VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO HODNOCENÍ A POROVNÁNÍ VIDEA TRÉNINKU SPORTOVCE

MOBILE APPLICATION FOR EVALUATION AND COMPARING AN ATHLETE'S TRAINING VIDEOS

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE **RASTISLAV DURÁNIK**
AUTHOR

VEDÚCI PRÁCE **Ing. ALENA TESAŘOVÁ**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

139641

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Duránik Rastislav**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Mobilní aplikace pro hodnocení a porovnání videa tréninku sportovce**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou návrhu a vývoje mobilních aplikací.
2. Prostudujte existující aplikace řešící podobný problém a analyzujte výhody a nevýhody.
3. Iterativně navrhujte a testujte dílčí prvky uživatelského rozhraní mobilní aplikace pro hodnocení videa tréninku sportovce trenérem, která bude umožňovat porovnání videí cviků v historii.
4. Implementujte navržené prvky do aplikace.
5. Testujte vytvořené řešení s uživateli a iterativně je vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakát a krátké video pro prezentování projektu.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- React Native: https://reactnative.dev/

Při obhajobě semestrální části projektu je požadováno:
body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Tesařová Alena, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

V tejto práci je rozoberaná tvorba mobilných aplikácií a návrh a implementácia vlastnej aplikácie pre porovnávanie a zdieľanie cvikov s trénerom. Cieľom tejto práce je zoznámiť sa s problematikou vytvárania mobilných aplikácií a vytvorenie vlastnej mobilnej aplikácie. Zameriaval som sa na operačný systém Android. Pre návrh užívateľského rozhrania bol použitý nástroj Figma. Samotná aplikácia bola vytvorená knižnicou React Native za využitia platformy Expo. Aplikáciu sa mi podarilo vytvoriť a otestovať so všetkými požadovanými vlastnosťami.

Abstract

This work discusses development of mobile phone applications and design and implementation of a custom application for comparing and sharing training exercises with a trainer/coach. My focus was on the Android operating system. The Figma tool was used for the design of the user interface. The application itself was created with the React Native library using the Expo platform. I managed to create and test an application with all of the required features.

Kľúčové slová

mobilné aplikácie, fitness, React Native, Android, Firebase, porovnanie videa, užívateľské rozhranie

Keywords

mobile application, fitness, React Native, Android, Firebase, video comparison, user interface

Citácia

DURÁNIK, Rastislav. *Mobilní aplikace pro hodnocení a porovnání videa tréninku sportovce.* Brno, 2023. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedúci práce Ing. Alena Tesařová

Mobilní aplikace pro hodnocení a porovnání videa tréninku sportovce

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pani Ing. Aleny Tesařovej. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Rastislav Duránik
10. mája 2023

Poďakovanie

Rád by som sa poďakoval mojej vedúcej práce pani Ing. Alene Tesařovej za odbornú pomoc pri tvorbe tejto bakalárskej práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Pohyb je pre telo nielen prirodzený, ale v určitej miere aj nevyhnutne potrebný. 'Športom k zdraviu' je pre množstvo ľudí z mnohých strán omieľaná fráza, bez pochýb postavená na pravdivom základe. Bohužiaľ sa už menej hovorí o spôsobe a správnosti spomínaného pohybu. Veľakrát môže snaha o zdravý životný štýl napáchať viac škôd, než úžitku. To hrozí najmä pri silovo zameraných športoch ktoré kladú veľkú a intenzívnu zátaž na pohybový aparát. Pri týchto cvičeniach môže dôjsť k zraneniu aj v kratšom časovom úseku tréningového procesu. Ako mnoho športovcov vie, nesprávna technika zdvíhania ťažkých váh môže veľmi rýchlo vyústiť do vážnych problémov až bolestí nielen s chrbtom. Takisto zdanlivo bezpečné a menej intenzívne cvičenia, ako sú cviky vlastnou váhou tela a izometrické cvičenia, môžu dlhodobo nevhodne zaťažovať až chronicky postihovať pohybový aparát. Dá sa teda povedať, že vykonávanie pohybovej činnosti v nesprávnom prevedení vôbec nemusí byť zdravé.

Športovec môže poznať celý teoretický priebeh cviku, ale aj napriek tomu mu jeho nevedomé chybné vykonávanie môže spôsobiť vážne zdravotné komplikácie.

Zbytočným zraneniam sa dá predísť nadobudnutím správnej techniky cviku. K tomuto účelu môže byť mobilná aplikácia pre porovnávanie cvikov veľmi užitočná. Okrem porovnávania svojich cvikov so svojimi predošlými videami alebo nahrávkami profesionálov by takáto aplikácia umožňovala aj odoslanie cviku trénerovi. Tréner by mal možnosť video ohodnotiť rovnako tak ako vyznačiť a okomentovať potrebné momenty videa. Takáto funkcia by umožňovala dostávať rady od trénerov, ku ktorým by sa športovec napríklad kvôli veľkej vzdialenosti či finančnej náročnosti osobných tréningov bežne nedostal. Taktiež by poskytla možnosť zlepšovať sa v technike cvikov bez potreby časovej fixácie na voľné miesta v rozvrhu trénera. Zároveň by to umožnilo včas zastaviť a zachytiť chybné tréningové vzorce, ktoré sa po ich naučení často ťažko menia. Týmto spôsobom by sa zabezpečil bezpečný a efektívny spôsob pohybu.

V tejto práci sa bližšie pozrieme na proces vývoja takejto aplikácie.

Kapitola 2

Vývoj mobilných aplikácií

S príchodom mobilných telefónov sa výrazne zmenilo odvetvie informačných technológií. Tento vynález umožnil ľudom prístup k aplikáciam jednoduchšie ako kedykoľvek predtým, čo viedlo k rýchlemu rastu odvetvia vývoja aplikácií. Ako najdominantnejšie operačné systémy sa ukázali Android a iOS, ktoré majú spoločne 99,3 percentný podiel na globálnom trhu. Pred vývojom je teda nutné zvážiť, pre akú platformu bude aplikácia vytvorená. V dnešnej dobe sa však väčšina aplikácií tvorí multi-platforomovo, čo znamená, že je použiteľná na rôznych zariadeniach s rôznym operačným systémom. Voľba platformy je len jeden z množstva aspektov, ktoré je nutné zohľadniť v začiatočných fázach vývoja. Najprv je nutné si položiť otázky, ako napríklad, kto je typickým užívateľom tejto aplikácie, čo je cieľom vytvorenia tejto aplikácie, aké benefity z toho bude mať vývojár, prípadne spoločnosť a aké technológie bude vhodné použiť. Medzi ďalšie významné kroky patrí prieskum existujúcich aplikácií, najmä úspešných, aby sa nevytváralo niečo, čo už existuje, a tiež z dôvodu inšpirácie. Nasleduje iteratívna tvorba návrhu užívateľského rozhrania, ktorá ideálne prebieha za pravidelnej konzultácie s klientom a testovania návrhu na typických užívateľoch. Po doladení nedostatkov sa môže začať s implementáciou a následným testovaním. V prípade úspechu pri testovaní je aplikácia pripravená na distribúciu. V opačnom prípade sa dolaďujú nedostatky a testovanie sa opakuje [4]. Tento postup sa nazýva proces vývoja mobilných aplikácií A.1.

V následujúcich podkapitolách je detailnejšie popísaný natívny a *cross-platformový* vývoj.

2.1 Natívny vývoj aplikácií

Natívny vývoj je jedným z najobľúbenejších spôsobov vytváranie aplikácii. Znamená to, že aplikácia bude určená a prispôsobená pre konkrétnu platformu, čo vo väčšine prípadov znamená buď Android alebo iOS. Aplikácie Android sú písané pomocou jazykov ako Java alebo Kotlin, zatiaľ čo aplikácie pre iOS su tvorené pomocou jazyka Swift a Objective-C. Vývoj si vyžaduje okrem súprav na vývoj softvéru (SDK), ktoré sú špecifické pre operačný systém, aj integrované vývojové prostredie (IDE). V prípade aplikácií pre systém Android je nutné používať Android Studio alebo IntelliJ IDEA. Tieto nástroje sú dostupné v systémoch Windows, macOS alebo Linux. Ak je aplikácia tvorená pre iOS, tak je potrebné použiť Xcode, alebo AppCode. Tieto nástroje sú narozdiel od systémov pre Android zariadenia dostupné iba v systéme macOS [21].

Obr. 2.1: Proces vývoja mobilných aplikácií. Prevzaté z [8].

Výhody

Pri natívnom vývoji majú aplikácie veľmi vysoký výkon, keďže sú optimalizované pre konkrétnu platformu a skompilované s jej základným programovacím jazykom a API. To prináša vyššiu pravdepodobnosť bezchybnosti aplikácie. Okrem rýchlosti, ktorá je pozorovateľná najmä pri väčších a komplexnejších aplikáciach, dokážu natívne aplikácie efektívnejšie reagovať na akcie používateľov. Tieto aplikácie majú taktiež prístup k zabudovaným bezpečnostným funkciám špecifickým pre platformu, čo zlepšuje zabezpečenosť aplikácie. Vzhľad a ovládanie sú konzistentné, pretože dedia rozhrania operačného systému zariadenia. Taktiež dodržiavajú dizajnové návody pre konkrétny operačný systém, vďaka čomu je tok aplikácie prirodzenejší [1]. Vyššie uvedené výhody možno zhrnúť do nasledovných bodov:

- rozsiahla funkcionalita, prístup k funkciám telefónu,
- lepšia podpora na App Store a Google play,
- vysoký výkon a skvelá užívateľská skúsenosť,
- lepšia zabezpečenosť.

Nevýhody

Medzi hlavné nedostatky natívneho vývoja patrí nákladnosť a časová náročnosť. Pri vytváraní takýchto aplikácii môže byť spustenie pre iOS aj Android pomerne nákladné, keďže je potrebné najať samostatné tímy vývojárov pre konkrétne platformy. Tento vývoj je časovo náročný, keďže prácu vykonanú pre jednu platformu nieje možné aplikovať na inú [15].

2.1.1 Android

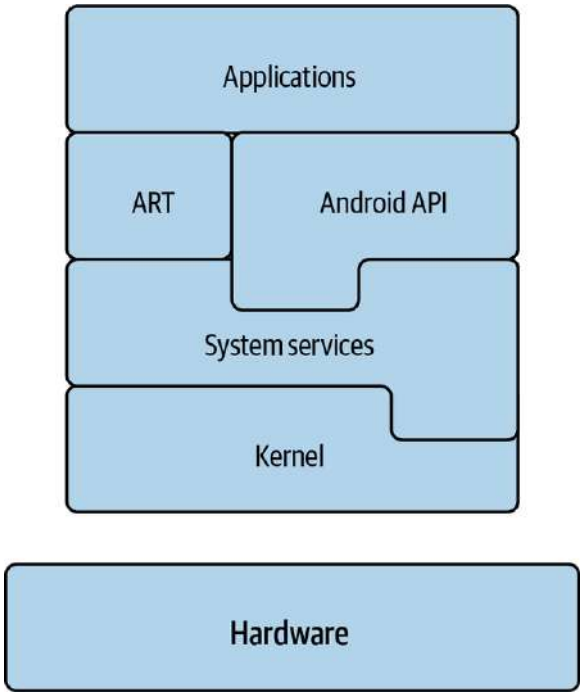
Android je operačný systém, podobne ako Windows či MacOS. Na narozdiel od týchto systémov je Android založený na Linuxovom jadre. Je vyvinutý spoločnosťou Google a silne optimalizovaný pre mobilné zariadenia – najmä pre mobilné zariadenia napájané batériami. Predvoleným jazykom vývoja pre Android je jazyk Java, ktorý je z veľkej časti používaný pri vývoji samotného operačného systému. V roku 2017 bol ako ďalší oficiálny jazyk pridaný Kotlin [23].

Na tento operačný systém sa dá pozerať ako na zásobník 2.2, kde každá vrstva má špecifickú úlohu, poskytuje špecifické služby a využíva funkcie vrstiev pod ňou. Následujúci zoznam je prevzatý z [18] a popisuje jednotlivé vrstvy tohoto zásobníka:

- **Hardvér** – je nutné si uvedomiť, že aj napriek tomu, že nie je priamo súčasťou zásobníka, kladie hardvér, pre ktorý bol systém Android navrhnutý, pomerne prísne obmedzenia. Najvýznamnejšie z týchto obmedzení je výkon.
- **Jadro** – Android závisí od jadra Linuxu. Jadro je zodpovedné za poskytovanie základných služieb, s ktorými vývojári pracujú. Medzi tieto služby patrí napríklad súborový systém, vlákna a procesy, prístup k sieti a rozhrania k hardvérovým zariadeniam.
- **Systémové služby** – vrstva systémových služieb zahŕňa širokú škálu nástrojov - od kódu, ktorý beží ako súčasť jadra (ovládače alebo moduly jadra), a dlhodobo bežiacich aplikácií, ktoré spravujú rôzne úlohy (démoni), až po knižnice, ktoré implementujú štandardné funkcie.
- **Prostredie pre beh systému Android** – prostredie *Android Runtime Environment* je kolekcia knižníc, ktoré sú používané v aplikácií príkazmi ako napríklad `import: android.view, android.os`. Sú implementované pomocou dvoch jazykov, zvyčajne Java a C alebo C++. Časť implementácie, ktorú aplikácia využíva prostredníctvom importu, je napísaná v jazyku Java.' Java však využíva rozhranie *Java Native Interface* (JNI) na vyvolanie natívneho kódu, zvyčajne napísaného v jazyku C alebo C++. Tento natívny kód následne komunikuje so systémovými službami.
- **Aplikácie** -– na vrchole zásobníka sú aplikácie systému Android. Skladajú sa z individuálne adresovateľných komponent, ktoré môžu byť volané inými aplikáciami. Programy *Camera* a *Contacts* sú príkladmi aplikácií systému Android, ktoré sú využívané ako služby inými aplikáciami.

Kotlin

Kotlin je staticky typovaný programovací jazyk, ktorý beží na *Java Virtual Machine* (JVM) a môže byť skompilovaný do zdrojového kódu JavaScript. Verejne bol vydaný vo februári roku 2016 [3]. Jedna z hlavných dizajnových smerníc pri vývoji jazyka Kotlin opisovala lepšie zaobchádzanie s nulovými hodnotami. Spracovanie nulových hodnôt v jazyku Java môže bez špecifických kontrol viesť k výnimkám *NullPointerException* (NPE), ktoré sú hlavnou príčinou zlyhania takmer 30% aplikácií pre Android. Medzi kľúčové vlastnosti jazyka Kotlin patria čitateľnosť a stručnosť, najmä pri deklarácií objektov a tried s početnými atribútmi. [2]



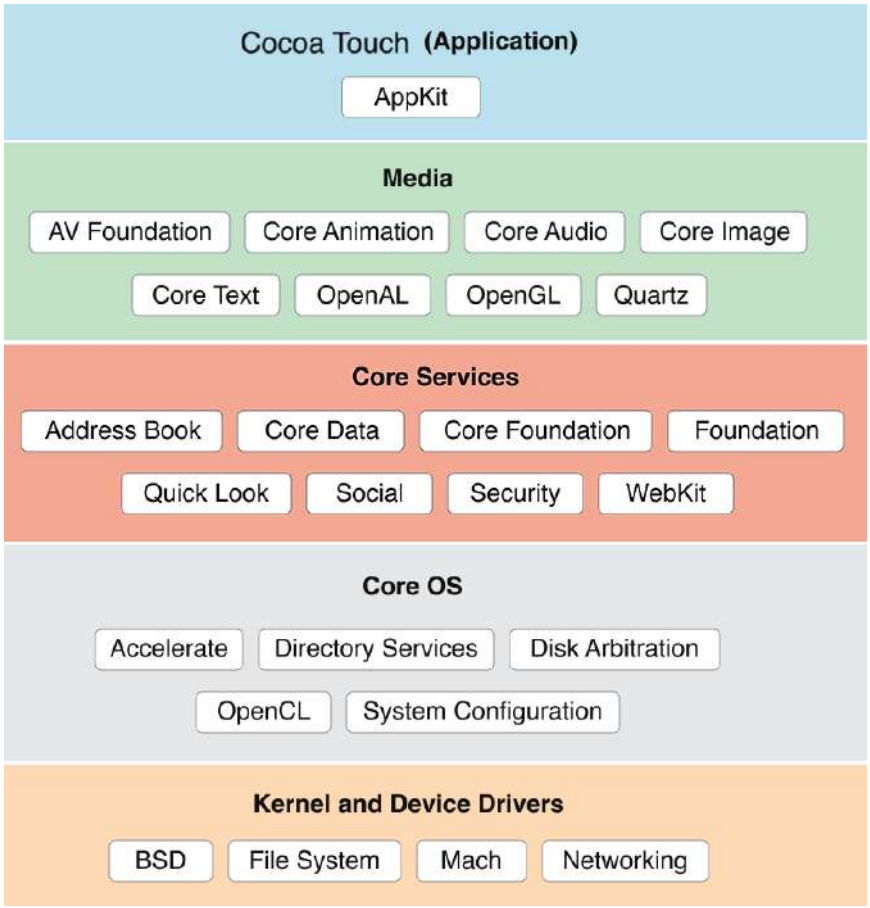
Obr. 2.2: Pohľad na Android, ako na zásobník komponent.

2.1.2 iOS

Operačný systém iOS je unixový zásobník softvérových komponent, ktorý pochádza priamo z vývojovej cesty operačného systému OS X [5]. Je využívaný mobilnými zariadeniami spoločnosti Apple, najmä v telefónoch iPhone a tabletoch iPad. Aplikácie, ktoré bežia v systéme iOS, sú dostupné v obchode App Store. Vďaka menšiemu počtu zariadení, na ktoré sa musia vývojári iOS zamerať, majú väčšiu kontrolu nad svojou aplikáciou a môžu ju vo väčšej miere prispôsobiť, čo robí vývoj pohodlnejším. Natívne aplikácie pre iOS je možné písať v jazyku Objective-C alebo Swift, za použitia vývojového prostredia Xcode IDE, ktoré je dostupné iba na zariadeniach s operačným systémom MacOS. Pre vývoj sa odporúča používať Swift, pretože je modernejší a používanejší [11]. Následujúci zoznam je prevzatý z [19, 20] a popisuje zásobník softvérových komponent systému iOS:

- **Jadro a ovládače zariadení** – ide o najnižšiu vrstvu systému iOS, ktorá zahŕňa najmä jadro a ovládače zariadení. Prostredie jadra je postavené na Mach 3.0 (mikrojadro, ktoré nahrádza jadro vo verzii BSD Unixu) a poskytuje vysoko výkonné sieťové prostriedky a podporu viacerých integrovaných súborových systémov.
- **Jadro operačného systému** – jadro operačného systému (OS) pozostáva z technológií a rámcov, ktoré poskytujú nízkoúrovňové služby súvisiace s nízkoúrovňovým hardvérom a sieťami. Tieto služby sú založené na zariadeniach z nižšej vrstvy.
- **Základné služby** – táto vrstva pozostáva zo základných služieb, ako sú adresár, bezpečnosť, sociálne siete a *foundation*, ktoré aplikáciám poskytujú základné funkcie.
- **Média** – vrstva médií obsahuje súbor základných technológií, ktoré môžu byť použité na podporu 2D a 3D grafiky, animácií, obrazových efektov a funkcionality zvuku a videa na profesionálnej úrovni.

- **Vrstva *Cocoa Touch*** – *Cocoa Touch* je najvyššia vrstva. Napriek tomu, že bol iOS odvodený z Mac OS X, tak tam vrstva *Cocoa Touch* nebola. Vznikla odvodením z *Cocoa* v Mac OS X a bola jedinečne navrhnutá pre dotykové rozhrania. Primárne zodpovedá za vzhľad aplikácií a poskytuje prístup k hlavným systémovým funkciám, ako sú kontakty, fotoaparát, dotykový vstup, zdieľanie s inými aplikáciami, *push* oznámenia a podobne.



Obr. 2.3: Hlavné časti unixového zásobníku softvérových komponent operačného systému iOS. Prevzaté z [5].

Swift

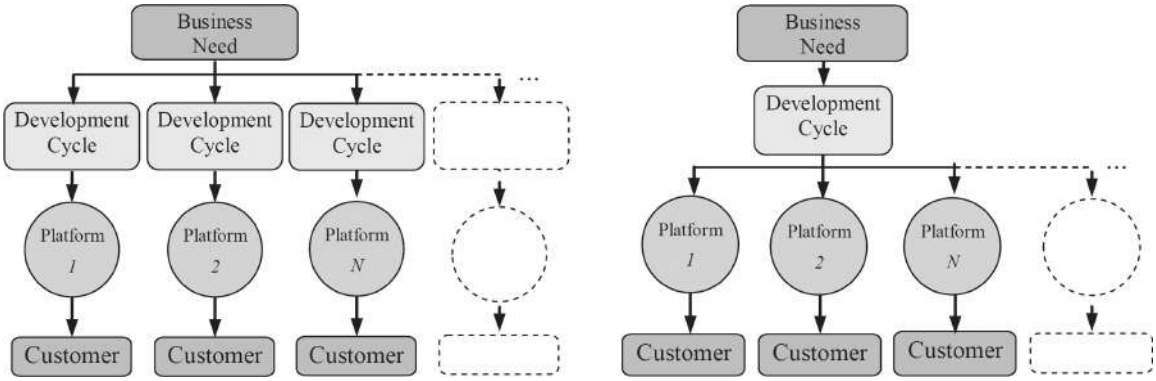
Swift je pomerne moderný jazyk vytvorený spoločnosťou Apple pre iOS. Bol zverejnený v roku 2014. Napriek tomu, že Objective-C bol obľúbený medzi vývojármi, tak bol pokladaný za zastaralý a mal svoje nedostatky. Je viac ako 30 rokov starý, založený na jazyku C, má zvláštnu a rozvláčna syntax a nebezpečný typový systém. To znamená, že môže byť náchylný na chyby alebo problémy, ktoré by mohli spôsobiť nestabilitu alebo nesprávne správanie aplikácie. Swift bol preto vytvorený ako moderná alternatíva a navrhnutý s ohľadom na konkrétne vylepšenia. Spoločnosť Swift zaviedla niekoľko programovacích konceptov na zníženie množstva bežných chýb programátorov. Patrí medzi ne napríklad silné písanie a spravovanie chýb. Taktiež boli zavedené interné optimalizácie pre rýchly chod jazyka Swift.

Xcode taktiež obsahuje upozornenia pri písaní kódu pre zaistenie optimálneho behu aplikácie. Ďalším vylepšením je expresívny kód, ktorý zachováva správnu rovnováhu medzi jasnosťou významu a stručnosťou. Jednoduchšie povedané, bol navrhnutý tak, aby bol kód ľahko zrozumiteľný bez toho, aby bol príliš dlhý alebo zložitý. Swift je teda považovaný za vylepšenú verziu Objective-C [13] v nasledovných aspektoch:

- bezpečnosť,
- výkon,
- expresivita.

2.2 Cross-platform vývoj aplikácií

Cross-platformové nástroje na vývoj mobilných aplikácií dokážu skompilovať zdrojový kód aplikácie pre viacero zdrojov. Prístupov k tvorbe mobilných aplikácií pomocou multiplatformových alternatív je veľa a sú trochu mätúce, preto je treba správne nástroje vyberať s rozvahou. Je nutné zvážiť viacero faktorov, ako napríklad výber SDK, užívateľskú skúsenosť (UX), stabilitu *frameworku*, jednoduchosť aktualizácie, náklady na vývoj a čas potrebný na uvedenie aplikácie na trh [24]. Často je potrebné dodať aplikáciu rýchlejšie a lacnejšie, čo je možné vďaka úspore času a úsilia vynaloženého na vývoj. Tento problém vedie k existencii multiplatformových riešení pre vývoj mobilných aplikácií. Hlavnou myšlienkou multiplatformových riešení je vyvinúť aplikáciu raz a spustiť ju kdekoľvek. Na obrázku 2.4 je možné vidieť rozšírenie tradičného životného cyklu vývoja softvéru tým, že aplikácia je napísaná raz, ale nasadená viackrát, aby podporovala rôzne platformy. Na vývoj mobilných aplikácií je k dispozícii niekoľko nástrojov a rámcov, ale žiadny z nich sa zatiaľ nestal definitívnym riešením, ktoré by úplne vyriešilo všetky problémy spojené s vývojom na viacerých platformách. Hoci sú k dispozícii niektoré komerčné riešenia na vývoj mobilných aplikácií pre rôzne platformy, mnohé z týchto riešení sú stále zdokonaľované a vylepšované a neustále sa vyvíjajú tiež nové platformy. Vývojári si musia z existujúcich možností v závislosti od svojich konkrétnych potrieb a požiadaviek. [10]



Obr. 2.4: Tradičný a *cross-platformový model vývoja*. Prevzaté z [10].

2.2.1 React Native

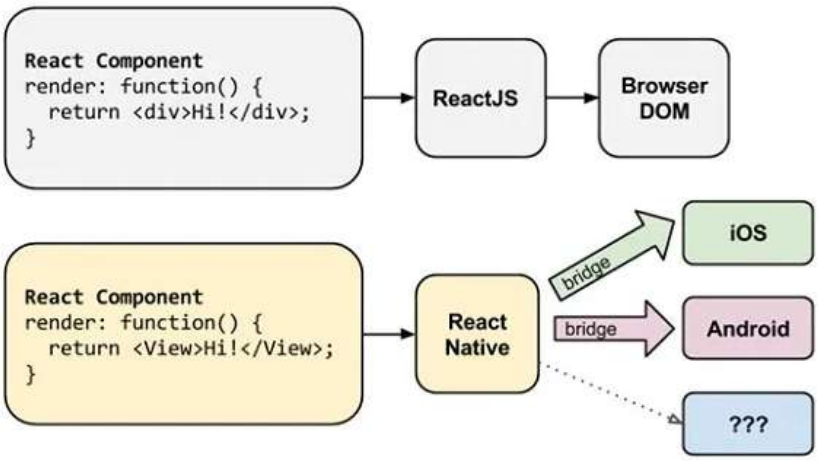
Knižnica React Native je natívny skriptovací *framework* na vývoj mobilných aplikácií pre rôzne platformy. Bol vyvinutý spoločnosťou Facebook a vydaný v roku 2015. Pôvodne mal

byť zameraný na vývoj pre iOS, ale komunita aktívnych vývojárov pridala podporu pre Android a urobila mnoho ďalších príspevkov do tohto open source projektu. React Native aplikácie sú vyvínuté pomocou JavaScriptu, presnejšie EcmaScriptu 2015 (ES2015 alebo ES6). EcmaScript je nadmnožinou jazyka JavaScript, ktorá do tohto jazyka pridáva množstvo funkcií a opravuje niektoré chyby pôvodne urobené v jeho špecifikácií [17]. Účelom knižnice React Native je postarať sa o to, aby od vývojára neboli vyžadované znalosti alebo trávenie nadbytočného času vytváraním dvoch mobilných aplikácií, keďže to by bolo minimum potrebné pre podporu iOS aj Android. Vzhľadom na to ako veľmi sa rôzne platformy líšia, či už sa jedná o vzhľad alebo schopnosti, neexistuje aplikácia, ktorá je homogénna na všetkých operačných systémoch. Keďže sa však líši grafické rozhranie, vývoj môže byť založený na rovnakom jazyku. Naopak grafika bude vykresľovaná v závislosti od cieľovej platformy, čím vzniknú skutočné natívne komponenty. Facebook tento prístup nazýva „učte sa raz, píšte kdekoľvek“, čo vlastne vysvetľuje podstatu knižnice React Native a multiplatformového programovania vo všeobecnosti.

React Native je založený na knižnici React, ktorá prináša svoje výhody do *frameworku* a následne ich aplikuje na natívne mobilné aplikácie. Namiesto toho, aby sa React spúšťal v prehliadači a vykresľoval *divy*, texty a podobné elementy, beží knižnica React Native vo vstavanej inštancii JavaScriptCore (pre iOS) alebo V8 (pre Android) priamo v aplikácii a vykresľuje komponenty špecifické pre danú platformu, ako môžeme vidieť na obrázku 2.5. Komponenty v jazyku JavaScript sú deklarované prostredníctvom sady vstavaných primitív, ktoré sú podporované iOS alebo Android komponentami.

Kľúčovou súčasťou knižnice React Native je most (bridge), ktorý umožňuje komunikáciu medzi kódom v jazyku JavaScript a natívnymi modulmi danej platformy. Keď komponenta jazyka JavaScript potrebuje vykonať nejakú akciu, ktorá je špecifická pre danú platformu (napríklad zobrazenie natívneho tlačidla), požiadavka sa posiela cez most do natívneho kódu, ktorý túto akciu vykoná a vráti výsledok späť do jazyka JavaScript.

Na rozdiel od väčšiny mobilných multiplatformových vývojových prístupov, knižnica React Native nie je hybridným riešením, čo je vývojový prístup, ktorý kombinuje natívny a webový vývoj a je možné ho distribuovať na rôzne platformy. Nespolieha sa na vykresľovanie vo *Web-Views* ani sa nepokúša napodobňovať HTML a CSS, a teda dokáže vytvárať aplikácie s natívnou responzivitou [7]. Následujúce pod-sekcie sú prevzaté z [27].



Obr. 2.5: Vykresľovanie v React a React Native. Prevzaté z [9].

JSX

Pri študovaní kódu React Native je pomerne jednoduché si všimnúť, že využíva JSX. JSX je špeciálne rozšírenie syntaxe jazyka JavaScript, ktoré sa používa fundamentálne na popísanie toho, ako by sa malo používateľské rozhranie (UI) zobrazovať. JSX bude po skompilovaní aplikácie konvertovaný do normálneho objektu JavaScript. Typický fragment kódu JSX môže vyzerať takto:

```
const SampleComponent = (props) => {
  const { label } = props;
  const handlePress = () => {
    console.log('Button clicked');
  };

  return (
    <View>
      <Text>
        Sample Component
      </Text>

      <TouchableOpacity onPress={handlePress}>
        <Text>
          {label}
        </Text>
      </TouchableOpacity>
    </View>
  );
};
```

Výpis 2.1: Ukážka kódu funkcionálnej komponenty React Native za použitia JSX.

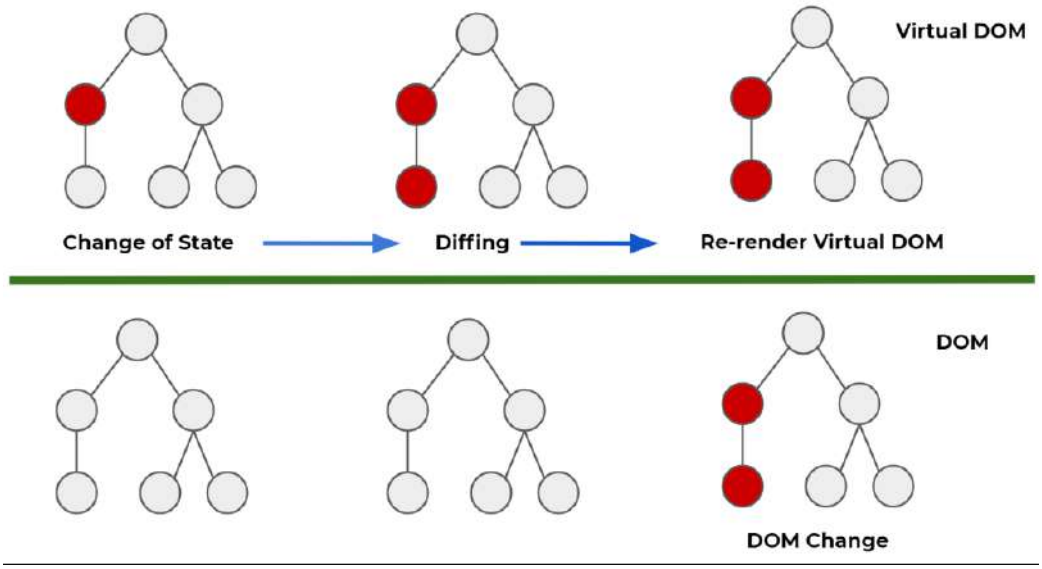
V ukážke kódu 2.1, je vytvorená funkcionálna komponenta s jedným tlačítkom. Funkcia *return* je kľúčová pre každú takúto komponentu knižnice React, kedže sa v nej vráti objekt pohľadu danej komponenty. Kučeravé zátvorky sa používajú na de-štrukturalizáciu atribútov komponenty vo vnútri funkcie komponenty, čo znamená rozdelenie objektov alebo polí na jednotlivé premenné. Využitie funkcie šípky v *handlePress* je dobrým príkladom toho, ako JSX využíva moderné funkcie jazyka JavaScript.

Keďže sa JSX používa na popis používateľského rozhrania, je možné sa domnievať, že JSX je jednoducho ďalší šablónový jazyk, ako napríklad HTML alebo XAML. To však nie je pravda. JSX a normálny objekt jazyka JavaScript sú multi-konvertibilné, čo znamená, že je možné napísať normálny výraz v jazyku JavaScript uprostred JSX. React DOM je knižnica, ktorá je zodpovedná za spravovanie virtuálneho DOM (viď pod-sekcia 2.2.1). Táto knižnica transformuje JSX do jazyka JavaScript, aby mohol byť kód spracovaný a vykreslený. Knižnica taktiež zaisťuje, aby bola akákoľvek zadaná hodnota užívateľom spracovaná do bežného reťazca. Používateľ preto nikdy nemôže vložiť skripty alebo príkazy do vašej aplikácie prostredníctvom jej rozhrania.

Virtuálny DOM

DOM, alebo *Document Object Model* je programovacie rozhranie pre webové dokumenty. Reprezentuje webové dokumenty vo forme stromovej štruktúry, kde jednotlivé uzly stromu predstavujú elementy webovej stránky.

Jedným z hlavných dôvodov, prečo môže React Native aplikácia bežať na rôznych platformách, je používanie virtuálneho DOM. Virtuálny DOM umožňuje Reactu manipulovať s ľahkým stromom DOM, ktorý je namapovaný so skutočným DOM stromom, pre zvýšenie výkonu. Pracovný postup virtuálneho DOM je vidieť na obrázku 2.6.



Obr. 2.6: Vizuálne zobrazenie zmien v DOM. Prevzaté z [6].

Keď používateľ interaguje s aplikáciou, napríklad kliknutím na tlačidlo, vytvára udalosti, ktoré ovplyvňujú štruktúru virtuálneho DOMu. Aplikácia pravidelne spúšťa *diffing* algoritmus, ktorý porovnáva aktualizovaný virtuálny DOM so staršou verziou pre efektívnu aktualizáciu skutočného modelu DOM. Vždy, keď je potrebné vykresliť prvok JSX, tak sa zároveň zaktualizuje jeho zodpovedajúci virtuálny DOM objekt. Pred aktualizáciou React urobí snímku aktuálneho stromu virtuálneho DOM. Vďaka tomu môže React po aktualizácii porovnať aktualizovaný DOM s predchádzajúcou snímkou, aby našiel tie časti, ktoré je potrebné znovu vykresliť v skutočnom strome DOM. Tento postup vyžaduje súbor algoritmov ako sú napríklad algoritmy na rýchle porovnávanie stromov alebo metódy pre efektívne zisťovanie rozdielov medzi virtuálnym a skutočným DOM.

Po nájdení minimálneho počtu krokov na aktualizáciu virtuálneho DOM, React vykoná všetky tieto kroky v jednej iterácií udalostí bez toho, aby sa dotkol reálneho DOM. Po dokončení tejto iterácií udalostí React prekreslí reálny DOM. Celý tento proces sa nazýva zosúladenie. V knižnici React sú zosúlaďovanie a vykresľovanie dve samostatné fázy. React a React Native teda môžu používať svoje vlastné metódy vykresľovania, pričom využívajú rovnaký mechanizmus zosúladenia, ktorý poskytuje jadro knižnice React.

Atribúty a stav

Knižnica React má dva hlavné dátové modely: atribúty a stav. Atribúty sa nastavujú nadradenými komponentmi a používajú sa ako parametre na konfiguráciu komponenty. Sú

nemenné, čo znamená, že ich po nastavení nemožno zmeniť. Stav sa inicializuje vrámci komponenty a môže byť aktualizovaný podľa potreby. Často sa stav komponenty odovzdáva ako atribút jeho podriadeným komponentám, čo je možné vidieť v ukážkovom kóde 2.2. Stav komponenty reprezentuje kolekciu stavových premenných, ktoré komponenta využíva na správu svojich vnútorných stavov.

```
1 function ParentComponent() {
2 const [childsName, setChildsName] = useState('');
3
4 return <ChildComponent name={childsName} />;
5 }
```

Výpis 2.2: Ukážka prenosu stavu z nadradenej komponenty na podriadenú.

Bez-stavové komponenty sa ľahšie testujú a znovu používajú, a preto vývojári často preferujú správu stavu v nadradených komponentách a odovzdávajú ho podriadeným komponentám ako atribúty.

2.2.2 Expo

Expo je súbor nástrojov postavený na knižnici React Native, ktorý zjednodušuje vytváranie a distribúciu multiplatformového softvéru. Spravuje stavebné prostredie, poskytuje nástroje na testovanie a opravu chýb, ponúka množstvo nástrojov a služieb, ktoré môžu byť použité na vývoj, stavbu a publikáciu React Native aplikácií. Okrem toho ponúka Expo platformovo neutrálne API, ktoré umožňuje prístup k natívnym funkciám zariadenia, ako je fotoaparát, GPS a *push* notifikácie [14]. Hlavné rozdiely medzi vývojovým prostredím Expo a knižnicou React Native sú:

- **Nastavenie a konfigurácia** – Expo zjednodušuje proces nastavenia a konfigurácie, zatiaľ čo React Native vyžaduje manuálnu konfiguráciu natívnych modulov a prepojenia.
- **Prispôsobenie** – React Native poskytuje väčšiu flexibilitu pre prispôsobenie natívneho kódu a používanie knižníc tretích strán. Naopak Expo má určité obmedzenia spôsobené riadeným pracovným postupom a preddefinovanej sady komponent a rozhrania API.

Stručne povedané, Expo je efektívnejšie riešenie pre začiatočníkov, zatiaľ čo React Native poskytuje väčšiu flexibilitu pre pokročilé prispôsobenie a prístup k natívnemu kódu.

2.2.3 Firebase

Firebase bol spustený v apríli 2012 a v roku 2014 získaný spoločnosťou Google, za účelom poskytovania serverového riešenia pre vývojárov. V roku 2016 Google oficiálne spustil Firebase s cieľom generovania príjmu pre vývojárov, vytvárania úspešných aplikácií a podpory obchodného rastu. Je považovaný za platformu webových aplikácií, ktorá pomáha vývojárom vytvárať vysokokvalitné aplikácie. Údaje, ako napríklad dáta používateľov a dáta aplikácií, ukladá vo formáte JSON, ktorý nevyžaduje dotazy na vkladanie, aktualizáciu, odstraňovanie alebo pridávanie. Je možné ho použiť ako serverové riešenie systému a používa sa taktiež ako databáza, konkrétne NoSql, na ukladanie údajov. Kombinuje funkcie, ktoré urýchľujú integráciu cloudových databáz, pre webové aj mobilné aplikácie. Taktiež

inovuje rozhranie API, ktoré umožňuje rozšírené funkcie, ako je odosielanie správ a overenie služby. Tým sa minimalizujú náklady a čas potrebný na vybudovanie týchto služieb [16, 26]. Medzi služby poskytované platformou Firebase patria [22]:

- **Analytics** – poskytuje prehľad o používaní aplikácie a umožňuje vývojárom aplikácie pochopiť, ako užívatelia interagujú s aplikáciou. Bez nutnosti písania akéhokoľvek kódu poskytuje táto funkcia vývojárovi štatistické informácie o množstve užívateľov, veku, pohlaví a geografickej lokácií.
- **Cloud Messaging (FCM)** – predtým známa ako *Google Clouds Messaging* (GCM), FCM je služba, ktorá je multiplatformovým riešením posielania správ a notifikácií pre Android, webové aplikácie, a IOS. Taktiež umožňuje vývojárom mobilných aplikácií posielať upozornenia konkrétnym používateľom.
- **Authentication** – umožnuje prihlásenie pomocou Facebook, Google GitHub a Twitter. Je to služba, ktorá môže autentizovať používateľov iba pomocou kódu na strane klienta. Zahŕňa tiež systém správy používateľov pomocou ktorého vývojári môžu povoliť autentizáciu pomocou e-mailu a prihlasovacieho hesla uloženého vo Firebase.
- **Realtime databáza** – škálovateľná a flexibilná cloudová databáza NoSQL, ktorá ukladá údaje do kolekcií a dokumentov. Poskytuje synchronizáciu údajov v reálnom čase, výkonné možnosti dotazovania a offline prístup k údajom. Podporuje Android, iOS a webové platformy.
- **Firestore (Cloud Firestore)** – Firebase ponúka databázu a serverovú podporu pre aplikácie v reálnom čase. Poskytuje vývojárom rozhranie API na synchronizáciu a ukladanie údajov aplikácií do cloudového úložiska Firebase. Spoločnosť dodáva aj knižnice, ktoré uľahčujú prácu s aplikáciami pre Android, iOS a JavaScript.
- **Storage** – umožňuje jednoduchý a bezpečný prenos súborov bez ohľadu na kvalitu siete pre Firebase aplikácie. Využíva službu Google Cloud Storage, ktorá poskytuje cenovo-efektívne ukladanie objektov v cloude. Vývojár ho môže používať na ukladanie obrázkov, zvuku, videa alebo iného obsahu vytvoreného užívateľom. Služba je bezplatná s denným limitom na prenos dát. V prípade potreby existuje aj platená verzia, v ktorej užívateľ platí na základe množstva ukladaných dát.
- **Test Lab** Poskytuje cloudovú infraštruktúru na testovanie aplikácií. Umožňuje vývojárovi spustiť testovanie aplikácie na širokej škále rôznych zariadení a konfigurácie. Rôzne výsledky testov, ako sú snímky obrazovky, videá a protokoly sú potom dostupné v konzole Firebase.
- **Crashlytics** Aplikácia má zavedený systém, ktorý zisťuje a kategorizuje chyby na základe ich podobnosti. Tieto chyby sú potom zoradené podľa stupňa závažnosti. Okrem toho môže vývojár zahrnúť svoje vlastné udalosti do systému, aby lepšie pochopil, čo mohlo spôsobiť výskyt chyby.

Kapitola 3

Existujúce aplikácie

Výskum existujúcich aplikácií zahŕňa aktívny zber relevantných údajov o daných aplikáciach, ako napríklad o ich funkčnosti, použiteľnosti, dizajne a možnostiach interakcie s užívateľom. Na základe týchto údajov je následne možné porovnávať existujúce riešenia a vyhodnocovať, ako rôzne aplikácie riešia rovnaké problémy.

Taktiež je dôležité zohľadniť, že užívatelia si vytvárajú takzvané 'mentálne modely' na základe interakcií s existujúcimi aplikáciami. Tieto mentálne modely majú vplyv na ich očakávania od nových aplikácií, preto je nutné ich pri tvorbe nových aplikácií brať do úvahy.

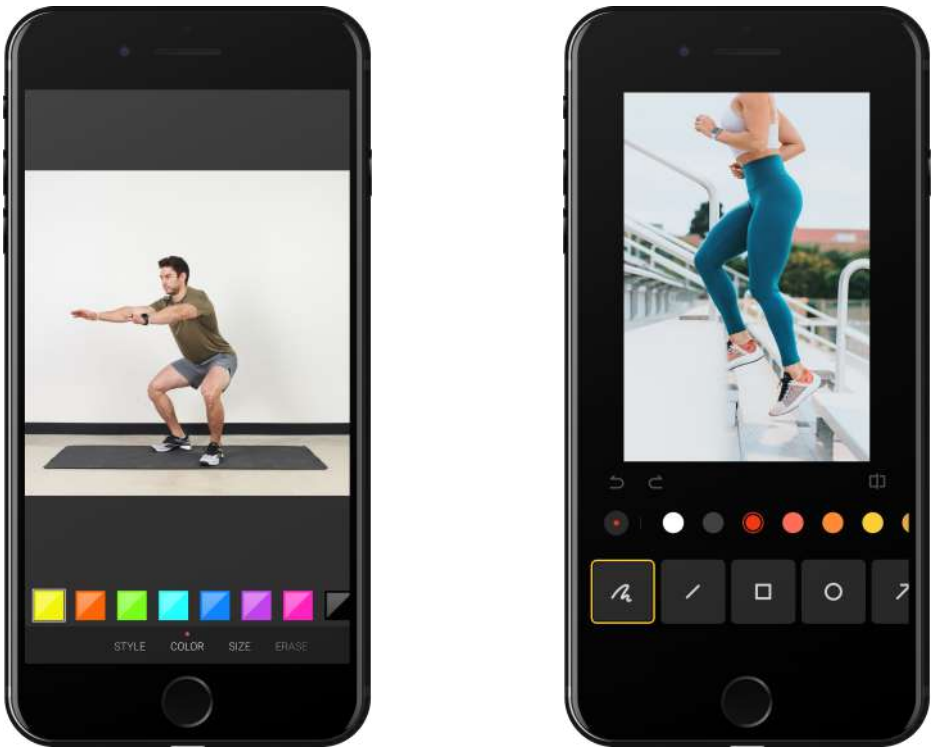
Pre vlastný výskum som musel dekomponovať problematiku svojej bakalárskej práce do niekoľkých častí. Dôvodom je, že úplne rovnaká aplikácia neexistuje, a preto som hľadal, ako sa tieto problémy riešia v iných aplikáciach. Patrí medzi ne:

- editácia obrázkov/videí,
- porovnanie videí,
- galéria pre videá,
- vkladanie komentárov do videí.

3.1 Editácia obrázkov

Napriek tomu, že v aplikácii ide o editáciu videí je prieskum zameraný na editáciu obrázkov, z dôvodu návrhu a implementácie riešenia bližšie popísaného v refieditvideo. Cieľom tohto prieskumu bolo zistiť, aké komponenty boli použité na editáciu a akým spôsobom riešili tieto aplikácie rozloženie jednotlivých komponent tak, aby bolo používanie tejto aplikácie pre užívateľa dostatočne intuitívne a priateľské. V mojom prípade však pojem editácia znamená možnosť kreslenia do obrázku, preto som v daných aplikáciach skúmal iba prvky umožňujúce túto funkcionalitu. Medzi aplikácie, z ktorých som sa inšpiroval vrámci prieskumu patria Youcam Perfect (viď obrázok 3.1a) a natívna aplikácia zariadenia Xiaomi (viď obrázok 3.1b). Pri oboch týchto aplikáciach je možné si všimnúť jednoduchého a priehľadného dizajnu. Hlavným rozdielom medzi týmito aplikáciami je to, že natívna Xiami aplikácia má všetky nástroje potrebné pre kreslenie viditeľné na obrazovke. Konkrétne sa jedná o rôzne farby, typ čiary alebo šírku nakreslenej čiary. Ikonu pre šírku čiary však nepokladám za dostatočne intuitívnu. Naopak aplikácia Youcam Perfect má posúvací súbor nástrojov. Pre použitie akéhokoľvek z týchto nástrojov je teda nutne ho najprv zvoliť, a následne nastaviť

jeho vlastnosti. Ak by teda užívateľ chcel napríklad nakresliť vlnitú čiaru s určitou šírkou a farbou, vyžadovalo by to dvojnásobok počtu krokov potrebných pre vykonanie rovnakej akcie v natívnej aplikácií Xiamoi.



(a) Kreslenie v aplikácií Youcam Perfect.

(b) Kreslenie v natívnej aplikácií zariadenia Xiaomi.

Obr. 3.1: Rozloženie obrazovky s galériou v populárnych aplikáciach pre galériu.

3.2 Porovnanie videí

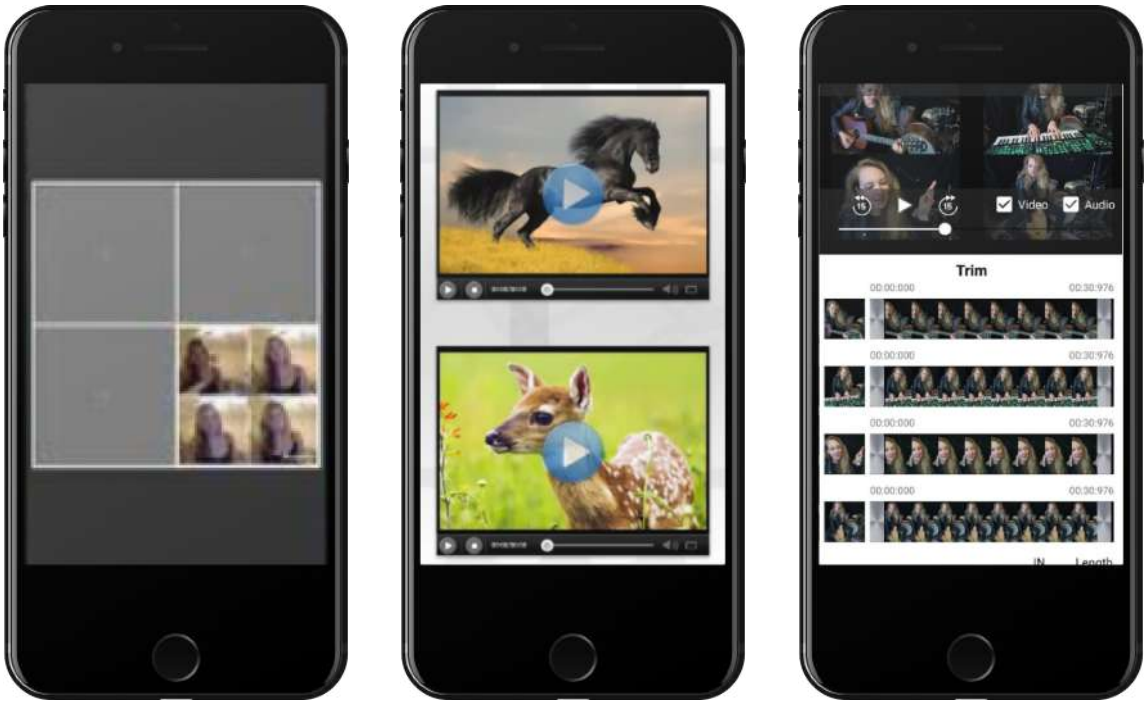
Cieľom implementácie tejto funkcie je umožniť ľuďom, ktorí trénujú doma, alebo profesionálom, ktorí sa snažia zdokonaliť svoju techniku, možnosť porovnať vlastné tréningové videá za účelom sledovania pokroku, alebo vlastné s referenčnými, pre zistenie správnej techniky cviku. To môže slúžiť nielen k zlepšeniu techniky cvičenia, ale pri začiatočníkoch aj ako prevencia proti zraneniu z dôsledku nesprávneho cvičenia.

Porovnanie videí predstavuje jednu z najdôležitejších funkcionalít tejto bakalárskej práce a teda je nutné si uvedomiť, čo je vlastne potrebné v existujúcich aplikáciach hľadať. Medzi hlavné problémy patrí spôsob, akým je možné pridať tieto videá do porovnania a následne rozloženie, v ktorom sa vybrané videá zobrazia.

Jednou z aplikácií, ktoré sa zaoberajú týmto problémom je napríklad Ubersense (viď obrázok 3.2). Pre účel výskumu však nie je nevyhnutné, aby skúmané aplikácie riešili porovnanie videí v športovom kontexte. V skutočnosti je možné využiť prvky rozloženia aj z aplikácií, ktoré využívajú takzvaný *split-screen* v akomkoľvek odvetví. *Split-screen* je technika úpravy videa, pri ktorej sa na obrazovke súčasne zobrazujú dva alebo viac videoklipov, zvyčajne vedľa seba alebo v oddelených častiach. Obrazovka je rozdelená na samostatné časti, pričom v každej časti sa prehráva iný videoklip.

Medzi aplikácie, ktoré implementujú *split-screen* patria napríklad 4xcamera, Video Merge a Acapella (viď obrázok 3.2). Je možné si povšimnúť množstva zaujímavých funkcií, ako napríklad možnosť pridávať videá do porovnania až po navigácií na obrazovku, čo je možné vidieť na ukážke 3.2a aplikácie 4xcamera. Je to jednoduché a intuitívne riešenie toho, ako užívateľovi umožniť pridať videá do porovnania. Užívateľ iba klikne na plochu, ktorá reprezentuje priestor pre video, a následne si nejaké vyberie.

Ďalšou zaujímavou funkciou je prehrávanie a pohyb vo všetkých porovnávaných videách iba pomocou jedných navigačných tlačidiel. To je vyriešené v aplikácií Video Merge (viď obrázok 3.2b), kde sa však videá dajú iba zlúčiť a neexistuje jednoduchý spôsob výmeny videa v porovnaní. Taktiež je táto funkcia implementovaná v aplikácií Acapella (viď obrázok 3.2c), z ktorej som sa pre tvorbu tlačidiel a ich rozloženia inšpiroval, však bolo nutné urobiť isté zmeny, keďže táto aplikácia nie je prioritne určená na porovnávanie videí športovcov a teda tomu nezodpovedá ani užívateľské rozhranie.



(a) Ukážka porovnania videí v aplikácii 4xcamera.

(b) Ukážka porovnania videí v aplikácii Video Merge.

(c) Ukážka porovnania videí v aplikácii Acapella.

Obr. 3.2: Riešenie problematiky *split-screen* v rôznych aplikáciach, z ktorých som využil prvky užívateľského rozhrania.

Ubersense

Ubersense sa považuje za aplikáciu 'vytvorenú s cieľom umožniť ľuďom, aby sa stali lepšími v športe'. Používatelia môžu nahrať video, na ktorom robia niečo atletické, pozrieť si ho v spomalenom zábere a zverejniť ho v komunite Ubersense, aby získali spätnú väzbu a povzbudenie. Nahrávanie videa pomocou aplikácie Ubersense je jednoduché, pretože klipy možno nahrávať aj priamo z fotoaparátu zariadenia. Po vložení do systému možno video prezerať v spomalenom zábere, označovať pomocou grafických nástrojov v aplikácii a ukladať na neskôr. Používatelia môžu tiež porovnať svoje zručnosti s inými používateľmi alebo

v niektorých prípadoch s profesionálnymi športovcami pomocou funkcie side-by-side v aplikácii.

Ďalšou funkciou je 'Komunita Ubersense', ktorá slúži ako kanál s možnosťou filtrovania, kde si používatelia môžu prezerať, *lajkovať*, zdieľať alebo ukladať videá nahrané ostatnými. Táto sociálna interakcia povzbudzuje používateľov, aby sa od seba navzájom učili, podporovali svoje pokroky a udržiavali si motiváciu v úsilí o dosiahnutie športovej dokonalosti. Celkovo je Ubersense cenným nástrojom pre športovcov a trénerov, ktorí sa snažia zlepšiť svoje športové výkony prostredníctvom analýzy videí, porovnávania a zdieľania v rámci podpornej komunity. [25]

Užívateľské rozhranie pri porovnávaní videí (viď obrázok 3.3) takmer presne odpovedá cielenému porovnaniu videí vo výslednej aplikácií tejto bakalárskej práce.

Medzi hlavné pozitíva tejto aplikácie patrí intuitívne umiestnenie tlačidla 'Compare', keďže je pravdepodobné, že ak bude chcieť užívateľ porovnať video, tak ho najprv otvorí. Ďalej je veľmi prehľadne navrhnutý dizajn a rozloženie prvkov pre ovládanie videa. Má to však zopár nedostatkov, ktorými sú napríklad:

- **Nemožnosť synchronizácie videí** – nie je možné uložiť porovnanie v zosynchronizovanej podobe, a potom si ho znova spustiť.
- **Jediný spôsob porovnania videí** – nie je možné dostať sa k porovnaniu videí iným spôsobom, ako po otvorení konkrétneho videa. Aplikácia, v ktorej porovnanie predstavuje jednu z hlavných funkcií by to však mala umožňovať nejakým ďalším spôsobom.
- **Náročná výmena porovnávaného videa** – v prípade, že užívateľ zvolí do porovnania iné video, než chcel tak sa musí spätne vrátiť a obrazovku video-prehrávača a zvoliť iné video do porovnania.

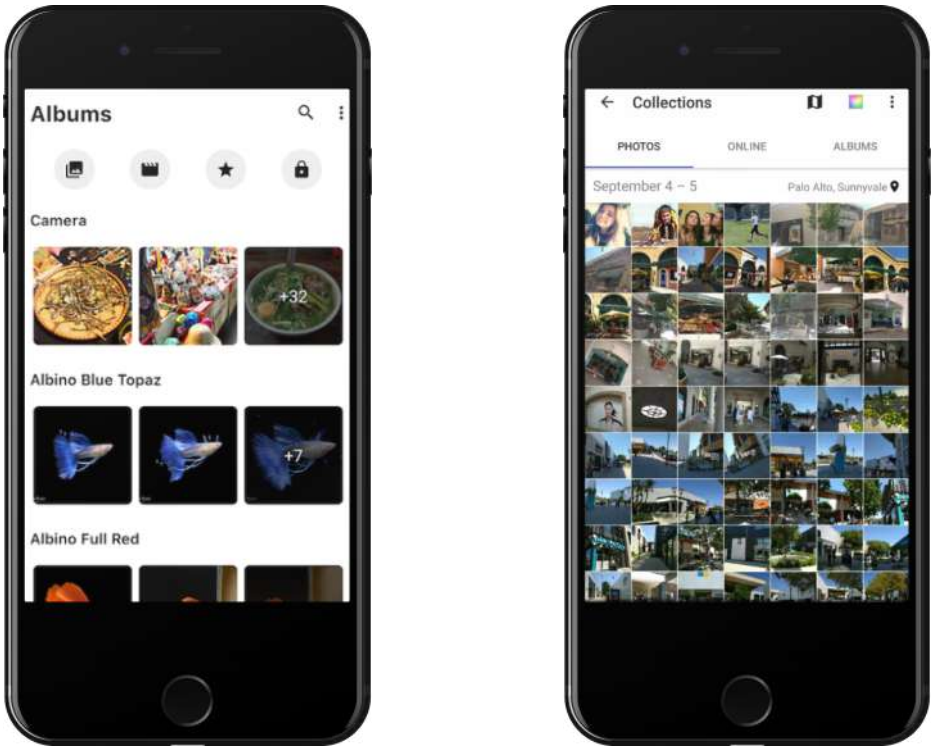
Obr. 3.3: Porovnanie videí v aplikácii Ubersense.

3.3 Video galéria

Galérie sú dôležitým aspektom mnohých aplikácií, ktoré pracujú s vizuálnym obsahom, ako sú obrázky alebo videá. Ak je užívateľské rozhranie vhodne implementované, tak umožní užívateľovi efektívne prehľadávať a interagovať s obsahom. Medzi najdôležitejšie vlastnosti aplikácií, ktoré je nutné zvážiť pri ich skúmaní patrí rozvrhnutie multimediálneho obsahu, jednoduchosť navigácie a dostupné metódy interakcie s obsahom. Medzi tieto interakcie môže patriť napríklad filtrovanie obsahu. Ukážky takýchto aplikácií sú znázornené na obrázku 3.4. V týchto riešeniach galérie je možné vidieť rôzne spôsoby kategorizácie.

Na prvom obrázku 3.4a sú jednotlivé kategórie usporiadané pod sebou a videá je možné horizontálne posúvať vrámci každej kategórie pre zobrazenie ďalších videí rovnakej kategórie. Dôsledkom toho je galéria prehľadná a používanie tejto aplikácie je intuitívne. Tento spôsob implementácie kategorizácie však v mojej aplikácie nestačí, keďže bude nutné rozlišovať nielen kategórie jednotlivých cvikov, ale aj typy videí. Konkrétne sa jedná o klasické videá, synchronizované videá a videá zdieľané s trénerom.

Ako je možné vidieť na druhej ukážke 3.4b, aplikácia obsahuje vrchný tab bar, ktorý umožní užívateľom zrýchliť proces hľadania obrázkov. Aplikácia taktiež umožňuje filtráciu mnohými spôsobmi, ako napríklad dátumom, lokalitou, ale aj farbou.



(a) Ukážka galérie 1Gallery. (b) Ukážka galérie A+ Gallery.

Obr. 3.4: Rozloženie obrazovky s galériou v populárnych aplikáciach pre galériu.

3.4 Vkladanie komentárov do videí

Za účelom zlepšenia techniky cvičenia klienta, je nevyhnutné, aby aplikácia poskytovala efektívnu komunikáciu medzi klientom a trénerom. Vkladanie komentárov do špecifického bodu vo videu umožní trénerovi poskytnúť presnú spätnú väzbu na konkrétne aspekty výkonu klienta. To pomôže klientovi pochopiť, ktorá časť jeho techniky si vyžaduje zlepšenie alebo úpravu. To bude zároveň v tejto aplikácií okrem editácie pomocou kreslenia do videa predstavovať jediný spôsob komunikácie medzi trénerom a klientom. Cieľom tejto funkcionality je umožniť trénerovi vyhodnotiť a posúdiť techniku cvičenia klienta a následne zobraziť toto vyhodnotenie klientovi, čo v konečnom dôsledku vedie k lepším výkonom a pokroku pri dosahovaní svojich športových cieľov. Ukážku zobrazenia takýchto komentárov je možné vidieť na obrázku 3.5. Napriek peknému a prehľadnému užívateľskému rozhraniu je táto implementácia pre moju aplikáciu nevhodná a dá sa využiť iba čiastočne. Dôvodom toho je, že komentáre, ktoré sú vo videu sa zobrazujú automaticky, čo by mohlo športovcovi pri prezeraní videa prekážať.

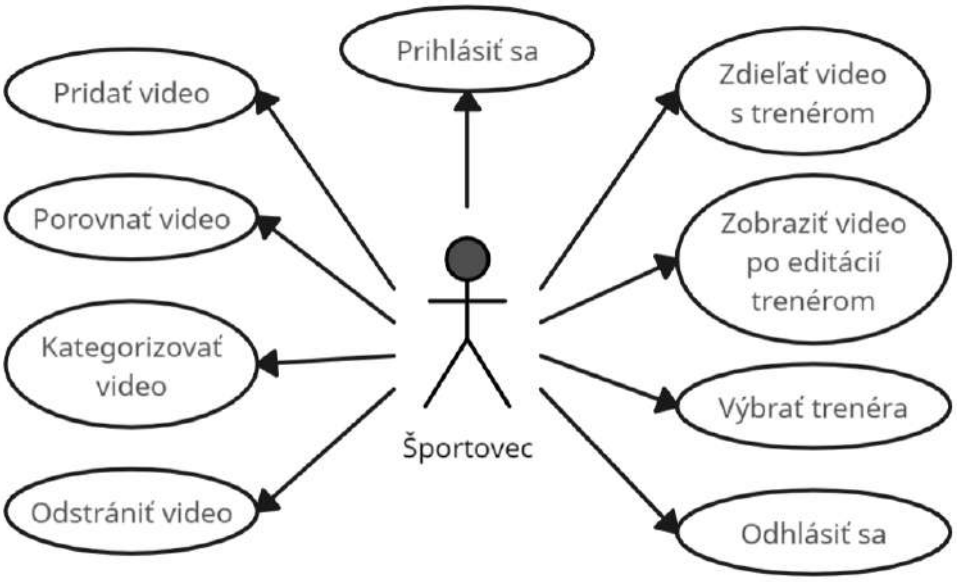
Obr. 3.5: Rozloženie obrazovky pri zobrazovaní komentárov aplikácie Frame.io. Pri prejdení cez konkrétne body, v ktorých sa komentár nachádza, sa daný komentár automaticky zobrazí.

Kapitola 4

Návrh aplikácie

Táto kapitola vychádza z prieskumu existujúcich aplikácií v predošlej kapitole 3. Popisuje, aké problémy sa vyskytli pri navrhovaní aplikácie, a ako boli riešené. Kapitola je rozdelená do dvoch hlavných sekcií, keďže v aplikácii existujú dve hlavné typy používateľov, konkrétne športovec a tréner. Prvým krokom pri návrhu bolo vytvoriť diagram prípadov použitia pre športovca (viď obrázok 4.1) a trénera (viď obrázok 4.6). Po vytvorení týchto diagramov, je jasne vidieť aké akcie je nutné implementovať pre jednotlivý typ užívateľa. V nasledujúcich sekciách sa zameriam na jednotlivé prípady použitia a ich návrh.

4.1 Športovec



Obr. 4.1: Diagram prípadov použitia – športovec

Prihlásenie

Prihlásenie je základným prvkom každej aplikácie, ktorá potrebuje odlíšiť užívateľov. Navrhnutie jednoduchého rozhrania pre prihlásenie však vôbec nie je náročné. Dobre navrhnutý

proces prihlásenia by mal uprednostňovať používateľské pohodlie a vizuálne príťažlivé rozhranie. Je napríklad dôležité jednoznačne vyznačiť polia, kde by mal užívateľ zadať prihlasovacie údaje, ponúknuť možnosť zostať prihlásený a prípadne ponúknuť viacero možností prihlásenia, ako je napríklad užívateľské meno, email alebo sociálne siete. Ďalej je vhodné pridať možnosť zobrazenia hesla pri zadávaní, čo môže pomôcť užívateľom s komplexným heslom a slúžiť ako prevencia proti chybám a opakovaným pokusom o prihlásenie. Aplikovaním týchto prvkov je možne užívateľovi spríjemniť skúsenosť s používaním danej aplikácie.

Domovská obrazovka

Po prihlásení bude užívateľ navigovaný na domovskú obrazovku (viď obrázok 4.3). Pomocou tejto obrazovky popíšem návrh následujúcich prípadov použitia:

- pridanie videa,
- odstránenie videa,
- kategorizovanie videa.

Pridanie videa

Pridanie videa predstavuje základnú funkciu tejto aplikácie, keďže celá funkcionalita bude postavená okolo videí. Je preto dôležité, aby hneď po prihlásení vedel užívateľ rýchlo a jednoznačne identifikovať spôsob pridávania videí. Navrhol som viacero verzií, ako napríklad umiestnenie tlačidla na pridanie videa do menu v pravej alebo ľavej časti obrazovky. To sa však ukázalo byť neprehľadné a teda ďalšou verziou bolo pridanie tlačidiel na viditeľné miesta obrazovky. To však taktiež nebolo možné, keďže vzhľadom na množstvo akcií potrebných na domovskej obrazovke by bolo nutné pridať príliš veľa tlačidiel. Problém s pridaním viacero tlačidiel bol v tom, že videá mali byť kvôli prehľadnosti kategorizované, a teda neostávalo veľa voľného priestoru na tlačidlá. Preto sa ako najrozumnejšie riešenie zdalo byť pridanie takzvaného plávajúceho akčného tlačidla (FAB), ktoré som nakoniec zvolil, ako je možné vidieť na návrhu 4.2a. Tlačidlo je vždy na vrchu obrazovky, a je ľahko dostupné pre užívateľa, kedže to bude po prihlásení jedna z prvých vecí, ktorú si užívateľ všimne. Toto tlačidlo som navrhol podľa odporúčaní z návrhových usmernení Material Design [12]. Obsahuje rozbaľovacie menu, ktoré umožní pridanie videa, kategórie a taktiež predstavuje spôsob navigácie do funkcie porovnania videí.

Odstránenie videa

Spôsob odstránenia videa je navrhnutý veľmi jednoducho. V pravom rohu každého videa sa nachádza tlačidlo'X', ktoré toto odstránenie umožní (viď obrázok 4.2a). Túto ikonu som zvolil z dôvodu, že ikona 'X' je typicky používaná ako symbol pre zatvorenie alebo odstránenie v rôznych aplikačných rozhraniach. Z tohto dôvodu užívateľ okamžite pochopí, k čomu toto tlačidlo slúži. Ďalším dôvodom je jednoduchý dizajn, kde takáto ikona jednoduchšie zapadne. Rovnaká ikona je taktiež využitá aj pri porovnávaní videí na rýchle odstránenie porovnávaného videa (viď obrázok 4.3b), čo zaistí konzistenciu používania tejto ikony skrz aplikáciu.

Odstránenie videí je navrhnuté týmto spôsobom z dôvodu, že pravdepodobne nebude nikdy nutné ich odstraňovať hromadne, keďže sa nejedná o galériu mobilného telefónu, kde sa videá a obrázky pridávajú automaticky. Každé video, ktoré bude pridané musí užívateľ

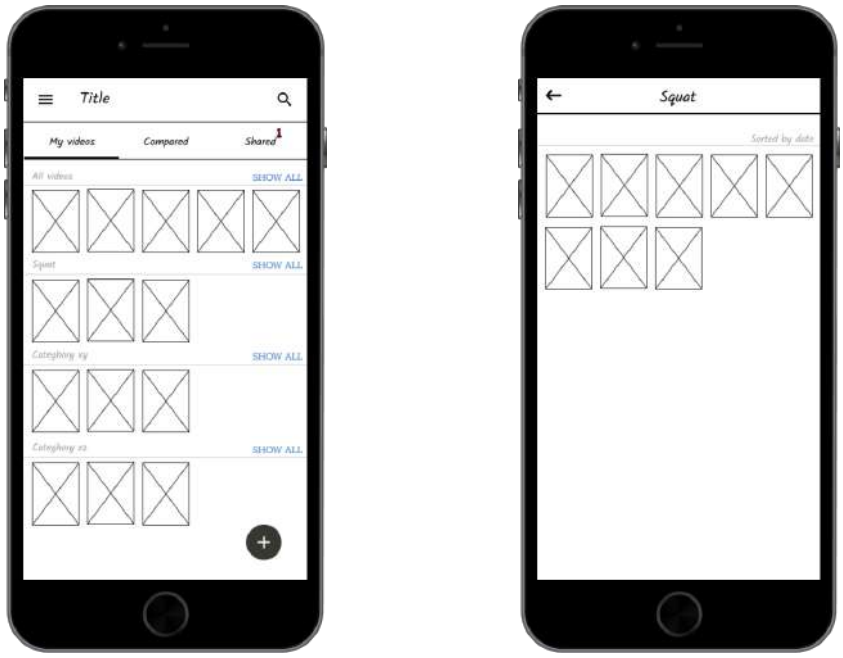
vybrať a pridať manuálne, a preto pri nutnosti odstránenia je to týmto spôsobom rýchle, očividné a jednoducho prístupné.

Kategorizovanie videa

Pri návrhu kategorizácie som využil prvky oboch aplikácií, ktorých ukážky boli zahrnuté vrámci výskumu existujúcich aplikácií 3.4.

Pri aplikácií A+ Gallery (viď obrázok 3.4b) sa jedná o hornú lištu na navigáciu medzi obrazovkami pridaných videí, porovnaných videí a taktiež videí, ktoré sú zdieľané s trénerom.

Druhým prevzatým prvkom je spôsob kategorizácie videí vrámci jednotlivých obrazoviek, ktorý je možné vidieť na ukážke 3.4a z aplikácie 1Gallery. Každá kategória je zobrazená v jednom riadku, v ktorom sa nachádza jej názov, prehľad niekoľkých videí a možnosť navigácie do konkrétnej kategórie. Ukážku návrhu konkrétnej kategórie je možné vidieť na obrázku 4.2b, kam sa vie užívateľ dostať pomocou tlačítka 'SHOW ALL'.



(a) Ukážka návrhu domovskej obrazovky.

(b) Ukážka návrhu detailu kategórie.

Obr. 4.2: Návrh domovskej obrazovky a prvkov užívateľského rozhrania.

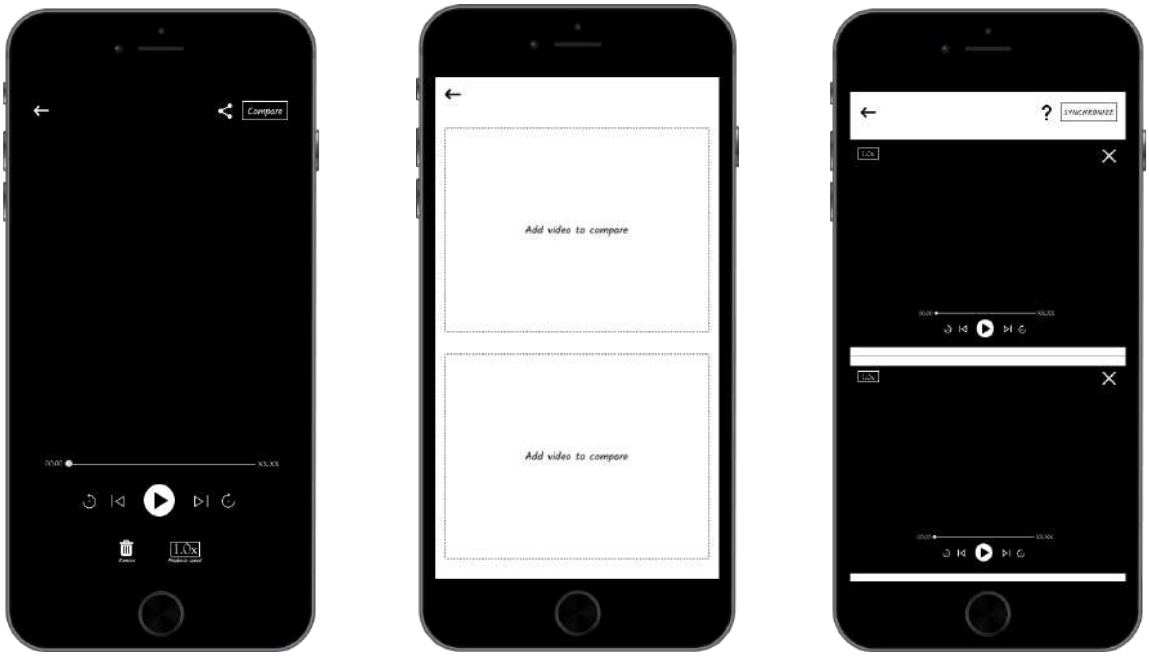
Porovnanie

Vzhľadom na to, že porovnávanie videí patrí medzi hlavné funkcie tejto aplikácie, tak som pokladal za vhodné umožniť užívateľovi dostať sa k porovnaniu viacerými spôsobmi. Jedným zo spôsobov je tlačidlo 'Compare', ktoré sa nachádza v pravom hornom rohu videoprehrávača obrazovky 'MyVideos' (viď obrázok 4.3a). Druhým spôsobom, ako bolo vyššie spomínané je rozbalenie menu plávajúceho akčného tlačidla a vybranie tejto funkcie. Ďalej na obrázku 4.3b je možné vidieť jednoduchý návrh porovnania videí, kde ťuknutie na každý z priestorov pre videá umožní výber a pridanie videa do porovnania. Následne po

vybraní oboch videí na porovnanie sa zobrazí tlačidlo 'Synchronize' a ikona otázniku, ktorá toto tlačidlo popisuje (viď obrázok 4.3c). Konkrétne sa jedná o funkciu, ktorá umožní vytvoriť synchronizáciu videí. Po synchronizácií bude možné spustiť tieto porovnané videa v rovnakom rozložení ako pri porovnávaní, však už v zosynchronizovanej podobe.

Zdieľanie videa

Video, ktoré chce užívateľ odoslať musí byť vždy nejakým spôsobom označené. Problém však nastáva v tom, že video sa v galérií zobrazuje iba ako obrázok, a teda hromadné označovanie v tomto prípade nedáva zmysel. To z dôvodu, že je pravdepodobné, že videá budú zachytávať podobnú činnosť s podobným pozadím. To môže spôsobiť neschopnosť užívateľa rozoznať chcené video na základe podobných obrázkových náhľadov. Z toho vyplýva, že najjednoduchším spôsobom označenia konkrétneho videa je zobrazenie daného videa, keďže v takom prípade má užívateľ istotu, že sa skutočne jedná o video, ktoré chcel poslať. Návrh je teda vytvorený na základe týchto predpokladov. Zdieľať video s trénerom je možné priamo stlačením ikony na zdieľanie (viď obrázok 4.3a).



(a) Návrh zobrazenia videa zo sekcie 'MyVideos'. V pravom hornom rohu sa nachádza ikona na zdieľanie videa s trénerom a tlačidlo 'Compare', ktoré užívateľa naviguje na obrazovku porovnania videí.

(b) Návrh rozloženia obrazovky v režime porovnania, kde užívateľ môže vybrať videá na porovnanie.

(c) Návrh rozloženia obrazovky po vybraní videí, kde sa zobrazí tlačidlo na synchronizáciu a ikona otáznika, ktorá toto tlačidlo po ťuknutí vysvetlí.

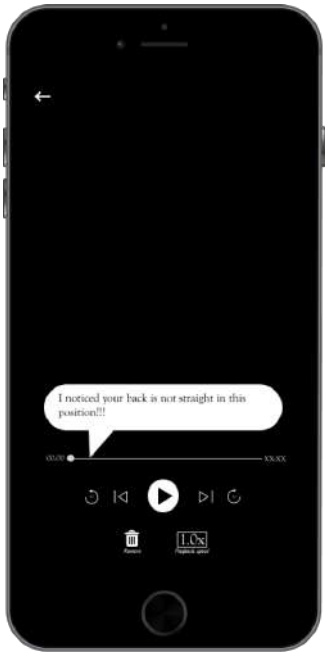
Obr. 4.3: Ukážka porovnania videí a videoprehrávača.

Zobrazenie videa po editácií trénerom

V situácii, v ktorej tréner skontroloval a vyhodnotil video športovca je nevyhnutné, aby boli spôsob zobrazenia a čas editácií trénera čo najpresnejšie a najpriehľadnejšie. Z tohto dôvodu som navrhol zobrazenie miniatúrnych ikon komentárov a editácií kreslením tesne nad časovou osou (viz obrázok 4.4a). Po stlačení ikony komentáru sa tento komentár zobrazí (viď obrázok 4.4b). Komentáre reprezentujú spôsob textového ohodnotenia videa trénerom, zatiaľ čo kreslenie poskytne presnú vizuálnu informáciu.

Užívateľ bude schopný nie len si zobraziť dané editácie, ale taktiež bude vedieť presne kedy a v akom okamihu pochybil. Tento dizajn teda zaisťuje, že spätná väzba je presná a intuitívna, čo pomáha športovcom zlepšovať sa podľa pokynov ich trénera.

(a) Zobrazenie videa po editácií trénerom. Na obrázku je možné vidieť miniatúrne komentáre, ktoré je možné zobraziť stlačením.

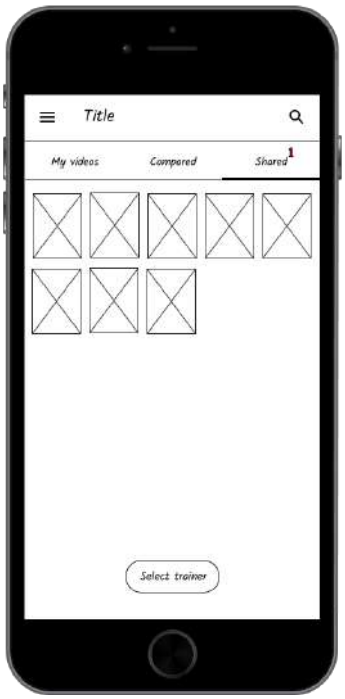


(b) Ukážka zobrazeného komentáru napísaného trénerom.

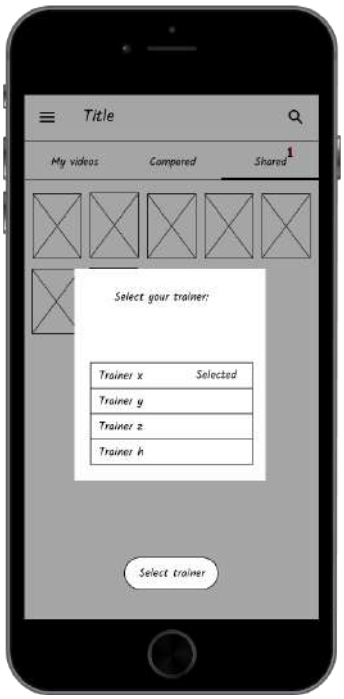
Obr. 4.4: Návrh zobrazenia videa po editácií trénerom.

Výber trénera

Voľba trénera je možná na obrazovke zdieľaných videí (viď obrázok 4.5a). Športovec by si mal byť schopný vybrať nového trénera kedykoľvek, keď už nie je s aktuálnym spokojný a preto je návrh vytvorený spôsobom ľubovoľnej zmeny trénera. Pri rozbalení okna pre výber (viď obrázok 4.5b) bude stále viditeľný aktuálne zvolený tréner na vrchole zoznamu. Dôsledkom toho bude športovec vždy presne vedieť, akého má aktuálne zvoleného trénera a v prípade potreby ho môže kedykoľvek zmeniť.



(a) Ukážka obrazovky pre zdieľané videá, na ktorej sa taktiež nachádza tlačidlo pre voľbu trénera.



(b) *Modálové* okno, ktoré slúži k výberu trénera. Na vrchu zoznamu je zobrazený a vyznačený aktuálne zvolený tréner.

Obr. 4.5: Návrh obrazovky zdieľaných videí a voľby trénera.

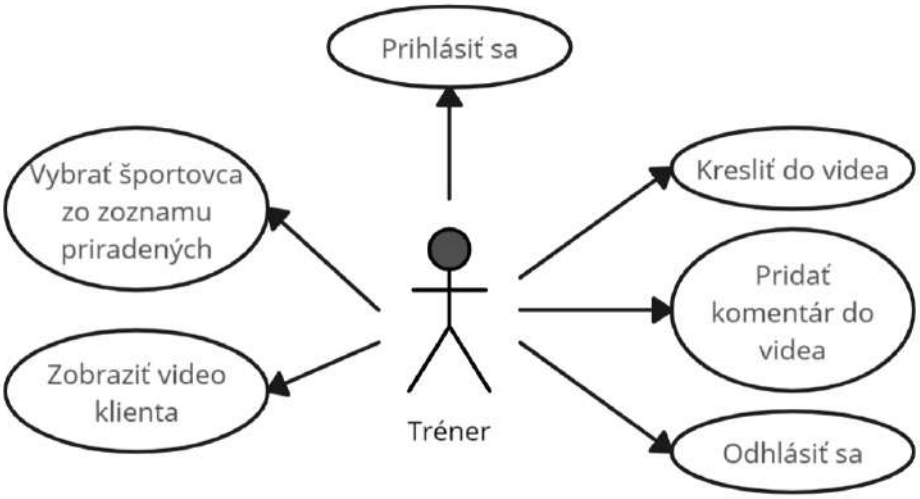
4.2 Tréner

Výber z priradených športovcov a zobrazenie videa

Po prihlásení bude tréner presmerovaný na hlavnú obrazovku (viď obrázok 4.7a), kde uvidí športovcov, ktorí si ho zvolili za trénera. Tréner musí mať jednoduchý prístup ku všetkým športovcom, čo by pri väčšom množstve predstavovalo problém, preto je v pravom hornom rohu pridané vyhľadávanie. Po vybraní športovca sa trénerovi zobrazia všetky zdieľané videá, ktoré mu športovec poslal na ohodnotenie. Tieto videá budú rozdelené do dvoch kategórií, a to nové a už spracované videá (viď obrázok 4.7a). Tento jednoduchý dizajn návrhu umožní trénerovi sa efektívne dopracovať k videám konkrétneho klienta, ktoré je potreba ohodnotiť.

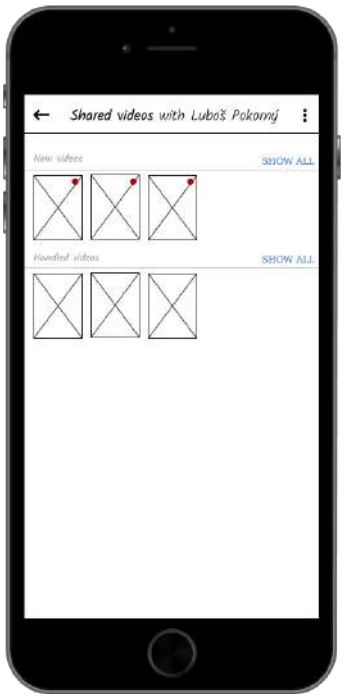
Zobrazenie a editácia videa klienta

Ako už bolo spomínané v sekcii 4.1, tréner má k dispozícií dva spôsoby, ako ohodnotiť video a poskytnúť tým čo najpresnejšiu spätnú väzbu. Konkrétne sa jedná o komentovanie a kreslenie do videa. Tieto funkcie sú dostupné stlačením tlačidiel v pravom hornom rohu obrazovky, po zobrazení videa športovca (viď obrázok 4.8a). Kreslením môže tréner zvýrazniť určité polohy, trajektórie alebo pohyby, ktoré by mohli byť pre športovca menej zrejmé len prostredníctvom komentára. Komentáre potom môžu poskytnúť ďalší kontext



Obr. 4.6: Diagram prípadov použitia — tréner

(a) Na obrázku je vidieť zoznam športovcov, ktorý si zvolili daného užívateľa ako trénera.



(b) Zdieľané videá užívateľa s trénerom.

Obr. 4.7: Ukážka návrhu výberu športovca a zobrazenia jeho videí.

alebo vysvetlenie k týmto vizuálnym značkám. Táto kombinácia metód umožňuje trénerovi detailne vysvetliť, kde má športovec nedostatky a pomôcť mu ich napraviť.

Pri zadaní komentáru a editácie kreslením, bude toto ohodnotenie zobrazené športovcovi presne v čase, v ktorom sa nachádzal prehrávač videa keď tréner ťukol na tieto tlačidlá. Pri stlačení ikony pre kreslenie, bude video v editačnom móde (viď obrázok 4.8b), kde sa

namiesto videa zobrazí iba obrázok momentu, kde chce tréner kresliť. Týmto spôsobom je zaistená presná spätná väzba. Užívateľské rozhranie v tomto editačnom móde bolo inšpirované aplikáciami z kapitoly prieskumu existujúcich aplikácií (viď obrázok 3.1).

(a) Na obrázku je možné vidieť zobrazené video športovca z perspektívy trénera. V pravom hornom rohu sa nachádzajú ikony komentáru a editácie kreslením.

(b) Návrh editačného módu trénera, pre ohodnotenie videa prostredníctvom kreslenia do snímky z videa.

Obr. 4.8: Ukážka návrhu zobrazenia videa športovca a editácie trénerom.

4.3 Testovanie prototypu

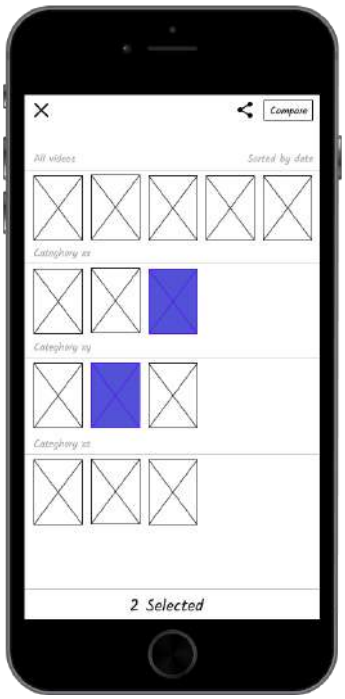
Testovanie prebiehalo iteratívne. V každom kole iterácie bol vytvorený nový, prípadne upravený a vylepšený návrh, ktorý bol testovaný na 5-10 užívateľoch. Užívatelia dostali za úlohu splniť niekoľko úloh, ktoré odpovedali výslednému a cielenému používaniu aplikácie. Počas toho, ako používatelia vykonávali dané úlohy som sledoval, akým spôsobom sa snažili dopracovať k cieľu, ako rýchlo to dokázali a či vôbec. Následne po testovaní som s každým užívateľom diskutoval jednotlivé úlohy a ich proces riešenia. Snažil som sa zistiť, či to bolo dostatočne intuitívne a prehľadné. Zadané úlohy boli nasledovné:

1. Predstav si, že si v roli športovca. Nahraj video a pozri si ho.
2. Vyber si z aplikácie existujúce video, a porovnaj ho s videom, ktoré si pridal.
3. Nie si si istý technikou cvičenia vo videu, ktoré si pridal. Pošli trénerovi video na ohodnotenie.

4. Teraz si v roli trénera. Prezeráš video klienta a všimneš si zhrbeného chrbátu pri cviku v druhej minúte daného videa. Akým spôsobom tu chybu zaznačíš do videa?

5. Znova si v roli športovca. Tréner si pozrel tvoje video, všimol si nedostatkov, napísal ti k nemu komentár a prišlo ti upozornenie. Nájdi jeho reakciu na tvoje video.

Výsledky tohto testovania mi poskytli cennú spätnú väzbu, keďže som si pri sledovaní užívateľoch počas testovania uvedomil, že nie je všetko tak intuitívne, ako sa mi pôvodne zdalo. Po prevedení testovania nasledovalo adresovanie daných chýb, konzultovanie zmien s vedúcou mojej práce a úprava návrhu podľa zistených nedostatkov. Konkrétna ukážka týchto iteratívnych úprav je znázornená na obrázku 4.9. Na obrázku 4.9a, je možné vidieť prvotný návrh na zdieľanie a porovnanie videí. Princíp spočíva v tom, že ak chce užívateľ poslať viacero videí, tak dlho podrží vybrané video, čo ho dostane do označovacieho módu. V tomto móde by mal užívateľ možnosť buď označiť a odoslať trénerovi ľubovoľné množstvo videí na vyhodnotenie, alebo v prípade že označí iba 2 videá, tak by mal možnosť tuknúť na tlačidlo 'Compare', ktoré by užívateľa navigovalo do porovnávania týchto videí. Tento návrh však neuspel, keďže ani jeden z užívateľov, ktorý testovali prototyp, neprišiel na tento spôsob porovnania a zdieľania videí. Dôsledkom toho bolo zdieľanie a porovnanie presunuté priamo do konkrétneho videa, ako je možné vidieť na obrázku 4.9b.



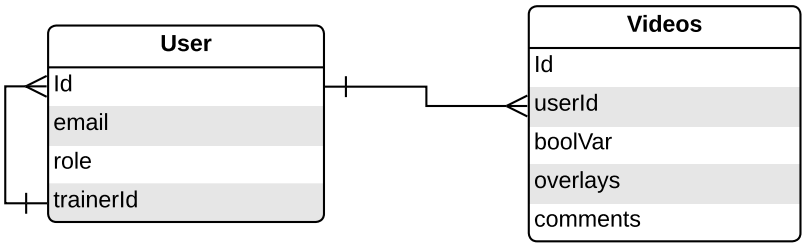
(a) Pôvodný návrh označovania videí za účelom ich porovnania a zdieľania.

(b) Ukážka finálneho návrhu polohy tlačidla pre porovnanie videí.

Obr. 4.9: Ukážka úprav návrhu po testovaní.

4.4 Návrh databázy

Ako *backend* projektu, čo predstavuje celú funkcionalitu aplikácie, ako napríklad vykonávanie výpočtov, a spracovanie dát, ktorá prebieha na pozadí a nie je viditeľná pre užívateľa, som použil Firestore 2.2.3. Schéma databázy je veľmi jednoduchá, keďže je nutné ukladať iba údaje o užívateľoch, a videách, ktoré títo užívatelia zdieľali trénerovi.



Obr. 4.10: Ukážka návrhu databázy, v ktorej je možné vidieť kolekcie užívateľa a videá

Kapitola 5

Implementácia

Táto kapitola popisuje implementáciu aplikácie, ktorú som nazval 'MotionMatch', a vychádza z návrhu predstaveného v predošlej kapitole 4. Pre vývoj klientskej časti aplikácie som použil Expo (viď sekcia 2.2.2) a pre serverovú časť Firebase (viď sekcia 2.2.3). Napriek tomu, že je aplikácia vyvíjaná pomocou *cross-platformového* frameworku React Native, tak je dostupná a testovaná iba na zariadeniach Android z toho dôvodu, však v budúcnosti bude vhodné poskytnúť ju tiež pre platformu iOS.

5.1 Štruktúra projektu

```
/
├── App.js ..... hlavný súbor aplikácie
├── app.json ..... konfiguračný súbor aplikácie
├── assets ..... adresár pre obrázky, fonty, atď.
├── components ..... opakovateľné komponenty React
│   ├── Category.js ..... komponent pre kategorizáciu
│   ├── FAB.js ..... komponent pre plávajúce akčné tlačidlo
│   ├── modals ..... adresár pre modálne komponenty
│   ├── screens ..... adresár pre komponenty obrazoviek
│   ├── tabs ..... adresár pre komponenty navigácie tabuliek
│   └── videoPlayers ..... adresár pre komponenty prehrávača videí
├── eas.json ..... konfiguračný súbor pre EAS Build
├── firebaseConfig.js ..... konfiguračný súbor Firebase
├── firebase.json ..... konfiguračný súbor pre Firebase CLI
├── hooks ..... adresár pre vlastné háčiky React
│   ├── useStoredData.js ..... háčik pre prístup k uloženým dátam
│   └── useVideoComparisons.js ..... háčik pre porovnávania videí
├── metro.config.js ..... konfiguračný súbor pre Metro bundler
├── package.json ..... závislosti a skripty projektu
├── package-lock.json ..... presný strom závislostí
├── utils ..... adresár pre pomocné funkcie
│   └── firebaseFunctions.js ..... pomocné funkcie pre Firebase
```

5.2 Navigácia

Pohyb medzi jednotlivými obrazovkami bol implementovaný pomocou knižníc:

- `@react-navigation/native` – poskytuje základnú infraštruktúru pre navigáciu, vrátane podpory pre prechody medzi obrazovkami, zdieľanie stavu navigácie a rôzne možnosti konfigurácie.
- `@react-navigation/material-top-tabs` – tento balíček poskytuje navigáciu pomocou horných kariet v štýle Material Design. Je to dobré pre prípady, keď chcete mať na vrchole obrazovky niekoľko kariet, medzi ktorými môžu používatelia prechádzať.
- `@react-navigation/native-stack` – Tento balíček poskytuje implementáciu zásobníka navigácie, ktorá je optimalizovaná pre výkon a poskytuje funkcie ako sú prechody medzi obrazovkami, modálne okná a možnosti prispôsobenia vzhľadu hlavičky.

Princíp fungovania navigácie je, že sa najprv vytvorí navigačný kontajner, ktorý prestavuje vrchol navigačnej hierarchie. Následne sa pomocou funkcie 'createNativeStackNavigator()' vytvorí navigačný zásobník, v ktorom každá ďalšia obrazovka, na ktorú sa užívateľ dostane je pridaná na vrch zásobníku, a zároveň na vrch predošlej obrazovky. Je teda možné povedať, že predošlú obrazovku vlastne iba prekryje. Z tohto vyplýva, že pri spätnej navigácií, a teda vracaní sa späť na predošlé obrazovky pomocou šípky späť, sa obrazovky vlastne iba odkrývajú a zároveň odstraňujú zo zásobníka. Šípka späť je automaticky pridaná obrazovkám pri používaní knižnice pre navigačný zásobník. Pri implementácií bolo teda potrebné dbať na to, aby sa zaistilo, že predošlé obrazovky nespôsobovali problémy s výkonom. To som pri kritických obrazovkách vyriešil tak, že som predošlé obrazovky *unmount*-ol hneď po tom, ako sa z nich odišlo.

Pre vizuálnu predstavu implementácie navigácie je pridaná ukážka navigácie v kóde 5.1, kde 'Home' a 'Details' predstavujú názov obrazoviek vrámci navigácie a 'HomeScreen' a 'DetailsScreen' sú názvom konkrétných komponent, ktoré tieto obrazovky implementujú.

```
1  const Stack = createNativeStackNavigator();
2
3  const App = () => {
4    return (
5      <NavigationContainer>
6        <Stack.Navigator initialRouteName='Home'>
7          <Stack.Screen name='Home' component={HomeScreen} />
8          <Stack.Screen name='Details' component={DetailsScreen} />
9        </Stack.Navigator>
10     </NavigationContainer>
11   );
12 };
```

Výpis 5.1: Ukážka kódu implementácie zásobníku obrazoviek

Druhý spôsob navigácie vrámci aplikácie bol implementovaný pomocou funkcie 'createMaterialTopTabNavigator()' z knižnice `@react-navigation/material-top-tabs`, ktorá vytvorí na vrchu obrazovky *tab*, medzi ktorými je možné ľubovoľne prechádzať. Konkrétne sa jedná o obrazovky 'MyVideos', 'Compared' a 'Shared'. Tento typ navigácie je v aplikácií prepojený s navigačným zásobníkom, a to tým spôsobom, že konkrétna obrazovka zásobníku

obsahuje *tab* navigáciu. Pri využití dvoch typov navigácií týmto spôsobom vrámci jednej aplikácie sa hovorí o vnorenej navigácii. Tento princíp implementácie je možné vidieť na následujúcej ukážke kódu 5.2 v kontexte s predošlou ukážkou.

```
function HomeScreen() {
  return (
    <Tab.Navigator>
      <Tab.Screen name='My videos' component={MyVideos} />
      <Tab.Screen name='Compared' component={ComparedVideos} />
      <Tab.Screen name='Shared' component={SharedVideos} />
    </Tab.Navigator>
  );
}
```

Výpis 5.2: Ukážka implementácie horných *tab* obrazoviek

5.3 Registrácia a prihlásenie

Registrácia a prihlásenie je zabezpečené pomocou *Authentication* 2.2.3 z Firebase. Aby mohol užívateľ používať aplikáciu, tak sa najprv musí zaregistrovať, čo vytvorí v kolekcií užívateľov nový dokument užívateľa, kde názov dokumentu predstavuje ID daného užívateľa. Toto ID je poskytnuté Firebase metódou 'createUserWithEmailAndPassword' z *Autentication*, ktorá sa využíva pri registrácií. Táto funkcia najprv overí platnosť údajov, zadaných do políčka email a heslo, ako napríklad formu emailu, alebo minimálnu dĺžku hesla. Následne v prípade, že kontroly prebehli bez problémov vytvorí v Firebase *Autentication* nový užívateľský účet, a vráti informácie o novo vytvorenom užívateľovi, ktoré následne ukladám do kolekcie užívateľov v databáze *Firestore*. Dokument užívateľa obsahuje email a rolu, ktoré užívateľ zadáva vrámci registrácie.

Pri prihlásení sa využíva metóda 'signInWithEmailAndPassword', ktorá overí údaje, pomocou ktorých sa užívateľ autentizoval. V prípade, že takýto užívateľ existuje tak je navigovaný na domovskú obrazovku trénera alebo športovca, v závislosti od jeho role.

5.4 Športovec

Domovská obrazovka u športovca využíva vrchnú *tab* navigáciu 5.2, na navigáciu medzi troma hlavnými obrazovkami. Medzi ktoré patria 'My Videos', 'Compared' a 'Shared'. Ďalej táto obrazovka umožňuje odhlásenia stlačením menu s troma čiarkami, z knižnice `@expo/vector-icons`, ktoré sa nachádza naľavo od názvu aplikácie. Po stlačení sa zobrazí menu, ktoré obsahuje jedinú možnosť, a to odhlásenia.

5.4.1 Moje videa

Všetky nahrané videá sa zobrazia zoradené v kategóriach po kliknutí na *tab* obrazovku 'My Videos'. Kategórie predstavujú dynamický zoznam zobrazený pomocou komponenty 'Flatlist' z knižnice React Native 5.1. Táto komponenta je navrhnutá tak, aby zvládala veľké množstvo dát. Vzhľadom na to zobrazuje iba viditeľné položky zoznamu. Každá kategória obsahuje pole objektov, pričom každý takýto objekt obsahuje informácie o videu ako napríklad jeho URI, čo predstavuje cestu k miestu, kde je video uložené v pamäti zariadenia. Videá sa v aplikácií zobrazujú iba ako náhľadové snímky. Môže sa jednať napríklad o prvý rámec z videa. Týmto spôsobom sa šetria výpočetné zdroje zariadenia, keďže sa nemusí zobrazovať celé video, ale iba jedna snímka. Pri kliknutí na túto snímku, sa následne zobrazí

skutočné video vo video-prehrávači 5.4.3. V aplikácií je umožnené horizontálne posúvanie medzi videami vrámci každej kategórie. Každé z videí na tejto obrazovke taktiež obsahuje tlačidlo 'X', ktoré umožní vymazanie videa s medzikrokom, v ktorom vyskočí potvrdenie pre vymazanie 5.1. Tieto vyskakovacie okná sú implementované pomocou 'Modal' komponenty taktiež z knižnice React Native. Ďalej je vrámci každej kategórie tlačidlo 'SHOW ALL', ktoré užívateľa presmeruje na všetky videá vybranej kategórie, kde je možné kategóriu vymazať alebo premenovať. Tlačidlá sú v aplikácií implementované pomocou komponenty 'TouchableOpacity', ktorá narozdiel od komponenty 'Button' umožňuje tlačidlo upravovať do požadovanej podoby. Akýkoľvek prvok, zabalený v komponente 'Touchable Opacity', sa stáva stlačiteľným.

Poslednou dôležitou častou tejto obrazovky je tlačidlo *Floating Action Button* (FAB). Táto komponenta je prevzatá z knižnice `react-native-paper` a pri stlačení sa rozbalí do menu, obsahujúceho možnosti pridania videa, pridania kategórie alebo porovania videí (viď obrázok 5.1).

Pridanie videa prebieha pomocou funkcie *pickVideo*, ktorá využíva aplikačné rozhranie (API) knižníc `MediaLibrary` a `ImagePicker` na prístup ku galérií a voľbu ľubovoľného videa s veľkosťou do 7MB zo zariadenia užívateľa. Obmedzenie veľkosti je pridané pre zaistenie hladkého a bezproblémového behu aplikácie. Po zvolení videa má užívateľ možnosť si vybrať kategóriu zvoleného videa, v prípade že existujú aspoň dve kategórie. Následne je video pridané do danej kategórie a zároveň ak bola zvolená kategória iná ako 'AllVideos', tak je pridané aj do tejto kategórie. Kategórie a ich videá sú načítané a uložené pomocou knižnice `AsyncStorage`, ktorá tieto dáta perzistentne ukladá v pamäti telefónu. Za účelom šetrenia pamäte sa ukladajú iba údaje ako názvy kategórií a video-objekty obsahujúce URI, namiesto ukladania skutočných videí.

Pri stlačení tlačidla 'Add category' vybehne okno, do ktorého je možné napísať názov kategórie. Po vytvorení bude v kategórii text, ktorý hovorí o tom že je zatiaľ prázdna.

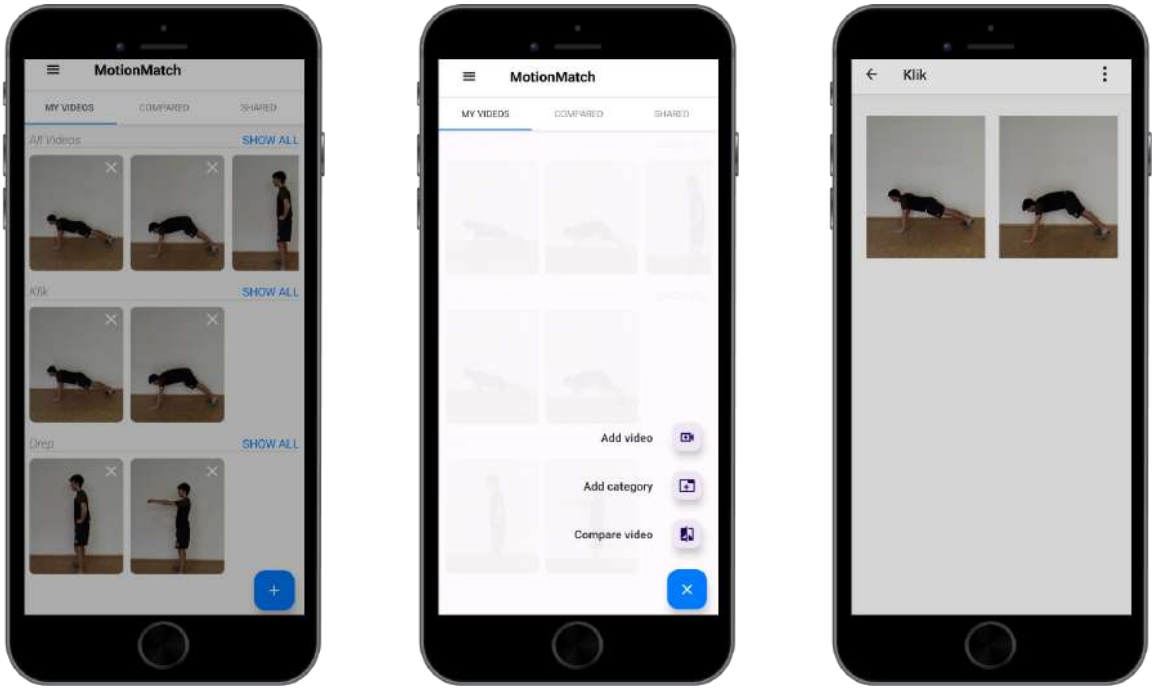
Poslednou možnosťou je ťuknutie na tlačidlo porovnania videí, ktoré presmeruje užívateľa na obrazovku pre porovnanie videí. Proces a ukážka porovnania videí sú popísané v pod-sekcii 5.4.2.

5.4.2 Porovnanie videí

Existujú dva spôsoby navigácie na túto obrazovku, jedným z nich je stlačenie tlačidla 'Compare videos', ktoré sa nachádza v menu FAB tlačidla 5.1. Druhým spôsobom je navigácia do porovnania priamo z video-prehrávača 5.4.3. Pri využití tohto druhého spôsobu, je zvolené video rovno pridané do porovnania.

Po navigácií na obrazovku pre porovnanie videí 5.2 je možné vidieť dva štvoruholníky, ktoré slúžia ako *placeholder* pre video 5.2. Oba tieto emelenty sú zabalené v komponente *TouchableOpacity*, vďaka čomu je na nich možné po celom obsahu tuknúť a sú usporiadané vertikálne vedľa seba. Po ťuknutí dostane užívateľ možnosť voľby medzi vybraním videa z kategórií v aplikácií alebo z pamäte zariadenia. V prípade výberu videa z pamäte telefónu je tam znova opatrenie na maximálnu veľkosť videa, rovnako ako pri pridávaní videa 5.4.1. Po zvolení oboch videí sa zobrazí tlačidlo 'SYNCHRONIZE' a vedľa neho ikona otázniku 5.2, ktorá toto tlačidlo po ťuknutí popíše. Konkrétne sa jedná o funkciu, ktorá umožňuje užívateľovi zastaviť obe videá v bode, v ktorom je pohyb oboch videí pri vykonávaní rovnakého cviku zosynchronizovaný.

Po stlačení tlačidla 'SYNCHRONIZE' bude užívateľ navigovaný na *tab* obrazovku 'Compared', kde sa táto synchronizácia uloží. Na ukladanie položky výsledku synchronizácie je



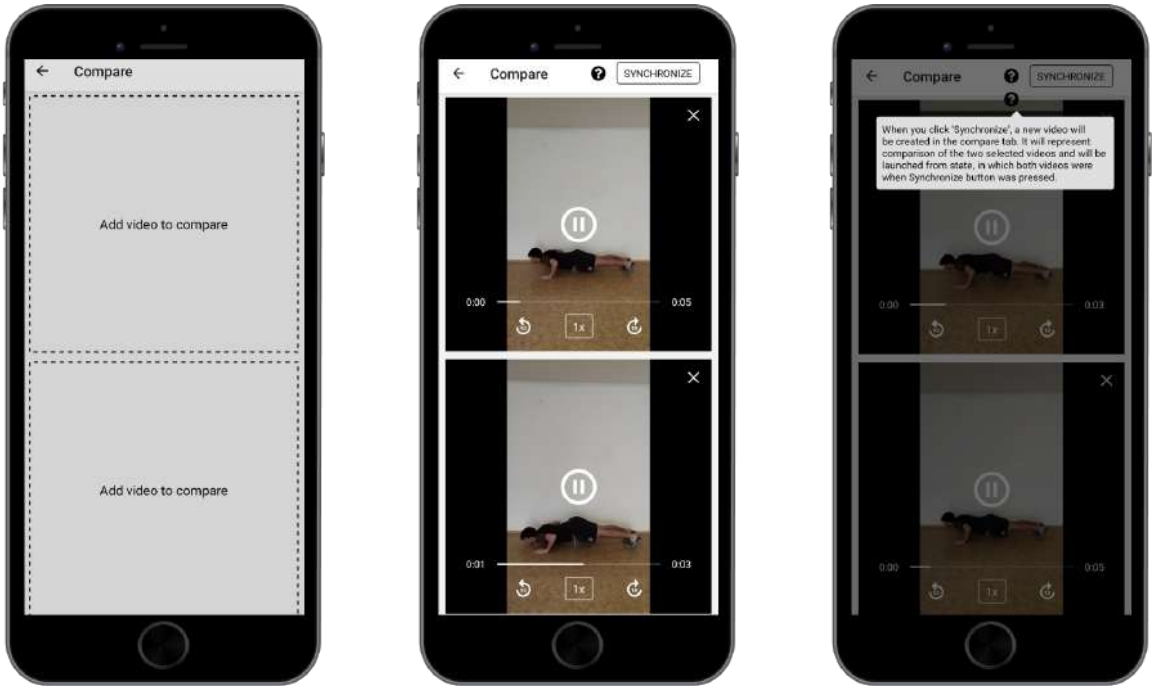
Obr. 5.1: Na obrázku vľavo je implementácia domovskej obrazovky športovca na tab obrazovke 'MyVideos'. Na obrázku v strede je možné vidieť menu tlačidla FAB po jeho stlačení. Na obrázku vpravo je vidieť implementáciu ukážky videí konkrétnej kategórie.

použitá knižnica 'AsyncStorage', ktorá umožňuje tieto dáta uložiť perzistentné na zariadení užívateľa.

Na tejto obrazovke je teda možné vidieť jednotlivé videá z porovnania v pároch, ktoré tvoria jednu položku. Každá z týchto položiek je vytvorená kombináciou náhľadov oboch zosynchronizovaných videí a textu 'VS', ktorý reprezentuje porovnanie týchto videí 5.3.

Ďalej je každá z týchto položiek interaktívna, a po ich stlačení sa spustí prehrávač videí, v ktorom sa nachádzajú tieto videa v rovnakom rozpoložení, v akom boli pri porovnaní videí. Je tam však niekoľko rozdielov. Obe videá zdieľajú tlačidlá pre ich ovládanie a sú ovládateľné iba jedným súborom tlačidiel, ktorý sa nachádza v spodnej časti obrazovky 5.3. Ďalším dôležitým prvkom je, že tieto videá budu spustené v bode, v ktorom ich užívateľ zastavil pred stlačením tlačidla 'SYNCHRONIZE', a teda už od spustenia budú zosynchonizované.

Na prehrávanie videí v porovnaní, ale taktiež už zosynchronizovaných videí som použil vlastný video prehrávač, ktorý je bližšie popísaný v následujúcej pod-sekcii 5.4.3.



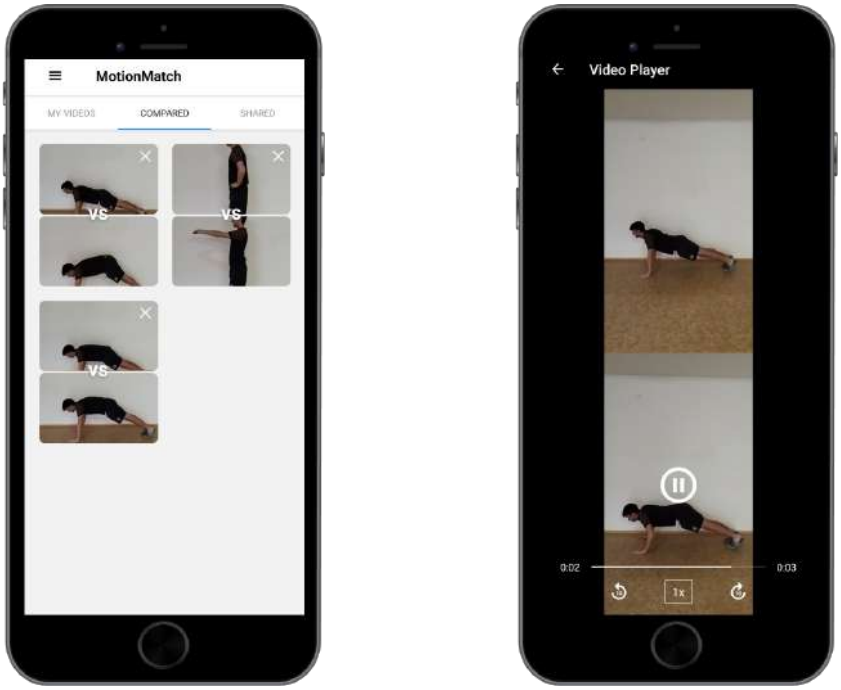
Obr. 5.2: Na prvom obrázku je možné vidieť obrazovku porovnania pred pridaním videí, v strede ukážku po pridania videa a na poslednom obrázku je vysvetlenie tlačidla 'SYNCHRONIZE'.

5.4.3 Video prehrávač

Pri ťuknutí na akékoľvek z videí bude užívateľ navigovaný na obrazovku video-prehrávača, kde sa video vybrané video spustí. Každá obrazovka v aplikácií, ktorá obsahuje video-prehrávač využíva na prehrávanie videa komponentu, ktorú som vytvoril. Oproti klasickému prehrávaču a jeho natívnym ovládaním, ktorý je poskytnutý knižnicou 'expo-av' to ma niekoľko výhod:

- **Prispôsobenie** – vytvorením vlastného prehrávača videa je možné získať väčšiu kontrolu nad používateľským rozhraním, správaním a funkciami prehrávača.
- **Ovládanie** – implementácia vlastného prehrávača umožňuje pridať funkcie, ktoré nemusia byť dostupné v natívnych ovládacích prvkoch prehrávača videa. V prípade mojej aplikácie je to napríklad ovládanie rýchlosti prehrávania, synchronizácia dvoch prehrávačov videa 5.3, alebo zobrazovanie a skrývanie ovládania podľa uváženia programátora. Tieto úpravy vylepšujú užívateľskú skúsenosť (UX) pri prehrávaní videí

Je však dôležité podotknúť, že napriek tomu že je na každej obrazovke, ktorá potrebuje prehrať video použitý rovnaký vlastný video-prehrávač, tak sa implementácia obrazoviek líši podľa potreby. Napríklad pri porovnaní videa bolo nutné vrámci obrazovky umiestniť dve prehrávače vertikálne vedľa seba, alebo vrámci obrazovky na prehrávanie videí z *tab* obrazovky 'MyVideos' sú pridané tlačidlá na porovnanie daného videa a jeho zdieľanie trénerovi 5.4. Videá je možné zdieľať iba v prípade, že si už športovec trénera zvolil. V opačnom prípade dostane užívateľ pri stlačený tohto tlačidla upozornenie, implementované pomocou React Native komponenty 'Alert'.



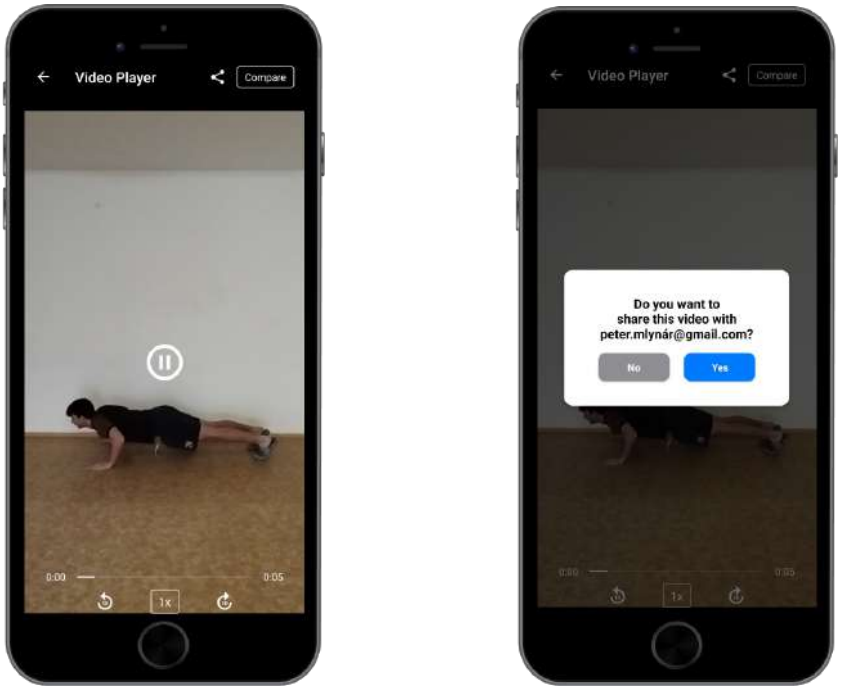
Obr. 5.3: Tab obrazovka 'Compared', ktorá obsahuje zosynchronizované videá a následne ukážka prehrania tejto synchronizácie vo video-prehrávači.

5.4.4 Prehrávač obrazovky 'MyVideos' a ukážka Modalu pri pokuse o odoslanie videa v prípade, že má užívateľ už zvoleného trénera.

Ako už bolo spomínané, v prípade, že chce užívateľ zdieľať video s trénerom, tak si nejakého najprv musí vybrať. Výber je implementovaný použitím komponenty 'Modal', pomocou ktorej sa užívateľovi po stlačení tlačidla 'Select Trainer' vylistujú všetci existujúci tréneri, zaregistrovaný v aplikácií. Po vybraní trénera sa zvolený tréner objaví na vrchole tohoto zoznamu, a bude vedľa neho výrazný zelený nápis, ktorý hovorí o tom, že daný tréner je aktuálne zvolený 5.5. Zároveň sa užívateľovi do jeho dokumentu v databáze pridá identifikátor zvoleného trénera pod políčkom 'trainerId'.

Po výbere trénera má užívateľ možnosť s ním zdieľať videá, a to dvoma spôsobmi. Jedným je podržanie akéhokoľvek videa na obrazovke 'MyVideos' a druhým spustenie jedného z týchto videí priamo v prehrávači 5.4 a ťuknutie na ikonu zdieľania. Oba tieto spôsoby užívateľovi vystavia 'Modal', ktorý slúži na potvrdenie tohoto zdieľania konkrétnemu trénerovi.

Po zdieľaní videa sa trénerovi toto video zobrazí pri konkrétnom športovcovi, ktorý s ním toto video zdieľal. Tréner ho následne môže ohodnotiť, a to tým spôsobom, že ho okomentuje a prípadne do neho niečo nakreslí. Komentáre sú po vytvorení trénerom uložené v databáze Firestore v dokumente konkrétneho videa, a obsahuje čas, text a obrázok, ktorý reprezentuje moment videa, v ktorom tréner video zastavil, a okomentoval a poskytuje užívateľovi vizuálnu predstavu k popisu trénera. Kresba je uložená rovnakým spôsobom vo formáte SVG. Okrem času obsahuje aj výšku a šírku videa na obrazovke zariadenia, na ktorom bola kresba vytvorená a taktiež dáta o čiarach kresby. Cieľom tohto spôsobu implementácie je zaistiť správne zobrazovanie kresby aj na zariadení s iným rozlíšením.



Obr. 5.4: Ukážka úprav po testovaní.

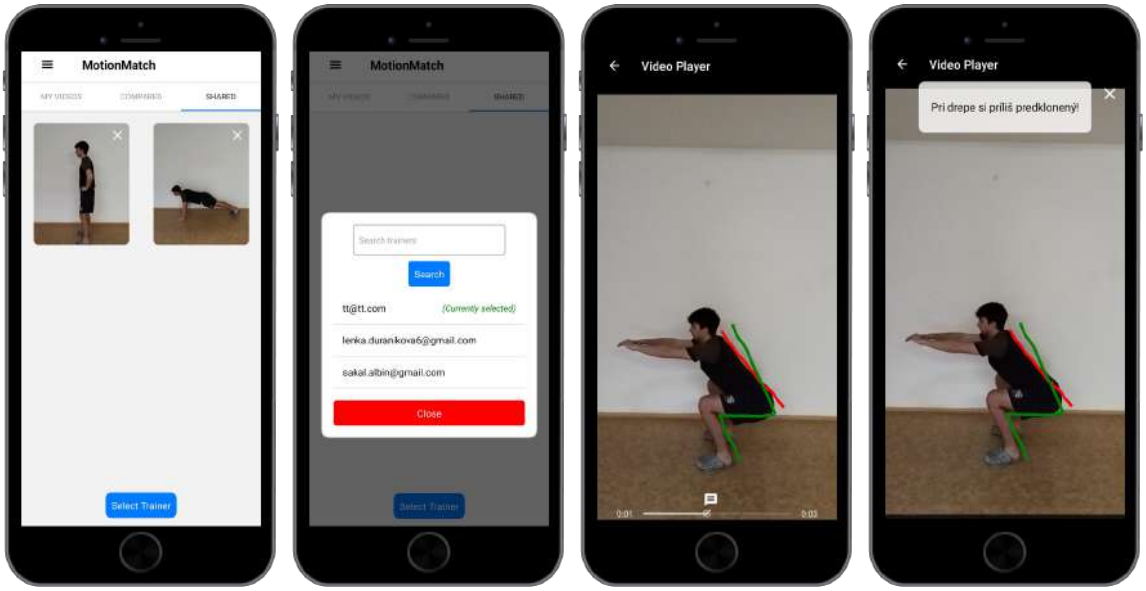
Po vyhodnotení videa trénerom si môže klient toto vyhodnotenie zobraziť v konkrétnom upravenom videu.

Komentár je zobrazený nad posuvníkom ako ikona, pomocou ktorého sa môže užívateľ pohybovať vo videu predne v čase, ktorý bol pri danom komentáre uložený v databázy. Po stlačení tohoto komentára vyskočí komponenta 'Modal', ktorá obsahuje text komentára a pod ním obrázok momentu, v ktorom bol komentár písaný 5.5.

Kresba je zobrazená pomocou knižnice `react-native-svg`, a implementovaná tak, aby sa zobrazila ako prekrytie cez video v momente, v ktorom tréner editoval video kreslením, čo je uložené v databáze vrámci kresby 5.5. Taktiež je rovnakým spôsobom ako komentár kresba označená ikonou, ktorá je zobrazená narozdiel od komentára priamo na posuvníku. Prekrytie sa zobrazí automaticky, a to sekundu pred, a po čase, v ktorom bolo vytvorené. To z dôvodu, že ak by toto prekrytie bolo zobrazené iba počas doby jedného rámca, užívateľ by mal problém si túto kresbu vôbec zobraziť, alebo zastaviť video v danom bode za účelom študovania hodnotenia trénera.

5.5 Tréner

Po prihlásení do aplikácie, uvidí tréner prehľad športovcov, ktorý si ho zvolili ako trénera. Zoznam týchto športovcov je získaný z databáze Firestore, a to tým spôsobom, že pomocou *auth().currentUser* funkcie z *Firebase Authentication* sú získané dáta o aktuálne prihlásenom užívateľovi, z čoho sa vyberie identifikátor tohoto užívateľa. Následne je v databáze prejdená celá kolekciu užívateľov, za účelom zistenia, ktorí z týchto užívateľov má vo svojom dokumente uložený identifikátor práve prihláseného trénera, v položke 'trainerId'. Po získaní zoznamu týchto užívateľov, sú v databáze nájdené všetky videá, ktoré im patria, ktoré užívatelia zdieľali s trénerom a užívateľ môže mať nanajvýš jedného trénera. Následne sú



Obr. 5.5: Ukážka 'Shared' tab obrazovky, určenej na prehľad zdieľaných videí, a voľbu trénera. Vpravo taktiež ukážka videa, ktoré bolo upravené trénerom.

pomocou komponenty 'Flatlist' zobrazené jednotlivé položky športovcov, a to zobrazením ich emailu 5.6.

V opačnom prípade, v ktorom si daného trénera zatiaľ nezvolil žiaden športovec, je trénerovi zobrazený iba text, ktorý túto situáciu vysvetľuje.

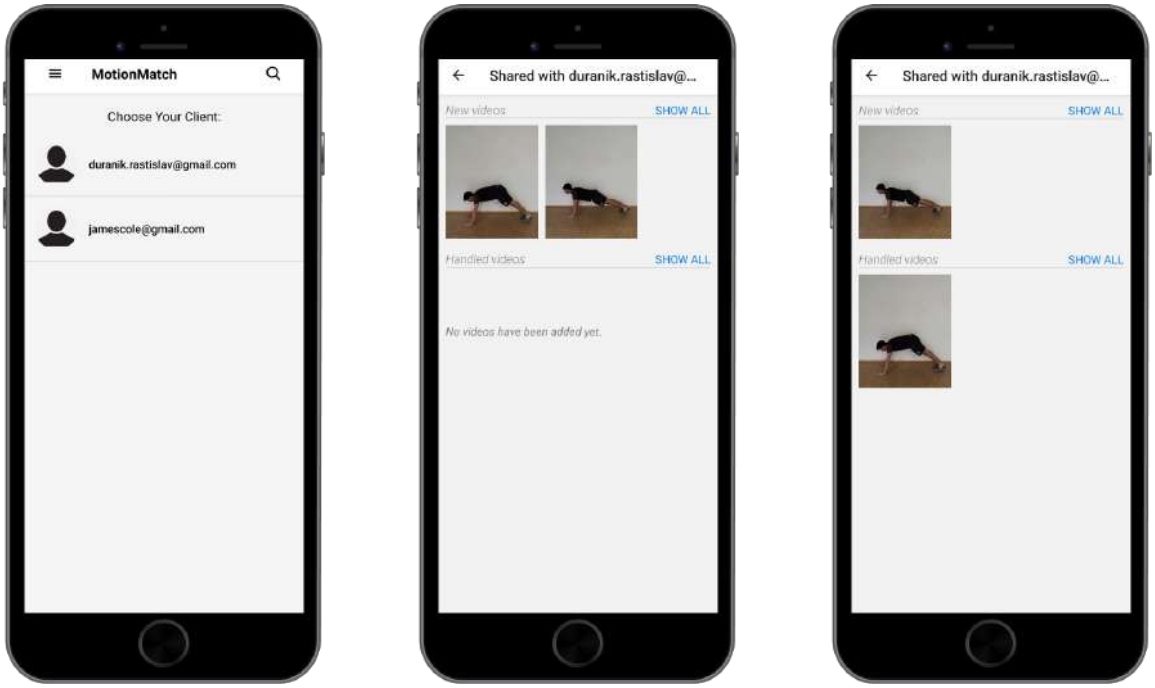
Na tejto obrazovke je taktiež možné vidieť ikonu lupy 5.6, ktorá umožňuje filtráciu športovcov v prípade väčšieho množstva, pre spríjemnenie užívateľskej skúsenosti (UX) trénera.

Na obrazovke sa taktiež nachádza 3-riadková ikona menu, pomocou ktorej sa tréner môže rovnakým spôsobom, ako športovec odhlásiť 5.4.

Jednotlivé položky vo *Flatliste* sú zabalené v komponente 'TouchableOpacity', a pri ťuknutí na konkrétneho klienta je tréner navigovaný na obrazovku zdieľaných videí so športovcom. Na tejto obrazovke sa zobrazia všetky videá, ktoré užívateľ zdieľal s trénerom. V prípade, že je to zatiaľ trénerom nevidené a neupravené video, tak bude umiestnené v kategórií 'New videos'. V opačnom prípade, a to že tréner dané video už videl a prípadne aj vyhodnotil, bude toto video zaradené do kategórie 'Handled videos', ktorá reprezentuje už spracované videá. To je možné vďaka ukladania *boolean* hodnoty vrámci každého videa, ktorá je pri pridaní videa do databázy vždy nastavená na *False*, a pri prezretí daného videa trénerom zmenená na *True*.

Videá v kategóriach je možné horizontálne posúvať, rovnako ako na obrazovke 'MyVideos' 5.4.1, a v komponente 'Flastlist' sú taktiež rovnakým spôsobom zobrazené. Rovnako platí, že po ťuknutí na tlačidlo 'SHOW ALL' budú zobrazené všetky videá danej kategórie na novej obrazovke. V prípade, že niesu pridané žiadne videá sa o tom vypíše vo vnútri kategórie textová správa 5.6.

Po stlačení akéhokoľvek z videí, je užívateľ navigovaný na obrazovku video-prehrávača trénera, kde je toto video možné ohodnotiť 5.5.1.



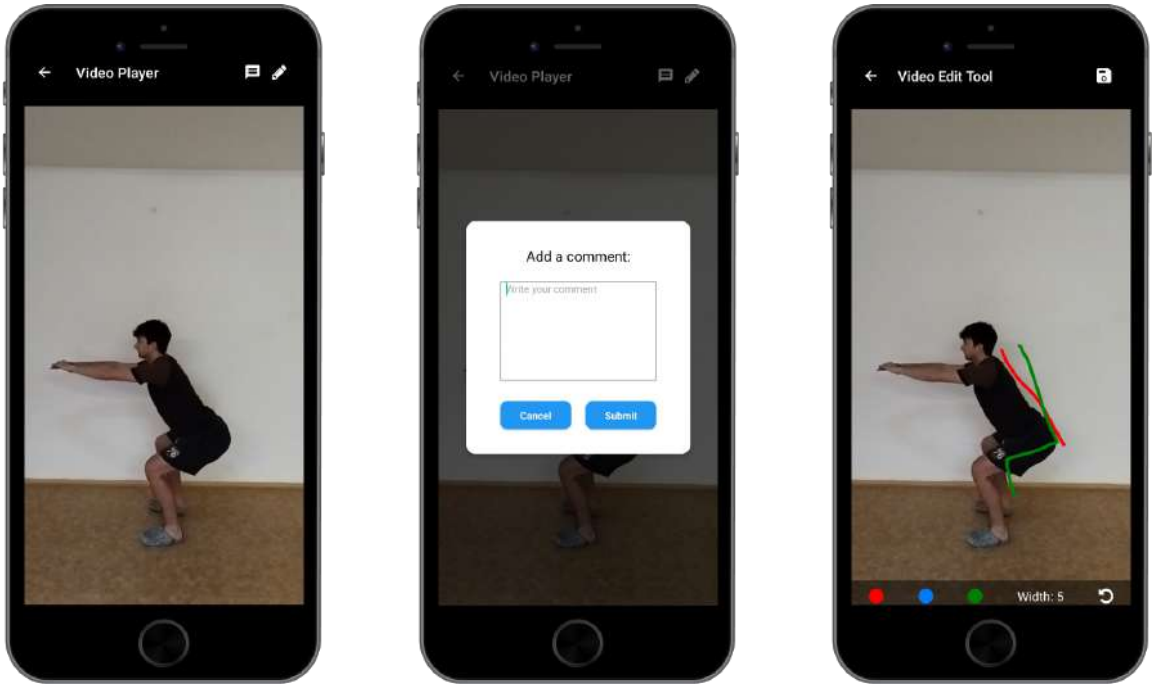
Obr. 5.6: Proces výberu športovca a prehľadom jeho videí z trénerovej perspektívy. Ukážka zaradenia videa do 'New videos' a 'Handled' kategórie pred a po upravení.

5.5.1 Editácia videa športovca

Na obrazovke videoprehrávača trénera je znova použitý vlastný videoprehrávač, avšak tentokrát sú pridané ikony komentáru a kreslenia v pravom hornom rohu 5.7, ktoré trénerovi slúžia na vyhodnotenie videa. V prípade, že chce tréner jednu z tých ikon stlačiť je nutné, aby najprv zastavil video, inak vyskočí okno vytvorené komponentou 'Alert', ktorá ho na toto upozorní. Tento spôsob implementácie zaistí, že tréner bude písať komentár v presom momente, v ktorom chce reagovať na konkrétny aspekt cvičenia športovca.

V prípade, že chce tréner video okomentovať a video je zastavené tak môže kliknúť na ikonu komentáru. Následne sa otvorí 'Modal', ktorý obsahuje priestor na vpísanie textu a tlačidlá pre potvrdenie, alebo zrušenie tohoto komentáru. Ak tréner komentár odošle, tak sa vytvorí v dokumente videa nový objekt komentára, ktorý obsahuje už spomínané položky, a to text, čas a snímku videa v momente písania komentáru. Po nahraní týchto údajov do databázy sa trénerovi zobrazí správa na vrchu obrazovky pomocou komponenty 'showMessage', prevzatej z knižnice `react-native-flash-message`, ktorá potvrdí úspešné pridanie komentáru k videu. V tomto momente je už športovec schopný vidieť tento komentár vo videu.

Za rovnakých okolností ako pri komentovaní môže tréner kresliť do videa. Pri stlačení ikony na kreslenie, je tréner navigovaný na obrazovku editácie, kde sa mu zobrazia nástroje na editáciu a snímka videa v čase, v ktorom tréner video zastavil 5.7. Týmto spôsobom môže tréner poskytnúť športovcovi presnú spätnú väzbu v presnom čase. Medzi nástroje na kreslene patrí voľba farieb, šírka čiary pri kreslení a možnosť zrušenia poslednej nakreslenej čiary. Kreslenie je implementované pomocou knižnice `react-native-svg`, komponenty 'PanResponder' a pomocných funkcií 'onPanResponderMove' a 'onPanResponderRelease', ktoré zaznámenávajú a zapisujú polohu jednotlivých bodov čiar nakreslených trénerom.



Obr. 5.7: Na obrázkoch je možné vidieť editáciu videa športovca, možnosť pridania komentáru alebo kreslenia.

Z každej nakreslenej čiary používateľom sa vytvorí prvok SVG cesty. *PanResponder* je zodpovedný za zachytenie dotyku používateľa a aktualizáciu aktuálnej cesty. Keď používateľ pohybuje prstom po obrazovke, vypočítajú sa relatívne súradnice X a Y a pridajú sa k údajom o ceste. Relatívne sú preto, že obrazovka zariadenia nemusí odpovedať ploche kreslenia. Keď používateľ zdvihne prst a teda ukončí kreslenie aktuálnej čiary, tak sa táto cesta sa pridá do poľa ciest.

Po ukončení kreslenia a stlačení tlačidla pre uloženie je toto pole ciest reprezentujúce každú nakreslenú čiaru uložené do dokumentu videa, s originálnou výškou a šírkou videa a časom, pre zaistenie presného zobrazenia na rôznych zariadeniach.

Dôvodom , prečo som použil formát SVG je, že poskytuje škálovateľný spôsob reprezentácie vektorovej kresby. To umožňuje jednoduchú zmenu veľkosti kresby a manipuláciu s ňou bez straty kvality. Okrem toho sú SVG dobre podporované na rôznych platformách, čo bolo možné uplatniť v prípade, že táto aplikácia bude v budúcnosti bežať aj na iOS zariadeniach.

Kapitola 6

Testovanie

Pre distribúciu aplikácie za účelom testovania som použil inštalačný balík zvaný APK (Android Package). APK je formát súboru, ktorý sa využíva na distribúciu a inštaláciu mobilných aplikácií na platforme Android. Ak je na implementáciu aplikácie použité Expo, nie je nutné zaoberať sa generovaním digitálnych podpisov alebo konfiguráciou súborov Gradle, pretože Expo tieto procesy zjednodušuje. Pri používaní 'čistého' React Native by to však potrebné bolo.

Po ukončení vývoja aplikácie, som vytvoril APK súbor pomocou príkazu `eas build -platform` android. Expo potom automaticky vytvorilo inštalačný balíček a poskytlo mi inštalačný odkaz, ktorý som následne mohol zdieľať s testovacími subjektmi. Inštalačný odkaz, ako napríklad 'https://expo.dev/artifacts/eas/vsswdwMDEzHfjUxb6uYP83.apk', je uložený na Expo serveroch. Následne už bolo jednoduché aplikáciu distribuovať, a to za použitia sociálnych sietí a emailu. Takýto spôsob distribúcie zabezpečil jednoduchý prístup pre testerov, keďže sa im spustilo sťahovanie aplikácie do telefónu len jedným stlačením odkazu. Nevýhodou tohto prístupu je však to, že aplikácie, ktoré nie sú stiahnuté z Google Play potrebujú opakované potvrdenia pri stiahnutí, keďže zariadenia testerov sa ich pokúšajú varovať pred neznámymi aplikáciami.

Dôvodom, prečo som aplikáciu neumiestnil na Google Play, aj napriek tomu, že funkčná a spĺňa jej účel bolo, že je aplikácia stále iba v testovacom štádiu. Pri testovaní bolo nájdených mnoho nedostatkov, ktoré sú popísane v sekcii testovania užívateľmi 6.1.

6.1 Testovanie užívateľmi

Testovanie prebiehalo buď osobne alebo online, však nebolo tak možné učiniť pri všetkých testovaných subjektoch, preto je tomu testovací scenár prispôsobený. Pri testovaní som sledoval, ako rôzny užívatelia interagovali s aplikáciou pri plnení úloh z testovacieho scenáru, ktorý vyzeral následovne:

1. Stiahni si aplikáciu MotionMatch z tohoto odkazu a otvor ju. Následne nájdi ďalšiu osobu, ktorá by ti pomohla pri otestovaní aplikácie, keďže aplikácia má 2 typy užívateľov, ktorý spolu interagujú.

   V prípade, že niekoho takého nájdeš, tak sa obaja zaregistrujte. Jeden ako užívateľ (user) a druhý ako tréner (trainer).

   Ak nikoho nenájdeš, tak najprv sa zaregistruj ako tréner, potom sa odhlás a zaregistruj sa ako užívateľ.

Do odpovedí jednotlivých otázok formulára napíš či bolo všetko dostatočne prehľadné intuitívne/prípadne čo by si pri danom kroku zmenil. Ak sa vyskytne chyba tak ju popíš, a taktiež vysvetli ako si sa k nej dopracoval. ,

2. **Športovec** - predstav si situáciu, v ktorej si športovec, a dávnejšie si si nahral video, v ktorom vykonávaš istý cvik. Dnes si sa natočil znova, a chceš zistiť, či sa tvoja technika zlepšila. Porovnaj tieto videá.

3. **Športovec** – zvoľ si trénera a zdieľaj trénerovi svoje video, ktoré chceš aby ohodnotil (nie je možné zdieľať synchronizáciu videí).

4. **Tréner(trainer)** – prihlás sa do svojho účtu a choď užívateľovi ohodnotiť video. V istý moment si všimneš, že vykonáva cvik technicky nesprávne, takže mu k tomu napíšeš komentár. Následne si všimneš, že má napr. pri cvičení kliku krivý chrbát, a teda mu nakreslíš čiaru, aby to videl.

5. **Športovec** – otvoríš si video, ktoré si zdieľal trénerovi a všimneš si, že ti tréner ohodnotil video. Pozri si čo ti tréner napísal, prípadne nakreslil.

6.2 Výsledky testovania a budúci vývoj

V prvej iterácii testovania bolo odhalených mnoho chýb, ktoré sa týkali aj implementácie, aj intuitivity. Medzi tieto chyby patrilo napríklad to, že pri prihlasovaní a registrácii pôvodne nebolo možné ukázať heslo počas jeho písania, zlyhal jeden zo spôsobov zdieľania videa a taktiež pri porovnávaní väčších videí, čo znamená 10MB a viac aplikácia zamrzla, prípadne sa vypla. Po oprave týchto prvotných chýb, ktoré bránili riadnemu otestovaniu užívateľmi som začal s druhou iteráciou testovania, ktorá prebehla relatívne hladko, však aj napriek množstvu kladných reakcií od užívateľov sa našli nedostatky. Tieto nedostatky boli však zaradené do budúceho vývoja, keďže priamo neovplyvňovali funkcionalitu aplikácie. Patrí medzi ne napríklad:

- Nie je priamo možné meniť kategórie videí po pridaní.
- Pri kreslení je v spodnej časti obrazovky mierne nevycentrovaný štetec.
- Tréner nie je schopný vidieť a upraviť svoje vyhodnotenie pre konkrétne video športovca po vytvorení a odoslaní.
- Nie je možné s trénerom zdieľať synchronizáciu videí a teda ju tréner nemôže ani vyhodnotiť
- Tréner nedostane upozornenie, ak s ním športovec zdieľa video, a ani keď si ho užívateľ zvolí za trénera. Takisto užívateľ nedostane upozornenie po tom, ako mu tréner vyhodnotí video.

Ako je možné z týchto bodov vidieť, aplikáciu MotionMatch je možné vylepšiť vo viacerých smeroch. Okrem odstránenia týchto nedostatkov je plánované túto aplikáciu vylepšiť, a to napríklad pridaním možnosti porovnávať viac ako dve videá, alebo vylepšením komunikácie medzi trénerom a športovcom.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo vytvorenie aplikácie pre každého, kto si chce vylepšiť techniku cvičenia, rovnako pre športovcov, ako aj samoukov cvičiacich doma. Aplikácia umožňuje porovnávanie videí, zasielanie videí trénerovi a možnosť získať detailnú spätnú väzbu k technike cviku. Vďaka tejto kontrole technickej správnosti sa môže športovec zlepšovať vo svojich cvičeniach a dlhodobo predchádzať možným zdravotným rizikám spojeným s chybnou záťažou a pohybmi.

Práca začala výskumom vývoja mobilných aplikácií, kde som si naštudoval informácie o existujúcich prístupoch vývoja, jazykoch, frameworkoch a knižnicách pre vývoj. Následoval prieskum existujúcich aplikácií a analýza ich výhod a nevýhod. Prieskum bol veľmi nápomocný, keďže som ním získal inšpiráciu a prehľad o užívateľskom rozhraní daných aplikácii. Ďalej bol vytvorený a otestovaný prvý návrh, ktorý mal nesmierne množstvo nedostatkov, avšak po iteratívnom testovaní a následne znova navrhovaní sa návrh dostal do užívateľsky prijateľnej podoby. Nasledovala implementácia a otestovanie.

Výsledná aplikácia umožňuje pridávanie videí do aplikácie, ich kategorizáciu a vzájomné porovnávanie ako aj porovnávanie s referenčnými videami. Užívateľ má možnosť zvoliť si svojho trénera, s ktorým môže zdieľať svoje videá a získavať spätnú väzbu pomocou časovo presných komentárov a editácií kreslením.

Tréner si po prihlásení do aplikácie vyberá klienta ktorého plánuje hodnotiť. Následne je jeho slovná a grafická spätná väzba odoslaná klientovi, a klientovo video sa presúva do kategórie vybavených videí.

Aplikácia bola testovaná iteratívne na skupinách 5-10 testerov. Testerom bol distribuovaný dotazník s odkazom na stiahnutie aplikácie. Niektoré testovania prebiehali pod mojím osobným dohľadom, čo mi prinieslo veľmi prínosné postrehy týkajúce intuitivnosti funkcií a ich použitia vrámci zadaných úloh.Opravou prvotných chýb odhalených v prvej fáze testovania bola zabezpečená funkcionalita aplikácie a ďalšie odhalené nedostatky.

Do budúceho vývoja bude implementované napríklad možnosť zdieľania synchronizácií s trénerom a jej vyhodnocovanie, pridania upozornení o novom videu pre trénera ako aj upozornenie o vyhodnotení pre klienta, a spätný náhľad na vyhodnotenie odoslané klientovi.

Literatúra

[1] Ahmad, A., Li, K., Feng, C., Asim, S. M., Yousif, A. et al. An Empirical Study of Investigating Mobile Applications Development Challenges. *IEEE Access*. 2018, zv. 6, s. 17711–17728, [cit. 2023-4-6]. DOI: 10.1109/ACCESS.2018.2818724.

[2] Ardito, L., Coppola, R., Malnati, G. a Torchiano, M. Effectiveness of Kotlin vs. Java in android app development tasks. *Information and Software Technology*. 2020, zv. 127, s. 106374, [cit. 2023-4-14]. DOI: https://doi.org/10.1016/j.infsof.2020.106374. ISSN 0950-5849. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0950584920301439.

[3] Bose, S. A COMPARATIVE STUDY: JAVA VS KOTLIN PROGRAMMING IN ANDROID APPLICATION DEVELOPMENT. *International Journal of Advanced Research in Computer Science*. Jún 2018, zv. 9, s. 41–45, [cit. 2023-4-14]. DOI: 10.26483/ijarcs.v9i3.5978.

[4] C G, T. a Devi, A. A Study and Overview of the Mobile App Development Industry. *International Journal of Applied Engineering and Management Letters*. Jún 2021, s. 115–130, [cit. 2023-4-5]. DOI: 10.47992/IJAEML.2581.7000.0097.

[5] Chauhan, S. *Understanding Xamarin iOS - Build Native iOS App* [online]. dotnettricks, august 2022 [cit. 2023-4-10]. Dostupné z: https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-ios-build-native-ios-app.

[6] codingmedic. *The Virtual DOM* [online]. codingmedic, november 2020 [cit. 2023-4-22]. Dostupné z: https://codingmedic.wordpress.com/2020/11/10/the-virtual-dom/.

[7] Danielsson, W. React Native application development : A comparison between native Android and React Native. In:. 2016 [cit. 2023-4-21].

[8] David, M. *Mobile application development* [online]. TechTarget, marec 2021 [cit. 2023-4-7]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/mobile-application-development.

[9] Eisenman, B. *Writing Cross-Platform Apps with React Native* [online]. infoQ, február 2016 [cit. 2023-4-20]. Dostupné z: https://www.infoq.com/articles/react-native-introduction/.

[10] El Kassas, W. S., Abdullah, B. A., Yousef, A. H. a Wahba, A. M. Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal*. 2017, zv. 8, č. 2, s. 163–190, [cit. 2023-4-20]. DOI:

https://doi.org/10.1016/j.asej.2015.08.004. ISSN 2090-4479. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2090447915001276.

[11] Goadrich, M. H. a Rogers, M. P. Smart Smartphone Development: IOS versus Android. In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2011, s. 607–612 [cit. 2023-4-9]. SIGCSE '11. DOI: 10.1145/1953163.1953330. ISBN 9781450305006. Dostupné z: https://doi.org/10.1145/1953163.1953330.

[12] Google. *Buttons: floating action button*. 2023 [cit. 2023-5-2]. Dostupné z: https://m2.material.io/components/buttons-floating-action-button.

[13] Grummitt, C. *IOS Development with Swift*. 1. vyd. Apress, 2017 [cit. 2023-4-14]. ISBN 9781638354031.

[14] Hutri, H. *Comparison of React Native and Expo*. 2023. [cit. 2023-4-22]. Diplomová práca. Lappeenranta–Lahti University of Technology LUT.

[15] Khandeparkar, A., Gupta, R. a B.sindhya. Article: An Introduction to Hybrid Platform Mobile Application Development. *International Journal of Computer Applications*. May 2015, zv. 118, č. 15, s. 31–33, [cit. 2023-4-6]. DOI: 10.5120/20824-3463.

[16] Khawas, C. a Shah, P. Application of Firebase in Android App Development-A Study. *International Journal of Computer Applications*. Jún 2018, zv. 179, s. 49–53, [cit. 2023-4-24]. DOI: 10.5120/ijca2018917200.

[17] Kuitunen, M. *Cross-Platform Mobile Application Development with React Native*. 2019. [cit. 2023-4-20]. Diplomová práca. Tampere University of Technology.

[18] Laurence, P., Hinchman Dominguez, A., Meike, G. a Dunn, M. *Programming Android with Kotlin*. 1. vyd. O'Reilly Media, Incorporated, 2021, 2021 [cit. 2023-4-7]. ISBN 9781492063001.

[19] Liu, C., Zhu, Q., Holroyd, K. A. a Seng, E. K. Status and trends of mobile-health applications for iOS devices: A developer's perspective. *Journal of Systems and Software*. 2011, zv. 84, č. 11, s. 2022–2033, [cit. 2023-4-13]. DOI: https://doi.org/10.1016/j.jss.2011.06.049. ISSN 0164-1212. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0164121211001610.

[20] Ole Henry Halvorsen, D. C. *OS X and iOS Kernel Programming*. 1. vyd. Apress Berkeley, CA, 2011 [cit. 2023-4-14]. ISBN 978-1-4302-3536-1.

[21] Pinto, C. M. a Coutinho, C. From Native to Cross-platform Hybrid Development. In: *2018 International Conference on Intelligent Systems (IS)*. 2018, s. 669–676. DOI: 10.1109/IS.2018.8710545.

[22] S, A. K. *Mastering Firebase for Android Development*. 1. vyd. Packt Publishing, 2018 [cit. 2023-4-22]. ISBN 9781788624251.

[23] Sarkar, A., Goyal, A., Hicks, D., Sarkar, D. a Hazra, S. Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems. In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile,*

Analytics and Cloud) (I-SMAC). 2019, s. 73–79 [cit. 2023-4-9]. DOI: 10.1109/I-SMAC47947.2019.9032440.

[24] Shah, K., Sinha, H. a Mishra, P. Analysis of Cross-Platform Mobile App Development Tools. In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. 2019, s. 1–7 [cit. 2023-4-15]. DOI: 10.1109/I2CT45611.2019.9033872.

[25] Sheridan, A. *App Review: Ubersense Keeps It Simple For Athletes Looking To Improve Skills* [online]. sportsbusinessjournal, január 2014 [cit. 2023-4-26]. Dostupné z: https://www.sportsbusinessjournal.com/Daily/Issues/2014/07/01/Media/App-Review.

[26] Tram, M. *Firebase*. 2019. [cit. 2023-4-26]. Diplomová práca. CENTRIA UNIVERSITY OF APPLIED SCIENCES.

[27] Wu, W. *React Native vs Flutter, cross-platform mobile application frameworks*. 2018. [cit. 2023-4-22]. Diplomová práca. Metropolia University of Applied Sciences.

Príloha A

Plakát

Obr. A.1: Plagát aplikácie *MotionMatch*.