Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Využití mobilních zařízení pro správu dokumentů

Plzeň 2016 Martin Bláha

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 20. června 2016

Martin Bláha

Poděkování

Chtěl bych poděkovat Ing. Davidovi Wegschmiedovi za vedení mé diplomové práce a odborný dohled. Děkuji také Ing. Martinovi Zímovi, Ph.D. za cenné rady.

Abstrakt

Tato diplomová práce nejdříve seznamuje čtenáře se systém pro správu dokumentů (DMS) a jejich funkcionalitu. Poté popisuje DMS systémy, které se využívají ve firmě CCA Group a.s., nebo o nich firma uvažuje. Po seznámení s DMS systémy se popisují existující aplikace, které slouží pro správu DMS systémů. V další části diplomové práce je čtenář seznámen s mobilním operačním systémem a s typy mobilních operačních systémů používajících se v současné době. Tyto mobilní operační systémy jsou v diplomové práci popsány, jsou zde uvedeny i základní informace pro vývoj aplikací a jejich distribuci. Po seznámení se s mobilními operačními systémy je provedena analýza a návrh mobilní aplikace a popis její implementace. Poté je popsána implementace Adaptéru. Ta slouží pro ověření funkčnosti mobilní aplikace. V poslední části diplomové práce je ověřena funkčnost a použitelnost implementované mobilní aplikace.

Abstract

This diploma thesis introduces the reader to Document management system (DMS) and their functionality. It describes DMS which are utilized in the CCA Group a.s. company, or those which are being considered for use there. After the introduction of those systems, a description of various applications which manage DMS is given. The next section deals with mobile operating systems used nowadays. A description of these systems is included, as well as elementary information pertaining to mobile applications development and distribution. An analysis and design of a mobile application is given, followed by a description of implementation of both the mobile application and the Adapter, which serves as a functionality test. The last section of the thesis deals with verification of usability of the implemented mobile application.

Obsah

1 Úvod

V dnešní době, kdy se extrémně zvyšuje počet dokumentů, které firma musí zpracovat, je důležité, aby firmy tyto dokumenty dokázaly efektivně zpracovávat. Tyto dokumenty mohou do firmy přijít z více zdrojů, jedná se např. o klasickou poštu, datovou schránku nebo e-mail. Firma všechny tyto dokumenty musí zpracovat a dokázat je vyřídit v co nejkratší možné době dle potřeby. Všechny tyto dokumenty je potřeba evidovat tak, aby se žádný dokument neztratil při koloběhu ve firmě. Některé dokumenty je potřeba také archivovat pro pozdější použití. Velmi důležitou roli hraje také bezpečnost, kdy dokument může číst jen oprávněná osoba.

Pro tyto účely si firmy pořizují systém pro správu dokumentů, neboli DMS. Tento systém se stará o dokumenty a jejich koloběh ve firmě. DMS systémy se ve většině případů ovládají pomocí webového rozhraní. S nástupem chytrých mobilů je ale potřeba také umožnit uživateli ovládat tyto systémy pomocí mobilního zařízení.

Z tohoto důvodu přišel požadavek na analýzu existujících mobilních aplikací pro správu DMS systému a vytvoření aplikace, která by umožňovala provádět základní funkce v DMS systému pomocí mobilního zařízení.

2 Systém pro správu dokumentů

Systém pro správu dokumentů [1, 2, 3, 4] je počítačový systém, který slouží k práci s elektronickými dokumenty. Tento systém se nezaměřuje na úpravu dokumentů, ale na jejich řízení. Systém se stará o vstup dokumentů, jeho správu, řízení a zajišťuje nad dokumentem bezpečnost.

Jako dokumenty můžeme brát texty, které byly do elektronické podoby převedeny pomocí skenování, ale i obrázky, zvukové záznamy, videa atd.

V minulých letech se DMS systémy dělaly pomocí intranetů. Výhodou těchto řešení bylo, že mimo firemní síť se k dokumentům nikdo nedostal a nemohlo dojít k úniků tajných dokumentů. Problém byl ale v tom, že nebylo možné s dokumenty pracovat, pokud uživatel nebyl přihlášen ve firemní síti. Z tohoto důvodu začaly vznikat DMS systémy, do kterých bylo možné se přihlásit z internetu. Vzhledem k tomu, že bylo možné se přihlásit z internetu, začalo důležitou roli hrát zabezpečení. Díky tomu, že dokumenty můžeme vidět z internetu, můžeme těmto dokumentům nastavit práva tak, aby veřejnost tyto dokumenty mohla případně nemohla číst.

I v dnešní době se můžeme setkat s DMS systémy, které jsou založené na intranetu. Tyto DMS systémy mohou složit pro nejdůležitější dokumenty, které se nesmějí dostat mimo firemní síť.

2.1 Základní funkce DMS systémů

V této části si popíšeme základní funkce, které by měl každý systém pro správu dokumentů splňovat. Každý systém může tyto funkce provádět rozdílně a je jen na zákazníkovi, který způsob provádění mu nejvíce vyhovuje.

2.1.1 Vyhledávání

Z důvodu velkého množství dokumentů uložených v DMS systému, musí systém umožnit vyhledávání. Nalezení dokumentů v systému by mělo fungovat podle více kritérií. Vyhledat dokument by mělo jít přes fulltextové vyhledávání, kde se prochází i obsah dokumentů, nebo podle metadat, které dokument má.

2.1.2 Přístup do systému

Důležitou funkcí, kterou každý systém musí obsahovat je přístup do systému. Přístup do systému se řeší většinou pomocí DMS klienta, který je instalován lokálně nebo pomocí webového rozhraní. Z nástupem moderních webových technologií se nyní do systému přistupuje převážně pomocí webového rozhraní. Některé systémy mohou mít také veřejné API, které může uživatel využít pro přístup.

2.1.3 Zabezpečení

Zabezpečení je jedno z nedůležitějších součástí DMS systémů a při výběru systému by mělo hrát důležitou roli. Jak vysoká úroveň bezpečnosti je závislá na tom, o jaký druh firmy se jedná a jaké dokumenty bude systém spravovat. Základní zabezpečení, které by měly všechny systémy podporovat je šifrované spojení mezi klientem a serverem, kde je nainstalován DMS systém.

2.1.4 Práva dokumentů

Práva dokumentů jsou silně spjatá se zabezpečím DMS systému. Systém musí umožňovat nastavit dokumentu práva tak, aby osoba, která nemá přidělená práva k dané činnosti, nemohla danou činnost provést. Nejzákladnější práva nad dokumentem, který by měl systém obsahovat, jsou:

- **čtení** - uživatel si daný dokument může otevřít a přečíst,
- **zápis** - uživatel může dokumentu změnit metadata,
- **vymazání** - uživatel má právo daný dokument vymazat.

V moderních systémech ale můžeme nalézt celou škálu práv, které umožňují přidávat práva například složkám, kde jsou dokumenty uloženy nebo skupinám uživatelů. Některé systémy dokonce umožňují přidávat práva podle metadat dokumentu.

2.1.5 Vstup dokumentů

DMS systém umožní vkládat dokumenty ručně pomocí formuláře nebo ze vstupů, jako je například e-mail, datová schránka atd. Systém se snaží co nejlépe pochopit obsah např. pomocí OCR a daný dokument správně zařadit do systému. Při vstupu dokumentu do systému může uživatel také doplnit metadata o dokumentu, a to např. název, autor, popisek atd. Systém dále doplní tato metadata např. o velikosti dokumentu, typu atd.

2.1.6 Verzování dokumentů

DMS systém eviduje všechny změny, které se s dokumentem provedly, a při každé změně je nastavená nová verze dokumentu. DMS systém si při změně dokumentu ukládá uživatele, který změnu provedl, čas, kdy byla změna provedena, a také údaje o změně. Protože DMS systém má informace o všech změnách nad dokumentem, můžeme díky tomu zobrazit starší verzi dokumentu nebo se ke starší verzi vrátit, kdyby se v dokumentu provedly nežádoucí změny, které je potřeba vrátit.

2.1.7 Archivace

Některé dokumenty, které prošly systémem, je potřeba archivovat. Těmito dokumenty mohou být třeba účetní doklady, které je podle legislativy nutné archivovat. Z tohoto důvodu je vhodné, aby DMS systém umožnil archivaci dokumentů. Součástí archivace by mělo být také rychlé vyhledávání archivovaných dokumentů.

2.1.8 Workflow

Moderní DMS systémy obsahují workflow, které říká, jak bude dokument v systému zpracováván. Jde o proces oběhu dokumentu v systému. Workflow vytváří úlohy pro uživatele, kteří danou úlohu musí vyřídit. Systém poskytuje nástroje, kde je vidět v jakém stavu se kterýkoliv dokument nachází. Systémy mohou také obsahovat funkce, které umožní uživateli vytvořit vlastní workflow v systému.

2.1.9 Zálohování

Zálohování je důležitou součástí DMS systému, i když ne každý sytém tuhle funkci podporuje. Při používaní systému mohou nastat nečekané situace, které mohou poškodit data nebo je vymazat. Z tohoto důvodu je potřeba, aby zvolený systém podporoval zálohování. Zálohovaní dat by se mělo provádět pravidelně a data by se měla ukládat na bezpečné místo, kam se může dostat jen zodpovědná osoba. Pokud se vyskytne problém a je potřeba provést obnovu, systém by měl podporovat rychlé nasazení zálohy.

2.2 DMS ve firmě CCA Group a.s.

CCA Group a.s. [5] je předním poskytovatelem informačních systémů v České republice se zaměřením na obory systémová integrace, správa dokumentů a business intelligence. Na trhu IT služeb působí od roku 1991 a je zavedenou firmou s více jak 80 zaměstnanci v Praze, Plzni a Brně. CCA je původem zkratkou slov Computer, Consultancy and Applications, které do současnosti popisují hlavní pilíře služeb, které společnost poskytuje.

Firma CCA Group a.s. využívá pro svojí činnost tři DMS systémy: CleverDOC, SharePoint a nově i Alfresco. Tyto tři systémy si nyní představíme.

2.2.1 CleverDOC

CleverDOC [6] je systém, který je postavený nad platformě SharePoint a slouží pro veškerou správu a oběh elektronických firemních dokumentů. CleverDOC byl vytvořen firmou CCA Group a.s. jako náhrada za dříve používaný systém EasyDOC, který byl také vytvořen firmou CCA Group a.s.

Systém obsahuje standardní firemní procesy pro nejčastější dokumenty. Díky této vlastnosti je možné rychlé nasazení aplikace u zákazníka. Pokud ale zákazníkovi nestačí standardní definované procesy, může si tyto procesy upravit nebo vytvořit nové podle potřeby bez toho, aniž by se něco muselo programovat. Systém také obsahuje nástroje pro digitalizaci dokumentů a jejich uložení nebo napojení na ostatní systémy jako je SAP.

2.2.2 SharePoint

SharePoint [7, 8, 9] je webová aplikační platforma vytvořená společností Microsoft Corporation v roce 2001. Jde o sadu aplikací, mezi kterými je i aplikace pro správu dokumentů.

Obrázek 2.1: Logo SharePointu

SharePoint je vydán ve čtyřech verzí.

- **Microsoft SharePoint Foundation**
  Jde o bezplatnou verzi SharePointu, kterou lze spustit pouze na softwaru, jež má proprietární licenci. Jde hlavně o software Microsoft Windows Server.
- **Microsoft SharePoint Standard**
  Jde o placenou verzi, která nabízí základní funkce. Jedná se o funkce Weby, Komunity, Obsah a Hledání. Tato verze je licencovaná pomocí CAL [10].
- **Microsoft SharePoint Enterprise**
  Tato verze nabízí plnou funkčnost SharePointu. Oproti verzi Standard obsahuje navíc funkce Obchodní řešení a Business Intelligence pro každého. Na tuhle verzi je možné přejít z verze Standard pouhým zadáním aktivačního klíče. Stejně jako ve verzi Standard je zde použita licence CAL.
- **SharePoint Online**
  SharePoint Online je verze, která je vydána jako SaaS. Služba je licencována jednotlivým uživatelům. SharePoint Online je možné koupit samostatně nebo jako součást Office 365.

Tento produkt je pro firmu CCA Group a.s. klíčový, protože nad touto platformou je postaveno jejich řešení CleverDOC.

2.2.3 Alfresco

Alfresco [11, 12] je systém pro správu dokumentů vytvořený firmou Alfresco Software, Inc. v roce 2005. Alfresco funguje na operačním systému Windows nebo na Unix-like operačních systémech.

Obrázek 2.2: Logo Alfresca

Alfresco je vydáno ve třech verzích, které jsou zaměřeny na potřebu uživatelů.

- **Alfresco Community Edition**
  Jde o bezplatnou open source verzi vydanou pod licenci LGPL.
- **Alfresco Enterprise Edition**
  Jde o verzi, která je určena pro komerční použití. Oproti verzi Community Edition obsahuje nástroje pro škálovatelnost systému a obsahuje podporu, která je předem definovaná pomocí SLA mezi poskytovatelem a zákazníkem.
- **Alfresco Cloud Edition**
  Tato verze je vydaná jako SaaS.

Firma CCA Group a.s. uvažuje tento systém využít a postavit nad touto platformou CleverDOC, který by poté nabízela klientům jako levnější variantu k variantě postavené nad SharePointem.

3 Mobilní aplikace pro DMS

V této kapitole provedeme analýzu existujících mobilních aplikací pro správu DMS systému a její funkcionalitu. Na konci kapitoly provedeme vyhodnocení těchto aplikací.

3.1 Alfresco Mobile

Alfresco Mobile [13] je mobilní aplikace pro správu Alfresca, která je vytvořená firmou Alfresco Software, Inc. Tato aplikace umožňuje zobrazit veškerý obsah uložený v Alfrescu, který je místně nebo v cloudu.

Aplikace je bezplatná a je vydaná pro mobilní operační systémy Android a iOS.

Z důvodu, že jde o aplikaci, z které se spravují firemní data, je zde kladen důraz na bezpečnost. Pro data, která byla stažena je možné nastavit čas platnosti, kdy po této době nebude možné data přečíst. Data, která jsou přenášena mezi aplikací a Alfrescem, jsou šifrovaná.

Alfresco Mobile obsahuje integraci s mnoha aplikacemi, které jsou používány v mobilu pro správu obsahu. Mezi těmito aplikacemi je např. Quickoffice HD, Documents To Go, iWorks atd.

Všechny změny provedené v aplikaci se ihned provedou i v Alfescu. Mezi funkce[14], které aplikace umožňuje, patří např:

- synchronizace veškerých dokumentů a složek pro prohlížení v offline režimu,
- vyhledávání dokumentů, složek a lidí včetně pokročilého vyhledávaní pomocí metadat,
- vytváření složek a nahrávání dokumentů.

3.2 Alfresco Activity

Alfresco Activity [15, 16] je stejně jako Alfresco Mobile mobilní aplikace pro správu Alfresca, vytvořená firmou Alfresco Software, Inc. Tato aplikace je určená pro práci s workflow.

Díky této aplikaci můžeme prohlížet své úkoly, vyplňovat formuláře a spolupracovat s ostatními uživateli v systému. V aplikaci můžeme vytvářet nové procesy, připojovat soubory nebo schvalovat úkoly.

Mobilní aplikace je dostupná pro operační systém Android a iOS s tím, že pro oba systémy je aplikace stále vyvíjená.

3.3 SharePlus

SharePlus [17, 18] je mobilní aplikace, která je vytvořena firmou Infragistics. Aplikace slouží pro správu SharePointu.

Mobilní aplikace je k dostání zdarma s funkčním omezením pro operační systémy Android a iOS. Pro plně funkční aplikaci je potřeba si koupit jednu z možných licencí. Nejlevnější licence je Invidual a stojí 17,99 dolarů ročně.

SharePlus podporuje SharePoint od verze 2007 a SharePoint Online (Office 365).

Mezi funkce, které aplikace umožňuje, patří např:

- připojit se k SharePoint místně nebo v cloudu. Není potřeba instalovat žádné součásti serveru, nebo mít zvláštní práva,
- přístup k rozšiřujícím datovým službám jako je OneDrive, Dropbox atd.,
- úprava veškerého obsahu a dokumentů na cestách pomocí bezproblémové integraci s externími editory.

3.4 Vyhodnocení mobilních aplikací

Aplikace, které jsme si výše představili, mají výhodu v tom, že jsou psané pro konkrétní DMS systém. Díky tomu, že aplikace je psaná pro jeden určitý DMS systém, je pro tento systém optimalizována a využívá všechny možnosti, které jim systém umožňuje. Aplikaci ale nemůžeme připojit na jiný systém, než na kterou je aplikace určena.

Co se týká funkcionality, jsou si aplikace Alfresco Mobile a SharePlus podobné. Tyto aplikace obsahují funkce pro plnou správu DMS systému, pro kterou je aplikace určená. Jediná aplikace, která má rozdílnou funkcionalitu je Alfresco Activity, která má funkce pouze pro práci s workflow. Cílem této aplikace je umožnit uživateli schvalovat úkoly pomocí mobilního zařízení. Nevýhodou této aplikace je, že je stále ve vývoji.

4 Mobilní operační systémy a tvorba mobilních aplikací

Mobilní operační systém [19] je upravený operační systém pro osobní počítače, který je přizpůsobený na přenosná zařízení jako jsou mobilní telefony, tablety atd. Tyto operační systémy se snaží kombinovat funkce operačního systému s funkcemi, které dovolí hardware daného zařízení. Tímto hardwarem může být například fotoaparát, GPS, hlasový záznamník atd. Každé zařízení může mít různou konfiguraci hardwaru a proto jsou tyto systémy často výrobcem mobilního zařízení upravovány na konkrétní hardware, který je v zařízených použit.

V dnešní době se používají nejvíce tři operační systémy. Jde o operační systém Android, iOS, a Windows 10 Mobile. Jejich zastoupení v mobilních telefonech je uvedeno v tabulce 4.1. Z tabulky můžeme vysledovat, že Android v posledních letech dominuje a v roce 2015 měl 82,8% zastoupení na trhu. Na druhém místě je iOS, který ale výrazně ztrácí na Android. V roce 2015 iOS měl podíl na trhu 13,9%. Na třetím místě je Windows Phone, předchůdce Windows 10 Mobile, který se v posledních letech pohybuje okolo 3% podílu na trhu. Na posledním místě je operační systém BB OS (BlackBerry OS), který poslední roky stagnuje a v roce 2015 má už pouhé 0,3% zastoupení na trhu.

Období	2012	2013	2014	2015
Android	69,3%	79,8%	84,8%	82,8%
iOS	16,6%	12,9%	11,6%	13,9%
Windows Phone	3,1%	3,4%	2,5%	2,6%
BlackBerry OS	4,9%	2,8%	0,5%	0,3%
Ostatní	6,1%	1,2%	0,7%	0,4%

Tabulka 4.1: Podíl na trhu s mobilními zařízeními . Zdroj IDC, srpen 2015

4.1 Android

Android [20, 21] je mobilní operační systém založený na jádru Linuxu, který je vytvořený společností Android Inc. v roce 2003. Tato firma byla v roce 2005 prodána společnosti Google. V roce 2007 Google společně s dalšími firmami zabývající se mobilní komunikací jako je Samsung, LG, HTC atd. vytvořily konsorcium Open Handset Alliance, které měla za úkol vytvořit mobilní operační systém, jenž by fungoval na mnoha zařízení s různými parametry a každá firma by si ho mohla upravit podle svých požadavků.

Obrázek 4.1: Logo Androidu

První mobil, který se s Androidem ve verzi 1.0 vydal do prodeje, byl v roce 2008 mobil T-Mobile G1 vyrobený firmou HTC.

Obrázek 4.2: Mobil T-Mobile G1

Aktuální verze Androidu 6.0 vstoupila na trh 5.října 2015.

4.1.1 Vývoj aplikací

Mobilní aplikace [22] pro Android se programují v programovacím jazyce Java. Všechny potřebné nástroje, které se využijí při programovaní, obsahuje nástroj Android Software Development Kit. Ten lze stáhnout zcela bezplatně.

Vývojové prostředí

Pro vývoj aplikací se využívalo vývojové prostředí Eclipse s pluginem Android Developer Tools. V roce 2013 Google ale oficiálně představil vývojové prostředí Android studio, které je založené na IntelliJ IDEA community edition, a nyní slouží jako hlavní vývojové prostředí.

Vývojové studio je nabízeno uživatelům zdarma. Je možné si jej stáhnout pro operační systémy Windows, Mac OS X a Linux.

4.1.2 Distribuce aplikace

Pro distribuci vytvořené aplikace slouží služba Google Play Store [23, 24], která nahradila službu Android Market. Služba byla vytvořená firmou Google a spuštěna v roce 2008. Google Play Store je oficiální cestou, jak distribuovat aplikace do Androidu. Android ale také obsahuje možnost získávat aplikace z jiných zdrojů než je oficiální. Tento způsob je ale potřeba povolit, protože je v základním nastavení vypnutý. Tyto zdroje mohou představovat bezpečnostní riziko a je doporučené se jim raději vyhýbat.

Google Play Store je ale o proti ostatním přístupům bezpečnější. O tuto službu se totiž stará přímo společnost Google a všechny aplikace, které jsou do obchodu nahrané prošly schvalovacím procesem. Pokud tedy Google zjistil, že aplikace představuje bezpečnostní riziko nebo aplikace nedělá to, k čemu je aplikace určená, Google tuto aplikaci neschválí a nebude nahrána do obchodu.

Abychom mohli distribuovat aplikace na Google Play Store, je potřeba se nejdříve registrovat. Při registraci se platí jednorázový poplatek, a to 25 dolarů. Všechny aplikace, které jsou nahrány do Google Play Storu jsou podepsané klíčem, který je spárován s vývojářským účtem. Z tohoto důvodu lze jednoznačně určit, kdo danou aplikaci napsal.

4.1.3 Výhody a nevýhody

Výhody

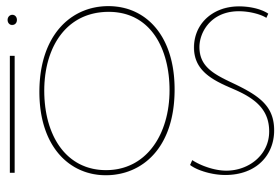
- **Rozšířenost**
  Operační systém android je jednoznačně nejrozšířenější operační systém na světě.
- **Komunita**
  Díky tomu, že Android je nejrozšířenějším operačním systémem, obsahuje také největší komunitu, která může pomoci při různých problémech, které můžou nastat při vývoji.
- **Mnoho nástrojů pro vývoj**
  Pro vývoj aplikací existuje mnoho nástrojů, které programátorům tento proces zjednodušují.

Nevýhody

- **Různorodá zařízení**
  Android je postaven tak, aby fungoval na jakékoliv mobilním zařízení. Problém tohoto řešení je v tom, že programátor ve většině případů dopředu neví, na jakém zařízený se daná aplikace bude spouštět, a nemůže proto aplikaci pro dané zařízení optimalizovat.
- **Verze systémů**
  Velký problém Androidu je v tom, že Google nechává aktualizaci operačního systému na výrobcích zařízení. Ti často poskytují aktualizaci Androidu jen u svých vlajkových lodích. Z tohoto důvodu najdeme ve spoustě zařízení různé verze Androidu, a programátor proto musí počítat s tím, že aplikace může fungovat na různých verzích.
- **Bezpečnost**
  Protože výrobci neaktualizují operační systémy ve výrobcích, viz bod Verze systému, jsou tato zařízení náchylnější na bezpečností rizika.

4.2 iOS

iOS [25, 26] je mobilní operační systém vytvořený firmou Apple Inc. v roce 2007 ještě pod názvem IPhone OS. Tento systém se nejdříve používal pouze v telefonech iPhone, ale následně byl tento systém nasazen i do ostatních zařízení, a proto tento tento systém nese jméno iOS.



Obrázek 4.3: Logo iOS

Systém funguje pouze na zařízeních, které vyrobila firma Apple, a nesmí být použita v zařízeních jiných výrobců.

První mobil, který použil tento systém ve verzi 3.1.3, byl iPhone 1G. Toto zařízení se dostalo na trh v roce 2008.

Obrázek 4.4: Mobil iPhone 1G

Aktuální verze systému je 9.3.1 a vyšla dne 31. března 2016.

4.2.1 Vývoj aplikací

Aplikace pro iOS se dají programovat pouze v operačním systému OS X, který funguje pouze na zařízeních vyrobených firmou Apple Inc. Vývoj aplikací se provádí pomocí programovacích jazyků Objective-C a Swift. Při vývoji se využívá nástroj iPhone SDK, které lze bezplatně stáhnout. Součástí SDK jsou také knihovny Cocoa Touch, ale i vývojové prostředí XCode, které je výchozím vývojovým prostředím.

Vývojové prostředí

XCode je vývojové prostředí pro psaní iOS aplikací vytvořené firmou Apple Inc. Vývojové prostředí obsahuje všechny nástroje a funkce, které programátor pro vývoj potřebuje. Je zde např. debugger pro testování aplikace nebo funkce, která umožňuje spuštění aplikace v emulátoru, nebo v reálném zařízení.

Vývojové prostředí XCode lze stáhnout bezplatně z Mac App Storu.

4.2.2 Distribuce aplikace

V roce 2008, kdy byl uveden obchod se softwarem App Store [27, 28], je tento obchod jediným místem, kde lze distribuovat vytvořené aplikace do iOS. Díky tomu je zajištěno, že v obchodě najdeme pouze důvěryhodné aplikace.

Aby vývojář mohl přidat svojí aplikaci do App Storu, musí si nejdříve za 99 dolarů na rok koupit vývojářský program. Vývojářský program obsahuje nástroje, které umožní spravovat naši aplikaci, kterou chceme publikovat.

Každou aplikaci, kterou chceme nahrát na App Store, musí projít schvalovacím procesem. Tento proces se týká i toho, kdy chceme pouze aktualizovat současnou aplikaci. Proces schvalování trvá několik dní. Z důvodu několikadenní prodlevy může pro vývojáře nastat vážný problém. Tento problém vznikne v případě, kdy je aplikace připravena ke stažení a zjistí se, že obsahuje bezpečností nebo funkční problém. I přesto, že vývojář problém rychle vyřeší, musí počkat, než aktualizace projde schválením.

4.2.3 Výhody a nevýhody

Výhody

- **Komunita**
  I přesto, že systém iOS není oproti Androidu tak rozšířený, má spoustu loajálních uživatelů, kteří na Apple nedají dopustit.
- **Známý hardware**
  Z důvodu, že systém iOS funguje pouze na zařízeních vyrobené Apple, je tedy dopředu známo, na jakém hardwaru může aplikace fungovat. Z tohoto důvodu mohou vývojáři svojí aplikaci optimalizovat pro dané zařízení.
- **Aktuální verze systému**
  Společnosti Apple se daří aktualizovat systém na svých zařízeních. Díky tomu není systém roztříštěný na zařízeních, a pro vývojáře je poté vývoj aplikace jednodušší.
- **Kvalita aplikací**
  Pokud aplikace nesplňuje přísná měřítka kvality, společnost Apple neschválí aplikaci do AppStoru.
- **Bezpečnost**
  Aplikace lze stáhnout pouze z oficiálního obchodu, kde najdeme jen důvěryhodné aplikace. Pro každou nainstalovanou aplikaci můžeme také jednotlivě přidělit práva.

Nevýhody

- **Vývoj pouze na OS X**
  Aplikace se dají vyvíjet pouze na OS X. Pokud tedy vývojář chce vyvíjet aplikace, musí mít možnost dělat na MACu.
- **Drahá zařízení**
  iOS funguje pouze na zařízeních, které vyrobila společnost Apple a nepatří zrovna mezi nejlevnější.
- **Drahá licence pro AppStore**
  Aby programátor mohl vydat svojí aplikaci, musí mít licenci do AppStore, která je každoročně zpoplatňována.

4.3 Windows 10 Mobile

Windows 10 Mobile [29] je mobilní operační systém vyrobený firmou Microsoft Corporation. Byl vydán v roce 2015 jako náhrada za Windows Phone. Cílem Microsoftu je sjednotit mobilní a PC verzi operační systému. Z marketingového důvodu se proto tomuto operačnímu systému říká pouze Windows 10 a má stejný název jako operační systém pro osobní počítače.

Obrázek 4.5: Logo Windows 10

Jeden z prvních mobilů, které se dostaly na trh s Windows 10, byl mobil Microsoft Lumia 950 z roku 2015.

Obrázek 4.6: Mobil Microsoft Lumia 950

Windows 10 ve verzi 10.0.10586.218 je aktuálně nejnovější verze operačního systému od Microsoftu. Tato verze vyšla 12. dubna 2016.

4.3.1 Vývoj aplikací

Pro vývoj [30] mobilních aplikací pro Windows 10 se využívá knihovna Windows Runtime. Tato knihovna umožňuje vytvářet mobilní aplikace ve více programovacích jazycích. Technologie, které můžeme použít pro vývoj aplikace jsou:

- C# nebo Visual Basic s XAML
- C++ s XAML
- JavaScript s HTML/CSS.

Vývojové prostředí

Pro vývoj mobilních aplikací pro operační systém Windows 10 se využívá nejčastěji vývojové prostředí Visual Studio, které bylo vyrobeno firmou Microsoft. Visual Studio obsahuje všechny potřebné nástroje pro vývoj mobilních aplikací.

Aplikace Visual Studio je dostupná pouze pro operační systém Windows a je možné ho získat zdarma nebo v placené verzi. Ty se mezi sebou liší svojí funkčností.

4.3.2 Distribuce aplikace

Pro distribuci aplikací se využívá Windows Store. Windows Store je obchod s aplikacemi vytvořený firmou Microsoft Corporation jako náhrada za Windows Marketplace, který přišel na trh společně s Windows 8 v roce 2012.

Všechny aplikace, které jsou dostupné na Windows Store museli být schváleny společností Microsoft, která se zaměřuje, jestli aplikace neobsahuje nevhodný obsah.

Registrace do Windows Store stojí programátora 19 dolarů a společnost 99 dolarů. Microsoft poté ještě nechává 30% nebo 20% z prodeje aplikace podle toho, jestli prodej aplikace přesáhne 25 000 dolarů nebo ne.

4.3.3 Výhody a nevýhody

Výhody

- **Více programovacích jazyků**
  Aplikace můžeme psát ve více programovacích jazycích a je jen na programátorovi, která technologie mu nejvíce vyhovuje.
- **Známý hardware v mobilních zařízeních**
  Operační systém funguje na zařízeních vyrobených firmou Microsoft. Díky tomu mohou vývojáři optimalizovat svoje aplikace pro daný hardware.

Nevýhody

- **Rozšířenost**
  Windows 10 na mobilních zařízeních oproti iOS a hlavně Androidu tak rozšířený.
- **Windows Store**
  Windows Store není mezi vývojáři oblíbený i přesto, že se Microsoft snaží nalákat vývojáře pro psaní aplikací na Windows 10.
- **Nekvalita aplikací**
  Ve Windows Store se objevují aplikace, které nemají dostatečnou kvalitu. Je to důsledek toho, že Microsoft chtěl mít ve Windows Store více aplikací, a proto nehlídal tolik jejich kvalitu.

5 Návrh mobilní aplikace pro DMS

V této kapitole si nejdříve provedeme analýzu požadavků na mobilní aplikaci, kterou jsme získali od zadavatele. Poté si analyzujeme procesy, které mohou být definované ve firmě. Následně provedeme výběr architektury a navrhneme mobilní aplikaci podle získaných požadavků.

5.1 Požadavky

Požadavky byly získány pomocí pravidelných schůzek ze zadavatelem, při němž jsme diskutovali o výsledné aplikaci.

5.1.1 Funkční požadavky

- **Přihlášení**
  Uživatel se musí do aplikace přihlásit, pokud uživatel není přihlášený, nemůže provádět akce. Přihlášení se bude provádět pomocí uživatelského jména a hesla.

- **Úlohy**
  Po přihlášení se do aplikace uvidí uživatel všechny úlohy, které má zpracovat. U každé úlohy je uveden název a datum, do kdy je třeba úlohu vyřídit. U každé úlohy musí jít provést akce. Po kliknutí na úlohu se zobrazí její detail.

- **Detail úlohy**
  Při zobrazení detailu úlohy musí být patrné všechny informace o úloze, jako např. jméno, datum vytvoření atd. Pokud úloha obsahuje hlavní dokument, musí být tento dokument hned vidět. Všechny dokumenty, které obsahuje úloha nemusí být vidět, ale musí se k nim uživatel jednoduše dostat. Uživatel si každý dokument může otevřít nebo zobrazit jeho metadata. Každý dokument může mít položky, které je potřeba mít možnost zobrazit. V úloze je také potřeba umožnit uživateli provést akci nad úlohou.

- **Zobrazení dokumentu**
  Dokument bude zobrazen externí aplikací, která je v mobilním zařízení

nainstalována. Pokud aplikace neexistuje, vypíše se hláška o tom, že dokument nelze otevřít z důvodu, že neexistuje v mobilním zařízení aplikace.

- **Metadata dokumentu**
  Na obrazovce budou zobrazena metadata. Metadata budou mít statické hodnoty, jako např. název dokumentu a dynamické hodnoty. Dynamické hodnoty budou popsány pomocí JSON objektu. Dynamická hodnota musí být minimálně popsána názvem, jestli je hodnota měnitelná, jaký má typ a jestli je povinná.

- **Akce**
  Zobrazí se pokud je potřeba k akci nad úlohou doplnit údaje. Akce obsahuje stejné dynamické hodnoty, jaké jsou popsány v bodě *Metadata dokumentu*.

- **Notifikace**
  Pokud je uživatel přihlášený a aplikace je na pozadí nebo vypnutá, bude spuštěna služba, která bude uživatele pomocí notifikací informovat o počtu nových úloh.

- **Nastavení**
  Z aplikace musí jít před přihlášením zadat IP adresu serveru a port, na kterém funguje aplikace Adaptér.

5.1.2 Nefunkční požadavky

- **Platforma**
  Půjde o mobilní aplikaci, která bude naprogramována nativně pro některý z operačních systémů, a bude optimalizována pro mobilní zařízení s úhlopříčkou cca 5 palců.

- **Komunikace**
  Aplikace nebude komunikovat přímo z DMS systémem, ale bude komunikovat s aplikací, která bude transformovat požadavky na konkrétní DMS systém. Komunikace mezi aplikacemi bude probíhat pomocí REST API [31], které bude navrženo tak, aby transformace byla možná.

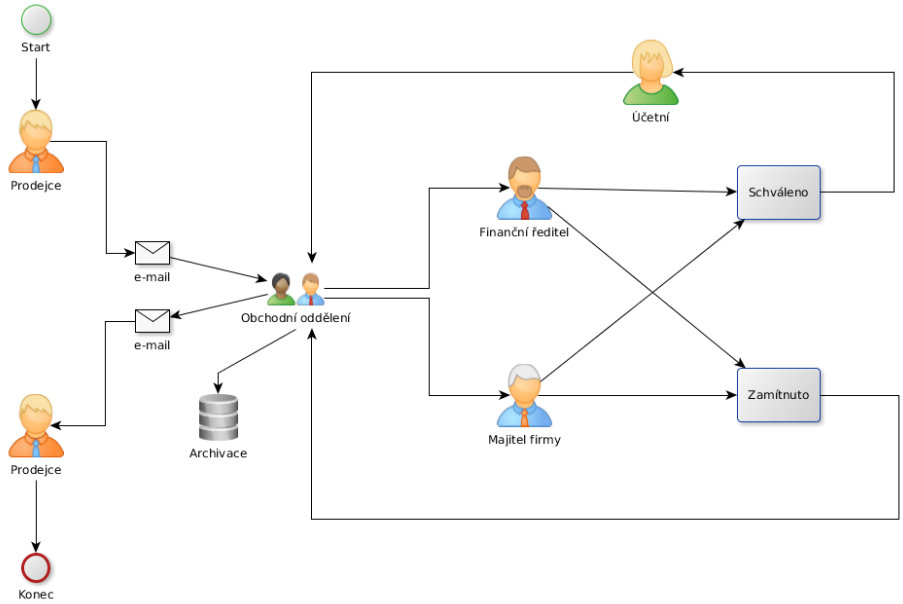
- **Bezpečnost**
  Z důvodu, že jde o tajné dokumenty, které smí vidět jen oprávněná osoba, musí být data při přenosu šifrovaná.

- **Vzhled aplikace**
  Uživatelské rozhraní aplikace musí být navrženo tak, aby se dalo jednoduše a intuitivně ovládat a bylo možné co nejrychleji zpracovat úlohu.
- **Ověření uživatele**
  Po ověření uživatele pomocí uživatelského jména a hesla se bude uživatel ověřovat pomocí tokenu. Pokud dojde platnost tokenu, aplikace si tento token sama obnoví bez nutnosti zásahu uživatele.

5.2 Procesy

Každá firma má podle svých potřeb definované procesy, jak se budou zpracovávat dokumenty, které přijdou do firmy. Těchto procesů může být ve firmě více podle velikosti firmy, hierarchii vedení atd. V této části si popíšeme jeden z možných procesů, který může být ve firmě definován.

Popíšeme si zde, jak může probíhat schvalování ceny nákupu nového hardwaru do firmy. Protože půjde o větší částku, je potřeba, aby nákup schválil jak finanční ředitel, tak i majitel firmy. Dokument přijde do firmy pomocí e-mailu. Celý proces je zachycen na obrázku 5.1.



Obrázek 5.1: Proces pro schvalování nového hardwaru

5.2.1 Vstupní bod

Každý proces musí mít svůj vstupní bod, který umožní vložit do DMS sytému nový dokument. Vložením nového dokumentu do systému vznikne nový proces, který vytváří uživatelům úlohy, jež se musí k vloženému dokumentu vyjádřit.

Vstupním bodem ve firmě může být např. obchodní oddělení. Obchodní oddělení může získat dokument z více zdrojů jako je například datová schránka, pošta, e-mail atd. V našem případě obchodní oddělení dostalo dokument od prodejce hardwaru pomocí e-mailu, který k nim přišel na oddělení. Lidé pracující v obchodním oddělení tento dokument vezmou a vloží ho do DMS systému a určí, které odpovědné osoby tento dokument musí zpracovat. Jelikož jde o nákup nového hardwaru s vysokou cenou, je třeba tento dokument poslat ke schválení jak finančnímu řediteli tak i majiteli firmy. Po vložení nového dokumentu se vytvoří majiteli firmy a finančnímu řediteli úloha, v níž tento vložený dokument schválí nebo zamítnou.

5.2.2 Proces schvalování nebo zamítnutí

Finanční ředitel a majitel firmy se k dokumentu vyjadřují paralelně. Pokud alespoň jeden nebude souhlasit, dokument bude zamítnut a koupě se neuskuteční. Pokud se oba vyjádří souhlasně, dokument bude schválen a poslán účetní, která koupi nového hardwaru zaúčtuje. Poté je dokument poslán při schválení i zamítnutí zpět na obchodní oddělení, které je výstupním bodem.

5.2.3 Výstupní bod

Všechny procesy musí mít také výstupní bod, kde bude proces ukončen.

Výstupním bodem může být obchodní oddělení, které vezme zpracovaný dokument a dá ho archivovat pro budoucí potřeby. Poté napíše prodejci podle vyjádření majitele firmy a finančního ředitele, jestli nakup bude realizován nebo ne. Informaci obchodní oddělení pošle prodejci pomocí e-mailu.

5.3 Architektura

V této části si provedeme analýzu architektury. Nejdříve si vybereme mobilní operační systém pro který bude aplikace napsaná, jak bude celkově aplikace nasazena a probereme zde také to, jak aplikaci zabezpečíme.

5.3.1 Výběr mobilního operačního systému

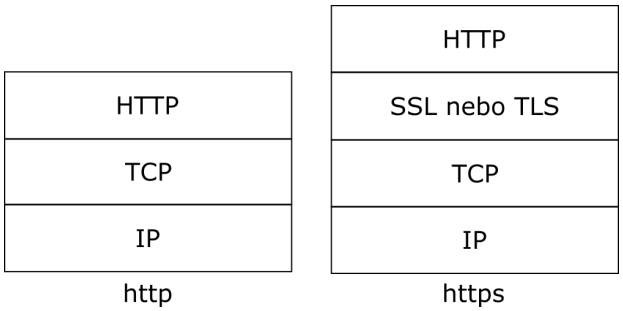
Pro vývoj aplikace jsem si vybral mobilní operační systém Android. Zvolil jsem si jej proto, že je to momentálně nejrozšířenější operační systém a také proto, že s tímto operačním systémem mám osobně nejvíce zkušeností.

5.3.2 Zabezpečení

Protože se pro komunikaci s Adaptérem bude využívat REST API, které je založeno na protokolu HTTP, je nejlepším možným řešením využít protokol HTTPS, jenž komunikaci šifruje.

Protokol HTTPS

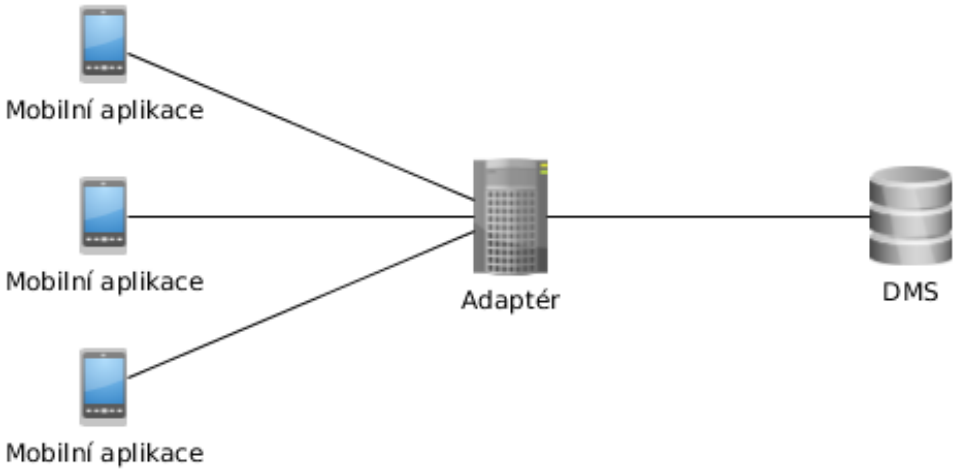
Protokol HTTPS [33] je nadstavba nad protokolem HTTP, který se stará o bezpečnou komunikaci mezi klientem a serverem. Data, která jsou přenášena mezi klientem a serverem jsou šifrována pomocí SSL nebo TLS.



Obrázek 5.2: Rozdíl mezi http a https ve vrstvách

5.3.3 Diagram nasazení

Celý systém bude rozdělen na tří části. Mobilní aplikace, Adaptér a DMS.



Obrázek 5.3: Diagram nasazení

Mobilní aplikace

Mobilní aplikace umožní přístup uživateli k provedení operací nad dokumentem, který má uživatel zpracovat. Uživatel nemusí vědět s jakým DMS systémem pracuje. Aplikace ho od toho odstiňuje. Aplikace musí znát pouze IP adresu serveru v němž funguje aplikace Adaptér.

Adaptér

Jde o aplikaci, která transformuje REST API mobilní aplikace na REST API konkrétního DMS. Aplikace musí vědět, s jakým DMS systémem mobilní aplikace pracuje.

DMS

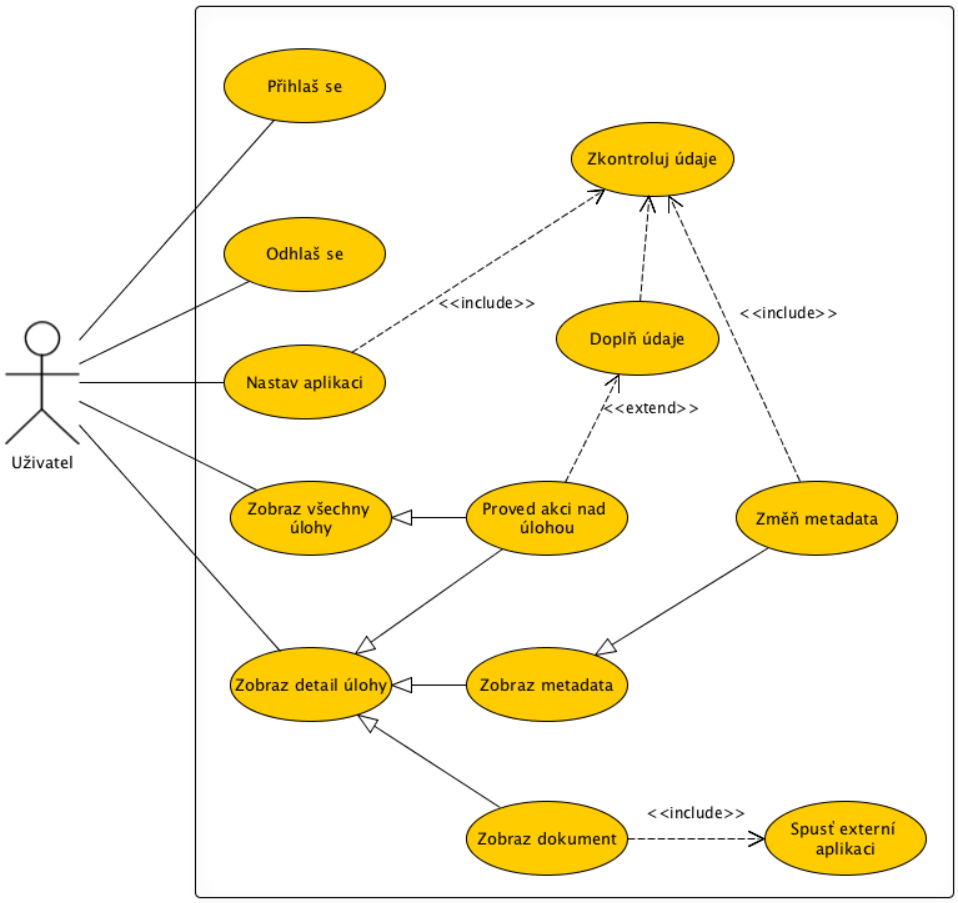
Jde o konkrétní DMS systém, s nímž bude mobilní aplikace pracovat. Zde může být například Alfresco, SharePoint atd.

5.4 Návrh

V této části si nejdříve určíme případy užití, na to si navrhneme datový model, uživatelské rozhraní aplikace a nakonec REST API pro komunikaci mezi mobilní aplikací a aplikací Adaptér.

5.4.1 Případy užití

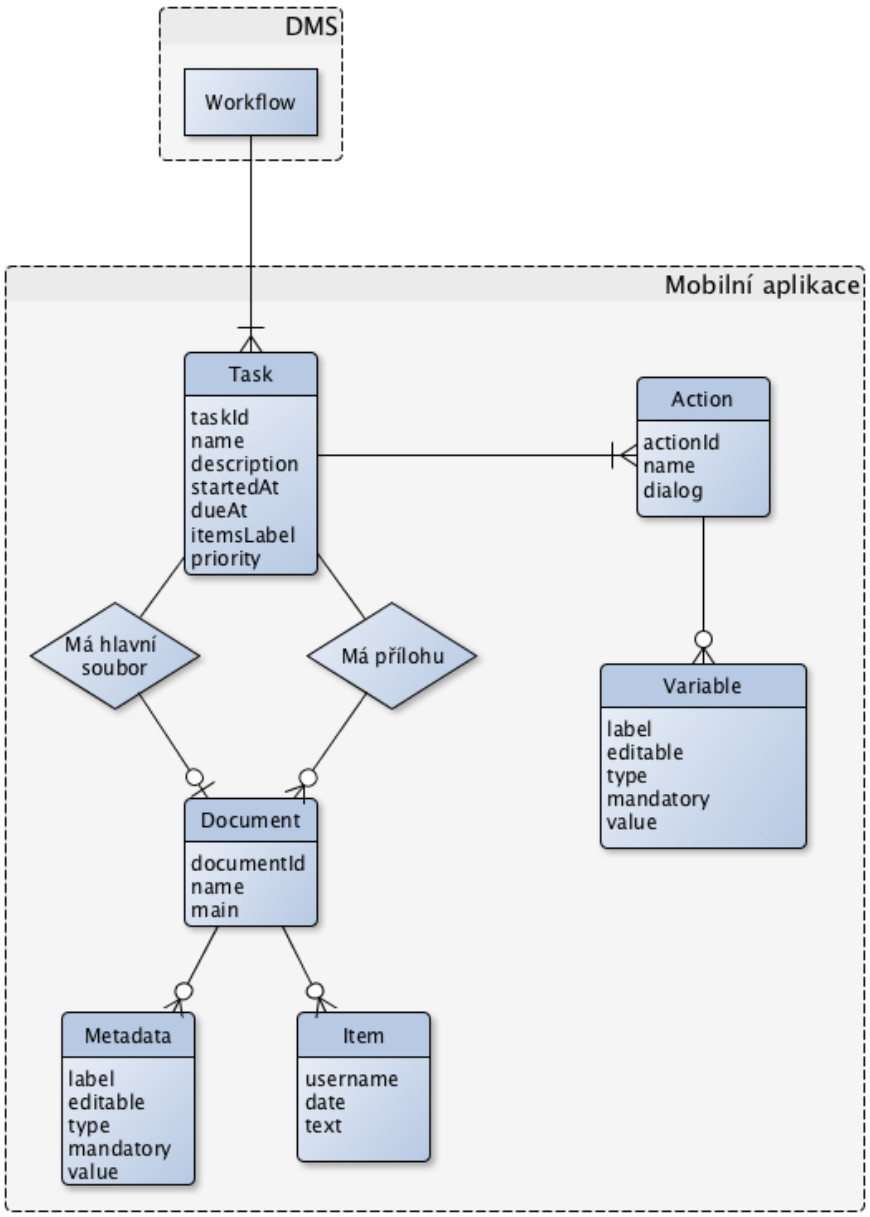
Případ užití zahrnuje pouze jednoho aktéra a to uživatele aplikace. Akce *Přihlaš se* a *Nastav aplikaci* může uživatel provádět bez přihlášení. Pro vykonání ostatních akci musí být uživatel přihlášen.



Obrázek 5.4: Diagram případu užití

5.4.2 Datový model

Na níže uvedeném obrázku (Obrázek 5.5) je vidět datový model na konceptuální úrovni. Datový model je rozdělen na dvě části, a to DMS a Mobilní aplikace.



Obrázek 5.5: Datový model

DMS

Ve Skupině DMS je entita Workflow, s níž mobilní aplikace nepracuje, ale tato entita generuje entitu Task.

Mobilní aplikace

Skupina Mobilní aplikace obsahuje entity, s nimiž už mobilní aplikace pracuje. Jde o entity Task, Item, Document, Metadata, Action a Variable.

- **Task**
  Task je entita, kterou generuje Workflow v DMS. Tato entita reprezentuje jednu úlohu, kterou má uživatel zpracovat.
- **Item**
  Jde o entitu, která reprezentuje nějakou informaci o dokumentu. Informací může být např. komentář.
- **Document**
  Document je entita, která reprezentuje soubor uložený v DMS. Tato entita je spojena s entitou Task. Task může mít 0 až 1 hlavní soubor, nebo 0 až N příloh.
- **Metadata**
  Metadata je entita, která reprezentuje jednu položku v metadat dokumentu.
- **Action**
  Action je entita reprezentující akce, které se mohou nad úlohou provést. Akce může mít proměnné, které je potřeba při akci vyplnit.
- **Variable**
  Variable je entita, která reprezentuje jednu proměnnou, jíž je potřeba vyplnit před provedením akce.

5.4.3 Uživatelské rozhraní

Zde si provedeme návrh uživatelského rozhraní. Uživatelské rozhraní by mělo být navrhnuto tak, aby bylo ovládání aplikace co nejjednodušší a pro uživatele intuitivní.

Přihlášení

Přihlášení je úvodní stránka aplikace. Na této stránce je nutné umožnit uživateli zadat uživatelské jméno a heslo. Z této stránky je třeba se také dostat do nastavení. Návrh přihlášení můžeme vidět na obrázku 5.6.

Nastavení

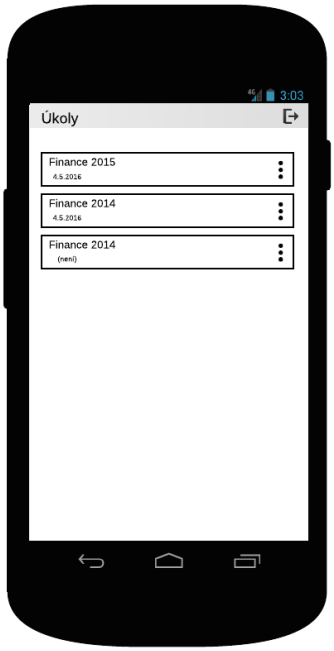
V nastavení je třeba mít možnost zadat IP adresu serveru a port, v němž funguje aplikace Adaptér. Návrh nastavení vidíme na níže uvedeném obrázku 5.7.

Obrázek 5.6: Návrh přihlášení

Obrázek 5.7: Návrh nastavení

Pracovní stůl

Stránka bude obsahovat všechny úlohy, jež musí uživatel vyřídit. U každé úlohy je potřeba zobrazit název a datum, do kdy se daná úloha musí zpracovat. U každé úlohy musí být také zobrazeny akce, které úloha má. Z této stránky se také musí jít odhlásit.



Obrázek 5.8: Návrh pracovního stolu

Notifikace

Notifikace se vytváří, pokud je aplikace zavřená nebo na pozadí. Notifikace bude obsahovat počet nových úloh.

Obrázek 5.9: Návrh notifikace

Detail úlohy

Na této stránce musí být zobrazeny všechny údaje o úloze a všechny soubory, které úloha má. Uživatel může soubor otevřít, nebo přejít na úpravu metadat. Uživatel také muže provést akci nad úlohou, nebo se odhlásit. Na této stránce si taky může uživatel přečíst historii dokumentu, kde bude uveden datum změny, kdo změnu provedl a její popis.

Obrázek 5.10: Návrh detailu úlohy

Dokument

Stránka, na které se bude zobrazovat soubor v úloze, bude externí aplikace. Díky této vlastnosti si uživatel může sám vybrat v jaké aplikaci se daný soubor otevře.

Metadata dokumentu

Stránka Metadata dokumentu (Obrázek 5.11) bude zobrazovat metadata dokumentu. Na této stránce je nutné uživateli umožnit změnu metadat dokumentu a jejich uložení. Může se zde také odhlásit z aplikace.

Akce

Tato stránka se zobrazí, pokud je potřeba k akci doplnit údaje k provedení akce. Musí umět zobrazit proměnné, které je potřeba vyplnit před odesláním. Zde je také možnost uživatele odhlásit. Návrh stránky lze vidět na obrázku 5.12.

Obrázek 5.11: Návrh metadat

Obrázek 5.12: Návrh akce

5.4.4 Komunikační rozhraní

Zde si navrhneme REST API Adaptéru, přes který bude mobilní aplikace komunikovat.

Pro přenos dat bude využit JSON formát. Jediná výjimka je při přenosu souboru, kde je využit datový stream.

Všechny požadavky kromě Přihlášení a Odhlášení musí obsahovat v hlavičce HTTP klíč **token**, který obsahuje token přihlášeného uživatele.

Každá odpověď ve formátu JSON obsahuje klíč **status**, která může nabývat tří hodnot a to *ok*, *unauthorized* a *error*. Hodnota *ok* signalizuje, že požadavek proběhl v pořádku. Hodnota *unauthorized* signalizuje, že uživatel není přihlášený a hodnota *error* signalizuje chybu.

V REST API se používají tyto objekty, které jsou popsány níže v tabulkách. Klíče označené hvězdičkou jsou povinné.

Úloha		
Klíč	**JSON typ**	**Popis**
taskId *	String	ID úlohy.
name *	String	Název úlohy.
description	String	Popis úlohy.
startedAt *	String	Datum vzniku úlohy ve formátu ISO 8601.
dueAt	String	Datum ukončení úlohy ve formátu ISO 8601.
priority	String	Priorita úlohy. Výchozí hodnota je Nízká.
itemsLabel	String	Obsahuje název uložených položek. Výchozí hodnota je Komentáře.
actions *	pole objektů	Obsahuje pole akcí.
documents	pole objektů	Obsahuje pole dokumentů.

Dokument		
Klíč	**JSON typ**	**Popis**
documentId *	String	ID dokumentu.
name *	String	Název dokumentu včetně přípony.
main	boolean	Říká, jestli je dokument hlavní nebo ne. Hlavní dokument může být jen jeden. Výchozí hodnota je false.
items	pole objektů	Obsahuje pole položek.

Akce		
Klíč	**JSON typ**	**Popis**
actionId *	String	ID akce.
name *	String	Název akce.
dialog	boolean	Výchozí hodnota false. Pokud je false, akce nepotřebuje doplnit údaje a může se hned odeslat. Pokud je hodnota true, akce obsahuje proměnné, které je potřeba před odesláním vyplnit.
variable	pole objektů	Obsahuje pole proměnných.

Položka		
Klíč	**JSON typ**	**Popis**
username *	String	Uživatelské jméno.
date *	String	Datum ve formátu ISO 8601.
text	String	Text.

Proměnná		
Klíč	**Typ**	**Popis**
label *	String	Název proměnné.
type *	String	Typ proměnné. Může být *string*, *number*, *select* a *date*.
value	String	Hodnota proměnné.
editable	boolena	Jestli je proměnná měnitelná. Výchozí hodnota je true.
mandatory	String	Jestli je proměnná povinná. Výchozí hodnota je false.
option	pole objektů	Pouze pro typ *select*. Objekt obsahuje pouze klíč *value*, který reprezentuje jednu možnost v selectboxu.

Přihlášení

Požadavek pro přihlášení uživatele.

```
POST /auth/login
```

Tělo obsahuje dva klíče, a to **username**, které obsahuje uživatelské jméno a **password**, kde je heslo uživatele. Oba klíče jsou povinné.

```
{
   "username": "martin",
   "password": "1234"
}
```

Pokud hodnota klíče **status** je *ok*, přihlášení se podařilo a jako další klíč je **token**, který obsahuje *token* přihlášeného uživatele. Pokud má **status** hodnotu *unauthorized*, nepodařilo se uživatele přihlásit z důvodu špatného uživatelského jména nebo hesla.

```
{
   "status": "ok",
   "token": "TICKET_ABS1S234978VS"
}
```

Odhlášení

Požadavek pro odhlášení uživatele.

```
DELETE /auth/logout/<token>
```

Odpověď obsahuje pouze klíč **status**, kde hodnota *ok* říká, že se uživatele podařilo odhlásit. Příklad úspěšného odhlášení.

```
{
   "status": "ok"
}
```

Získání úloh

Požadavek pro získání základních údajů všech úloh uživatele.

```
GET /tasks
```

Pokud obsahuje klíč **status** hodnotu *ok*, klíč **data** obsahuje pole, kde jsou uloženy objekty úloh. Úloha má klíče **taskId**, **name**, **startedAt**, **dueAt** a pole akcí. Akce obsahuje klíče **actionId**, **name** a **dialog**.

```
{
  "status": "ok",
  "data": [
    {
      "taskId": "4679",
      "name": "Adhoc Task",
      "startedAt": "2016-04-28T16:51:00.593+0000",
      "dueAt": "2016-04-29T22:00:00.000+0000",
      "actions": [{
              "actionId": "taskDone",
              "name": "Dokoncit",
              "dialog": "false"
        } , {
              "actionId": "taskDone_note",
              "name": "Dokoncit s poznamkou",
              "dialog": "true"
        }
        ]
    }
  ]
}
```

Získání úlohy

Požadavek pro získání jedné úlohy pomocí ID.

```
GET /tasks/<id>
```

Pokud má klíč **status** hodnotu *ok*, klíč **data** obsahuje objekt úlohy. Úloha obsahuje všechny možné klíče. Akce v úloze obsahují klíče **actionId**, **name** a **dialog**. Dokumenty v úloze obsahují všechny možné klíče a položky v dokumentu také.

```
{"status": "ok",
  "data": {
    "taskId": "4774",
    "name": "Verify Adhoc Task Completed.",
    "description": "Logo skupiny",
    "startedAt": "2016-04-28T22:34:12.217+0000",
    "dueAt": "2016-04-29T22:00:00.000+0000",
    "priority": "Stredni",
    "actions": [{ "actionId": "taskDone", "name":
       "Dokoncit","dialog": "false"}],
    "documents": [{
         "documentId": "7d9a087b-7b3f-408f-92c7-fe6f6a96a432",
         "name": "sb-logo.png",
         "label": "Logo",
         "main": true,
         "items": [{ "username": "jmeno1", "date":
            "2016-04-29T22:00:00.000+0000", "text": "zmena
            metadat"}]
    }]
  }
}
```

Získání akce

Požadavek na získání akce pomocí ID.

```
GET /actions/<id>
```

Pokud má klíč **status** hodnotu *ok*, klíč **data** obsahuje objekt akce. Akce obsahuje všechny možné klíče. Všechny možné klíče také obsahují proměnné, které se v akci nacházejí.

```
{
  "status": "ok",
  "data": {
    "actionId": "taskDone_note",
    "name": "Dokoncit s poznamkou",
    "dialog": "true",
    "value": "",
    "variables": [
      {
        "label": "Poznamka",
        "type": "string",
        "value": "",
        "editable": "true",
        "mandatory": "true"
      }
    ]
  }
}
```

Získání počtu nových úloh

Požadavek na získání počtu nových úloh od určitého datu.

```
GET /document/number
```

V URL se předává v proměnné **date** datum ve formátu ISO 8601.

```
/document/number?date=2016-04-11T12:24:47.168+0000
```

Pokud v klíči **status** hodnotu *ok*, v klíčí **number** je uložený počet nových úloh.

```
{
    "status": "ok",
    "number": 3
}
```

Získání metadat dokumentu

Požadavek pro získání metadat dokumentu.

```
GET /document/<id>/metadata
```

Pokud má **status** hodnotu *ok*, klíč data obsahuje pole proměnných. Proměnné mohou mít všechny možné klíče.

```
{
  "status": "ok",
  "data": [{
        "label": "Poznamka",
        "type": "string",
        "value": "",
        "editable": "true",
        "mandatory": "true"
   }, {
        "label": "Pracoviste",
        "type": "select",
        "value": ""
        "option": [{
          "value": "Plzen"
         }, {
           "value": "Praha"
         }, {
           "value": "Brno"
         }],
         "editable": "true",
         "mandatory": "false"
   }]
}
```

Získání dokumentu

Požadavek na získání dokumentu.

```
GET /document/<id>
```

Odpověď na požadavek je datový stream.

Provedení akce na úlohou

Požadavek na provedení akce nad úlohou.

```
POST /tasks/<id>
```

Tělo obsahuje objekt akce, která má stejné klíče, jako při získání úlohy.

```
{
    "actionId": "taskDone_note",
    "name": "Dokoncit s poznamkou",
    "dialog": "true",
    "value": "",
    "variables": [
      {
        "label": "Poznamka",
        "type": "string",
        "value": "doplneny udaj",
        "editable": "true",
        "mandatory": "true"
      }
    ]
}
```

Odpověď od serveru, pokud je vše v pořádku.

```
{
    "status": "ok"
}
```

Uložení metadat dokumentu

Pomocí tohoto požadavku provedeme změnu metadat.

```
POST /document/<id>
```

Tělo obsahuje klíč **data**, ve kterém je uložené pole proměnných, které obsahují stejné klíče, jako při získání metadat.

```
"data": [{
       "label": "Poznamka",
       "type": "string",
       "value": "",
       "editable": "true",
       "mandatory": "true"
  }, {
       "label": "Pracoviste",
       "type": "select",
       "value": ""
       "option": [{
          "value": "Plzen"
        }, {
           "value": "Praha"
        }, {
           "value": "Brno"
        }],
        "editable": "true",
        "mandatory": "false"
  }]
```

6 Implementace mobilní aplikace

V této kapitole se budeme věnovat mobilní aplikaci, kterou jsem vytvořil podle návrhu z kapitoly 5.

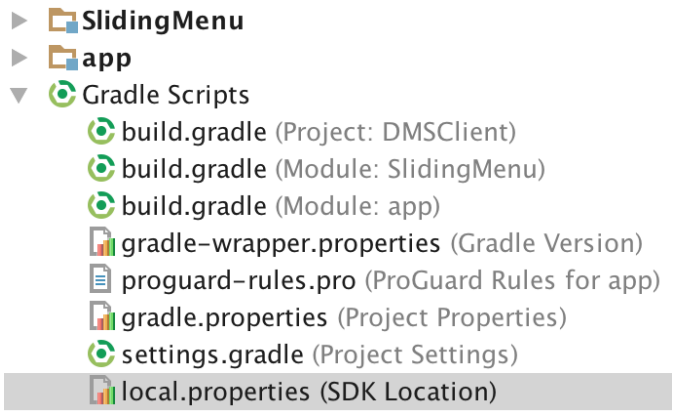
6.1 Základní informace

Aplikace je naprogramována pro mobilní operační systém Android. Jako minimální a cílová verze API je použita verze 16. Jde tedy o Android ve verzi 4.1. Tu jsem si vybral z důvodu, že jsem chtěl použít verzi, kterou lze spustit na většině mobilních zařízení, na nichž běží operační systém Android. Podle statistik společnosti Google [32] lze aplikaci spustit na 95,7% zařízeních běžících na Androidu.

Pro aplikaci jsem zvolil název DMSClient.

6.2 Struktura projektu

Název projektu je DMSClient. Ten obsahuje dva moduly. Jedná se o aplikační modul app, který obsahuje mnou vytvořenou aplikaci a modul SlidingMenu viz kapitola 6.2.2. Ten byl do projektu importován.



Obrázek 6.1: Struktura projektu

6.2.1 Struktura modulu app

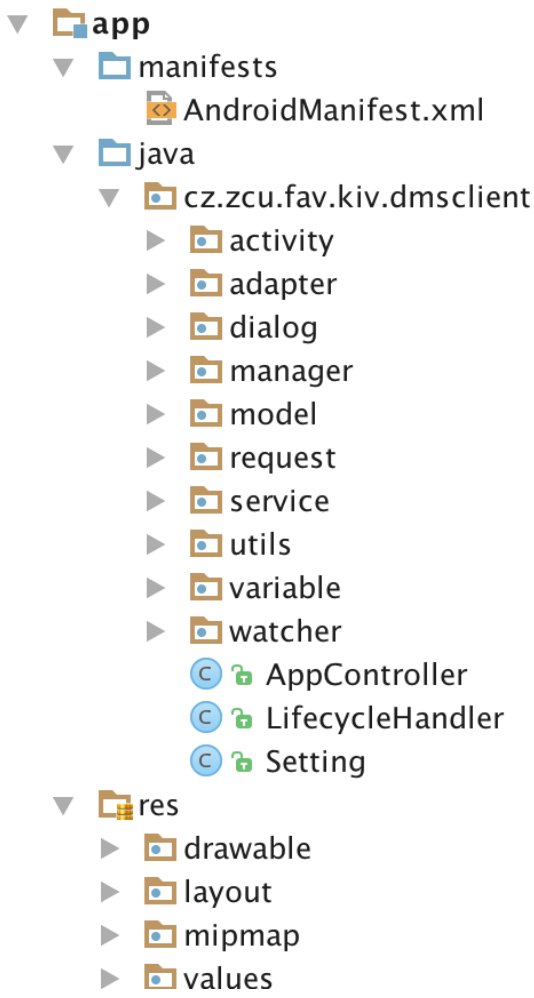
Nyní bych rád popsal strukturu mojí aplikace po balíčkách a co jednotlivé balíčky zahrnují:

- **cz.zcu.fav.kiv.dmsclient**
  Hlavní balíček aplikace, který obsahuje třídy, jež se starají o aplikaci. Jde o třídy *AppController*, *LifecycleHandler* a *Setting*.
- **cz.zcu.fav.kiv.dmsclient.activity**
  V tomto balíčku nalezneme všechny aktivity, které se v aplikaci nacházejí. Je zde třída *GenericActivity*, která dědí od *AppCompatActivity* a stará se o odhlášení z aplikace. Třídu *GenericActivity* dědí všechny třídy, až na třídu *LoginActivity*.
- **cz.zcu.fav.kiv.dmsclient.adapter**
  Balíček obsahuje třídy, které jsou použity pro zobrazení dat. Všechny tyto třídy dědí od *BaseAdapter*.
- **cz.zcu.fav.kiv.dmsclient.dialog**
  Obsahuje mnou vytvořené dialogy. Tyto dialogy jsou vytvořeny pomocí dědění od třídy *Dialog*.
- **cz.zcu.fav.kiv.dmsclient.manager**
  Tento balíček obsahuje pouze jednu třídu, která se stará o ověřování certifikátu od serveru. Třída implementuje *X509TrustManager*.
- **cz.zcu.fav.kiv.dmsclient.model**
  V tomto balíčku nalezneme všechny datové třídy, které se využívají v aplikaci.
- **cz.zcu.fav.kiv.dmsclient.request**
  Balíček request obsahuje všechny třídy, které provádějí https požadavky pomocí knihovny Volley viz kapitola 6.2.2.
- **cz.zcu.fav.kiv.dmsclient.impl.request**
  Obsahuje pouze jednu třídu, která dědí od třídy *Request* a slouží pro přijetí pole bajtů.
- **cz.zcu.fav.kiv.dmsclient.service**
  Zde nalezneme třídu, která vytváří službu pro notifikaci uživatele.

- **cz.zcu.fav.kiv.dmsclient.utils**
  V tomto balíčku nalezneme obecné třídy, které se v aplikaci používaji. Jde hlavně o třídu *JsonToObject*, jež převádí JSON objekt na datové třídy.

- **cz.zcu.fav.kiv.dmsclient.variable**
  Balíček obsahuje třídy, které reprezentují proměnnou. Je zde třída *GenericVariable*, od níž dědí všechny ostatní třídy v balíčku.

- **cz.zcu.fav.kiv.dmsclient.watcher**
  Zde nalezneme třídy, které se starají o vstup pomocí textového pole. Tyto třídy implementují rozhraní *TextWatcher*.



Obrázek 6.2: Struktura modulu app

Manifest

AndroidManifest.xml je důležitý soubor aplikace. Manifestu se např. určují, jaké má aplikace aktivity či služby. Zde se také nastavuje, jaká aktivita se spustí po spuštění aplikace. V manifestu lze také nastavit práva, která jsou potřeba pro chod aplikace. Práva, která zde můžeme nastavit, jsou např. přístup k internetu, zápis/čtení z externí paměti atd.

```
<activity
    android:name=".activity.LoginActivity"
    android:label="@string/app_name">
    android:configChanges="orientation|screenSize">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Obrázek 6.3: Příklad aktivity v manifestu

6.2.2 Knihovny

V aplikaci jsou použity knihovny, které nám usnadňují implementaci aplikace. Knihovny jsou staženy pomocí nástroje Gradle až na knihovnu SlidingMenu. Tato knihovna je stažena z GitHubu [35] a importována do projektu ručně. Knihovny, které jsem v aplikaci použil jsou:

- **appcompat-v7**
  Pomocná knihovna, která umožňuje využívat API, jež není dostupné na starší platformě.
- **Volley**
  Volley je knihovna, která je doporučená [36] pro provádění HTTP požadavků.
- **SlidingMenu**
  Jde o knihovnu, která nám usnadňuje vytváření menu pomocí přejíždění prstu zleva doprava, resp. zprava doleva.

6.3 Komunikace s Adaptérem

V této části si popíšeme jak je v aplikaci řešená komunikace s Adaptérem.

6.3.1 HttpsTrustManager

Kvůli tomu, že komunikace bude prováděna pomocí https, použil jsem třídu *HttpsTrustManager* implementující statickou metodu *allowAllSSL*, která povoluje všechny certifikáty. Metoda *allowAllSSL* je zavolána po spuštění aplikace ve třídě *AppController*. Pro ostré nasazení aplikace je doporučeno tento příkaz zakomentovat z důvodu, že slouží pouze pro zjednodušené testování aplikace.

6.3.2 Použití knihovny Volley

Instance objektu RequestQueue

Protože aplikace neustále provádí požadavky, je doporučeno [34] pro efektivnější zpracování požadavků vytvořit jednu instanci objektu *RequestQueue* v aplikačním kontextu. *RequestQueue* slouží pro přidání požadavků, které se mají vykonat.

Instanci *RequestQueue* vytvářím ve třídě *AppController*. Instanci je možné získat pomocí metody *getRequestQueue*, která nejdříve ověří, jestli je vytvořená instance. Pokud není, je tato instance vytvořena a následně předána. Pokud instance už existuje, je tato instance pouze předána. Celá metoda vypadá takto.

Zdrojový kód 6.1: Implementace metody pro získání instance RequestQueue

```
public RequestQueue getRequestQueue() {
    if (mRequestQueue == null) {
        mRequestQueue = Volley.newRequestQueue(getContext());
    }
    return mRequestQueue;
}
```

V této třídě také nalezneme metodu *addToRequestQueue*, jež přidá do instance *RequestQueue* požadavek, který se má vykonat.

Provedení požadavku

Knihovna Volley provádí požadavky asynchronně a proto nevíme, kdy dostaneme odpověď. Nejčastějším řešením je proto skutečnost, že se požadavky implementují v aktivitě, kdy se po přijetí odpovědi aktivita aktualizuje. Toto řešení se mi ale nezdálo adekvátní, neboť jsem chtěl mít oddělené třídy pro provedení požadavků. Proto jsem vymyslel způsob, jak tyto požadavky od aktivit oddělit.

Řešením bylo vytvořit rozhraní *VolleyCallback*, které obsahuje metody *onSuccess* pro úspěšný požadavek a *onError* pro neúspěšný požadavek. Třída implementující *VolleyCallback* je předávána v konstruktoru třídě, která bude vykonávat požadavek. Ve třídě se poté nacházejí metody, které provádějí požadavek. Po vykonání požadavku jsou data předána při úspěšném požadavku metodě *onSuccess* nebo při neúspěšném metodě *onError*. Jako parametr se také předává typ dotazu ze souboru *RequestType*, který obsahuje výčtový typ.

Zdrojový kód 6.2: Příklad metody pro provedení požadavku odhlášení

```
public void executeLogoutRequest() {
    String url = LOGOUT + "/" + Setting.getToken();
    JsonObjectRequest jsonRequest = new
        JsonObjectRequest(Request.Method.DELETE,
        RequestUtils.setIPAndPort(url),
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(final JSONObject response) {
                    volleyCallback.onSuccess(response,
                        RequestType.LOGOUT);
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    volleyCallback.onError(error,
                        RequestType.LOGOUT);
                }
            });

    AppController.getInstance().addToRequestQueue(jsonRequest);
}
```

Při provádění požadavku se využívají soukromé statické konstanty, které obsahují URL, kde místo IP adresy je <*ip*> a místo portu je <*port*>. URL adresa může vypadat například takto:

```
"https://<ip>:<port>/auth/login"
```

A to proto, že v průběhu aplikace se IP adresa a port mohou měnit. Před použitím URL se proto zavolá statická metoda *setIPAndPort* ze třídy *RequestUtils*, která doplní aktuálně nastavenou IP adresu a port. Celá metoda vypadá takto:

```
public static String setIPAndPort(String template) {
  template = template.replace("<ip>", Setting.getIP());
  template = template.replace("<port>", Setting.getPort() + "");

  return template;
}
```

Spuštění požadavku a zpracování dat

Každá aktivita, která chce spustit požadavek, implementuje rozhraní *VolleyCallback*. V aktivitě se proto implementuje metoda *onSuccess* a *onError*. Tyto metody zpracovávají data, která přišla po provedení požadavku.

Zdrojový kód 6.3: Příklad implementace metody onSuccess

```
public void onSuccess(Object response, RequestType type) {
  super.onSuccess(response, type);
  if (type == RequestType.ACTION_GET) {
    JSONObject res = (JSONObject) response;
    try {
      if (res.getString("status").equals("ok")) {
         setAction(
            JsonToObject.mapJsonToAction(res.getJSONObject("data")
         ));
      }
    } catch (JSONException e) { e.printStackTrace(); }
    progress.dismiss();
  }
}
```

Spuštění požadavku provedeme vytvořením instance třídy, která provádí požadavek. Do konstruktoru se předává instance aktivity, z níž se spouští požadavek. Poté je zavolána metoda, která požadavek provede.

Zdrojový kód 6.4: Spuštění požadavku pro přihlášení uživatele

```
new AuthRequest(this).executeLoginRequest(username, password);
```

6.4 Části aplikace

V této části si představíme jednotlivé části aplikace a jejich implementaci.

6.4.1 Přihlášení

Přihlášení slouží pro ověření, že daný uživatel existuje v DMS systému. Ověřovaní funguje pomocí uživatelského jména a hesla.

Přihlášení je úvodní aktivita, jež následuje po spuštění aplikace. O přihlášení se stará aktivita *LoginActivity*, jež nejdříve ověří, jestli je uživatel přihlášený. Přihlášení uživatele je ověřeno statickou metodou *getToken* ze třídy *Setting*, která vrací token uživatele. Pokud token není prázdný, je uživatel přihlášený a spustí se aktivita *WorktableActivity*. Pokud uživatel není přihlášený, zobrazí se aktivita pro přihlášení.

Zdrojový kód 6.5: Ověření, jestli je uživatel přihlášený.

```
if (!Setting.getToken().equals("")) {
   Intent intent = new Intent(LoginActivity.this,
      WorktableActivity.class);
    startActivity(intent);
}
```

Po stisknutí na tlačítko *Přihlásit se* se nejdříve zjistí, jestli je vyplněné uživatelské jméno. Pokud není uživatelské jméno vyplněné, zobrazí se toast s informací *Uživatelské jméno musí být vyplněno*. Pokud je vyplněno, zavolá se metoda *executeLoginRequest* ve třídě *AuthRequest*. Jako parametry metody se předávají uživatelské jméno a heslo. Tato metoda provede požadavek, který ověří uživatele. Pokud při požadavku nastane chyba, například

tím, že server neběží, je zobrazena pomocí toastu informace o chybě. Pokud požadavek proběhne v pořádku, zjistíme nejdříve hodnotu statusu. Pokud má status hodnotu *unauthorized*, uživatele se nepodařilo přihlásit a zobrazí se toast s informací *Neplatné uživatelské jméno nebo heslo*. Pokud má status hodnotu *ok*, je uživatel přihlášený. Po úspěšném přihlášení se uloží token pomocí metody *setToken* ze třídy *Setting*, kde se jako parametr předává token uživatele. Ukládá se zde také uživatelské jméno a heslo pomocí metod *setUsername* a *setPassword* ze třídy *Setting*. Uživatelské jméno a heslo se ukládá z toho důvodu, aby při konci platnosti tokenu mohla aplikace tento token obnovit bez vědomí uživatele. Token, uživatelské jméno a heslo se vymazávají, pokud se uživatel odhlásí z aplikace. Po uložení informací se spustí aktivita *WorktableActivity*.

Zdrojový kód 6.6: Uložení informací po úspěšném přihlášení

```
Setting.setToken(res.getString("token"));
Setting.setUsername(username);
Setting.setPassword(password);
```

Po kliknutí na nápis *Nastavení* se provede spuštění dialogu pomocí třídy *SettingDialog*.

6.4.2 Dialog nastavení

O dialog nastavení se stará třída *SettingDialog*. V dialogu je možné nastavit IP adresu serveru a port, na kterém běží aplikace Adaptér.

Při spuštění dialogu jsou nastaveny textovým políčkům uložené hodnoty IP adresy a portu ve třídě *Setting*. TextEditu, který slouží pro napsání IP adresy, je nastavena třída *IpTextWatcher*, jež kontroluje vstup od uživatele a nedovolí zadat znak, který nemůže obsahovat IP adresa. Dialog také obsahuje tlačítko s názvem *Zrušit*. Při jeho stisku se zruší všechny provedené změny tak, že se nastaví hodnoty IP adresy a portu, které byly nastaveny při spuštění dialogu. Dialog také obsahuje tlačítko Uložit. Po jeho stisknutí se nejdříve zkontroluje, jestli vstup obsahuje platnou IP adresu a jestli je port vyplněný. Pokud je všechno v pořádku, IP adresa se uloží pomocí metody *setIP* ze třídy *Setting* a port pomocí metody *setPort* ze třídy *Setting*, dialog se poté zavře.

6.4.3 Pracovní stůl

Pracovní stůl je aktivita, která slouží pro zobrazení základních informací všech úloh, které uživatel má vyřídit. O pracovní stůl se stará třída *WorktableActivity*.

Po vytvoření aktivity se zavolá metoda *executeRequest* ze třídy *WorktableRequest*. Tato metoda slouží k získání dat. Po získání dat, se nejdříve ověřuje, že uživatel je stále přihlášený. Pokud tak není, je proveden požadavek na přihlášení uživatele. Uživatelské jméno a heslo se použije takové, které bylo uloženo při přihlášení uživatele. Po obnovení tokenu je znovu odeslán požadavek na získání dat.

Po úspěšném získání dat se nejdříve namapuje JSON objekt na datové třídy. Převod se provede pomocí metody *mapJsonToTasks* ze třídy *JsonToObject*, která vrátí pole objektů *Task*. Pokud je pole délky nula, aktivita zobrazí nápis, že uživatel nemá žádné úlohy pro zpracování. Pokud je délka nenulová, pole se projde a zjistí se nejnovější datum, které se poté uloží pomocí metody *setLastDate* ze třídy *Setting*. Pokud datum nebylo nalezeno, uloží se hodnota null.

Zdrojový kód 6.7: Nalezení a uložení nejnovějšího datumu

```
Date lastDate = null;
for (Task task : tasks) {
   if (lastDate == null || task.getStartedAt().after(lastDate)) {
      lastDate = task.getStartedAt();
   }
}
Setting.setLastDate(lastDate);
```

Po nalezení nejnovějšího datumu se data předají třídě *WorktableListAdapter*, která se postará o zobrazení dat. Tato třída je nastavena jako adaptér třídě *ListView*.

Položka v *ListView* obsahuje informace a obrázek, po jehož kliknutí se zobrazí dialog s akcemi. Pokud se ale klikne na položku mimo obrázek, je spuštěna aktivita *TaskActivity*, kam se přenáší ID úlohy, která se má zobrazit.

6.4.4 Detail úlohy

Detail úlohy je aktivita, která se stará o zobrazení detailních informacích o úloze. O aktivitu se stará třída *TaskActivity*.

Při vytváření aktivity se vytvoří pravé a levé menu pomocí třídy *SlidingMenu*. Po vytvoření aktivity je zavolána metoda *executeRequest* ze třídy *TaskRequest*. Jako parametr se předává ID úlohy, která se má zobrazit. Po obdržení dat se nejdříve převede JSON objekt na datovou třídu *Task* pomocí metody *mapJsonToTask* ze třídy *JsonToObject*. Data ze třídy *Task* jsou poté zobrazena. Pokud instance *Task* neobsahuje hlavní dokument, není tento dokument zobrazen. Dokumenty, jež obsahuje úloha, jsou předány třídě *TaskLeftMenuListAdapter*, která se postará o zobrazení dokumentů v levém menu. Třídě *TaskRightMenuListAdapter* předáme instanci *Task*. Ta se postará o zobrazení položek dokumentů v pravém menu.

Po kliknutí na obrázek se třemi tečkami se spustí dialog pomocí třídy *ActionsDialog*. Pokud se klikne na dokument, je zavolána metoda *executeRequest* ve třídě *FileRequest*. Po obdržení dat je objekt přetypován na pole bytů, které se poté uloží jako dočasný soubor. Tento soubor je pak předán metodě *showFile* ze třídy *FileUtils*, která se postará o zobrazení souboru v externí aplikaci.

Zdrojový kód 6.8: Vytvoření dočastného souboru

```
File file = File.createTempFile("tmp", downloadFilename,
    getExternalCacheDir());
FileOutputStream outputStream = new FileOutputStream(file);
outputStream.write(res);
outputStream.close();
```

U všech dokumentů lze také kliknout na metadata, která spustí aktivitu *DocumentActivity*.

6.4.5 Dialog akce

Akce se provede pomocí dialogu, který se vytváří prostřednictvím třídy *ActionsDialog*. Tato třída zobrazí všechny akce, které lze nad úlohou provést. Po kliknutí na akci se nejdříve testuje, zda má akce dialog. Pokud je tomu tak, je spuštěna aktivita *ActionActivity* a její předáno ID akce. Pokud akce nemá

dialog, je zavolána metoda *executePostRequest* ve třídě *ActionRequest*, která provede akci a po úspěšném provedení je dialog zavřen.

6.4.6 Metadata dokumentu

DocumentActivity je aktivita, jež se stará o zobrazení metadat dokumentu.

Při vytváření aktivity se získá ID dokumentu, které bylo předáno při spuštění aktivity a provede se zavolání metody *executeGetRequest* ze třídy *DocumentRequest*, která provede požadavek na získání dokumentu. Metodou *mapJsonToDocument* ze třídy *JsonToObject* poté převedeme získaná data na objekt *Document*, který je následně zobrazen. *Document* obsahuje metadata. Pokud nejsou metadata prázdná, zobrazí se tlačítko pro zrušení a uložení. Po kliknutí na tlačítko pro zrušení se všechny hodnoty metadat vrátí do původní hodnoty. Pokud se klikne na tlačítko uložit, provede se kontrola zadaných hodnot a zavolá se metoda *executePostRequest* ze třídy *DocumentRequest*. Ta provede uložení.

6.4.7 Akce

Akce je aktivita, která slouží pro doplnění údajů pro provedení akce. O aktivitu se stará třída *ActionActivity*.

Při vytváření je získáno ID akce a ID úlohy, které bylo předáno při spuštění aktivity. Poté je zavolána metoda *executeGetRequest* ze třídy *ActionRequest*, která získá údaje o akci. Po získání dat jsou data převedena na objekt *Action* pomocí metody *mapJsonToAction* ze třídy *JsonToObject* a následně jsou zobrazena. V aktivitě je tlačítko Odeslat, po jeho kliknutí se nejdříve zjistí, jestli jsou všechny údaje zadány správně, pokud ano, jsou data odeslána pomocí metody *executePostRequest* ze třídy *ActionRequest*. Po potvrzení akce se spustí aktivita *WorktableActivity*.

6.4.8 Notifikace

Notifikace je služba, která se spouští pokud je uživatel přihlášený a aplikace je na pozadí nebo je zavřená. O notifikaci se stará třída *NotificationService*,

která každých 30 sekund volá metodu *executeRequest* ze třídy *NotificationRequest*. Získaná data obsahují počet nových úloh. Pokud je počet větší jak nula, je zobrazena notifikace, která upozorní uživatele na počet nových úloh. Po kliknutí na notifikaci se spustí aktivita *WorktableActivity* a služba se ukončí.

6.5 Vzhled aplikace

Třídy starající se o vzhled aplikace nalezneme v balíčku *cz.zcu.fav.kiv.activity*, který obsahuje všechny aktivity. Dále to je balíček *cz.zcu.fav.kiv.dialog*, který se stará o zobrazení dialogů. V balíčku *cz.zcu.fav.kiv.adapter* jsou třídy, které se starají o zobrazení dat např. pomocí *ListView*. Posledním balíčkem, v němž se pracuje s grafikou je *cz.zcu.fav.kiv.variable*, kde každá třída obsahuje metodu *getElement*, která vrací objekt *View*.

Vzhled jednotlivých aktivit a dialogů jsou také popsány pomocí souborů xml v adresáři res/layout.

6.5.1 Přihlášení

O vzhled přihlášení do aplikace se stará soubor *login_activity.xml*. V tomto souboru jsou vytvořeny dva *EditTexty*, které zajišťují vstup uživatelského jména a hesla. Pod těmito *EditTexty* je tlačítko, které slouží pro přihlášení.

Dole uprostřed je umístěn text *Nastavení*. Po jeho stisknutí se spustí dialog s nastavením . Aktivitu lze vidět na obrázku 6.4.

6.5.2 Dialog nastavení

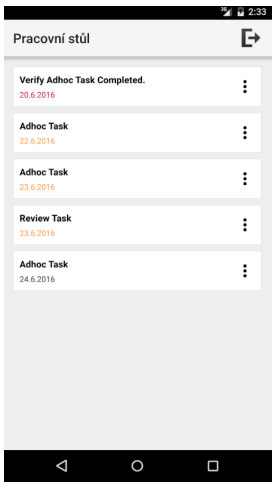
Nastavení aplikace se provedlo pomocí dialogu (Obrázek 6.5), v němž je možné zadat IP adresu serveru a port, na kterém server běží. Jsou zde také tlačítka pro zrušení a uložení. Nastavení vzhledu se provádí pomocí souboru *dialog_setting.xml*.

Obrázek 6.4: Aktivita Přihlášení

Obrázek 6.5: Dialog nastavení

6.5.3 Pracovní stůl

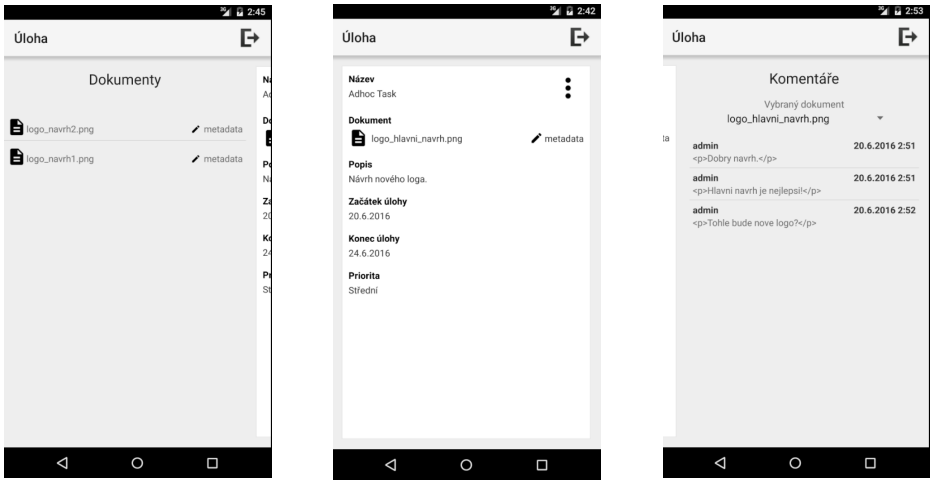
Pracovní stůl se stará o zobrazení všech úloh, které uživatel má vyřídit. U každé úlohy je zobrazen název, datum, do kdy musí být úloha vyřešena a akce, které jdou provést nad úlohou. Vzhled je nastaven v souboru *activity_worktable.xml*.



Obrázek 6.6: Aktivita Pracovní stůl

6.5.4 Detail úlohy

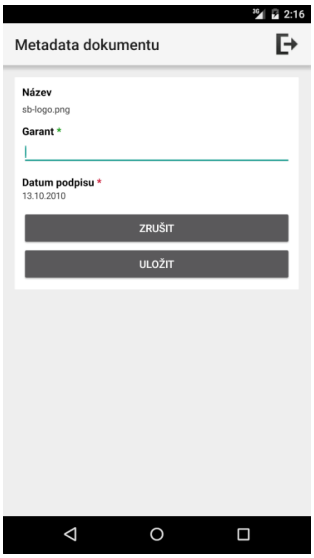
Detail úlohy zobrazuje všechny informace o úloze. O zobrazení aktivity se stará soubor *activity_task.xml*. Tato aktivita obsahuje dvě menu. První menu se vysouvá pomocí přejetí prstu zleva doprava a obsahuje všechny dokumenty v úloze, které nejsou označeny jako hlavní. Druhé menu se vysouvá přejetím prstu zprava doleva a obsahuje informace o dokumentech. O zobrazení levého menu se stará soubor *left_menu_task.xml* a o zobrazení pravého se stará soubor *right_menu_task.xml*.



Obrázek 6.7: Detail úlohy

6.5.5 Metadata dokumentu

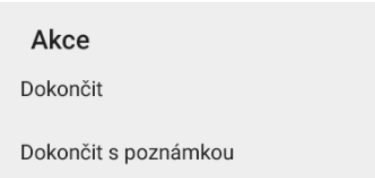
Aktivita zobrazí všechny dostupná metadata o dokumentu. Okno je rozdělené na dvě části. V první části jsou zobrazené statické informace, které se nemohou měnit. V druhé části jsou poté zobrazena metadata. Ta je možné, pokud je to dovoleno, upravit.



Obrázek 6.8: Aktivita Metadata dokumentu

6.5.6 Dialog akce

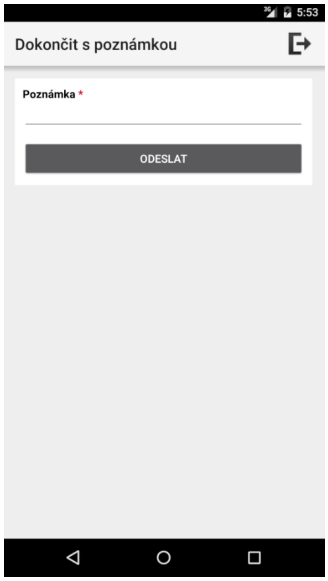
Výběr akce, která se má provést nad úlohou se realizuje pomocí dialogového okna. Dialogové okno je definované v souboru *dialog_actions.xml*. V dialogu se vypisují názvy akcí na samotné řádce.



Obrázek 6.9: Dialog s akcemi

6.5.7 Akce

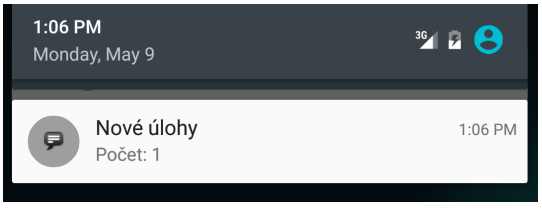
Obrazovka slouží pro zadání potřebných údajů, před odesláním akce. O zobrazení aktivity se stará soubor *activity_action.xml*. Obsah aktivity se generuje podle informací, které přijdou v JSON objektu.



Obrázek 6.10: Aktivita Akce

6.5.8 Notifikace

Notifikace se nedefinují pomocí souboru, ale využívá se vzhled, který android nabízí. V notifikaci je pouze nastaven textový výstup, který uživateli ukazuje počet nových úloh.



Obrázek 6.11: Notifikace

7 Implementace Adaptéru

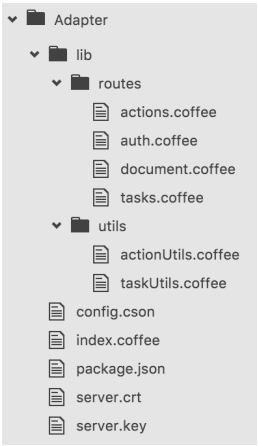
V této kapitole si popíšeme implementaci Adaptéru, který bude sloužit pro ověření funkčnosti mobilní aplikace. Aplikace Adaptér implementuje navrhnuté rozhraní v kapitole 5.4.4 a to převádí na REST API Alfresca, s kterým Adaptér pracuje.

7.1 Technologie

Adaptér je napsán v NodeJS pomocí CoffeeScriptu. Balíčky jsou stahovány pomocí nástroje npm. Jako webový server je použit balíček Express. Z důvodu, že přenos mezi mobilní aplikací a Adaptérem musí být šifrován, je pro tento účel použit balíček https.

7.2 Struktura aplikace

V hlavním adresáři projektu je soubor index.coffee, který je vstupním bodem aplikace a package.json, kde je konfigurace aplikace. Ve složce lib je backend projektu. V této složce nalezneme složku routes, která obsahuje soubory pro obsluhu požadavků a složku utils, která obsahuje soubory pro obecné použití. Dále zde nalezneme soubory server.crt a server.key, které reprezentují certifikát a slouží pro komunikaci pomocí https.



Obrázek 7.1: Struktura projektu Adaptér

7.3 Stažené balíčky pomocí npm

V aplikaci jsou použity balíčky, které jsou staženy pomocí npm a které se starají o určitou funkčnost.

- **body-parser**
  Balíček body-parser je middleware, který se stará o parsování těla požadavků
- **moment**
  Moment je javascriptová knihovna pro práci s daty. Obsahuje funkce pro parsování, validaci, manipulaci a formátovaní datumů.
- **express**
  Express je minimalistický webový aplikační framework.
- **reqwest**
  Reqwest je balíček pro vytváření asynchronních http požadavků. Použití tohoto balíčku jde vidět na níže uvedeném zdrojovém kódu 7.1.
- **xhr2**
  XMLHttpRequest emulace pro Node.js.
- **https**
  Slouží pro vytvoření serveru, který komunikuje pomocí protokolu https.

Zdrojový kód 7.1: Přiklad použití balíčku reqwest

```
reqwest
  url: process.config.ALFRESCO_SERVICE + '/login'
  type: 'json'
  method: 'post'
  contentType: 'application/json'
  data: JSON.stringify req.body
  error: (err) ->
    if err.status is 403
      res.send '{"status": "unauthorized"}'
    else
      res.send '{"status": "error"}'
    success: (resp) ->
      res.send '{"status": "ok", "token": ' + resp.data.ticket + '}'
```

7.4 Popis souborů

Zde si popíšeme soubory, které se v aplikaci nacházejí, a jejich význam. U souborů, které implementují REST API budou vyjmenované názvy požadavků z kapitoly 5.4.4, které daný soubor implementuje.

7.4.1 Hlavní složka

package.json

Soubor package.json obsahuje konfiguraci aplikace. Je zde uveden např. název aplikace, verze, vstupní soubor atd. Důležitou části jsou závislosti, kde jsou definované balíčky a jejich verze, které se stáhnou pomocí npm.

config.cson

V tomto souboru se nastavují konstanty, které se využívají v aplikaci. Je zde nastaven např. port, na kterém poběží aplikace nebo IP adresa serveru, na kterém běží Alfresco.

Celý soubor config.cson vypadá takto.

Zdrojový kód 7.2: Soubor config.cson

```
IP_ALFRESCO  = '127.0.0.1'
PORT_ALFRESCO = 8070

URL_ROOT = 'http://' + IP_ALFRESCO + ':' + PORT_ALFRESCO

PORT: 8080

ALFRESCO_CONTENT: URL_ROOT + '/alfresco/s/api/node/content'
ALFRESCO_SERVICE: URL_ROOT + '/alfresco/service/api'
ALFRESCO_API: URL_ROOT +
   '/alfresco/api/-default-/public/workflow/versions/1'
```

index.coffee

Soubor index.coffee je vstupním souborem aplikace. V tomto souboru se nejdříve načte konfigurační soubor a poté se vloží balíčky *express*, *body-parser*, *fs*, který načte certifikát a *https*. Po vložení balíčků se načte certifikát do proměnné *options*, která se poté předává při spuštění serveru. Dále je nastaven instanci *express* middleware, který bude tělo požadavků parsovat na json objekt. Poté jsou vloženy do proměnných soubory, které se starají o požadavky a ty jsou předány instanci *express* jako middleware. Nakonec je pomocí *https* spuštěný server, kde jako parametr se předává načtený certifikát a instance *expressu*. V metodě listen je předáno číslo portu. Po úspěšném spuštění se vypíše na konzoli hláška **Started!**.

Celý soubor index.coffee vypadá takto.

Zdrojový kód 7.3: Soubor index.coffee

```
require('cson-config').load()
fs         = require 'fs'
https      = require 'https'
app        = require('express')()
bodyParser = require 'body-parser'

options = {
   key  : fs.readFileSync('server.key'),
   cert : fs.readFileSync('server.crt')
}

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

actionsRoute = require './lib/routes/actions'
authRoute = require './lib/routes/auth'
tasksRoute = require './lib/routes/tasks'
documentRoute = require './lib/routes/document'

app.use '/actions', actionsRoute
app.use '/auth', authRoute
app.use '/tasks', tasksRoute
app.use '/document', documentRoute

https.createServer(options, app).listen process.config.PORT, ->
   console.log 'Started!'
```

7.4.2 Složka lib/routes/

Složka obsahuje soubory, které zpracovávají požadavky.

- **actions.coffee**
  Tento soubor implementuje požadavek *Získání akce*.
- **auth.coffee**
  V souboru jsou implementovány požadavky *Přihlášení* a *Odhlášení*.
- **document.coffee**
  Zde jsou implementovány požadavky *Získání dokumentu*, *Získání metadat dokumentu* a *Uložení metadat dokumentu*. *Získání metadat dokumentu* a *Uložení metadat dokumentu* je v Adaptéru pouze simulováno.
- **tasks.coffee**
  Soubor obsahuje implementaci požadavků *Získání úloh*, *Získání počtu nových úloh*, *Získání úlohy* a *Provedení akce nad úlohou*. Implementaci získávaní nových úloh můžeme vidět ve zdrojovém kódu 7.4.

Zdrojový kód 7.4: Implementace požadavku Získání počtu nových úloh

```
router.get '/number', (req, res) ->
  token = req.get 'token'
  date = req.get 'date'

  reqwest
      url: process.config.ALFRESCO_API+'/tasks'+'?alf_ticket='+token
      type: 'json'
      method: 'get'
      contentType: 'application/json'
      error: (err) ->
          if err.status is 401
              res.send '{"status": "unauthorized"}'
          else
              res.send '{"status": "error"}'
      success: (resp) ->
          number = 0
          resp.list.entries.forEach (d, i) ->
              if moment(date).isBefore moment(d.entry.startedAt)
                  number++

          res.send '{"status": "ok", "count": ' + number + '}'
```

7.4.3 Složka lib/utils/

- **actionUtils.coffee**
  V souboru jsou implementované dvě funkce, a to *getActions: (type)*, která převede akci z Alfresca na akce pro mobilní aplikaci a *getAction: (type)*, která podle typu akce vrátí akci s proměnnými, již jsou potřeba vyplnit před odesláním.

- **taskUtils.coffee**
  Tento soubor obsahuje pouze jednu funkci, a to *getStringPriority: (intPritority)*, která převede číselnou prioritu na string. Funkce je ukázána ve zdrojovém kódu 7.5.

Zdrojový kód 7.5: Příklad převodu int priority na string

```
getStringPriority: (intPritority) ->
       if intPritority is 1
           return 'Vysoka'
       else if intPritority is 2
           return 'Stredni'
       else
           return 'Nizka'
```

7.5 Problém

Při transformaci provedení akce na schválení a zamítnutí jsem narazil na problém, že i přesto, že do Alfresca odesílám požadavek na schválení úlohy, je tato úloha vždy zamítnuta. Tento problém vznikl buď špatným pochopením požadavku na schválení nebo jsem narazil na bug v Alfrescu. Druhá možnost je reálná, protože nejsem jediný viz diskuze [37], komu se tento problém vyskytl. Z důvodu, že Adaptér je jen pro testovací účely, více jsem tento problém neřešil.

8 Ověření aplikace

V této kapitole si nejdříve popíšeme jakým způsobem byla ověřena funkčnost implementované mobilní aplikace a následně si ověříme její použitelnost.

8.1 Ověření funkcionality

Ověření funkcionality probíhalo pomocí implementovaného Adaptéru v předešlé kapitole, který komunikoval s Alfrescem. V němž jsem si předem připravil testovací data a testovacího uživatele.

Testování aplikace probíhalo nejdříve pomocí emulátoru s Android verzí API 16 a 21. Aplikace byla poté také nainstalována a testována na mobilním zařízením Samsung Galaxy Mini s verzí Androidu 5.1.1.

Správné chování aplikace jsem testoval po aktivitách, kdy jsem se snažil nasimulovat všechny možné stavy, které mohou nastat. Abych mohl nasimulovat některé stavy, byl v průběhu testu upravován Adaptér tak, aby do mobilní aplikace odeslal data, která způsobila chování, jež jsem chtěl ověřit.

8.1.1 Příklad ověření aktivity Přihlášení

Nejdříve jsem ověřil, zda se po prvotním spuštění aplikace se zobrazí aktivita *Přihlášení*, a zda byla tato aktivita správně zobrazena. Poté jsem ověřil skutečnost, že po kliknutí na *Nastavení* se otevře dialog pro nastavení aplikace. Následně jsem se pokoušel nejdříve přihlásit pomocí nevyplněných údajů, kdy se zobrazil správně toast s informací, že uživatelské jméno musí být vyplněno. Po vyplnění údajů jsem se snažil znovu přihlásit, ale IP adresa a port nebyly správně nastaveny. Aplikace zobrazila informaci, že se nelze připojit na server. Po zadání IP adresy a portu, na němž je spuštěna aplikace Adaptér, jsem přihlášení opakoval. Uživatelské jméno ale v DMS systému nebylo, a proto aplikace zobrazila informaci o tom, že uživatelské jméno nebo heslo je spatně. Nakonec jsem zadal uživatelské jméno a heslo, které v DMS systému existovalo a pokusil jsem se přihlásit. Aplikace po chvíli zobrazila aktivitu Pracovní stůl. Nakonec jsem testoval zapamatování si uživatele tak, že jsem aplikaci vypnul bez odhlášení. Po spuštění aplikace aktivita *Přihlášení*

správně zjistila, že uživatel je už přihlášen a hned zobrazila aktivitu *Pracovní stůl*.

8.1.2 Příklad ověření notifikace uživatele

Ověření notifikace probíhala tak, že jsem si nejdříve zobrazil aktivitu *Přihlášení*. Z aktivity *Přihlášení* jsem aplikaci dal na pozadí a v Alfrescu jsem vytvořil novou úlohu. Po chvílí čekání se správně nezobrazila notifikace z důvodu, že uživatel není přihlášen. Poté jsem se přihlásil a z aktivity Pracovní stůl jsem aplikaci dal na pozadí a vytvořil v Alfrescu novou úlohu, která byla přiřazena přihlášenému uživateli. Po chvíli čekání se zobrazila notifikace, která uživatele upozorňovala na jednu novou úlohu. Poté jsem znovu vytvořil novou úlohu pro přihlášeného uživatele. Po nějaké době se číslo jedna v notifikaci změnilo na číslo dva. Po kliknutí na notifikaci se zobrazila aktivita Pracovní stůl, která už obsahovala nové úlohy. Následně jsem znova vytvořil pro přihlášeného uživatele novou úlohu. Po chvíli čekání se nezobrazila notifikace a to z důvodu, že notifikace funguje pouze, pokud je aplikace na pozadí nebo je vypnutá.

8.2 Ověření použitelnosti

Ověření použitelnosti výsledného řešení můžeme rozdělit na dvě části, a to REST API a mobilní aplikace, které lze ještě rozdělit na funkce a uživatelské rozhraní.

8.2.1 REST API

Navržené REST API je natolik obecné, že je možné do něj transformovat API některého z DMS systémů.

Příkladem může být moje implementace Adaptéru v kapitole 8, kde jsem do mnou navrhnutého REST API transformoval REST API Alfresca. Příklad transformace získání všech úloh uživatele lze vidět ve zdrojovém kódu 8.1

Tato transformace je možná proto, že v REST API byly navrhnuty pouze požadavky, které obsahuje v nějaké podobě většina dnešních DMS systémů.

Zdrojový kód 8.1: Příklad transformace úloh

```
success: (resp) ->
  tasks = []
  resp.list.entries.forEach (d, i) ->
  tasks.push {
    taskId: d.entry.id,
    name: d.entry.name,
    startedAt: d.entry.startedAt,
    dueAt: d.entry.dueAt,
    actions: actionUtils.getActions d.entry.activityDefinitionId
  }
  res.send '{"status": "ok", "data": ' + JSON.stringify(tasks) + '}'
```

8.2.2 Mobilní aplikace

Aby byla mobilní aplikace použitelná, musí obsahovat množinu funkcí, která dostačuje k používání, a uživatelské rozhraní, které lze ovládat pomocí dotekové obrazovky.

Funkce

Minimální množinou funkcí, kterou musela aplikace splňovat, aby byla použitelná, jsou:

- přihlášení pomocí uživatelského jména a hesla, které se ověřuje pomocí DMS systému,
- zobrazení všech úloh uživatele,
- zobrazení detailu úlohy,
- zobrazení dokumentu v úloze,
- provedení akce nad úlohou,
- zobrazení a úprava metadat dokumentu,
- šifrovaný přenos dat.

Všechny tyto funkce mnou implementovaná mobilní aplikace obsahuje, a proto z hlediska funkčnosti je použitelná.

Uživatelské rozhraní

Protože bude mobilní aplikace ovládána pomocí dotykové obrazovky, musí být uživatelské rozhraní přizpůsobeno tak, aby byla aplikace použitelná. Důležitou roli zde také hraje jednoduchost.

Aplikace DMSClient je naprogramována tak, aby umožňovala jednoduché ovládání pomocí prstu. Je toho docíleno díky tomu, že všechny dotekové prvky jsou dostatečně velké. Celkově je v aplikaci vytvořené uživatelské rozhraní tak, aby neobsahovalo zbytečné věci a bylo jednoduché. Pro jednoduchost uživatelského rozhraní také slouží vyjíždějící menu, které bylo v aplikaci použito.

9 Závěr

V diplomové práci jsem se nejdříve seznámil se systémem pro správu dokumentů a jejich funkcionalitou. Následně jsem popsal DMS systémy, které se používají ve firmě CCA Group a.s. nebo o nich firma uvažuje.

Po seznámení s DMS systémy jsem provedl analýzu existujících mobilních aplikací pro správu těchto systémů. Tyto aplikace jsem poté vyhodnotil a popsal jejich výhody a nevýhody.

Následně jsem se seznámil s mobilním operačním systémem a jeho druhy, které se v současné době nejvíce používají. U každého operačního systému jsem popsal základní údaje, možnosti vývoje aplikace a v jakém vývojovém prostředí se aplikace nejčastěji píšou. Poté jsem popsal základní vlastnosti distribuce aplikace pro daný systém. Nakonec jsem u každého systému popsal jeho výhody a nevýhody.

Po seznámení s operačními systémy jsem provedl analýzu a návrh mobilní aplikace, kterou jsem měl vytvořit. Z návrhu jsem poté implementoval požadovanou mobilní aplikaci pro mobilní operační systém Android. Protože mobilní aplikace nekomunikují přímo s DMS systémem, ale s Adaptérem, implementoval jsem tento Adaptér tak, abych mohl ověřit funkcionalitu mé vytvořené aplikace.

V poslední části mé diplomové práce jsem ověřil pomocí Adaptéru a Alfresca mnou vytvořenou aplikaci a také jsem zde popsal její použitelnost.

Seznam zkratek

API - Application Programming Interface
CAL - client access license
DMS - Document management system
GPS - Global Positioning System
HTTP - Hypertext Transfer Protocol
HTTPS - Hypertext Transfer Protocol Secure
ID - identifikace
IP - Internet Protocol
JSON - JavaScript Object Notation
LGPL - GNU Lesser General Public License
OCR - Optical Character Recognition
PC - personal computer
REST -Representational state transfer
SaaS - Software as a Service
SDK - Software development kit
SLA - Service Level Agreement
SSL - Secure Sockets Layer
TLS - Transport Layer Security
URL - Uniform Resource Locator

Literatura

[1] DMS: systémy pro správu a oběh dokumentů
URL: <http://www.systemonline.cz/clanky/dms-systemy-pro-spravu-a-obeh-dokumentu.htm>

[citováno: 20.5.2016]

[2] Document management systems
URL: <http://www.inflow.cz/document-management-systems>

[citováno: 20.5.2016]

[3] Kamil Kočí *Systémy pro správu dokumentů*
URL: <http://is.muni.cz/th/39540/fi_m/diplomova_prace.pdf>

[citováno: 20.5.2016]

[4] Lukáš Michálek *Návrh modulu pro správu dokumentů*
URL: <https://www.vse.cz/vskp/id/24982>

[citováno: 20.5.2016]

[5] CCA Group a.s.
URL: http://www.cca.cz/o-spolecnosti/profil-spolecnosti

[citováno: 20.5.2016]

[6] CleverDoc
URL: <http://www.cca.cz/produkty-a-sluzby/cleverdoc>

[citováno: 20.5.2016]

[7] Licencování SharePointu
URL: <https://products.office.com/cs-cz/sharepoint/sharepoint-licensing-overview>

[citováno: 20.5.2016]

[8] SharePoint
URL: <https://en.wikipedia.org/wiki/SharePoint>

[citováno: 20.5.2016]

[9] Co je SharePoint?
URL: <https://support.office.com/cs-cz/article/Co-je-SharePoint-97b91 5e6-651b-43b2-827d-fb25777f446f>

[citováno: 20.5.2016]

[10] Microsoft CAL
URL: <http://www.networkworld.com/article/2230729/microsoft-subnet/microsoft-cals-and-external-connector-licenses-part-i.html>

[citováno: 20.5.2016]

[11] Štěpán Čech *Implementace produktu Alfresco v malé firmě*
URL: <https://www.vse.cz/vskp/id/1169039>

[citováno: 20.5.2016]

[12] Alfresco
URL: <https://en.wikipedia.org/wiki/Alfresco_(software)>

[citováno: 20.5.2016]

[13] Alfresco Mobile
URL: <https://www.alfresco.com/alfresco-mobile-document-management>

[citováno: 20.5.2016]

[14] Alfresco Mobile - funkce
URL: <https://play.google.com/store/apps/details?id=org.alfresco.mobile.android.application>

[citováno: 20.5.2016]

[15] Alfresco Activiti
URL: <https://play.google.com/store/apps/details?id=com.activiti.android.app>

[citováno: 20.5.2016]

[16] Alfresco Activiti Mobile
URL: <https://github.com/Alfresco/activiti-android-app>

[citováno: 20.5.2016]

[17] SharePlus
URL: <http://www.infragistics.com/enterprise-solutions/enterprise-mobility/shareplus>

[citováno: 20.5.2016]

[18] SharePlus - SharePoint Mobile
URL: <https://play.google.com/store/apps/details?id=com.infragistics.shareplus>

[citováno: 20.5.2016]

[19] Mobile operating system
URL: <https://en.wikipedia.org/wiki/Mobile_operating_system>

[citováno: 20.5.2016]

[20] Android
URL: <https://en.wikipedia.org/wiki/Android_(operating_system)>

[citováno: 20.5.2016]

[21] Veronika Dudová *Rozvrh hodin pro mobilní zařízení*
URL: <http://www.acmspy.cz/wpcontent/uploads/2014/05/acmspy2012_submission_24.pdf>

[citováno: 20.5.2016]

[22] Android programování - Vývojové prostředí
URL: <http://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-vyvojove-prostredi/>

[citováno: 20.5.2016]

[23] Vyvíjíme pro Android: Nahraváme aplikaci na Google Play Store
URL: <https://www.zdrojak.cz/clanky/vyvijime-pro-android-nahravame-aplikaci-na-google-play-store/>

[citováno: 20.5.2016]

[24] Google Play
URL: <https://en.wikipedia.org/wiki/Google_Play#Play_Store_.28Android_application.29>

[citováno: 20.5.2016]

[25] iOS
URL: <https://en.wikipedia.org/wiki/IOS>

[citováno: 20.5.2016]

[26] Jaký operační systém a smartphone si vybrat? / Část druhá – iOS
URL: <http://www.mobilnizivot.cz/jaky-operacni-system-a-smartphone-si-vybrat-cast-druha-ios/>
[citováno: 27.3.2016]

[27] Koutek vývojáře – App Store
URL: <http://www.appliste.cz/koutek-vyvojare-app-store-cast-i/>
[citováno: 20.5.2016]

[28] Jak dostat aplikaci do libovolného iPhone
URL: <http://www.mujmac.cz/rubriky/software/programovani-pro-ios-11-jak-dostat-aplikaci-do-libovolneho-iphone-58519cz>
[citováno: 20.5.2016]

[29] Windows 10 Mobile
URL: <https://en.wikipedia.org/wiki/Windows_10_Mobile>
[citováno: 20.5.2016]

[30] Začínáme psát aplikace pro Windows Store v HTML5
URL: <https://www.zdrojak.cz/clanky/zaciname-psat-aplikace-pro-windows-store-v-html5/>
[citováno: 20.5.2016]

[31] Stopařův průvodce REST API
URL: <http://www.itnetwork.cz/nezarazene/stoparuv-pruvodce-rest-api/>
[citováno: 20.5.2016]

[32] Android Dashboards
URL: <https://developer.android.com/about/dashboards/index.html>
[citováno: 20.5.2016]

[33] HTTPS
URL: <https://www.instantssl.com/ssl-certificate-products/https.html>
[citováno: 20.5.2016]

[34] Setting Up a RequestQueue
URL: <https://developer.android.com/training/volley/requestqueue.html>
[citováno: 20.5.2016]

[35] GitHub - SlidingMenu
URL: <https://github.com/jfeinstein10/SlidingMenu>
[citováno: 20.5.2016]

[36] Transmitting Network Data Using Volley
URL: <https://developer.android.com/training/volley/index.html>
[citováno: 20.5.2016]

[37] Completing review task with REST API
URL: <https://forums.alfresco.com/forum/developer-discussions/alfresco-api/completing-review-task-rest-api-08282015-0627>
[citováno: 20.5.2016]

Přílohy

Seznam příloh

Uživatelská příručka

Uživatelská příručka

Alfresco

Pro otestování aplikace je potřeba stáhnout a nainstalovat Alfresco Community. Aplikace byla testována na verzi 5.1.0, a proto je tato verze doporučována.

Instalace

Postup nainstalování produktu Alfresco Community naleznete na těchto stránkách: `http://docs.alfresco.com/community/concepts/simpleinstalls-community-intro.html`

Adaptér

Pro spuštění programu Adaptér je potřeba nainstalovat Node.js, správce balíčků npm a CoffeeScript.

Instalace

Postup nainstalování nástrojů Node.js a npm můžete nalézt na těchto stránkách: `http://blog.nodeknockout.com/post/65463770933/how-to-install-nodejs-and-npm`

Po nainstalování Node.js a npm je ještě potřeba nainstalovat CoffeeScript. CoffeeScript nainstalujeme příkazem *npm install -g coffee-script*.

Spuštění

Složku *Adapter*, která se nachází na přiloženém CD, je potřeba zkopírovat na místo, kde je možný zápis souborů do složky. Poté je potřeba ve složce *Adapter* zadat příkaz *npm install*, který stáhne všechny potřebné balíčky. Po stáhnutí balíčků už je nutné pouze otevřít soubor *config.cson* a nastavit proměnné IP_ALFRESCO a PORT_ALFRESCO, které obsahují IP adresu a port, na kterém běží Alfresco. IP_ALFRESCO má výchozí hodnotu 127.0.0.1

a PORT_ALFRESCO má jako výchozí hodnotu 8070. Následně je ještě potřeba nastavit port, na kterém poběží aplikace Adaptér. Port se nastaví pomocí proměnné PORT, která má výchozí hodnotu 8080. Poté už stačí spustit aplikaci příkazem *coffee index.coffee*. Pokud se aplikace spustí v pořádku, zobrazí se hláška *Started!*.

DMSClient

Instalace

Nejjednodušší způsob instalace aplikace je přenesení souboru DMSClient.apk, které se nachází na přiloženém CD do mobilního telefonu. Poté musíme v telefonu povolit instalaci aplikací pomocí souboru apk. Po povolení stačí v mobilu kliknout na tento soubor a spustí se instalace aplikace. Po nainstalování aplikace uvidíme v menu ikonu s názvem DMSClient, přes kterou můžeme aplikaci spustit. Po spuštění aplikace je poté potřeba nastavit IP adresu serveru a port, na němž běží aplikace Adaptér.

První spuštění

Při prvním spuštění nebo odhlášení se z aplikace se nám zobrazí obrazovka pro přihlášení. Na této stránce se uživatel může přihlásit do aplikace zadáním uživatelského jména a hesla, které používá v DMS systému. Na této stránce také nalezneme nápis Nastavení, po jehož kliknutím se zobrazí dialog pro nastavení IP adresy serveru a portu.

Dialog nastavení

V dialogu nastavení lze zadat IP adresu serveru a port, na němž je aplikace Adaptér spuštěna. Po kliknutí na tlačítko Uložit se uloží nastavená data a dialog se zavře. Pokud chceme změny zrušit, klikneme na tlačítko Zrušit, změny se neuloží a dialog se zavře.

Obrázek 1: Přihlášení

Obrázek 2: Dialog s nastavením

Odhlášení

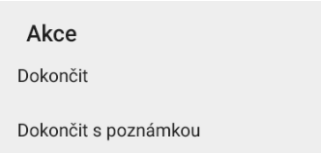
Po přihlášení do aplikace se v pravém horním roku zobrazí ikona pro odhlášení, která uživatele vrátí na obrazovku pro přihlášení.

Obrázek 3: Ikona odhlášení

Dialog s akcemi

Tento dialog se zobrazí po kliknutí na tři tečky na pracovním stole nebo v detailu úlohy. V dialogu jsou zobrazeny názvy akcí, které se dají provést.

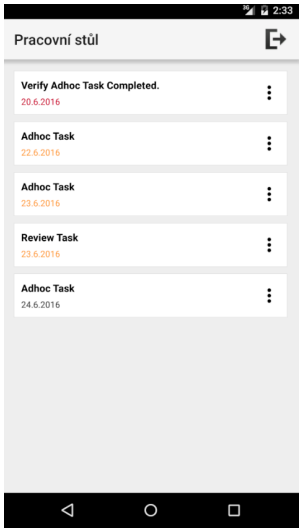
Pokud chceme provést akci, klikneme na jednu z možných položek. Po kliknutí mohou nastat dvě situace. Akce se provede hned nebo je potřeba zadat pro provedení akce dodatečné informace. Pokud se akce provede hned, je dialog zavřen a zobrazí se pracovní stůl. Pokud akce potřebuje doplnit dodatečné informace, zobrazí se obrazovka Akce.



Obrázek 4: Dialog s akcemi

Pracovní stůl

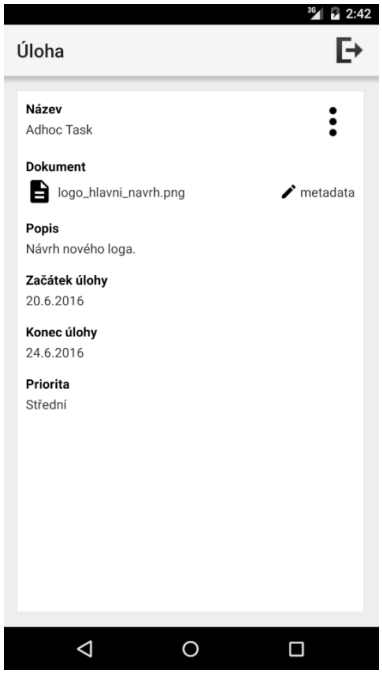
Na pracovní stůl se dostaneme po přihlášení do aplikace. Na pracovním stole jsou vidět všechny úlohy, které uživatel má vyřešit. U každé úlohy vidíme její název a datum, do kdy se má úloha vyřešit. Pokud úloha nemá určené datum, jde zde napsáno (není). Datum je také obarveno podle toho, kolik dní zbývá do dokončení. Pokud zbývá jeden den, je barva červená, pokud tři dny, je barva oranžová a pro zbylé dny je barva černá. Pomocí třech teček zobrazíme dialog pro provedení akce nad úlohou.



Obrázek 5: Pracovní stůl

Detail úlohy

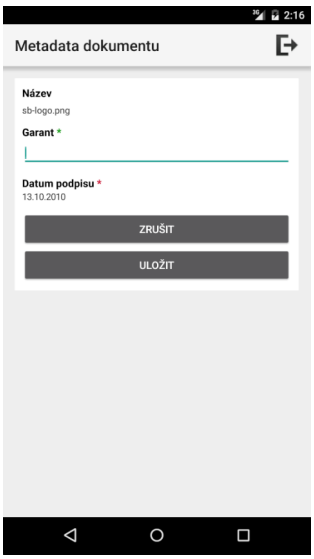
Po zobrazení detailu úlohy vidíme hned všechny informace, které o úloze máme. Můžeme zde vidět například název úlohy, popis atd. Pokud má úloha nastavený hlavní soubor, tento soubor bude zde vidět. Z této obrazovky můžeme také zobrazit levé menu přetažením prstu z levého okraje doprava. Menu obsahuje dokumenty, které nejsou označeny jako hlavní. Na obrazovce je také pravé menu, které se otevře přetažením prstu z pravého okraje doleva. Pravé menu obsahuje položky od dokumentu. Po kliknutí na dokument se tento dokument zobrazí v externí aplikaci. Po kliknutí na metadata se zobrazí obrazovka Metadata.



Obrázek 6: Detail úlohy

Metadata dokumentu

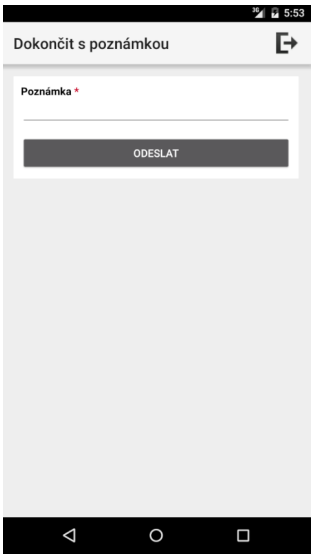
Obrazovka obsahuje metadata o úloze, pokud metadata obsahují měnitelnou položku, jsou zde zobrazeny tlačítka Zrušit a Uložit. Tlačítko Zrušit vrátí všechny provedené změny. Po kliknutí na tlačítko Uložit se metadata uloží a už poté nejdou vrátit pomocí tlačítka Zrušit.



Obrázek 7: Metadata dokumentu

Akce

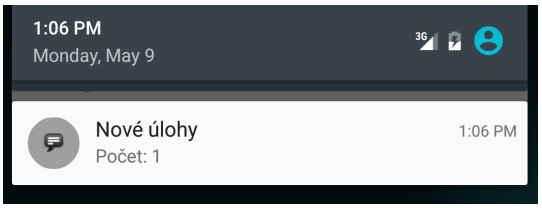
Tato obrazovka se zobrazí, pokud jsou dodatečné informace k akci. Pokud je nutné informaci před odesláním vyplnit, je u jejího názvu červená hvězdička. Pokud chceme akci provést, klikneme na tlačítko Odeslat. Po kliknutí na tlačítko se provede akce a zobrazí se Pracovní stůl.



Obrázek 8: Doplnění údajů před odesláním akce

Notifikace

Pokud je uživatel v aplikaci přihlášený a aplikace je na pozadí nebo je vypnutá, uživatel dostává notifikace o počtu nových úloh. Po kliknutí na notifikaci se zobrazí pracovní stůl.



Obrázek 9: Notifikace