FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace pro zaměstnance ČSOB
Student:	Viktor Müller
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Vytvořte mobilní aplikaci pro zaměstnance Československé obchodní banky, a.s.
Aplikace bude především klást dotazy na interní vyhledávač a zobrazovat strukturované výsledky v uživatelsky přívětivých modulech. Implementujte minimálně tři takovéto moduly (například pro zobrazení dokumentů ve formátu pdf, pro zobrazování chronologicky řazených dat, pro zobrazování výsledků vyhledávání včetně rychlého náhledu). Zajistěte, aby aplikace reagovala na oznámení od serveru, jako je například "Požární cvičení". V případě, že aplikace nebude mít přístup k bankovnímu intranetu, zobrazte alespoň výsledky uložené z dřívějšího vyhledávání. Uložené informace ukládejte do lokálního úložiště s řízeným přístupem. Uživatele autentizujte proti serveru. Důraz klaďte na rozšiřitelnost modulů pro výsledky hledání a další funkcionality mobilní aplikace. Aplikaci vytvořte pro operační systém Android. Výsledkem bude otestovaná a zdokumentovaná aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 10. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Mobilní aplikace pro zaměstnance ČSOB

Viktor Müller

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

13. května 2019

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Miroslavu Balíkovi, Ph.D. za čas strávený kontrolou této práce. Dále bych chtěl poděkovat Mgr. Petru Kutálkovi za poskytnutí mnoha užitečných informací ze strany ČSOB.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2019 .....

Odkaz na tuto práci

Müller, Viktor. *Mobilní aplikace pro zaměstnance ČSOB*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

V této bakalářské práci je popsána softwarová analýza, návrh a implementace mobilní aplikace pro operační systém Android v jazyce Kotlin. Práce se dále zabývá popisem použitých technologií pro vývoj a testování Android aplikace.

Výsledkem práce je mobilní aplikace pro zaměstnance Československé obchodní banky, a.s., pomocí které lze pohodlně vyhledávat interní informace napříč bankovními systémy. Aplikace je připravena pro integrační testování se skutečnými bankovními servery.

Klíčová slova mobilní aplikace pro zaměstnance, Android, vyhledávání v interních zdrojích, Československá obchodní banka a.s., Kotlin

Abstract

This bachelor thesis describes software analysis, design and implementation of a mobile aplication for Android OS written in Kotlin. The thesis also examines technologies used for Android application development and testing.

The result of this thesis is a mobile application for employees of Československá obchodní banka, a.s. The application provides a comfortable way to search for internal information throughout the bank's systems. The resulting application is ready for integration testing with real servers of the bank.

Keywords mobile application for employees, Android, internal data search, Československá obchodní banka a.s., Kotlin

Obsah

Seznam obrázků

Seznam tabulek

Ukázky kódu

Úvod

Československá obchodní banka, a.s. (dále ČSOB) je korporátní společnost s desítkami různých interních systémů, ve kterých může být často složité se zorientovat.

Z tohoto důvodu je v bance vyvíjen software pro efektivní vyhledávání napříč těmito systémy.

Jelikož má každý zaměstnanec ČSOB k dispozici pracovní mobilní zařízení s operačním systémem Android, nabízí se vytvoření mobilní aplikace, která bude sloužit jako klientská aplikace k již zmíněnému vyhledávači.

Tato bakalářská práce se zabývá vývojem právě této mobilní aplikace pro Android.

Rešeršní část práce je zaměřena na seznámení se s bankovním softwarem pro vyhledávání, softwarovou analýzou požadavků na mobilní aplikaci a analýzou aktuálních technologií pro vývoj aplikací pro operační systém Android.

V praktické části je popsán softwarový návrh a samotná implementace této mobilní aplikace.

Výsledná mobilní aplikace má za cíl svou jednoduchostí a přívětivým uživatelským rozhraním zjednodušit zaměstnancům vyhledávání interních informací napříč bankovními systémy.

Kapitola 1

Cíl práce

Cílem práce je vytvoření mobilní aplikace pro zaměstnance ČSOB. Tato aplikace bude klást dotazy na bankovní vyhledávací server a zobrazovat jejich výsledky na uživatelsky přívětivých obrazovkách. Aplikace má poskytovat zaměstnancům jednodušší způsob získávání informací z mnoha bankovních systémů pomocí rychlého odpovídání na textové dotazy.

K dosažení tohoto cíle je nejprve potřeba seznámit se s aplikačním rozhraním bankovního vyhledávače, vytvořit softwarovou analýzu požadavků od ČSOB a analyzovat vývoj mobilní aplikace pro Android s využitím aktuálních technologií. Aplikace musí být navržena tak, aby ji bylo možné snadno rozšiřovat o další typy výsledků vyhledávání, či o další funkcionality zaměstnanecké aplikace.

Následuje samotné vytvoření aplikace podle softwarového návrhu a zdokonalování funkcionalit aplikace podle zpětné vazby uživatelů.

Veškeré části aplikace je nutné řádně otestovat. Aplikace bude zpracovávat data o zaměstnancích ČSOB a uchovávat informace z interních bankovních systémů, proto musí být kladen důraz na testování bezpečnosti aplikace.

Kapitola 2

Informační systém pro interní bankovní vyhledávání

V této kapitole je popsán zaměstnanecký vyhledávací systém, se kterým bude mobilní aplikace komunikovat pomocí aplikačního rozhraní.

Informace o tomto bankovním systému byly čerpány z interní emailové a osobní komunikace [1], [2], [3].

2.1 Projekt

Počínaje rokem 2018 začal být vyvíjen v ČSOB systém pro vyhledávání nad skupinou databází obsahující interní data o společnosti a jejích zaměstnancích.

Projekt má dosáhnout větší přehlednosti mezi bankovními systémy, a tím šetřit čas zaměstnancům, kteří se budou moci lépe soustředit na svou práci.

2.2 Komponenty projektu

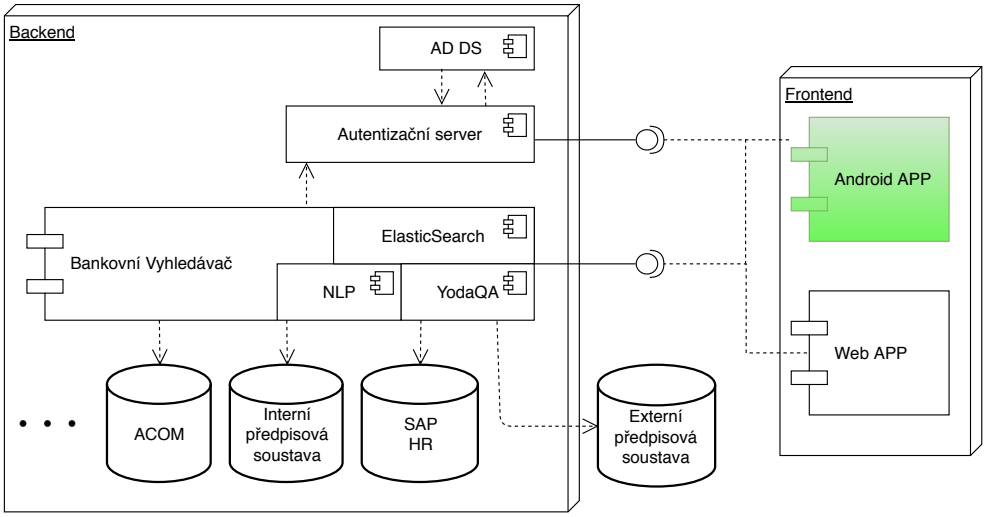
Systém se skládá ze dvou vrstev.

- Backendová vrstva

  Backendová vrstva má za úkol pochopit záměr z vyhledávaného řetězce a následně efektivně vyhledávat nad velkým množstvím dat v interních databázích. Tato vrstva poskytuje aplikační rozhraní, které lze využívat ve frontendových aplikacích.

- Frontendová vrstva

  Frontendová vrstva poskytuje zaměstnancům uživatelské rozhraní, pomocí kterého mohou toto vyhledávání používat. První frontendovou aplikací je vyhledávání na stránkách bankovního intranetu. Druhou možností, jak využívat backendový server, je pomocí mobilní aplikace, jejíž vývoj je popsán v této práci.



Obrázek 2.1: Jednotlivé komponenty interního bankovního vyhledávače

2.3 Použité technologie na interním serveru

2.3.1 Elasticsearch

Elasticsearch je vyhledávací software založený na knihovně Apache Lucene. Mezi hlavní výhody tohoto softwaru patří open-source licence, webové aplikační rozhraní, práce s volně strukturovanými JSON dokumenty a dobrá přizpůsobivost pro velikost projektu. [4]

Díky těmto vlastnostem je Elasticsearch nejpopulárnější vyhledávací nástroj pro korporátní společnosti podle hodnocení DB-Engines. [5]

2.3.2 YodaQA

YodaQA je systém pro odpovídání na otázky pokládané v přirozením jazyce [6]. Činí tak pomocí účinných metod pro extrakci konkrétních informací z textu a chápání významu zadané věty (Natural Language Understanding).

Projekt YodaQA je opět postaven na open-source licenci. [6]

2.3.3 NLP komponenty

NLP (Natural language processing) komponenty slouží k rozpoznání záměru obsaženého ve vstupní větě napsané v přirozeném jazyce.

Spell check Slouží k opravení chyb ve vstupní větě, jako je například doplnění interpunkce nebo náhrada jednoho až dvou znaků.

Lemmatizace Úkolem lemmatizace je převedení slov na základní tvar slov (slovníkový tvar). [7]

Vektorizace Převedení slov na číselné vektory v prostoru pro algoritmy strojového učení. [8]

2.4 Zdroje dat

V následující sekci jsou popsány vybrané aplikace a databáze, nad kterými systém vyhledává.

2.4.1 SAP HR

ERP (Enterprise Resource Planning) je zkratka pro podnikový informační systém navržený k integraci a optimalizaci byznysových procesů a transakcí napříč korporací. [9]

V ČSOB se pouzívá ERP systém SAP. Jedním z modulů systému SAP je modul SAP HR. Ten je zaměřený na správu lidských zdrojů, zejména evidenci zaměstnanců, jejich pracovních kontraktů, úvazků a docházky, mzdové agendy. [2]

2.4.2 ACOM

„Aplikace ACOM slouží k evidenci entit z různých oblastí a vazeb mezi nimi. Skládá se z několika modulů zaměřených na jednotlivé agendy.“ [2]

Hlavním modulem je *Organizace*. Tento modul slouží ke správě osob, útvarů, budov, poboček a místností. [2]

Další důležitý modul je *Hardware*, který spravuje informace o veškerém přiřazeném hardwaru v bance. [2]

Příklady dalších modulů aplikace ACOM jsou *Servery* (správa serverových zařízení), *Infrastruktura* (evidence kabelových vazeb, telefonních linek a síťových segmentů) nebo *Provoz* (evidence událostí v provozu a správa certifikátů). [2]

2.4.3 AD DS

Služba Microst Active Directory Domain Services slouží k uchovávání objektů v adresářové struktuře a poskytování přístupu k těmto objektům uživatelům a administrátorům sítě. [10]

V ČSOB tato služba umožňuje přiřazovat uživatelům určité role a oprávnění napříč celým systémem. Tyto role jsou následně ověřovány autentizačními servery bankovních aplikací. [3]

2.4.4 Autentizační server

Tato kapitola vychází z [3].

Jedná se o server implementující protokol OAuth2, který je blíže popsán v kapitole 4.4.2. Tento server poskytuje klientským aplikacím potřebné autentizační a autorizační tokeny pro bezpečné využívání služeb bankovního vyhledávače.

Autentizační server komunikuje se službou AD DS pokaždé, kdy je třeba ověřit prvotní oprávnění uživatele v systému. Obrácená komunikace je možná v případě, že byly uživateli změněny role v Active Directory. Pak je nutné zrušit platnosti přístupových tokenů udržovaných na autentizačním serveru.

Kapitola 3

Analýza

Tato kapitola se zabývá analýzou stávajícího řešení, průzkumu uživatelských rozhraní úspěšných vyhledávacích aplikací pro Android a softwarovou analýzou požadavků a procesů pro zadanou aplikaci.

3.1 Stávající řešení

Informace o stávajících řešeních jsou brány z intranetu ČSOB [11], [12].

Zaměstnanci banky mají v tuto chvíli k dispozici několik způsobů, jak vyhledávat informace. Mezi následujícími systémy není zahrnutý informační systém pro interní bankovní vyhledávání z předchozí kapitoly, který je v době zpracování této práce stále ve vývoji.

3.1.1 Vyhledávací centrála

Vyhledávání v interních a externích informačních zdrojích lze pomocí nástroje „Vyhledávací centrála“ na hlavní stránce intranetu.

Zejména kvůli starému uživatelskému rozhraní určenému pro internetový prohlížeč Internet Explorer 8 se nejedná o uživatelsky přívětivý způsob zjišťování informací. Snímek obrazovky s výsledky vyhledávání je přítomný v příloze na obrázku A.2.

3.1.2 Telefonní seznam

Hledání ostatních zaměstnanců je možné v nástroji „Telefonní seznam“. Ten poskytuje očekávané detailní informace o zaměstnancích, a je tak v bance velmi používanou aplikací.

Nevýhodou této aplikace je však způsob zadávání hledaného výrazu. Aplikace vrátí výsledky pouze pro přesnou shodu se zadaným řetězcem. Chyba v zapsání jména zaměstnance tedy vede k absenci této osoby ve výsledcích vyhledávání.

Zastaralé uživatelské rozhraní pro prohlížeč Internet Explorer rovněž nepatří mezi silné stránky aplikace Telefonní seznam. Vzhled aplikace je zachycen v příloze na obrázku A.3.

3.1.3 Chatboti

Znatelně novější nástroj pro získávání informací jsou chatboti odpovídající na dotazy zaměstnanců.

Příkladem takového automatického chatu je HR chatbot Viki, který odpovídá na dotazy týkající se oblasti lidských zdrojů.

Uživatelské rozhraní chatbotů je naprostým opakem již zmíněných vyhledávačů. Zaměstnanec má možnost zadat libovolnou otázku, třeba i ve větách, na kterou se chatbot snaží odpovědět pomocí pochopení záměru dané otázky.

Každý chatbot musí mít však zadefinovaný seznam odpovědí, které může nabídnout. Znalosti chatbota jsou tedy vyhrazeny pouze nad daty jednoho konkrétního typu (například často kladené dotazy na HR oddělení).

V příloze na obrázku A.1 je snímek obrazovky s uživatelským rozhraním chatbota.

3.2 Existující aplikace pro Android

V této kapitole jsou popsány vybrané Android aplikace pro interakci s interními systémy společnosti.

3.2.1 Jostle

Android aplikace Jostle představuje klientskou aplikaci pro serverovou část Jostle intranet a poskytuje následující funkcionality:

- vyhledávání a sdílení interních zdrojů,
- přehled interního zpravodajství společnosti,
- vyhledávání kolegů a interakce s kolegy pomocí přímých zpráv nebo diskuzních fór,
- organizace týmů v rámci společnosti. [13]

Možnosti aplikace Jostle tedy značně převyšují požadavky na aplikaci vyvíjenou v rámci této práce. Mezi společné funkcionality patří hlavně vyhledávání interních zdrojů a kolegů.

Cena řešení Jostle se v době psaní této práce pohybuje okolo 2 USD za každého zaměstnance za měsíc. Pro firmu s více než 3000 zaměstnanci je cena řešena individuálně. [14]

Použití aplikace Jostle vyžaduje kompletní výměnu bankovního intranetu. Takové řešení může být vhodné pro začínající společnosti s několika desítkami či stovky zaměstnanců. Kompletní změna intranetu společnosti ČSOB však nepřipadá v úvahu.

3.2.2 Microsoft SharePoint

Mobilní aplikace SharePoint je další způsob, jak se dostat k interním informacím společnosti. Možnosti aplikace SharePoint jsou:

- navigace napříč portálem Microsoft SharePoint,
- vyhledávání a zobrazování profilů kolegů,
- vyhledávání informací napříč systémy společnosti Microsoft. [15]

Android aplikace SharePoint představuje klientskou aplikaci komunikující se sadou interních aplikací společnosti Microsoft, jako je Office 365 Entrerprise, SharePoint Server 2013 a SharePoint 2016. [15]

Data, se kterými mobilní aplikace Microsoft SharePoint pracuje, jsou tedy omezena na zmíněné zdroje.

Vyvíjená aplikace v této práci komunikuje s bankovním vyhledávacím serverem, který by měl být schopen vyhledávat informace z většího množství interních zdrojů, a tím poskytovat zaměstnancům relevantnější informace.

3.3 Cílová skupina

Uživatelé Android aplikace mohou být všichni zaměstnanci, kteří mají přidělené pracovní mobilní zařízení. Z výroční zprávy roku 2017 je patrné, že Československá obchodní banka, a.s. zaměstnává bezmála 8300 zaměstnanců [16].

Dle [17] je nejčastější zaměstnanecké zařízení Samsung Galaxy J5. Další používaná zařízení jsou Samsung Galaxy J3 a Samsung Galaxy A5 (2016). Všechna tato mobilní zaměstnanecká zařízení používají operační systém Android. Nejstarší verze operačního systému pro zařízení Samsung Galaxy A5 (2016) je Android 5.1 [18]. Podle tohoto minima je zvolena minimální podporovaná verze systému pro vyvíjenou aplikaci, tedy Android 5.1 Lollipop (API level 22).

Ze zmíněných aplikací v kapitole o stávajících řešeních je nejpoužívanější interní bankovní aplikace telefonní seznam. Ta zaznamenává průměrně 100 000 vyhledávání měsíčně [1]. Vyvíjená aplikace obstarává požadavky i na další bankovní aplikace. Lze tedy předpokládat, že počet využití vyhledávání v bankovní aplikaci bude ještě vyšší.

3.4 Softwarová analýza

V následující kapitole je popsána softwarová analýza podle požadavků od společnosti ČSOB.

3.4.1 Požadavky

Požadavky na Android aplikaci jsou rozděleny do dvou skupin.

Funkční požadavky popisují, jaké funkcionality bude aplikace svým uživatelům umožňovat.

Nefunkční požadavky určují omezení kladená na systém a zajištění bezpečnosti při používání funkcionalit aplikace.

3.4.1.1 Funkční požadavky

F1: Vyhledání informací Mobilní aplikace dokáže klást požadavky na bankovní server a zobrazovat různé typy výsledků.

F2: Upozornění Aplikace bude reagovat na upozornění, která rozesílá bankovní server, jako jsou například mimořádné zaměstnanecké události, pravidelné zprávy z pracovního prostředí ČSOB nebo varovná upozornění pro zaměstnance.

Tato upozornění lze uživatelem zobrazit, odstranit, či úplně blokovat.

F3: Historie Uživatel si bude moci zobrazit své poslední navštívené výsledky vyhledávání. Při vyhledávání budou minulé dotazy použity k našeptávání vyhledávaného řetězce. V případě, že aplikace nebude mít přístup k síti, bude alespoň vracet výsledky předchozích vyhledávání.

Dále si bude možné zobrazit historii příchozích upozornění.

F4: Přihlašování Při prvotním spuštění aplikace se uživatel přihlásí svým ČSOB identifikátorem a heslem. Od této chvíle bude přihlášen, dokud mu neskončí platnost přihlášení na serveru, nebo dokud si nezruší účet ve správě účtů v zařízení.

Před přihlášením jiného uživatele do aplikace na stejném zařízení musejí být smazána všechna historická data o posledním uživateli.

3.4.1.2 Nefunkční požadavky

N1: Verze Mobilní aplikace bude spustitelná na operačním systému Android, a to od verze 5.1 (API level 22). Výběr verze vychází z minimální verze operačního systému na zařízeních zaměstnanců ČSOB. Pokrytí cílové skupiny zařízení tedy bude 100%.

N2: Bezpečnost a autentizace Aplikace zaručuje bezpečné uložení dat na lokálním úložišti tak, aby k nim měla přístup pouze tato aplikace. Opakovaná autentizace při spuštění aplikace se provádí pomocí autentizačního tokenu, proto si aplikace nebude uchovávat heslo uživatele v jakékoli podobě.

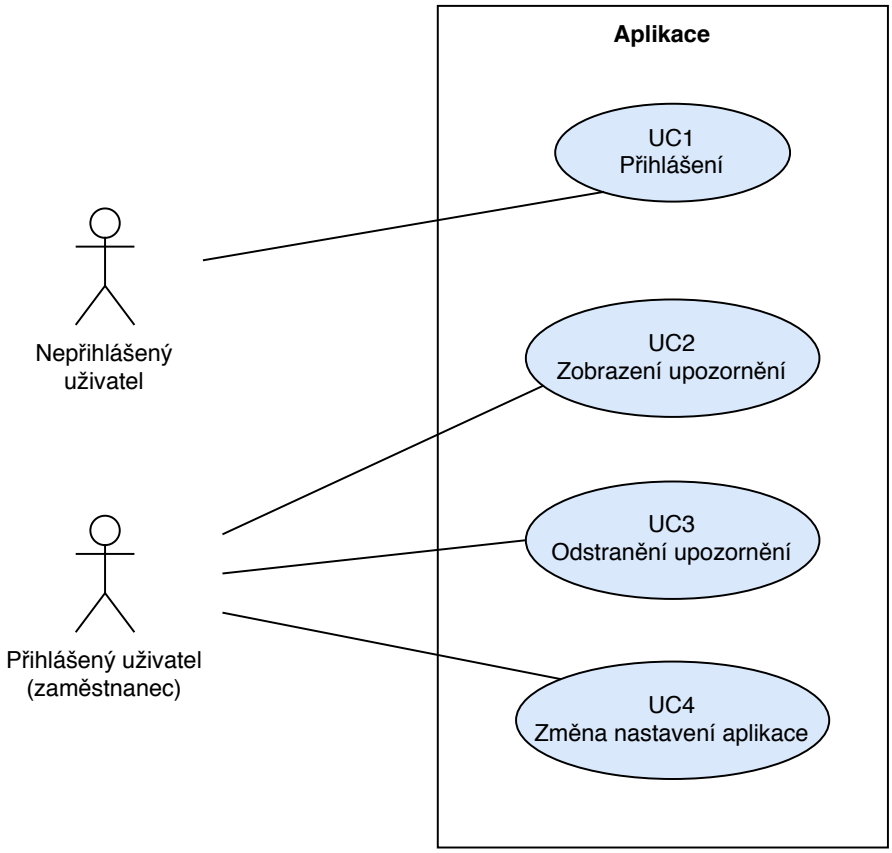
Komunikace s bankovním serverem je realizována pouze přes zabezpečenou komunikaci pomocí HTTPs.

N3: Přístup k síti Mezi oprávnění aplikace bude patřit přístup k síti. Síť reprezentuje jak Internet, tak bankovní intranet.

3.4.2 Případy užití

3.4.2.1 Obecné

Obecné případy užití zahrnují takové případy, které se netýkají vyhledávání v aplikaci. Diagram těchto případů užití je znázorněn na obrázku 3.1.



Obrázek 3.1: Diagram případů užití netýkajících se vyhledávání

UC1: Přihlášení Jediným případem užití nepřihlášeného uživatele je Přihlášení. Tento případ nastane ihned po prvotním spuštění mobilní aplikace.

Nejprve se aplikace zeptá komponenty spravující uživatelské účty v zařízení. Pokud nebude nalezen v zařízení uživatelský účet pro tuto aplikaci, bude uživatel vyzván k poskytnutí přihlašovacích údajů (zaměstnanecké ID a heslo). Po odeslání validních údajů se zaregistruje zaměstnanecké ID a obdržený autentizační token do nového systémového účtu pro tuto aplikaci. V opačném případě se zaměstnanecké ID a autentizační token nahraje z aktuálního účtu v zařízení.

Proces popisující přihlášení zobrazuje diagram 3.2.

UC2: Zobrazení upozornění Aplikace bude reagovat na obdržení upozornění od serveru tím, že zobrazí zprávu do horního panelu s upozorněními.

Po stisknutí upozornění v panelu je uživatel přesměrován na obrazovku aplikace zobrazující upozornění.

UC3: Odstranění upozornění Uživatel bude mít možnost odstranit vybraná upozornění na obrazovce s upozorněními.

UC4: Změna nastavení aplikace Aplikace umožňuje uživateli měnit její nastavení. Konkrétně automaticky mazat historii svého vyhledávání nebo vypnout a zapnout přijímání upozornění od vzdáleného serveru.

3.4.2.2 Vyhledávání

Případy užití týkající se vyhledávání zobrazuje obrázek 3.3. Předpokladem těchto případů je již přihlášený uživatel, tedy zaměstnanec.

UC5: Vyhledávání informací Hlavním případem užití celé aplikace je vyhledávání informací.

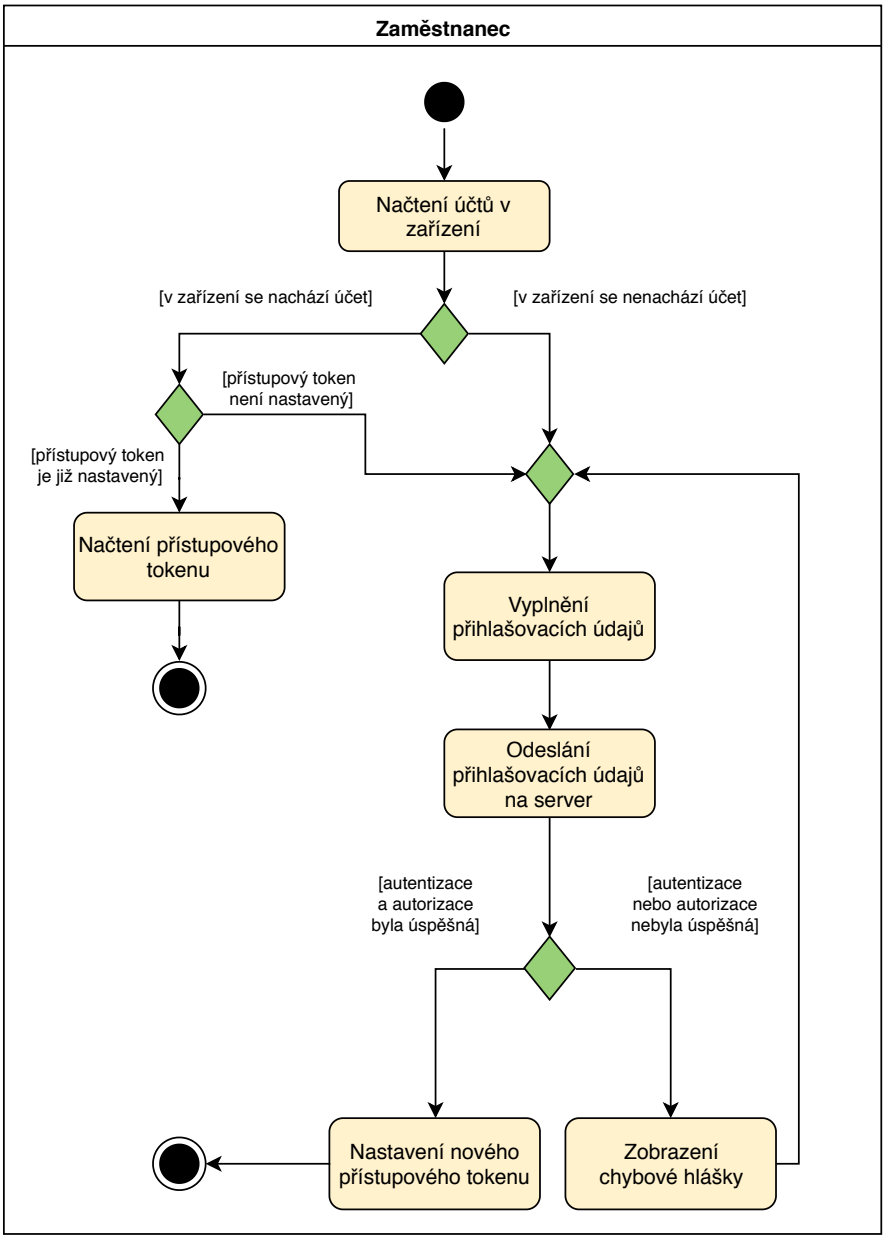
Aplikace zašle dotaz od uživatele na bankovní server. Ten vrátí odpověď s výsledky vyhledávání ve formátu JSON.

Pokud aplikace nedostane odpověď od serveru, například kvůli výpadku spojení, zobrazí uživateli chybovou hlášku a možnost opakovaného odeslání dotazu.

Proces celého vyhledávání je zobrazen na diagramu 3.4.

UC6: Zobrazení historie Při vyplňování textového pole pro dotaz k vyhledání našeptává aplikace uživateli podobné předchozí dotazy. Pokud uživatel vybere jeden z těchto historických dotazů, není potřeba klást dotaz na bankovní server, jelikož bude výsledkem již známý záznam v lokání databázi.

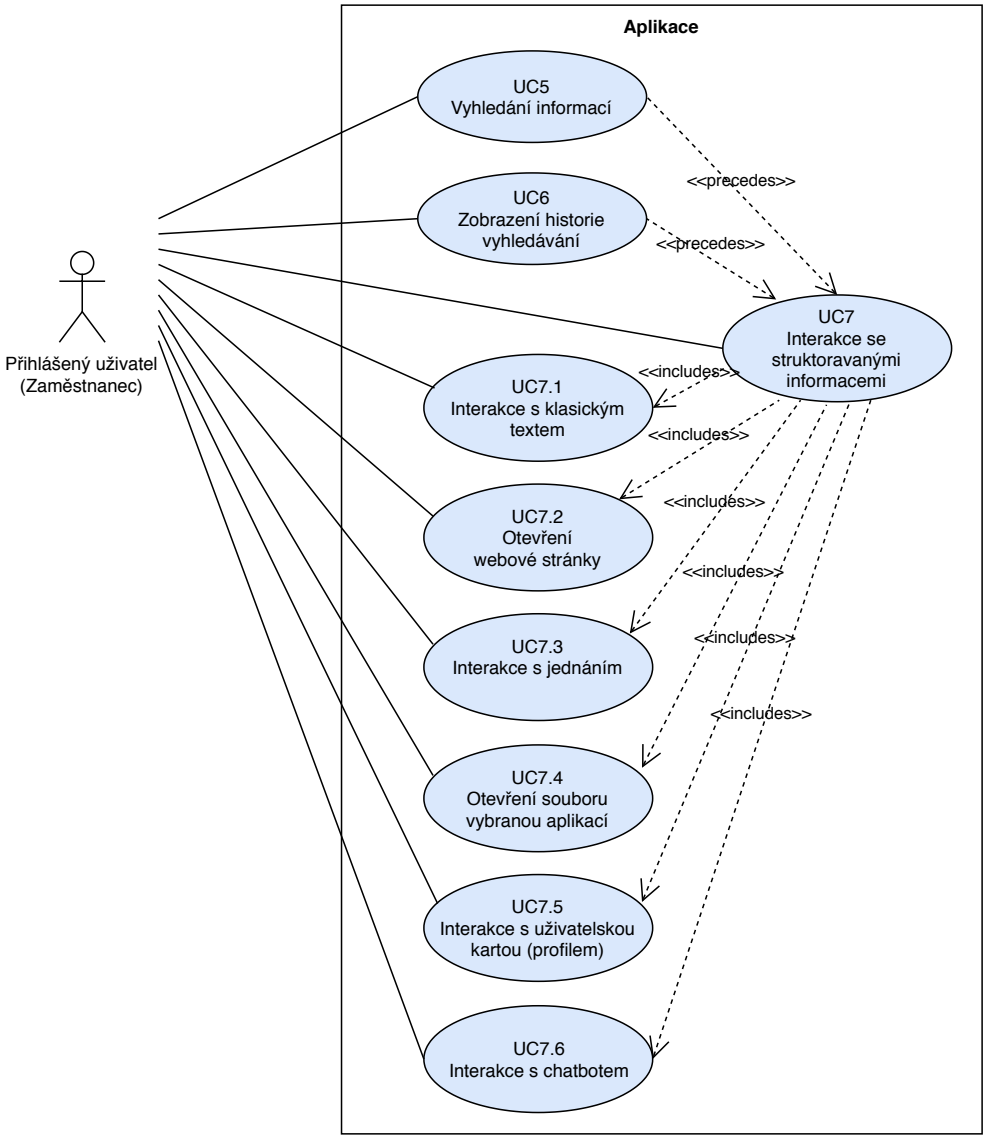
Historii vyhledávání je možné zobrazit i na své vlastní obrazovce.



Obrázek 3.2: Diagram procesu přihlášení uživatele

UC7: Interakce s výsledky vyhledávání Po úspěšném vyhledání informací se uživateli zobrazí jednotlivé výsledky v graficky odlišných záznamech.

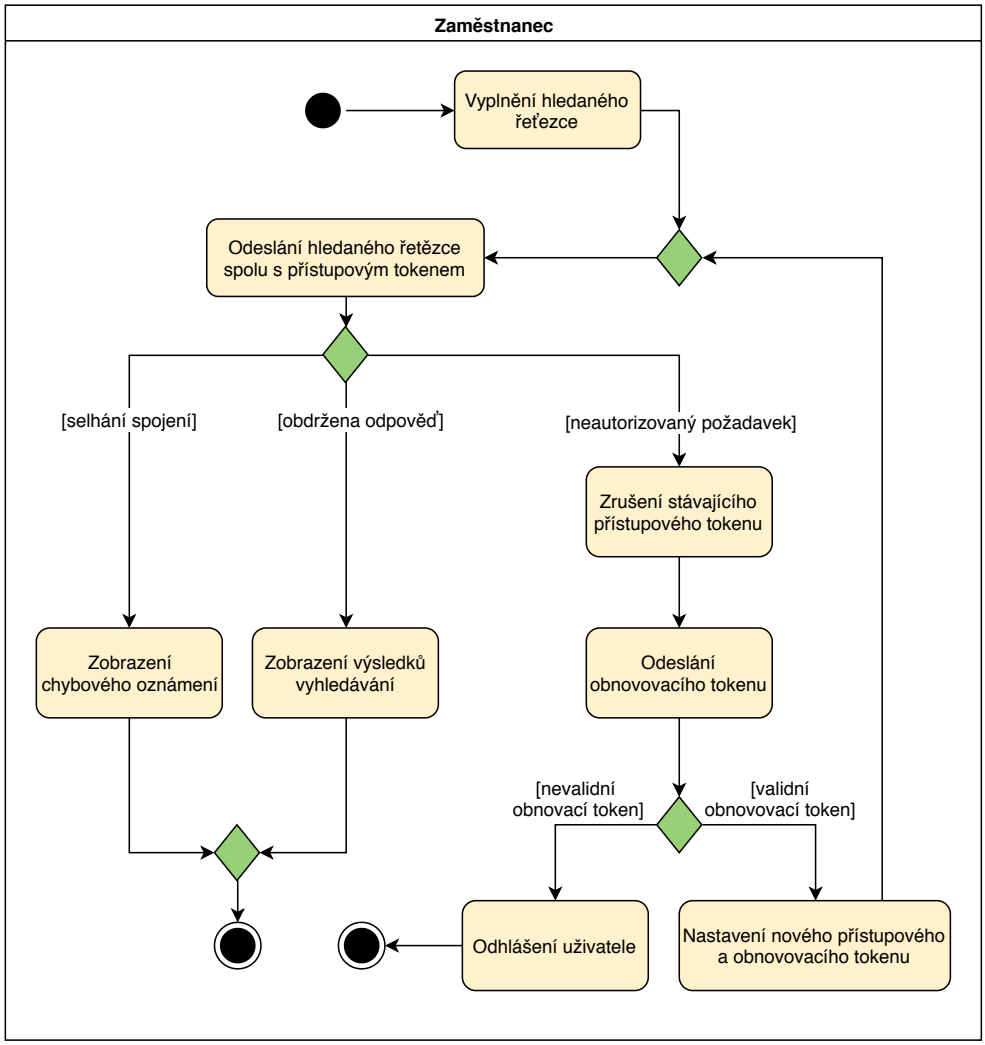
UC7.1: Klasický text Nejobecnějším záznamem je klasický text. Ten se skládá z krátkého úryvku popisujícím obsah textu (určuje backendový server) a samotného těla textu. V seznamu vyhledaných výsledků je zobrazen pouze



Obrázek 3.3: Diagram případů užití spojených s vyhledáváním

krátký úryvek. Po zobrazení detailu výsledku si může uživatel přečíst celý text.

UC7.2: Odkaz na webovou stránku Velmi častým výsledkem bude odkaz na webovou stránku. Uživateli bude zobrazen nadpis webové stránky a URL adresa pro její zobrazení. Po zobrazení detailu tohoto výsledky bude stránka otevřena ve výchozím prohlížeči pro dané zařízení.



Obrázek 3.4: Proces získání výsledků vyhledávání od bankovního serveru

UC7.3: Jednání Možným výsledkem je jednání (schůzka). Poskytnuté informace jsou místo a čas konání tohoto jednání. Aplikace bude umožňovat přidání události do vybraného kalendáře v zařízení.

UC7.4: Odkaz na soubor Podobně jako u webové stránky se ihned zobrazí název a URL adresa souboru. Po zobrazení detailu je uživatel vyzván k výběru aplikace, ve které si přeje soubor zobrazit.

UC7.5: Kontaktní karta zaměstnance V případě výsledku typu „Kontaktní karta zaměstnance“ je uživateli zobrazeno profilové foto, jméno,

příjmení, tituly a email hledaného zaměstnance.

Po zobrazení detailu budou poskytnuty detailnější informace o zaměstnanci, jako je například popis jeho pracovního místa, adresa pracoviště nebo název útvaru, ve kterém působí. Dále se zobrazí tlačítko zahájení telefonické komunikace.

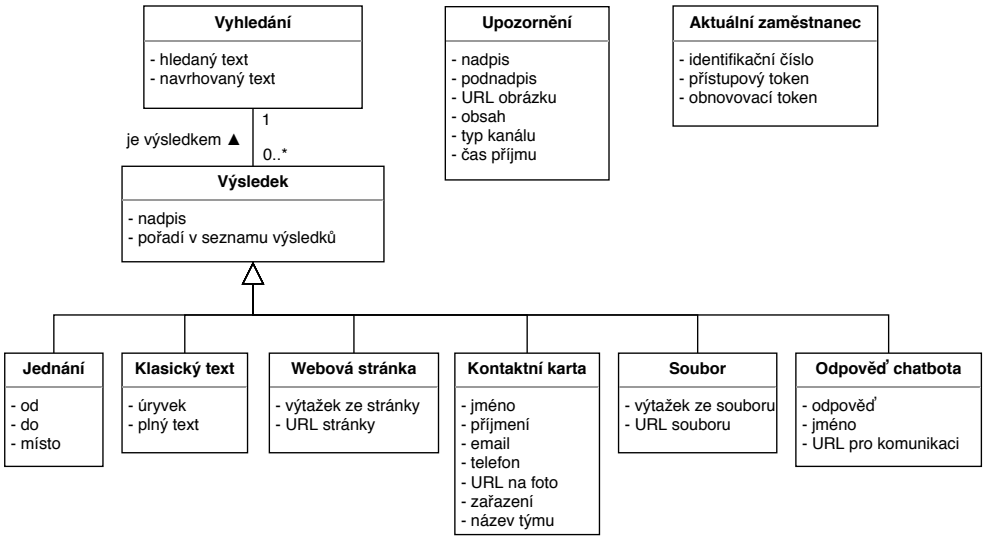
UC7.6: Odpověď chatbota Výsledek typu „Odpověď chatbota“ bude zobrazovat textovou odpověď chatbota a obrázek obsahující ikonu konkrétního chatbota.

Detail tohoto výsledku zahájí internetovou komunikaci s daným chatbotem. Uživatel bude mít možnost zasílat další dotazy chatbotovi.

3.4.3 Doménový model

Tato aplikace se zaměřuje převážně na vyhledávání. S jedním vyhledáváním může být spojených několik výsledků. Každý výsledek má určitý typ. Jednotlivé typy výsledků mohou mít různé atributy, je tedy na místě zde použít dědičnost.

Mimo samotné vyhledávání popisuje model na obrázku 3.5 také uchovávání příchozích upozornění a aktuálně přihlášeného zaměstnance.



Obrázek 3.5: Doménový model aplikace

Kapitola 4

Návrhové vzory a technologie pro Android

4.1 Návrhové vzory

V následující kapitole jsou vysvětleny pojmy týkající se návrhových vzorů použitých při tvorbě aplikace.

4.1.1 Observer

Tato kapitola vychází z [19].

Návrhový vzor *Observer* slouží k zadefinování *1:N* vazby mezi objekty tak, že když jeden objekt změní svůj stav, jsou na tuto změnu upozorněny všechny objekty, které jsou na něm závislé. Takové objekty jsou nazývány „pozorovatelé“.

Vzor *Observer* má své použití zejména v případě, kdy změna jednoho objektu vyžaduje změnu několika dalších objektů. Počet, stav a chování těchto objektů přitom původní změněný objekt nezná.

4.1.2 Architektura MVVM

Kapitola vychází z článku [20].

Architektura *MVVM* se skládá ze tří vrstev:

- *Model*,
- *View*,
- *ViewModel*.

Modelová vrstva obsahuje manipulaci s daty v aplikaci. Veškerá business logika je obsažena v této vrstvě. Všechny náročné a dlouhotrvající operace se nacházejí v modelové vrstvě.

Vrstva *View* obstarává komunikaci s uživatelem. V této vrstvě se nenachází žádná logika aplikace. Proto může být vývoj této vrstvy úplně oddělen od vývoje zbytku aplikace. Často je tato vrstva tvořena pouze uživatelským rozhraním, definovaném například formou XML dokumentů. Počítá se zde s častými změnami v uživatelském rozhraní, které neovlivní chod zbytku aplikace.

ViewModel vrstva je prostředníkem mezi vrstvami *Model* a *View*. Výsledky operací v modelové vrstvě jsou oznámeny vrstvě *View* pomocí návrhového vzoru *Observer*. Realizaci tohoto návrhového vzoru zprostředkovává právě *ViewModel*, který váže data z vrstvy *Model* s prvky uživatelského rozhraní vrstvy *View*.

Tento architektonický vzor odděluje vývoj uživatelského rozhraní od vývoje logiky aplikace. Výsledná aplikace je tedy rozdělená do vrstev podle funkcionality a je kladen důraz na nízkou provázanost mezi jednotlivými vrstvami.

Závislosti mezi jednotlivými vrstvami jsou následující:

- vrstva *View* je datově vázána s vrstvou *ViewModel*,
- vrstva *ViewModel* drží závislost na vrstvu *Model*.

Těmito pouhými dvěma závislostmi dodržuje architektonický vzor *MVVM loose coupling* (nízká provázanost mezi jednotlivými komponentami systému) a *high cohesion* (vysoká soudružnost odpovědnosti jednotlivých komponent systému).

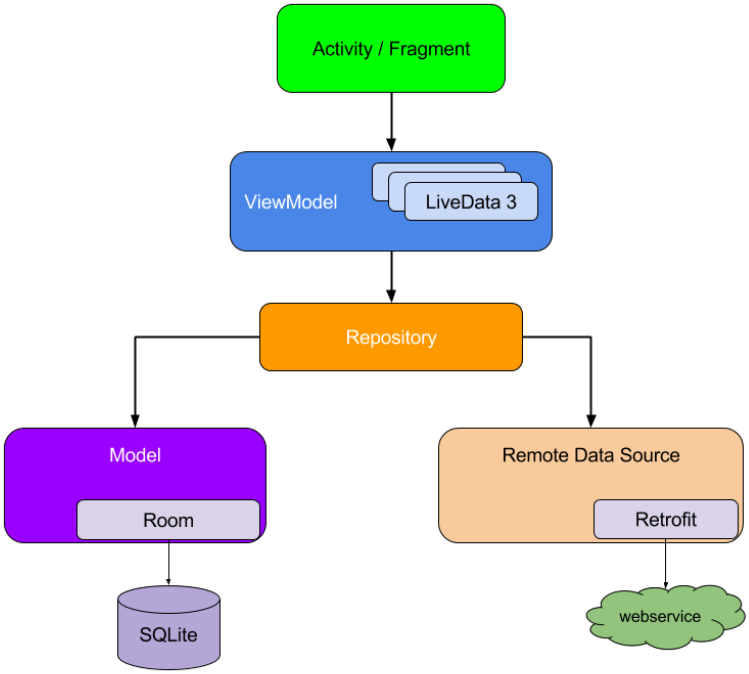
4.1.3 Singleton

Vzor *Singleton* zajišťuje pro danou třídu pouze jedno vytvoření její instance. Referenci na tuto jedinou instanci mohou získat ostatní objekty pomocí veřejného přístupového rozhraní třídy *Singleton*. [19]

4.1.4 Vkládání závislostí

Vkládání závislostí (anglicky *dependency injection*) je způsob, jak v objektově-orientovaném jazyce předávat závislosti mezi objekty.

Dle [21] je závislost objekt či služba, která může být využívána jinými objekty. Vkládání je proces předání této služby jinému objektu v konstruktoru, namísto vytváření nebo hledání závislosti na službu přímo uvnitř takového objektu.



Obrázek 4.1: Doporučená architektura pro tvorbu Android aplikací [22]

4.2 Architektonické komponenty

Android poskytuje několik knihoven pro vytváření robustních, udržovatelných a dobře testovatelných aplikací. Pomocí těchto knihoven je možné vytvořit v aplikaci komponenty, které realizují architektonický vzor *MVVM*.

Obrázek 4.1 zobrazuje doporučené závislosti mezi komponentami Android aplikace. Obrázek je převzatý z oficiálního průvodce společnosti Google pro vývoj stabilních, udržovatelných a snadno rozšiřitelných aplikací pro operační systém Android. Jednotlivé komponenty na obrázku jsou popsány v následujících sekcích.

4.2.1 LiveData

LiveData je datová třída poskytující možnost pozorování změn. Hlášení o změnách je aktualizováno pouze pro takové pozorovatele, kteří jsou v aktivním stavu z hlediska celé aplikace. Tím je zajištěna například správná práce s pamětí. [23]

Díky třídě *LiveData* je zajištěno, že se stav dat v modelu aplikace přesně shoduje se stavem dat zobrazeným v uživatelském rozhraní.

4.2.2 ViewModel

ViewModel je třída, která zachovává stav dat při změnách konfigurace uživatelského rozhraní (jako je například rotace obrazovky). [24]

Jako veřejný atribut obsahuje data ve formě *LiveData* třídy, a tím vystavuje tato data třídám implementující datové provázání s uživatelským rozhraním.

4.2.3 Room

Tato kapitola vychází z [25].

Knihovna Room poskytuje třídy, které zprostředkovávají abstrakci nad lokální databází SQLite. Z hlediska architektury slouží lokální SQLite databáze jako vyrovnávací paměť pro ostatní datové zdroje a poskytuje uživatelům data i bez přístupu k Internetu.

Na rozdíl od většiny knihoven pro objektové mapování relační databáze nevynechává knihovna Room jazyk SQL úplně. Pro databázové dotazy zaměřené na výběr záznamů z tabulek je nutné použití jazyka SQL. Takové dotazy je možné parametrizovat pomocí parametrů funkcí, které tyto dotazy anotují. Výsledné dotazy jsou tedy parametrizované při práci s SQLite databází tak, že předcházejí útoku *SQL injection*.

Ověření korektnosti každého dotazu se děje při kompilaci. Při ověření se kontroluje, zda lze návratovou tabulku SQL dotazu namapovat na třídu programovacího jazyka. Dále se kontroluje, zda typy parametrů SQL dotazu odpovídají skutečným typům sloupců v databázi.

4.2.4 Fragment & Activity

Informace uvedené v této kapitole pocházejí z [26].

Fragment je třída představující nějakou část uživatelského rozhraní na obrazovce. Každý *Fragment* má svůj životní cyklus a může být odstaven nebo zničen v jakýkoliv moment. Z tohoto důvodu je důležité pracovat s daty výhradně přes *ViewModel*, na který si *Fragment* vytváří závislost.

Activity je třída, která představuje možnost vstupu uživatele do mobilní aplikace. Těchto vstupů může být v aplikaci několik. Jedna *Activity* obsahuje prvky uživatelského rozhraní a může se skládat z více různých instancí třídy *Fragment*.

Mezi jednotlivými instancemi třídy *Activity* by neměly být silné vazby. K předávání informací při spouštění jiné *Activity* se používají instance třídy typu *Intent* (záměr). Pomocí těch lze spouštět jak své vlastní *Activity*, tak *Activity* naprosto odlišných aplikací.

Tímto způsobem lze využívat již hotová řešení (například aplikaci internetového prohlížeče nebo videokamery) namísto nutnosti vlastní implementace. Uživatel je pak odkázán na již známé uživatelské rozhraní dané aplikace, což je pro něj výhodou.

4.2.5 Navigace

Navigační komponenta umožňuje jednoduché navigování mezi jednotlivými obrazovkami v aplikaci. Jednoduchost obstarává především grafický editor v programu *Android Studio*, který z grafické reprezentace vztahů mezi jednotlivými fragmenty generuje třídy potřebné pro realizaci přesunů v aplikaci. Navigace se dále stará o správu paměti při tvorbě a zániku aplikačních fragmentů a přenosu dat mezi nimi. [27]

Výhodou použití navigační komponenty je zejména konzistence s ostatními android aplikacemi z hlediska uživatelského rozhraní.

4.3 Uchovávání dat

Následující kapitola a její podkapitoly čerpají z [28].

V operačním systému Android jsou tři hlavní způsoby, jak lze ukládat data na lokálním zařízení:

- interní souborové úložiště,
- externí souborové úložiště,
- sdílené preference.

4.3.1 Interní úložiště

K internímu úložišti má Android aplikace ve výchozím nastavení privátní přístup. Samotný uživatel ani žádná jiná aplikace v zařízení nemá za normálních okolností přístup k souborům uložených v interním úložišti.

Z tohoto důvodu je vhodné zde ukládat data specifické pro danou aplikaci.

Přístup k datům je však povolen procesům běžícím pod uživatelem *root*. Pokud se zařízení nachází ve stavu tzv. „Rooted Device“, může vlastník zařízení využívat přístupových práv uživatele *root* k tomu, aby přečetl data z interního úložiště aplikace. Není tedy vhodné zde ukládat citlivé informace, jako jsou například přihlašovací údaje.

Uchovávání dat v interním úložišti začíná a končí se samotnou aplikací. Při odinstalování aplikace jsou smazány všechny soubory z interního úložiště.

V interním lokálním úložišti je u každé aplikace připraven speciální adresář pro soubory obsahující vyrovnávací paměť. O takové soubory se musí aplikace starat sama. Operační systém však může tento adresář kdykoliv vyprázdnit, například pokud dochází místo v souborovém systému celého zařízení.

4.3.2 Externí úložiště

Externí úložiště v zařízení je takové úložiště, které může uživatel prohlížet a upravovat bez ohledu na aplikaci, která ho spravuje.

Existence externího úložiště je z pohledu aplikace nejistá. V případě paměťových karet může například dojít k úplnému fyzickému odpojení úložiště.

Do tohoto typu úložiště se nejčastěji ukládají například multimediální soubory, které mají význam i mimo kontext dané aplikace.

4.3.3 Sdílené preference

Malé množství nestrukturovaných dat lze ukládat jako sdílené preference. Jedná se o data typu klíč-hodnota, která jsou uložena do XML souboru.

Hlavní využití tohoto úložiště je pro ukládání uživatelského nastavení aplikace.

4.4 Správa účtů

Následující informace jsou převzaty z [29] a [30].

Preferovaný způsob, jak v Android zařízeních uchovávat údaje o uživatelských účtech, je pomocí třídy *AccountManager*.

Tato třída dokáže získat, modifikovat a ukládat uživatelské účty přímo do Android zařízení. To s sebou přináší několik výhod:

- vlastník zařízení má přehled o tom, jaké účty jsou v zařízení zaregistrované,
- vlastník zařízení má kontrolu nad těmito účty a může je případně smazat,
- ostatní aplikace mohou využívat tyto existující účty,
- při obnovení zařízení je možné obnovit i takto uložené účty, jelikož jsou zálohované systémem.

4.4.1 Bezpečnost účtů v zařízení

Pro účely této aplikace je třeba vytvořit nový typ účtu pro Android zařízení, kterým se budou autorizovat požadavky na bankovní vyhledávací server.

Základní údaje pro vytvoření účtu v zařízení jsou název a heslo. Tyto a všechny další pomocné údaje jsou uloženy v databázi nešifrovaně (tedy v otevřeném textu).

Přístup k této databází má pouze *root* a požádat o tyto údaje mohou pouze specifické aplikace (například aplikace, která účet vytvořila). To ale nepřináší bezpečnost uložených dat. Na zařízeních, které mají přístup k právům uživatele *root* je možné zobrazit celý obsah této databáze.

Z tohoto důvodu se zásadně nedoporučuje uchovávat jako heslo účtu skutečné heslo uživatele. Namísto hesla lze vyplnit tento údaj například kryptograficky bezpečným tokenem pro obnovení autorizačního tokenu.

Více o autentizačních tokenech v následující sekci 4.4.2.

4.4.2 OAuth 2.0

Následující kapitola čerpá z RFC článku [31].

OAuth 2.0 je autorizační framework, který umožnuje aplikacím třetích stran omezený přístup k HTTP službě.

Smyslem *OAuth 2.0* je autentizovat a autorizovat klienta bez nutnosti opakovaných odesílání přihlašovacích údajů na server.

OAuth 2.0 zavádí přístupové tokeny. Ty jsou kryptograficky bezpečné (jsou náhodné – nedají se predikovat). Pomocí přístupového tokenu dostává klient přístup k zabezpečeným prostředkům na serveru. Přístupový token slouží jak k autentizaci uživatele, tak k určení autorizačních práv (rolí) pro daného uživatele v rámci aplikace. Tyto tokeny mají časově omezenou platnost a mohou být také zrušeny předčasně na straně serveru.

V případě, že klient ještě nezná přístupový token, nebo vypršela platnost jeho přístupového tokenu, zašle přihlašovací údaje (například uživatelské jméno a heslo) na autorizační server, kde proběhne autentizace a autorizace uživatele. Při úspěšné autentizaci a autorizaci je vrácen přístupový token, který může klientská aplikace dále používat pro autorizování požadavků na server.

Volitelně může autorizační server při vystavení přístupového tokenu přidat ještě obnovovací token. Obnovovací token slouží k obnovení přístupového tokenu v případě, že vyprší platnost přístupového tokenu. V takovém případě má klient možnost získat nový přístupový token tak, že zkusí odeslat obnovovací token na autorizační server namísto odesílání přihlašovacích údajů uživatele. Obnovovací tokeny mají také omezenou platnost.

4.5 Firebase

Google Firebase je „cloudová“ služba umožnující získávat organizovaná data z mobilních aplikací. Podporované platformy jsou:

- Android aplikace,
- iOS aplikace,
- webové aplikace. [32]

Aby mohla aplikace využívat těchto služeb, je potřeba aplikaci propojit s existujícím projektem služby Google Firebase. Firebase projekt se základní funkcionalitou je možné založit zdarma. [32]

Služba Firebase umožňuje velké množství funkcionalit. Pro účely této aplikace však bude využívaná pouze služba *Firebase Cloud Messaging*.

4.5.1 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) je služba umožňující spolehlivé zasílání zpráv aplikacím, a to s nulovými náklady. Následující informace o FCM jsou převzaty z [33].

Hlavní vlastností této služby je zasílání upozorňovacích a datových zpráv přímo do jednotlivých zařízení.

FCM rozlišuje dvě komponenty pro realizaci komunikace:

- důvěryhodné prostředí (server), kde lze vytvářet, cílit a odesílat zprávy,
- iOS, Android nebo webová klientská aplikace, která dokáže zachytit zprávu.

Zprávy lze odesílat následujícím způsobem:

- pomocí *FireBase Admin SDK*,
- přes *FCM Server protokoly*,
- s použitím *FCM Notification composer*, jenž slouží výhradně pro testovací účely.

Z hlediska této práce je důležitý způsob odesílání zpráv pomocí *FireBase Admin SDK*.

Knihovna *FireBase Admin SDK* [34] je implementována pro několik programovacích jazyků (JavaScript, Java, Python, Go nebo C#). Tato knihovna umožňuje následující funkcionality:

- odeslání zprávy konkrétnímu zařízení podle jeho ID,
- odeslání zprávy všem zařízením odebírajícím určité téma (například „zprávy z banky“ nebo „mimořádné události pro zaměstnance“),
- přidat nebo odebrat konkrétní zařízení z odběratelů určitého tématu,
- vytvářet specifický obsah zprávy pro konkrétní platformy příjemců (jinými slovy zaslat jiný obsah zprávy pro Android aplikaci než pro iOS aplikaci).

Jak je uvedeno v [35], lze zasílané zprávy přes FCM rozlišit na dva typy: *Data message* a *Notification message*. Podle tohoto typu příchozí zprávy se odvíjí reakce zařízení. Jak reaguje Android zařízení je popsáno v [36].

V případě příchozí zprávy typu *Data message* je Android aplikace probuzena a má možnost na zprávu reagovat podle svého uvážení. Cílová Android aplikace je probuzena jak v případě, kdy běží pouze na pozadí, tak v případě, kdy je přímo používána uživatelem v popředí. *Data message* zprávy je vhodné používat pokud je při obdržení zprávy potřeba nějaká akce ze strany aplikace.

Pokud je příchozí zpráva typu *Notification message* a Android aplikace se nachází na pozadí, pak je automaticky zobrazeno upozornění v hlavním panelu Android zařízení. Toto upozornění se tváří jako upozornění z cílové aplikace. Ta ale není probuzena příchozí zprávou dokud uživatel nezareaguje na dané upozornění. V případě, že je Android aplikace v popředí (právě používaná), je reakce aplikace na zprávu typu *Notification message* je obdobná s reakcí na zprávu typu *Data Message*.

4.6 Kotlin jako programovací jazyk

Pro vývoj aplikací pro OS Android jsou společností Google oficiálně podporovány jazyky Kotlin a Java. [37]

Programovací jazyk Kotlin je výrazně mladší než Java. První stabilní verze byla vydána v únoru roku 2016, tedy přibližně o 20 let později než první stabilní verze jazyku Java. [38], [39]

Základní vlastností Kotlinu je interoperabilita s jazykem Java. Kotlin je tedy plně kompatibilní s kódem napsaným v Javě a naopak. Z toho plyne hlavní cíl Kotlinu, kterým je použivatelnost. [39]

Následující vlastnosti Kotlinu vycházejí z [40].

Kotlin je praktický jazyk navržený pro řešení reálných problémů. Nemá za cíl nahradit jazyk Java ve všech projektech. Nejedná se o pokus o převrat ve světě programování, nýbrž o shromáždění funkcionalit ostatních programovacích jazyků a stylů, které se ukázaly přínosné pro efektivní tvorbu software.

Dalším cílem Kotlinu je stručnost. Proto se syntaxe Kotlinu snaží docílit toho, aby každá část napsaného kódu nesla nějaký logický význam.

Oproti jazyku Java podporuje syntaxe Kotlinu znatelně kratší zápis často používaných situací. Ukázka 1 zobrazuje stručný zápis datové třídy v jazyce Kotlin oproti dlouhému zápisu stejné třídy v jazyce Java.

Posledním cílem filosofie Kotlinu je bezpečnost. V případě programovacích jazyků se bezpečností myslí navržení syntaxe tak, aby předcházela a zabránila co nejvíce chybám, kterých se může vývojář dopustit.

Velká část bezpečnostních opatření je zajištěna díky tomu, že Kotlin běží na JVM (Java Virtual Machine). JVM zajišťuje například správnou práci s dynamicky alokovanou pamětí.

Kotlin je staticky typovaný jazyk, proto ve spolupráci s JVM zaručuje bezpečnou práci s typy proměnných za běhu programu. Statické deklarování typů proměnných je však v Kotlinu opět vyřešeno stručnějším zápisem než v Javě. Deklaraci typu proměnné je možné vynechat pokaždé, když má kompilátor dostatek informací pro jednoznačné určení daného typu.

Ukázka 2 porovnává statické typování proměnných v jazycích Kotlin a Java na příkladu použití třídy *AccountManager* pro práci s účty v Android zařízení.

```kotlin
// Kotlin
data class Foo (
    val exampleString : String?,
    val exampleList : List<Int>
)

// Java
class Foo {
    private String exampleString;
    private List<Int> exampleList;

    public Foo(String exampleString, List<Int> exampleList) {
        this.exampleString = exampleString;
        this.exampleList = exampleList;
    }

    public String getExampleString() {
        return this.exampleString;
    }

    public void setExampleString(String newString) {
        this.exampleString = newString;
    }

    public List<Int> getExampleList() {
        return this.exampleList;
    }

    public void setExampleList(List<Int> newList) {
        this.exampleList = newList;
    }
}
```

Ukázka kódu 1: Stručný zápis datové třídy v Kotlinu oproti dlouhému zápisu stejné třídy v Javě

Další bezpečnostní opatření Kotlinu je takzvaná „Null Safety“. Cílem této vlastnosti je předcházet nebezpečí manipulace s neexistujícími objekty (problém známý v Javě jako „NullPointerException“). Toho Kotlin dosahuje pomocí rozlišovaní datových typů, které mohou obsahovat ukazatel na *null*, a typů, které nemohou. [41]

```
// Kotlin
val am = AccountManager.get(context)
val accounts = am.getAccountsByType("some.type")
for (account in accounts) {
    // do something with the account
}

// Java
AccountManager am = AccountManager.get(context);
Account[] accounts = am.getAccountsByType("some.type");
for (Account account : accounts) {
    // do something with the account
}
```

Ukázka kódu 2: Využití automatického přiřazení typu proměnné v Kotlinu

4.7 Testování Android aplikací

Následující kapitola čerpá z [42], [43].

Řádně otestovaná Android aplikace obsahuje testy rozdělené do tří částí:

- malé testy (anglicky *Small Tests*),
- střední testy (anglicky *Medium Tests*),
- velké testy (anglicky *Large Tests*).

V případě malých testů se jedná o jednotkové testy nezávislých komponent. Takové testy se zaměřují pouze na izolovanou část aplikace a musejí splňovat následující vlastnosti:

- důkladně prověřit funkcionalitu testované komponenty,
- s každým opakováním vracet stejné výsledky,
- prověřovat pouze jeden specifický aspekt testované komponenty,
- velmi rychle provádět testy a vracet výsledky.

Rychlost provádění testů závisí především na zařízení, na kterém jsou spouštěny. Pokud test nezávisí na aplikačním frameworku Android, je možné ho spouštět přímo na vývojovém zařízení. Takové testy se nazývají „lokální jednotkové testy“ (anglicky *local unit tests*) a jsou z pravidla velmi rychlé, jelikož nevyžadují instalaci zdrojového kódu do externího nebo virtuálního zařízení se systémem Android. Testy spouštěné tímto způsobem tedy nemohou obsahovat závislosti na aplikační framework Android. Odstranění těchto

závislostí lze pomocí takzvaného „mockování“ (anglicky *mocking*), neboli vytvoření falešné implementace třídy představující závislost. Funkcionalita této falešné třídy se pak odvíjí od požadavků na konkrétní test.

Některé testy však vyžadují běh na Android zařízení. V takovém případě jsou prováděny například na virtuálním zařízení a nazývají se „instrumentační testy“ (anglicky *instrumented tests*). Rychlost těchto testů je znatelně pomalejší, neboť musí nejprve proběhnout instalace zdrojového kódu do zařízení.

Střední testy jsou integrační testy prověřující spolupráci více komponent. Tyto testy jsou vždy prováděny na Android zařízení.

Velké testy jsou také integrační testy, které slouží k ověření funkčnosti očekávaných průchodů uživatelským rozhraním aplikace. Zajišťují, že hlavní případy užití aplikace fungují na skutečných (nebo alespoň virtuálních) Android zařízeních.

4.8 Knihovny třetích stran

4.8.1 Retrofit

Knihovna Retrofit [44] slouží pro implementaci webového rozhraní.

Nutností je definice tříd pro očekávanou strukturu odchozích a příchozích dat. Tím je celý proces webové komunikace typovaný, což vede k jednoduše odhalitelným chybám.

Díky příslušným anotacím knihovny Retrofit je použití této open-source knihovny ve zdrojovém kódu přehledné.

Retrofit podporuje asynchronní volání webových rozhraní. To je z hlediska vývoje responzivní aplikace naprosto zásadní.

Aby mohla aplikace využívat přístup k Internetu, je potřeba definovat oprávnění v manifestu. Nutná oprávnění jsou *INTERNET*, který poskytuje možnost komunikace pomocí síťového rozhraní a *INTERNET_STATUS*, jenž je nezbytný k reagování na nedostupnost připojení a odchytávání případných výjimek.

4.8.2 Kodein

Kodein [45] je framework pro usnadnění práce s vkládáním závislostí pro Android aplikace.

Tento framework váže jednotlivá rozhraní s jejich konkrétními implementacemi a na jednom místě v programu vkládá závislosti objektům bez nutnosti dodržení přesného pořadí vytváření jednotlivých služeb.

Vhodné místo pro vytvoření všech závislostí je přetížená metoda *onCreate* třídy *Application*, což je první metoda, která se spustí při startu aplikace.

4.8.3 Mockito

Mockito [46] je „mockovací“ nástroj pro lokální jednotkové testy v jazyce Java.

Pomocí tohoto nástroje lze nahradit implementaci třídy nebo rozhraní pomocí falešné implementace. Chování metod této falešné implementace lze definovat pro každý test. Tímto způsobem lze zadefinovat požadované chování závislostí testované třídy a soustředit test pouze na konkrétní třídu.

4.8.3.1 Mockito-Kotlin

Knihovna Mockito je navržena pro programovací jazyk Java a použití knihovny v programovacím jazyce Kotlin s sebou přináší problémy. Hlavním důvodem je ověřování datových typů v Kotlinu, zda obsahují hodnotu *null*.

Tyto problémy úspěšně řeší knihovna Mockito-Kotlin [47]. Knihovna nijak nemění chování knihovny Mockito, pouze přidává třídy a metody pro pohodlnější „mockování“ tříd a rozhraní v jazyce Kotlin.

4.8.4 Robolectric

Testovací knihovna Robolectric [48] vytváří virtuální prostředí aplikačního frameworku Android na JVM vývojového zařízení. To umožňuje provádět jednotkové testy vyžadující některé prostředky aplikačního frameworku Android přímo na vývojovém zařízení. Výsledná rychlost provádění testů je tak znatelně rychlejší, jelikož není nutné instalovat aplikaci na virtuální nebo reálné zařízení před každým spuštění testů.

4.8.5 Esspresso

Knihovna Esspresso [49] slouží k psaní automatizovaných testů uživatelského rozhraní. Knihovna simuluje chování uživatele a umožňuje testovat stav prvků uživatelského rozhraní po každém kroku.

4.9 Distribuce firemních aplikací

Pro distribuci firemního software na zařízení zaměstnanců se používá řešení MDM (*Mobile Device Management*) popsané v následujících sekcích vycházejících z [50].

4.9.1 MDM (*Mobile Device Management*)

Toto řešení poskytuje centrální správu všech firemních zařízení.

V souladu s firemní politikou pak lze vzdáleně aktivovat, konfigurovat, podporovat a kontrolovat zabezpečení těchto zařízení. Ta lze také v případě potřeby například vzdáleně uzamknout nebo smazat jejich obsah.

4.9.2 MAM (*Mobile Application Management*)

Podobně jako u MDM přebírá firma kontrolu, tentokrát však pouze nad jednotlivými aplikacemi v mobilním zařízení.

MAM typicky poskytuje vlastní *enterprise app store* (nástroj pro distribuci a správu aplikací v mobilním zařízení), pomocí kterého lze do zařízení distribuovat interně vyvíjené mobilní aplikace. Pro takto distribuované aplikace poskytuje MAM centrální správu jejich licencí, konfigurací a lokálně uložených dat.

Kapitola 5

Návrh

Tato kapitola popisuje návrh mobilní zaměstnanecké aplikace.

5.1 Použitá architektura

Použitý architektonický vzor v aplikaci je vzor *MVVM* (*Model View ViewModel*).

Modelová vrstva zahrnuje takzvané *Úložiště* (*Repository*), které poskytuje rozhraní pro přístup k datům. Toto úložiště vytváří abstrakci nad všemi datovými zdroji, tedy lokální SQLite databází a serverovou databází, dostupnou přes aplikační rozhraní. *Úložiště* se může vyskytovat v aplikaci vždy právě jednou, proto je navržené jako *Singleton*.

Uživatelské rozhraní je realizováno pomocí skládání objektů *Fragment* v hlavní aktivitě (instance třídy *Activity*). Každý *Fragment* si udržuje závislost na příslušný *ViewModel*, přes který se dostane až k *Úložišti*. *Fragment* detekuje změny v *Úložišti* pomocí posluchačů (*Observer*) pro data.

Použitá architektura tedy přesně odpovídá doporučené architektuře Android aplikací zobrazené na obrázku 4.1.

5.2 Komunikace mezi třídami architektonických komponent

Dle [51] jsou operace pro uživatelské rozhraní každé android aplikace prováděny v hlavním vlákně procesu. Je tedy nezbytné toto vlákno neblokovat dlouhotrvající operací.

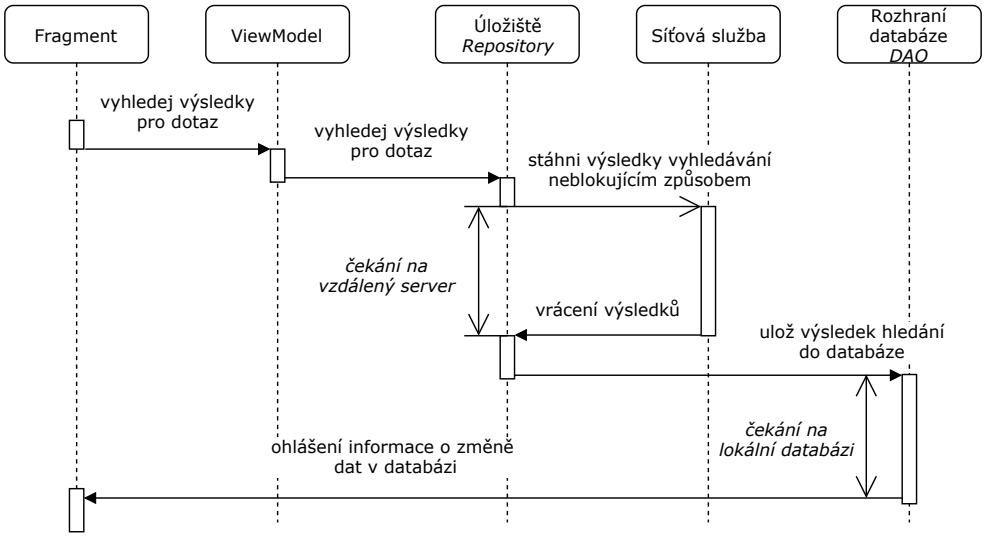
Příkladem takové operace může být čekání na odpověď vzdáleného serveru, či načtení objektů z lokální databáze. Po spuštění takové operace blokujícím způsobem na hlavním vlákně by uživatelské rozhraní aplikace přestalo reagovat, dokud by operace neskončila.

Na obrázku 5.1 je znázorněna komunikace tříd zmíněných architektonických komponent při odeslání dotazu k vyhledání na vzdálený bankovní server.

Z obrázku je vidět, že hlavní vlákno provádějící operace ve třídě *Fragment* odešle dotaz přes *ViewModel* a dále už se nestará o žádné další výpočetní operace, pouze pasivně pozoruje změny v datech.

Úložiště poté vytvoří nové vlákno, které zařídí celý zbytek operací pro stažení hledaných výsledků a jejich uložení do databáze.

V tu chvíli je pomocí návrhového vzoru *Observer* ohlášena změna dat naslouchající instanci třídy *Fragment*, která na tento povel aktualizuje své uživatelské rozhraní.

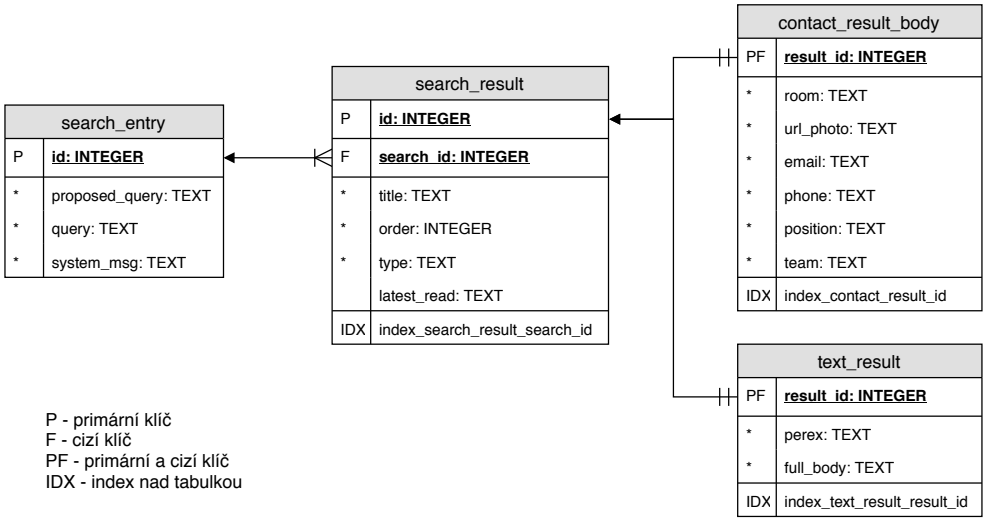


Obrázek 5.1: Komunikace tříd při vyhledání dotazu

5.2.1 Databázový model

Databázový model pro realizaci vyhledávání vycházející přímo z doménového modelu je naznačen se dvěma typy výsledků na obrázku 5.2. Skutečný model v příloze na obrázku B.1 obsahuje více typů výsledků a tabulku pro ukládání upozornění. Relace mezi třídami zůstávají stejné (ke každému vyhledávání se váže *N* výsledků). Dědičnost výsledků vyhledávání je nahrazena vazbou *1:1*, tedy každý výsledek má vazbu na nějaký typ doplňujících informací k danému výsledku.

Toto schéma databáze je jednoduše rozšířitelné pro přidání dalších typů výsledků pomocí přidání nové tabulky. Modifikace jednoho typu výsledku znamená přidání nového sloupce k příslušnému typu, což je také jednoduchá



Obrázek 5.2: Ukázka části databázového modelu

úprava databáze. Přidání atributu pro obecný výsledek lze rovněž provést přidáním jednoho sloupce do tabulky. Tím jsou pokryty nejpravděpodobnější rozšíření pro vyhledávání v této aplikaci.

V databázovém schématu v příloze B.1 nejsou přítomny entity pro správu uživatelských účtů aplikace. Účty je doporučené spravovat pomocí třídy *AccountManager*, která ukládá data do speciálně vyhrazeného úložiště pro všechny uživatelské účty v daném zařízení Android, jak bylo uvedeno v kapitole 4.4.1. Ukládání informací o uživatelích přímo ve vyvíjené aplikaci proto není potřeba.

Jako lokální databáze bude použita databáze SQLite a již zmíněná knihovna Room, která poskytuje rozhraní pro objektové mapování SQLite databáze.

Generování tabulek a sloupců v databázi obstarává knihovna Room. Implementaci databázových entit, které vytvoří příslušné databázové schéma, popisuje kapitola 6.1.1.

5.3 Obrazovky aplikace

Drátový model obrazovek aplikace (anglicky *wireframes*) obsahuje pouze abstraktní prvky uživatelského rozhraní a slouží k představě rozložení těchto prvků ve finální aplikaci. Výsledné návrhy obrazovek jsou v příloze na obrázku C.1.

Kapitola 6

Implementace

V následující kapitole jsou popsány všechny nástroje, knihovny a postupy, které byly použity při vývoji této mobilní aplikace.

6.1 Modelová vrstva

6.1.1 Rozhraní pro SQLite databázi

Pro práci s lokální SQLite databází byla použita knihovna Room.

Pomocí anotací knihovny Room lze definovat třídy, ze kterých se stanou entity, které pak knihovna při překladu použije pro vytvoření skutečných tabulek v databázi. Ukázka 3 vystihuje entitu jednoho výsledku vyhledávání.

```
@Entity(tableName = "search_entry")
data class SearchEntry(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val proposedQuery: String,
    val query: String,
    val systemMsg: String
)
```

Ukázka kódu 3: Definice databázové entity v jazyce Kotlin

Jednotlivé dotazy nad vytvořenými tabulkami se provádějí pomocí zadefinovaných rozhraní pro přístup k datům (*Data Access Object*). Pomocí tohoto rozhraní pak knihovna Room vygeneruje potřebné třídy a metody pro práci s objekty v databázi.

Výhodou je, že se celý proces děje při kompilaci, proto jsou případné chyby v SQL dotazech objeveny před spuštěním aplikace.

Knihovna Room navíc podporuje vracení objektů obalených do třídy *LiveData*. To je klíčový bod celé architektury aplikace. Při každé změně

výsledku daného SQL dotazu se ohlásí změna instanci třídy *LiveData*, což vede k oznámení změny v datech všem aktivním pozorovatelům (například uživatelskému rozhraní).

Vybrané metody rozhraní pro přístup k datům jsou zobrazeny v ukázce 4.

```
@Insert(onConflict = OnConflictStrategy.REPLACE)
fun insert(searchEntry: SearchEntry) : Long

@Query("select * from search_entry where id = :searchId")
fun getSearchEntry(searchId: Long): LiveData<SearchEntry>
```

Ukázka kódu 4: Příklady vybraných metod pro přístup k datům pomocí knihovny Room

6.1.2 Síťové služby

Veškerá síťová komunikace je prováděna pomocí knihovny Retrofit.

Pro každý koncový bod bankovního vyhledávacího serveru je vytvořeno rozhraní s anotovanými metodami pro knihovnu Retrofit. Jedna metoda rozhraní představuje popis jednoho HTTP požadavku na server. Na ukázce 5 se nachází příklad asynchronního volání webového rozhraní.

```
@POST("/search")
fun searchForQueryAsync(
    @Query("query") searchQuery : String
): Deferred<SearchResponse>
```

Ukázka kódu 5: Použití anotovaných metod knihovny Retrofit

O provádění těchto požadavků se starají implementace rozhraní pro práci se sítí (v aplikaci například *SearchNetworkDataSourceImpl* nebo *AuthNetworkDataSourceImpl*). Ty spouštějí asynchronní volání požadavků v jiném vlákně a na pozadí vyčkávají na příslušné odpovědi serveru. Zde také probíhá odchytávání výjimek vznikajících při síťové komunikaci. Odchycené výjimky jsou zobrazeny v uživatelském rozhraní pomocí ohlášení změn v datech přes *Úložiště*.

Výsledná aplikace pracuje se čtyřmi rozhraními pro správu webových volání.

Jedná se o rozhraní *SearchNetworkDataSource*, které zajišťuje komunikaci s bankovním vyhledávačem. Každému odeslanému požadavku je přidána autorizační hlavička, do které se zapíše aktuální přístupový token zapsaný v Android zařízení pro používaný účet. V případě obdržení stavového kódu HTTP protokolu 401 (stavový kód pro neautorizovaný požadavek) dojde ke zrušení platnosti přístupového tokenu pro aktuálního uživatele v zařízení.

Rozhraní *AuthNetworkDataSource* spravuje webovou komunikaci s autorizačním serverem. Mezi funkce tohoto rozhraní patří obdržení přístupového tokenu po úspěšné autentizaci uživatele a obdržení nového přístupového tokenu po úspěšném odeslání obnovovacího tokenu. Toto rozhraní tedy implementuje síťovou komunikaci potřebou pro realizaci autorizačního protokolu OAuth2 popsaného v kapitole 4.4.2.

Rozhraní *FCMNetworkDataSource* umožňuje registrovat FCM token daného zařízení na bankovní server. Společně s FCM tokenem se odesílá uživatelské jméno aktuálně používaného účtu v Android zařízení. Více informací o FCM je v kapitolách 4.5.1 a 6.3.

Poslední síťové rozhraní je *RickyNetworkDataSource*, které obstarává komunikaci mezi Android aplikací a bankovní serverovou aplikací Ricky. Aplikace Ricky představuje bankovní chatboty a vystavuje aplikační rozhraní pro komunikaci s konkrétním chatbotem. Toto síťové rozhraní implementuje detail výsledků vyhledávání typu *CHATBOT*.

6.1.3 Jednotné úložiště

Třída *SearchRepositoryImpl* v aplikaci implementuje *Úložiště* (*Repository*) z použité architektury.

Tato třída spravuje data z lokální databáze a síťových služeb. Udržuje závislost na rozhraní pro přístup k datům lokální databáze a závislost na síťovou službu pro vyhledávání. Úkolem třídy je rozhodovat z jakého zdroje čerpat data v dané situaci.

6.2 Vlastní typ účtu v zařízení

Veškerá správa uživatelských účtů je vykonávána přes třídu *AccountManager*, která pracuje s uživatelskými účty v Android zařízení.

Pro definování vlastního specifického účtu pro tuto aplikaci v Android zařízení bylo potřeba obstarat následující požadavky:

- XML definici účtu v zažízení (*authenticator.xml*),
- vlastní službu *CustomAccountService* pro zacházení s definovaným typem účtu,
- specifickou implementaci autentizace uživatelů s tímto typem účtu,
- přihlašovací obrazovku potřebnou k získání přihlašovacích údajů od uživatele. [30]

Zmíněný soubor *authenticator.xml* definuje typ, název a ikony nového účtu.

Služba *CustomAccountService* umožňuje provádění operací s účty tohoto typu i z jiného kontextu než z vyvíjené aplikace. Tato vlastnost se využívá například při manipulaci s účty přímo v nastavení Android zařízení.

Specifická implementace autentizace pro nově definovaný typ účtu je obsažena ve třídě *CustomAuthenticator*. Pomocí metod této třídy je uživatel odkázán na přihlašovací obrazovku pokaždé, když je vytvářen nový účet tohoto typu.

Při vytváření nového účtu se nejprve provede kontrola, zda již zařízení neobsahuje nějaký účet tohoto typu. Pokud již v zařízení takový účet existuje, přidání nového účtu skončí neúspěšně. V zařízení tedy může být v jednu chvíli pouze jeden účet pro tuto aplikaci. V případě, že přidání nového účtu proběhne úspěšně, provede se vyčištění dat z lokální databáze v souladu s funkčním požadavkem *F4: Přihlašování*.

Při přidávání nového účtu jsou zaslány přihlašovací údaje na autorizační server. Úspěšná autentizace vrátí přístupový a obnovovací token pro tohoto uživatele. Výsledný účet je tedy vytvořen s následujícími položkami:

- uživatelské jméno (identifikační číslo zaměstnance v bance),
- obnovovací token uložený z bezpečnostních důvodů na místo hesla (heslo je k účtu v Android zařízení vyžadováno),
- přístupový token.

Výsledná uložená struktura v Android zařízení tedy neobsahuje uživatelské heslo v jakékoli podobě, čímž je splněn nefunkční požadavek *N2: Bezpečnost a autentizace*.

Pomocí třídy *CustomAuthenticator* se také obstarává přístupový token spojený s Android účtem. Tento token vzniká na autorizačním serveru a v případě úspěšné autentizace je ukládán mezi údaje uživatelského účtu v zařízení. V případě, že aplikace vyžaduje tento token pro své potřeby, je volána příslušná metoda třídy *CustomAuthenticator*.

V této metodě je přístupový token jednoduše vrácen v případě, že pro daný uživatelský účet tento token existuje. To však nemusí nutně platit (například po zrušení platnosti přístupového tokenu aplikací). V takovém případě se zkusí provést obnova přístupového tokenu pomocí obnovovacího tokenu. Tento proces je blíže popsán v kapitole 4.4.2. Pokud i tato cesta selže, musí být uživatel vyzván k zadání nových přihlašovacích údajů a je tedy odkázán na přihlašovací obrazovku. V tomto případě uživatel zadává pouze heslo, jelikož je uživatelské jméno známé.

6.3 Upozornění

Zobrazení upozornění v horním panelu Android zařízení je pouze reakcí na zachycení datové zprávy aplikací. Přijímání datových zpráv zasílaných službou FCM je možné díky aplikační službě *CustomFirebaseMessagingService*. Tato služba má dvě funkce, které musí plnit.

První funkcí je registrování FCM tokenu pro dané zařízení přiděleného službou FCM na bankovní server. Bankovní server musí mít možnost spárovat zaměstnance s jejich zařízeními, aby mohl rozesílat cílená upozornění. Registrační token je aplikaci přidělen při její instalaci. Tento token se však může změnit (například při přeinstalování aplikace na zařízení), proto se registrování tokenu na bankovní server opakuje s každou změnou FCM tokenu.

Druhou funkcí služby je reagování na příchozí datové zprávy. Z obsahu datové zprávy je vytvořena entita *Notification*, která je uložena do databáze. Následně je vytvořeno upozornění zobrazené v horním panelu Android zařízení. Toto upozornění se do horního panelu dostane pomocí nějakého upozorňovacího kanálu.

Definované upozorňovací kanály v aplikaci jsou:

- informační kanál pro zaměstnance ČSOB,
- varovný kanál pro zaměstnance ČSOB.

Tyto dva kanály jsou vytvořeny ihned po startu aplikace. Vytváření kanálů je podmíněno příslušnou verzí zařízení. Dle [52] je Android 8.0 (API level 26) minimální verze operačního systému Android pro vytváření kanálů. Pokud má aktuální zařízení nižší verzi operačního systému, potom jsou všechna upozornění zobrazována stejným způsobem.

Zobrazování historie upozornění je možné na vlastní obrazovce, kde se zobrazují všechna upozornění uložená v databázi. Uživatel má možnost zobrazit jejich detail nebo je odstranit z databáze pomocí přetažením do strany. Tímto je implementován funkční požadavek *F2: Upozornění*.

6.4 Grafické uživatelské rozhraní

GUI aplikace vychází z drátěných modelů obrazovek aplikace. Jednotlivé prvky uživatelského rozhraní využívají stylů *Material Components* ze sady *Material Desing* [53].

Výsledná podoba vybraných obrazovek se nachází v příloze na obrázku E.1.

6.5 Testování

Rozdělení jednotlivých testů se řídí kapitolou 4.7.

6.5.1 Malé testy

Malé jednotkové testy se zaměřují především na co nejvyšší pokrytí modelové vrstvy aplikace.

K otestování tříd implementující síťovou komunikaci se využívá knihovna Mockito-Kotlin, pomocí které jsou mockovány volání metod síťových rozhraní knihovny Retrofit.

Tabulka 6.1: Technické parametry testovacích zařízení

Zařízení	verze Android OS	obrazovka
Samsung Galaxy A5 (2017)	8.0 (API 26)	5,2” 1080 x 1920
Lenovo A2010-a	5.1 (API 22)	4,5” 480 × 854
Nexus 5 (emulated)	7.0 (API 24)	4,95” 1080 x 1920
Nexus 5X (emulated)	9.0 (API 28)	5,2” 1080 x 1920
Google Pixel (emulated)	8.0 (API 26)	5,0” 1080 × 1920

Pro otestování přístupu k datům v databázi je vytvořena nová databáze v hlavní paměti počítače před každým spuštěním testu.

Při testování hlavního úložiště (*Repository*) se opět využívá knihovna Mockito-Kotlin, pomocí které se mockují síťová a databázová rozhraní. To má za následek velmi úzce zaměřené testování této třídy.

Díky knihovně Robolectric mohou být všechny jednotkové testy spuštěny na vývojovém zařízení.

Testy pracující s aplikačním frameworkem Android jsou díky konfiguraci knihovny Robolectric prováděny zvlášť pro každou verzi frameworku. Každý takový test je tedy spouštěn postupně s Android API level 22, 23, 24, 25, 26, 27 a 28.

6.5.2 Střední testy

Střední testy jsou prováděny přímo na testovacích zařízeních a zaměřují se na testování uživatelského rozhraní a testování přístupu k lokální databázi a lokálním účtům v zařízení.

Pro testování uživatelského rozhraní se používá knihovna Esspresso. Scénáře testů uživatelského rozhraní byly nahrány pomocí nástroje pro snímání uživatelských akcí pro tvorbu Esspresso testů. Takto vytvořené scénáře bylo třeba značně upravit pro širší zaměření testů.

Střední testy byly prováděny na zařízeních z tabulky 6.1.

6.5.3 Velké testy

Velké testy se zaměřují na ověření funkčnosti všech komponent Android aplikace.

Pro spuštění těchto testů je nutné mít spuštěný testovací Python server. Více informací o testovacím serveru se nachází v kapitole 6.5.4.

Aplikace během testovaných scénářů komunikuje s testovacím serverem přes reálná síťová rozhraní, ukládá data do reálné databáze v lokálním úložišti a spravuje účty v zařízení.

6.5.4 Testovací server

Pro testování komunikace aplikace se serverem byl vytvořen testovací server poskytující testovací data podle návrhu aplikačního rozhraní. Tento server navíc simuluje práci s autorizačními tokeny protokolu OAuth2 a zasílání upozornění přes službu FCM.

Server je implementován v jazyce Python s využitím webového frameworku Django [54].

Pro zabezpečenou komunikaci přes HTTPs mezi aplikací a testovacím serverem byla využita služba serveo.net [55].

6.5.5 Uživatelské testování

Chování prvků uživatelského rozhraní bylo prověřeno skupinou šesti zaměstnanců ČSOB a následně upraveno podle jejich zpětné vazby. Průběh uživatelského testování a výsledky formulářů pro zpětnou vazbu se nacházejí v příloze D. Počet respondentů (šest) je podle [56] dostatečný pro odhalení více než 80 % problémů s uživatelským rozhraním.

Na základě zpětné vazby z uživatelského testování byly opraveny odhalené nedostatky uživatelského rozhraní:

- klávesnice na přihlašovací obrazovce již nepřekrývá formulář,
- spodní navigační menu se již nezobrazuje při zadávání textu k vyhledání,
- dlouhý text pozice zaměstnance se již zobrazuje správně,
- mezi našeptávanými vyhledáváními se nyní zobrazují také nejčastěji zaznamenané dotazy na server.

Mezi složitější opravy chyb odhalených uživatelským testování patří hromadné mazání upozornění po označení. Tato funkcionalita bude implementována v budoucnu.

6.6 Distribuce

Dle [57] se v ČSOB používá pro distribuci interních aplikací MDM a MAM *Workspace ONE* od společnosti WMware. Bližší informace o tomto způsobu distribuce lze nalézt v kapitole 4.9.

Touto cestou lze distribuovat i APK této vyvíjené aplikace. Díky použití bankovního MDM bude mít aplikace navíc přístup k bankovnímu intranetu. Postup, jak instalovat aplikaci na bankovním zařízení, popisuje instalační příručka v příloze F.1.

Závěr

Cílem této práce bylo vytvořit mobilní aplikaci pro zaměstnance ČSOB umožňující pohodlnější vyhledávání informací napříč interními bankovními systémy. Zároveň bylo potřeba vytvořit softwarovou analýzu požadavků a případů užití společně se softwarovým návrhem aplikace před samotnou implementací.

Výsledná aplikace využívá aplikační rozhraní bankovního vyhledávacího serveru a zobrazuje výsledky vyhledávání. Dále aplikace reaguje na upozornění zasílaná bankovním serverem. Požadavky ze softwarové analýzy byly splněny.

Implementace aplikace se řídila softwarovým návrhem dodržujícím architektonické vzory podle aktuálních doporučení společnosti Google pro tvorbu robustních a snadno rozšiřitelných aplikací.

Uživatelské rozhraní aplikace využívající základní styly *Material Design* pro Android aplikace se ukázalo jako velice přehledné při uživatelském testování. Vizuální stránka aplikace je kompletně oddělená od aplikační logiky, proto lze pro aplikaci snadno vytvořit nové styly uživatelského rozhraní zkušenými designéry.

Výsledkem této práce je funkční prototyp připravený pro integrační testování s bankovním vyhledávacím serverem.

Aplikace je přizpůsobena k rozšíření funkcionalit. V budoucnu by mohla aplikace pomáhat zaměstnancům například s vyplňováním docházky, či s plánováním dovolené.

Seznam použité literatury

1. KUTÁLEK, Petr. *FW: návštěvnost telefonního seznamu.* [elektronická pošta]. Zasláno z pkutalek@csob.cz 12. 4. 2019 11:35 [cit. 2019-04-14]. Osobní komunikace.
2. KUTÁLEK, Petr. *Informace o datových zdrojích.* [elektronická pošta]. Zasláno z pkutalek@csob.cz. 5. 4. 2019 11:22 [cit. 2019-04-15]. Osobní komunikace.
3. KUTÁLEK, Petr. *Autentizace služeb ČSOB.* [elektronická pošta]. Zasláno z pkutalek@csob.cz. 25. 4. 2019 18:50. [cit. 2019-04-25]. Osobní komunikace.
4. ELASTICSEARCH B.V. *Elasticsearch* [online]. 2018 [cit. 2019-03-17]. Dostupné z: `https://github.com/elastic/elasticsearch`.
5. SOLIDIT. *DB-Engines Ranking of Search Engines* [online]. 2019 [cit. 2019-03-17]. Dostupné z: `https://db-engines.com/en/ranking/search+engine`.
6. AILAO. *YodaQA* [online]. 2012 [cit. 2019-03-17]. Dostupné z: `http://ailao.eu/yodaqa/`.
7. PLISSON, Joël; LAVRAC, Nada; MLADENIC, Dunja et al. A rule based approach to word lemmatization. *Proceedings of IS-2004*. 2004, s. 83–86.
8. LE, Quoc V.; MIKOLOV, Tomas. Distributed Representations of Sentences and Documents. *CoRR* [online]. 2014, roč. abs/1405.4053 [cit. 2019-04-14]. Dostupné z: `https://dblp.org/rec/bib/journals/corr/LeM14`.
9. MOON, Young. Enterprise Resource Planning (ERP): a review of the literature. In: [online]. 2007 [cit. 2019-04-07]. Dostupné z: `https://pdfs.semanticscholar.org/0029/244f210e2cb4fffb415e68907c6294a5c514.pdf?_ga=2.41512609.2007584071.1554669472-665863645.1554669472`.

10. MICROSOFT. *Active Directory Domain Services Overview* [online] [cit. 2019-04-25]. Dostupné z: https://bit.ly/2L7uv87.

11. HR LINKA ČSOB. *HR Chatbot Viki* [intranet ČSOB] [cit. 2019-05-06]. [cit. 2019-05-06].

12. ČSOB. *Telefonní seznam* [intranet ČSOB] [cit. 2019-05-06]. [cit. 2019-05-06].

13. JOSTLE CORPORATION. *Jostle Video Tour* [online] [cit. 2019-04-18]. Dostupné z: https://www.jostle.me/video-tour/.

14. JOSTLE CORPORATION. *Pricing* [online] [cit. 2019-04-18]. Dostupné z: https://www.jostle.me/pricing/.

15. MICROSOFT. *SharePoint mobile app for Android* [online] [cit. 2019-04-18]. Dostupné z: https://bit.ly/2VTacMm.

16. ČSOB. *Výroční zpráva ČSOB za rok 2017* [online]. Praha, ČR, 2017 [cit. 2019-03-17]. Dostupné z: https://www.csob.cz/portal/documents/10710/444804/vz-csob-2017.pdf.

17. KUTÁLEK, Petr. *Mobilní zařízení zaměstnanců.* [elektronická pošta]. Zasláno z pkutalek@csob.cz. 6. 5. 2019 12:51. [cit. 2019-05-06]. Osobní komunikace.

18. MOBILMANIA.CZ. *Samsung Galaxy A5 (2016)* [online] [cit. 2019-05-06]. Dostupné z: https://www.mobilmania.cz/samsung-galaxy-a5-2016.

19. GAMMA, Erich. *Design Patterns: Elements of Reusable Object-Oriented Software.* Boston: Addison-Wesley, 1995. ISBN 0-201-63361-2. [cit. 2019-04-20].

20. LI, Xiaolong; CHANG, DaLiang; PEN, Hui Min; ZHANG, Xiaoyu; LIU, Yuanxin; YAO, YaXian. Application of MVVM design pattern in MES. *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)* [online]. 2015 [cit. 2019-04-20]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7288144.

21. YANG, Hong Yul; TEMPERO, Ewan D.; MELTON, Hayden. An Empirical Study into Use of Dependency Injection in Java. *19th Australian Conference on Software Engineering (aswec 2008)* [online]. 2008 [cit. 2019-04-20]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4483212.

22. GOOGLE. *Guide to app architecture* [online]. San Francisco, USA [cit. 2019-03-17]. Dostupné z: https://developer.android.com/jetpack/docs/guide.

23. GOOGLE. *LiveData* [online]. San Francisco, USA [cit. 2019-03-20]. Dostupné z: https://developer.android.com/reference/android/arch/lifecycle/LiveData.

24. GOOGLE. *ViewModel Overview* [online]. San Francisco, USA [cit. 2019-03-20]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel.

25. GOOGLE. *Introduction to Activities* [online]. San Francisco, USA [cit. 2019-03-18]. Dostupné z: https://developer.android.com/training/data-storage/room/index.html.

26. GOOGLE. *Introduction to Activities* [online]. San Francisco, USA [cit. 2019-03-18]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities.

27. GOOGLE. *Navigation* [online]. San Francisco, USA [cit. 2019-03-24]. Dostupné z: https://developer.android.com/guide/navigation.

28. GOOGLE. *Data and file storage overview* [online]. San Francisco, USA [cit. 2019-03-18]. Dostupné z: https://developer.android.com/guide/topics/data/data-storage.

29. GOOGLE. *Remember your user* [online]. San Francisco, USA [cit. 2019-03-24]. Dostupné z: https://developer.android.com/training/id-auth/identify.

30. GOOGLE. *Create a custom account type* [online]. San Francisco, USA [cit. 2019-03-24]. Dostupné z: https://developer.android.com/training/id-auth/custom_auth.

31. HARDT, Dick. The OAuth 2.0 Authorization Framework. *RFC* [online]. 2012, roč. 6749 [cit. 2019-03-24]. Dostupné z: https://bit.ly/2UG5ArP.

32. STONEHEM, Bill. *Google Android Firebase: Learning the Basics*. First Rank Publishing, 2016.

33. GOOGLE. *Firebase Cloud Messaging* [online]. San Francisco, USA [cit. 2019-04-11]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/.

34. GOOGLE. *Firebase Admin SDK for FCM* [online]. San Francisco, USA [cit. 2019-04-11]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/server%5C#firebase-admin-sdk-for-fcm.

35. GOOGLE. *About FCM messages* [online]. San Francisco, USA [cit. 2019-04-11]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/concept-options.

36. GOOGLE. *Receive messages in an Android app* [online]. San Francisco, USA [cit. 2019-04-11]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/android/receive.

37. GOOGLE. *Develop Android apps with Kotlin* [online]. San Francisco, USA [cit. 2019-03-18]. Dostupné z: https://developer.android.com/kotlin.

38. SUN MICROSYSTEMS. *JAVASOFT SHIPS JAVA 1.0* [online]. Santa Clara, CA, USA, 2007 [cit. 2019-03-18]. Dostupné z: https://web.archive.org/web/20070310235103/http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml.

39. JETBRAINS. *Kotlin 1.0 Released: Pragmatic Language for JVM and Android* [online]. Praha, ČR, 2016 [cit. 2019-03-18]. Dostupné z: https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/.

40. DMITRY, Jemerov; SVETLANA, Isakova. *Kotlin in action*. Manning, 2017. ISBN 1617293296, 9781617293290.

41. JETBRAINS. *Null Safety* [online]. Praha, ČR, 2016 [cit. 2019-03-24]. Dostupné z: https://kotlinlang.org/docs/reference/null-safety.html.

42. GOOGLE. *Fundamentals of Testing* [online]. San Francisco, USA [cit. 2019-04-14]. Dostupné z: https://developer.android.com/training/testing/fundamentals.

43. GOOGLE. *Test your App* [online]. San Francisco, USA [cit. 2019-04-14]. Dostupné z: https://developer.android.com/studio/test.

44. SQUARE INC. *Introduction to Retrofit* [software]. San Francisco, USA [cit. 2019-03-18]. Dostupné z: https://square.github.io/retrofit/.

45. KODEINKODERS. *Kodein DI on Android* [software] [cit. 2019-04-25]. Dostupné z: https://github.com/Kodein-Framework/Kodein-DI/blob/master/doc/android.adoc.

46. MOCKITO CONTRIBUTORS. *Mockito v2.27.0* [software]. 2018 [cit. 2019-04-14]. Dostupné z: https://github.com/mockito/mockito/wiki.

47. HAARMAN, Niek. *Mockito v2.27.0* [software]. 2018 [cit. 2019-04-14]. Dostupné z: https://github.com/nhaarman/mockito-kotlin/wiki.

48. GOOGLE. *Robolectric* [software] [cit. 2019-04-25]. Dostupné z: https://github.com/robolectric/robolectric.

49. GOOGLE. *Espresso* [online]. San Francisco, USA [cit. 2019-04-25]. Dostupné z: https://developer.android.com/training/testing/espresso.

50. VMWARE, INC. *What is MDM and MAM* [online] [cit. 2019-05-06]. Dostupné z: https://www.air-watch.com/faq/what-is-emm.

51. GOOGLE. *Processes and threads overview* [online]. San Francisco, USA [cit. 2019-04-25]. Dostupné z: https://developer.android.com/guide/components/processes-and-threads.

52. GOOGLE. *Create and Manage Notification Channels* [online]. San Francisco, USA [cit. 2019-04-15]. Dostupné z: https://developer.android.com/training/notify-user/channels.

53. GOOGLE. *Material Components for Android* [software]. 2019 [cit. 2019-04-25]. Dostupné z: https://github.com/material-components/material-components-android.

54. DJANGO SOFTWARE FOUNDATION. *Django* [software] [cit. 2019-05-06]. Dostupné z: https://github.com/django.

55. DIXON, Trevor. *Expose local servers to the internet* [online] [cit. 2019-05-06]. Dostupné z: https://serveo.net/.

56. JAKOB, Nielsen; K., Landauer Thomas. A Mathematical Model of the Finding of Usability Problems. In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* [online]. Amsterdam, The Netherlands: ACM, 1993, s. 206–213 [cit. 2019-05-12]. CHI '93. ISBN 0-89791-575-5. Dostupné z DOI: 10.1145/169059.169166.

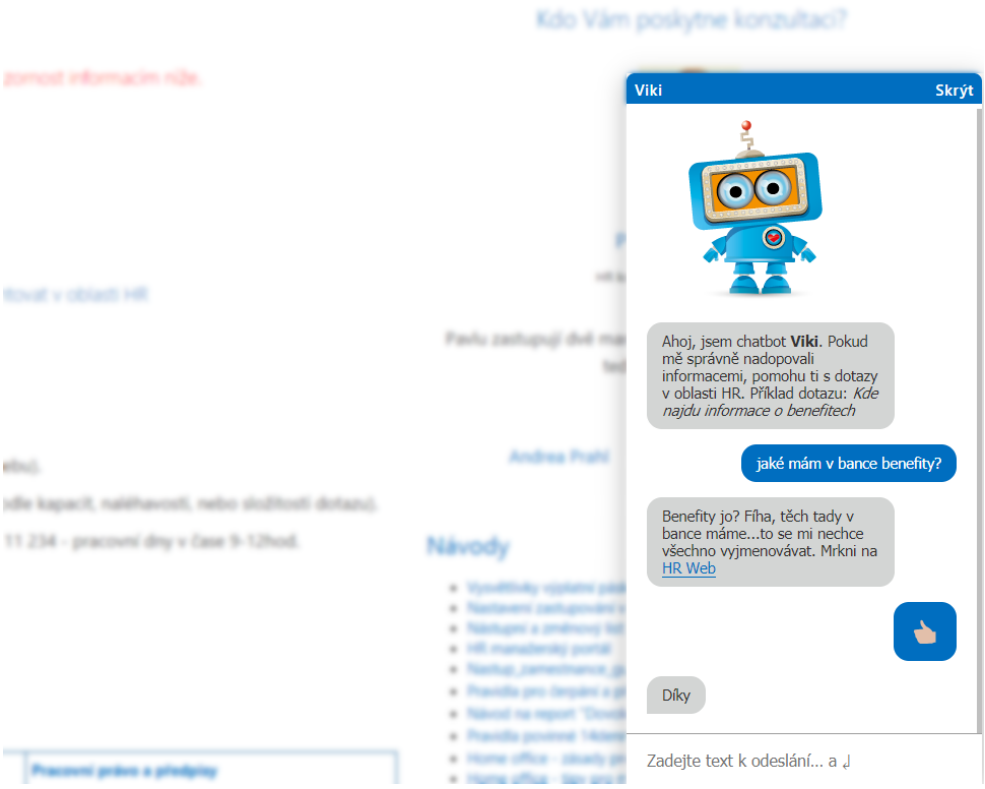
57. KUTÁLEK, Petr. *Distribuce aplikací v ČSOB*. [elektronická pošta]. Zasláno z pkutalek@csob.cz. 6. 5. 2019 20:51. [cit. 2019-05-06]. Osobní komunikace.

58. HECORAT. *AZ Screen Recorder - No Root* [online] [cit. 2019-05-02]. Dostupné z: https://play.google.com/store/apps/details?id=com.hecorat.screenrecorder.free&hl=cs.

59. ČSOB. *Instalační a uživatelská příručka pro ČSOB* [intranet ČSOB] [cit. 2019-05-10].

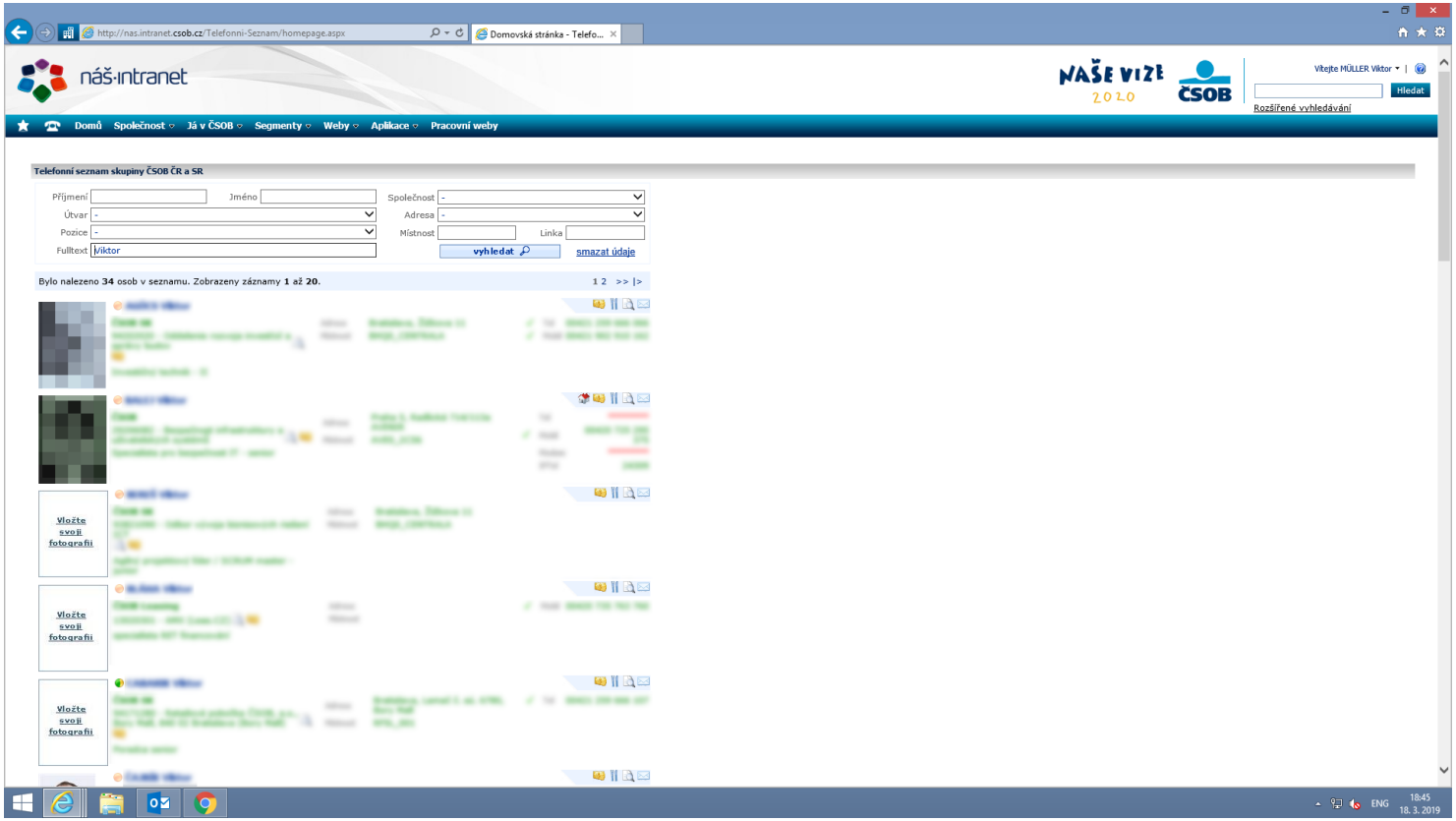
Příloha A

Ukázky stávajícího řešení



Obrázek A.1: Chatbot pro dotazy na HR linku

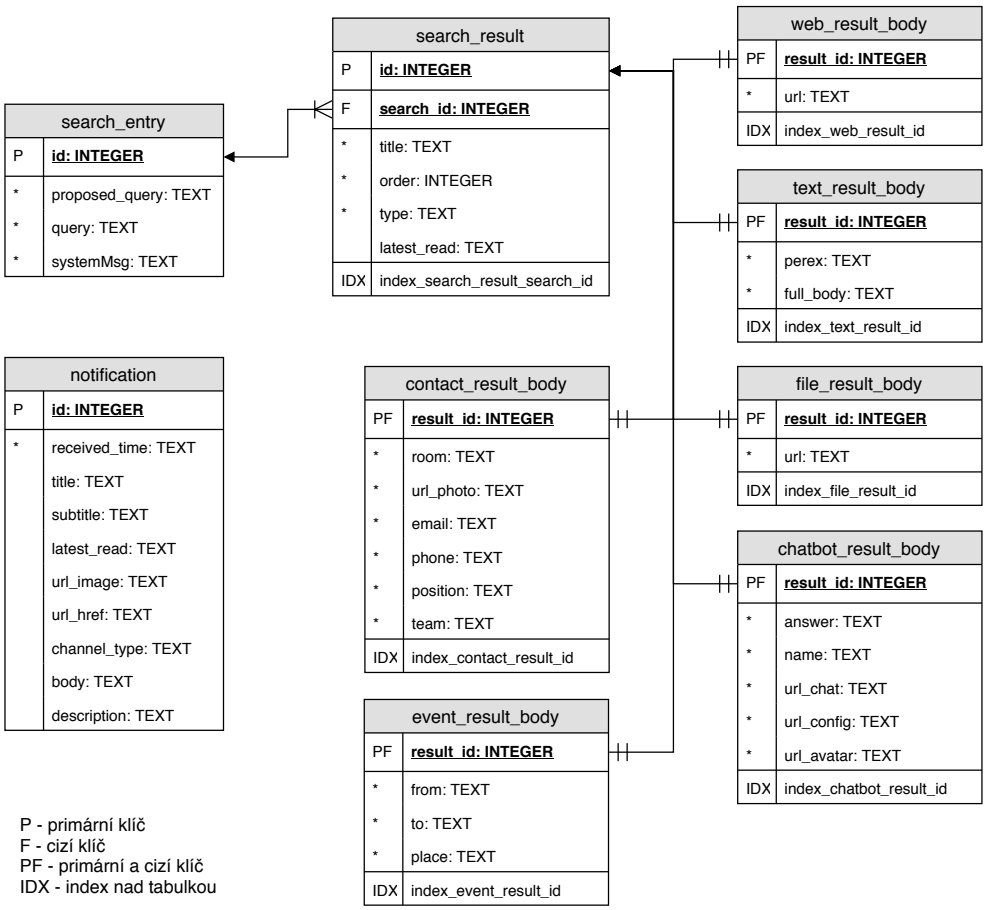
Obrázek A.2: Stávající vyhledávání na bankovním intranetu



Obrázek A.3: Stávající telefonní seznam

Příloha B

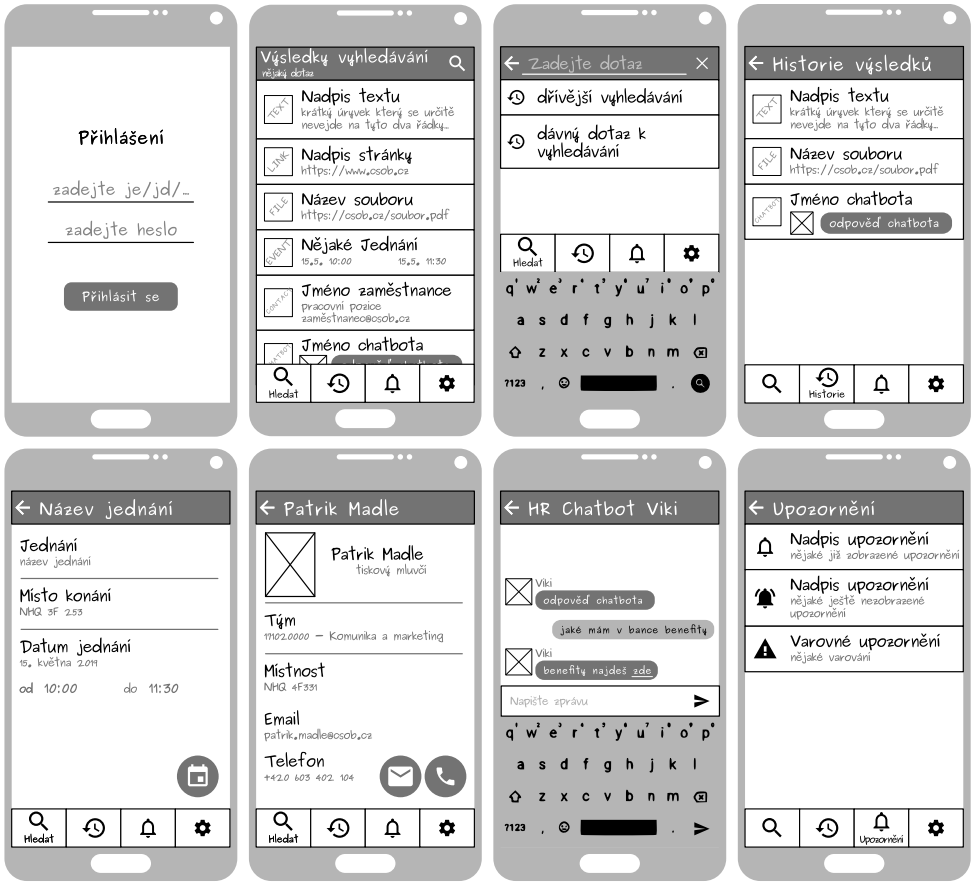
Databázový model



Obrázek B.1: Databázový model pro SQLite

Příloha C

Návrh obrazovek aplikace



Obrázek C.1: Drátové modely obrazovek

Příloha D

Testování uživatelského rozhraní na zaměstnancích

D.1 Průběh

Dne 29. 5. 2019 proběhlo testování aplikace šesti zaměstnanci ČSOB. Cílem tohoto testování bylo shromáždit jejich dojmy z uživatelského rozhraní aplikace pro případné změny v chování uživatelského rozhraní.

Každý respondent dostal za úkol splnit jednoduchý scénář za pomoci této aplikace. S aplikací přišel respondent do styku úplně poprvé. Každý zaměstnanec měl pouze následující informaci o testované aplikaci: „Aplikace slouží k vyhledávání interních informací napříč zaměstnaneckými systémy ČSOB.“

Scénář obsahoval následující kroky (začínalo se na domovské obrazovce se seznamem aplikací v zařízení):

1. otevřít aplikaci a přihlásit se testovacími údaji,
2. poslat email tiskovému mluvčímu ČSOB,
3. otevřít právě obdržené upozornění obsahující zprávy z banky a smazat již zobrazená upozornění,
4. upravit nastavení informačního kanálu pro upozornění,
5. zjistit číslo místnosti, ve které lze zastihnout tiskového mluvčího,
6. stáhnout si grafický manuál ČSOB.

Jednotlivé položky měly v ideálním případě prověřit následující funkcionality aplikace (číslo položky scénáře odpovídá číslu položky testované funkcionality).

1. Co nejrychleji identifikovat vyvíjenou aplikaci mezi ostatními aplikacemi v zařízení a bez potíží vyplnit přihlašovací formulář.
2. Nalézt ovládací prvky vyhledávání a zorientovat se mezi různými výsledky pro dotaz „tiskový mluvčí“. Pochopit, jak poslat zaměstnanci email pomocí jednoho tlačítka.
3. Identifikovat upozornění pro vyvíjenou aplikaci v horním panelu Android zařízení a prověřit manipulaci se seznamem s upozorněními v aplikaci.
4. Přejít do nastavení aplikace a nalézt tam nastavení pro upozornění.
5. Využít obrazovku s historií vyhledávání a zobrazit opět zobrazit kartu zaměstnance.
6. Zadat další dotaz pro vyhledávání a otevřít správný výsledek obsahující soubor ve formátu PDF.

Obrazovka zařízení byla při průchodu scénářem nahrávána pomocí aplikace AZ Screen Recorder [58]. Po průchodu scénářem byl každý respondent vyzván k vyplnění krátkého dotazníku skládajícího se z následujících dotazů.

A. Byly ovládací prvky aplikace na očekávaných místech?
   - Rozhodně ano.
   - Spíše ano.
   - Spíše ne.
   - Rozhodně ne.

B. Jak hodnotíte přehlednost aplikace?
   - V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.
   - Chvílemi jsem nevěděl/a, co se po mě chce. Aplikace nepůsobila přehledně.
   - V aplikaci jsem se často ztrácel/a.

C. Narazili jste na něco nejasného při průchodu scénářem?
   - *Otevřená odpověď.*

D. Jak na Vás působila vizuální stránka aplikace?
   - Parádní vzhled.
   - Šlo se na to koukat.
   - Spíše ošklivá.

- *Jiná odpověď.*

E. Všimli jste si při používání aplikace nějakých chyb v uživatelském rozhraní?

- *Otevřená odpověď.*

D.2 Výsledky

V této sekci jsou výsledky formuláře pro jednotlivé zaměstnance (písmeno odpovědi se schoduje s již zmíněným seznamem otázek).

Respondent A

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. *Odpověď nevyplněna.*

D. Parádní vzhled.

E. „Na úvodní obrazovce nebylo jasné, na jakou lupu jsem měl kliknout. První jsem viděl tu v menu, ta ale nešla použít.“

Respondent B

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. „hromadné smazání notifikací“

D. Šlo se na to koukat.

E. „zalomení textu pozice pracovníka“

Respondent C

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. „Problém se zadáním hesla, protože bylo překryto klávesnicí.“

D. Šlo se na to koukat.

E. *Odpověď nevyplněna.*

Respondent D

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. *Odpověď nevyplněna.*

D. Parádní vzhled.

E. „menu lišta nad klávesnicí“

Respondent E

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. „prihlasovanie“

D. „nevsimol som si nic specialne“

E. „prihlasovanie“

Respondent F

A. Rozhodně ano.

B. V každou chvíli bylo jasné, co jsem měl/a dělat. V aplikaci bylo snadné se vyznat.

C. „Z nastavení nebylo jasné jak změnit zvuky upozornění.“

D. Parádní vzhled.

E. *Odpověď nevyplněna.*

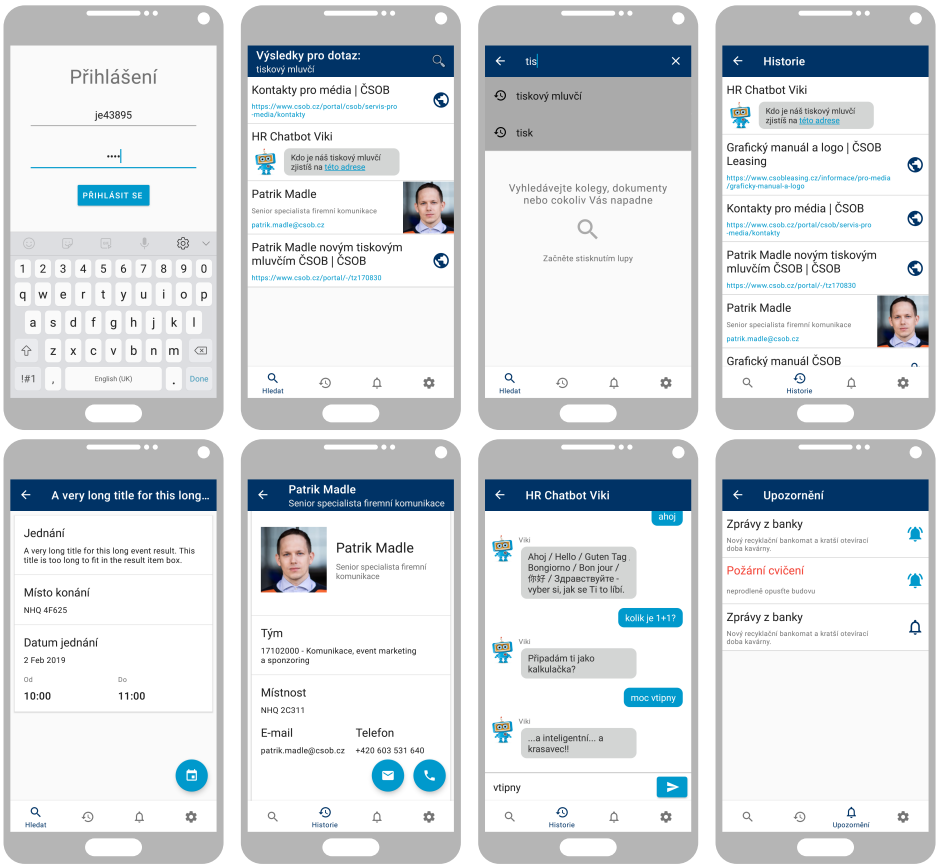
D.3 Technické údaje

Průchod scénářem byl prováděn na zařízení Samsung Galaxy A5 (Android OS verze 8.0.0). Aplikace používala skutečnou lokální databázi, skutečnou správu uživatelských účtů v zařízení, skutečné reagování na příchozí upozornění a skutečné síťové služby pro komunikaci se serverem.

Server byl pouze testovací s připravenými výsledky vyhledávání (implementován Python serverem popsaným v kapitole 6.5.4). Veškeré komunikace se serverem odpovídala skutečnému návrhu aplikačního rozhraní (včetně autorizačních hlaviček kontrolovaných testovacím serverem).

Příloha E

Vzhled výsledných obrazovek aplikace



Obrázek E.1: Vybrané obrazovky výsledné aplikace

Příloha **F**

Příručky

F.1 Instalační příručka

Tato instalační příručka rozšiřuje interní instalační příručku aplikací na bankovní zařízení v ČSOB [59]. Použité obrázky obrazovek jsou převzaty ze zmíněné instalační příručky.

F.1.0.1 Před instalací

Zařízení musí splňovat následující požadavky:

- operační systém Android verze 5.1 nebo vyšší,
- připravený účet Google v zařízení,
- nainstalovanou aplikaci *Airwatch (Intelligent Hub)*,
- připojení k síti přes datové služby nebo wifi.

Zaměstnanec musí mít v systému zaměstnaneckých oprávnění (ITIM) roli „GLOW_PRO_CZ_mobilní synchronizace (MDM) – standard“ pro možnost využívání aplikace *Airwatch (Intelligent Hub)*.

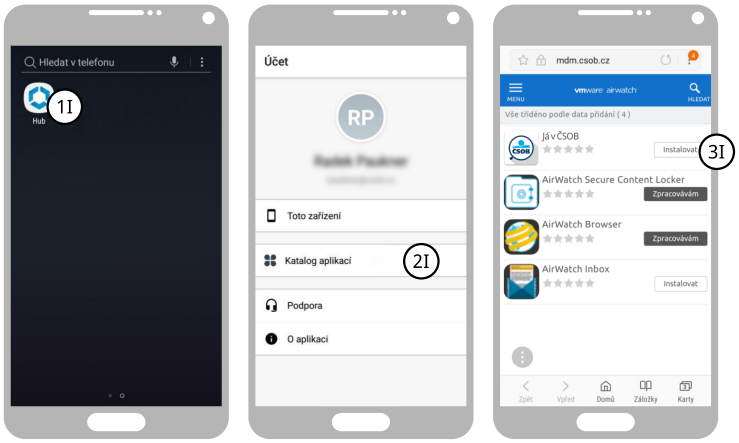
F.1.0.2 Instalace

Instalaci aplikace *Já v ČSOB* provádí uživatel v aplikaci *Airwatch (Intelligent Hub)*. Tu lze spustit stisknutím ikonky aplikace „Hub“ **1I**.

V aplikaci *Airwatch (Intelligent Hub)* pokračujte volbou „Katalog aplikací“ **2I** a následným přihlášením pomocí uživatelského jména (je/jd/...) a hesla do systémů GLOW.

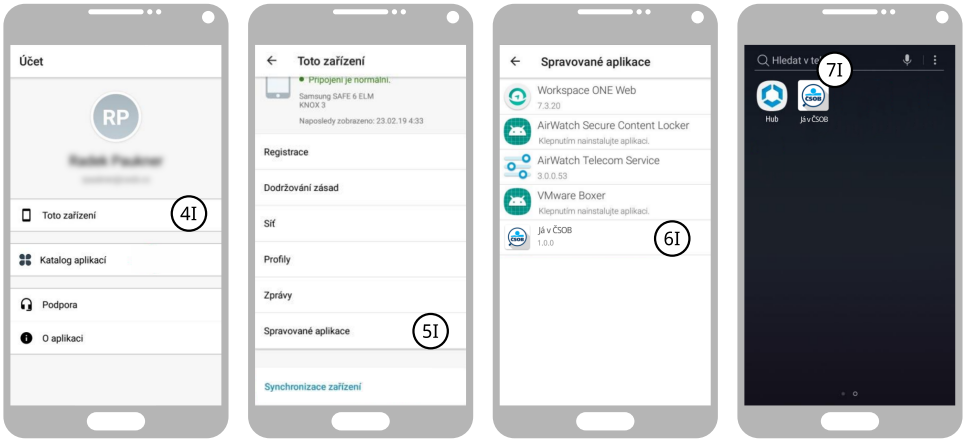
Zobrazí se seznam dostupných aplikací. Vyberte aplikaci „Já v ČSOB“ a stiskněte tlačítko „Instalovat“ **3I**. Následně potvrďte tlačítkem „Instalovat“.

Instalace probíhá na pozadí bez dalšího zásahu. Průběh instalace je možné sledovat v panelu s upozorněními systému Android.



Obrázek F.1: Instalace v aplikaci *Airwatch (Intelligent Hub)*

I po dokončení instalace se bude aplikace *Já v ČSOB* v seznamu aplikací zobrazovat ve stavu „Zpracovávám“. To, že byla instalace dokončena poznáte tak, že zmizí instalační upozornění z upozorňovacího panelu systému Android.



Obrázek F.2: Kontrola instalace v aplikaci *Airwatch (Intelligent Hub)*

Po dokončení instalace se vraťte do aplikace *Airwatch (Intelligent Hub)* a zkontrolujte, že pod volbou „Toto zařízení“ **4I** a následně „Spravované aplikace“ **5I** máte novou položku: „Já v ČSOB“ **6I**.

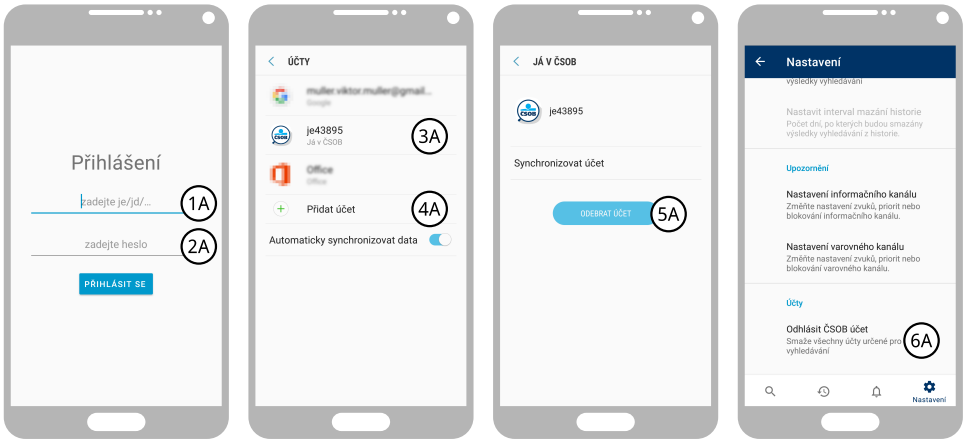
Tímto jste úspěšně dokončili instalaci aplikace *Já v ČSOB*. Spustit aplikaci můžete pomocí ikonky „Já v ČSOB“ **7I**, která byla přidáná na domovskou

obrazovku s aplikacemi. Jak s touto aplikací zacházet se dozvíte v uživatelské příručce F.2.

F.2 Uživatelská příručka

Uživatelská příručka mobilní aplikace *Já v ČSOB* pro zaměstnance ČSOB.

F.2.1 Přihlášení



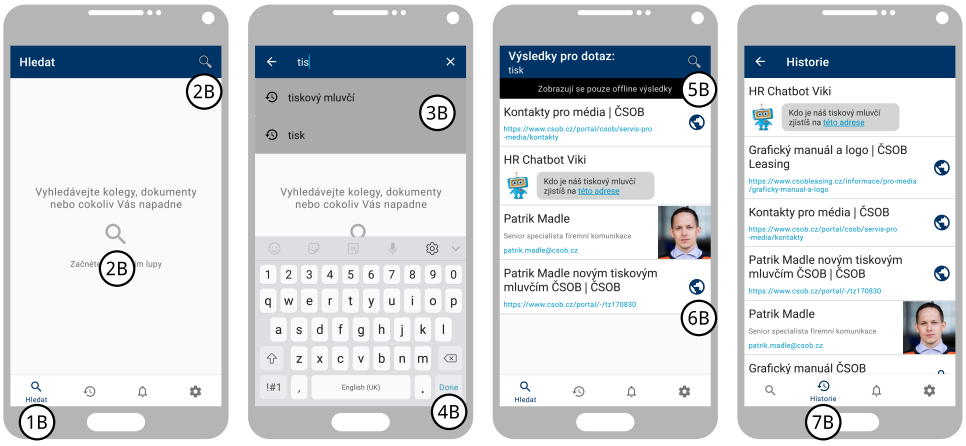
Obrázek F.3: Správa účtů pro aplikaci

Funkcionality aplikace vyžadují přihlášení. Po prvním spuštění aplikace je uživatel vyzván k poskytnutí přihlašovacích údajů. Přihlašovací údaje jsou stejné jako do systémů GLOW (tedy uživatelské jméno **1A** v podobě ČSOB identifikátoru je/jd/... a systémové heslo **2A**).

Aplikace umožňuje v jednu chvíli spravovat pouze jeden účet. V případě potřeby změny uživatelského účtu je potřeba předchozí účet nejprve odstranit. To lze buď v nastavení operačního systému Android v sekci *uživatelské účty*, kde lze účet aplikace vybrat **3A** a následně odstranit **5A**, nebo přímo v nastavení aplikace *Já v ČSOB* **6A**. Berte na vědomí, že odstranění aktuálního účtu smaže všechna ukládaná data touto aplikací (například historii výsledků vyhledávání). Nový účet pro tuto aplikaci lze vytvořit opět v nastavení Android zařízení **4A**, nebo opětovným spuštěním aplikace *Já v ČSOB*.

F.2.2 Vyhledávání informací

Úvodní obrazovka aplikace **1B** slouží k vyhledávání interních informací. Začít vyhledávat lze stiskem libovolného z tlačítek **2B**.



Obrázek F.4: Vyhledávání pomocí aplikace

Při psaní hledaného textu lze zvolit jeden z historických výsledků vyhledávání **3B**. Vyhledání nového dotazu lze spustit pomocí tlačítka **6B** na klávesnici zařízení.

V případě, že aplikace nemá přístup k síti, pokusí se vyhledat výsledky alespoň v historických datech minulých vyhledávání. Takto vyhledané výsledky však mohou být neaktuální, což je naznačené pomocí zprávy **5B**.

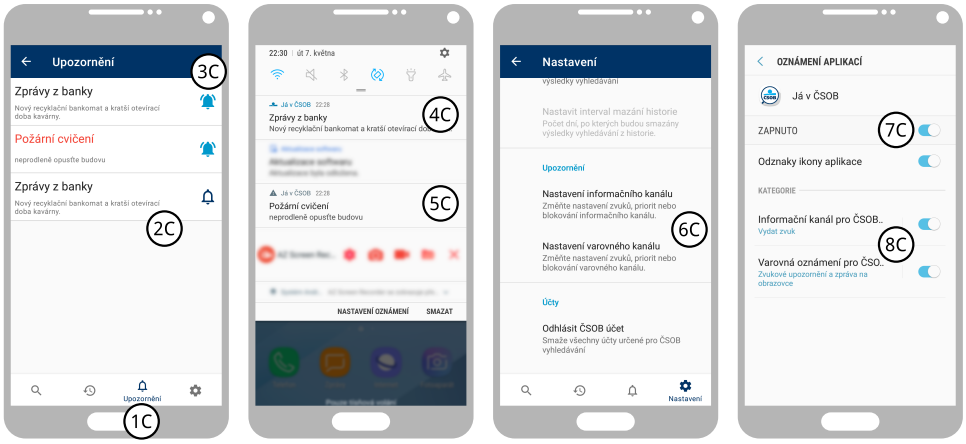
Jednotlivé výsledky jsou zobrazeny v posuvném seznamu **6B**. Bližší informace o konkrétním výsledku lze zobrazit pomocí stisknutí příslušné položky seznamu.

Každý výsledek vyhledávání, který byl navštíven je zobrazen v historii vyhledávání **7B**. Bližší informace o historickém výsledku lze zobrazit opět pomocí stisknutí příslušné položky seznamu. Odstranit položku z historie lze pomocí přetažení položky do pravé strany obrazovky nebo pomocí dlouhého stisku konkrétní položky.

F.2.3 Příjem upozornění

Obrazovka **1C** zobrazuje všechna obdržená upozornění v posuvném seznamu **2C**. Upozornění s červeným nadpisem značí varovná upozornění. Ostatní upozornění jsou z informačního kanálu.

Modrý vyplněný obrázek zvonečku **3C** značí doposud nezobrazená upozornění. Obsah upozornění je možné zobrazit pomocí stisknutí dané položky seznamu. Označit upozornění za zobrazené bez nutnosti otevření obsahu lze pomocí stisknutí zmíněného zvonečku. Zobrazit upozornění je také možné přímo z upozorňovacího panelu Android zařízení, jak je naznačeno na **4C** a **5C**.



Obrázek F.5: Příjem upozornění aplikací

Nastavení chování příchozích upozornění lze v nastavení operačního systému Android. Rychlý způsob, jak se dostat k tomuto nastavení je přímo z nastavení aplikace *Já v ČSOB* **6C**. Android umožňuje obecně blokovat všechna příchozí upozornění pro danou aplikaci **7C** nebo upravovat nastavení, jako je priorita nebo zvuky upozornění, pro konkrétní kanály **8C**.

Příloha **G**

Seznam použitých zkratek

AD DS Microsoft Active Directory Domain Services - název služby společnosti Microsoft

API Application Programming Interface - rozhraní pro programování aplikací

APK Android Package - instalační soubor aplikace pro Android

DAO Data Access Object - objekt poskytující rozhraní pro přístup k datům databáze

FCM Firebase Cloud Messaging - název služby společnosti Google

F *X* Functional Requirement number X - funkční požadavek číslo X

GUI Graphical User Interface - grafické uživatelské rozhraní

HTTP Hypertext Transfer Protocol - internetový protokol pro přenos hypertextových dokumentů

HTTPS Hypertext Transfer Protocol Secure - zabezpečená verze internetového protokolu HTTP

JSON JavaScript Object Notation - formát zápisu objektu jazyka JavaScript

JVM Java Virtual Machine - Modul virtuálního stroje ke spuštění programů napsaných v jazyce Java nebo Kotlin.

MDM Mobile Device Manager - nástroj pro centrální správu mobilních zařízení

MAM Mobile Application Manager - nástroj pro centrální správu aplikací na mobilních zařízeních

MVVM Model View ViewModel - architektonický vzor

N *X* Non-functional Requerement number X - nefunkční požadavek číslo X

SDK Software Development Kit - sada nástrojů pro vývoj software

SQL Structured Query Language - strukturovaný dotazovací jazyk

UC *X* Use Case number X - případ užití číslo X

XML Extensible Markup Language - rozšiřitelný značkovací jazyk

Příloha **H**

Obsah přiloženého CD

```
/
├── impl ..... implementační část práce
│   ├── ja_v_csob.apk ..... instalační soubor aplikace
│   ├── src_csob_app ..... zdrojové kódy implementace Android aplikace
│   ├── src_test_server ..... zdrojové kódy testovacího Python serveru
│   └── user_testing ..... video záznamy z uživatelského testování
└── thesis
    ├── src_thesis ..... zdrojové soubory práce
    └── thesis.pdf ..... text práce ve formátu PDF
```