný uživatel

Scénář:

1. Uživatel provede UC2.8 - Zobrazení oprávnění správců pravidel.
2. Uživatel vybere oprávnění a klikne na možnost „Detail“.
3. Systém zobrazí detail oprávnění správce pravidel.

UC3.1 - Vytvoření nového pravidla přístupu

Tento případ užití popisuje vytvoření nového pravidla přístup pro dveře nebo jejich skupinu.

Aktér: Správce pravidel

Zahrnuje: UC2.6 - Zobrazení pravidel přístupu

Scénář:

1. Uživatel provede UC2.6 - Zobrazení pravidel přístupu.
2. Uživatel vybere možnost „Vytvořit pravidlo“, na základě které mu systém zobrazí formulář pro vytvoření pravidla.
3. Uživatel vyplní formulář a klikne na „Vytvořit“.
4. Systém potvrdí vytvoření nového pravidla a přesměruje uživatele do části UC2.6 - Zobrazení pravidel přístupu.

UC3.2 - Zneplatnění pravidla přístupu

V tomto případu užití chce uživatel zneplatnit existující pravidlo.

Aktér: Správce pravidel

Zahrnuje: UC2.6 - Zobrazení pravidel přístupu

Scénář:

1. Uživatel provede UC2.6 - Zobrazení pravidel přístupu.
2. Uživatel vybere pravidlo k zneplatnění a klikne u něj na možnost „Zneplatnit“.
3. Systém potvrdí zneplatnění pravidla.

UC3.3 - Delegace oprávnění správce

V tomto případu užití chce uživatel delegovat oprávnění na jinou identitu, která se nově stane správcem pravidel pro dané dveře.

Aktér: Správce pravidel

Zahrnuje: UC2.8 - Zobrazení oprávnění správců pravidel

Scénář:

1. Uživatel provede UC2.8 - Zobrazení oprávnění správců pravidel.
2. Uživatel vybere oprávnění a klikne na možnost „Delegovat“.
3. Systém zobrazí formulář pro možnost delegovat oprávnění na jiné identity.
4. Uživatel vyplní formulář a klikne na „Delegovat“.
5. Systém potvrdí delegaci oprávnění a přesměruje uživatele do části UC2.8 - Zobrazení oprávnění správců pravidel.

UC3.4 - Zneplatnění delegace práva správce

V tomto případu užití chce uživatel zneplatnit delegaci, kterou udělil.

Aktér: Správce pravidel

Zahrnuje: UC2.8 - Zobrazení oprávnění správců pravidel

Scénář:

1. Uživatel provede UC2.8 - Zobrazení oprávnění správců pravidel.
2. Uživatel vybere delegaci k zneplatnění a klikne u ní na možnost „Zneplatnit“.
3. Systém potvrdí zneplatnění delegace.

UC4.1 - Vytvoření delegace správce pravidel

Tento případ užití popisuje vytvoření delegace pro správce pravidel.

Aktér: Administrátor aplikace

Zahrnuje: UC2.8 - Zobrazení oprávnění správců pravidel

Scénář:

1. Uživatel provede UC2.8 - Zobrazení oprávnění správců pravidel.
2. Uživatel vybere možnost „Vytvořit oprávnění“.
3. Systém zobrazí formulář.
4. Uživatel vyplní formulář a klikne na „Vytvořit“.
5. Systém potvrdí vytvoření nového správce a přesměruje uživatele do části UC2.8 - Zobrazení oprávnění správců pravidel.

UC4.2 - Zneplatnění delegace správce pravidel

Tento případ užití popisuje zneplatnění delegace správce pravidel.

Aktér: Administrátor aplikace

Zahrnuje: UC2.8 - Zobrazení oprávnění správců pravidel

Scénář:

1. Uživatel provede UC2.8 - Zobrazení oprávnění správců pravidel.
2. Uživatel vybere delegaci k zneplatnění a klikne u ní na možnost „Zneplatnit“.
3. Systém potvrdí zneplatnění delegace.

UC4.3 - Přidání dveří

V tomto případu užití chce uživatel přidat do ACS-M nové dveře.

Aktér: Administrátor aplikace

Zahrnuje: UC2.2 - Zobrazení dveří

Scénář:

1. Uživatel provede UC2.2 - Zobrazení dveří.
2. Uživatel vybere možnost „Vytvořit dveře“.
3. Systém zobrazí formulář.
4. Uživatel vyplní formulář a klikne na „Vytvořit“.
5. Systém potvrdí přidání nových dveří a přesměruje uživatele do části UC2.2 - Zobrazení dveří.

UC4.4 - Odebrání dveří

Tento případ užití popisuje odebrání dveří z ACS-M.

Aktér: Administrátor aplikace

Zahrnuje: UC2.2 - Zobrazení dveří

Scénář:

1. Uživatel provede UC2.2 - Zobrazení dveří.
2. Uživatel vybere dveře k odstranění a klikne u nich na možnost „Odstranit“.
3. Systém potvrdí odstranění dveří.

UC4.5 - Úprava dveří

Tento případ užití popisuje úpravu dveří v ACS-M.

Aktér: Administrátor aplikace

Zahrnuje: UC2.2 - Zobrazení dveří

Scénář:

1. Uživatel provede UC2.2 - Zobrazení dveří.
2. Uživatel vybere dveře k úpravě a klikne u nich na možnost „Upravit“.
3. Systém zobrazí formulář.
4. Uživatel upraví data ve formuláři a klikne na „Upravit“.
5. Systém potvrdí upravení dveří a přesměruje uživatele do části zobrazující všechny existující (UC2.2).

2.2.2 Případy užití lokálního režimu

V této kapitole jsou zahrnuty průchody, které je možné vykonávat pouze v lokálním režimu. Jedná se pouze o jeden případ užití, který pokrývá funkcionalitu importu dveří.

UC5.1 - Import dveří

Tento případ užití popisuje importování dveří do ACS-M.

Hlavní průchod: Import bude probíhat skrze serverové CLI, konkrétně pomocí příkazu v konzoli ACS-M.

Aktér: Správce serveru

Předpoklad: Přístup k CLI serveru, na kterém aplikace běží.

Alternativní průchod: Alternativně bude možné provést import dveří skrze uživatelské rozhraní ACS-M.

Aktér: Administrátor aplikace

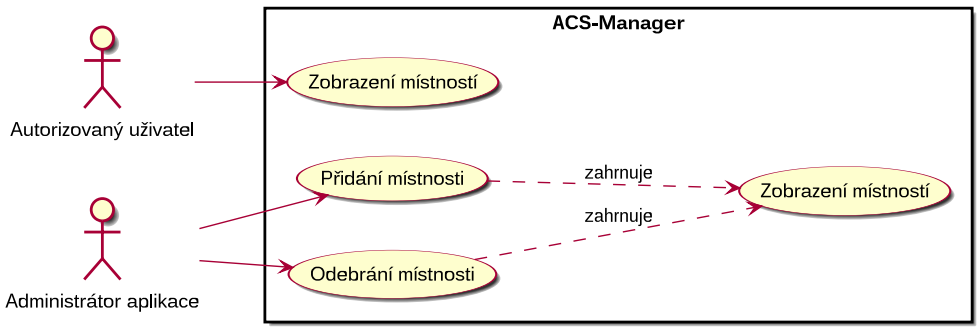
Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Importovat dveře“.
3. Systém zobrazí formulář pro import dveří.
4. Uživatel vyplní formulář a klikne na „Importovat“.
5. Systém potvrdí import dveří.

2.2.3 IdM případy užití

V této kapitole jsou zahrnuty průchody, které je možné vykonávat pouze v IdM režimu. Jedná se o případy užití, které pokrývají funkcionalitu týkající se správy místností.

Místnosti jsou ve fakultní IdM službě získány s ČVUT služby Usermap. Aby bylo možné tyto místnosti v aplikaci ACS-M používat, je nutné je do aplikace nejprve přidat. Veškerá správa těchto místností je popsána případy užití zmíněnými níže.



Obrázek 2.3: IdM případy užití aplikace ACS-M

Vzhledem k tomu, že se v IdM režimu přistupuje ke skupině dveří stejně jako ke dveřím v lokálním režimu, nejsou explicitně uvedeny případy užití pro skupinu dveří. Veškerá práce se skupinami dveří je pokryta případy užití uvedenými v tabulce 2.1.

Případ užití	Pokrytí
UC2.10 - Zobrazení skupiny dveří	UC2.2 - Zobrazení dveří
UC2.11 - Zobrazení detailu skupiny dveří	UC2.3 - Zobrazení detailu dveří
UC4.9 - Přidání skupiny dveří	UC4.3 - Přidání dveří
UC4.10 - Odebrání skupiny dveří	UC4.4 - Odebrání dveří
UC4.11 - Úprava skupiny dveří	UC4.5 - Úprava dveří

Tabulka 2.1: Pokrytí IdM případů užití skupin dveří

UC4.6 - Zobrazení místností

V tomto případu užití si chce uživatel zobrazit seznam všech místností.

Aktér: Administrátor aplikace

Scénář:

1. Systém zobrazí dashboard ACS-M.
2. Uživatel vybere možnost „Místnosti“.
3. Systém zobrazí všechny místnosti.

UC4.7 - Přidání místnost

V tomto případu užití chce uživatel přidat do ACS-M novou místnost.

Aktér: Administrátor aplikace

Zahrnuje: UC4.6 - Zobrazení místností

Scénář:

1. Uživatel provede UC4.5 - Úprava dveří.
2. Uživatel vybere místnost k přidání a klikne na „Přidat“.
3. Systém potvrdí přidání nové místnosti.

UC4.8 - Odebrání místnosti

Tento případ užití popisuje odebrání místnosti z ACS-M.

Aktér: Administrátor aplikace

Zahrnuje: UC4.6 - Zobrazení místností

Scénář:

1. Uživatel provede UC4.5 - Úprava dveří.
2. Uživatel vybere místnost k odebrání a klikne na „Odebrat“.
3. Systém potvrdí odebrání místnosti.

Kapitola 3

Návrhová fáze

Tato kapitola popisuje samotný návrh aplikace ACS-M. Nejprve v podkapitole 3.1 Aplikační logika přibližuji princip fungování aplikace. Následně v části 3.2 Datový model definuji strukturu datového modelu ACS-M. A na závěr v podkapitole 3.3 Rozhraní popisuji rozhraní, jenž aplikace poskytuje.

3.1 Aplikační logika

V této podkapitole je přiblížen princip fungování aplikace ACS-M. Popisuji zde chod aplikace v obou režimech provozu, využití sdíleného balíku funkcí Identity[6] a způsob komunikace aplikace se čtecími zařízeními u jednotlivých dveří.

3.1.1 Režimy aplikace

Jak již bylo zmíněno v 1. kapitole Analytická fáze, ACS-M bude umožňovat chod aplikace ve dvou režimech. Režim aplikace bude určen na základě její konfigurace. ACS-M bude tuto konfiguraci reflektovat za pomoci návrhového vzoru strategie.

První z nich je režim IdM, ve kterém pracuje aplikace ACS-M pouze s daty získanými z fakultní IdM služby. V tomto režimu tedy aplikace funguje pouze jako jakési front-end uživatelské rozhraní pro práci s IdM daty.

Druhý je lokální režim, ve kterém má aplikace veškerá data uložena v lokální databázi. Vzhledem k tomu, že ACS-M bude využívat část dat identických s aplikací Card-Manager, bude databáze s touto aplikací sdílena.

3.1.2 Identity

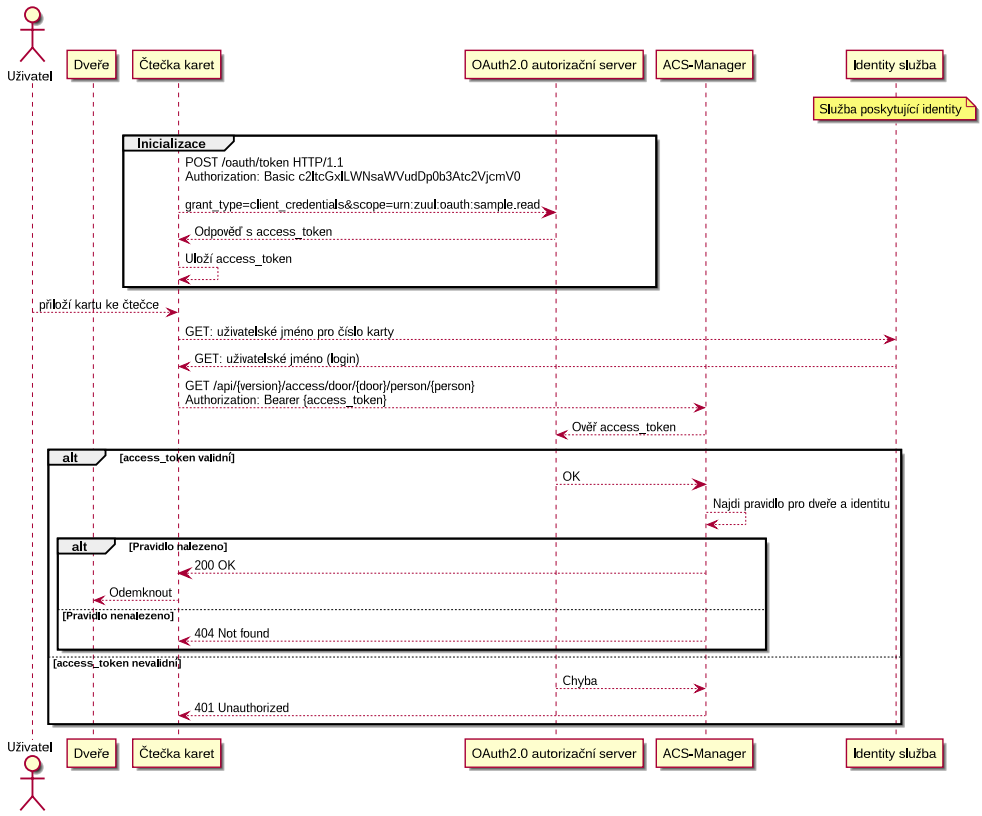
Identity představují data o osobách. Jedná se o osobní údaje jednotlivých osob a jejich role. Tato data jsou identická jak pro aplikaci ACS-M, tak i C-M a jsou jimi vzájemně sdílena.

K tomuto účelu je vytvořen v rámci Oddělení ICT FIT ČVUT balík funkcí Identity[6]. Tento balík pokrývá požadovanou funkcionalitu obou aplikací pro práci s identitami v obou režimech provozu aplikace.

3.1.3 Aplikační rozhraní

ACS-M poskytuje REST API pro ověření přístupu. Toto API využívají všechna čtecí zařízení za účelem ověření a umožnění vstupu konkrétní osoby do dveří dotazovaného čtecího zařízení.

Požadavek musí obsahovat identifikátor dveří, identifikátor osoby a Authorization hlavičku. Ta se liší dle použitého režimu provozu aplikace. V případě lokálního režimu obsahuje údaje pro HTTP basic autorizaci, IdM režim využívá autorizační službu OAuth 2.0 a tedy v hlavičce se posílá její token. Celý proces v IdM režimu je znázorněn diagramem níže.



Obrázek 3.1: Příklad komunikace s API v IdM režimu

3.1.4 Autentizace a autorizace uživatelů

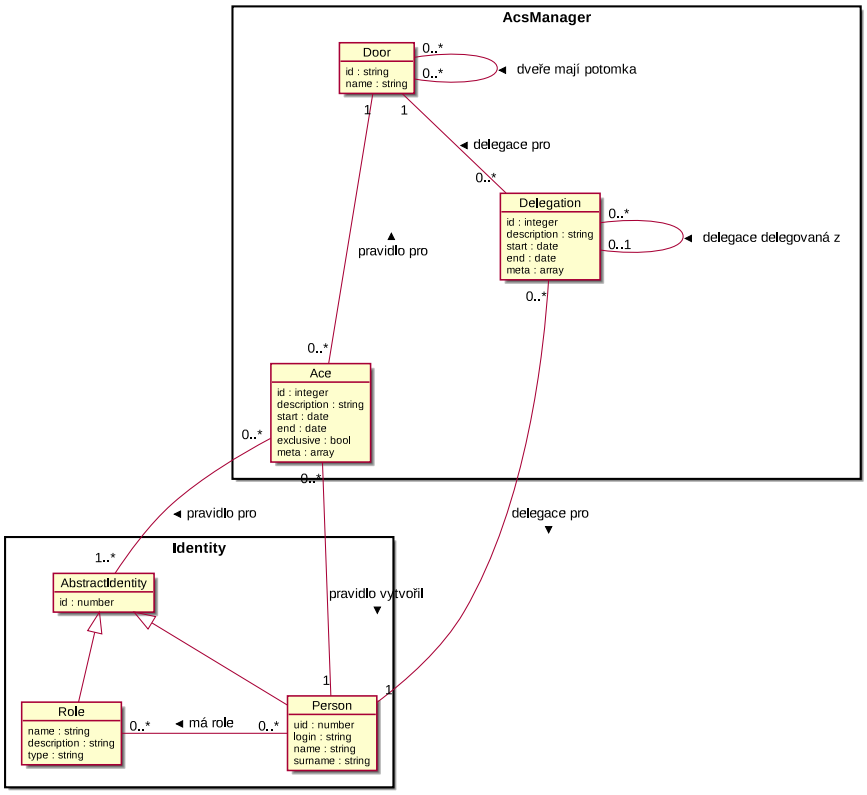
Autentizace a autorizace uživatelů je rozdílná dle režimu aplikace. V lokálním režimu probíhá autentizace i autorizace vůči datům z lokální databáze.

V režimu IdM probíhá autentizace vůči fakultní Shibboleth službě. Autorizace získaného uživatele pak probíhá na základě jeho rolí získaných z fakultní IdM služby.

3.2 Datový model

Datový model popisuje strukturu dat aplikace a vazby mezi jednotlivými entitami. Entity jsou objekty datového modelu aplikace a zapouzdřují data používaná aplikací. Části datového modelu se liší s ohledem na použitý režim aplikace, protože ve fakultní IdM službě jsou data reprezentována v obecném datovém modelu, za účelem možného použití v dalších aplikací. Tyto rozdíly jsou popsány níže.

3.2.1 Lokální režim



Obrázek 3.2: Datový model lokálního režimu aplikace ACS-M

V lokálním režimu jsou entity rozděleny do dvou částí. První část reprezentuje samotný datový model aplikace ACS-M, konkrétně se jedná o entity Ace, Delegation a Door. Druhá část reprezentuje data identit abstraktní předek AbstractIdentity a jeho potomci Person a Role.

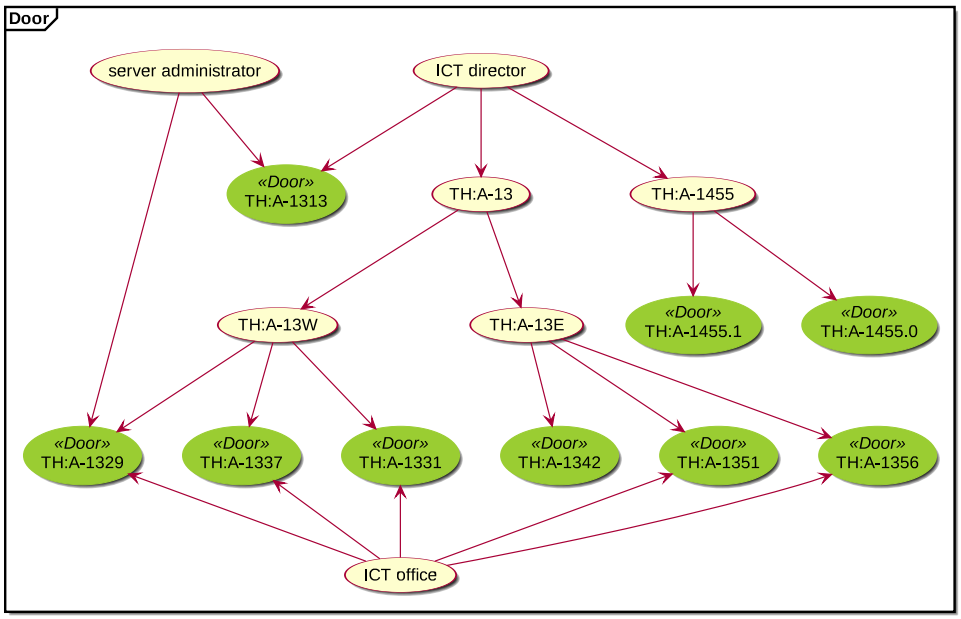
3.2.1.1 Door

Entita Door reprezentuje dveře nebo skupinu dveří, k nimž je řízen přístup. Dveře mohou být v 0..n skupinách a zároveň skupina dveří může mít 1..n dveří a nebo skupin. Atributem této entity je její identifikátor a pro člověka čitelný název.

Ke vzniku skupiny dveří dochází na základě vztahu mezi dvěma entitami Door. Dveře od skupiny rozlišujeme tak, že skupina má alespoň jednoho potomka, kdežto dveře samotné nemají žádného potomka.

Na základě vlastností entity Door zmíněných výše vyplývá, že dveře spolu se skupinami lze vyjádřit formou orientovaného grafu. Graf nemá jeden konkrétní kořen a má n koncových uzlů, které reprezentují dveře. Všechny ostatní uzly považujeme za skupiny dveří. V grafu nemohou vznikat smyčky.

Na diagramu níže je znázorněn příklad takového grafu. Zeleně jsou označeny uzly koncové, tedy dveře, žlutě pak uzly reprezentující skupiny.



Obrázek 3.3: Příklad struktury entity Door

3.2.1.2 Ace

Entita Ace reprezentuje pravidla přístupu. Každé takové pravidlo určuje dveře nebo skupinu dveří, ke kterým je přístup udělen, a 1 až n osob a/nebo skupin osob, jež mají k přístupu oprávnění. Dále pravidlo může mít popis, časové omezení platnosti od - do a má informaci, jaká osoba a kdy jej vytvořila.

Pravidlu je možné udělit příznak exkluzivity. Taková pravidla jsou využívána pro udělení dočasného přístupu do dveří pouze vybraným osobám (např. krátkodobý pronájem respiria). Exkluzivní pravidla v daný čas pro dané dveře ruší všechna pravidla neexkluzivní.

Pravidla nelze upravovat, lze je pouze smazat. Veškeré operace s pravidly jsou logovány pro umožnění provádění jejich auditu.

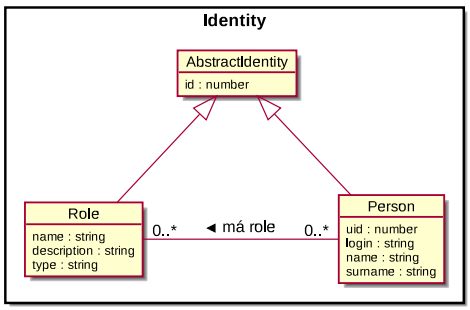
3.2.1.3 Delegation

Entita Delegation reprezentuje pravidla pro správce. To jsou takové identity, které mají oprávnění na vytváření pravidel pro správu přístupu. Právo správce je přiděleno konkrétní osobě a jedněm konkrétním dveřím, respektive skupině dveří, k nimž je možné pravidla vytvářet.

Může mít popis, časově omezenou platnost od - do a volitelně je možné právo správce delegovat na identitu jinou. Delegované právo správce má omezenou dobu trvání na základě práva, ze kterého bylo delegováno. Stejně jako v případě pravidel, delegace nelze upravovat, lze je pouze smazat.

3.2.1.4 AbstractIdentity

AbstractIdentity je abstraktním předkem osoby, respektive role. Tato entita je reprezentována číselným identifikátorem. Hlavním účelem tohoto předka je možnost vytvářet pravidla jak pro jednotlivé osoby, tak i pro jejich skupinu na základě business rolí.



Obrázek 3.4: Datový model Identity balíku

3.2.1.5 Person

Entita Person reprezentuje jednu konkrétní osobu. Každá osoba má své vlastní unikátní osobní číslo, unikátní uživatelské jméno a dále jméno a příjmení. Každá osoba může mít 0 až n rolí.

3.2.1.6 Role

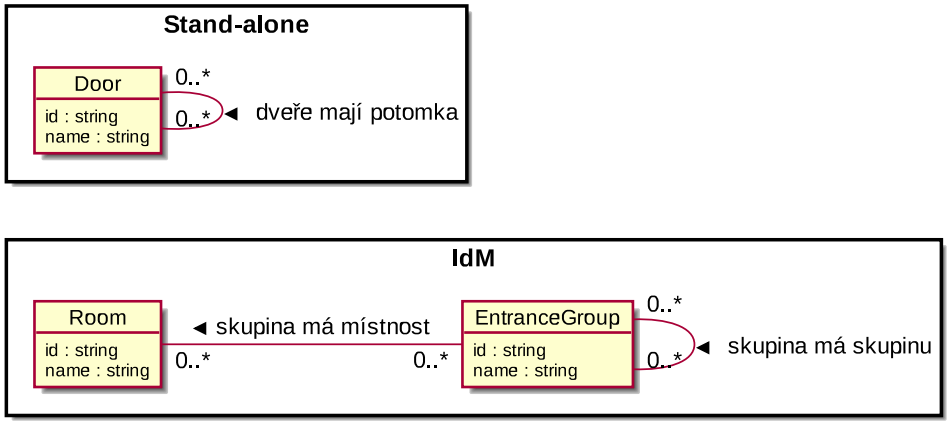
Entita Role představuje jednotlivé role osob. Role je reprezentována jejím názvem, popisem a typem, jež rozlišuje, zda-li se jedná o interní roli aplikace nebo obecně business roli.

3.2.2 IdM režim

IdM režim pracuje s obecným datovým modelem. Struktura dat se tak oproti datovému modelu ACS-M liší a promítla se do všech entit ACS-M.

3.2.2.1 Door

Entita Door je na úrovni IdM rozdělena do dvou podentit. První z nich je Room, která reprezentuje místnosti. Druhá pak EntranceGroup, jež reprezentuje skupinu vstupů.



Obrázek 3.5: Porovnání reprezentace Door v lokálním a IdM režimu

Entita Room vychází z reprezentace dat o místnostech v rámci ČVUT. To na úrovni aplikací nepracuje s jednotlivými vstupy, ale s celými místnostmi. Data o místnostech jsou uložena v ČVUT službě Usermap. IdM služba tato data s Usermap službou synchronizuje.

Entita EntranceGroup reprezentuje skupinu místností. V této skupině mohou být obsazeny jak jednotlivé místnosti (entita Room), tak i další skupiny

místností (entita EntranceGroup). Stejně jako v případě entity Door v lokálním režimu platí, že EntranceGroup tvoří orientovaný graf bez cyklů.

3.2.2.2 Ace a Delegation

Naopak entity Ace a Delegation jsou v rámci IdM reprezetovány jednou entitou Entitlement. Pravidla a delegace od sebe rozlišuje atribut activity.

Důvodem tohoto sjednocení je velká podobnost obou entit a možné opětovné znovupoužití entity Entitlement k reprezentaci dalšího podobného datového nosiče v rámci IdM.

V tomto režimu se pravidla ani oprávnění nemažou, ale prohlásí se za neplatné. Ke smazání může dojít až po uplynutí ochranné lhůty 3 měsíců a pouze poté, co je vytvořen a uložen textový export. Do té doby jsou v IdM pravidla uložena s příznakem smazaná.

3.3 Rozhraní

Kapitola rozhraní popisuje možnosti komunikace s ACS-M. Aplikace poskytuje webové uživatelské rozhraní pro správu delegací a pravidel, konzolový režim pro účely správy aplikace a výše zmíněné REST aplikační rozhraní.

3.3.1 Webové uživatelské rozhraní

ACS-M poskytuje uživatelské rozhraní pro:

- správu pravidel,
- správu oprávnění správců pravidel,
- správu dveří, respektive místností a jejich skupin,
- přehledy identit.

Návrh tohoto rozhraní je popsán v samostatné podkapitole 3.4 - Uživatelské rozhraní.

3.3.2 Konzolový režim

Konzolový režim aplikace poskytuje funkce pro:

- zálohování a mazání pravidel,
- import dveří,
- určení administrátora aplikace v případě lokálního režimu.

3.3.3 REST API

Aplikace ACS-M poskytuje REST aplikační rozhraní. To slouží pro účely ověření, zda má daná osoba přístup do daných dveří, či nikoliv. Konkrétní příklad lze vyčíst z diagramu v části Aplikační rozhraní podkapitoly 3.1 Aplikační logika.

Toto API poskytuje jednu HTTP GET metodu, která se nachází na adrese */api/{version}/access/door/{door}/person/{person}*, kde *{version}* značí verzi API, *{door}* je identifikátorem konkrétních dveří a *{person}* je uživatelské jméno sloužící k identifikaci osoby.

3.4 Uživatelské rozhraní

Cílem této práce je vytvořit a zdokumentovat aplikaci pro správu pravidel přístupu, kterou bude možné nasadit a používat v podmínkách FIT ČVUT. Vývoj takové aplikace s dodržením běžných či doporučených postupů ve všech částech svým rozsahem řádově převyšuje časovou dotaci bakalářské práce. Z toho důvodu bylo nutné zpracovat některé části práce tak, aby bylo možné aplikaci implementovat bez ohledu na běžné či doporučené postupy.

Jelikož cílovou skupinou aplikace je přibližně deset osob, byl po dohodě s vedoucím práce návrh a ověření uživatelského rozhraní řešen odlišně vůči doporučeným postupům, jež stanovuje i dokument Pravidla a zásady projektů FIT[1]. Návrh uživatelského rozhraní byl pouze konzultován s vedoucím práce bez ověření s uživateli. To by mohlo být náplní dalšího rozšíření této práce.

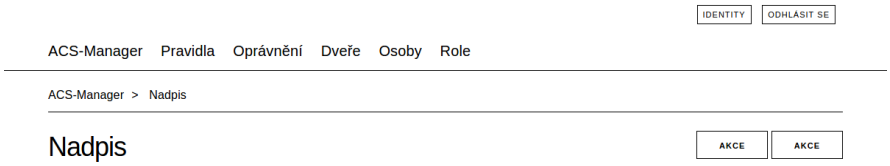
Návrh uživatelského rozhraní ACS-M je inspirován aplikacemi, které byly Oddělením ICT FIT vyvinuty. Důraz je kladen zejména na rozložení funkčních prvků tak, aby se uživatelé, již tyto aplikace používají, snadno orientovali.

Uživatelské rozhraní ACS-M je tvořeno třemi základními pohledy: výpisem seznamu záznamů, detailem záznamu a formulářem pro jeho vytvoření. Jednotlivé pohledy jsou popsány v samostatných částech níže.

3.4.1 Hlavička

Základním stavebním kamenem všech pohledů je hlavička vyobrazena na obrázku 3.6. V pravé horní části se nachází menu pro odhlášení uživatele a přepnutí do uživatelského rozhraní balíku Identity. Pod ním se nachází hlavní menu, kde jednotlivé položky odkazují na výpis seznamu dané entity.

Pod hlavním menu se nachází drobečková navigace, jež usnadňuje uživateli orientaci v pohledech. Nalevo pod ní se nachází nadpis aktuálního pohledu. Napravo od něj pak ovládací prvky k akcím, které je možné v rámci daného pohledu vykonat.

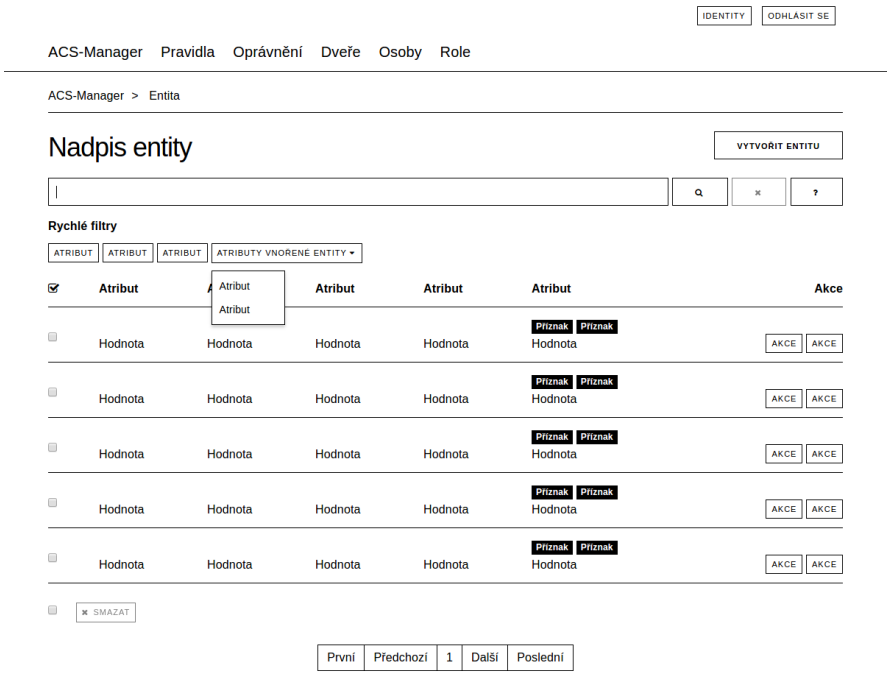


Obrázek 3.6: Návrh hlavičky aplikace ACS-M

3.4.2 Seznam záznamů

Pohled seznamu zobrazuje výpis všech záznamů dané entity. Tento výpis je stránkovaný a je v něm možné filtrovat. K filtraci slouží textový vstup spolu s ovládacími prvky, jenž se nachází pod nadpisem pohledu. K zjednodušení filtrace je možné použít i tzv. rychlé filtry, jež jsou vytvořeny pro všechny atributy dané entity.

Výpis záznamů se vykresluje v tabulce. V jejích sloupcích jsou podstatné atributy pro danou entitu a v řádcích pak samotné hodnoty atributů záznamů. Seznam atributů pro jednotlivé pohledy se nachází níže. Nad každým ze záznamů je možné dělat akce, jež se nachází v pravé části tabulky. Nad celou tabulkou je poté možné provádět hromadné mazání. Ve spodní části pohledu se nachází ovládací prvek pro stránkovaní.



Obrázek 3.7: Návrh pohledu pro výpis seznamu záznamů

Pravidla zobrazují název dveří, název identity nebo jejich počet, počáteční datum, koncové datum, popis, příznak exkluzivity a případně příznak značící, že se jedná o aktivní pravidlo.

Oprávnění zobrazují název dveří, jméno a příjmení osoby, počáteční datum, koncové datum, popis, příznak delegace a případně příznak aktivního pravidla.

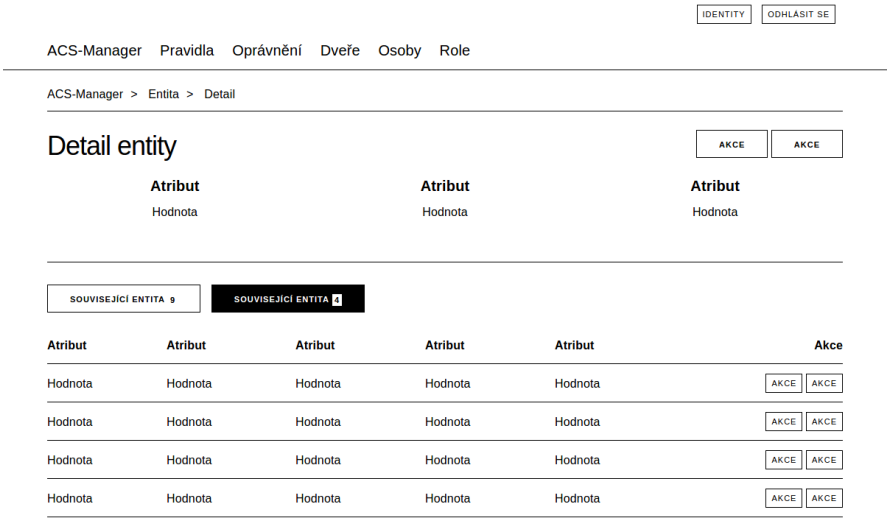
Dveře zobrazují identifikátor, název a příznak značící, zda-li se jedná o dveře nebo jejich skupinu.

Osoby zobrazují osobní číslo, uživatelské jméno, jméno, příjmení a počet rolí.

Role zobrazují identifikátor, název a příznak značící, zda-li se jedná o interní nebo externí roli.

3.4.3 Detail záznamu

Tento pohled detailně vypisuje hodnoty atributů vybraného záznamu. Společně s atributy jsou v některých pohledech zobrazeny také související entity.



Obrázek 3.8: Návrh pohledu detailu záznamu

Detail záznamu obsahuje ve výpisu všechny atributy pro danou entitu zmíněné v podsekci 3.4.2 - Seznam záznamů. Atributy, jež se vyskytují v detailu a nejsou vypsány v seznamu, jsou zmíněny níže.

Pravidlo navíc zobrazuje jeho identifikátor a osobu, která jej vytvořila. Dále jsou ve výpisu vypsány všechny identity, pro které platí.

Oprávnění navíc zobrazuje jeho identifikátor.

Dveře vypisují vztažený seznam oprávnění a pravidel, které pro ně platí. V případě, že se jedná o skupinu, je zobrazen seznam dveří, které do této

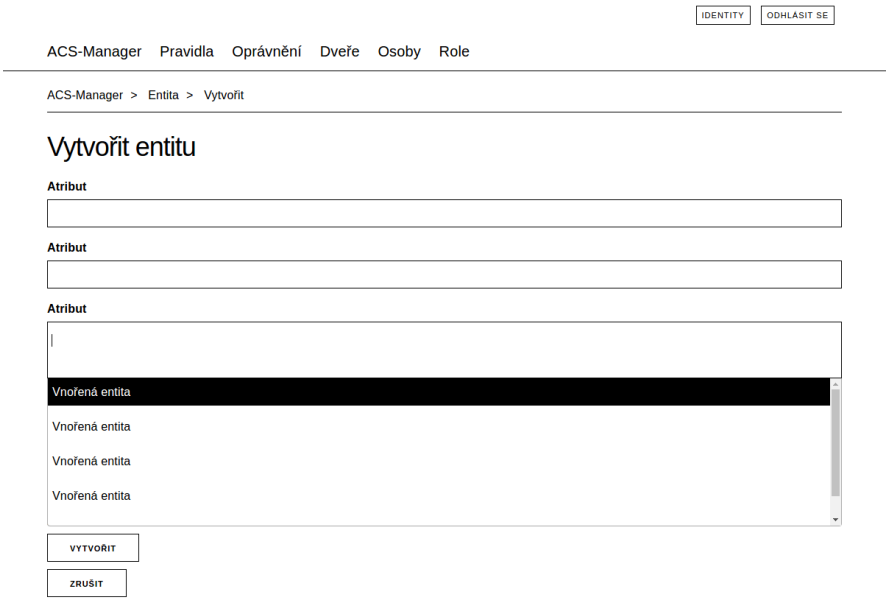
skupiny spadají.

Osoba má ve výpisu seznam jejích rolí, oprávnění, pravidla, která vytvořila, a pravidla, která pro ni platí.

Role vypisuje seznam pravidel, která pro ni platí.

3.4.4 Formuláře

Formuláře slouží k vytváření nových záznamů entit. Hodnoty jednotlivých atributů jsou zadávány pomocí vstupů, které se liší v závislosti na typu atributu. Vstupem může být čistý text, výběr data nebo výběr ze seznamu, ve kterém lze vyhledávat.



Obrázek 3.9: Návrh pohledu formuláře

Kapitola 4

Implementační fáze

Tato kapitola popisuje implementační fázi aplikace ACS-M. Celá implementační dokumentace je součástí přiložených zdrojových kodů aplikace. V této kapitole popisuji pouze podstatné části implementační fáze s ohledem na dokument Pravidla a zásady projektů FIT[1], jimiž se tato práce řídí.

Na úvod nejpre představuji použité technologie, následně popisuji architekturu celé aplikace. Poté přibližuji způsob řešení provozu aplikace ve dvou režimech, dále popisuji řešení událostí v aplikaci a na závěr definuji aplikační programové rozhraní.

4.1 Použité technologie

V této podkapitole popisuji použité technologie k realizaci implementace aplikace ACS-M. Vybrané technologie byly zvoleny v souladu s doporučením dokumentu Pravidla a zásady projektů FIT[1].

4.1.1 Symfony PHP Framework

Symfony je sada znovupoužitelných PHP komponent a PHP framework pro webové projekty.[7] Jedná se o celosvětově rozšířený nástroj pro tvorbu webových aplikací. Stavebním kamenem tohoto frameworku je jeho jádro, na nějž jsou navázány komponenty.

Komponenta je obecně balíček funkcí, jenž se jako celek týká nějaké problematiky. Symfony se spolu s komponentami skládá pomocí nástroje Composer[8], který je správcem závislostí mezi nimi. Pomocí Composeru je možné rovněž tyto balíčky aktualizovat.

Technologie PHP a framework Symfony byly pro implementaci vybrány jednak z důvodu doporučení zmíněných v úvodu této podkapitoly. Druhým faktorem je pak zachování konzistence projektů v rámci Oddělení ICT FIT,

kde je velké množství webových aplikací implementováno právě v tomto frameworku.

4.1.1.1 Komponenty

K implementaci aplikace ACS-M byla využita celá sada komponent Symfony frameworku. Ta však nepokrývá veškerou potřebonou funkcionalitu, a tak k realizaci aplikace bylo rovněž využito dalších komponent:

Guzzle je rozšiřitelný PHP HTTP klient[9], který poskytuje sadu funkcí pro posílání HTTP požadavků.

FOSRestBundle poskytuje sadu různých nástrojů pro vytváření RESTful aplikačních rozhraní v Symfony[10].

KnpMenuBundle je knihovna pro tvorbu menu za pomoci objektově orientované deklarace[11].

4.1.2 Persitence datového uložiště

Jak již bylo zmíněno v 3. kapitole Návrhová fáze, aplikace ACS-M funguje ve dvou režimech provozu. V každém z nich pracuje s rozdílným datovým uložištěm.

4.1.2.1 Doctrine

V lokálním režimu provozu je jako uložiště dat použita relační databáze. Pro komunikaci s databází je použito objektově relační mapování (ORM). To zajišťuje automatickou konverzi dat databáze na objekty aplikace. K realizaci ORM je použita PHP knihovna Doctrine[12].

Doctrine poskytuje abstrakci nad relační databází. Díky tomu je možné v aplikaci pracovat s libovolnou podporovanou relační databází. V případě aplikace ACS-M je využito databázového systému MySQL. Ten je platformově nezávislý a ke komunikaci využívá jazyk SQL.

4.1.2.2 Identity Management

IdM režim provozu používá jako datové uložiště Identity Management službu FIT ČVUT. IdM slouží primárně ke správě identit a jejich rolí v rámci fakulty. Služba je postavena na technologii OpenIDM.

Komunikace s IdM probíhá skrze vystavené RESTful API. Stejně jako v případě lokálního režimu i zde je potřeba převádět získaná data na objekty. K tomuto účelu byly zkonstruovány normalizéry, jejichž popis se nachází v podkapitole 4.4 - Datový model.

4.1.3 Frontendové technologie

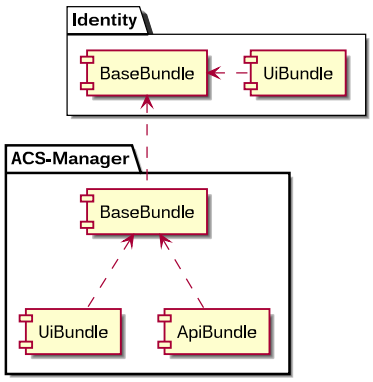
Při realizaci uživatelského rozhraní jsem využil několik javascriptových komponent postavených na knihovně jQuery, jež mi poskytly pokročilejší funkce pro realizaci použitelného uživatelského prostředí (např. výběr data a času, interaktivní selekce ve formulářích apod.).

Stěžejní při vytváření pohledů na kolekci dat byla knihovna Underscore[13] a její rozšíření underscore-query[14]. To poskytuje rozhraní pro filtraci kolekcí, jež je v těchto pohledech využito. Popis implementace této knihovny se nachází v samostatné podkapitole 4.5 - Filtrace kolekcí.

4.2 Architektura aplikace

Jak již bylo zmíněno v podkapitole 4.1 - Použité technologie, aplikace je postavena na frameworku Symfony. Ten poskytuje vícevrstvou architekturu, jež od sebe odděluje datový model, aplikační logiku a rozhraní aplikace.

Aplikace je členěna do funkčních celků respektující DRY a SOLID principy designu aplikací. ACS-M tvoří dvě hlavní komponenty: samotné ACS-M a balík Identity. Každá z těchto komponent obsahuje takzvané Bundles (balíky), jež právě ohraničují funčkní celky aplikace.



Obrázek 4.1: Architektura komponent aplikace ACS-M

4.2.1 Acs-Manager

Samotné ACS-M je rozděleno do tří Symfony balíků, kde každý z nich zastřešuje rozdílné funkční celky aplikace. Rozdělení balíků uvedených níže respekuje zažité *„best practices“* Oddělení ICT FIT ČVUT.

BaseBundle je jádrem celé aplikace. Tento balík zastřešuje veškerou business logiku nad aplikací. Dále zajištuje persistenci dat s datovým uložištěm a realizuje strategii pro výběr režimu provozu aplikace.

UiBundle zastřešuje uživatelské rozhraní. V balíku jsou obsaženy jak funkce pro samotné renderování pohledů, tak obsahuje další nezbytné součásti k utvoření uživatelského rozhraní.

ApiBundle obsahuje aplikační programové rozhraní, jež umožňuje standardizovanou komunikaci s ostatními aplikacemi. API je popsáno v samostatné kapitole 4.8 - Aplikační programové rozhraní.

4.2.2 Identity

Komponenta Identity poskytuje sadu funkcí pro práci s osobami a jejich rolemi. Byla vytvořena mnou a kolegou Martin Vondrákem v rámci Oddělení ICT FIT ČVUT.

Motivací k jejímu vytvoření byla potřeba identického přístupu a práce s daty identit v této bakalářské práci i bakalářské práci Card-Manager: aplikace pro správu RFID karet kolegy Martina Vondráka.

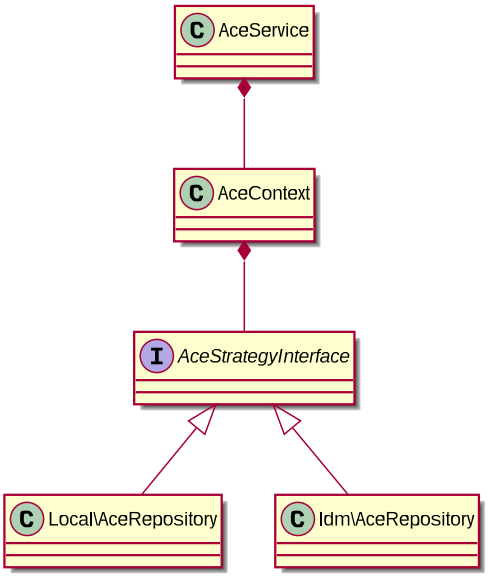
Komponenta Identity rovněž podporuje dva režimy provozu. V závislosti na nich se mění nabízená funkcionalita. V režimu IdM komponenta poskytuje funkce pro získání osob a rolí. Lokální režim provozu pak ještě navíc obsahuje CRUD funkce pro správu osob a rolí.

ACS-M využívá komponentu Identity ve dvou úrovních. První z nich jsou třídy business logiky (Services), druhou úrovní jsou pak třídy zajišťující persistenci dat (Repositories).

4.3 Strategie režimu provozu

Aplikace ACS-M podporuje dva režimy provozu: lokální a IdM. Každý z nich pracuje s jiným datovým uložištěm. Při návrhu implementace bylo cílem najít takové řešení, ve kterém by byla business logika aplikace stejná pro oba dva režimy provozu.

Díky tomu, že framework Symfony pracuje s vícevrstvou architekturou aplikace, bylo možné oddělit oba dva režimy provozu na úrovni tříd zajištující persistenci dat. Každý z režimů má k tomuto účelu vždy vytvořenou svou třídu, jež implementuje jednotný interface. Třídy business logiky pak pracují s tímto interfacem odstíněně od samotné implementace v obou režimech provozu.



Obrázek 4.2: Příklad tříd s využitím návrhového vzoru strategie

Implementace tohoto návrhu je docíleno za použití návrhového vzoru strategie. K výběr režimu provozu dochází na základě konfigurace ACS-M, jež je reflektována při sestavení kontejneru aplikace.

Sestavení kontejneru probíhá v Symfony technikou vkládání závislostí tzv. *dependency injection*. Velmi zjednodušeně to je metoda, během níž jsou při sestavení kontejneru inicializovány všechny zaregistrované třídy. Výhodou tohoto přístupu je, že třídy jsou inicializovány pouze jednou během běhu aplikace.

Během sestavování kontejneru aplikace je také zavolán tzv. *compiler pass*, pomocí nějž se dají ovlivňovat vkládané závislosti. Toho je využito i při realizaci návrhového vzoru strategie. V compiler passu je vyhodnocena konfigurace režimu provozu aplikace, na základě níž jsou poté do příslušných tříd business logiky dosazeny správné třídy pro persistenci dat.

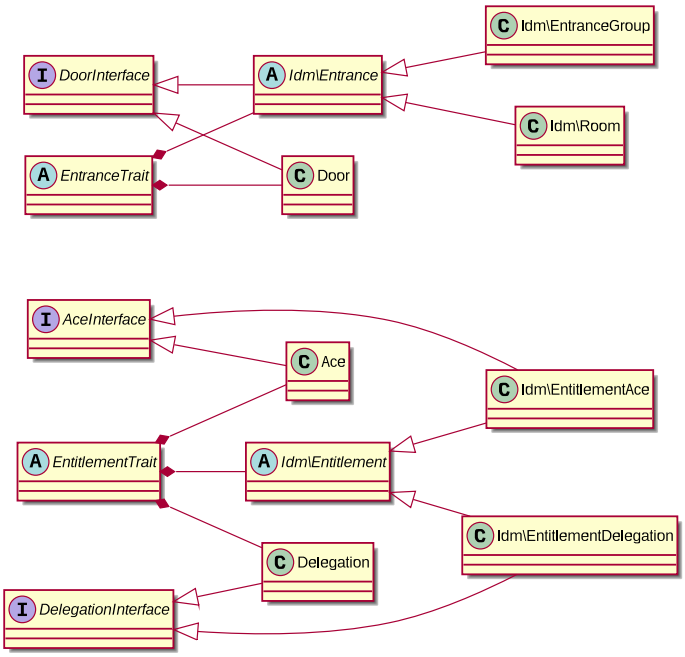
4.4 Datový model

Jak již bylo zmíněno v podkapitole 3.2 - Datový model kapitoly Návrhová fáze, aplikace ACS-M pracuje s různými datovými modely dle režimu provozu aplikace. Tomu bylo nutné rovněž přízpůsobit implementační návrh datového modelu.

Stejně jako v případě tříd pro persistenci dat jsou i entitní třídy rozděleny dle režimu provozu a implementují předem definovaný interface. Jedním z důvodů pro jejich rozdělení je odlišný způsob uchovávání informací o stavu persistence.

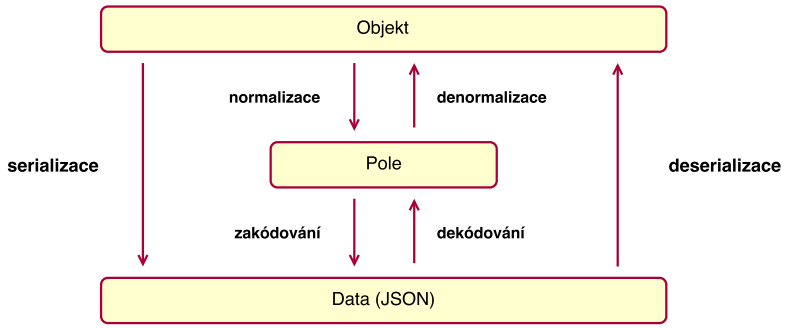
Obrázek 4.3: Interface entit ACS-M

I přes nastíněné důvody výše je, narozdíl od tříd pro persistenci dat, poměrně značná část jednotlivých tříd identická v obou režimech provozu. Z tohoto důvodu bylo při implementaci využito rozsáhlé míry abstrakce s ohledem na DRY princip vývoje aplikace.



Obrázek 4.4: Abstrakce entitních tříd

Jak jsem již zmiňoval v podkapitole 4.1 - Použité technologie, pro komunikaci s IdM je potřeba objekty ACS-M serializovat, respektive získaná data deserializovat. K tomuto procesu bylo využito Symfony komponenty Serializer a popisuje jej diagram 4.5.



Obrázek 4.5: Proces serializace a deserializace[15]

Dekódování, respektive zakódování umí implicitně sama komponenta. Pro denormalizaci, respektive normalizaci bylo potřeba vytvořit vlastní třídy, jež dokáží specificky převádět data na objekty a naopak.

4.5 Filtrace kolekcí

Jeden z funkčních požadavků aplikace ACS-M je umožnit uživateli filtrovat v datech pohledů zobrazující kolekce. Jak již bylo zmíněno v pokapitole 4.1 - Použité technologie, k realizaci tohoto požadavku jsem využil javascriptovou knihovnu Underscore.

Předpokladem pro úspěšné filtrování je jistá technická zdatnost uživatele. Té je však s ohledem na cílové uživatele v rámci FIT ČVUT dosaženo. Filtrování kolekcí probíhá pomocí dotazů, jež jsou svou syntaxí podobné dotazování v databázi Mongo. Pro rychlejší psaní těchto dotazů poskytuje ACS-M tzv. rychlé filtry, což jsou již předpřipravené dotazy pro konkrétní filtrování.

Filtrování probíhá v současné implementaci v celé kolekci, získané z datového uložiště ACS-M. Řešením v budoucnu by poté mohla být implementace dotazovacího jazyka, jenž bude schopný provádět tyto dotazy přímo v samotné databázi.

4.6 Uživatelské role a oprávnění

V 2. kapitole Případy užití jsem popisoval aktéry vystupující v případech užití ACS-M. Tito aktéři jsou v aplikaci reprezentováni uživatelskými rolemi, jež definují rozsah oprávnění v aplikaci.

4.6.1 Uživatelské role

Uživatelskou roli má přiřazenou každý z uživatelů aplikace. V lokálním režimu se o přiřazení stará ACS-M, v IdM režimu je role získána na základě dat z IdM a následně převedena na kompatibilní roli v ACS-M. Tento převod je nutný s ohledem na jinou reprezentaci rolí v IdM a frameworku Symfony.

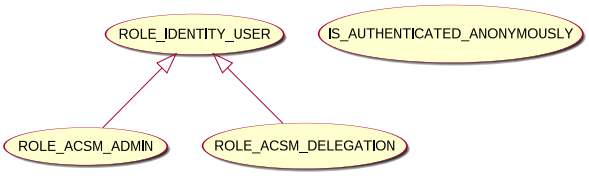
Uživatelské role aplikace však nemají všichni aktéři. Správce serveru pracuje pouze s konzolí aplikace, ve které již k autorizaci nedochází, s ohledem na její předešlé provedení při přístupu na server. Autorizovaný uživatel je abstraktním předkem pro správce pravidel a administrátora aplikace a z tohoto důvodu rovněž nemá svou roli.

Anonymní uživatel je neautentizovaný a neautorizovaný. Takovému uživateli je přiřazena role *IS_AUTHENTICATED_ANONYMOUSLY*. Symfony tuto roli přiřazuje implicitně.

Administrátor aplikace má práva pro správu oprávnění a dveří. Jeho uživatelská role je *ROLE_ACSM_ADMIN*. Její ekvivalent v IdM je technická role *T-ACS-18000-ADMINISTRATOR*.

Správce pravidel má práva související s agendou správy pravidel přístupu. Jeho role je *ROLE_ACSM_DELEGATION* a její ekvivalent v IdM je opět technichá role *T-ACS-18000-SPRAVCE-PRISTUPU*.

Do role správce pravidel a administrátora aplikace se ještě projeví role balíku Identity. Obě tyto role obsahují implicitně roli *ROLE_IDENTITY_USER*, jež je opravňuje k práci s osobami a jejich rolemi. Hierarchie uživatelských rolí je znázorněna na diagramu 4.6.



Obrázek 4.6: Hierarchie uživatelských rolí

4.6.2 Oprávnění správce pravidel

Samotná role správce pravidel však nedokáže pokrýt všechna jeho práva v aplikaci. Uživateli s touto rolí se totiž zobrazuje obsah dle jeho aktivních oprávnění správce pravidel. To znamená, že daný uživatel vidí pouze dveře, k nimž má aktuálně oprávnění správce pravidel, a pravidla, jež jsou k těmto dveřím vázaná.

K účelu této filtrace obsahu jsou použity tzv. *votery*. Použití voterů je jeden ze způsobů, jak v Symfony autorizovat uživatele vůči danému obsahu. Votery jsou samostatné třídy, v nichž je implementována libovolná logika, jež tuto autorizaci provádí.

Oprávnění správce pravidel má však ještě další úskalí. Uživatel tuto roli získá na základě aktivního oprávnění pro správu pravidel. Roli je tedy potřeba dynamicky přidělovat a odebírat. V IdM režimu se o to stará samotná IdM služba, pro lokální režim je k tomuto účelu vytvořen konzolový příkaz, jenž se spouští pomocí CRON mechanismu v předem definovaných intervalech.

Použitím tohoto mechanismu se může stát, že uživatel, kterému vypršela všechna aktivní oprávnění správce pravidel, bude stále disponovat rolí správce pravidel, dokud nebude provedeno odebrání jeho role konzolovým příkazem. Díky použití mechanismu voterů je však docíleno toho, že v takovém případě se uživateli již žádný obsah nezobrazí a tedy v aplikaci již žádné akce nemůže provádět.

4.7 Události

Framework Symfony nabízí, stejně jako mnoho dalších frameworků, sadu funkcí pro obsluhu událostí. Událost značí udání nějaké akce během vykonávání požadavku v aplikaci a je ji možné vyvolat libovolně v jakémkoliv místě kódu.

Na údalosti jsou navázány tzv. listenery, které na ně reagují. Výhodou systému událostí je programová nezávislost. Listener události je možné zaregistrovat bez jakéhokoliv zásahu nebo ovlivnění části aplikace, jež k vyvolání události vedla.

Systému událostí v ACS-M je využito v repository třídách lokálního režimu provozu aplikace. Jednou z hlavních motivací jeho použití je logování. Díky tomuto systému je možné při mazání některé z instancí entit podrobně do logu zaznamenat veškeré vyvolané akce v závislosti na této události.

4.8 Aplikační programové rozhraní

Aplikační programové rozhraní (API) poskytuje rozhraní pro zisk dat a komunikaci s aplikací navenek. Pro implementaci API jsem použil RESTful architekturu, která poskytuje platformově nezávislý interface pro zisk dat z aplikace. K tomuto rozhraní se přistupuje skrze definovaná URI.

4.8.1 Ověření přístupu do dveří

Stěžejním RESTful rozhraním aplikace ACS-M je zdroj sloužící k účelům ověření přístupu osoby do dveří. Tento zdroj je zabezpečen technologií OAuth2.0, díky které k němu mají přístup pouze autorizované služby.

Požadavek

GET */api/{version}/access/door/{door}/person/{person}*

Parametry URL požadavku

- *{version}* verze API
- *{door}* identifikátor dveří
- *{person}* uživatelské jméno osoby

Hlavička požadavku

Authorization: Bearer *{access_token}* - autorizační token

Odpovědi

- 200 OK - přístup povolen
- 404 Not found - přístup zamítnut
- 401 Unauthorized - neplatný autorizační token

Kapitola 5

Testování aplikace

V této kapitole popisuji způsoby, jakými byla aplikace ACS-M testována. K testování bylo využito jak metodik statické analýzy, jež jsou zaměřeny na kvalitu kódu, tak i automatické testování pomocí jednotkových testů, které je zaměřeno na dílčí části aplikace.

5.1 Kvalita kódu

Tato část testování aplikace je zaměřena na statickou analýzu samotného kódu. Při ní dochází zejména ke kontrole syntaxe kódu, správného typování i dodržování doporučených standardů PHP Standards Recommendations (PSR)[16].

Velká část statické analýzy kódu probíhala přímo při samotném programování aplikace. K vývoji jsem použil program PHPStorm[17], který má v sobě integrovanou kontrolu syntaxe, díky níž jsem byl schopen odhalit některé chyby již v jejich zárodku bez nutnosti kód spustit. V PHPStormu je rovněž možné nastavit automatické formátování kódu, pomocí nějž se dá snadno docílit dodržování PSR.

K pokročilejší statické analýze kódu jsem využil nástroj PHP Static Analysis Tool (PHPStan)[18]. Ten provádí analýzu kódu bez jeho samotného spuštění podobně jako kompilátory v jiných programovacích jazycích. Nástroj PHPStan provádí analýzu na různých úrovních striktnosti a například zkoumá správné typování a přístupnost atributů a metod, dokáže odhalit nepoužívané proměnné a nadbytečné přetypování a mnoho dalšího. Na základě této analýzy jsem byl schopný samotný kód zpřehlednit a případně odstranit implementační nesrovnalosti.

V neposlední řadě byla kontrola kvality kódu prováděna také v rámci jeho vývoje pomocí metodiky GitLab Flow[19]. V ní dochází ke kontrole kódu dalšími osobami v rámci tzv. merge requestů, což přináší další pohled na daný funkční celek. Vzhledem k tomu, že tato metodika dbá na jejich pravidelnou kontrolu, bylo možné implementaci efektivně optimalizovat a refactorovat.

5.2 Jednotkové testy

Jednotkové testy jsou úzce zaměřeny na dílčí části aplikace. Jejich hlavním cílem je testovat jednotlivé třídy a jejich metody. K jednotkovým testům v PHP je určen nástroj PHPUnit[20]. Testování pomocí jednotkových testů je možné automatizovat a provádět opakovaně (pro každý commit do VCS git). Tuto funkcionalitu poskytuje také přímo samotná verzovací služba GitLab, kde je zahrnuta do procesu merge requestu zmíněného výše.

V rámci jednotkových testů jsem testoval entitní třídy, kde bylo hlavním smyslem ověřit správné typování parametrů a návratových hodnot. Dále byly tyto testy využity u tzv. normalizérů, tedy tříd, jejichž funkcí je transformace dat z fakultní IdM služby.

V neposlední řadě se pak jednotkové testy použily u tříd zajišťujících persistenci dat v lokálním režimu. Zde bylo jejich smyslem zejména ověření správné komunikace s databází a v případě vyhledávacích metod také testování navrácených dat. Aby bylo možné tyto testy provést, byla vytvořena testovací data. V Symfony je k tomuto účelu určena komponenta DoctrineFixturesBundle[21].

5.3 Integrační testy

Integrační testy jsou vytvořeny s cílem ověření integrace fakultní služby IdM do aplikace ACS-M a je je možné rovněž automatizovat. S ohledem na to, že v době odevzdání práce není fakultní IdM služba plně doimplementována, jak více objasnuji v závěru, nejsou integrační testy funkční či končí chybou.

5.4 Výsledky automatického testování

Jak jsem již zmínil v 5.2. podkatitole Jednotkové testy, automatického testování je docíleno za použití nástroje PHPUnit. Testy jsou konfigurovány v souboru `phpunit.coverage.xml`. Jejich spuštěním se provede nejenom samotné automatické testování aplikace, ale také analýza pokrytí kódu testy tzv. code coverage. Výstup testů ACS-M se nachází níže.

```
$ php vendor/bin/phpunit -c phpunit.coverage.xml
PHPUnit 5.7.19 by Sebastian Bergmann and contributors.
.........................................S  45 / 142 ( 31%)
SSSSSSSSSS..SSSSSSSS..SSSSSSSSSS..SSSSS.....  90 / 142 ( 63%)
..................................................... 135 / 142 ( 95%)
.......                                            142 / 142 (100%)
Time: 13.18 seconds, Memory: 60.00MB
OK, but incomplete, skipped, or risky tests!
Tests: 142, Assertions: 371, Skipped: 34.
```

Závěr

V rámci této bakalářské práce jsem provedl analýzu současné správy pravidel přístupu do místností. Dále jsem provedl kvalitativní průzkum, na základě kterého vznikly hypotézy, z nichž byly vytvořeny požadavky na funkce a kvality aplikace ACS-M. Tyto požadavky byly vyhodnoceny vůči stávajícímu řešení.

Ze vzniklých hypotéz byli definováni aktéři aplikace a případy užití, ve kterých vystupují. Na základě požadavků byl vytvořen návrh a implementována aplikace ACS-M. Tato aplikace poskytuje rozhraní pro správu pravidel přístupu do místností. Dále umožňuje určit správce těchto pravidel a poskytuje rozhraní pro komunikaci se čtecími zařízeními zámků.

Plné implementace dle požadavků z analytické fáze se podařilo docílit v případě lokálního režimu aplikace. Implementace IdM režim provozu v době odevzdání této práce není kompletní. Důvodem je vyžadované doplnění aplikační logiky fakultní IdM služby, které nebylo v době odevzdání práce dokončeno. Po dohodě s vedoucím práce Ing. Tomášem Kadlecem byl v průběhu realizace práce upřednostněn vývoj lokálního režimu provozu aplikace.

Po doplnění chybějící implementace bude možné aplikaci nasadit v prostředí FIT ČVUT. V rámci prvotního nasazení aplikace bude provedeno beta testování s uživateli, které poskytne zejména prostor pro doladění uživatelského rozhraní.

Současná implementace obsahuje pouze nezbytné části. Aplikace však může být rozšiřována o další funkce. Dle priorit fakulty to jsou:

- z pohledu uživatelského rozhraní možná editace a duplikace již vytvořených pravidel přístupu a oprávnění správce pravidel,
- plná integrace se systémem K4, jež by mohla umožnit libovolné osobě z IdM služby nahlédnout na svá pravidla přístupu do místností,
- optimalizace filtrování v seznamech záznamů, kde by uživatelé mohli sestavovat dotazy přímo pro databázi.

Literatura

[1] Oddělení ICT FIT ČVUT v Praze: *Pravidla a zásady projektů FIT*. březen 2017, [cit. 2017-04-28]. Dostupné z: https://docs.google.com/document/d/1umkLCuvYY1EYMat8jLnYfUj5WyccC3s-30thPvePot4/

[2] Oddělení ICT FIT ČVUT v Praze: ACS-Controller. [software], říjen 2016, [cit. 2017-04-04]. Dostupné z: https://ict.fit.cvut.cz/gitlab/mar/acs_controller

[3] Python Software Foundation: Python. [software], ©2001 - 2017, [cit. 2017-04-04]. Dostupné z: https://www.python.org/

[4] Cook, B.; Messina, C.: OAuth. [software], ©2006 - 2017, [cit. 2017-04-04]. Dostupné z: https://oauth.net/

[5] W3C: Web Content Accessibility Guidelines (WCAG) 2.0. [online], ©2008, [cit. 2017-05-13]. Dostupné z: https://www.w3.org/TR/WCAG20/

[6] Oddělení ICT FIT ČVUT v Praze: Identity-Bundle. [software], ©2017, [cit. 2017-04-08]. Dostupné z: https://docs.google.com/document/d/1H0pOPB0-V8WZbvr8ZjgxQQmD4QPCD-sFEgTDfNoNi4E

[7] SensioLabs: Symfony, High Performance PHP Framework for Web Development. [software], ©2006 - 2017, [cit. 2017-04-28]. Dostupné z: http://symfony.com/

[8] Toran Proxy: Composer, Dependency Manager for PHP. [software], ©2012 - 2016, [cit. 2017-04-28]. Dostupné z: https://getcomposer.org/

[9] Dowling, M.: Guzzle, an extensible PHP HTTP client. [software], ©2015, [cit. 2017-04-28]. Dostupné z: http://guzzlephp.org/

[10] FriendsOfSymfony: FOSRestBundle. [software], ©2017, [cit. 2017-04-28]. Dostupné z: https://github.com/FriendsOfSymfony/FOSRestBundle

[11] KnpLabs: KnpMenuBundle. [software], ©2017, [cit. 2017-04-28]. Dostupné z: https://github.com/KnpLabs/KnpMenuBundle

[12] Doctrine Team: Doctrine Project. [software], ©2006 - 2017, [cit. 2017-04-28]. Dostupné z: http://www.doctrine-project.org/

[13] Ashkenas, J.: underscore. [software], ©2009 - 2017, [cit. 2017-04-28]. Dostupné z: https://github.com/davidgtonge/underscore-query

[14] Tonge, D.: underscore-query. [software], ©2013, [cit. 2017-04-28]. Dostupné z: https://github.com/davidgtonge/underscore-query

[15] Potencier, F.: Serializer component schema. ©2006 - 2017, [cit. 2017-05-13]. Dostupné z: http://symfony.com/doc/current/_images/serializer_workflow.png

[16] PHP Framework Interop Group: *PHP Standards Recommendations*. ©2016, [cit. 2017-05-12]. Dostupné z: http://www.php-fig.org/psr/

[17] JetBrains s.r.o: PHPStorm. [software], ©2000 - 2017, [cit. 2017-05-12]. Dostupné z: https://www.jetbrains.com/phpstorm/

[18] Mirtes, O.: PHP Static Analysis Tool. [software], ©2016 - 2017, [cit. 2017-05-12]. Dostupné z: https://github.com/phpstan/phpstan

[19] Sijbrandij, S.: GitLab Flow. [online], září 2014, [cit. 2017-05-12]. Dostupné z: https://about.gitlab.com/2014/09/29/gitlab-flow/

[20] Bergmann, S.: PHPUnit. [software], ©2001 - 2017, [cit. 2017-05-12]. Dostupné z: https://phpunit.de/

[21] Potencier F. a Doctrine Project: DoctrineFixturesBundle. [software], ©2011, [cit. 2017-05-13]. Dostupné z: https://github.com/doctrine/DoctrineFixturesBundle

[22] Twitter a další: Bower, A package manager for the web. [software], ©2012, [cit. 2017-05-13]. Dostupné z: https://bower.io/

Příloha A

Seznam použitých zkratek

Ace Access control entry

ACS-C ACS-Controller

ACS-M ACS-Manager

API Application Programming Interface

HTTP Hypertext Transfer Protocol

IdM Identity Management

LDAP Lightweight Directory Access Protocol

ORM Object-relational mapping

PHP PHP: Hypertext Preprocessor

PSR PHP Standards Recommendations

REST Representational state transfer

Příloha **B**

Obsah přiloženého CD

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

Příloha C

Instalační příručka

Tato příručka popisuje způsob nasazení aplikace na operačním systému Linux.

Požadavky:

- PHP >= 7.0
- MySQL >= 5.7
- Node.js >= 4.2.6

Instalace:

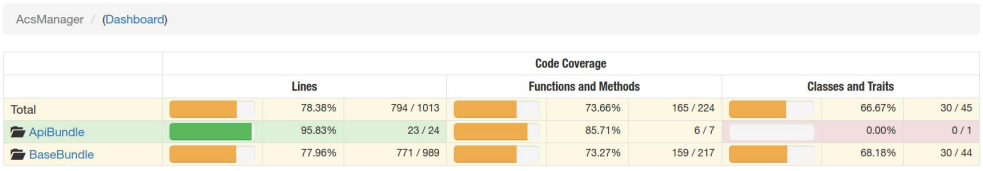
1. Zkopírujte obsah složky `impl` do počítače a přejděte do této složky.
2. Spusťte skript `getcomposer.sh` pro stažení nástroje Composer[8].
3. Proveďte příkaz `php composer.phar install` pro instalaci závislostí a nastavte konfigurační parametry aplikace. (Parametry jsou uloženy v souboru `app/config/parameters.yml` a je možné je měnit v textovém editoru.)
4. Nainstalujte nástroj Bower[22] pomocí příkazu `npm install -g bower`.
5. Nainstalujte front-endové závislosti pomocí příkazu `bower install`.
6. Zkompilujte tyto závislosti příkazem `php bin/console assetic:dump`.
7. (Lokální režim) Příkazem `php bin/console doctrine:database:create` vytvořte databázi.
8. (Lokální režim) Příkazem `php bin/console doctrine:schema:create` vytvořte schéma databáze.

9. (Lokální režim) Příkazem `php bin/console identity:init-roles` a následně `php bin/console acs-manager:init-roles` incializujte uživatelské role v aplikaci.
10. (Lokální režim) Příkazem `php bin/console identity:create-person` vytvořte administrátora aplikace a nastavte mu příslušné role z nabídky.
11. Pomocí příkazu `php bin/console cache:clear` vyčistěte mezipaměť.
12. Pomocí příkazu `php bin/console server:run` spusťte aplikaci.

Příloha D

Analýza pokrytí kódu testy

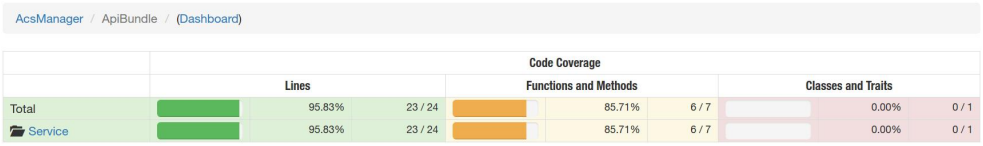
V této příloze je zobrazen výstup z tzv. code coverage jednotkového testování.



AcsManager / (Dashboard)

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	78.38%	794 / 1013	73.66%	165 / 224	66.67%	30 / 45
ApiBundle	95.83%	23 / 24	85.71%	6 / 7	0.00%	0 / 1
BaseBundle	77.96%	771 / 989	73.27%	159 / 217	68.18%	30 / 44

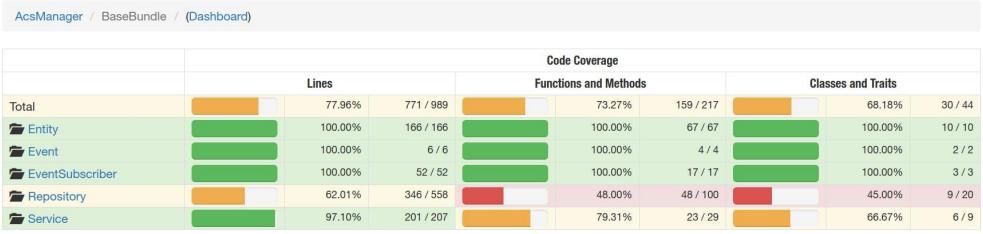
Obrázek D.1: Analýza pokrytí kódu celé ACS-M



AcsManager / ApiBundle / (Dashboard)

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	95.83%	23 / 24	85.71%	6 / 7	0.00%	0 / 1
Service	95.83%	23 / 24	85.71%	6 / 7	0.00%	0 / 1

Obrázek D.2: Analýza pokrytí kódu ApiBundle



AcsManager / BaseBundle / (Dashboard)

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	77.96%	771 / 989	73.27%	159 / 217	68.18%	30 / 44
Entity	100.00%	166 / 166	100.00%	67 / 67	100.00%	10 / 10
Event	100.00%	6 / 6	100.00%	4 / 4	100.00%	2 / 2
EventSubscriber	100.00%	52 / 52	100.00%	17 / 17	100.00%	3 / 3
Repository	62.01%	346 / 558	48.00%	48 / 100	45.00%	9 / 20
Service	97.10%	201 / 207	79.31%	23 / 29	66.67%	6 / 9

Obrázek D.3: Analýza pokrytí kódu BaseBundle

Příloha E

Pravidla a zásady projektů FIT

Práce se řídí dokumentem Pravidla a zásady projektů FIT[1], jenž popisuje preferovaný způsob řešení projektů v rámci FIT. Dokument je zahrnut níže v této příloze.

FIT ČVUT, 2017-02-21

Pravidla a zásady projektů FIT

Tento dokument popisuje preferovaný způsob řešení projektů FIT. Pokud některá část tohoto dokumentu není vůči konkrétnímu projektu efektivní nebo na ní není dostatek prostředků, je jí snížena úroveň nezbytnosti[1] nebo je od ní zcela odstoupeno. O takovém rozhodnutí musí existovat záznam.

Tento dokument je nedílnou součástí projektové dokumentace jako příloha ke kapitole Požadavky na kvalitu. Veškeré oblasti tohoto dokumentu, které uvedená kapitola nezmíní, se předpokládají a požadují. Výchozí úrovní nezbytnosti požadavků (není-li uvedeno jinak) je [MUST].

Obsah dokumentu

Dokumenty

Ke každému projektu vzniká sada samostatných dokumentů v čele s harmonogramem. Dokumenty jsou (až na výjimky) psané česky. Výchozím dokumentem je Harmonogram, který podléhá schvalování. Není-li uvedeno jinak, o projektu rozhoduje vedení fakulty (grémium děkana).

[1] Key words for use in RFCs to Indicate Requirement Levels

Harmonogram

Stěžejním rozcestníkem projektu je iterativní harmonogram. V každém běhu se předpokládá realizace jen takových funkcí a vlastností aplikace, které náleží do daného běhu. Každý běh (iterace) obsahuje všechny uvedené fáze harmonogramu. Pro každou fázi je stanoven termín předpokládaného dokončení. Každá fáze podléhá schvalování vedením písemnou formou. Obsah harmonogramu se s postupem času upřesňuje.

1. **Analytická fáze**
    - Průzkum existujících řešení (dále jen SOTA[2]).
    - Vytvoření hypotéz jako kvalitativním průzkum, typicky formou rozhovorů.
        i. Uživatelské skupiny
        ii. Potřeby uživatelů
    - Ověření hypotéz jako kvantitativní průzkum, typicky dotazníkem v souladu s explicitně zmíněnou metodikou[3].
    - Sestavení požadavků na kvality (vycházející z tohoto dokumentu).
    - Sestavení požadavků na funkce (prioritní seznam rozdělený dle úrovní nezbytnosti).
    - Posouzení SOTA vůči požadavkům na funkce.
2. **Návrhová fáze**
    - Procesy (back-end, front-end)
    - Prototypování (paralelní).
    - Diagramy (procesní, aktivit).
    - Scénáře průchodu, user-stories, případové studie
        i. pro všechny vznikající funkce
        ii. testování (např. formou storyboarding, inspekce).
    - Hi-fi prototypy (grafický návrh, …).
3. **Implementační fáze**
    - Koncepty řešení dílčích požadavků (funkční / nefunkční).
    - Technická specifikace (API, parametry, manuál) jako příloha k projektové dokumentaci nebo přímo u projektu v repozitáři.
    - Testování (inspekce, heuristika).
4. **Vyhodnocení**
    - Testování, logování.
    - Vyhodnocení (feedback, statistiky, logy).

Ostatní dokumenty projektu

V rámci vývoje vznikají další typy dokumentů s níže uvedenými náležitostmi pro různé skupiny čtenářů. Obsah každého dokumentu je cílený na příslušnou skupinu uživatelů a zohledňuje jejich schopnosti a možnosti.

- **Projektová dokumentace**
    - je množina samostatných dokumentů vznikajících pro jednotlivé iterace agilního vývoje aplikace[4],
    - má strukturu podle fází harmonogramu,
    - odkazuje na související legislativní úpravu problémové domény aplikace,
    - je určená pro zadavatele, návrháře a vývojáře projektu a případně pro uživatele.

[2] State of the Art
[3] 6 kroků, jak vytvořit dotazník
[4] Agile software development

- **Instalační a provozní příručka**
  - obsahuje kompletní postup pro sestavení (build) a nasazení (deployment) aplikace a nových verzí,
  - popisuje dostupná prostředí (staging/produkční verze) v návaznosti na kap. Údržba a rozvoj,
  - popisuje provozuschopnost v případě nedostupnosti souvisejících služeb,
  - popisuje proces obnovení provozu v případě výpadku.
- **Uživatelská dokumentace**
  - je průběžně udržovaný samostatný dokument,
  - obsahuje informaci, k čemu a komu aplikace souží,
  - je určená koncovým uživatelům frontendové aplikace[5], resp. aplikačního rozhraní (RESTful API)[6],
  - je dostupná z webu FIT (stačí odkazem)[7],
  - zahrnuje CHANGELOG (viz kap. Verzování).
- **Dokumentace vnitřního API**
  - je sada dokumentů generovaná z kódu průběžně udržovaná společně s kódem aplikace,
  - je psaná anglicky v příslušné standardizované syntaxi[8], přičemž dokumentace veřejných entit zahrnuje minimálně:
    - souhrnný popis dokumentované entity (funkce, třídy, metody, proměnné, …),
    - souhrnný popis parametrů (funkce/metody) nebo typových proměnných (generické typy),
    - popis vyhazovaných výjimek (které výjimky a kdy vznikají),
    - popis návratové hodnoty (a její význam).

Architektura a integrace do infrastruktury

Projekt je webovou aplikací, která efektivně využívá existující technologie a služby FIT. Aplikace je členěná na nezávislé části, které je možné vyměnit a je provozuschopná i v případě výpadků souvisejících služeb.

- Projekt je webovou aplikací s
  - uživatelským rozhraním (UI) pro webový prohlížeč, nebo
  - RESTful API s upřesněním standardu vč. formátu[9].
- Architektura aplikace
  - striktně odděluje frontend a backend,
  - správa uživatelů (user-management) je zajištěna fakultním IDM,
  - využívá maximum dostupných služeb (např. notifikace),
  - [SHOULD] podporuje použití pro více fakult na jediné instanci.
- Aplikace
  - je součástí katalogu služeb FIT[10] od počátku práce na projektu (stav „připravuje se“),
  - využívá mezipaměť pro urychlení obsluhy požadavků,

[5] 10 Examples of Great End User Documentation
[6] Doporučené nástroje pro dokumentaci RESTful API: RAML, Swagger / OpenAPI
[7] Návod ke psaní dokumentace ICT FIT
[8] Např. JavaDoc nebo DoxyGen
[9] Doporučujeme vycházet ze standardu JSON API
[10] Katalog služeb ICT FIT

    - je provozuschopná i v případě nedostupnosti (zpomalení) souvisejících služeb[11].
- [MAY] Aplikace je implementovaná na platformě/jazyku:
    - Ruby,
    - JavaScript, resp. izomorfní JavaScript[12],
    - Groovy/Java na Spring Frameworku,
    - Python,
    - PHP na frameworku Symfony.

Webová přístupnost

Aplikace je přístupná pro uživatele bez ohledu na jejich omezení a zařízení, kterým k aplikaci přistupují.

- Aplikace respektuje požadavky WCAG 2.0 AA[13], zejména
    - sémantické značkování výstupu HTML,
    - jednoznačné perzistentní URL jednotlivých stránek[14],
    - podpora tisku,
    - *progressive enhancement*[15].
- Výstup aplikace (HTML) je v souladu s principem *mobile-first*[16], *media-first*[17], zejména
    - přizpůsobivé uživatelské rozhraní[18],
    - použitelnost ovládacích prvků pro manipulaci prsty,
    - nenáročnost s ohledem na výkon CPU a spotřebu baterie,
    - minimalizace přenesených dat.
- [MAY] Aplikace je odolná vůči výpadkům připojení a funkčnost bez připojení k Internetu.[19]
- [MAY] Aplikace podporuje *Web App Manifest*[20] a integraci do operačního systému[21] zahrnující
    - podporu push notifikací[22],
    - synchronizaci na pozadí přes *Service Workers*[23].

[11] Např. bez datového, resp. autentifikačního, zdroje, informace z mezipaměti, resp. zobrazí jen veřejné informace (s příslušným upozorněním).
[12] Viz Isomorphic JavaScript.
[13] Web Content Accessibility Guidelines (WCAG) 2.0
[14] Viz Cool URIs a Why JavaScript web applications should embrace traditional URLs.
[15] Poskytnout klientovi úplnou funkcionalitu i v případě, že nepodporuje dynamické technologie; viz článek na Gov.UK.
[16] Viz kniha Mobile First (Luke Wroblewski).
[17] Společná definice zobrazení od sémantického obsahu pro čtečky a textové interprety, přes tisk a malé obrazovky až po velké obrazovky.
[18] Viz Responsive Web Design.
[19] Tzv. Offline-First.
[20] Viz Web App Manifest.
[21] Viz články Progressive Web Apps: Escaping Tabs Without Losing Our Soul a Getting started with Progressive Web Apps.
[22] Viz Push Notifications on the Open Web.
[23] Viz Introduction to Service Worker.

Kód aplikace

Veškerý kód je psaný kompletně v angličtině s prioritou udržitelnosti a čitelnosti. Vývoj kódu přehledně odděluje provozní větev od vývojových. Před nasazením prochází každý nově vzniklý kód kontrolou kvality na několika úrovních.

- Aplikace respektuje *best practices* pro psaní udržitelného a čitelného kódu[24]; zejména
  - logické členění kódu do modulů podle funkcionality,
  - specifikace konvencí používaných technologií (např. CSS[25], JavaScript[26], Java[27]),
  - dodržování stylu autora při editaci cizího kódu,
  - minimalizace importů[28] (import, include, using, atd.),
  - používání existujících knihoven[29], kdykoli je to efektivní a smysluplné,
  - používání návrhových vzorů[30],
  - komentování potenciálně nejasných částí.
- Veškerý kód je psaný
  - v UTF-8 s unixovým koncem řádek (řídící znak LF / 0x0A),
  - anglicky (názvy funkcí a proměnných, komentáře, systémová a jiná hlášení).
- Veškeré výstupy (texty pro uživatele) podporují lokalizaci a internacionalizaci.
- Pro aplikaci existuje kompletní česká lokalizace.

Verzování

Vývoj kódu je organizovaný s přehledným oddělením provozní a vývojové větve. Umožňuje operativní opravy kritických chyb (hotfix) a nezávislý vývoj nových funkcí. Podporuje bezpečný model nasazování nových verzí[31] pro účely testování a ladění (akceptační testy).

- Kód aplikace je vyvíjen na revizním systému Git[32] využívající
  - repozitář na službě GitLab provozované fakultou[33] nebo oddělením ICT[34],
  - standardní branching model Git Flow[35]
    (nástroje OMGF[36] nebo Git-Flow Cheatsheet[37]) a
  - sémantické verzování[38].
- Používání revizního systému se řídí pravidly *commitování*, zejména
  - *commit* každé dílčí změny funkcionality,
  - zachování funkcionality celku přes jednotlivé *commity*,

[24] Viz Best Practices a kniha The Pragmatic Programmer.
[25] Konkrétně konvenci SUIT CSS.
[26] JavaScript Quality Guidelines and Recommendations
[27] Code Conventions for the Java Programming Language
[28] Např. neimportovat celý balíček, když z něj bude použita jen malá část.
[29] Pod svobodnými nebo open-source softwarovými licencemi a respektovat podmínky těchto licencí.
[30] Gang of Four Design Patterns či Design Patterns na Wiki
[31] Deployment environment
[32] Jak na Git
[33] GitLab FIT ČVUT
[34] GitLab ICT
[35] Git Flow
[36] OMGF
[37] Git-Flow Cheatsheet
[38] Semantic Versioning 2.0.0

    - používání rozkazovacího tvaru v přítomném čase[39].
- Součástí vývoje je udržování aktuálního souboru CHANGELOG[40] dostupného z webu na úrovni
    - nových funkcí (či inovací) vždy při jejich začlenění do vývojové větve,
    - nových MINOR verzí vždy při začlenění vývojové větve do provozní.
- Na společných (sdílených) větvích není povoleno přepisování historie.
- Veškeré texty verzování jsou anglicky.

Kontrola kvality (QA)

Veškerý kód se před nasazením patřičně kontroluje na úrovni automatizovaných nástrojů a dílčích (jednotkových a dalších) testů. Součástí kontroly kódu je (jednoduchý) schvalovací proces. Alternativně se kód vyhodnocuje, zda splňuje stanovené kvalitativní metriky.

- Vývoj kódu se opírá o kontrolní nástroje jako zejména
    - příslušný *linter*[41].
- Veškeré nasazování změn kódu (merge) procházejí kontrolním procesem[42] s následujícími pravidly.
    - Veškeré merge jsou prováděny formou požadavků na začlenění[43] (dále PR).
    - Veškeré PR (bez ohledu na svou podstatu a závažnost) budou potvrzované minimálně druhým členem týmu – programátorem, alternativně nadřízeným.
    - PR kritického požadavku si může jeho řešitel sám akceptovat. O takovém úkonu neprodleně vyrozumí členy týmu. Povinnost potvrzení podle předchozího bodu zůstává, však může být učiněno dodatečně (bez zbytečného prodlení).
    - Součástí kontrolního procesu nasazování je continuous integration[44] na GitLabu[45].
- Kód obsahuje automatické testy, mezi které patří zejména
    - jednotkové testy,
    - integrační testy (API, resp. automatizované průchody).
- Klíčová funkcionalita je ošetřena testy, které jsou specifikované v dokumentaci s maximální mírou automatizace.
- Vývoj kódu je řízený testy[46] [SHOULD].
- Minimální požadované hodnoty metrik[47] pomocí fakultní služby Sonar[48] jsou stanoveny následujícím způsobem: [SHOULD]
    - Method Total Length (< 30 lines)
    - Class Total Length (< 300 lines)
    - Unit Tests Line coverage (> 70 %)
    - Unit Tests Branch coverage (> 70 %)
    - Density of duplicated lines (< 5 %)
    - Lack of cohesion of methods (< 3)
    - Average complexity by method (< 5)

[39] How to Write a Git Commit Message
[40] Keep a Changelog
[41] Platí zejm. pro dynamické a značkovací jazyky; viz např. doporučení pro JavaScript a CSSLint.
[42] Best Practices for Code Review a Code reviews v praxi
[43] Code Review Via GitLab Merge Requests
[44] Continuous integration
[45] GitLab Continuous Integration
[46] Test-Driven Development
[47] Sonar Metric Definitions
[48] Sonar FIT ČVUT

    - Rules compliance index (žádné závady úrovně „blocker“ ani „critical“)

Provoz, údržba a rozvoj aplikace a podpora uživatelů

Veškerá (nově vznikající) funkcionalita je uživatelům dostupná přehledně a jednoduše. Aplikace (nová verze) se nasazuje do provozu po splnění akceptačních testů. Součástí údržby a dlouhodobého rozvoje aplikace je sběr informací o používání a jejich pravidelné vyhodnocování následované patřičným zapracováním do aplikace.

- Vzhled uživatelského rozhraní (UI) aplikace je moderní, přehledný, vzdušný a tvořený obsahem.[49]
- Sada akceptačních testů je specifikovaná pro účely testování všech dostupných a nově vznikajících funkcí.
- Sběr dat se provádí na základě
    - zpětné vazby uživatelů prostřednictvím
        - funkce issue tracking v rámci GitLabu a
        - e-mailu na helpdesk,
    - logování[50] a integrace s monitorovacími službami na úrovni
        - chyb (fatal, warning),
        - informačních zpráv o používání[51] (používanost funkcí, doby trvání, přístupy) a
        - systémových zpráv a dalších výstupů.
- Proces vyhodnocování dat zahrnuje
    - podporu uživatelů a
    - opravy chyb včetně klasifikace jejich závažnosti.
- [SHOULD] Proces rozvoje od návrhu po realizaci za účelem
    - vylepšování stávajících funkcí (optimalizace chodu a procesů),
    - přidávání nových funkcí.
- [SHOULD] Proces testování nových verzí aplikace zahrnuje
    - provoz nezávislé (beta) verze,
    - podporu AB testování,
    - provádění inspekcí a heuristik,
    - pozorování.

Bezpečnost a ochrana osobních údajů

Aplikace je standardně zabezpečená; zejména nepracuje s hesly uživatelů a veškerá komunikace probíhá přes šifrovaný protokol. Aplikace také respektuje nařízení rektora o ochraně osobních údajů.

- Aplikace respektuje principy bezpečných webových aplikací[52], jmenovitě
    - veškerá komunikace (S2S, S2C) probíhá přes HTTPS,
    - jako API poskytuje různé úrovně oprávnění pomocí *scopes*[53],

[49] UXMyths: You don’t need the content to design a website.
[50] Logging Best Practices
[51] Google Analytics s využitím událostí (events)
[52] Viz principy OWASP.
[53] Securing Access with OAuth2: How to deal with OAuth Scopes

    - nepracuje s hesly uživatelů; autentizace, resp. autorizace, probíhá přes Shibboleth (není-li potřeba autorizace), resp. autorizační server FIT (protokol OAuth 2.0)[54].
- Aplikace respektuje nařízení rektora o ochraně osobních údajů[55], zejména dokumentace definuje,
    - jaké informace jsou citlivé/osobní,
    - jaká data jsou dostupná kterým uživatelům v závislosti na autorizaci uživatele (např. anonymní uživatel, přihlášený uživatel, student, vyučující, administrátor),
    - které citlivé/osobní informace jsou přístupné v rozporu s nařízením.

Další koncepty webových aplikací (W2.0)

Aplikace explicitně zohledňuje možnosti využití níže uvedených webových konceptů a případně dalších. Vzhledem k omezení jednotlivých projektů mohou být využití konceptů pouze součástí projektové dokumentace – byť jen jako potenciální rozšíření funkcionality s uvedenými přínosy a konkrétními příklady.

- Dokumentace (např. v příloze) popisuje možnosti a úroveň nezbytnosti využití všech následujících konceptů:
    - RSS,
    - personalizace,
    - customizace,
    - folksonomie (tagování),
    - social networking,
    - real-time web[56],
    - crowdsourcing[57],
    - kolaborace,
    - průvodce (wizardy),
    - konfigurátory (rozšířených dotazů vyhledávání, parametrů služby),
    - gamifikace[58],
    - mikrodata[59].

[54] Viz Autorizační server FIT (OAuth 2.0).
[55] Příkaz rektora č. 5/2015 Ochrana osobních údajů na ČVUT v Praze
[56] Real-time web (Wiki)
[57] Crowdsourcing (Wiki)
[58] Gamification (Wiki)
[59] Microdata (Wiki)