FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání diplomové práce

| | |
|---|---|
| **Název:** | Platforma na monitorovanie zdravotného stavu hráčov športového klubu |
| **Student:** | Bc. Juraj Filan |
| **Vedoucí:** | Ing. Ján Vrábeľ |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové a softwarové inženýrství, zaměření Softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2021/2022 |

## Pokyny pro vypracování

Navrhnite a implementujte platformu na monitorovanie zdravotného stavu hráčov športového klubu. Platforma bude tvorená serverovou časťou a mobilnou aplikáciou.

Na základe analýzy poskytovateľov služieb PaaS (Platform as a Service) vyberte službu, na ktorej navrhnete a implementujete serverovú časť platformy.

Navrhnite a implementujte mobilnú aplikáciu Android.

Pre rolu fyzioterapeuta alebo kondičného trénera bude aplikácia poskytovať tieto služby:
- Evidencia profilu hráčov s históriou zranení a celkového zdravotného stavu.
- Pridávanie a odstraňovanie príloh.
- Prehľadné štatistiky v tabuľkovej aj grafickej podobe.
- Zobrazenie 3D modelu ľudského tela na pridávanie a zobrazenie zranení.

Pre rolu hráča bude aplikácia poskytovať tieto služby:
- Zadávanie momentálneho zdravotného stavu na dennej báze, na ktorého vyplnenie bude upozorňovaný notifikáciou.
- Zadávanie zdravotného stavu po tréningu a zápase

Po implementácii aplikáciu dôkladne otestujte.

---

*Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 16. ledna 2021 v Praze.*

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práca

# Platforma na monitorovanie zdravotného stavu hráčov športového klubu

***Bc. Juraj Filan***

Katedra softwarového inženýrství
Vedúci práce: Ing. Ján Vrábeľ

6. mája 2021

# Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 6. mája 2021 . . . . . .

### Odkaz na túto prácu

Filan, Juraj. *Platforma na monitorovanie zdravotného stavu hráčov športového klubu*. Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

# Abstrakt

Táto diplomová práca opisuje proces vývoja platformy na monitorovanie zdravotného stavu hráčov športového klubu. Práca je zameraná na analýzu, návrh a implementáciu mobilnej aplikácie Android so serverovou časťou implementovanou pomocou služby `AWS Amplify`. Výsledkom práce je funkčná aplikácia, ktorej hlavnou funkcionalitou je zhromažďovanie potrebných dát a súborov na monitorovanie zdravotného stavu hráčov. Nakoniec je táto aplikácia dôkladne otestovaná.

**Klíčová slova** Android, Kotlin, mobilná aplikácia, Amazon Web Services, AWS Amplify

# Abstract

This Master's thesis describes the process of development of a platform for monitoring health condition of sport club players. The thesis is mainly focused on analysis, design and implementation of an Android mobile application with a server's part implemented by `AWS Amplify`. Result of the thesis is a working mobile application whose main functionality is collecting necessary data and files for monitoring health condition of players. Finally, the application is thoroughly tested.

**Keywords** Android, Kotlin, mobile application, Amazon Web Services, AWS Amplify

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Zoznam zdrojových kódov

# Úvod

Mobilné aplikácie či chytré hodinky sú dnes bežnou súčasťou tréningového procesu športovcov a celého podporného tímu. Vhodne navrhnutá aplikácia môže pomôcť športovcom rozvíjať svoje schopnosti, riadiť sa pokynmi trénera, sledovať svoje životné a telesné funkcie, prípadne si zaznamenávať rôzne udalosti. Mobilná aplikácia je výbornou pomôckou aj pre trénerov, ktorí okrem evidencie svojich zverencov, môžu prostredníctvom aplikácie nastaviť tréningovú záťaž podľa aktuálneho stavu jednotlivých hráčov. Fyzioterapeutom a lekárom môže vhodne navrhnutá aplikácia pomôcť pri sledovaní zdravotného stavu hráča.

Mobilná aplikácia pre športovcov, fyzioterapeutov, trénerov a ostatných členov športového tímu je výbornou pomôckou aj v čase pandémie, kedy hygienické obmedzenia neumožňujú športovým klubom vykonávať tréningové procesy v štandardnom režime. Aplikácia môže čiastočne nahradiť osobný kontakt hráča s vedením športového klubu.

Diplomová práca opisuje návrh a implementáciu aplikácie, ktorá v sebe zahŕňa komplexné riešenie pre hráčov, trénerov, fyzioterapeutov, lekársky personál a iných členov športového tímu. Návrh aplikácie vychádzal z funkčných požiadavkov konkrétneho športového klubu. Okrem možnosti zhromažďovania potrebných údajov na jednom mieste, aplikácia umožňuje hráčom klubu pravidelne posielať svoje pocity po fyzickej záťaži alebo spánku, prípadne pravidelne pridávať aktuálne hodnoty svojich životných funkcií. Aplikácia tiež umožňuje zobrazenie histórie zranení jednotlivých hráčov pomocou 3D modelu ľudského tela. V neposlednom rade bola splnená jedná zo základných požiadaviek, aby bolo ovládanie aplikácie jednoduché a intuitívne.

Kapitola 1

# Cieľ práce

Hlavným cieľom práce je navrhnúť a implementovať aplikáciu, ktorej úlohou bude zjednodušiť prácu trénerom, fyzioterapeutom, lekárskemu personálu a iným členom vedenia športových klubov pomocou zhromažďovania potrebných dát a súborov na jednom mieste. Táto aplikácia je taktiež pre hráčov, ktorí môžu pravidelne posielať svoje pocity po fyzickej záťaži alebo spánku a pridávať svoje aktuálne telesné merania. Ďalším cieľom je na základe analýzy poskytovateľov služieb `PaaS` vybrať službu, na ktorej bude navrhnutá a implementovaná serverová časť aplikácie.

Čiastkovým cieľom je analýza podobných existujúcich riešení, ktoré môžu slúžiť ako inšpirácia. Ďalej bude potrebné túto aplikáciu nasadiť do `Google Play Store` a pripraviť na testovaciu fázu, na ktorej sa budú podieľať členovia vedenia klubu a vybraní hráči.

Kapitola 2

# Analýza

Táto kapitola sa zaoberá analýzou poskytovateľov služieb `PaaS` pre implementáciu serverovej časti a analýzou funkčných a nefunkčných požiadavkov aplikácie. Súčasťou tejto kapitoly je taktiež analýza vývoja na mobilnej platforme Android, ktorá obsahuje analýzu existujúcich riešení a analýzu `best practises` moderného vývoja na tejto platforme.

## 2.1 Požiadavky aplikácie

Táto sekcia sa zaoberá analýzou funkčných a nefunkčných požiadavkov navrhovanej aplikácie, ktoré sú spracované pre určité požiadavky konkrétneho športového klubu, ktorý bude túto aplikáciu následne testovať.

### 2.1.1 Funkčné požiadavky

#### 2.1.1.1 Správa účtu

**Registrácia** – Od používateľa je požadované vytvorenie účtu, ktorým sa dá prihlásiť do aplikácie až po schválení správcom.

**Schválenie** – Po úspešnej registrácií je potrebné, aby správca nový účet schválil a zaradil ho do správnej kategórie. Správca môže tento účet taktiež zamietnuť.

**Prihlásenie** – Používateľ má umožnený prístup do aplikácie novým účtom až po schválení správcom. Po prihlásení je potrebné zistiť rolu používateľa, na základe čoho sú používateľovi sprístupnia funkcie k danej role.

**Odhlásenie** – Používateľ má možnosť sa z účtu odhlásiť, čím stratí prístup do aplikácie..

#### 2.1.1.2 Rozdelenie používateľov

**Kategórie** – Každý registrovaný a schválený účet je zaradený správcom do kategórie. Medzi kategórie patria jednotlivé kategórie športového klubu a taktiež špeciálna kategória správcov aplikácie. Kategóriu môže vytvoriť iba správca aplikácie.

**Hráč** – Ako už z názvu vyplýva, táto rola je určená pre samotných hráčov. Používateľ s touto rolou má prístup k základným funkciám aplikácie a k správe výhradne len svojho účtu.

**Správca** – Jedná sa o rolu používateľa, ktorý má väčšie právomoci ako hráč. Okrem možnosti schvaľovania nových účtov, má možnosť spravovania kategórií s hráčmi a ich informáciami. Do tejto roly patria fyzioterapeuti, tréneri a iní členovia vedenia klubu. Používateľ s touto rolou je členom špeciálnej kategórie správcov aplikácie.

#### 2.1.1.3 Správa dát a súborov

**RPE (Rating of Perceived Exertion) hodnoty** – Sledujú dva typy `RPE` hodnôt, po tréningu a po zobudení. Obidva typy by mal hráč zadávať na dennej báze (okrem dňa kedy nemá tréning alebo inú fyzickú záťaž). Hráč si môže vytvoriť upozornenie na vyplnenie dát v určitý čas, na ktorý bude upozornený denne notifikáciou. Históriu nie je možné meniť a je reprezentovaná grafovou formou.

**Namerané telesné hodnoty** – Tento typ údajov bude hráč (v krajnom prípade správca) pridávať raz týždenne. V prípade možných chýb je možné hodnoty upravovať alebo mazať. Ich história je reprezentovaná grafovou formou a zoznamom.

**História zranení** – Správca môže spravovať zranenia jednotlivých hráčov, hráč si ich potom môže iba zobraziť. Pre prehľadnejšie zobrazenie budú zranenia reprezentované 3D modelom ľudského tela.

**Prílohy** – K jednotlivým hráčom si môže správca uložiť prílohy rôznych typov, ako dokumenty, obrázky a iné súbory.

Prehľadnejšie rozdelenie práv pre `CRUD (Create, Read, Update, Delete)` operácie pri správe dát a súborov pre obidva typy používateľov, je farebne znázornené v tabuľke 2.1.

Tabuľka 2.1: Rozdelenie práv pre `CRUD` operácie pri správe dát a súborov

| | Správca | | | | | Hráč | | | |
|---|---|---|---|---|---|---|---|---|---|
| | C | R | U | D | | C | R | U | D |
| RPE hodnoty | | | | | | | | | |
| Namerané telesné hodnoty | | | | | | | | | |
| História zranení | | | | | | | | | |
| Prílohy | | | | | | | | | |

### 2.1.2 Nefunkčné požiadavky

**Platforma Android** – Táto platforma bola vybraná na základe skúseností s danou platformou. Po úspešnej fáze testovania a dohode s vedením, bude vyvinutá aj `iOS` aplikácia.

**Minimálna verzia Android 7** – Táto verzia bola vydaná začiatkom roka 2016. Momentálne je najnovšia verzia Android 11. Tým, že je aplikácia vyvíjaná pre športové kluby, ktoré sú prevažne tvorené mladými ľudmi, sa znižuje pravdepodobnosť používania starších verzií Android.

**Jednoduché používanie** – Aplikácia je určená pre športovcov a aj z tohto dôvodu je potrebné, aby rozmiestnenie funkcií v aplikácií bolo zrozumiteľné a intuitívne.

**Anglický a slovenský jazyk** – Aplikácia je primárne cielená na športový klub zo Slovenska. Samozrejme môžu byť v športovom klube aj členovia zo zahraničia. Z tohto dôvodu je nutné, aby aplikácia bola aj v anglickom jazyku. Samotné pridávanie nových jazykov je otázka prekladu a ich konfigurácia nie je náročná.

## 2.2 Analýza poskytovateľov služieb PaaS

V tejto sekcií bude vysvetlené, prečo práve služba typu `PaaS` je správnym riešením pre navrhnutú aplikáciu. Na začiatok by bolo vhodné vysvetliť všetky typy cloudových služieb, ktoré ponúkajú ich poskytovatelia:

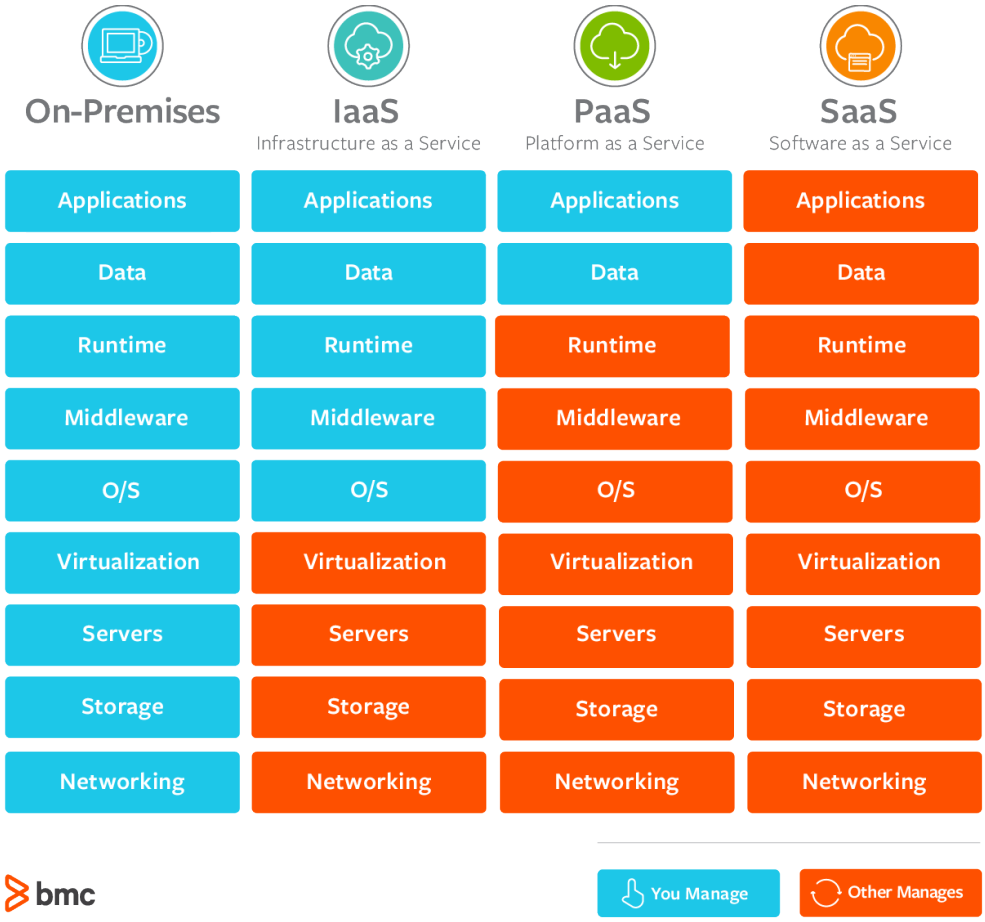
**IaaS (Infrastucture as a Service)** – Táto služba je určená pre zdatnejších vývojárov. Poskytne im fyzické servery s nainštalovanými operačnými systémami, s ktorými majú vývojári možnosť vzdialene komunikovať a nainštalovať si tak na nich vlastné programy, skripty a aplikácie. Patria sem napríklad `Amazon EC2`, `Compute Engine` a `Azure Virtual Machines`.

**PaaS (Platform as a Service)** – Služba určená pre vývojárov, ktorí pre svoju aplikáciu potrebujú jednoduchý `backend`. Vývojár sa nemusí starať o konfiguráciu a inštaláciu potrebných nástrojov, ale používa už priamo pripravené nástroje. Najpopulárnejšie príklady sú `Windows Azure`, `Google App Engine` alebo `Heroku`.

**SaaS (Software as a Service)** – Jedná sa o službu, ktorej cieľom je rýchla a jednoduchá dostupnosť produktu pre koncového používateľa. Medzi tieto služby patrí napríklad `Google Workspace (Gmail, Google Drive, Google Calendar...)` a `Dropbox`.

[1]

Obrázok 2.1 podrobne zobrazuje zodpovednosti vývojára pri používaní jednotlivých služieb.




Obr. 2.1: Zodpovednosti vývojára pri službách IaaS, PaaS, SaaS [1]

Cieľom tejto práce je vytvoriť aplikáciu, ktorej serverová časť bude určená na spravovanie používateľov, dát a súborov, a nebude obsahovať žiadnu zložitú logiku. V službe typu `PaaS` sú všetky tieto požiadavky už pripravené. V prípade `IaaS` by bolo potrebné nakonfigurovať a nainštalovať servery, čo je pre takýto typ aplikácie zbytočné a časovo neefektívne.

### 2.2.1 MBaaS ako vhodné riešenie

MBaaS patrí pod typ služby PaaS s rozdielom, že vývojár má ešte menej zodpovednosti. Tento typ služby má pripravených mnoho často používaných funkcií a pre frontend vývojárov dostupné SDK alebo API, ktoré výrazne zjednodušia a zrýchlia vývoj. Vďaka tomu má vývojár možnosť venovať viac času samotnej aplikácií. [2]

Táto sekcia sa bude ďalej zaoberať analýzou poskytovateľov služby MBaaS:

**Azure Mobile Apps** – Služba od spoločnosti Microsoft, ktorá poskytuje množstvo služieb a funkcií, ako SQL a NoSQL databázu, správu účtov a SDK pre natívne mobilné platformy Android a iOS. Poskytuje ročnú verziu bezplatne pre niektoré služby s určitým limitom. Jednou z nevýhod tejto služby je nejasná dokumentácia. [3] [4]

**MongoDB Realm** – Výhodou tejto služby je komunikácia s databázou pomocou GraphQL. Okrem toho poskytuje aj ďalšie služby, ktoré sú nevyhnutné pre MBaaS, ako napríklad správa účtov a serverless služba na spúšťanie logiky na serverovej časti. Túto službu je možne používať s určitými limitmi bezplatne iba na jeden mesiac, čo je veľká nevýhoda a nevyhovuje k riešeniu aplikácie. [3] [5]

**Kinvey** – Aj táto služba poskytuje všetky potrebné služby a funkcie. Má veľmi komplexnú a detailnú dokumentáciu, čo môže byť nevýhoda pre nováčikov. Túto službu je možne používať s určitými limitmi bezplatne iba na jeden mesiac, čo je veľká nevýhoda a nevyhovuje k riešeniu aplikácie. Ďalšou nevýhodou je cena za službu, pretože je potrebné jednorazovo zaplatiť vysokú sumu a nepočíta sa cena za spotrebu jednotlivých služieb. [6]

Najznámejšími poskytovateľmi MBaaS sú AWS Amplify a Google Firebase, ktorým sú venované samostatné sekcie 2.2.1.1 a 2.2.1.2, kde sú analyzované jednotlivé funkcie, ktoré sú použité v riešení aplikácie.

#### 2.2.1.1 AWS Amplify

Tento typ služby zahrňuje všetky potrebné funkčné požiadavky serverovej časti. Taktiež má ročnú skúšobnú verziu zadarmo, kde je po prekročení limitov potrebné zaplatiť určitú vypočítanú čiastku podľa používania danej služby. Na testovaciu fázu je limit postačujúci a nepredpokladá se jeho prekročenie.

`AWS` má množstvo funkcií a služieb. Z tohto dôvodu sa tento typ služby neodporúča pre začiatočníkov, pretože je pomerne náročne sa v ich systéme zorientovať. Táto služba bola vybraná z osobných pozitívnych skúseností. Výhodou je ročná bezplatná verzia, `GraphQL` a rýchly vývoj pomocou `AWS Amplify Android SDK`.

Pre serverovú časť aplikácie budú použité tieto služby:

**Amazon Cognito** – Používa sa na správu účtov. Je možné nastaviť množstvo vedlajších atribút a podmienok pri registrácií (pravidlo pre tvar hesla, nutnosť vyplnenia emailu, telefónu a iné). Taktiež má pripravené overovanie emailu, resetovanie hesla zaslaním emailu a dostupné `API` pre správu účtov. [7]

**Amazon DynamoDB** – Jedná sa o `NoSQL` databázu, s kotoru sa komunikuje pomocou `GraphQL`. Tento typ databázy má vlastný systém dotazovania, pomocou ktorého je veľmi jednoduché riešiť podmienečné dotazovanie priamo z `frontendu`. Dáta sú uchovávané v kľúč-hodnota forme `JSON`. [8]

**Amazon S3** – Slúži na správu súborov rôzneho typu. Má prostredie, kde je možné nastaviť prístup a správu súborov. [9]

**AWS Lambda** – `Serverless` služba, ktorá slúži na spúšťanie kódu. Pomocou tejto služby je možné naprogramovať jednoduchú logiku, ktorá je potrebná pri riešení aplikácie. Podporuje väčšinu moderných jazykov, ktoré sa používajú pri implementácií `backendu`, ako `Python`, `Go`, `Java` a `NodeJS`. Pre jej správu je pripravené prostredie, kde je možné písať kód, pridávať podporné knižnice a spravovať `deploy`. Táto služba beží na serveroch s operačným systémom `Linux`, takže možné pomocou nej spúšťať skripty skompilované na tomto operačnom systéme. [10]

**AWS Amplify Android SDK** – Pomocou tohto `SDK` je výrazne zjednodušená implementácia komunikácie s vyššie spomenutými `AWS` službami. Podporuje Android aplikácie vyvíjané v jazyku `Java` alebo `Kotlin` a taktiež podporuje rôzne asynchrónne technológie ako `RxJava` alebo `Kotlin Coroutines`. Inštalácia tohto `SDK` zahrňuje nástroj `Amplify CLI`, pomocou ktorého je možné ovládať a konfigurovať nastavenia služieb priamo z príkazového riadka. [11]

#### 2.2.1.2 Google Firebase

Táto služba je v Android komunite dobre známa a poskytuje množstvo výhod pri tvorbe serverovej časti. Tým, že sa na serverovú časť bude používať služba `AWS Amplify`, budú z tejto služby použité iba následujúce nástroje:

**Analytics** – Pomocou tohto nástroja je možné sledovať rôzne dáta ako počty používateľov v rôznych časových intervaloch, demografické údaje alebo počty zobrazení jednotlivých častí aplikácie.

**Crashlytics** – Tento nástroj slúži na vzdialené sledovanie pádov a chýb aplikácie. Pri vzniknutej chybe v aplikácií sa objaví v zozname chýb s množstvom údajov od samotného záznamu po typ zariadenia. V aplikácií je možné nastaviť identifikátor zariadenia, aby bolo pri vzniknutej chybe možné vyhľadať jeho používateľa.

Obidva nástroje sú súčasťou `Firebase SDK`, ktorého inštalácia a konfigurácia je veľmi jednoduchá. Na ich správu sa používa `Firebase Console`, kde je možné sledovať množstvo informácií vo forme grafov alebo zoznamov. [12]

## 2.3 Mobilná aplikácia Android

Táto sekcia je venovaná analýze vývoja na mobilnej platforme Android. Okrem analýzy samotnej mobilnej platformy, najnovších nástrojov a knižníc pre moderný vývoj Android aplikácií, budú analyzované aj existujúce riešenia.

### 2.3.1 Analýza existujúcich riešení

Tým, že sa jedná o biznis aplikáciu, ktorá je určená pre malú klientelu, tak je hľadanie existujúcich riešení pomerne náročné. Preto budú analyzované aplikácie, ktorých funkcie sa budú aspoň z časti podobať danej aplikácií. Medzi tieto aplikácie patria:

**Megafit** – Táto jednoduchá aplikácia slúži na sledovanie rôznych telesných hodnôt pomocou chytrej váhy, s ktorou komunikuje cez bezdrôtovú technológiu `Bluetooth`. Po odvážení sa namerané hodnoty uložia do zariadenia a je možné sledovať históriu jednotlivých nameraných hodnôt v grafickej podobe. Aplikácia obsahuje moderné `UI` komponenty. [13]

**Google Fit** – Táto aplikácia sleduje rôzne telesné hodnoty, či už pomocou chytrých hodiniek alebo priamo mobilným zariadením. Aplikácia je schopná zaznamenávať napríklad počet krokov alebo vzdialenosť. Na niektorých novších zaradeniach Android, je možné sledovať hodnoty ako tep srdca alebo pravidelnosť dýchania. [14]

**Nethlete** – Veľmi komplexná aplikácia s množstvo funkciami. Jedná sa o platenú aplikáciu so 7 dňovou bezplatnou skúšobnou verziou. Aplikácia má vlastný dizajn bez použitia natívnych `UI` komponentov a vzorov, čo pri takto komplexnej aplikácií výrazne zhoršuje `UX` s danou aplikáciou z dôvodu ťažkého zorientovania sa. Veľkým prekvapením je 26 video návodov pre rôzne funkcie aplikácie, ako pridávanie nových hráčov, správa zápasov alebo tréningov, video analýza, správa kalendára a množstvo iných funkcií. Bez týchto video návodov by bolo pochopenie všetkých funkcionalít pre bežného člena vedenia športového klubu nemožné. Takto komplexná aplikácia by mala mať jednoznačne rozumnejšie oddelené jednotlivé funkcie. Takto to pôsobí, že je príliš veľa vecí pokope, čo pravdepodobne vzniklo pridávaním nových funkcií do aplikácie po nejakom čase. Aplikácia je dostupná na `Google Play Store` a `iOS` verzia `Apple store`. [15]

Tieto aplikácie sú dobrou inšpiráciou pri navrhovaní aplikácie. Niektoré ich funkcie sú veľmi podobné funkčným požiadavkám analyzovaných v sekcií 2.1.1.

### 2.3.2 Analýza technológií a knižníc

Táto sekcia obsahuje zoznam použitých technológií a knižníc v mobilnej aplikácií na platforme Android:

**Kotlin** – Vývoj Android aplikácií prešiel za 11 rokov jeho existencie mnohými zmenami. Jedným z najväčších bol prechod z programovacieho jazyka `Java` na programovací jazyk `Kotlin`. Tento jazyk sa stal preferovaným pre vývoj Android aplikácií v roku 2019, odvtedy rástol vývoj pomocou tohto jazyka exponenciálne a predbehol tak jazyk `Java`. Ovládanie jazyku `Java` je ale stále nevyhnutné, z dôvodu množstva starých knižníc a návodov práve v tomto jazyku. Pri vývoji je možné používať obidva jazyky zároveň, vďaka tomu je možné jednoducho inovovať staré projekty napísané v jazyku `Java`. Výhodou jazyka `Kotlin` je jeho moderná syntax s prvkami funkcionálneho programovania a `null safety`. [16] [17]

**Android Jetpack** – Kolekcia Android knižníc a tried, ktorá je braná ako `best practises` pre vývoj Android aplikácií priamo od `Google`. Jej cieľom je zjednotiť a zjednodušiť vývoj na tejto platforme. Obsahuje veľké množstvo knižníc, ktoré sú opísané nižšie. [18]

**Activity** – Jedná sa o fundamentálny komponent Android aplikácie, ktorá riadi väčšinu procesov aplikácie. Poskytuje okno, v ktorom aplikácia vykresľuje `UI`. Pre efektívnejšie a rýchlejšie fungovanie aplikácie sa na samotné zobrazovanie a správu `UI` odporúča používať `Fragment`, ktorý musí byť hostiteľom `Activity`. [19]

**Fragment** – Jedná sa o triedu reprezentujúcu základný `UI` komponent, ktorý ma svoj vzhľad definovaný pomocou `XML`. Každý fragment ma svoj vlastný životný cyklus, ktorý je potrebné kontrolovať kvôli únikom pamäte. Musí byť hostiteľom `Activity`, bez nej nemôže existovať. [20]

**ViewModel** – Táto trieda sa používa pri architektúre `MVVM`. Je súčasťou `Android Jetpack` a je schopná reagovať na životný cyklus aplikácie. Vďaka nej prežijú jej dáta znovu-vytváranie `UI` komponentov (napríklad pri rotácií zariadenia). Umožňuje aj použitie knižnice `Data Binding`, ktorá je opísaná nižšie. [21]

**Kotlin Coroutines** – Slúžia na spúšťanie asynchrónneho kódu. Spúšťajú sa v rôznych prostrediach, ktoré reagujú na zmeny životného cyklu vybraných komponentov aplikácie. `Coroutine` je návrhový vzor, ktorý používajú aj iné jazyky. [22]

**Kotlin Flow** – Na rozdiel od `suspend` funkcií v `Kotlin Coroutines`, `Flow` umožňuje vracať väčšie množstvo dát. Tento systém sa nazýva `stream of data`. Funguje na princípe producenta a konzumenta, medzi ktorými môže byť prostredník, ktorý mení alebo upravuje dáta. [23]

**Livedata** – Táto trieda obaľuje premennú, na ktorej zmeny reaguje a notifikuje jej sledovateľov. Je súčasťou `Android Jetpack` a je navrhnutá tak, aby rozumne reagovala na životný cyklus aplikácie. [24]

**Data Binding** – Táto knižnica výrazne zjednodušuje prepojenie potrebných dát na zobrazenie s `UI` komponentami priamo v `XML` daného komponentu. Okrem toho spolupracuje s `Livedata`, vďaka čomu môže `UI` komponent jednoducho reagovať na zmeny dát. [25]

**Room** – Knižnica z kolekcie `Android Jetpack`, ktorá slúži na implementovanie lokálnej perzistentnej `SQL` databázy, ktorá podporuje `Coroutines` a `Flow`. Pri implementácií sa používajú anotácie a `DAO` rozhranie. Základné typy vie spracovať bez dodatočnej implementácie, pre neznáme typy je potrebné implementovať `converter`. [26]

**Hilt** – Oficiálna knižnica pre Android, pomocou ktorej sa dá jednoducho implementovať `dependency injection`. Táto knižnica je postavená na populárnom predchodcovi `Dagger`. Pri použití knižnice `Dagger` vznikalo množstvo kódu, ktoré efektívne odstráni práve knižnica `Hilt` pomocou jednoduchých anotácií. [27]

**DataStore** – Táto knižnica slúži na ukladanie dát v kľúč-hodnota forme. Narozdiel od jej predchodcu `SharedPreferences`, je navrhnutá na vývoj s technológiami `Coroutines` a `Flow`. Taktiež patrí do `Android Jetpack`. [28]

**Navigation** – Táto technológia slúži ako nástroj pre zjednodušenie navigácie medzi fragmentami v aplikácií. Graf reprezentujúci túto navigáciu je implementovaný v jazyku `XML`. Najnovšie verzie vývojového prostredia `Android Studio` podporujú aj vizuálne zobrazenie tohto grafu, čo výrazne zjednodušuje jeho implementáciu a zlepšuje prehľadnosť. [29]

**Glide** – `Open source` knižnica, ktorá okrem načítavania obrázkov do `UI` má taktiež na starosti dekódovanie obrázkov alebo správu pamäti a `cache` [30]

**Sceneform** – Oficiálna knižnica od `Google`, ktorá umožňuje prácu s 3D modelmi priamo v aplikácií. Knižnica je primárne mierená na vývoj aplikácií s `Augmented Reality`, ale taktiež umožňuje vytvoriť scénu a v nej načítať modely vo formáte `gltf` alebo `glb`. [31]

**MPAndroidChart** – `Open source` knižnica, ktorá slúži na vizualizáciu dát v grafickej podobe vo forme grafu rôzneho typu. [32]

**Timber** – Táto knižnica sa používa pre zjednodušenie a zrýchlenie výpisu logov pri debugovaní aplikácie. [33]

### 2.3.3 Verzia systému

Pri vývoji mobilných Aplikácií Android, hra veľkú rolu verzia systému, na ktorý je táto aplikácia cielená. Vďaka tomu je možné používať najnovšie funkcie, ktoré prinášajú nové verzie Android zariadení. `Google` vydáva nové verzie každý rok, čo prináša stále nové nástroje a technológie, ktoré môžu vývojári používať. Pri vývoji aplikácie je potrebné nastaviť tieto dve hodnoty:

**Minimálna verzia SDK** – Jeho hodnota určuje najnižšiu verziu systému, na ktorom sa dá daná aplikácia nainštalovať. Používateľ so zariadením so staršou verziou systému si nemôže nainštalovať túto aplikáciu.

**Cielená verzia SDK** – Minimálna hodnota nemôže byť nižšia ako nastavená minimálna verzia SDK. `Google Play Store` každý rok navyšuje minimálnu cielenú verziu, ktorú musí aplikácia pred nahratím splňovať.

Pri voľbe týchto dvoch verzií je potrebné nájsť distribúciu verzií systému Android momentálne aktívnych zariadení. Túto funkciu poskytuje priamo vývojové prostredie `Android studio` priamo pri vytváraní nového projektu. [34] Momentálnu distribúciu je vidieť na obrázku 2.2. Keďže je aplikácia cielená na športové kluby, ktorá je tvorená mladými ľudmi, tak sa dá predpokladať, že nebudú používať staré verzie. Z tohto dôvodu bude minimálna verzia `SDK` 24, čo reprezentuje zariadenie s verziou Android 7, ktorá je stará 5 rokov. Od augusta 2021 bude povinná cielená verzia `SDK` 30 (Android 11), takže je rozumné tak nastaviť aj cielenú verziu. [35] Aplikácia bude niekoľko mesiacov v testovacej fáze, takže je potrebné pri výbere `SDK` verziách myslieť dopredu.

| ANDROID PLATFORM VERSION | API LEVEL | CUMULATIVE DISTRIBUTION |
|---|---|---|
| 4.0 Ice Cream Sandwich | 15 | |
| 4.1 Jelly Bean | 16 | 99,8% |
| 4.2 Jelly Bean | 17 | 99,2% |
| 4.3 Jelly Bean | 18 | 98,4% |
| 4.4 KitKat | 19 | 98,1% |
| 5.0 Lollipop | 21 | 94,1% |
| 5.1 Lollipop | 22 | 92,3% |
| 6.0 Marshmallow | 23 | 84,9% |
| 7.0 Nougat | 24 | 73,7% |
| 7.1 Nougat | 25 | 66,2% |
| 8.0 Oreo | 26 | 60,8% |
| 8.1 Oreo | 27 | 53,5% |
| 9.0 Pie | 28 | 39,5% |
| 10. Android 10 | 29 | 8,2% |

Obr. 2.2: Distribúcia Android verzií aktívnych zariadení [34]

### 2.3.4 Vývoj na mobilnej platforme Android

Komunita zaoberajúca sa vývojom na platforme Android je veľmi rozsiahla, preto je na internete veľké množstvo tutoriálov, knižníc a aplikácií. Na vývoj sa používa oficiálne vývojové prostredie `Android Studio`, ktoré je postavené na známom vývojovom prostredí `IntelliJ IDEA` od spoločnosti `JetBrains`. Toto vývojové prostredie je možné nainštalovať na operačných systémoch `Windows`, `Mac`, `Linux` a dokonca aj `Chrome OS`. Výhodou je jeho dlhodobá podpora s pravidelnými aktualizáciami. Ďalšou výhodou je množstvo nástrojov, ktoré nemôžu chýbať v žiadnom modernom vývojovom prostredí, ako `debugger`, `logger` a `profiler`. Okrem toho je z neho možné priamo spravovať emulátor zariadení pomocou nástroja `AVD manager`, vďaka čomu je možne aplikáciu testovať na rôznych rozlíšeniach a verziách Android systému. [36]

Kapitola 3

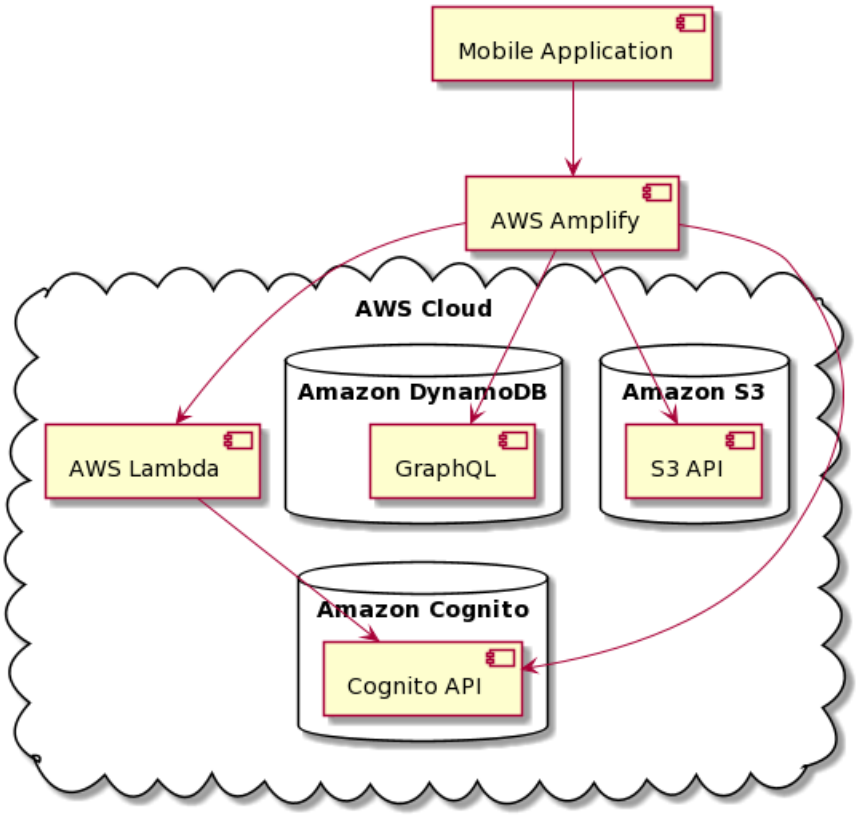
# Návrh

V tejto kapitole bude opísaný návrh mobilnej aplikácie na platforme Android spolu so serverovou časťou v `AWS Amplify`. Samotný návrh je odvodený z funkčných a nefunkčných požiadavkov aplikácie, ktoré boli analyzované v sekcií 2.1. Kapitola je rozdelená do dvoch hlavných sekcií. V prvej z nich 3.1 je opísaný návrh serverovej časti a v druhej 3.2 je opísaný návrh mobilnej aplikácie Android.

## 3.1 Serverová časť

Z analýzy poskytovateľov služieb `PaaS` v sekcií 2.2, bola vybraná služba `AWS Amplify`. Na tejto platforme je navrhnutá serverová časť aplikácie s funkčnými požiadavkami správy účtov, dát a súborov.

Diagram serverovej časti aplikácie tvorený zo služieb `AWS Amplify`, ktoré sú analyzované v sekcií 2.2.1.1, je znázornený na obrázku 3.1. Pomocou služby `AWS Amplify`, ktorej `SDK` zjednodušuje komunikáciu s ostatnými službami, bude prebiehať kompletná komunikácia so serverovou časťou aplikácie. Služba `Amazon Cognito` poskytuje jednoduchú správu účtov. Služba `Amazon DynamoDB`, ktorá predstavuje `NoSQL` databázu, s ktorou sa komunikuje pomocou `GraphQL`, je použitá na správu dát v aplikácií. Na správu súborov rôzneho typu sa používa služba `Amazon S3`. `AWS Lambda` je na `backend` logiku, ktorá má za úlohu spravovať účty.




Obr. 3.1: Diagram serverovej časti aplikácie

### 3.1.1 Správa účtov

Túto funkciu rieši služba `Amazon Cognito` spolu s `AWS Lambda`. Na strane servera je potrebné navrhnúť následujúce funkcie:

**Registrácia** – Registrácia nového účtu je súčasťou `Amazon Cognito`, kde je možné nakonfigurovať množstvo nastavení a podmienok pre účet. V tomto riešení je pri registrácií povinné zadať celé meno, email a heslo s minimálnou dĺžkou 8 znakov, ktoré obsahuje aspoň jedno číslo. Medzi vedľajšie atribúty patrí údaj, či je daný účet potvrdený správcom a `id` tímu (ktoré je zatiaľ stále rovnaké). Po registrácií obdrží používateľ na svoj email kód, ktorý zadá do aplikácie a overí tak svoj email, na ktorý mu môže byť v prípade zabudnutia hesla zaslané nové heslo. Všetko sa dá nakonfigurovať priamo v službe `Amazon Cognito`.

**Schválenie** – Po úspešnej registrácií je potrebné, aby správca nový účet schválil a zaradil ho do správnej kategórie. Na to je potrebné, aby správca obdržal zoznam všetkých nepotvrdených účtov, s čím pomôže práve služba `AWS Lambda`, kde je možné vytvoriť `API` s `endpointom` na zoznam týchto účtov. Následne je na schválenie vytvorený ďalší `endpoint`, kde sa vybranému účtu nastaví údaj, že je daný účet potvrdený správcom. Po zaradení účtu do kategórie sa účet schváli a vytvorí sa v databáze nový hráč. Následne sa môže tento schválený účet použiť na vstup do aplikácie.

**Odstránenie** – Správca môže zo zoznamu nepotvrdených účtov vybraný účet zamietnuť, a tým sa pomocou `endpointu` v `AWS Lambda`, určeného na vymazanie, účet vymaže z databázy účtov služby `Amazon Cognito`.

### 3.1.2 Správa dát

Na správu dát je použitá služba `Amazon DynamoDB`, ktorej `NoSQL` databáza je ideálnym nástrojom pre správu dát v tejto aplikácií. Ďalšou výhodou je komunikácia pomocou `GraphQL`. Diagram tried pre správu dát na obrázku 3.2 znázorňuje návrh jednotlivých entít vychádzajúcich z funkčných požiadavkov, ktoré sú súčasťou riešenia aplikácie. Zoznam jednotlivých entít:

**Team** – Táto entita predstavuje športový klub, teda najvyššiu entitu. Zatiaľ je aplikácia funkčná pre práve jeden športový klub, ale v budúcnosti je možne ju pripraviť pre iný športový klub niekoľkými zmenami v aplikácií.

**Category** – Jedná sa o kategóriu, do ktorej sú pridávaní noví hráči. Jedna z kategórií je určená pre správcov (fyzioterapeuti, tréneri a iní členovia vedenia športového klubu). Ostatné kategórie je možne vytvoriť alebo vymazať a pridávať do nich hráčov pri schvaľovaní nových účtov.

**Player** – Nejedná sa iba o hráča ale aj o správcu. Tento rozdiel rozpoznáva aplikácia podľa kategórie, v ktorej sa nachádza.

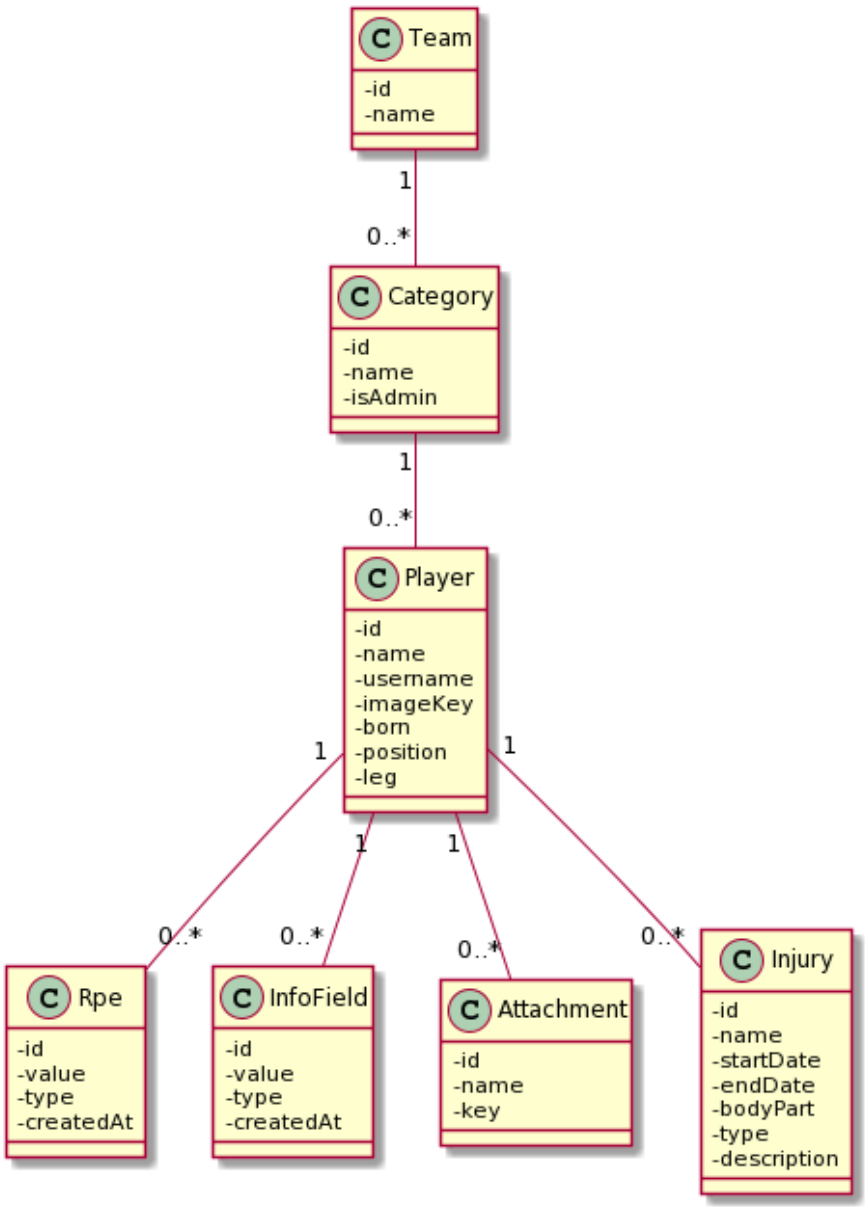
**Rpe** – Táto entita predstavuje systém monitorovania únavy, ktorý je reprezentovaný škálou hodnôt od 1 (malá únava) po 10 (vysoká únava). V aplikácií sú dva typy RPE:

- Po tréningu – Zadáva sa po tréningu alebo inej fyzickej záťaži.
- Po zobudení – Zadáva sa ihneď ráno po zobudení.

**InfoField** – Jedná sa o nameranú telesnú hodnotu. Momentálne sa monitorujú tieto typy: výška, váha, BMI, telesný tuk, hmotnosť bez tuku, podkožný tuk, viscerálny tuk, telesná voda, kostrové svalstvo, svalová hmota, kostná hmota, proteín a iné.

**Attachment** – Entita, ktorá reprezentuje prílohu. Obsahuje odkaz na súbor uložený v `Amazon S3`.

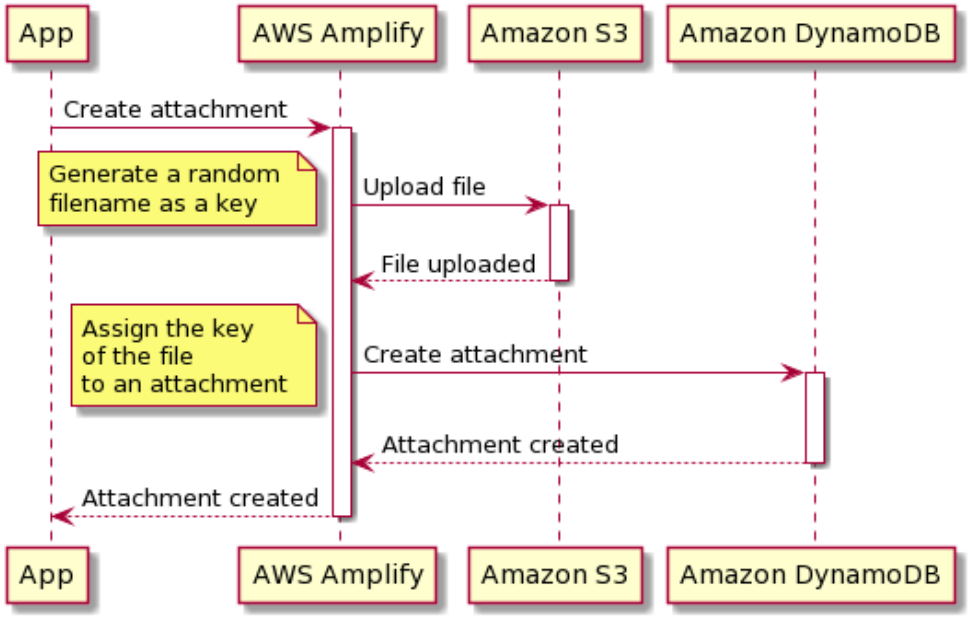
**Injury** – Zranenia sú rozdelené do 28 častí ľudského tela. Okrem toho sa rozlišujú typy: kontaktné zranenie, bezkontaktné zranenie, choroba alebo iný typ zranenia.




Obr. 3.2: Diagram tried pre správu dát

### 3.1.3 Správa súborov

Služba `Amazon S3` je cielená na správu súborov rôzneho typu. Služba umožňuje jednoducho vytvoriť databázu súborov, s ktorou je aplikácia schopná komunikovať priamo pomocou `AWS Amplify SDK`. V aplikácií sa používajú súbory pri obrázkoch jednotlivých hráčov a ich príloh. Tieto súbory sa musia namapovať na jednotlivé dáta uložené v `Amazon DynamoDB` pomocou kľúča, ktorý je náhodne vygenerovaný vždy pri nahrávaní súboru. Vytváranie prílohy v `Amazon DynamoDB` spolu s nahrávaním súboru do `Amazon S3` je ukázaný na sekvenčnom diagrame na obrázku 3.3.



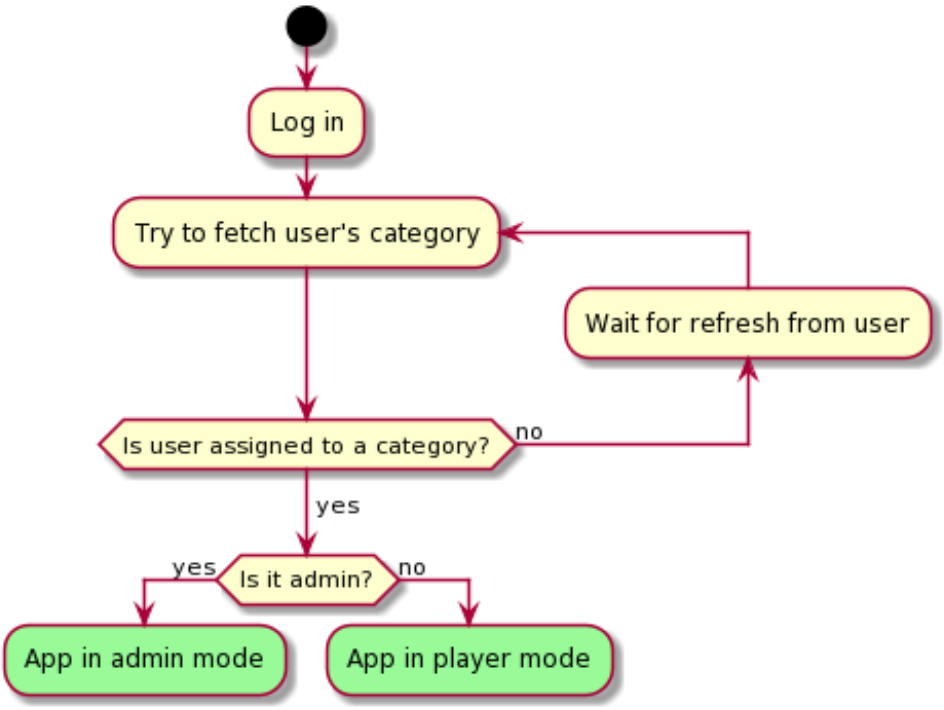

Obr. 3.3: Sekvenčný diagram pre vytvorenie prílohy

## 3.2 Mobilná aplikácia Android

V predošlej sekcií 3.1 bola opísaný návrh serverovej časti aplikácie. Táto sekcia je venovaná opisu návrhu mobilnej aplikácie na platforme Android. Ako prvé je opísaný návrh rozdelenia a vplyv jednotlivých rolí používateľov v sekcií 3.2.1. Následne je v sekcií 3.2.2 opísaný návrh jednotlivých funkčných požiadavkov zo sekcie 2.1.1. Nakoniec je opísaný návrh používateľského rozhrania v sekcií 3.2.3.

### 3.2.1 Role používateľov

Po prihlásení používateľa na svoj účet je potrebné zistiť jeho rolu. V aplikácií sa rozlišujú dve role, a to rola správcu a hráča. Návrh týchto rolí je opísaný v ich jednotlivých sekciách. Pre vyhodnotenie role je potrebné zistiť, do ktorej kategórie je účet zaradený. Na základe typu kategórie sa ďalej vyhodnotí rola používateľa a aplikácia sa spustí v móde pre danú rolu. Tento postup je znázornený diagramom aktivít na obrázku 3.4.




Obr. 3.4: Diagram aktivít pre priradenie role po prihlásení do aplikácie

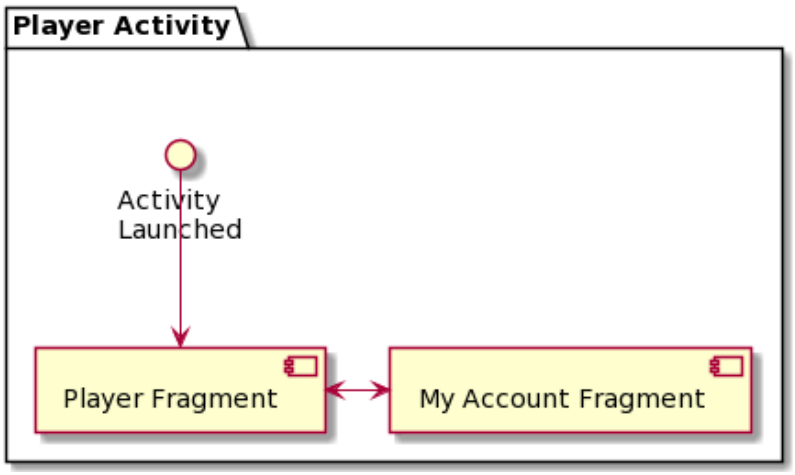
#### 3.2.1.1 Rola hráča

Používateľ s touto rolou môže sledovať a spravovať (s rôznymi obmedzeniami, viz tabuľka 2.1) iba svoj účet. Vytvorenie účtu s rolou hráča vznikne tak, že správca zaradí nový nezaradený účet do správcom vytvorenej kategórie. Ak je rola účtu po prihlásení typu hráč, spustí sa `Player Activity`. Tu sa používateľ môže pohybovať s pomocou `Bottom Navigation View` medzi fragmentami:

**Player Fragment** – Predvolený fragment, kde sú všetky dáta a prílohy pre hráčsky profil prihláseného používateľa. Jeho návrh je detailnejšie opísaný v sekcií 3.2.2.2

**My Account Fragmnet** – V tomto fragmente si používateľ môže nastaviť rôzne údaje svojho hráčskeho profilu, medzi ktoré patrí napríklad fotografia, dátum narodenia alebo pozícia na ihrisku. Okrem toho je tu možné nastaviť upozornenie na vyplnenie dát v aplikácií, na ktoré bude používateľ následne upozornený notifikáciu vo vybraný čas každý deň. Taktiež je tu funkcia odhlásenia z účtu.

Rozdelenie fragmentov pre rolu hráča je znázornene na obrázku 3.5.




Obr. 3.5: Rozdelenie fragmentov pre rolu hráča

#### 3.2.1.2 Rola správcu

Táto rola má väčšie právomoci ako rola hráča. Správca môže sledovať a spravovať (s rôznymi obmedzeniami, viz tabuľka 2.1) profily všetkých hráčov. Vytvorenie účtu s rolou správca vznikne tak, že iný správca zaradí nový nezaradený účet do kategórie určenej pre správcov. Ak je rola účtu po prihlásení typu správca, spustí sa `Admin Activity`. Tu sa používateľ môže pohybovať s pomocou `Bottom Navigation View` medzi fragmentami:

**Categories Fragment** – Predvolený fragment, ktorý slúži na správu kategórií. Správca tu môže vytvoriť alebo vymazať kategóriu. Obsahuje zoznam kategórií, kde sa po kliknutí na jednu z kategórií otvorí `Players Fragment`.

**Cognito Users Fragment** – Fragment, ktorý slúži na schvaľovanie a zamietnutie nových nezaradených účtov. Správca v ňom vidi zoznam týchto účtov, v ktorých môže vykonávať funkcie zaradenia do kategórie alebo funkciu zamietnutia (a následne vymazanie).

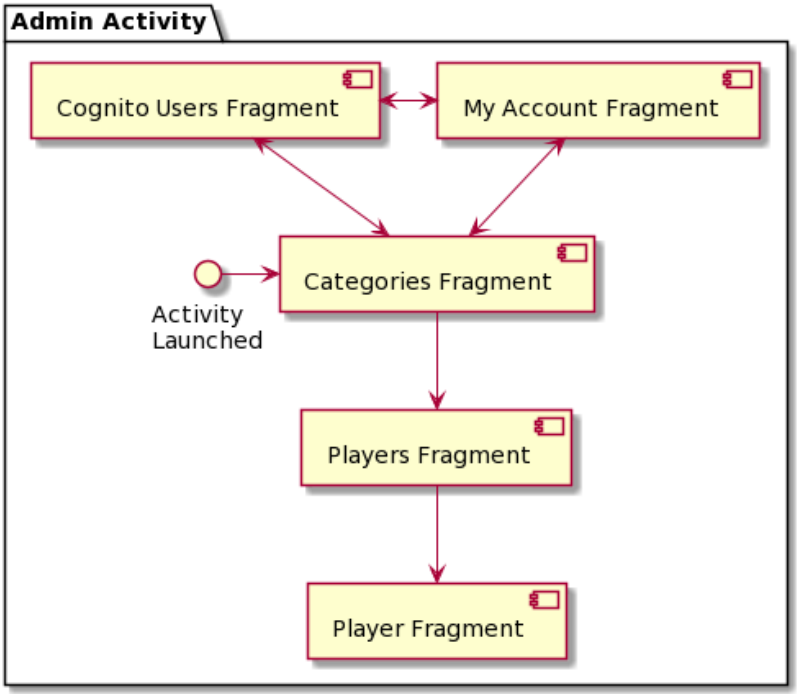
**My Account Fragment** – V tomto fragmente si môže správca nastaviť fotografiu. Okrem toho je tu možné nastaviť upozornenie na vyplnenie dát v aplikácií, na ktoré bude správca následne upozornený notifikáciou vo vybraný čas každý deň. Taktiež je tu funkcia odhlásenia z účtu.

V `Admin Activity` sú okrem vyššie spomínaných fragmentov, vyhradené fragmenty pre správu hráčov:

**Players Fragment** – Fragment so zoznamom hráčov vybranej kategórie. Riadok hráča obsahuje jeho fotografiu a meno. Po kliku na hráča zo zoznamu sa otvorí `Player Fragment`.

**Player Fragment** – Fragment, kde sú všetky dáta a prílohy pre profil vybraného hráča. Jeho návrh je detailnejšie opísaný v sekcií 3.2.2.2

Rozdelenie fragmentov pre rolu správcu je znázornene na obrázku 3.6.




Obr. 3.6: Rozdelenie fragmentov pre rolu správcu

Táto rola je nastavená trénerom, fyzioterapeutom, lekárskemu personálu a iným členom vedenia športového klubu.

### 3.2.2 Funkcie mobilnej aplikácie

Táto sekcia opisuje návrh funkčných požiadavkov aplikácie analyzovaných v sekcií 2.1.1, ktoré sú rozdelené do jednotlivých sekcií.

#### 3.2.2.1 Sprava účtov

Medzi funkcie, ktoré reprezentujú správu účtov priamo z mobilnej aplikácie patrí:

**Prihlásenie** – Po spustení aplikácie sa používateľovi zobrazia polia na vyplnenie prihlasovacích údajov a tlačidlo na prihlásenie. Táto funkcia je súčasťou `Login Fragment`. Po prihlásení aplikácia vyhodnotí rolu, ktorej návrh je opísaný v sekcií 3.2.1. Po prihlásení sa uloží token do lokálnej databázy. Vďaka tomu používateľ nemusí pri opätovnom spustení aplikácie znova zadávať prihlasovacie údaje a je prihlásený automaticky.

**Registrácia** – Ak používateľ ešte nemá vytvorený svoj účet, môže ho jednoducho vytvoriť priamo v aplikácií. Na registráciu je potrebné vyplniť údaje: celé meno, email a heslo. Na vyplnenie týchto údajov slúžia zreteľne opísané polia. Správnosť vyplnených údajov je kontrolovaná na strane aplikácie. Meno nesmie obsahovať špeciálne znaky a čísla, email musí mať správnu štruktúru a heslo musí mať minimálne 8 znakov a aspoň jednu číslicu. Na chybné vyplnenie je používateľ upozornený zrozumiteľnými chybovými hláškami. Registrácia prebieha v `Sign Up Fragment`. Po potvrdení registračných údajov a kliknutí na potvrdzovacie tlačidlo, sa aplikácia presunie do `Email Confirmation Fragment` a na registrovaný email bude zaslaný potvrdzovací kód. Tento fragment obsahuje pole na vyplnenie 6-miestneho kódu. Po vyplnení správneho kódu a potvrdením, sa aplikácia presunie do `Login Fragment`, kde sa môže používateľ prihlásiť pomocou nového, práve registrovaného účtu.

**Odhlásenie** – Po odhlásení sa aplikácia dostane do stavu, kde vyžaduje prihlasovacie údaje od používateľa (`Login Fragment`). Táto funkcia je dostupná ako tlačidlo v `My Account Fragment`. Po kliknutí na toto tlačidlo sa zobrazí `Alert Dialog`, kde môže akciu potvrdiť alebo zamietnuť. Po odhlásení sa vymažú všetky uložené dáta z lokálnej perzistentnej databázy aplikácie (`Room`).

**Schvaľovanie** – Správca môže priamo z aplikácie schvaľovať alebo zamietať nové nezaradené účty. Táto funkcia je v `Cognito Users Fragment`, ktorá obsahuje zoznam nezaradených účtov. Pre schválenie je potrebné vybrať kategóriu, do ktorej chce správca zaradiť nového používateľa. Ak chce účet odstrániť, vyberie túto možnosť, ktorá je reprezentovaná jasným a zrozumiteľným označením. Funkcia schvaľovania je dostupná iba pre rolu správcu.

#### 3.2.2.2 Správa dát a súborov

Táto sekcia opisuje návrh správy dát a súborov v mobilnej aplikácií Android. Jedná sa o primárne funkčné požiadavky a ich obsadenie predstavuje veľkú časť aplikácie.

Návrh tejto časti mobilnej aplikácie vychádza z návrhu správy dát v serverov časti aplikácie opísanej v sekcií 3.1.2, medzi ktoré patria:

**Team** – Túto entitu nie je možné vytvoriť, ani spravovať z aplikácie. Jej `id` je vložené priamo v aplikácií.

**Category** – Zoznam kategórií spolu s ich správou (pridávanie a vymazanie) je v `Categories Fragment`.

**Player** – Táto entita sa vytvorí po schválení nového účtu správcom, ktorého súčasťou je zaradenie do kategórie. V `My account Fragment` má hráč možnosť vyplniť údaje svojho hráčskeho profilu, ako dátum narodenia, preferovaná noha a pozícia na ihrisku. Okrem toho si tu môže hráč nahrať svoju fotografiu, pomocou ktorej môže neskôr správca identifikovať daného hráča.

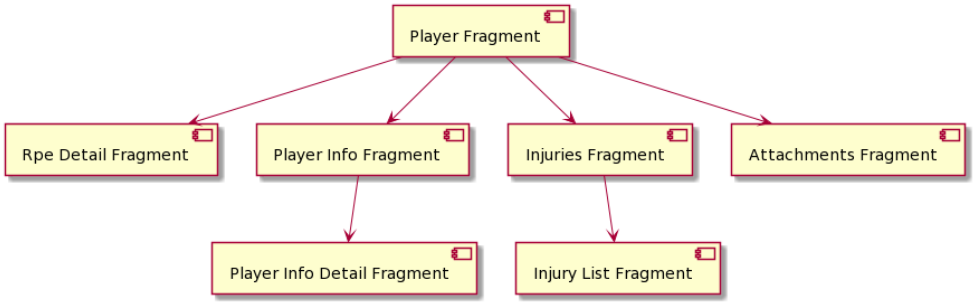
**Rpe** – V prípade role hráča, ktorý má právo vytvárať túto entitu, sú na primárnom fragmente reprezentujúceho profil hráča (`Player Fragment`), polia so sliderom, pomocou ktorých môžu zadávať `RPE` hodnoty pre obidva typy v daný deň. Správcovi sa na tomto mieste zobrazí posledná hodnota, ktorú hráč zadal. Pre zobrazenie histórie týchto hodnôt pre obidva typy v grafickej podobe, je určený `Rpe Detail Fragment`.

**InfoField** – V `Player Info Fragment` je zoznam posledných vyplnených hodnôt pre všetky typy tejto entity. Po kliku na jednu z nich sa otvorí `Player Info Detail Fragment`, kde je história konkrétneho typu v zozname a grafickej podobe. V tomto fragmente je možné pridávať nové hodnoty. Taktiež v ňom majú obidve role možnosť vybrané hodnoty mazať alebo upravovať.

**Injury** – Rozdelenie zranení častí ľudského tela je reprezentované 3D modelom ľudského tela s bodmi jednotlivých častí. Ak sa v danej časti nachádza aktívne zranenie (nemá dátum ukončenia), tak je daný bod vyznačený červenou farbou, v opačnom prípade je bod vyznačený modrou farbou. Každý bod má číslo s počtom všetkých zranení (aktívnych aj neaktívnych) v danej oblasti. Tento 3D model je zobrazený v `Injuries Fragment`. Po kliknutí na určitý bod sa otvorí `Injury List Fragment`, ktorý obsahuje zoznam zranení pre daný bod. V tomto fragmente môže správca spravovať zranenia pre daný bod.

**Attachment** – Prílohu môžu vytvoriť a vymazať iba správcovia. Na správu tejto entity je určený `Attachments Fragment`. Pre vytvorenie prílohy je potrebné načítať súbor z pamäte mobilného zariadenia zariadenia. Hráč si môže pozrieť všetky svoje prílohy.

Rozdelenie fragmentov pre jednotlivé funkcie na správu dát a súborov určených pre hráčsky profil, ktorých návrh je opísaný vyššie, je znázornené na obrázku 3.7.




Obr. 3.7: Rozdelenie fragmentov pre jednotlivé funkcie na správu dát a súborov

### 3.2.3 Návrh používateľského rozhrania

Samotný návrh používateľského rozhrania má hlbší význam ako navrhnúť výzor aplikácie. Pri návrhu používateľského rozhrania mobilnej aplikácie je potrebné sústrediť sa na jej použiteľnosť a intuitívnosť. Existuje množstvo návodov pre správny návrh `UI` mobilných aplikácií Android použitím už pripravených `UI` komponentov. Tieto komponenty sú opísané v sekcií 3.2.3.2.

Pre návrh `UI` sa dá taktiež použiť univerzálny návod, ktorý vytvoril `UI` expert `Jakob Nielsen`. Tento návod pozostáva z desiatich pravidiel, na ktoré by sa mal brať ohľad pri návrhu `UI`. Medzi tieto pravidla patrí:

1. **Viditeľnosť stavu systému** – Aplikácia by mala vždy informovať o stave, v ktorom sa nachádza, pomocou spätnej väzby. Nemalo by sa napríklad stať, že by aplikácia zamrzla na nejaký čas bez upozornenia používateľa. V aplikácií sa používa natívny `Content Loading Progress Bar`, ktorý sa vždy zobrazí pri komunikácií so serverom alebo načítaní dát a súborov.

2. **Zhoda medzi systémom a realitou** – Aplikácia by mala hovoriť jazykom používateľa, ktorý mu je známy a takisto používať prirodzené slová a súvetia. Aplikácia je navrhnutá pre 2 jazyky, slovenský a anglicky, s použitými odbornými slovami z bežného slovníka mobilných aplikácií.

3. **Minimálna zodpovednosť** – Používateľ sa často dostane vlastnou chybou do stavu, z ktorého sa chce vrátiť. Na to sa pri mobilnom vývoji používa `Alert Dialog`, ktorý pri určitých akciách vytvorí okno s potvrdením akcie. Napríklad akcie, ako je správa dát, umožňujú v niektorých prípadoch vymazanie alebo úpravu.

4. **Zhoda s použitou platformou a obecnými štandardmi** – `UI` aplikácie je tvorené z natívnych `UI` komponentov, ktoré sú opísané v sekcií 3.2.3.2. Aj vďaka tomu je aplikácia intuitívna pre bežných používateľov Android aplikácií.

5. **Prevencia chýb** – V aplikácií by malo vznikať minimum chýb, čomu je veľmi ťažké zabrániť. Na miestach, kde môže vzniknúť chyba, by mal byť používateľ na to vopred upozornený. Jedným z príkladov v aplikácií môže byť, keď chce správca vymazať kategóriu, v ktorej sa nachádzajú hráči. V takom prípade vyskočí chybová hláška, že kategória musí byť pred vymazaním prázdna.

6. **Pozriem a vidím** – Aplikácia by mala byť intuitívna a nemala by vyžadovať od používateľa pamätať si nejaké zložité postupy. Toto pravidlo za dá docieliť rozumným rozdelením `UI` a zvýraznenia dôležitých prvkov. Cieľom je znížiť kognitívne úsilie používateľa.

7. **Flexibilita a efektivita** – Aplikácia by mala mať možnosť využitia rôznych skratiek alebo šablón, ktoré sú cielené na skúsenejších používateľov danej aplikácie. Táto funkcia by nemala nijako ovplyvniť alebo zmiasť nového používateľa.

8. **Minimalizmus** – Aplikácia by mala v určitom stave zobrazovať iba dôležité informácie a nemala by rušiť pozornosť používateľa zbytočnými a prebytočnými informáciami. Aplikácia je preto navrhnutá pre väčší počet fragmentov, ktoré sú viazané k určitej funkcií a vďaka tomu fragment neobsahuje prebytočné informácie.

9. **Zmysluplné chybové hlášky** – Každá chybová hláška aplikácie by mala jasne opisovať príčinu problému a návrh na jeho riešenie. V aplikácií je na výpis chybových hlášok použitý `Toast`, ktorý vypíše danú chybovú hlášku na spodnej časti obrazovky zariadenia. Chybové hlášky sú taktiež zobrazené pri zle vyplnených údajov v poli na vyplnenie.

10. **Pomoc a dokumentácia** – Podľa tohto pravidla by mala byť aplikácia navrhnutá tak, aby nebolo potrebné na jej používanie čítať dokumentáciu alebo iný typ pomoci.

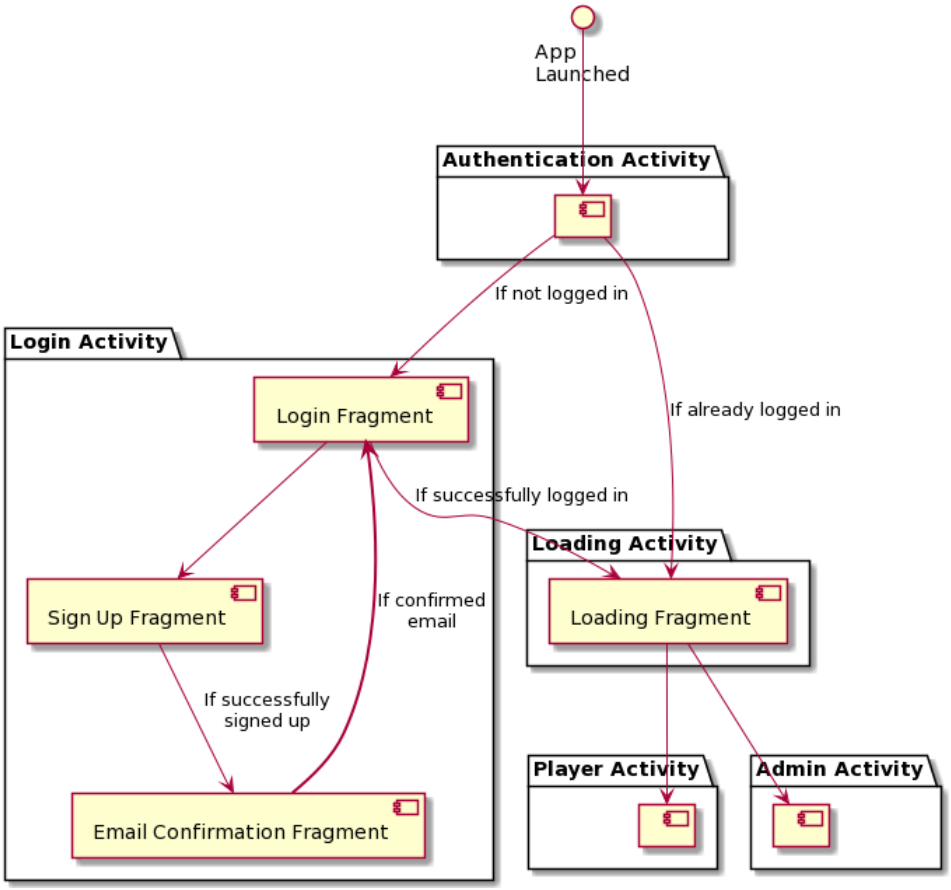
[37]

Tieto pravidlá výrazne pomohli pri návrhu mobilnej aplikácie Android. Samotné aplikovanie týchto pravidiel do dokonalosti nie je úplne triviálne. Pre zdokonalovanie používateľského rozhrania je potrebné reagovať na spätnú väzbu od používateľov. Ďalej tomu môže pomôcť nástroj `Analytics`, ktorý je súčasťou služby `Google Firebase`, ktorá bola analyzovaná v sekcií 2.2.1.2.

#### 3.2.3.1 Rozdelenie fragmentov aplikácie

Aplikácia má fragmenty rozdelené do 5 `activity` častí. Predvolená `Authentication Activity` má za úlohu zistiť, či je v aplikácií uložený token po prihlásení. Na základne toho rozhodne, či ma aplikácia pokračovať ďalej a načítať dáta prihláseného používateľa (`Loading Activity`) alebo ísť do stavu, kde sa môže používateľ prihlásiť alebo registrovať (`Login Activity`). V `Loading Activity` aplikácia po načítaní dát prihláseného používateľa rozhodne, o ktorú rolu sa jedná a podľa toho prejde do `Player Activity` alebo `Admin Activity`, čo je bližšie opísané v návrhu rolí v sekcií 3.2.1.

Rozdelenie fragmentov aplikácie v jednotlivých `activity` častiach je znázornené na obrázku 3.8.

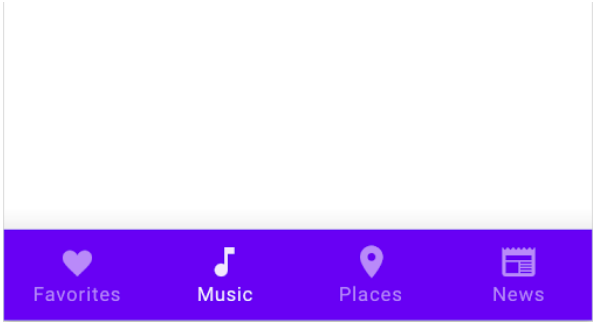



Obr. 3.8: Rozdelenie fragmentov aplikácie

#### 3.2.3.2 UI komponenty používateľského rozhrania mobilnej aplikácie Android

Podľa určitých pravidiel sa odporúča používať natívne `UI` komponenty. Jedným z dôvodov je aj to, že používatelia sú zvyknutí na určitej platforme používať práve určené komponenty pre danú platformu, čo zaručí intuitívnosť pri používaní aplikácie. Pri návrhu mobilnej aplikácie na platforme Android sa vychádzalo z týchto natívnych `UI` komponentov:

**Bottom Navigation View** – Tento `UI` komponent je súčasťou kolekcie `Material Components`, ktorá je vyvíjaná spoločnosťou `Google`. tento komponent poskytuje panel na celkom dolnej časti aplikácie, pomocou ktorého umožňuje používateľovi pohyb medzi rôznymi časťami aplikácie. Ukážka tohto `UI` komponentu je na obrázku 3.9. [38]

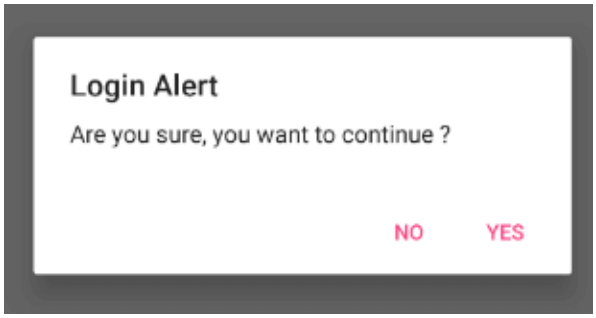



Obr. 3.9: Bottom Navigation View príklad [38]

**Text View** – Jednoduchý `UI` komponent, ktorý sa používa na zobrazenie ľubovolného textu. Text tohto komponentu môže byť statický alebo dynamicky sa meniaci priamo zo zdrojového kódu aplikácie.

**Text Input Edit Text** – `UI` komponent predstavujúci pole na vyplnenie, ktorý slúži na získanie spätnej väzby od používateľa textovou formou. Tento `UI` Komponent obsahuje popis a dokáže upozorňovať na rôzne chyby pri vyplnení. V aplikácií sa používa napríklad pri zadávaní prihlasovacích údajov, údajov pri registrácií alebo pri zadávaní textu pre nejaký typ dát (názov prílohy, nameraná telesná hodnota atď.).

**Alert Dialog** – Pomocou tohto nástroja môže vývojár zobraziť okno s ľubovolným textom a tlačidlami, ktorých kliknutie reprezentuje určitú akciu. Väčšinou je využívaný na potvrdenie určitého netriviálneho kroku, pri ktorom sa mení stav aplikácie. Pozostáva z nadpisu, textového obsahu a tlačidiel. Jeden z príkladov použitia znázorňuje obrázok 3.10 [39]




Obr. 3.10: Alert Dialog príklad [39]

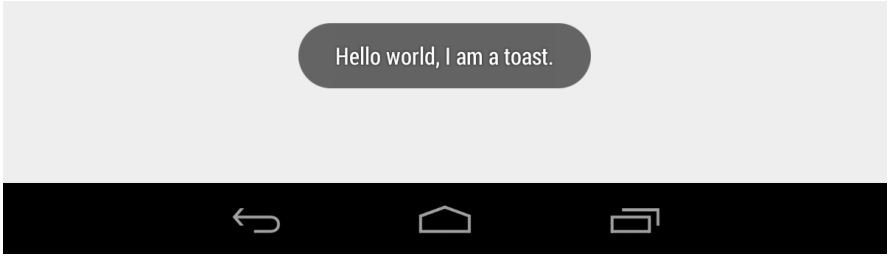
**Dialog Fragment** – Špeciálny typ fragmentu, ktorý vytvorí okno s ľubovolným `UI`. Ide tak o prispôsobiteľnejší nástroj na vytváranie okien ako `Alert Dialog`. V aplikácií sa použije na pridávanie alebo upravovanie dát.

**Floating Action Button** – Tento `UI` komponent predstavuje okrúhle tlačidlo, najčastejšie umiestnené v pravom dolnom rohu aplikácie. V aplikácií sa použije ako tlačidlo na otvorenie `Dialog Fragment`, pomocou ktorého môže správca vytvoriť novú kategóriu. Väčšinou sa tento komponent používa vo fragmente so zoznamom nejakých dát.

**App Compat Button** – Tento `UI` komponent predstavuje jednoduché tlačidlo, ktorého štýl sa dá jednoducho prispôsobiť k dizajnu aplikácie. Využíva na miestach aplikácie, kde môže používateľ vykonať určitý typ akcie. V aplikácií sa používa na potvrdenie, pridávanie, úpravu atď.

**Image View** – Jednoduchý `UI` komponent, ktorý sa používa na zobrazenie ľubovolného obrázku. Obrázok tohto komponentu môže byť statický alebo dynamicky sa meniaci priamo zo zdrojového kódu aplikácie. S dynamickými zmenami pomôže knižnica `Glide`, ktorá bola analyzovaná v analýze technológií a knižníc mobilnej aplikácie Android v sekcií 2.3.2.

**Toast** – Tento nástroj slúži na zobrazenie rýchlej správy pre používateľa. Táto správa sa zobrazí v dolnej časti aplikácie. V aplikácií je použitý pri zobrazovaní chybových hlášok alebo iných upozornení pre používateľa. Na obrázku 3.11 je uvedený príklad správy zobrazenej pomocou nástroja Toast.



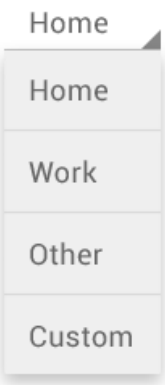

Obr. 3.11: Toast príklad [40]

**Content Loading Progress Bar** – Aplikácia sa niekedy dostane do stavu, kedy potrebuje určitý čas na spracovanie nejakej požiadavky. Aby mohol byť používateľ o tomto stave oboznámený, používa sa na to práve tento `UI` komponent. Dajú sa použiť dva štýly tohto komponentu, okrúhly a horizontálny. Obidva štýly svojou animáciou dávajú používateľovi jasne najavo, že aplikácia na niečom pracuje alebo sa niečo načítava. V aplikácií sa používa okrúhly typ a nachádza sa väčšinou na miestach kde sa vyžaduje komunikácia so serverom.

**Recycler View** – Pomocou tohto `UI` komponentu sa dá jednoducho a efektívne zobraziť väčší počet dát vo forme vertikálneho zoznamu, horizontálneho zoznamu alebo ako `grid`. Výhodou tohto komponentu je jeho chytré zaobchádzanie s jednotlivými elementami. Ak sa nejaký element dostane mimo viditeľný obraz, tak sa jeho `view` nezničí, ale použije sa pre iný element. Tento systém výrazne zlepší výkon aplikácie a taktiež spotrebu energie zariadenia, na čo sa pri vývoji mobilných aplikácií berie ohľad. [41]

**Swipe Refresh Layout** – Tento nástroj sa používa na aktualizáciu určitého obsahu dát aplikácie. Najčastejšie sa používa spolu s `Recycler View`, ale môže sa tiež použiť s ľubovolným skrolovacím `UI` komponentom. Pomocou neho môže používateľ ťahom nadol zobraziť `Content Loading Progress Bar` v štýle kruhu, ktorý označuje aktualizáciu daného obsahu. Po aktualizácií kruh zmizne a používateľ vidi najnovšiu verziu dát. V aplikácií sa tento nástroj používa na všetkých miestach, kde sa zobrazujú dáta zo serveru.

**Spinner** – Po kliku na tento `UI` komponent, sa zobrazí zoznam na výber názvov určitých dát vo formáte `String`. Po výbere sa zmení text komponentu podla názvu vybranej položky. Príklad tohto komponentu je na obrázku 3.12.




Obr. 3.12: Spinner príklad [42]

Pomocou týchto komponentov sa výrazne zjednodušil návrh používateľského rozhrania mobilnej aplikácie Android. Už pred samotnou implementáciou aplikácie sa vďaka nim dá jednoducho predstaviť, ako by mala aplikácia vyzerať. K správnemu použitiu týchto komponentov pomôžu rôzne návody a nasledovania `best practises`.

Kapitola 4

# Implementácia

V tejto kapitole je opísaný proces implementácie serverovej časti aplikácie v sekcií 4.1 a mobilnej aplikácie na platforme Android v sekcií 4.2. Implementácia serverovej časti a mobilnej aplikácie vychádza z návrhu, ktorý bol opísaný v kapitole 3.

## 4.1 Serverová časť

Táto sekcia opisuje implementáciu serverovej časti aplikácie. Na začiatku bude opísaný postup implementácie a konfigurácie `AWS Amplify` a následne bude opísaný postup implementácie aj konkrétnych služieb, ktoré poskytuje.

### 4.1.1 AWS Amplify

Na začiatok je potrebné vytvoriť si `AWS` účet na ich oficiálnej stránke. Pri registrácií je potrebné vyplniť osobne údaje a číslo kreditnej karty, ktorá bude použitá pri platbe. Tým, že táto služba poskytuje ročnú bezplatnú verziu a nepredpokladá sa prekročenie limitov, nemali by byť pri použití tejto služby žiadne poplatky.

Po úspešnej registrácií je potrebné nainštalovať `Amplify CLI`. Na jeho inštaláciu sa odporúča použiť nástroj `Node.js`. Pomocou `Amplify CLI` je možné ovládať niektoré služby a nastavenia `AWS Amplify` priamo z príkazového riadka počítača. Po inštalácií je potrebné `AWS Amplify` nakonfigurovať a spojiť ho so svojim registrovaným účtom. Táto konfigurácia spočíva vo vytvorení `API` kľúča v `AWS` konzole svojho účtu.

Po úspešnej konfigurácií `AWS Amplify`, je na čase vytvoriť nový Android projekt s potrebnými knižnicami pre správu `AWS Amplify` služieb. [43]

### 4.1.2 Amazon Cognito

Pre implementáciu správy účtov v `AWS Amplify` sa používa služba `Amazon Cognito`. Táto služba sa dá jednoducho pridať do mobilnej aplikácie pomocou `Amplify CLI` jedným príkazom. Pred pridaním je potrebné pre správnu konfiguráciu služby nastaviť primárne nastavenia priamo z príkazového riadka. Medzi tieto nastavenia patria napríklad typ prihlasovacích údajov (email alebo prihlasovacie meno), podmienky hesla a iné.

Po úspešnej konfigurácií služby `Amazon Cognito`, je potrebné pridať do Android projektu potrebnú knižnicu pre správu účtov z tejto služby pomocou `AWS Amplify` priamo z aplikácie.

Po prihlásení do `AWS` konzoly svojim účtom, je možné spravovať túto službu pomocou ich prostredia. Okrem zoznamu registrovaných účtov, pridávania a odstraňovania nových účtov, sa tu dá nastaviť množstvo funkcií, od sekundárnych atribútov pri registrácií účtov, po nastavenie potvrdenia emailu pri registrácií zaslaním kódu na email. [44]

### 4.1.3 Amazon S3

Na nahrávanie, sťahovanie a vymazávanie súborov rôzneho typu v `AWS Amplify` sa používa služba `Amazon S3`. Implementácia tejto služby sa dá taktiež zrealizovať pomocou `Amplify CLI` jedným príkazom. Implementácia spočíva vo vytvorení `bucketu`, ktorý obsahuje priečinky a v nich súbory.

Pre pridanie tejto služby do Android projektu, je takisto potrebné mať už nakonfigurované `AWS Amplify` v projekte. Následne je potrebné pridať knižnicu pre službu `Amazon S3`, aby sa mohli nahrávať, sťahovať a vymazávať súbory priamo z aplikácie.

V `AWS` konzole, je možne pri tejto službe nastaviť práva a viditeľnosť jednotlivých `bucketov` alebo priečinkov, v ktorom sú súbory uložené. [45]

### 4.1.4 Amazon DynamoDB

Táto služba slúži na vytvorenie `NoSQL` databázy, ktorá podporuje komunikáciu pomocou `GraphQL`. Tak, ako pri predchádzajúcich službách, aj služba `Amazon DynamoDB` sa dá jednoducho pridať do Android projektu s `Amplify CLI`. Aj pri tejto službe je potrebné pridať konkrétnu knižnicu do Android projektu.

Na rozdiel od predchádzajúcich služieb je implementácia náročná v tom, že je potrebné navrhnúť model entít, ktorý bude reprezentovať dáta, ktoré je potrebné spravovať na serverovej časti aplikácie. Model pre túto aplikáciu je navrhnutý v sekcií 3.1.2.

Po úspešnom pridaní tejto služby do Android projektu, je potrebné vytvoriť daný model entít. Ten sa implementuje ako `GraphQL` schéma v automaticky vygenerovanom súbore `schema.graphl`. Časť tejto schémy implementovanej v serverovej časti aplikácie je v zdrojovom kóde 1. V zdrojovom kóde je vidieť, ako sa tvoria jednotlivé entity pomocou anotácie `@model` a ich atribút so základnými alebo vlastnými typmi. Taktiež ukazuje, ako sa implementuje prepojenie a vzťah vybraných entít. [46]

```
type Team @model {
  id: ID!
  name: String!
  categories: [Category]
                @connection(keyName: byTeam, fields: [id])
}

type Category @model
@key(
name: byTeam,
fields: [teamID],
queryField: getCategoriesByTeamId) {
  id: ID!
  name: String!
  teamID: ID!
  isAdmin: Boolean!
  players: [Player]
            @connection(keyName: byCategory, fields: [id])
}
```

Zdrojový kód 1: Časť GraphQL schémy

Aj pre túto službu je vytvorené prostredie v AWS konzole, kde je možné vidieť všetky dáta v databáze a nad nimi robiť ručne CRUD operácie. Taktiež je možné využiť prostredie na tvorbu dotazov a mutácií, ktoré sa v ňom dajú priamo aplikovať na správu tejto databázy.

### 4.1.5 AWS Lambda

Táto služba slúži na implementáciu REST API s business logikou na strane serverovej časti aplikácie. Jej pridanie je taktiež možne realizovať cez príkazový riadok pomocou Amplify CLI. Pri pridávaní tejto služby je potrebné vytvoriť endpoint, ktorý bude môcť aplikácia používať na volanie tohto API.

V AWS konzole je prostredie, kde je možné písať zdrojový kód v jazykoch Python, Go, Java a NodeJS. Odporúčaný jazyk na implementáciu tohto API je NodeJS a preto bol zvolený. Prostredie taktiež umožňuje pridávať knižnice a rôzne skripty, ktoré sa potom dajú spúšťať zo zdrojového kódu. Na to sa dajú použiť vrstvy, do ktorých sa nahrajú pridané knižnice alebo skripty a tieto vrstvy sa dajú zdielať medzi viacerými AWS Lambda funkciami. Taktiež sa dajú pridať globálne environmentálne premenné (process.env).

Táto služba sa v serverovej časti aplikácie používa na správu účtov. Pre komunikáciu s Amazon Cognito je použité AWS SDK pre NodeJS. Pomocou tohto SDK je možné používať všetky CRUD operácie. Medzi endpointy na správu účtov implementované v tomto REST API patrí:

**GET /users/:teamId** – Pomocou tohto endpointu má správca možnosť získať zoznam aktuálne neschválených účtov daného tímu. Správca obdrží všetky potrebné informácie na identifikáciu účtu, medzi ktoré patrí celé meno a email. Okrem toho získa aj unikátne userId, ktoré potrebuje na použitie nižšie opísaných operácií.

**POST /users/:userId/confirm** – Tento endpoint slúži na schvaľovanie účtu. Správca tak môže schvaľovať neschválené účty pomocou ich userId, ktoré získal z predošlého GET endpointu.

**DELETE /users/:userId** – Správca môže týmto endpointom vymazať vybraný účet.

V mobilnej aplikácií má právo používať tieto funkcie iba používateľ s rolou správcu. Toto obmedzenie nie je žiadnym spôsobom ošetrené na serverovej časti aplikácie.

## 4.2 Mobilná aplikácia Android

Táto sekcia opisuje proces implementácie mobilnej aplikácie na platforme Android. V sekcií je opísaná komunikácia so serverovou časťou aplikácie a použitá architektúra MVVM. Okrem toho sú opísané použité nástroje a knižnice pri implementácií niektorých problémov.

### 4.2.1 Komunikácia so serverovou časťou aplikácie

Na komunikáciu so serverovou časťou aplikácie implementovanej pomocou `AWS Amplify` používa mobilná aplikácia Android `AWS Amplify SDK`. Vďaka tomuto `SDK` a knižniciam pre konkrétne služby od `AWS` je implementácia tejto komunikácie na pár riadkov. `SDK` poskytuje `singleton` objekt `Amplify`, ktorý je možné použiť v ľubovolnej časti kódu. V mobilnej aplikácií sa komunikuje s jednotlivými službami serverovej časti aplikácie takto:

**Amazon Cognito** – Na prihlasovanie, odhlasovanie a registráciu sa v aplikácií používa objekt `Amplify.Auth`, ktorý má na tieto operácie pripravené funkcie `signIn, signOut` a `signUp`. Objekt taktiež obsahuje funkciu `confirmSignUp`, ktorá slúži na schválenie emailu.

**Amazon S3** – Pre nahratie, stiahnutie a vymazanie súboru z databázy sa používa objekt `Amplify.Storage`, ktorý obsahuje funkcie `uploadFile, downloadFile` a `remove`. Na nahratie súbora zo zariadenia je potrebné, aby používateľ povolil čítanie súborov zo zariadenia.

**Amazon DynamoDB** – Na dotazy a mutácie s `NoSQL` databázou sa používa objekt `Amplify.API`, kde sa ako argument predá samotný dotaz alebo mutácia vo forme `GraphQL`. Pri dotazovaní sa objekty prekonvertujú na objekty z automaticky vygenerovaných tried pomocou `AWS Amplify SDK`. Tieto objekty je potrebné v aplikácií prekonvertovať na objekty z tried, ktoré sa používajú v zdrojovom kóde aplikácie.

**AWS Lambda** – Na komunikáciu s touto `REST API` službou sa taktiež používa objekt `Amplify.API`. Argumentom je však jeden z `ednpointov`, ktorých implementácia bola opísaná v sekcií 4.1.5. Odpoveď je vo formáte `JSON`, ktorý je pri získavaní zoznamu neschválených účtov nutné prekonvertovať na objekty pomocou knižnice `Gson`.

### 4.2.2 MVVM architektúra

Pre lepšiu čitateľnosť a testovateľnosť zdrojového kódu, je dôležité dodržiavať zásadu `separation of concerns`. V tejto zásade sa hovorí o rozdelení častí aplikácie do vrstiev podľa ich typu alebo funkčnosti.

Pre vývoj mobilných aplikácií na platforme Android, sa na toto rozdelenie používajú architektonické vzory, medzi ktoré patri aj `MVVM`. Tento vzor rozdeľuje logiku spojenú s `UI` do nasledujúcich troch vrstiev:

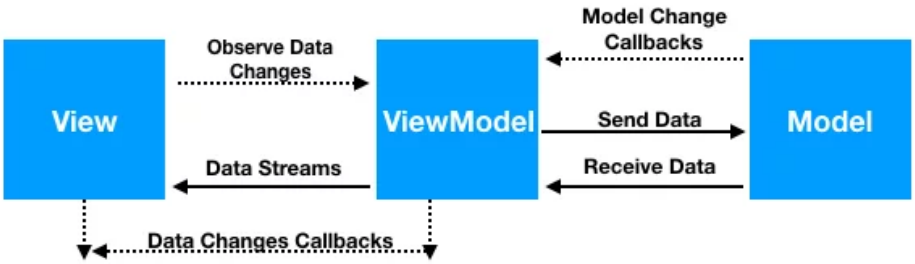
**Model** – Táto vrstva má na starosti správu dát. Jej úlohou je komunikovať s lokálnou alebo vzdialenou databázou a získane dáta vystavovať vrstve `ViewModel` pomocou `Livedata` alebo `Flow`.

**View** – Úlohou tejto vrstvy je správa `UI`. V Android aplikácií sa pod touto vrstvou rozumie fragment alebo `activity`. Sleduje vybrané dáta vo vrstve `ViewModel` a reaguje na ich zmeny prekreslením.

**ViewModel** – Jedná sa o prostredník medzi vrstvami `Model` a `View`. Úlohou tejto vsrtvy je zbierať dáta z vrstvy `Model` a vystavovať ich vrstve `View` pomocou `Livedata` alebo `Flow`.

[47]

Pre lepšie pochopenie je komunikácia medzi jednotlivými vrstvami architektonického vzoru `MVVM` zobrazená na obrázku 4.1.

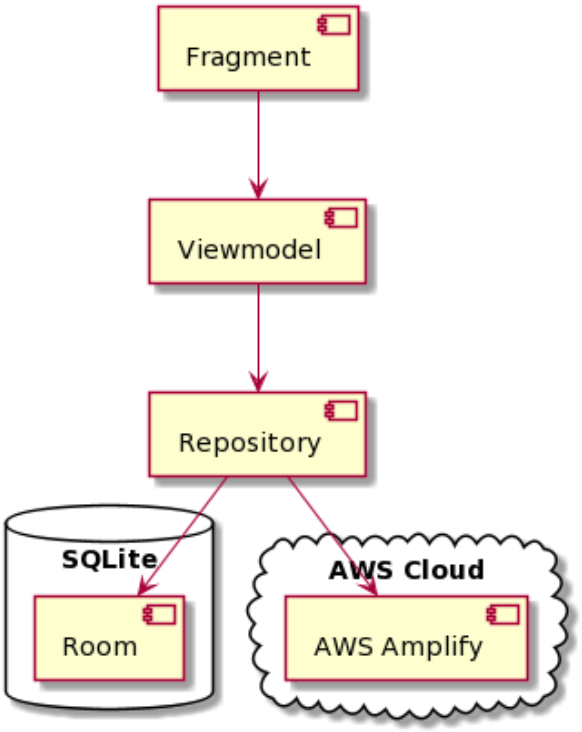



Obr. 4.1: MVVM architektúra [47]

Tento architektonický vzor sa odporúča na vývoj mobilných aplikácií na platforme Android.

Trieda `ViewModel` je súčasťou `Android Jetpack` a je schopná reagovať na životný cyklus aplikácie. Vďaka nej prežijú jej dáta znovu-vytváranie `UI` komponentov (napríklad pri rotácií zariadenia). Umožňuje aj použitie knižnice `Data Binding`, ktorá výrazne zjednoduší komunikáciu medzi `UI` a `ViewModel`. [21]

Použitie tohto architektonického vzoru v aplikácií je zobrazené na obrázku 4.2, kde `Fragment` reprezentuje vrstvu `View` a `Repository` vrstvu `Model`.




Obr. 4.2: MVVM v mobilnej aplikácií

### 4.2.3 Použité nástroje a knižnice

Na vývoj mobilnej aplikácie Android je použitý jazyk `Kotlin` s momentálne najmodernejšími nástrojmi a knižnicami pre vývoj na tejto platforme. V tejto sekcií sú opísané nástroje a knižnice použité pri implementácií niektorých problémov.

#### 4.2.3.1 Asynchrónne programovanie

Asynchrónne programovanie je neoddeliteľnou súčasťou nielen mobilných aplikácií ale aj všetkých moderných systémov. Myšlienkou tohto typu programovania je, aby bolo možné vykonávať väčšie množstvo procesov oddelene od hlavného vlákna na vedľajších vláknach a následne notifikovať vlákno, z ktorého bol tento proces spustený. Spustený proces môže notifikovať toto vlákno buď počas samotného vykonávania, alebo na konci pri úspešnom alebo neúspešnom ukončení.

Pri mobilnom vývoji na platforme Android sa najnovšie na implementáciu asynchrónneho programovania používajú nástroje `Kotlin Coroutines` a `Kotlin Flow`. Výhodou týchto nástrojov je ich plná podpora od `Google`, a z funkčného hľadiska ich reakcia na zmeny životného cyklu vybraných komponentov aplikácie. Obidva nástroje je pre podporu reakcie na zmenu životného cyklu možné spustiť pomocou jedného z týchto dvoch prostredí:

**ViewModelScope** – Toto prostredie je súčasťou objektu `ViewModel`. Spustené `Coroutines` v tomto prostredí sú automaticky zrušené, ak sa objekt `ViewModel` vyčistí.

**LifecycleScope** – Súčasťou každého objektu, ktorý má svoj životný cyklus je prostredie `LifecycleScope`. Medzi tieto objekty patrí napríklad `activity` alebo fragment.

`Kotlin Coroutines` sa používaju pri jednoduchých asynchrónnych procesoch v aplikácií.

Na asynchrónne prenášanie dát sa využíva nástroj `Kotlin Flow`. Jeho výhodou je, že narozdiel od `Kotlin Coroutines` vie vracať väčšie množstvo dát, nazývané aj `stream of data`. Ďalšou výhodou je možnosť využiť funkcie známe z funkcionálneho programovania, ako `map, filter, reduce, forEach` a iné. [22] [23]

#### 4.2.3.2 Dependency Injection

Táto technika je súčasťou dobrého dizajnu aplikácie. Myšlienkou tejto techniky je, pri vytváraní nejakého objektu presunúť jeho závislosť od iných objektov mimo miesto, kde sa práve vytvára. Pomocou toho sa výrazne zefektívni písanie aplikácie výhodami ako zmenšenie veľkosti zdrojového kódu, testovateľnosť a zníženie možných duplikácií.

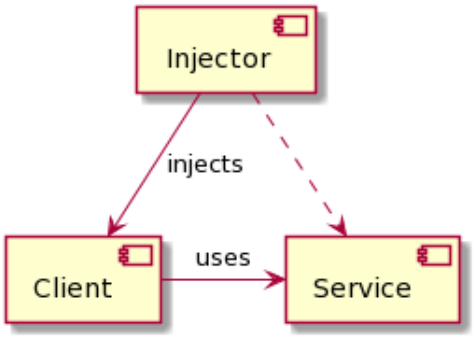
Pri implementácií techniky `Dependency Injection` je potrebné rozlišovať nasledujúce tri typy tried:

**Klient** – Táto trieda je závislá od triedy služba.

**Služba** – Trieda, ktorá poskytuje službu triede klient a je nevyhnutnou súčasťou jej vzniku.

**Injektor** – Pomocou tejto triedy sa chytro vloží závislosť vyššie spomínaných tried.

Role týchto tried a ich vzájomné prepojenie je vidieť na obrázku 4.3. [48]




Obr. 4.3: Dependency Injection

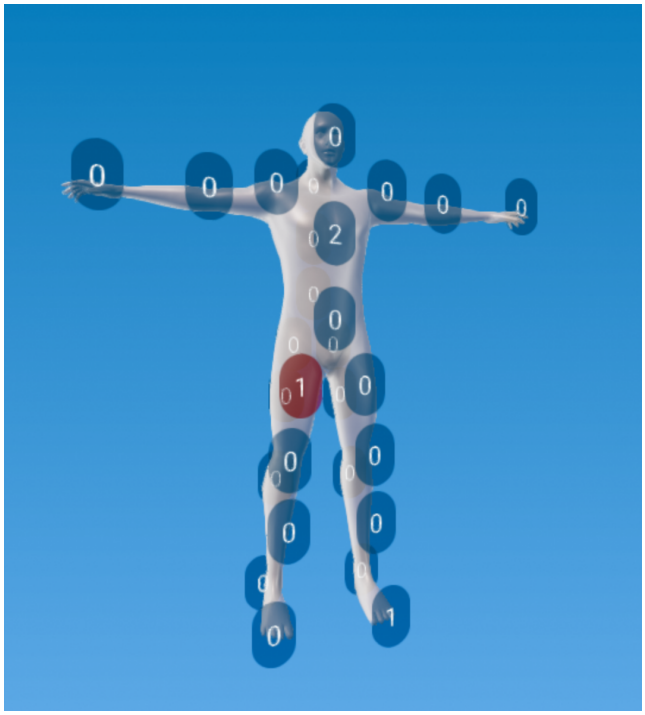
Na využitie tejto techniky pri vývoji mobilnej aplikácie na platforme Android, sa dá použiť oficiálna knižnica `Hilt`. Veľkou súčasťou tejto knižnice je používanie anotácií. Práve táto knižnica obsahuje množstvo anotácií, ktoré sú mierené na triedy a komponenty natívneho vývoja Android aplikácií, medzi ktoré patria `Application`, `ViewModel`, `Activity`, `Fragment`, `View`, `Service` a `BroadcastReceiver`. [27]

#### 4.2.3.3 3D model ľudského tela

Jedna z primárnych funkčných požiadaviek na navrhovanú aplikáciu je zobrazenie zranení pomocou 3D modelu ľudského tela. Našťastie existuje pre vývoj na mobilnej platforme Android knižnica `Sceneform` vyvíjaná spoločnosťou `Google`. Táto knižnica je primárne mierená na vývoj aplikácií s `Augmented Reality`, ale taktiež umožňuje vytvoriť scénu a v nej načítať modely vo formáte `gltf` alebo `glb`. [31]

V aplikácii je použitý 3D model ľudského tela vo formáte `glb`, čo predstavuje binárnu verziu súboru, zatiaľ čo formát `gltf` predstavuje `JSON` verziu. Podľa odporučení je `glb` efektívnejší na načítanie. Výhodou `gltf` formátu je jednoduchšie modifikovanie súboru.

Pre rozdelenie častí tela je implementovaných 28 bodov, ktoré reprezentujú určité časti tela. Ak sa v danej časti nachádza aktívne zranenie (nemá dátum ukončenia), tak je daný bod vyznačený červenou farbou, v opačnom prípade je bod vyznačený modrou farbou. Každý bod má číslo s počtom všetkých zranení (aktívnych aj neaktívnych) v danej oblasti. Na obrázku 4.4 je znázornený model s bodmi, ktorý je implementovaný v mobilnej aplikácií Android.



Obr. 4.4: 3D model ľudského tela

Tento 3D model sa ovláda pomocou jednoduchých gest. Model sa dá otáčať okolo vertikálnej osi jednoduchým ťahaním a priblížiť známym pohybom dvoma prstami.

### 4.2.4 Správa verzií mobilnej aplikácie Android

Na správu verzií bol použitý nástroj `Git`. Jedná sa o distribuovaný systém na správu verziu. V centralizovanom systéme sa kód pri zmenách uzamkne. Pri distribuovanom systéme sa každému vývojárovi repozitár naklonuje a tým pádom má každý k dispozícií vlastnú kópiu. Pri distribuovanom systéme tak môže dôjsť k problému, kedy dvaja vývojári zmenia rovnaký riadok zdrojového kódu. Tento problém sa nazýva `merge conflict` a na jeho odstránenie je potrebné, aby vývojár manuálne vybral správnu časť kódu.

Pri vývoji sa používajú dve hlavné vetvy. Jedna z nich je vetva `main`, ktorá reprezentuje aktuálne verziu v produkcií. Druhá je vetva `dev`, kde sa robia najnovšie zmeny a nové funkcie aplikácie. Ďalej môže mať projekt vedľajšie vetvy, ktoré sa používajú pre vývoj nejakej zložitejšej funkcie, ktorej vývoj môže trvať dlhšiu dobu. Takáto vetva je jasne pomenovaná podľa názvu novej funkcie, pre ktorú je táto vetva určená.

Pri vývoji sa ďalej využíva `Github Issue Tracker`, kde sa vytvárajú nájdené problémy. Pre vytvorenie problému je potrebné tento problém správne pomenovať. Okrem toho je vhodné, aby obsahoval jasný a stručný opis daného problému. V niektorých prípadoch je nevyhnutné, aby obsahoval časti kódu, obrázky, odkazy alebo iné prílohy. Po vytvorení problému v `Github Issue Tracker` sa tomuto problému automaticky priradí číslo, ktoré reprezentuje chronologické poradie problémov (prvý vytvorený problém ma číslo 1, druhý má číslo 2 atď.). Následne sa pri vytváraní nového `commitu` môže do jeho komentára napísať číslo daného problému predchádzajúcim mriežkou. Napríklad pre problém, ktorý je číslovaný číslom 7 by komentár obsahoval `#7`.

Kapitola 5

# Testovanie

Táto kapitola sa zaoberá opisom testovacej časti aplikácie. Testovanie prevažne prebiehalo používateľským testovaním a zachytávaním pádov aplikácie pomocou `Google Firebase Crashlytics`.

## 5.1 Testovanie počas vývoja

Počas vývoja aplikácie je náročné vyvíjať a popritom testovať aplikáciu. Tým, že sa jedná o pomerne rozsiahlu aplikáciu, tak nie je úplne jednoduché zachytiť všetky možné vzniknuté chyby. Používaním rôznych programovacích vzorov, natívnych nástrojov a knižníc sa vznik chýb výrazne znižuje. Pri vývoji na mobilnej platforme Android boli pri testovaní aplikácie použité tieto nástroje:

**Logcat** – Tento nástroj, ktorý je súčasťou natívneho vývojového prostredia `Android Studio`, slúži na vypisovanie logov pri debugovaní aplikácie. Pomocou knižnice `Timber` a tohto nástroja bolo možné eliminovať množstvo vzniknutých chýb.

**Debugger** – Nástroj, ktorý nesmie chýbať v žiadnom modernom vývojovom prostredí. Slúži na debugovanie aplikácie, kedy je možné pozastaviť aplikáciu vo vybranom riadku v zdrojovom kóde, pomocou bodu nazývaného `breakpoint`.

**Profiler** – Jedná sa o pomerne nový nástroj vývojového prostredia `Android Studio`. Pomocou neho je možné sledovať využívanie pamäte a `CPU` zariadenia, na ktorom beží aplikácia. Taktiež je možné sledovať internetovú komunikáciu a využívanie batérie. Pomocou tohto nástroja sa dajú odchytiť chyby, ako neefektívne počítanie a časté sťahovanie dát.

Ako testovacie zariadenie sa dá použiť reálne Android zariadenie, ktorému je potrebné nastaviť určité povolenia v nastaveniach zariadenia. Taktiež je možne využiť natívny emulátor priamo z vývojového prostredia `Android Studio`, ktorý poskytuje veľký výber zariadení rôzneho typu a veľkostí obrazovky. Taktiež je možné nainštalovať ľubovolnú verziu Android, čo môže výrazne ovplyvniť proces testovania, kedy veľa chýb môže vznikať práve na jednej verzií Android. Väčšinou sa jedná o tie staršie verzie Android, v niektorých prípadoch tie najnovšie.

### 5.1.1 Unit testy

Na písanie `unit` testov pri vývoji mobilných aplikácií na platforme Android, je možné použiť knižnicu `JUnit`, ktorá automaticky je súčasťou každého vytvoreného projektu vo vývojovom prostredí `Android Studio`.

Súčasťou registrácie nového účtu je verifikácia vyplnených údajov. Táto verifikácia prebieha na mobilnej aplikácií Android z dôvodu včasného upozornenia používateľa pomocou chybovej hlášky v `UI` komponente `Text Input Edit Text`. Ukážka `unit` testov pre verifikáciu vyplneného mena, hesla a kódu na potvrdenie emailu je prezentovaná v zdrojových kódoch 2, 3 a 4.

```
@Test
fun inputCheckName() {
    // name should contain only valid characters
    // (no numbers and special characters)
    assertEquals(true, "Juraj".isValidName())
    assertEquals(false, "Juraj1".isValidName())
    assertEquals(false, "".isValidName())
    assertEquals(false, "-#9$!@#)".isValidName())
    assertEquals(false, "a4214214".isValidName())
    assertEquals(true, "Juraj Filan".isValidName())
}
```

Zdrojový kód 2: Unit test pre verifikáciu mena pri registrácií

```
@Test
fun inputCheckPassword() {
    // password should have at least 8 characters
    // with at least 1 number
    assertEquals(false, "password".isValidPassword())
    assertEquals(false, "aaa".isValidPassword())
    assertEquals(false, "".isValidPassword())
    assertEquals(true, "-#9$!@#)".isValidPassword())
    assertEquals(true, "67618942142".isValidPassword())
    assertEquals(false, "1111111".isValidPassword())
    assertEquals(true, "password1".isValidPassword())
    assertEquals(false, "676189a".isValidPassword())
}
```

Zdrojový kód 3: Unit test pre verifikáciu hesla pri registrácií

```
@Test
fun inputCheckVerificationCode() {
    // email verification code should contain 6 digit numbers
    assertEquals(false, "abcdef".isValidCode())
    assertEquals(false, "673".isValidCode())
    assertEquals(false, "676a89".isValidCode())
    assertEquals(false, "676-89".isValidCode())
    assertEquals(true, "676189".isValidCode())
    assertEquals(false, "6761891".isValidCode())
    assertEquals(false, "676189a".isValidCode())
    assertEquals(false, "676189a".isValidCode())
    assertEquals(false, "".isValidCode())
}
```

Zdrojový kód 4: Unit test pre verifikáciu kódu na potvrdenie emailu

## 5.2 Nasadenie verzie na testovanie

Hlavnou časťou testovania je testovanie aplikácie na vybraných hráčoch konkrétneho športového klubu. Aby používateľ nemusel sťahovať a inštalovať aplikáciu manuálne priamo z `APK`, bolo potrebné aplikáciu nasadiť na `Google Play Store`. Ďalšou výhodou nasadenia je jednoduchá aktualizácia pri nových verziách aplikácie.

### 5.2.1 Google Play Store

Na nasadenie aplikácie do `Google Play Store` sa používa webová aplikácia `Google Play Console`. Na pridávanie aplikácií je potrebné mať zaregistrovaný vývojársky účet a zaplatiť určitý poplatok. Pred samotným nasadením aplikácie je potrebné vytvoriť profil aplikácie. Po vytvorení profilu je potrebné dokončiť množstvo krokov, medzi ktoré patrí napríklad vyplňovanie rôznych informácií o aplikácií, pridávanie ukážok z aplikácie vo forme obrázkov, typ cielenej klientely, štáty, v ktorých sa bude aplikácia používať atď.

Po splnení týchto krokov je potrebné vytvoriť podpísané `APK`. Podpis sa dá jednoducho vytvoriť priamo vo vývojovom prostredí `Android Studio`. Podpis je tvorený súborom vo formáte `jks`, ktorý reprezentuje repozitár tvorený certifikátmi a privátnymi kľúčmi. Pre odomknutie tohto súboru je potrebné zadať údaje, ktoré sa zadávali pri vytváraní podpisu.

Pred vytvorením nového podpísaného `APK` aplikácie, ktorá už je nasadená v `Google Play Console`, je potrebné zmeniť verziu kódu v súbore `build.gradle`, ktorý je v projekte aplikácie.

Po dokončení spomínaných krokov je konečne možné podpísané `APK` nahrať v `Google Play Console` do týchto troch prostredí:

**Alpha** – Toto prostredie je určené na interné testovanie pre `QA` tím. Prístup k nemu majú iba vybraní používatelia, ktorí boli pridaní pomocou ich emailovej adresy. Výhodou tohto prostredia je, že aplikácia nemusí byť po nasadení preskúmaná službou `Google review`.

**Beta** – K tomuto prostrediu majú prístup používatelia, ktorí majú na svojom zariadení v `Google Play Store` nastavené, že chcú mať prístup k `beta` verziám aplikácií. Aplikácia musí byť pred nasadením preskúmaná `Google review`. Po nasadení do produkcie už nie je potrebné ďalšie preskúmanie.

**Produkcia** – Jedná sa o produkčné prostredie. Aplikácia musí byť pred nasadením preskúmaná `Google review`.

V testovacej fáze aplikácie bolo zvolené produkčné prostredie. Dôvodom je zbytočná komplikácia pri nastavovaní prístupu k `beta` verziám aplikácií v `Google Play Store`. Pri využívaní testovacieho prostredia `alpha` je potrebné dopredu vedieť email budúceho používateľa, čo pri plánovanom testovaní práve dostupnými hráčmi iba spomalí inštaláciu aplikácie (viac o testovaní používateľmi v sekcií 5.2.3). Tým, že je pre používanie aplikácie potrebná registrácia a následné schválenie účtu správcom, tak potenciálnych nežiadúcich používateľov môže správca jednoducho vymazať. Už aj samotná registrácia do neznámej aplikácie by mohla takýchto nežiadúcich používateľov odradiť.

### 5.2.2 Google Firebase Crashlytics

Tento nástroj je nevyhnutnou súčasťou nielen testovacej fázy, ale aj fázy, kedy je mobilná aplikácia v produkcii a používajú ju reálni používatelia. Implementácia tohto nástroja spočíva v pár krokoch, ako registrácia a pridanie knižnice do projektu aplikácie. V poslednom kroku je potrebné donútiť aplikáciu padnúť, čo je možne docieliť rôznymi jednoduchými trikmi.

`Google Firebase Crashlytics` slúži na sledovanie pádov aplikácie. Vďaka nemu je možné zachytiť všetky pády aplikácie, ktoré sa mohli stať na zariadeniach používateľov produkčnej verzie. Nevyhnutnou súčasťou záznamu je log, ktorý dôkladne opisuje príčinu pádu a miesto v zdrojovom kóde aplikácie. Okrem toho je dôležitým údajom aj typ zariadenia a verzia Android systému. Ďalej záznam obsahuje informácie ako orientácia aplikácie (`portrait` alebo `landscape`), voľná pamäť disku a `RAM` zariadenia a mnoho iných užitočných informácií, ktoré môžu pomôcť pri riešení daného pádu.

### 5.2.3 Testovanie používateľmi

Aplikácia bola testovaná na 12 používateľoch. Iba jeden z týchto používateľov mal rolu správcu. Jednalo sa o fyzioterapeuta z konkrétneho športového klubu. Ostatní používatelia boli hráči toho istého športového klubu, ktorí mali v tom čase zranenie alebo boli vo fáze rehabilitácie.

Spomínaný fyzioterapeut bol súčasťou tvorby zadania a funkčných požiadavkov aplikácie. Testovanie aplikácie z jeho strany prebiehalo už pri samotnom vývoji aplikácie. Vybraní hráči mali za úlohu splniť niekoľko primárnych úloh:

**Úloha 1** – Stiahnuť mobilnú aplikáciu Android z `Google Play Store` a nainštalovať ju na svojom zariadení.

**Úloha 2** – Zaregistrovať sa. Vyplniť svoje reálne meno a email.

**Úloha 3** – Potvrdiť vyplnený email kódom, ktorý je zaslaný na daný email.

**Úloha 5** – Oznámiť správcovi (fyzioterapeutovi), že sa úspešne zaregistrovali. Následne ich správca mohol priradiť do kategórie a schváliť.

**Úloha 6** – Prihlásiť sa do aplikácie.

**Úloha 7** – Nahrať svoju fotografiu.

**Úloha 8** – Vyplniť potrebné údaje, ako pozícia na ihrisku, preferovaná noha a dátum narodenia.

**Úloha 9** – Nastaviť si čas upozornenia na vyplnenie dát.

**Úloha 10** – Vyplňovať obidve `RPE` hodnoty na dennej báze.

Medzi úlohy správcu patria primárne správa zranení, pridávanie príloh a sledovanie vyplnených `RPE` hodnôt všetkých hráčov. Výsledky z testovania používateľmi je opísaná v sekcií 5.3.

## 5.3 Vyhodnotenie testovacej fázy

Táto sekcia opisuje vyhodnotenie zozbieraných informácií a pripomienok z testovania používateľmi, ktoré bolo opísané v sekcií 5.2.3. Hráči nemali žiadne veľké problémy s uvedenými úlohami.

Vyhodnotenie testovacej fázy je rozdelené do dvoch častí. Prvá časť opisuje nájdené chyby a riešenia pri ich odstránení. V druhej sú analyzované pripomienky k možnému rozšíreniu aplikácie. Na zaznamenávanie týchto informácií a pripomienok bol použitý `Github Issue Tracker`.

### 5.3.1 Nájdené chyby v aplikácií

V testovacej fáze používateľmi sa bohužiaľ našli aj chyby. Väčšinou sa jednalo o chyby nejakej funkčnosti aplikácie, ktoré boli nahlásené používateľmi. Našli sa aj chyby, ktoré spôsobovali pád aplikácie, a pomocou `Google Firebase Crashlytics` sa ich podarilo zachytiť a následne tieto chyby replikovať.

V testovacej fáze boli používateľmi ohlásené tieto chyby:

**Správca môže schváliť účet s nepotvrdeným emailom** – Táto chyba vznikla na serverovej časti aplikácie v `AWS Lambda`. Na odstránenie bolo potrebné správne odfiltrovať zoznam účtov. Teda pridať podmienku pre nepotvrdený email.

**Na Android verzií 7.0 sa nezobrazujú profilové obrázky hráčov** – Pri implementácií profilových obrázkov hráčov sa používa `UI` komponent `Card View`, ktorý sa použil na orezanie obrázka do tvaru kruhu. Tento komponent nefungoval správne na najstaršej podporovanej verzií aplikácie a bolo potrebné spraviť určité zmeny.

**Nedá sa zmeniť typ už vytvoreného zranenia** – Chyba sa odstránila úpravou určitej časti zdrojového kódu v aplikácií.

**Po reštartovaní zariadenia prestane fungovať upozornenie** – Pre odstránenie tohto problému bolo potrebné implementovať `receiver`, ktorý reaguje na `boot` zariadenia. Ten môže následne nastaviť upozornenie na vybraný čas.

Pomocou nástroja `Google Firebase Crashlytics` sa podarilo odstrániť tieto chyby:

**Pád aplikácie v zraneniach č. 1** – Aplikácia padla, ak používateľ klikol na nejaký bod ľudského tela, a potom rýchlo klikol na iný bod pred tým, ako sa otvoril zoznam zranení v oblasti prvého kliknutého bodu. Túto chybu zachytil `Google Firebase Crashlytics` a vďaka tomu bolo jednoduché ju replikovať. Odstránenie chyby spočívalo k uzamknutiu klikania na body, kým sa otvára zoznam zranení.

**Pád aplikácie v zraneniach č. 2** – V niektorých prípadoch sa zle načítavali body ľudského tela a končilo to pádom aplikácie. Táto chyba bola zaznamenaná `Google Firebase Crashlytics` a výrazne pomohla pri odstránení tejto chyby. Pri odstránení bolo potrebné všetky objekty z 3D scény presunúť z fragmentu do `ViewModel`, aby si uchovávali svoj stav pri zmenách životného cyklu aplikácie.

### 5.3.2 Možnosti rozšírenia

Častým používaním aplikácie väčším množstvom používateľov, vzniká množstvo nových nápadov, ako by sa mohla aplikácia rozšíriť. V testovacej fáze aplikácie sa podarilo zozbierať tieto pripomienky na zlepšenie funkčnosti aplikácie:

**Pridať druhé upozornenie** – V aplikácií sa momentálne nachádza jedno upozornenie. Hráčom by sa páčilo, ak by mali možnosť nastaviť si aj druhé upozornenie. Tým pádom by bolo potrebné tieto dve upozornenia rozdeliť a pomenovať podľa vybraného typu `RPE`.

**Filter v zozname hráčov** – Správcovi by sa hodilo mať v zozname hráčov možnosť vyhľadávania a filtrovania podľa aktívne zranených hráčov. Aktívne zranených hráčov by bolo taktiež dobré v zozname hráčov vhodne označiť.

**Filter v zozname zranení** – V zozname zranení by sa hodilo mať možnosť filtrovania zranení podľa ich typu.

**Verejné a súkromné prílohy** – Niektoré prílohy hráča by bolo potrebné nastaviť tak, aby k nim mal prístup iba správca a nie samotný hráč. Taktiež by bolo vhodné, aby sa v aplikácií dala premenovať príloha. Táto funkcia momentálne nie je implementovaná.

**Sledovanie internetového pripojenia** – V aplikácií by sa hodilo mať upozornenie pri strate internetového pripojenia. Túto funkciu má väčšina kvalitných a moderných aplikácií.

**Načítanie nameraných telesných hodnôt z chytrej váhy** – Momentálne je potrebné vyplniť tieto hodnoty manuálne. Chytré váhy však obsahujú `SDK`, pomocou ktorého je možné túto funkciu zautomatizovať. Zariadenie bude potrebné s danou váhou spárovať pomocou technológie `Bluetooth` a následne je možné plne využívať toto `SDK` na získanie potrebných údajov z chytrej váhy hneď po odvážení a nameraní.

**Zaznamenávanie nových údajov** – Aplikácia je vďaka svojmu návrhu pripravená na rozšírenie, medzi ktoré patrí aj pridávanie funkcionality na sledovanie a zaznamenávanie nových údajov. Jedným z takýchto rozšírení môže byť sledovanie rôznych hodnoť pri fyzickej záťaži pomocou chytrého hrudného pásu. Pomocou neho sa dajú sledovať hodnoty srdcového tepu, prebehnutej vzdialenosti, rýchlosti atď. Namerané hodnoty je možné automaticky získať pomocou `SDK` a technológie `Bluetooth`.

# Záver

Cieľom tejto práce bolo navrhnúť a implementovať mobilnú aplikáciu na platforme Android, ktorej úlohou je zjednodušiť prácu trénerom, fyzioterapeutom, lekárskemu personálu a iným členom vedenia športových klubov pomocou zhromažďovania potrebných dát a súborov na jednom mieste. V aplikácií sú dva typy rolí, hráč a správca, pri ktorých sa výrazne líši dostupnosť funkcií v aplikácií. Hráč môže pomocou aplikácie posielať svoje pocity po fyzickej záťaži alebo spánku a pridávať svoje aktuálne telesné merania.

Ďalším cieľom bolo analyzovať poskytovateľov služieb `PaaS` a potom vybrať službu, na ktorej bude navrhnutá a implementovaná serverová časť aplikácie. Na základe tejto analýzy bola vybraná `MBaaS` služba `AWS Amplify`. Pre získanie potrebných informácií v testovacej fáze, boli vybrané nástroje `Analytics` a `Crashlytics` služby. `Google Firebase`

Hlavnou časťou testovania bolo testovanie používateľmi, medzi ktorých patrilo vedenie a hráči konkrétneho športového klubu. Pomocou tejto testovacej fázy sa podarilo odstrániť niekoľko chýb v aplikácií. Ďalšou výhodou tohto testovania a častým používaním aplikácie bol vznik nových nápadov na rozšírenie aplikácie.

V budúcnosti bude cieľom zautomatizovať vypĺňanie nameraných telesných hodnôt pomocou technológie `Bluetooth` a `SDK` chytrej váhy. Okrem toho sa plánujú pridávať funkcionality na sledovanie a zaznamenávanie nových údajov. Jedným z nich je sledovanie rôznych hodnôt pri fyzickej záťaži pomocou hrudného pása, ktorého dáta je taktiež možné získať pomocou technológie `Bluetooth` a `SDK`. Okrem toho je v aplikácií miesto na vylepšenie používateľského rozhrania.

# Literatúra


[1] Stephen Watts, M. R.: SaaS vs PaaS vs IaaS: What's The Difference & How To Choose. *In: bmc*, 2019, [online] [vid. 2021-04-12]. Dostupné z: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/

[2] Introduction to Backend as a Service (BaaS). *In: Alibaba Cloud*, 2017, [online] [vid. 2021-04-12]. Dostupné z: https://www.alibabacloud.com/blog/introduction-to-backend-as-a-service-baas_213665

[3] Devathon: Firebase vs MongoDB Stitch vs AWS Amplify vs Azure Mobile Apps: MBaaS Comparison. *In: Medium*, 2020, [online] [vid. 2021-04-18]. Dostupné z: https://medium.com/@devathon_/firebase-vs-mongodb-stitch-vs-aws-amplify-vs-azure-mobile-apps-mbaas-comparison-47e8d6d1813d

[4] Create your Azure free account today. *In: Microsoft*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://azure.microsoft.com/en-us/free/

[5] Introduction to MongoDB Realm for Mobile Developers. *In: mongoDB*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://docs.mongodb.com/realm/get-started/introduction-mobile/

[6] Clark, J.: What is Kinvey? *In: Back4App*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://blog.back4app.com/kinvey/

[7] Amazon Cognito. *In: Amazon*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://aws.amazon.com/cognito

[8] Amazon DynamoDB. *In: Amazon*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://aws.amazon.com/dynamodb

[9] Amazon S3. *In: Amazon*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://aws.amazon.com/s3

[10] AWS Lambda. *In: Amazon*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://aws.amazon.com/lambda

[11] Amplify Android (AWS Mobile SDK for Android). *In: Amazon*, 2021, [online] [vid. 2021-04-18]. Dostupné z: https://docs.aws.amazon.com/sdk-for-android

[12] Add Firebase to your Android project. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://firebase.google.com/docs/android/setup

[13] QingNiu: Megafit. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.qingniu.megafit

[14] Google LLC: Google Fit: Health and Activity Tracking. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness

[15] Janos Dupai: Nethlete. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-21]. Dostupné z: https://play.google.com/store/apps/details?id=com.matchmeeting.matchmeetingapp

[16] Android overview. *In: Jetbrains*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://kotlinlang.org/docs/android-overview.html

[17] Kotlin on Android. Now official. *In: Jetbrains*, 2017, [online] [vid. 2021-04-20]. Dostupné z: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/

[18] Getting started with Android Jetpack. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://developer.android.com/jetpack/getting-started

[19] Introduction to Activities. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities

[20] Fragments. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://developer.android.com/guide/fragments

[21] ViewModel Overview. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

[22] Kotlin coroutines on Android. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: https://developer.android.com/kotlin/coroutines

[23] Kotlin flows on Android. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/kotlin/flow`

[24] LiveData Overview. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/livedata`

[25] Data Binding Library. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/topic/libraries/data-binding`

[26] Save data in a local database using Room. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/training/data-storage/room`

[27] Dependency injection with Hilt. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/training/dependency-injection/hilt-android`

[28] DataStore. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/datastore`

[29] Navigation. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/guide/navigation`

[30] bumptech: Glide. *In: GitHub, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://github.com/bumptech/glide`

[31] google ar: Sceneform SDK for Android. *In: GitHub, Inc.*, 2020, [online] [vid. 2021-04-20]. Dostupné z: `https://github.com/google-ar/sceneform-android-sdk`

[32] PhilJay: MPAndroidChart. *In: GitHub, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://github.com/PhilJay/MPAndroidChart`

[33] JakeWharton: Timber. *In: GitHub, Inc.*, 2020, [online] [vid. 2021-04-20]. Dostupné z: `https://github.com/JakeWharton/timber`

[34] Distribution dashboard. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/about/dashboards`

[35] Meet Google Play's target API level requirement. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/distribute/best-practices/develop/target-sdk`

[36] Android Studio. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-20]. Dostupné z: `https://developer.android.com/studio`

[37] Abhishek Jain: 10 Heuristic Principles – Jakob Nielsen's (Usability Heuristics). *In: UXness*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://www.uxness.in/2015/02/10-heuristic-principles-jakob-nielsens.html

[38] Bottom navigation. *In: Material.io*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://material.io/components/bottom-navigation/android

[39] Android AlertDialog with Examples. *In: tutlane*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://www.tutlane.com/tutorial/android/android-alertdialog-with-examples

[40] Spinners. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://developer.android.com/guide/topics/ui/controls/spinner

[41] Create dynamic lists with RecyclerView. *In: Google, Inc.*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/recyclerview

[42] How to Show a Toast for a Specific Duration in Android. *In: Code Project*, 2021, [online] [vid. 2021-04-28]. Dostupné z: https://www.codeproject.com/Articles/988256/How-to-Show-a-Toast-for-a-Specific-Duration-in-And

[43] Prerequisites, Android. *In: Amazon*, 2021, [online] [vid. 2021-04-29]. Dostupné z: https://docs.amplify.aws/start/getting-started/installation/q/integration/android

[44] AUTHENTICATION, Getting Started, Android. *In: Amazon*, 2021, [online] [vid. 2021-04-29]. Dostupné z: https://docs.amplify.aws/lib/auth/getting-started/q/platform/android

[45] STORAGE, Getting Started, Android. *In: Amazon*, 2021, [online] [vid. 2021-04-29]. Dostupné z: https://docs.amplify.aws/lib/storage/getting-started/q/platform/android

[46] API (GRAPHQL), Getting Started, Android. *In: Amazon*, 2021, [online] [vid. 2021-04-29]. Dostupné z: https://docs.amplify.aws/lib/graphqlapi/getting-started/q/platform/android

[47] Android MVVM Design Pattern. *In: JournalDev*, 2021, [online] [vid. 2021-04-30]. Dostupné z: https://www.journaldev.com/20292/android-mvvm-design-pattern

[48] Dependency Injection. *In: Tutorials Teacher*, 2021, [online] [vid. 2021-04-30]. Dostupné z: `https://www.tutorialsteacher.com/ioc/dependency-injection`

Dodatok A

# Zoznam použitých skratiek

**AWS** Amazon Web Services

**RPE** Rating of perceived exertion

**PaaS** Platform as a Service

**MBaaS** Mobile Backend as a Service

**CRUD** Create, Read, Update, Delete

**UI** User Interface

**API** Application Programming Interface

**REST** Representational State Transfer

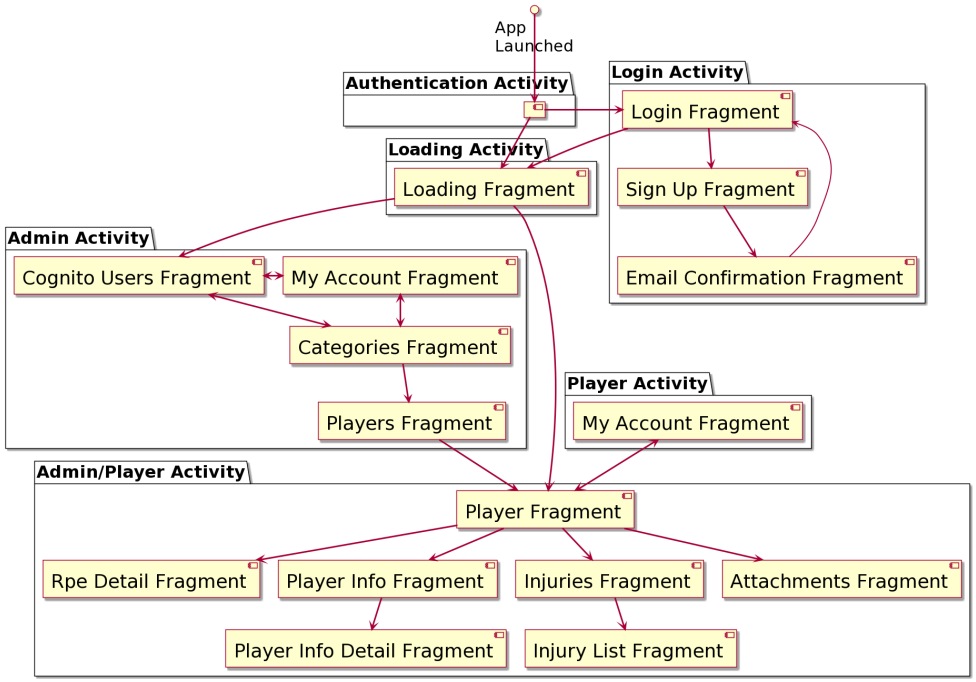
**SDK** Software Development Kit

**MVVM** Model View ViewModel

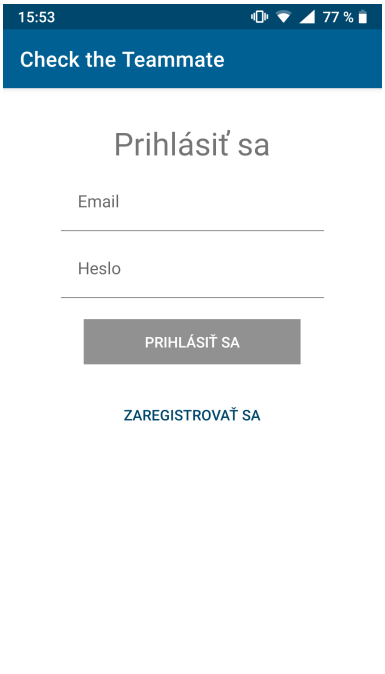
**APK** Android application package

Dodatok B

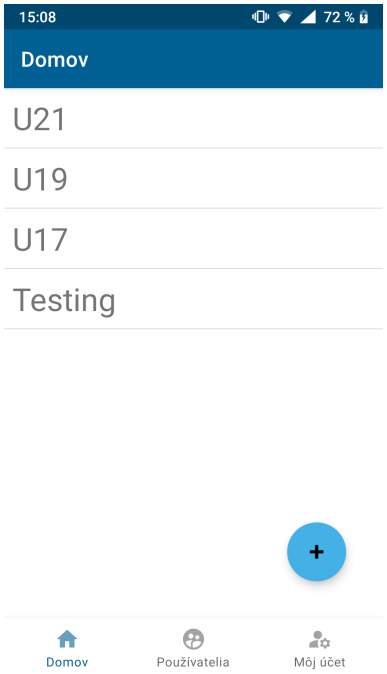
# Ukážky aplikácie



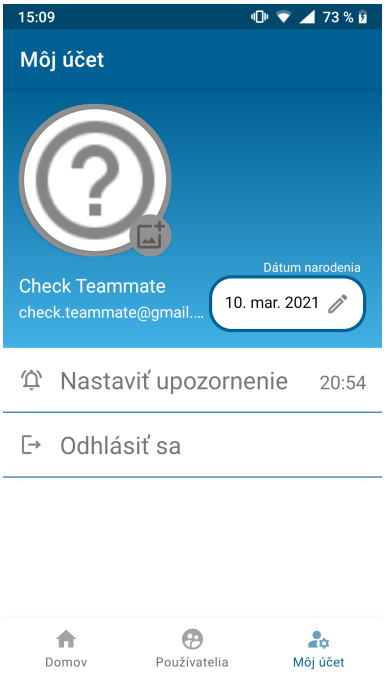

Rozdelenie fragmentov mobilnej aplikácie Android



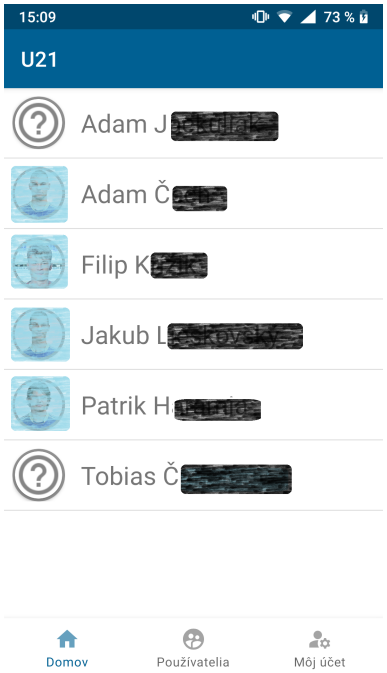

Login Fragment




Categories Fragment

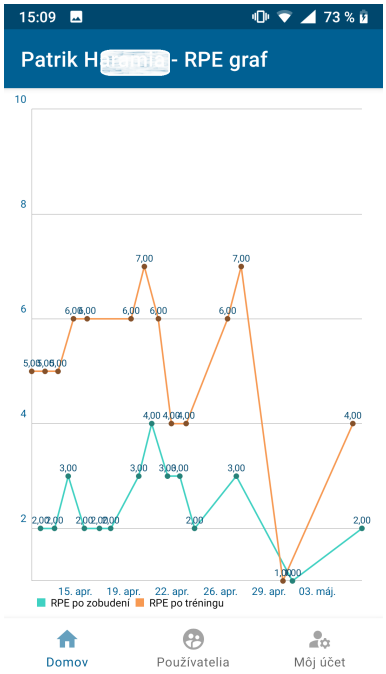



My Account Fragment




Players Fragment

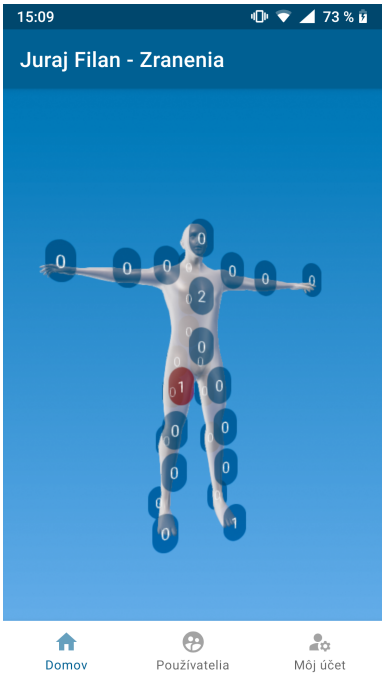

Player Fragment



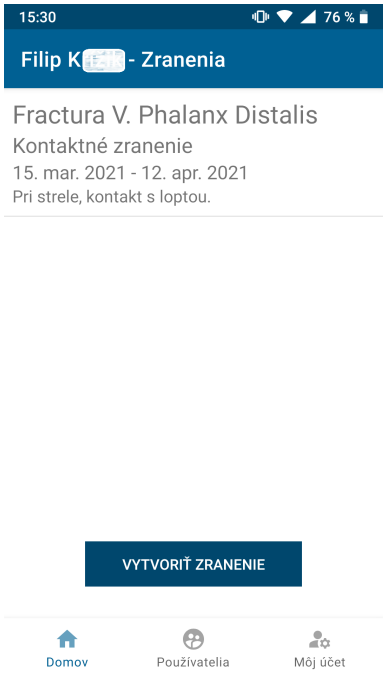

Rpe Detail Fragment


Player Info Fragment

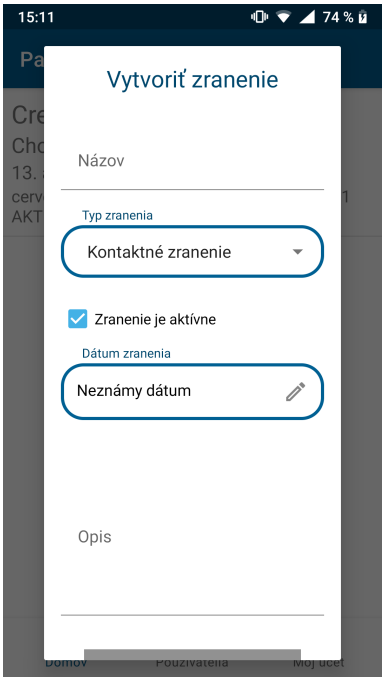

Player Info Detail Fragment




Injuries Fragment




Injury List Fragment




Update Injury Dialog


Attachments Fragment

Dodatok C

# Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `build.txt` ..... odkaz na Google Play Store
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF