

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA
Študijný odbor: **Informatika**

Jozef Chmelár
Manažérsky systém pre správu projektov

Vedúci: **Ing. Viliam Tavač, PhD.**

Reg.č. 237/2015

ŽILINA 2017

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v bakalárskej práci som uviedol v súlade s predpismi.

Súhlasím so zverejnením práce a jej výsledkov.

V Žiline, dňa 28.3.2017

.....

Jozef Chmelár

Podakovanie

Týmto by som chcel poďakovať všetkým, ktorí mi pomohli pri vypracovaní bakalárskej práce. Ďakujem vedúcemu bakalárskej práce Ing. Viliam Tavač, PhD. za odbornú pomoc, pripomienky a usmerňovanie pri tvorbe práce.

V Žiline, dňa 28.3.2017

.....

Jozef Chmelár

Abstrakt

CHMELÁR JOZEF: *Manažérsky systém pre správu projektov* [Bakalárska práca]

Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra softvérových technológií

Vedúci práce - Ing. Viliam Tavač, PhD. Stupeň odbornej kvalifikácie - Bakalár.- Žilina: FRI ŽU v Žiline, 2017 - Počet strán 42

Cieľom bakalárskej práce je navrhnúť a vytvoriť aplikáciu pre Android, zameranú na správu projektov ktorá umožňuje efektívnejšiu komunikáciu medzi všetkými riešiteľmi projektu vo firme.

Kľúčové slová: Android, aplikácia, projekt manažment

Abstract

CHMELÁR JOZEF: *Project's manager system for project manager* [Bachelor thesis]

University of Žilina, Faculty of Management Science and Informatics, Department of Software Technologies

Supervisor: Ing. Viliam Tavač, PhD. - Qualification level : Bachelor.- Žilina: FRI ŽU in Žilina ,2017 - Number of pages: 42

Goal of the bachelor thesis is to design and create Android application which will provide a more efficient way to communicate between all collaborators of projects within a company.

Keywords: Android, application, project management

Obsah

| | |
|---|-----------|
| Zoznam obrázkov | 9 |
| Zoznam tabuliek | 10 |
| Zoznam skratiek a termínov | 11 |
| Úvod | 12 |
| 1 Súčasný stav | 13 |
| 1.1 História | 14 |
| 1.2 Verzie Androidu | 14 |
| 2 Použité technológie | 16 |
| 2.1 Nástroje | 16 |
| 2.2 Android Studio | 17 |
| 2.3 IntelliJ Idea | 17 |
| 2.4 Kotlin | 17 |
| 2.5 ReactiveX | 18 |
| 2.6 Cloudové služby | 18 |
| 2.6.1 Heroku | 18 |
| 2.6.2 OpenShift | 19 |
| 3 Analýza | 20 |
| 3.1 Klient-server | 20 |
| 3.2 Základné komponenty aplikácie | 20 |
| 3.2.1 Aktivita | 20 |
| 3.2.2 Služba | 21 |
| 3.2.3 Poskytovateľ obsahu | 22 |
| 3.3 Aktuálne riešenia | 22 |
| 3.3.1 JIRA Software | 23 |
| 3.3.2 Slack | 24 |

| | | |
|----------|--|-----------|
| 3.4 | Zber požiadaviek | 25 |
| 4 | Návrh a implementácia aplikácie | 27 |
| 4.1 | Architektúra aplikácie | 27 |
| 4.2 | Prihlasovacia obrazovka | 29 |
| 4.3 | Obrazovka projektov | 31 |
| 4.4 | Detail projektov | 33 |
| 4.4.1 | Stav | 34 |
| 4.4.2 | Riešitelia | 35 |
| 4.4.3 | Služobné cesty | 36 |
| 4.5 | Serverová časť | 37 |
| 4.5.1 | Bezpečnosť | 38 |
| 4.5.2 | Databáza | 38 |
| | Záver | 40 |
| | Literatúra | 41 |

Zoznam obrázkov

| | | |
|------|--|----|
| 1.1 | Podiel verzií Androidu | 13 |
| 2.1 | Pozorovateľ - Vydavateľ - bdiagram tried | 18 |
| 3.1 | Životný cyklus aktivity | 21 |
| 3.2 | Životný cyklus služby | 22 |
| 3.3 | Kanban board | 23 |
| 3.4 | Slack, hlavné okno | 25 |
| 3.5 | Prípady použitia | 26 |
| 4.1 | Diagram nasadenia | 27 |
| 4.2 | Porovnanie MVP s MVC | 28 |
| 4.3 | Nasadenie MVP v Androide | 29 |
| 4.4 | Obrazovka prihlásenia | 30 |
| 4.5 | Sekvenčný diagram - Prihlasovanie | 31 |
| 4.6 | Obrazovka projektov | 32 |
| 4.7 | Obrazovka vytvorenia projektu | 33 |
| 4.8 | Obrazovka stavov | 34 |
| 4.9 | Obrazovka riešiteľov | 35 |
| 4.10 | Obrazovky stretnutia | 36 |
| 4.11 | Detail služobnej cesty | 37 |
| 4.12 | Model databázy | 39 |

Zoznam tabuliek

| | | |
|-----|--|----|
| 2.1 | Percentuálny podiel jazykov v AOSP | 16 |
| 3.1 | REST komunikácia | 20 |

Zoznam skratiek a termínov

| | |
|-------------|---|
| AOKP | Android Open Kang Project |
| AOSP | Android Open Source Project |
| API | Application programming interface |
| APK | Android package |
| ART | Android Runtime Enviroment |
| GUI | Graphical User Interface |
| JDK | Java Development Kit |
| JRE | Java Runtime Enviroment |
| JSON | JavaScript Object Notation |
| JVM | Java Virtual Machine |
| JWT | JSON Web Token |
| MIUI | Mi User Interface |
| MVP | Model, View, Presenter |
| ORM | Object-relational mapping |
| PaaS | Platform as a service |
| REST | Representational state transfer |
| SDK | Software Development Kit - Súbor nástrojov pre vývoj softvéru |
| SSH | Kryptovaný sieťový protokol |
| URL | Jednotný vyhľadávač prostriedku |
| XML | eXtensible Markup Language |

Úvod

Žijeme v dobe kedy výkonné mobilné zariadenia nepatria medzi majetok vyvolených ľudí, ale takmer o samozrejmu výbavu každého človeka, či už sa jedná o študenta, alebo prezidenta. Je úžasné a zároveň úsmevné ako namiesto obyčajného budíka používame zariadenie s porovnateľným výkonom ako počítač z pred pár rokov. Veľa ľudí pred spánkom zapojí svoj telefón do nabíjačky, ráno ich zobudí a sprevádza ho celý deň. Objekty reálneho sveta ako poznámkový blok, telefónny zoznam, mapa, fotoaparát alebo prehrávač hudby nahradili rovnomenné aplikácie a všetky sa zmestia do vrečka nohavíc.

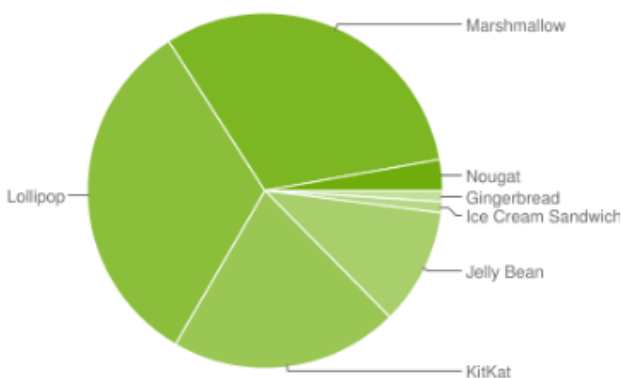
Počas pobytu v rýchlo rastúcej spoločnosti kde nie každý pracovník firmy mal prístup k počítaču, rôzne oddelenia mali vlastné systémy na internú komunikáciu, pričom neraz to bol obyčajný email, ale s rastom spoločnosti prestali dostačovať, a videl som potenciál práve na zariadeniach ktoré má väčšina zamestnancov, ich inštalácia a nasadzovanie netrvá mesiace, ale minúty. Rovnako som chcel zabezpečiť jednoduchý kontakt na rôznych riešiteľov projektov, keďže veľa ľudí na seba navzájom nemala kontakt, prístup na server k excel tabuľke s kontaktmi, o ktorej existencii viacerí zamestnanci ani nevedeli.

Táto práca sa zaoberá vytvorením mobilnej aplikácie pre Android, ktorá používa najznámejšie technológie v oblasti mobilného vývoja a umožní uľahčenie komunikácie medzi zamestnancami, ktorí nemajú svoj pracovný počítač, ale ich prácu by uľahčilo a zrýchlilo práve zariadenie, ktoré dávno majú.

1. Súčasný stav

Android je operačný systém ktorý je založený na licencií open source[1]. To znamená, že hocikto môže bezplatne sťahovať a distribuovať jeho zdrojový kód. To dospelo do súčasného stavu, kedy má prístup k výkonným mobilným zariadeniam väčšia časť populácie ako kedykoľvek predtým. Každá spoločnosť si môže upraviť Android podľa vlastných predstáv a poskytnúť zákazníkovi jedinečné prostredie. Okrem toho získali vývojári prístup k veľkému publiku, keďže vývojári nemuseli platiť za licencie, alebo vyvíjať vlastný operačný systém. To umožnilo vývojárom znížiť náklady na vývoj a od roku 2011 do 2013 klesla priemerná cena smartphonov na celom svete o 25% [2]. Pokiaľ by sme chceli stráviť jednu minútu prezeraním každého jedného zariadenia s Androidom, trvalo by nám to dva a pol týždňa bez spánku. Momentálne existuje na svete 24000 zariadení od takmer 1300 značiek [3]. Vďaka Androidu vznikli rôzne verzie, medzi ktoré patrí CyanogenMod, MIUI, AOKP momentálne je najpoužívanější verzia Androidu je 5.0-5.1 Lollipop s 32,5% podielom na trhu [4], ostatné verzie a ich podiel na trhu môžeme vidieť na grafe (obr.1.1.)

| Verzia | Názov | API | Podiel |
|---------------|--------------------|-----|--------|
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 1.0% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 1.0% |
| 4.1.x | Jelly Bean | 16 | 3.7% |
| 4.2.x | | 17 | 5.4% |
| 4.3 | | 18 | 1.5% |
| 4.4 | KitKat | 19 | 20.8% |
| 5.0 | Lollipop | 21 | 9.4% |
| 5.1 | | 22 | 23.1% |
| 6.0 | Marshmallow | 23 | 31.3% |
| 7.0 | Nougat | 24 | 2.4% |
| 7.1 | | 25 | 0.4% |



Obr. 1.1: Podiel verzií Androidu

Ponuka, ktorú nám Android prináša, nekončí pri počte zariadení, ale aj aplikáciami, ktorých je cez Google Play dostupných viac ako 1 milión [1]. Dostupné sú aj alternatívy ku Google a to Amazon Appstore, GetJar, Mobogenie, SlideME, F-Droid [5].

1.1 História

Spoločnosť Android Inc. bola založená v Palo Alto, California v Októbri 2003 [7]. Pri vzniku stáli Andy Rubin, Rich Miner, Nick Sears a Chris Whie s myšlienkou vytvoriť múdrejšie mobilné zariadenia, ktoré by si boli vedomé o svojej pozícii a používateľových preferenciách. Pôvodný úmysel spoločnosti bol vyvinúť pokročilý operačný systém pre digitálne fotoaparáty, ale malý trh ich naviedol na cestu vývoja pre mobilné zariadenia a konkurovať tak Symbianu a Windows Mobile. [6]

1.2 Verzie Androidu

Najvyššie číslo dostupnej verzie operačného systému Android je *7.0* a vzhľadom k tomu sa už staršie verzie vo veľkom počte nepoužívajú, preto sa budeme v nasledujúcich podkapitolách venovať len verziám s minimálnym podielom 15%

KitKat Vydaný v Októbri 2013 bol vylepšený oproti predošlým verziám o nástroj na analýzu pamäte, nahrávanie obrazovky, API k infračervenému blastru a audio monitoring. Od tejto verzie majú vývojári prístup k SMS API. [8]. Dalšie podverzie 4.4.1 až 4.4.4 priniesli hlavne bezpečnostné opravy.

Lollipop Najväčšiu zmenu, ktorú Lollipop priniesol ku koncu roka 2014 do androidu bol nový dizajn používateľského prostredia vybudovaný pomocou dizajnového jazyka *Material Design* ktorý je hlavným vizuálnym prvkom Googlu. Virtuálny stroj Dalvik bol nahradený Android Runtime[9]. Lollipop 5.0.1 - 5.1.1, pridali podporu viacerých SIM a ochranu zariadenia pri krádeži.

Marshmallow Pribudol správca povolení pre aplikácie. API ,ktoré umožňuje pristupovať k práve zobrazeným dátam na obrazovke. Od 6.0 Android natívne podporuje odtlačky prstov a USB-C a funkcia *deep sleep*, ktorá pomáha šetriť batériu v prípade potreby.

Nougat Predstavil výrazné zmeny. Pribudla možnosť zobrazit' viaceré aplikácie na jednej obrazovke, rozdelenie obrazovky na časti, instantné odpovede priamo v notifikačnom centre a optimalizačné prvky pre virtuálnu realitu.

2. Použité technológie

Operačný systém Android je naprogramovaný hlavne v jazyku Java [11] (tab. 2.1), rovnako ako aj aplikácie. Pre vývoj je potrebné mať nainštalované JDK spolu s SDK Manager, pomocou ktorého môžeme stiahnuť potrebné verzie Android SDK, obsahujúce nástroje, ktoré kompilujú kód spolu s inými časťami aplikácie ako sú obrázky, jazykové súbory do APK archívu, ktorý obsahuje všetky súbory potrebné na inštaláciu. Android je multi-user Linux systém, v ktorom je každá aplikácia definovaná ako používateľ. Každá aplikácia má priradené unikátne používateľské ID pomocou ktorého systém obmedzí povolenia pre aplikáciu. Každá aplikácia má svoj Linux proces, ktorý je vytvorený vždy, keď je potrebné spustiť komponent aplikácie, a je ukončený vždy keď proces nie je naďalej potrebný alebo systém potrebuje pamäť pre iné aplikácie.

| Jazyk | Podiel kódu |
|---------|-------------|
| Java | 46,1% |
| C | 30,8% |
| C++ | 13,4% |
| Ostatné | 9,7% |

Tabuľka 2.1: Percentuálny podiel jazykov v AOSP

2.1 Nástroje

Vývoj softvéru sa nezaobíde bez výberu správnych nástrojov. Nie sme nútení písať aplikácie v obyčajných textových editoroch, máme k dispozícii veľké množstvo prepracovaných vývojových nástrojov ktoré umožňujú rýchly re-factoring kódu, analýzu kódu, build systémy pre ľahší a rýchlejší vývoj. Za nástroje považujeme aj súčasné technológie, ktoré uľahčujú prácu s rôznymi časťami aplikácie a poskytujú nám abstrakciu, ktorá umožní písať kvalitný a čitateľný kód.

2.2 Android Studio

Od 26.Júna.2015 prestal Google oficiálne vyvíjať plugin pre IDE Eclipse [15] a Android Studio sa stalo jediným oficiálne podporovaným nástrojom na vývoj pre platformu Android. Na jeho vývoji sa podieľa Google a taktiež spoločnosť JetBrains ktorá vyvíja IntelliJ Idea, na ktorom je Android Studio založené. Medzi jeho výhody patria [16]

- Build nástroj Gradle, ktorý automaticky zostavuje aplikáciu spolu s knižnicami, alebo inými projektami
- Refaktoring špecifický pre Android
- ProGuard ktorý znižuje veľkosť aplikácie, maže zbytočný kód a komplikuje reverzné inžinierstvo
- Android špecifické šablóny
- GUI editor
- Hĺbková analýza kódu

2.3 IntelliJ Idea

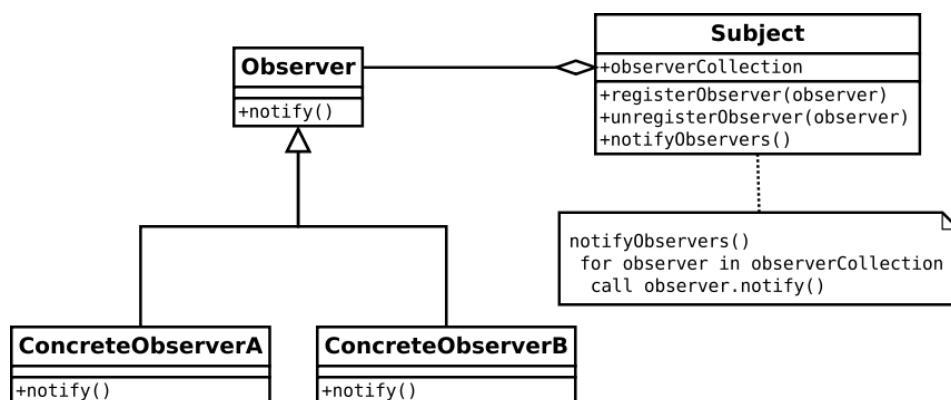
Android Studio a Android plugin pre IntelliJ Idea sú postavené na rovnakom kóde [17] pričom nedostaneme pred-konfigurovaný nástroj na vývoj priamo od inštalácie. Jedná sa o produkt, ktorý je primárne IDE pre Javu a J2EE. Projekty, vytvorené v Android Studio, je možné otvoriť v IntelliJ Idea a vice versa. Spoločnosť JetBrains poskytuje pluginy ktoré poskytujú lepšiu prácu s verzionovacím systémom alebo databázou.

2.4 Kotlin

Kotlin je staticky typovaný jazyk bežiaci nad JVM vyvíjaný *JetBrains*, ktorý síce nie je syntakticky kompatibilný s Javou, ale je navrhnutý tak, že umožňuje využívať všetky Java knižnice. Narozdiel od Javy si nevyžaduje veľa písania, podporuje lambda výrazy, dátové triedy a automaticky generuje metódy na prístup k premenným. Jedná sa o čoraz viac využívaný jazyk Android vývojármi [18].

2.5 ReactiveX

Reactive Extensions (Rx) je knižnica na vytváranie asynchrónnych udalostne založených programoch. Uľahčuje prácu pri súbežných výpočtoch a poskytuje abstrakciu, ktorá nás odbremení najmä od synchronizácií vlákien. Rx je založený na návrhovom vzore Pozorovateľ - Vydavateľ



Obr. 2.1: Pozorovateľ - Vydavateľ - bdiagram tried

Vždy ak vydavateľ zmení hodnotu, všetci pozorovatelia budú upozornení, čo je užitočné hlavne pri zobrazovaní informácií pre používateľa, každá zmena v premennej môže byť automaticky zobrazená používateľovi. Pri Http požiadavke môžeme na jednom vlákne sledovať stav požiadavky, pokiaľ požiadavka skončí, či už chybou alebo úspešne, sme schopní zobraziť informáciu používateľovi okamžite na UI vlákne.

2.6 Cloudové služby

Väčšina aplikácií komunikuje so vzdialeným serverom, ktorý odpovedá na požiadavky klienta. Nastavovanie serverovej časti aplikácie, inštalácia potrebného softvéru a následná údržba stojí čas, ktorý je pri tvore softvéru veľmi vzácny, preto sa na trhu začali objavovať PaaS služby ktoré, vývojárovi ušetria čas.

2.6.1 Heroku

Heroku je jedna z prvých cloudových platforiem, ktorá je vyvíjaná od roku 2007. Vtedy podporovala nasadenie aplikácií len v jazyku Ruby, dnes podporuje Javu, Node.js,

Scala, Clojure, Python, PHP, a Go. Heroku poskytuje jednoduché nasadenie aplikácie priamo z verzionovacieho systému Git, ktorý poskytuje prehľad všetkých zmien v kóde. Heroku poskytuje vlastný Git repozitár, alebo napojenie na GitHub, pričom pri aktualizácii hlavnej vetvy repozitáru sa aplikácia sama skompiluje a nainštaluje. [19]

2.6.2 OpenShift

OpenShift je PaaS produkt za ktorým stojí spoločnosť RedHat, ktorá poskytuje hosting s vývojársky prívetivými službami ako MongoDB, Node.js prístup cez webové rozhranie alebo príkazový riadok. Obsahuje všetky závislosti, potrebné na vývoj Java Enterprise aplikácií, integráciu s IDE a automatizačnými systémami ako Maven alebo Jenkins. OpenShift dbá na zabezpečenie, preto je možné komunikovať len pomocou SSH kľúča. Rovnako ako na Heroku, aplikácia sa builduje z Git repozitáru.

3. Analýza

3.1 Klient-server

Architektúra klient server rozdeľuje aplikáciu na dve časti. Serverová aplikácia, ktorá je spustená na výkonnom zariadení spracováva dáta, poskytuje veľkú pamäť, alebo databázu a klientskú časť, ktorá poskytuje používateľské prostredie na odosielanie požiadaviek a zobrazenie odpovede servera[20]. Android aplikácie najčastejšie používajú REST architektúru, ktorá využíva URL adresu a HTTP požiadavky na interakciu so serverom.

Tabuľka 3.1: REST komunikácia

| <i>URL/Http Metóda</i> | <i>GET</i> | <i>PUT</i> |
|----------------------------------|--|----------------------------|
| <i>uniza.sk/student</i> | Vráti kolekciu študentov | Nahradí kolekciu novou |
| <i>uniza.sk/student/1</i> | Zobrazí informácie o študentovi s id 1 | Nahradí staré údaje novými |
| <i>URL/Http Metóda</i> | POST | DELETE |
| <i>uniza.sk/student/1</i> | Vytvorí novú položku a vráti jej URI | Vymaže kolekciu |
| <i>uniza.sk/student/1</i> | Vo všeobecnosti sa nepoužíva. | Vymaže študenta z kolekcie |

Klient posiela požiadavky na server buď pomocou URL adresy alebo s dodatočnými dátami uložené v tele požiadavky vo formáte Json, alebo XML.

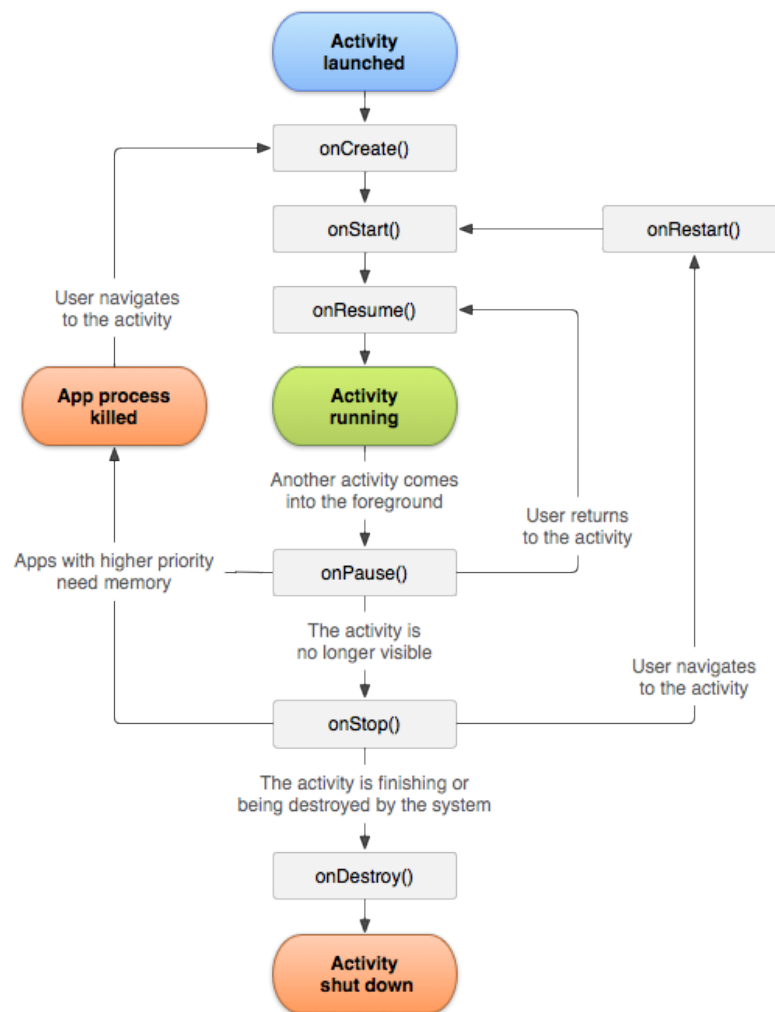
3.2 Základné komponenty aplikácie

Všetky prvky operačného systému sú prístupné pre programátora pomocou API, ktoré sú naprogramované v jazyku Java. Všetky prvky tvoria stavebné bloky, ktoré potrebujeme na vývoj aplikácie. Sú to znova použiteľné a modulárne komponenty.

3.2.1 Aktivita

Aktivita reprezentuje jednu obrazovku, ktorá obsahuje všetky komponenty nutné pre interakciu s aplikáciou. Aktivity sú v systéme spravované ako zásobník aktivít, nová aktivita je vždy pridaná na vrchol zásobníka, pričom predošlé aktivity ostávajú v

zásobníku a nebudú zobrazené kým neodídeme z vrchnej aktivity. Aktivita má štyri stavy. Pokiaľ je aktivita zobrazená na obrazovke je aktívna, alebo bežiaca. Ak aktivitu prekryje iné okno, ale je stále viditeľná tak je pozastavená, čo znamená, že celý jej stav je uložený a je stále aktívna. Ak je aktivita celá prekrytá novou, pôvodná je zastavená. Hoci si o sebe uchováva všetky informácie, nie je používateľovi viditeľná a môže byť kedykoľvek zmazaná z pamäte, ak systém potrebuje pamäť na inom mieste. Nasledujúci diagram (obr.3.1.) zobrazuje životný cyklus aktivity.

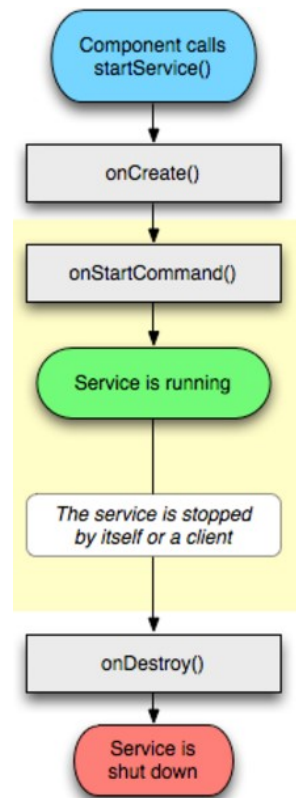


Obr. 3.1: Životný cyklus aktivity

3.2.2 Služba

Service, alebo služba, je aplikačný komponent, ktorý umožňuje vykonávať dlhotrvajúce činnosti na pozadí ako je napríklad prehrávanie hudby alebo získavanie dát z GPS.

Služba je spustená aplikáciou a ostáva spustená aj po jej ukončení. Má jednoduchý životný cyklus. Aplikácia ju spustí a beží až kým nie je zastavená operačným systémom, alebo spúšťajúcou aplikáciou 3.2.



Obr. 3.2: Životný cyklus služby

3.2.3 Poskytovateľ obsahu

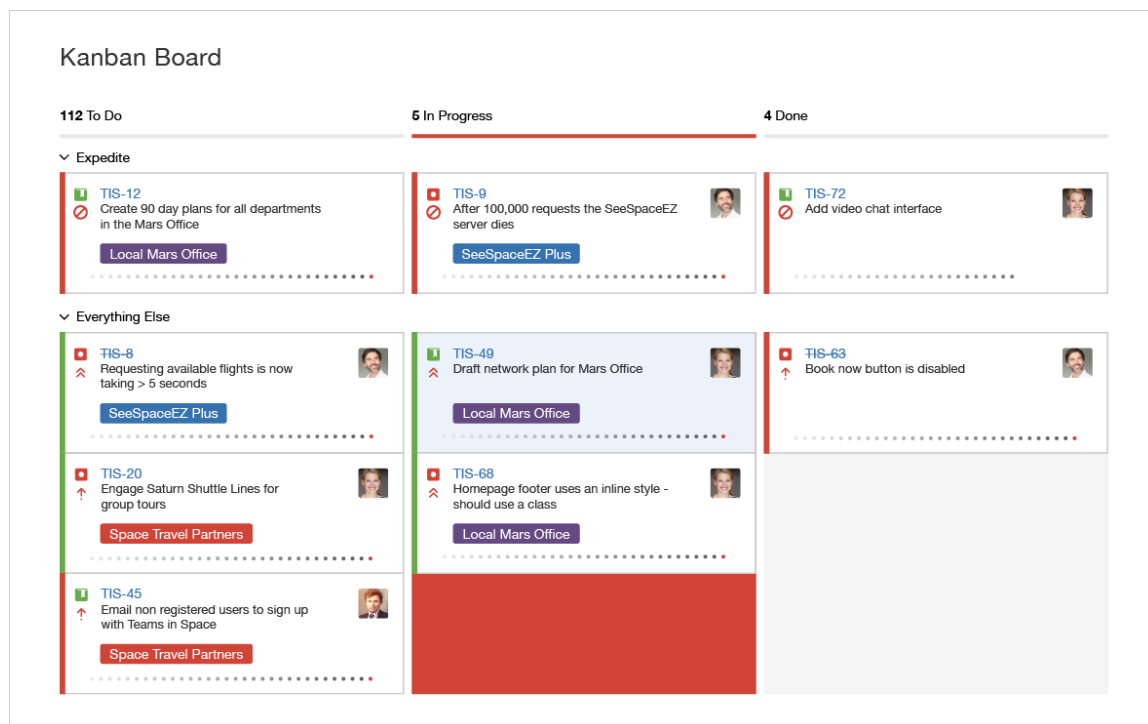
Poskytovateľ obsahuje spravuje dáta, ktoré môžu byť zdieľané napriek aplikáciami pomocou SQLite databázy. Ak má aplikácia práva, môže meniť obsah dát, ktoré sú dostupné, následne iná aplikácia dostane po vyžiadaní dát, nové a upravené hodnoty. Príklad využitia takejto služby je telefónny zoznam, kedy môže aplikácia tretej strany pristupovať k číslam, využívať ich, prípadne meniť.

3.3 Aktuálne riešenia

Cieľom tejto analýzy je porovnať dostupnosť, jednoduchosť a cenu súčasných najpoužívanejších riešení, ktoré sú dostupné na trhu.

3.3.1 JIRA Software

JIRA Software od spoločnosti Atlassian je software na správu projektov pre agilné tímy. Pomocou takzvaných Kanban boardov, ktoré umožňujú celému tímu vidieť aká práca ich v najbližšom období čaká, momentálne prebieha alebo je už dokončená. Jedná sa o tabuľku s kartičkami s obsahom práce, ktoré môžu následne členovia tímu premiestňovať do adekvátnych stĺpcov (obr.3.5.)



Obr. 3.3: Kanban board

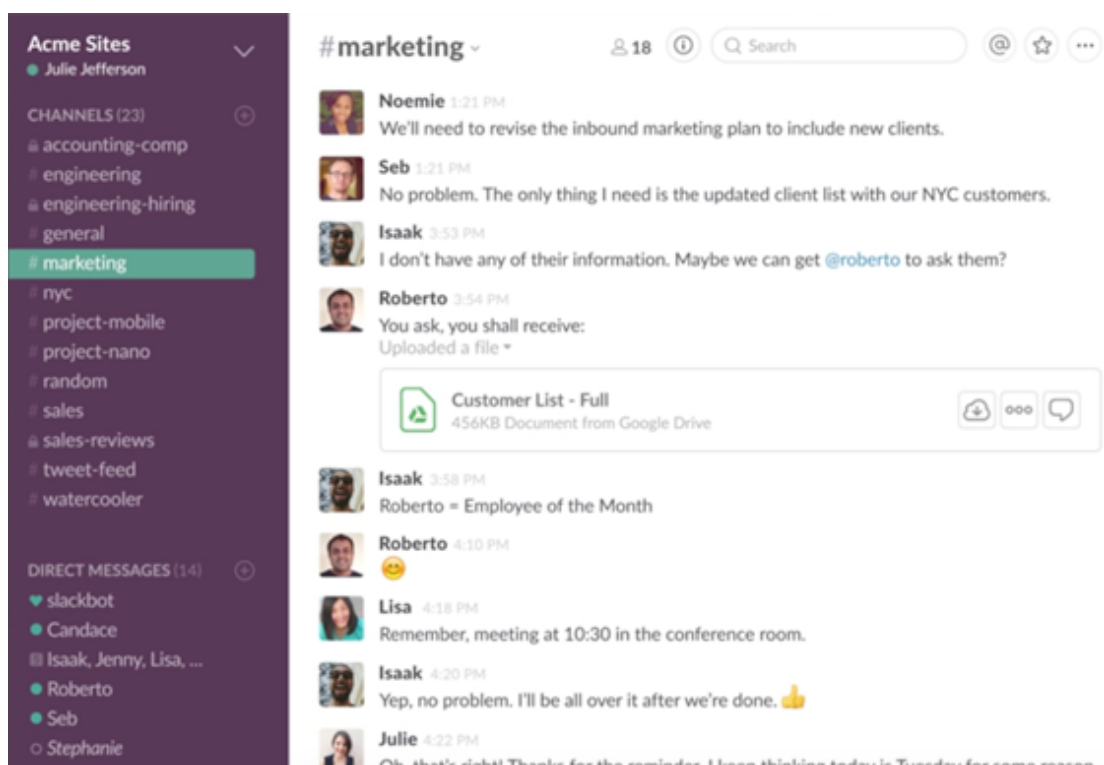
Každá kartička v tabuľke reprezentuje úlohu, ktorá má vždy zadávateľa a osobu, ktorá bude danú úlohu vykonávať a následne aktualizovať jej stav. Pri vytvorení a priradení úlohy príde riešiteľovi emailové oznámenie o priradení. Jedná sa o relatívne zdĺhavý proces, ktorý zamestnanci neradi vykonávajú. JIRA software je o mnoho obsiahlejší, tým pádom komplikovanejší, a napriek user-friendly prostrediu vyžaduje zaškolenie nového zamestnanca a firmy potrebujú vlastného JIRA správcu. Jedná sa ale o najobľúbenejšie riešenie pre projektový manažment ktorý umožňuje firmám lepšiu organizáciu pri riadení práce, ktorého výsledkom je kvalitný finálny produkt dodaný načas. Taktiež je rozširiteľný o ďalšie produkty spoločnosti ako sú Confluence, Bitbucket, Trello. Chýba im ale dedikovaná, jednoduchá a funkčná aplikácia pre mobilné telefóny

Atlassian ponúka tri verzie ich softvéru. Verzia, ktorá sa nainštaluje na vlastný server, verzia v data centre a cloudová inštancia ich produktu. Základná verzia JIRA Core pre 25 ľudí stojí \$1800 za serverovú aplikáciu a \$180 mesačne za cloudovú inštanciu .

3.3.2 Slack

Slack je nástroj na tímovú kolaboráciu, ktorý vznikol ako interný nástroj, a v súčasnej dobe sa jedná o cloudovú službu. Jedná sa v podstate o interný chat, ktorý je rozdelený do rôznych kanálov. Kanál (obr.3.4) reprezentuje tému, projekt, lokácie alebo skupinu ľudí.

Ľudia môžu pomocou Slack-u posilať správy aj konkrétnej osobe, uskutočňovať hovory, zdieľať súbory. Externé firmy, ktoré taktiež využívajú Slack, môžu využívať spoločný kanál. Najväčšou výhodou Slack-u je, že odpadáva nutnosť odosielať hromadné maily, v ktorých sa po čase stráca prehľad, sprehľadňuje email a prenecháva mu jeho dôležitú funkciu a to komunikáciu s dôležitými klientmi pre formálnejší spôsob posielania správ. Na rozdiel od JIRA Software je Slack dostupný len z cloudu, na ktorý sa možno pripojiť cez webový prehliadač alebo aplikáciou pre všetky desktopové a mobilné platformy, pričom sa jedná o stabilné a prepracované produkty. Ponúka verziu zadarmo, ktorá ale na rozdiel od platených, neposkytuje šifrovanie, externý prístup pre klientov alebo video konferencie.



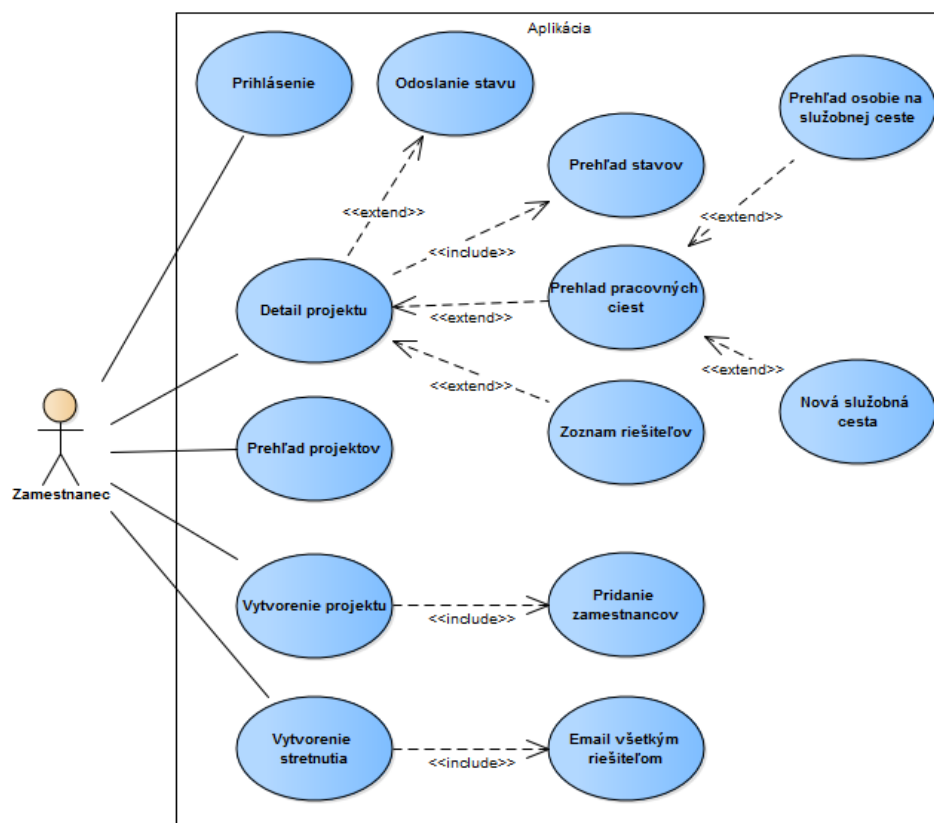
Obr. 3.4: Slack, hlavné okno

Jedná sa o veľmi obľúbený nástroj, ktorý plní viac úlohu rýchleho chatu medzi kolegami v práci, ktorí niečo rýchlo potrebujú, ale nie sú dostatočne urgentné na telefonický rozhovor alebo posielanie mailu.

Cieľom a úlohou tejto práce bolo navrhnuť a vytvoriť aplikáciu, ktorá by zahŕňala výhody predchádzajúcich riešení a v spojení so zadaním by bola vo finále ideálnym riešením pre malý a stredný podnik.

3.4 Zber požiadaviek

Aplikácia má poskytnúť jednoduché a intuitívne používateľské prostredie, ktoré dovoľí používateľovi vidieť na akých projektoch pracuje, správy ktoré si riešitelia poslali, zapisovanie služobných ciest a jednoduchú komunikáciu medzi používateľmi.

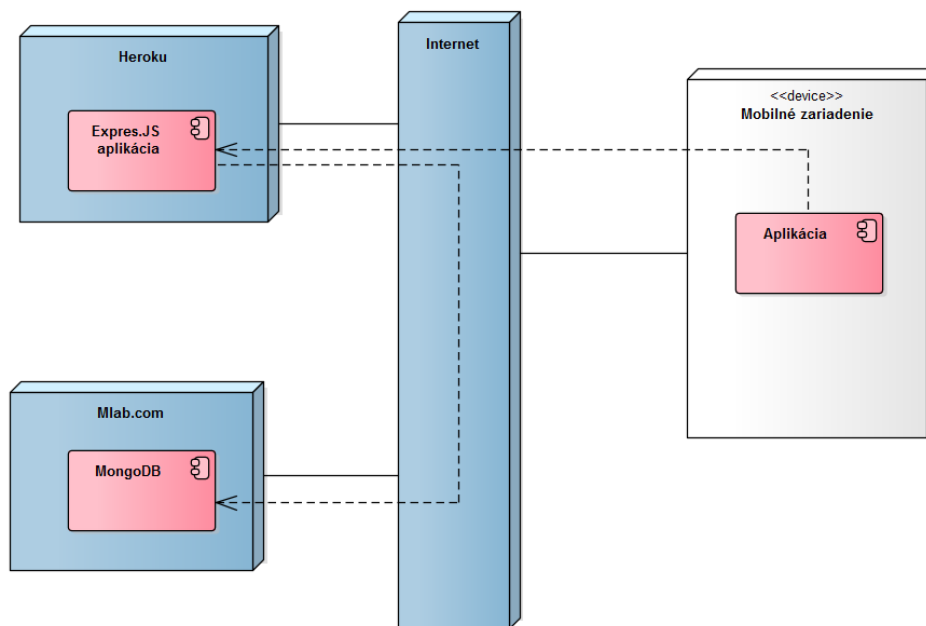


Obr. 3.5: Prípady použitia

Projekt by mal obsahovať všetkých ľudí ktorí na ňom pracujú, ich telefónne čísla, email a pracovnú pozíciu. Organizácia pracovného stretnutia je oznámená všetkým riešiteľom s dátumom a miestom konania na ich emailové adresy.

4. Návrh a implementácia aplikácie

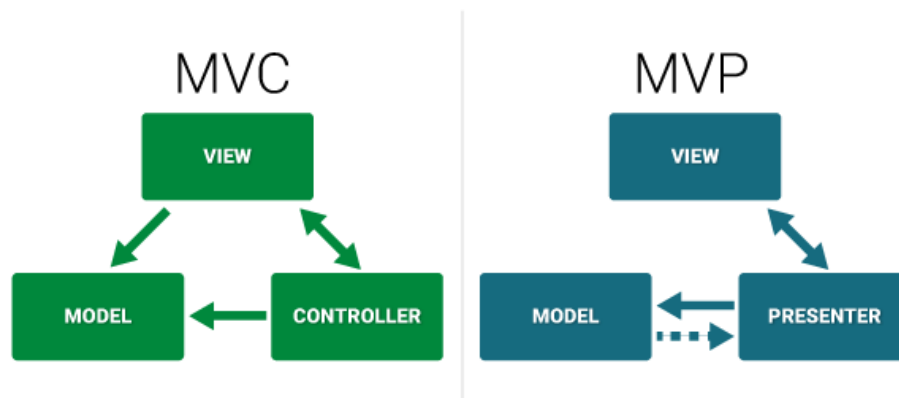
Aby aplikácia podporovala čo najväčšie množstvo zariadení, bude minimálna podporovaná verzia Androidu 4.4, a keďže sa jedná o natívne vyvíjanú aplikáciu, použijeme oficiálny nástroj Android Studio. Za programovací jazyk bola zvolená kombinácia Javy a Kotlinu, keďže sú podporované zvoleným IDE od inštalácie a majú veľkú komunitu spolu s množstvom knižníc. Serverovú časť aplikácie má na starosti služba Heroku, na ktorej je nasadená Node.JS aplikácia, ktorá sa správa ako most medzi MongoDB databázou ktorú poskytuje služba mlab.com 4.1. Na rozdiel od Slacku a JIRA, táto aplikácia ponúkne minimalistické používateľské prostredie a jednoduchosť. V návrhu popíšeme klientskú aj serverovú časť architektúry aplikácie.



Obr. 4.1: Diagram nasadenia

4.1 Architektúra aplikácie

Pri návrhu aplikácie sme sa držali princípu oddelenia zodpovedností. Aby bol kód ľahšie rozširiteľný a v budúcnosti znova použiteľný rozhodli sme sa pre našu aplikáciu aplikovať architektonický vzor Model-View-Presenter.



Obr. 4.2: Porovnanie MVP s MVC

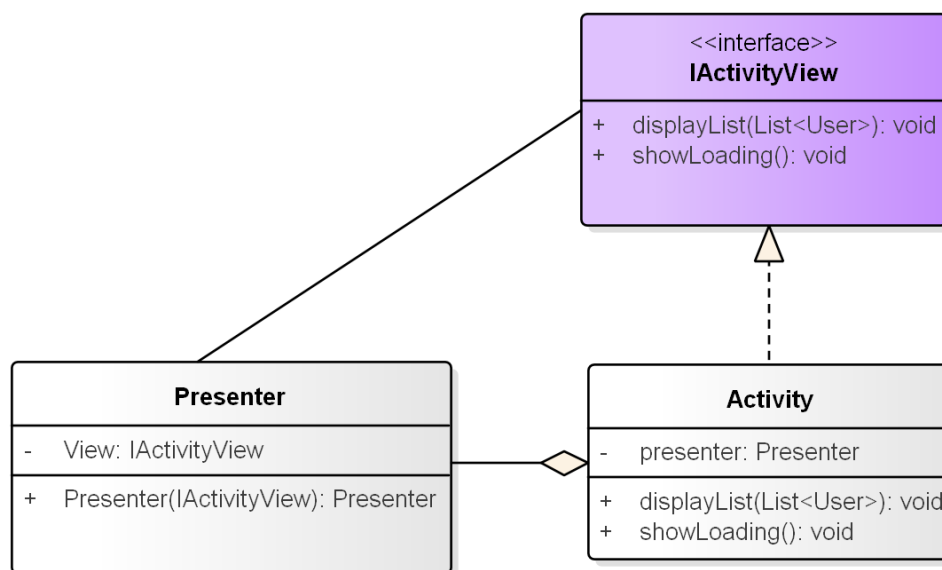
Vzor nám umožní kompletne oddeliť biznis logiku od používateľského prostredia a to rozdelením ich zodpovedností [14].

Presenter plní úlohu mostu medzi View a Modelom. Získava dáta z modelu a naformátované ich presúva do View modulu. Na rozdiel od MVC, rozhoduje aj o tom čo sa stane, keď nastane akcia vo view.

View je implementovaný ako aktivita, ktorá si uchováva referenciu na presenter. Jediná vec ktorú view robí, je že volá metódy presentera.

Model v našej aplikácii predstavuje reprezentáciu dát, ktoré sú uložené v databáze, a aj s ňou komunikuje.

Aplikovanie vzoru (obr.4.2) na Android je jednoduché. Vytvoríme rozhranie *IView* (obr.4.3) ktoré reprezentuje všetky používateľské akcie, ako napríklad zobrazenie animácie, oznámenie o dokončení požiadavky na server a pod.



Obr. 4.3: Nasadenie MVP v Androide

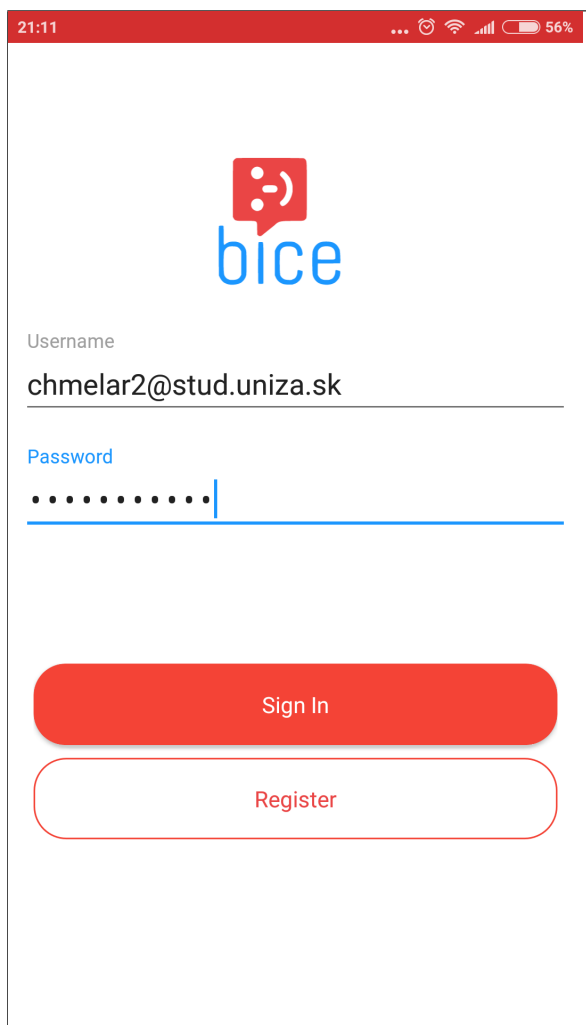
Presenter obsahuje objekt rozhrania, ktorý je aj parametrom konštruktora. Následne môže presenter volať metódy rozhrania. View implementuje rozhranie a všetky jeho metódy. V nich následne implementujeme kód, ktorý súvisí s View, ako je komunikácia s komponentami Androidu ako sú ListView, Progressbar a iné. Takýmto spôsobom oddelíme kód ktorý súvisí s Androidom od našej logiky, tým pádom presenter obsahuje kód, nezávislý na platforme na ktorý je možné napísať unit testy.

Aplikácia obsahuje triedu *App*, ktorá je potomkom triedy *Application*, ktorej inštancia je vytvorená skôr, ako ktorákoľvek iná trieda v balíčku aplikácie. To využívame na vytvorenie objektu klienta na komunikáciu s REST API, ktorý následne vkladáme do presentera aktivity, ktorá využíva sieťovú komunikáciu. Vytvorenie viacerých inštancií Http klienta by bolo zbytočné, keďže sa skladá z obsiahlych knižníc ako serializér a deserializér JSON objektov *GSON*, Http knižnice *OkHttp* a *Retrofit*, ktorý uľahčuje prácu s REST API.

4.2 Prihlasovacia obrazovka

Prihlasovacia obrazovka slúži ako vstupná brána do aplikácie. Obsahuje polia pre email a heslo a tlačidlá prihlásenie, registráciu. Pri nesprávnom formáte emailu ktoré overuje regulárny výraz, alebo nevyplnených poliach, aplikácia používateľa upozorní

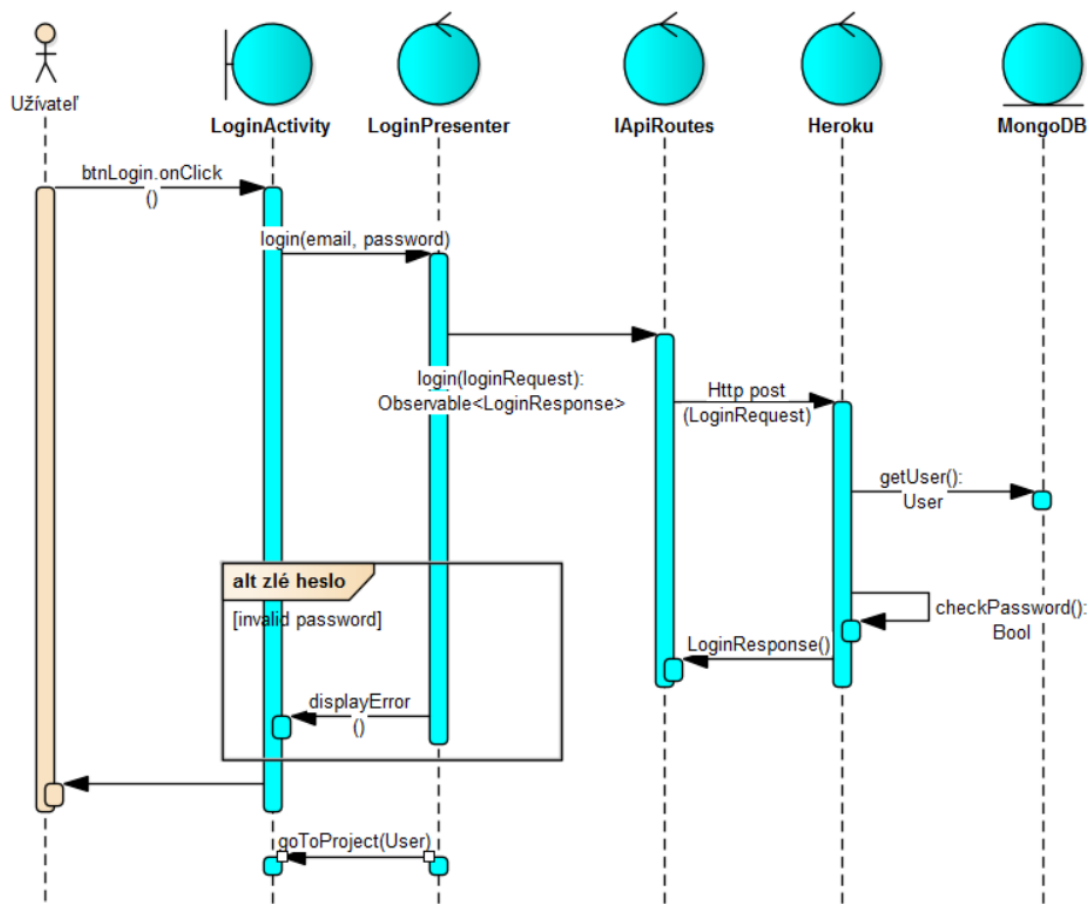
zobrazením chybovej hlášky v príslušnom poli. Ak server vráti odpoveď o nesprávnom hesle, používateľ je rovnako informovaný.



Obr. 4.4: Obrazovka prihlásenia

Pri inicializácii obrazovky vytvoríme inštanciu prezentéra, ktorý ako parametre akceptuje objekt typu *IApiRoutes* na komunikáciu so serverom, a rozhranie *IProjectsView*. Objekt *IApiRoutes* je vložený do prezentéra z hlavnej triedy *App*. Trieda *ProjectsActivity* implementuje rozhranie *IProjectsView* prezentéra, môže spracovať všetky správy odoslané prezentérom prostredníctvom rozhrania.

Po zadaní platnej adresy, ktorá je overovaná regulárnym výrazom, a heslom sa po stlačení červeného tlačítka na prihlásenie zavolá metóda prezentéra, ktorá vola POST požiadavku, ktorej telo obsahuje JSON objekt s menom a heslom.

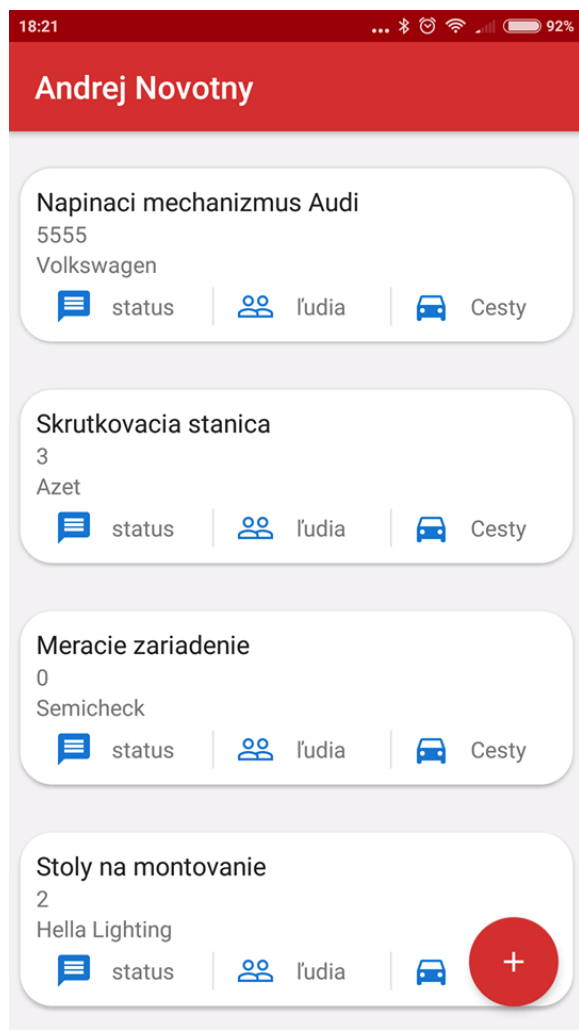


Obr. 4.5: Sekvenčný diagram - Prihlasovanie

Server následne vyhľadá používateľa s požadovanou emailovou adresou. Spracuje heslo z tela požiadavky a porovná ho s kryptovanou verziou hesla. Po úspešnom porovnaní odošle server odpoveď v podobe JSON objektu s dátami o prihlásenom užívateľovi a *JWT* kľúčom, ktorým sa môžeme identifikovať na serveri po dobu 24 hodín. Kľuč sa vkladá do Http klienta, ktorý ho automaticky vloží do hlavičky nasledujúcich požiadaviek.

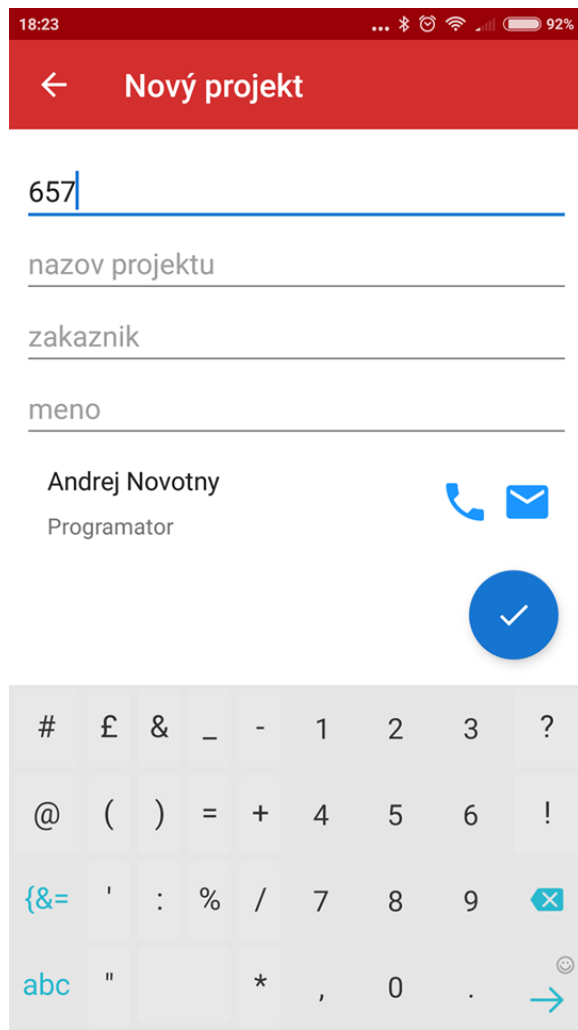
4.3 Obrazovka projektov

Prihlasovacia obrazovka odoslala identifikačné číslo všetkých projektov, ktoré má používateľ zapísané. Prezenter odošle požiadavku na koncový bod servera */projects/:id*, kde *:id* nahradí číslo projektu. Postupne sa projekty zobrazia na hlavnej obrazovke (obr.4.6), odkiaľ používateľ pristupuje ku konkrétnym projektom.



Obr. 4.6: Obrazovka projektov

Po kliknutí na červené tlačítko so znakom “+” nás aplikácia presunie na obrazovku vytvorenia projektu (obr.4.7). Obsahuje polia s nápodou a automatickým dopĺňaním zamestnancov. Po vyplnení požadovaných textových polí odošle prezentér POST požiadavku na koncový bod `/projects/:id` s objektom JSON, ktorý obsahuje vyplnené dáta a čísla zamestnancov, ktorí budú pridaní do projektu. Následne sme presunutý na predošlú obrazovku. Aplikácia nás informuje o stave požiadavky *toast* notifikáciou.



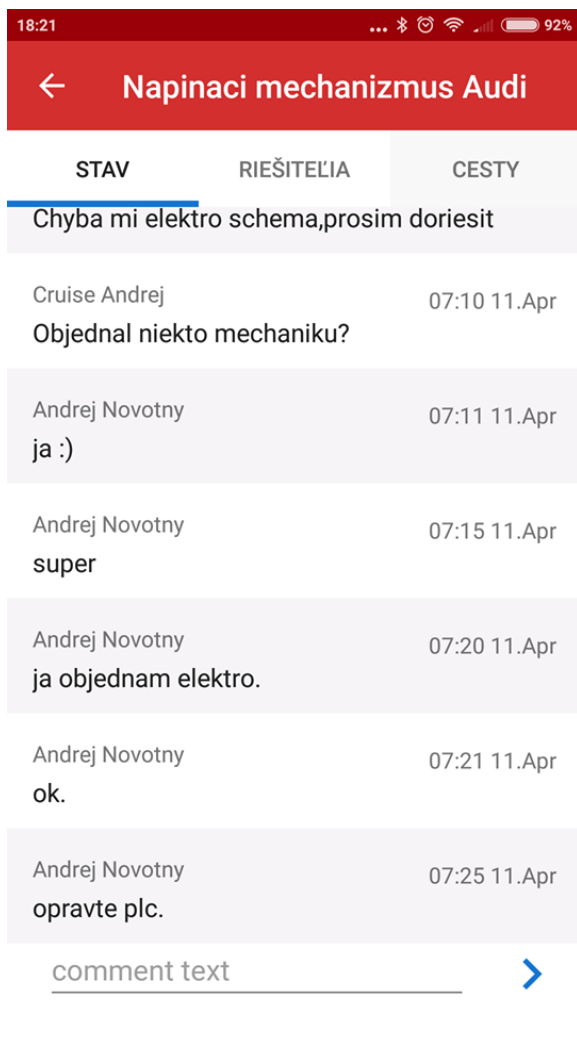
Obr. 4.7: Obrazovka vytvorenia projektu

4.4 Detail projektov

Detail obsahuje tri sekcie, medzi ktorými sa môžeme pohybovať kliknutím na príslušajúcu záložku alebo potiahnutím prsta. Táto časť aplikácie je naprogramovaná v jazyku Kotlin. Obsahuje viacero komponentov ktoré sú veľmi podobné, ale s odlišnou logikou. Vďaka vlastnosti jazyka, ktorá umožňuje posielat funkciu ako parameter sme mohli znova použiť existujúce komponenty.

4.4.1 Stav

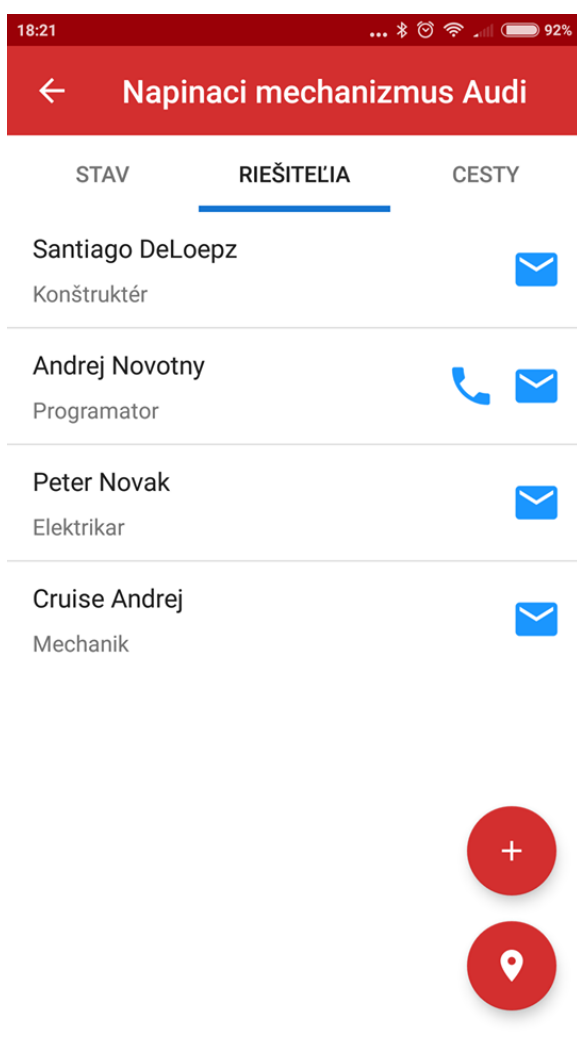
Zobrazuje posledné správy ktoré boli odoslané riešiteľmi projektu. Umožňuje odoslať aktualizáciu stavu prostredníctvom poľa v dolnej časti obrazovky. Podobne ako v predchádzajúcich obrazovkách, prezentér odosiela POST požiadavku na koncový bod */projects/:id/comments* s textom komentára, číslom projektu a číslom užívateľa. Úspešné aktualizovaný stav sa zobrazí v zozname s časom odoslania a menom autora.



Obr. 4.8: Obrazovka stavov

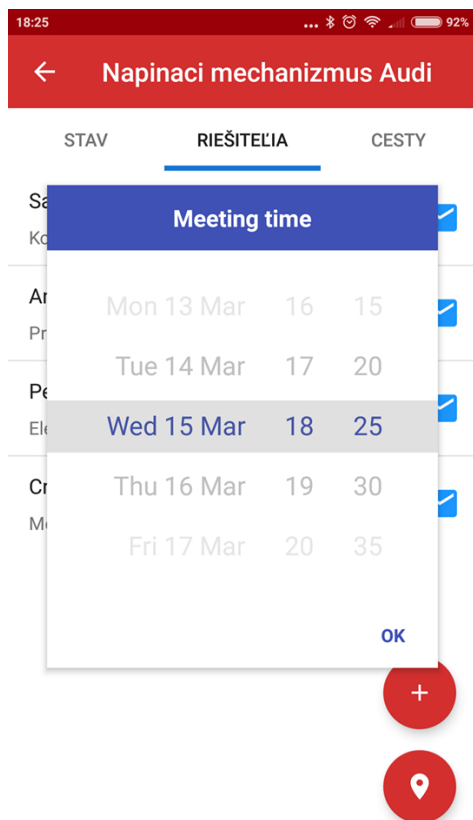
4.4.2 Riešitelia

V tejto záložke je vidieť všetkých používateľov, ktorí boli pridaní do projektu (obr.4.9). Pri mene používateľa je zobrazená jeho pracovná pozícia, možnosť odoslať email alebo uskutočniť hovor. V spodnej časti obrazovky je možné pridať používateľa do projektu, alebo zorganizovať pracovné stretnutie. Kliknutím na ikonu pracovného stretnutia sa zobrazí dialóg na výber dátumu a času. Následne je používateľ presunutý na obrazovku s pred-vyplneným emailom na odoslanie (obr.4.11b). Email je adresovaný všetkým riešiteľom projektu.

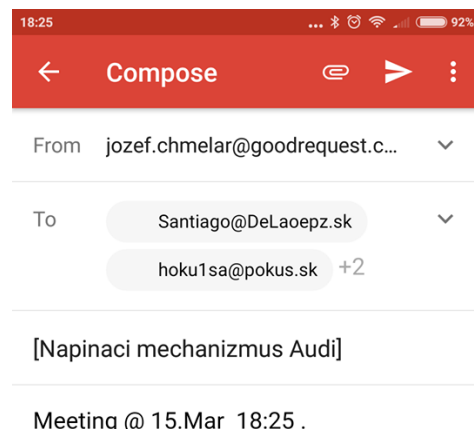


Obr. 4.9: Obrazovka riešiteľov

Výber emailového klienta je prenechaný na používateľa. Môže teda použiť akýkoľvek klient podľa vlastných preferencií



(a) Výber času stretnutia

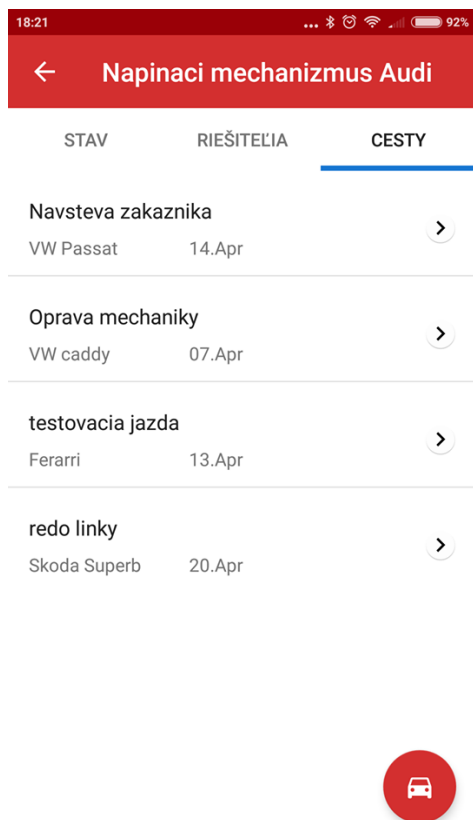


(b) Pred-vyplnený email

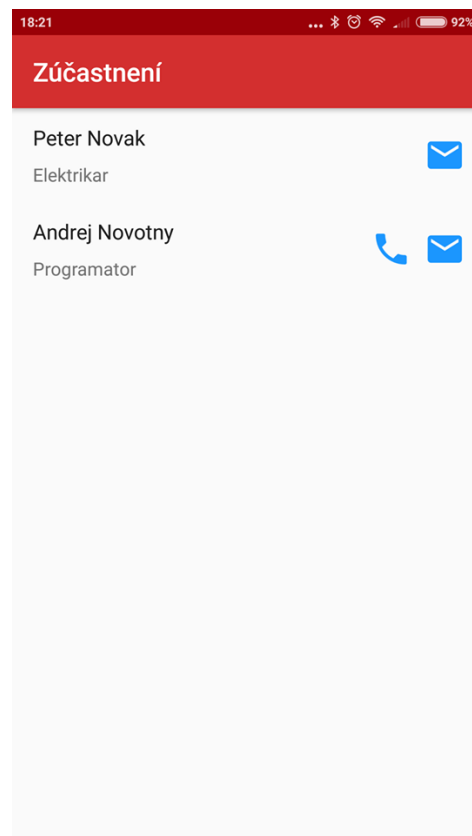
Obr. 4.10: Obrázok stretnutia

4.4.3 Služobné cesty

Obrázok ktorá poskytuje prehľad všetkých pracovných ciest, s dôvodom cesty, autom a dátumom. Po kliknutí na špecifickú cestu sa zobrazí detail so zúčastnenými osobami. Po kliknutí na tlačítko s logom auta sme presmerovaný na obrázok ktorá umožňuje pridanie novej cesty. Tu využijeme polia s automatickým dopĺňaním textu, a dialógu na výber dátumu



(a) Prehľad služobných ciest



(b) Pred-vyplnený email

Obr. 4.11: Detail služobnej cesty

4.5 Serverová časť

Aplikácia musí fungovať z akejkoľvek siete bez nutnosti využívať VPN (virtual private network) na prístup do firemnej siete, rozhodli sme sa teda pre využitie zadarmo prístupných služieb *Heroku* ktoré poskytuje hosting pre NodeJS aplikáciu, ktorá je naprogramovaná použitím frameworku ExpressJS, ktorý uľahčuje prácu pri vytváraní REST API. Aplikácia je napojená na databázu ktorú poskytuje server *Mlab.com*. Jedná sa o bezplatnú inštanciu noSQL databázy MongoDB. MongoDB je vhodný nástroj, pretože na komunikáciu medzi zariadeniami používame formát JSON, ktorý je prirodzený pre Mongo,Node.JS a najpoužívanejší pre REST API. API poskytuje koncové body na vytváranie, editovanie, mazanie a získavanie dát z databázy.

4.5.1 Bezpečnosť

Dôležitou súčasťou každej aplikácie je bezpečnosť a zamedzenie prístupu neoprávneným osobám k citlivým dátam. Framework s ktorým pracujem, *Express.JS*, poskytuje funkciu zvanú “middleware”[21], ktorá umožňuje narušiť cyklus požiadavky. Pri každej požiadavke na server sa spustí práve táto funkcia, ktorá hľadá v hlavičke požiadavky kľúč *x-access-token*. Ak je dekodovaný kľúč platný, funkcia zavolá metódu *next()*, ktorá spustí požadovaný proces. Pokiaľ kľúč nie je platný, server odpovie Http kódom 403, prístup zamietnutý.

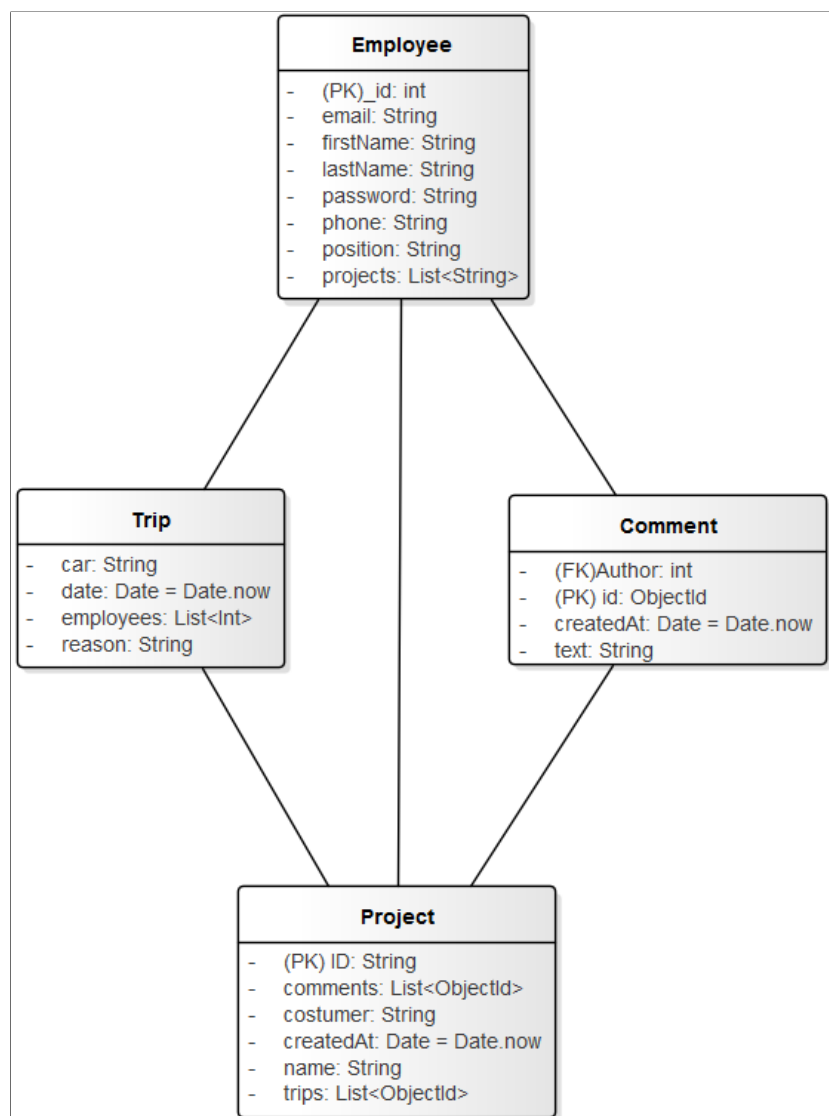
Pri vytvorení nového používateľa sa do databázy ukladá zakódovaná verzia jeho hesla. Hash hesla je generovaný pomocou knižnice JWT a bcrypt. Každé heslo je “osolené” náhodným a tajným reťazcom znakov, ktorý je uložený ako premenná prostredia na Heroku. Pokiaľ majú dvaja užívatelia rovnaké heslo, nie sme to schopní zísť pri pohľade na tabuľku užívateľov v databáze, keďže sú uložené ako úplne odlišné 60 znakové reťazce.

4.5.2 Databáza

Ako databázový systém bol zvolená noSQL databáza MongoDB, ktorej inštanciu nám poskytuje služba mLab. Aplikácia sa po spustení na serveri okamžite pripája k databázovej URL, ktorá je uložená na Heroku v premenných prostredia, pretože obsahuje meno a heslo potrebné k pripojeniu. Následne komunikujeme s databázou pomocou *ORM Mongoose*. *ORM* automaticky mení odpoveď databázy na objekt, vyhľadáva, filtruje a automaticky spája tabuľky pomocou cudzieho kľúča. MongoDB nie je relačná databáza, označuje sa termínom noSQL - “not only SQL”. Nie je viazaná modelom a môže byť kedykoľvek zmenená, nezaručuje teda konzistentnosť dát. Preto používame objekty typu *commentSchema*. V schéme definujeme štruktúru dát a referencie na ostatné tabuľky.

```
1 var commentSchema = new Schema({
2   author: {type: Number , ref:'Employee', required: true},
3   text: { type: String, required: true },
4   createdAt: {type: Date, default: Date.now}
5 }
```

Ako príklad, použijeme najjednoduchšiu schému *commentSchema*. Obsahuje referenciu na tabuľku *Employee* s rovnakým typom ako je primárny kľúč v odkazovanej tabuľke. Automaticky je vytvorená aj premenná *objectId*, ktorá je jednoznačným identifikátorom pre *commentSchema*.



Obr. 4.12: Model databázy

Záver

Bakalárska práca sa zaoberá vývojom aplikácie pre Android, ktorá by uľahčila organizáciu práce projektovému manažérovi. Sprehľadnila stav projektu, zjednotila ľudí pracujúcich na projekte pod spoločné rozhranie, ktoré na komunikáciu využíva stále používaný email, obyčajné telefonáty a stav projektu ktorý už nie je zakopaný v nekonečnej spleti emailov.

Celá práca bola navrhnutá tak, aby sa do budúcnosti ľahko implementovalo webové, alebo ďalšie mobilné rozhranie, nové funkcie a kvôli rozšíriteľnosti databázy, ďalšie dáta o projekte, prípadne používateľoch.

Pri vypracovávaní práce sme získali veľké množstvo skúseností. Vďaka analýze sme si mohli prezrieť veľké množstvo frameworkov, databáz, architektonických vzorov pre vývoj aplikácií a posledných technológií. Z veľkého výberu programovacích jazykov a vzorov sme si vybrali tie najjednoduchšie a najintuitívnejšie, aby výsledkom našej práce bol čistý a prehľadný kód s minimálnym technologickým dlhom.

Prišli sme k záveru, že aj zdanlivo jednoducho znejúce zadanie si na vypracovanie vyžaduje veľa času, prehľad v technológiách a trpezlivosť. Vypracovali sme aplikáciu, ktorá funguje na rovnakom princípe ako väčšina najpoužívanějších aplikácií ako sú napríklad Facebook, preto máme do budúcnosti vzácne skúsenosti ohľadom vývoja mobilných aplikácií, ktoré sú zo dňa na deň žiadanejšie.

Do aplikácie by sme v budúcnosti radi implementovali pridelenie požiadavky konkrétnemu riešiteľovi s notifikáciou o blížiacom sa konečnom termíne a webové rozhranie, ktoré bude už vďaka hotovej serverovej časti rýchlejšie implementované. Ďalej službu monitorujúcu aktivitu používateľov na všetkých obrazovkách a získavanie údajov o chybách pomocou služby *Crashlytics*. Aplikácia je ľahko rozšíriteľná, a preto sa vieme rýchlo prispôbiť požiadavkám firmy o ďalšiu funkcionálnosť.

Literatúra

- [1] Android, *Android - Android is for everyone* [online], [Marec,2017], Dostupné na internete: android.com/everyone/enabling-opportunity/
- [2] bcg.perspectives, *The Growth of the Global Mobile Internet Economy* [online], [Marec,2017], Dostupné na internete: bcgperspectives.com/content/articles/telecommunications_connected_world_growth_global_mobile_internet_economy/
- [3] OpenSignal, *Android Fragmentation Visualized* [online], [Marec,2017], Dostupné na internete: opensignal.com/reports/2015/08/android-fragmentation/
- [4] Dashboards, *Home - Android - Dashboards* [online], [Marec,2017], Dostupné na internete: developer.android.com/about/dashboards/index.html#Platform
- [5] Digitaltrends, *Tired of Google Play?* [online], [Marec,2017], Dostupné na internete: digitaltrends.com/mobile/android-app-stores/
- [6] Verge, *Before it took over smartphones, Android was originally destined for cameras* [online], [Marec,2017], Dostupné na internete: theverge.com/2013/4/16/4230468/
- [7] PhoneArena, *Google's Android OS: Past, Present, and Future* [online], [Marec,2017], Dostupné na internete: phonearena.com/news/Googles-Android-OS-Past-Present-and-Future_id21273/
- [8] Social Compare, *Android versions comparison* [online], [Marec,2017], Dostupné na internete: socialcompare.com/en/comparison/android-versions-comparison
- [9] Engadget, *'ART' experiment in Android* [online], [Marec,2017], Dostupné na internete: engadget.com/2013/11/06/new-android-runtime-could-improve-battery-life/
- [10] Android Developers, *Activity* [online], [Marec,2017], Dostupné na internete: developer.android.com/reference/android/app/Activity.html

- [11] OpenHub, *The Android Open Source project on OpenHub* [online], [Marec,2017], Dostupné na internete: openhub.net/p/android/analyses/latest/languages_summary
- [12] Android Authority, *Java vs C app performance* [online], [Marec,2017], Dostupné na internete: androidauthority.com/java-vs-c-app-performance-689081/
- [13] Quartz Media LLC, *Android just hit a record 88% market share of all smartphones* [online], [Marec,2017], Dostupné na internete: qz.com/826672/android-goog-just-hit-a-record-88-market-share-of-all-smartphones/
- [14] Antonio Leiva *MVP for Android: how to organize the presentation layer* [online], [Marec,2017], Dostupné na internete: antonioleiva.com/mvp-android/
- [15] Android Developers Blog *An update on Eclipse Android Developer Tools* [online], [Marec,2017], Dostupné na internete: android-developers.googleblog.com/2015/06/an-update-on-eclipse-android-developer.html
- [16] Stack tips *Android Studio Features* [online], [Marec,2017], Dostupné na internete: stacktips.com/tutorials/android/android-studio-features
- [17] Idea blog *IntelliJ IDEA and Android Studio FAQ* [online], [Marec,2017], Dostupné na internete: blog.jetbrains.com/idea/2013/05/intellij-idea-and-android-studio-faq/
- [18] Mike Hearn *Why Kotlin is my next programming language* [online], [Marec,2017], Dostupné na internete: medium.com/@octskyward/why-kotlin-is-my-next-programming-language-c25c001e26e3/
- [19] crunchbase *Heroku*[online], [Marec,2017], Dostupné na internete: crunchbase.com/organization/heroku#/entity
- [20] Business Dictionary *What is client-server architecture?* [online], [Marec,2017], Dostupné na internete: www.businessdictionary.com/definition/client-server-architecture.html
- [21] Express *Using Express middleware*[online], [Marec,2017], Dostupné na internete: expressjs.com/en/guide/using-middleware.html