ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Návrh mobilní aplikace pro sportovní centrum

Ondřej Tomek
Obor: Otevřená informatika - Software

Květen 2024
https://gitlab.fel.cvut.cz/tomekon2/choketopus-gym
Vedoucí práce: doc. Ing. Ivan Jelínek, CSc.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tomek** Jméno: **Ondřej** Osobní číslo: **507669**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro tréninkové centrum

Název bakalářské práce anglicky:

Mobile app for training center

Pokyny pro vypracování:

Navrhněte a implementujte mobilní aplikaci pro podporu klientů tréninkového centra Choketopus Gym. Aplikace umožní:
informování o základních funkcích gymu,
sledování aktualit gymu,
sledování výukových videí
docházka členů,
hodnocení trenérem a tréninkových plánů a jejich vyhodnocování
připojení do ligy (turnaje) zákazníků Gym
sledování momentálního stavu ligy
zadávání výsledků zápasu
Proveďte rešerši již existujících řešení. Specifikujte a upřesněte požadavky na aplikaci. Do aplikace se budou moci přihlašovat klienti, trenéři centra a administrátor. Navrhněte vhodné implementační prostředí. Navrhněte strukturu a rozsah aplikace pro implementaci. Aplikaci implementujte. Navrhněte způsob testování aplikace, aplikaci otestujte a testy vyhodnoťte.

Seznam doporučené literatury:

Flutter dokumentace, https://flutter.dev/
Rap Payne, Beginning App Development with Flutter, Apress, 2019
Udemy Flutter Course, https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Ivan Jelínek, CSc. kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

doc. Ing. Ivan Jelínek, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování / Prohlášení

Tímto bych chtěl poděkovat panu Stehlíkovi. Také chci poděkovat Ivanu Jelínkovi za pomoc při utváření struktury práce a společné konzultace v průběhu práce na bakalářské práci.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. 5. 2024

.....

Abstrakt / Abstract

Tento dokument představuje postup vývoje mobilní aplikace pro pražskou posilovnu Choketopus Gym za použití novějšího frameworku Flutter. Aplikace umožňuje sledování aktualit a základních informací o sportovním centru, sledovat výuková videa, přijímat docházku a nakonec také připojení do ligy, která je uzavřená pro zákazníky centra. V první části práce je popsána technologie Flutter, další použité technologie a design aplikace. V druhé části jsou implementační detaily, konečný výsledek práce a nakonec testování aplikace. Nakonec je popsána budoucnost aplikace.

Klíčová slova: Flutter; posilovna; mobil; sport; Firebase.

This document introduces step by step implementation of mobile application for Choketopus Gym based in Prague with usage of newly developed framework Flutter. Application allows viewing newsletters and basic information about the sports center, watching lessons, accepting attendance and lastly also joining league, that is private just for clients of the center. First part of my work describes Flutter technology, other used technologies and application design. Second part is the implementation details, final results of my work and lastly testing. At the end is described the future of my work.

Keywords: Flutter; gym; mobile; sport; Firebase.

Title translation: Development of mobile application for Sport center

Obsah /

Kapitola 1
Úvod

V současnosti se vše kolem nás pohybuje okolo technologií,které usnadňují život běžným uživatelům. Urychlují, zpříjemňují nebo nám také automatizují základní všední činnosti. Propojují nás, informují a umožňují se rozvíjet. Téma této bakalářské práce je funkční aplikace, která umožní a usnadní přístup ke sportovnímu centru a zároveň zlepší komunikaci ze strany centra k jejím klientům. Aplikace umožní sledování aktualit a základních informací o sportovním centru, sledovat výuková videa, přijímat docházku,připojení do ligy, apod.

Společnost Choketopus Gym, nabízí sportovní aktivity zaměřené na bojová umění a sebeobranu. Jedná se však o poměrně rozšířenou firmu, jenž se nachází již na čtyřech místech v Praze. Není pochyb o tom, že společnosti prospěje lepší komunikace s klienty a umožnění klientům se vzdělávat v rámci bojových umění i z pohodlí domova.

Mobilní aplikace se jeví jako jasná volba. Přeci jenom, kdo v dnešní době nevlastní mobilní zařízení? Vývoj pro mobilní zařízení mě okouzlil už při mé první interakci s nativním vývojem a chci poznat i druhou stranu mince, tedy Cross-platform vývoj za použití již vytvořených nástrojů (frameworku) Flutter.

1.1 Struktura práce

Práce je rozdělená do několika částí, které odpovídají postupnému vývoji aplikace.

V první části je představení s technologiemi, které jsou v aplikace použité. Framework Flutter, kterému jsem věnoval celou samostatnou kapitolu, vytváří kostru celé aplikace. Následně pak je popsán návrh aplikace.

Nyní se dostáváme do části, která souvisí se samotnou implementací aplikace. Prvně přijde na řadu popis implementace, problémy na které se při vypracování práce narazilo a jejich vyřešení. Poté seznámení se způsobem testování aplikace, zpětná vazba a návrh na její úpravy. Nakonec je na řadě závěrečná podoba aplikace, kde ukáži konečný stav aplikace.

Na úplný závěr je popsána budoucnost aplikace, zhodnocení mé práce, co se povedlo a co naopak ne.

1.2 Seznámení s funkčními požadavky

Podle domluvy se zadavatelem a vedoucím práce jsme se dohodli na několika službách aplikace, jenž má poskytnout, jak uživateli, tak trenérovi. Já ovšem přidal i roli admina neboť je dobré ho mít, už jen z důvodu možnosti přidávat a odebírat trenérské účty z aplikace.

1.2.1 Uživatel

Uživatel se bude do aplikace přihlašovat (registrovat). Každý uživatel může zadávat docházku a sledovat graf své přítomnosti na profilu. Uživatel si také může spouštět výukové kurzy na jednotlivé sporty a také od určitých trenérů. Kurzy jsou zatím pouze zdarma, ale v budoucnu bude možnost i placených. Také je přání založit projekt „Liga“, ve které by se klienti centra mohli dohodnout na společných zápasech.

Zápasníci se vyzvou a po zápase jeden z nich zadá výsledek a ten druhý ho může potvrdit. Po přijmutí výsledku se výherci připíší body v tabulce. Z mé strany mi také dává smysl, aby se uživatel aplikace mohl podívat na rozvrh tělocvičen, sledovat dění (zrušení tréninku, otevření nového termínu, apod.), přečíst si informace o trenérech a prohlédnout si svůj vlastní profil. Změna profilu a změna hesla je samozřejmostí.

1.2.2 Trenér

Trenér se přihlašuje stejně jako běžný uživatel jen je jeho účet označen jako trenérský adminem. Jeho UI aplikace bude odlišné, ale jen v malých detailech. Může posílat/odstraňovat novinky centra uživatelům, vytvářet/odstraňovat výukové kurzy a nahrávat/mazat v nich videa. Mimo jiné také trenéři se smějí připojit do ligy.

1.2.3 Admin

Admin má speciální účet. Má všechny vlastnosti již zmíněné u trenéra a běžného uživatele. Mimo jiné může přidávat a odstraňovat trenéry. Samozřejmě i odlišné UI, které nelze přepínat.

Kapitola 2
Použité technologie pro vývoj

Technologie jsem vybíral na základě svých zkušeností nových vývojových trendech. Na UI návrh použiji Figmu, kterou jsem používal v předmětu „Principy tvorby mobilních aplikací“. Jako textový editor jsem vybral VSCode, neboť se mi na něm povedlo zprovoznit emulátory a mám s ním dobré zkušenosti. A na databázi použiji cloud službu Firebase od Google, jenž poskytuje služby zdarma pro malé projekty.

2.1 Figma

Jedná se o webovou aplikaci, která poskytuje služby pro vývoj uživatelských rozhraní softwarových aplikací nehledě jaký framework nebo jazyk se použije v implementaci. Implementace samozřejmě bude mít odlišnosti oproti návrhu ve Figmě v závislosti na časové náročnosti naprogramování a mých zkušenostech. Ovšem budu se snažit držet původního Figma designu. Nyní představím jedny z hlavních výhod používání Figmi. [1]

2.1.1 Spolupráce

I když se jedná o službu pro mě nadbytečnou v tomto projektu, je nutné ji zmínit, neboť se jedná o jednu z hlavních výhod Figmy. Tou výhodou je schopnost týmů pracovat na jednom projektu současně v reálném čase. Vývojáři a designéři mohou spolupracovat na různých částech projektu, vidět okamžité změny včetně viditelnosti kurzorů jednotlivých uživatelů. Psaní komentářů a možná konzultace mezi projektanty pomáhá urychlit konečný návrh.

2.1.2 Responzivita

Figma usnadňuje vytváření responzivních designů, které se přizpůsobují různým zařízením a obrazovkám. Uživatelé mohou definovat varianty pro různé velikosti obrazovek a vytvářet interakce, které reagují na uživatelské vstupy. Simuluje se vlastně ovládání opravdové aplikace se statickými daty na obrazovkách.

2.1.3 Veřejné komponenty

Figma umožňuje vytvářet a sdílet komponenty, což jsou znovupoužitelné prvky designu. Tímto způsobem lze snadno udržovat konzistenci v designu a usnadňuje aktualizace v celém projektu. Stylům lze přiřazovat proměnné, což zjednodušuje úpravy designu a udržuje konzistenci barev, fontů a dalších prvků. Tyto komponenty lze pak také sdílet veřejně zdarma nebo dokonce prodávat.

2.2 Visual Studio Code

Visual Studio Code je svobodný a open-source kódový (textový) editor vyvinutý společností Microsoft pro vývoj software. Je navržen tak, aby byl lehký, rychlý a nabízel

rozsáhlé možnosti rozšíření, což ho činí populárním nástrojem pro vývojáře na různých platformách. S propojením k XCode na MacOS je možnost spouštět emulátor iOS a propojením k Android Studio také Android rovnou z VSCode. [2]

2.3 Firebase

Firebase je komplexní platforma poskytovaná společností Google, která nabízí širokou škálu nástrojů a služeb pro vývoj mobilních a webových aplikací. Jedná se o cloudovou platformu, která umožňuje vývojářům rychle a efektivně budovat, testovat a nasazovat aplikace s využitím škálovatelné infrastruktury Google.

Firebase poskytuje komplexní řešení pro vývoj moderních aplikací s vysokou dostupností a bezpečností, což z ní činí oblíbenou volbu mezi vývojáři po celém světě. Zde si přiblížíme několik klíčových aspektů Firebase a jak přispívají k celkovému procesu vývoje aplikací. [3]

2.3.1 Realtime Database

Realtime Database (databáze v realném čase) je jednou z hlavních funkcí Firebase, která umožňuje vývojářům ukládat a synchronizovat data mezi klienty a serverem v reálném čase. To znamená, že jakmile jsou data změněna na jednom zařízení, tyto změny se okamžitě propagují na všechna ostatní připojená zařízení.

2.3.2 Firebase Authentication

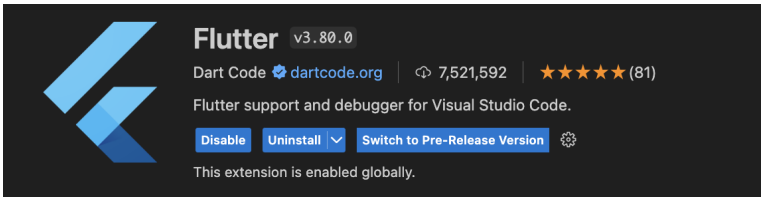
Firebase Authentication (ověření) poskytuje jednoduché a bezpečné řešení pro správu uživatelských účtů a ověřování identity ve vaší aplikaci. Podporuje různé metody ověření, jako jsou e-maily a hesla, sociální média nebo telefonní čísla.

2.4 Emulátory a mobilní zařízení

V práci na mobilní aplikaci se samozřejmě vyskytuje potřeba testovat, jak na emulátorech tak na opravdových zařízeních. Zajistit tyto potřeby je ovšem větší oříšek než se na první pohled zdá. Je potřeba nakonfigurovat jak vaše zařízení s operačním systémem (Windows, Mac, ...), tak samotné mobilní zařízení. Informace o nastavení prostředí jsem čerpal z knihy „Beginning App Development with Flutter“ [4]

2.4.1 Nastavení IDE

Je potřeba zmínit, že i přes to, že používám VSCode jako IDE, není dělané exklusivně pro Flutter vývoj. Proto je nutné nainstalovat vývojářské nástroje pro Flutter framework. Tyto nástroje pomáhají vývojáři debuggovat kód, prohlížet logy a snadno propojit a přepínat emulátory. Pro každé IDE se vývojářské nástroje instalují odlišně. Pro VSCode je to ve „View->Extensions“ tam pak vyhledáte „Flutter“ a kliknete nainstalovat. Tak jednoduché to je.

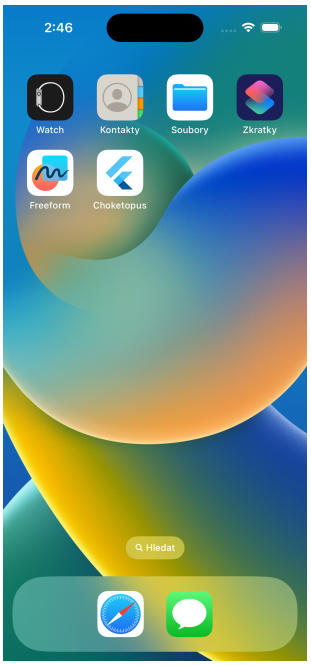


Obrázek 2.1. Vývojářské nástroje pro Flutter ve VSCode

2.4.2 Nastavení emulátorů

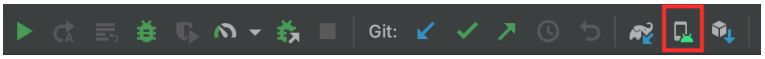
Nyní je tu ta obtížná a nepříjemná část. Apple jako společnost se snaží mít uzavřený systém a tak v určitém směru háže vývojářům pro iOS klacky pod nohy. Jedná se o omezení vývoje pro iOS pouze s Apple zařízení (tj. pouze z MacBooku nebo iMac). Já naštěstí vlastním MacBook Air a tak mohu vyvíjet také pro iOS. Teď si ukážeme, jak na to.

Prvně je potřeba mít nainstalované XCode, které je vyvíjeno společností Apple jako Nativní IDE pro iOS zařízení. Toto prostředí obsahuje nástroje pro používání emulátoru na Mac zařízeních. Lze ho nainstalovat na App Storu. Po instalaci musíme spustit XCode a přijmout podmínky. Přijmutím podmínek se zobrazí možnost nainstalovat simulátor, který se bude nalézat v tomto repozitáři „/Applications/Xcode.app/Contents/Developer/Applications“. Po otevření simulátoru se zapne emulátor iOS telefonu. Tento emulátor se automaticky spustí po spuštění aplikace ve VSCode.

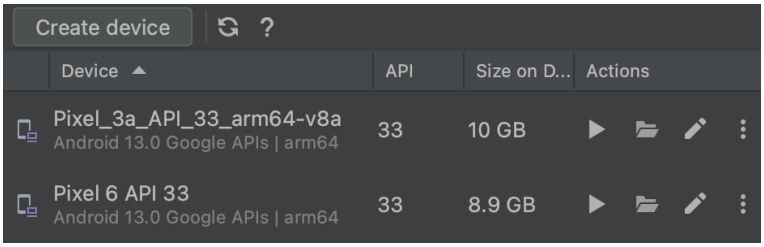


Obrázek 2.2. Příklad iOS simulátoru (iPhone15 Pro Max, iOS 17.2)

Pro vytvoření Android emulátoru je potřeba nejdříve nainstalovat Android Studio. Android Studio lze nainstalovat z oficiálních stránek. Poté otevřete konfigurace virtuálních mobilních zařízení.

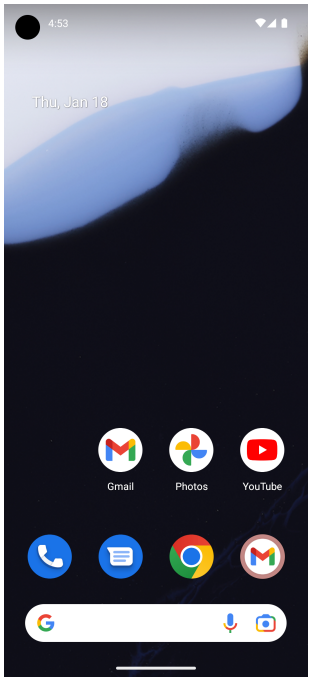


Obrázek 2.3. Ikona pro konfiguraci mobilních zařízení



Obrázek 2.4. Seznam všech android emulátorů

Vytvoříme zařízení. Následně vybereme, jaký telefon chceme a jeho verzi Androidu (já vybral Pixel6, Tiramisu). A tím máme hotovo. Nyní otevřeme VSCode a vpravo dole zvolíme námi vytvořený emulátor.



Obrázek 2.5. Příklad Android simulátoru (Pixel6, Tiramisu)

2.4.3 Nastavení mobilních zařízení

Opět se setkáváme s problémem Android vs iOS. Jak víme Apple si rád dělá věci po svém a nastavení reálných zařízení pro vývoj není výjimkou.

Pro použití reálného zařízení k testování aplikací vývojáři v Android operačním systému, musíme povolit USB debugging v nastavení zařízení. Toto je obvykle nalezeno v „Nastavení > O telefonu > Informace o telefonu > Stisknutí číslo šestnáctkrát (pro aktivaci režimu vývojáře) > Možnosti vývojáře > Zapnout USB debugging“.

Apple se rozhodl, že pokud chce vývojář bit jenom svojí aplikaci otestovat na iOS, musí zaplatit roční poplatek za vývojářskou licenci. Udělal jsem tedy kompromis a aplikace bude testována na Androidech reálně a na iOS pouze v emulátorech.

Kapitola 3
Flutter

„Flutter je open source framework od Google pro vytváření krásných, nativně kompilovaných, multiplatformních aplikací z jediné kódové základny.“ [5]

Proč ale právě Flutter pro mojí práci? Možností pro mobilní vývoj je mnoho. Již mám nějaké zkušenosti s Kotlinem a vývojem mobilních aplikací pro Android. Moje idea jak projekt vypracuji je taková, že výsledná aplikace bude cross-platform abych si vyzkoušel i jiný přístup k vývoji než Native development a tak jsem vybral framework Flutter. Proč ale ne například React Native, který je lépe podporován v rámci přístupnosti informací? Důvod je takový, že si moc nerozumím s Javascriptem. Navíc na trhu práce nabídnu i dovednost s ne tak momentálně rozšířeným frameworkem. Tato kapitola shrnuje relevantní obsah o Flutteru z knih, výukových videí a oficiální dokumentace. [4, 6]

3.1 Native vs Cross-platform

Native vývoj je spojen s použitím platformě specifických jazyků, jako je Kotlin pro Android a Swift nebo pro iOS. Toto přináší výhody jako optimální výkon a plný přístup k funkcím specifickým pro každou platformu. Vývojáři mohou využívat širokou škálu nástrojů a API poskytovaných danou platformou, což umožňuje detailní ladění a optimalizaci.

Na druhou stranu je cross-platform vývoj, reprezentovaný například frameworkem Flutter, zaměřen na sdílení kódu mezi různými platformami. Flutter používá jazyk Dart a poskytuje vlastní sadu nástrojů a widgetů. To vede ke zrychlení vývoje, ale někdy může snížit výkon, zejména v náročnějších scénářích.

3.2 Historie

První verze Flutteru byla vydána v květnu 2017 a od té doby prošel významným vývojem a zdokonalením. V roce 2018, na konferenci Google I/O, byl představen Flutter 1.0, což bylo oficiální vydání frameworku pro veřejnost. V průběhu dalších let byly pravidelně vydávány aktualizace, které přidávaly nové funkce, opravovaly chyby a zvyšovaly stabilitu frameworku.

Flutter získal popularitu díky své schopnosti vytvářet atraktivní a plynulé uživatelské rozhraní s minimálním úsilím při zachování konzistence napříč platformami. Vývojáři ocenili i hot reload funkcionalitu, která umožňuje okamžitý náhled změn ve zdrojovém kódu během vývoje.

V současné době je Flutter široce využíván pro vývoj mobilních aplikací na platformách Android a iOS, ale také se rozšiřuje do oblasti vývoje desktopových a webových

aplikací. S podporou a aktivní komunitou se očekává, že bude mít Flutter v budoucnu důležitou roli ve světě multiplatformního vývoje mobilních aplikací. To je také jeden z hlavních důvodů proč jsem zvolil právě Flutter

3.3 Dart

Dart je moderní programovací jazyk, vyvinutý společností Google. Od svého uvedení na veřejnost v roce 2011 prošel Dart výrazným vývojem a stal se klíčovým hráčem v oblasti vývoje webových a mobilních aplikací. S jednoduchou a čitelnou syntaxí nabízí vývojářům efektivní nástroj pro tvorbu výkonných a plynulých aplikací.

3.3.1 Syntax

Syntaxe Dartu je jednoduchá, čitelná a srozumitelná, což z něj činí přístupný jazyk pro vývojáře různých úrovní zkušeností. Silná typová kontrola umožňuje bezpečné programování a minimalizuje chyby během vývoje.

Dart je objektově orientovaný jazyk. Funkce jsou firstclass citizens a podporují koncepty jako jsou anonymní funkce a closures. Mimo jiné svojí syntaxí velice připomíná C/C++.

3.4 Základní znalosti o Flutteru

Ve Flutteru se nahlíží na problém jako na skládačku. Nejdříve promyslíte problém a popíšete ho. Následně ho dekomponujete na menší oddělené problémy „komponenty“. Tyto komponenty se skládají z menších komponent a tak dál dokud se nedostanete na nejmenší problém, který musíte řešit. Tento proces se jmenuje „dekomponizace“, ale nás zajímá hlavně proces jemu opačný „komponizace“ tedy skládání kusů zase do sebe a není to žádná novinka.
O komponizaci se poprvé mluvilo už v roce 1968, ale popularity nabila teprve nedávno díky frameworkům jako je React, Angular, Vue, atd. Ve světě Flutteru se těmto komponentám říká „Widgety“. Proto se také každá aplikace rozděluje na dvě části.

1. Chování - co program dělá (logika).

2. Prezentace - jak program vypadá (UI).

3.4.1 Všechno je Widget

Flutter nabízí asi 160 základních widgetů. [7] Je lepší se je učit jako Widgety s různými účely a rozdělit je do kategorií.

1. Hodnotové Widgety

Checkbox	FlatButton
CircularProgressIndicator	FloatingActionButton
Date & Time Pickers	FlutterLogo
DataTable	Form
DropdownButton	FormField

- Icon
- IconButton
- Image
- LinearProgressIndicator
- PopupMenuButton
- Radio
- RaisedButton
- Rawimage
- RefreshIndicator
- RichText
- Slider
- Switch
- Text
- TextField
- Tooltip

2. Layout Widgety

- Align
- AppBar
- AspectRatio
- Baseline
- BottomSheet
- ButtonBar
- Card
- Center
- Column
- ConstrainedBox
- Container
- CustomMultiChildLayout
- Divider
- Expanded
- ExpansionPanel
- FittedBox
- Flow
- FractionallySizedBox
- GridView
- IndexedStack
- IntrinsicHeight
- IntrinsicWidth
- LayoutBuilder
- LimitedBox
- ListBody
- ListTile
- ListView
- MediaQuery
- NestedScrollview
- OverflowBox
- Padding
- PageView
- Placeholder
- Row
- Scaffold
- Scrollable
- Scrollbar
- SingleChildScrollView
- SizedBox
- SizedOverflowBox
- SliverAppBar
- SnackBar
- Stack
- Table
- Wrap

3. Navigační Widgety

- AlertDialog
- BottomNavigationBar
- Drawer
- MaterialApp
- TabBar
- Navigator

TabBarview	SimpleDialog

4. Ostatní Widgety

GestureDetector	Dismissible
Cupertino	Theme
Transitions	Transforms

Je tu také možnost vytvořit si své vlastní Widgety z již existujících Widgetů zapomocí kompozice. Widgety lze také rozlišit jako „Stateless“ (beze stavové) a „Stateful“ (stavové). Když se vytváří vlastní Widgety je lepší návyk začít se Stateless a když se tomu nelze vyhnout tak upravit Widget na Stateful. O stavech až v podkapitole „Stavy“.

3.4.2 Hodnotové Widgety

Tyto widgety mají tu vlastnost oproti ostatním widgetům, že udržují nějakou hodnotu. Jejich hodnoty jsou získány buď defaultní hodnotou od programátora nebo od uživatele. Já zde budu mluvit pro příklad pouze o Text, Icon a Image widgetech.

Text je widget, který umožňuje zobrazit text uživately. Má neupravitelnou hodnotu, tedy její hodnotu určuje programátor. Ve Flutteru se definuje takto:

```
Text('Hello world'),
```

Ovšem je potřeba také zdůraznit že pokud se jedná o konstantní hodnotu pak je lepší říct kompilátoru že se hodnota nebude měnit slovem „const“. Je to nejenom dobrý zvyk z pohledu debugování, ale také umožňuje aplikaci rychleji načítat obrazovku, i přes paměťově náročnější aplikaci.

Icon umožňuje vkládat do aplikace icony již definované ve frameworku. Seznam všech ikon s jejich označením lze nalést na oficiálních stránkách Flutteru. [1]

```
Icon(
  Icons.bluetooth,
  color: Colors.blue,
  size: 10,
)
```

Poslední, který popíšu je Image. Image může být připojený jako soubor nebo jako url odkaz. Pokud víme, že se obrázek nebude nikdy měnit, tak by měl být vždy vložen jako obrázek. Pokud je obrázek soubor, musíme ho vložit do repozitáře a oznámit Flutteru že ho budeme používat v souboru pubspec.yaml.

```
flutter:
  assets:
   - assets/images/screen-welcome.png
   - assets/images/screen-home.jpg
   - assets/images/logo.jpg
```

Obrázek lze pak snadno ve Flutteru zpřístupnit pomocí:

[1] https://api.flutter.dev/flutter/material/Icons-class.html

```
Image.asset('assets/images/screen-home.jpg',),
```

Přístup k obrázku přes internet je poměrně jednoduchý, stačí zadat url. Tento přístup je sice paměťově méně náročný, ale je o dost pomalejší.

```
Image.network("https://www.google.com/path/to/image/logo.jpg"),
```

3.4.3 Layout Widgety

Layout widgety ve Flutteru hrají klíčovou roli při definování vizuálního uspořádání uživatelského rozhraní. Pomáhají organizovat a umisťovat widgety na obrazovce podle potřeby. Zde jsou některé z nejčastěji používaných layout widgetů:

Container: Tento widget umožňuje definovat vlastnosti kontejneru, jako je velikost, pozice a dekorace. Může obsahovat jeden widget a používá se k organizaci a stylizaci ostatních widgetů.

Row: Row je layout widget, který uspořádá své děti ve vodorovném směru. Je užitečný pro uspořádání více widgetů vedle sebe.

Column: Column je podobný Row, ale uspořádá své děti ve svislém směru. Používá se pro uspořádání widgetů nad sebou.

Stack: Stack umožňuje překrývání widgetů jedním nad druhým. Každý widget ve stacku může být umístěn na základě relativního pozicionování.

Expanded: Tento widget rozšiřuje své dítě na maximální dostupnou šířku nebo výšku ve směru rodiče. Používá se obvykle uvnitř Row nebo Column pro rozdělení dostupného prostoru.

ListView: ListView je widget, který organizuje své děti ve scrollovatelném seznamu, buď ve svislém, nebo vodorovném směru.

Tyto layout widgety společně poskytují flexibilní možnosti pro design UI ve Flutteru.

3.4.4 Navigační Widgety

Navigační widgety jsou klíčové pro navigaci mezi různými obrazovkami ve vaší aplikaci. Ve Flutteru existuje několik navigačních widgetů, které umožňují správu navigace:

MaterialApp: MaterialApp je kořenový widget vaší aplikace a poskytuje infrastrukturu pro navigaci pomocí widgetů jako je Navigator a MaterialApp.navigatorKey.

Navigator: Navigator je widget, který spravuje zásobník tras (routes) vaší aplikace. Pomocí navigatoru můžete přejít na jinou trasu nebo se vrátit zpět na předchozí trasu.

Scaffold: Scaffold je layout widget, který implementuje základní layout design pro aplikace podle Material Design specifikací. Obsahuje mnoho prvků, jako je AppBar, BottomNavigationBar a Drawer, které usnadňují navigaci a interakci s aplikací.

BottomNavigationBar: BottomNavigationBar je navigační prvek, který umožňuje uživatelům přepínat mezi různými hlavními obrazovkami aplikace pomocí ikon nebo textu umístěného na spodní části obrazovky.

Drawer: Drawer je navigační prvek, který poskytuje boční menu, které může být vyvoláno uživatelem přetáhnutím z levého nebo pravého okraje obrazovky.

Tyto navigační widgety nám umožňují snadno implementovat složité navigační vzory.

3.4.5 Ostatní Widgety

Kromě layout a navigačních widgetů existuje v Flutteru mnoho dalších užitečných widgetů pro různé účely:

Text: Widget pro zobrazení textu s možností formátování a stylování.

Icon: Widget pro zobrazení ikony z definované sady ikon, například z Material Icons.

Image: Widget pro zobrazení obrázku ze souboru, assetů nebo z URL.

Button: Flutter poskytuje několik různých typů tlačítek, jako je RaisedButton, FlatButton a IconButton, pro zachycení uživatelských interakcí.

Input: Pro interakci s uživatelem můžete použít widgety jako je TextField pro zadávání textu a TextFormField pro formulářové pole s ověřením.

List: Kromě ListView je k dispozici mnoho dalších widgetů pro zobrazení seznamů, jako je například GridView pro zobrazení dat v mřížce.

Dialog: Pro zobrazení upozornění a dialogových oken můžete použít widgety jako AlertDialog a SimpleDialog.

Animation: Pro přidání animací do vaší aplikace můžete použít widgety jako AnimatedContainer nebo FadeTransition.

Tyto widgety vám umožňují vytvářet bohaté a interaktivní uživatelské rozhraní ve vaší aplikaci Flutter.

3.4.6 Stylování

Ve Flutteru můžeme stylovat své widgety pomocí různých technik, včetně použití vestavěných témat, definování vlastních stylů pomocí ThemeData a použití widgetů jako Container a BoxDecoration pro přímé nastavení stylů jako je barva a okraje. Můžeme také použít rozšíření jako TextStyle pro stylování textu a BoxDecoration pro stylování pozadí kontejnerů. Použití konzistentního a esteticky příjemného stylování může výrazně zlepšit vzhled aplikace a uživatelskou zkušenost.

3.4.7 Stavy

Ve Flutteru existují dva základní typy widgetů ohledně zacházení se stavy: Stateless a Stateful widgety.

Stateless Widgety neuchovávají stav mezi jednotlivými vykresleními a jsou statické. Jednou inicializované, jejich stav zůstává neměnný po celou dobu životního cyklu widgetu. Tyto widgety jsou vhodné pro prezentaci statického obsahu nebo komponent, které se nemění v průběhu času.

Stateful Widgety na druhou stranu uchovávají stav mezi jednotlivými vykresleními. Tyto widgety mohou reagovat na uživatelské interakce a aktualizovat svůj stav, což vede k opětovnému vykreslení widgetu s novými daty. Tento dynamický stav umožňuje vývojářům vytvářet interaktivní uživatelské rozhraní a reagovat na události v aplikaci.

Při práci s Stateful widgety je zásadním konceptem `setState` metoda, která signalizuje frameworku, že se stav widgetu změnil a že je třeba znovu vykreslit widget s aktualizovanými daty.

3.5 Přístup k databázím

Při výběru metody přístupu k databázím je důležité zvážit požadavky vaší aplikace, jako je typ dat, rozsah operací a závislosti na externích službách. Pro přístup k databázím ve Flutteru existuje několik možností, včetně použití vestavěných pluginů a balíčků třetích stran. V každém typu také ukazuji vzorový příklad použití dané databáze.

3.5.1 SQLite

SQLite je vestavěná relační databáze, která je často používána pro ukládání dat v mobilních aplikacích. Existují balíčky pro Flutter, které umožňují přístup a manipulaci s SQLite databázemi přímo z vaší aplikace.

```
// await znamená že aplikace čeká na vykonání příkazu
await database.then((db) async {
  // Vložení dat do tabulky 'uzivatele'
  await db.insert(
    'uzivatele',
    {'jmeno': 'Pepa Novak', 'vek': 30, 'pohlavi': 'muz'},
    conflictAlgorithm: ConflictAlgorithm.replace,
  );
});
```

3.5.2 Firebase

Firebase je oblíbená platforma poskytující širokou škálu služeb, včetně databáze, autentifikace, analýzy a mnoho dalšího. Existují oficiální balíčky pro Flutter, které umožňují integraci s Firebase službami a snadný přístup k jejich funkcím.

```
//připojím se do NoSQL kolekce 'users' a přidám uživatele
FirebaseFirestore.instance
    .collection('users')
    .add({'name': 'John Doe', 'age': 30, 'gender': 'male'});
```

3.5.3 HTTP requests

Pro komunikaci s externími API nebo webovými službami můžete využít vestavěného balíčku „http“ pro Flutter, který umožňuje provádět HTTP požadavky a zpracovávat odpovědi v rámci vaší aplikace.

```
//používá metodu GET na příspěvek s identifikátorem 1
final response = await http.get(Uri.parse('https://jsonplaceholder.
typicode.com/posts/1'));
```

3.6 Rozdíly oproti React Native

Při porovnání s React Native existuje několik klíčových rozdílů, které je důležité vzít v úvahu při výběru mezi oběma frameworky:

3.6.1 Programovací jazyk

Flutter používá programovací jazyk Dart, zatímco React Native používá JavaScript. Zatímco mnoho vývojářů je seznámeno s JavaScriptem, Dart může vyžadovat určitou učící křivku, ale mnoho vývojářů ocení jeho statickou typovou kontrolu a další vlastnosti.

3.6.2 Vykreslování UI

V React Native je UI vykreslováno pomocí nativních komponentů platformy, zatímco ve Flutteru je UI vykreslováno pomocí vlastního frameworku. To může mít vliv na výkon a vzhled aplikace.

3.6.3 Hot Reload

Obě frameworky poskytují funkci Hot Reload, která umožňuje okamžité zobrazení změn v kódu během vývoje. Flutter však obecně považuje tuto funkci za rychlejší a spolehlivější.

3.6.4 Komunita a ekosystém

React Native má delší historii a větší komunitu v porovnání s Flutterem. To může mít vliv na dostupnost knihoven a nástrojů třetích stran.

3.6.5 Shrnutí

Při rozhodování mezi Flutterem a React Native je důležité zvážit potřeby vaší aplikace, vaše zkušenosti a preference v programování, a také dostupnost knihoven a nástrojů ve vašem ekosystému.

Kapitola 4
Návrh aplikace

Při návrhu aplikace je potřeba si uvědomit, že i kdyby byl návrh perfektní, vždy je něco, co se nakonec změní až při implementaci. Ovšem je to nevyhnutelná a esenciální část projektu. Dobrý návrh významně ušetří výsledný čas implementace a zabrání problémům jimž je možné se vyhnout.

4.1 Rešerše existujících řešení

4.1.1 ProKlub

Jedná se o momentální používanou aplikaci Choketopus Gym. Aplikace umožňuje kontrolu plateb, docházky, přihlašování na akce, komunikovat se sportovním klubem, apod.

Aplikace vyniká ve své jednoduchosti a přehlednosti. Také poskytuje přihlašování na akce tvořené centrem.

Má ovšem několik nedostatků, které je třeba zmínit. Do aplikace se lze přihlásit pouze přes QR kód, který musí zadavatel manuálně poslat. Pokud v systému není založená docházka, tak se na akci nelze přihlásit. Po přihlášení na akci se nelze z aplikace odhlásit. Také nelze platit přímo z aplikace.

4.1.2 Five-O

Jelikož jeden ze zadavatelových požadavků je umožnit klientovy posilovny sledovat výuková videa a udělat si přehled nad vyučujícími technikami. Je nutné do rešerše zakomponovat aplikaci, která právě toto umožňuje.

Ikdyž je aplikace vcelku elegantně zpracovaná, například dobrá orientace či kvalitní videa, má také pár problémů. Jedním z těchto problémů je nedostatek detajlních popisů jednotlivích technik, které by si uživatel mohl číst jako zápisky. Také je zde složitost pro začínající sportovce, kteří se se sportem teprve seznámily a nevědí, kde začít. Mimo jiné aplikace také obsahuje málo obsahu zdarma a je nutné platit za jednotlivá videa.

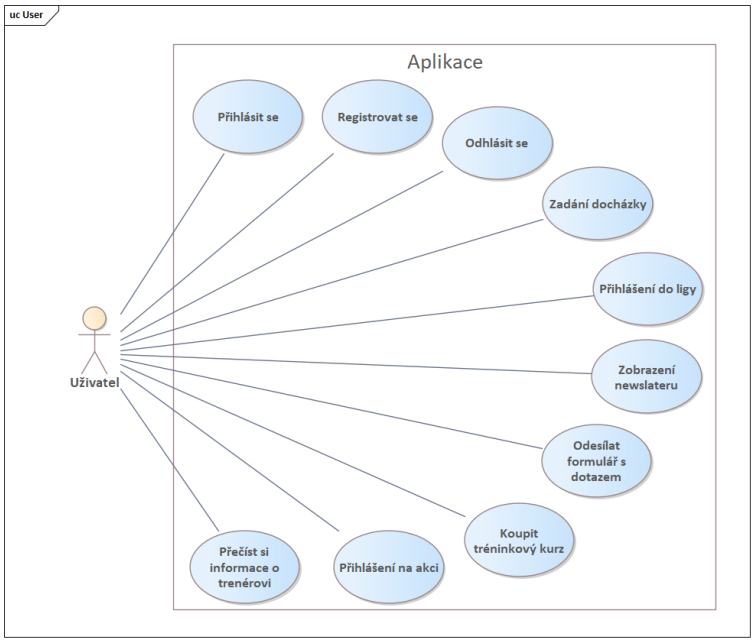
4.1.3 JeFit

Je nutné se podívat také na aplikace, které sice souvisí se zdravím, ale nijak se nekonkretizují na jednotlivé sporty. Jednou z těchto aplikací je právě JeFit. Jedná se o fitness aplikaci, která může poskytnout odlišný pohled na problematiku, či poskytnout nápady, co chybí v jiných aplikacích.

Aplikace poskytuje rozsáhlé možnosti ve směru stravování a fitness cvičení. Lze například vypíchnout některé UI komponenty které jsou hezky řešené. Ovšem aplikace je velice nepřehledná. Obsahuje spousty UI komponent bez intuitivního významu a uživatel proklikává, dokud nenajde. Také je zde veliký počet reklam, které moc přehlednosti nepomáhají.

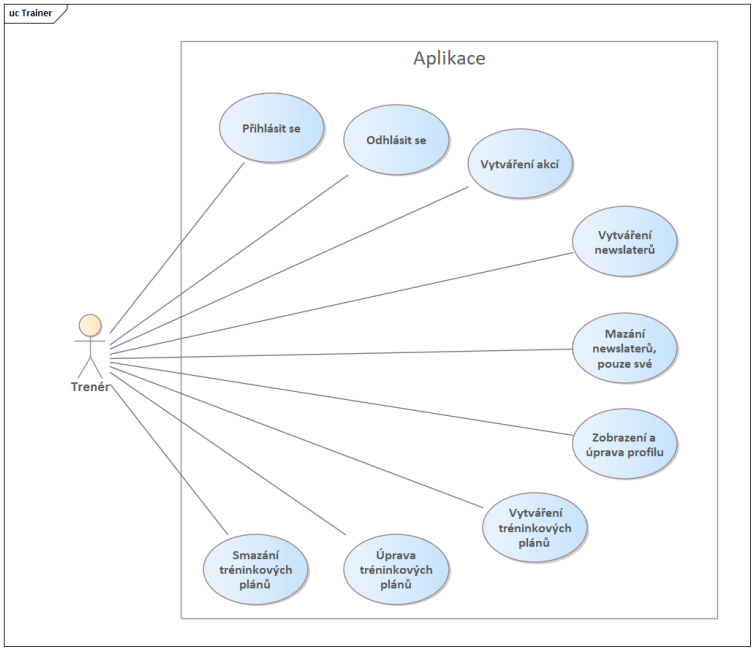
4.2 Use-Case diagramy

4.2.1 Uživatel



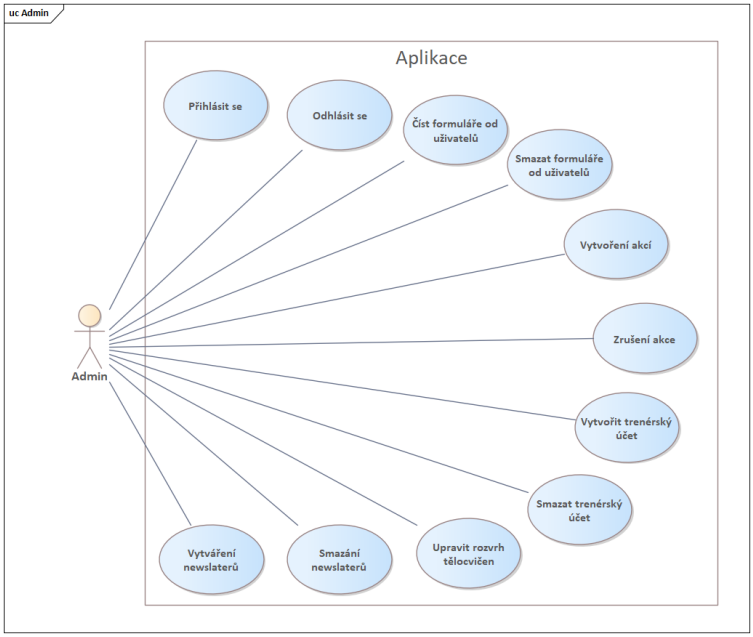
Obrázek 4.1. Use-case pro běžného uživatele aplikace

4.2.2 Trenér



Obrázek 4.2. Use-case pro trenéra

4.2.3 Admin



Obrázek 4.3. Use-case pro admina

4.3 Uživatelské rozhraní

Uživatelské rozhraní (nyní pouze UI) pro telefony je klíčovým prvkem, který ovlivňuje celkový uživatelský zážitek.

Mělo by být intuitivní a snadno ovladatelné. I uživatelé bez technického pozadí by měli být schopni rychle pochopit, jak aplikaci používat. Klienti sportovního centra jsou totiž všech věkových kategorií. Je také důležité nepřeplňovat obrazovky informacemi, ve kterých se uživatel snadno ztratí. Nelze zapomenout také na udržování stejného designu (barev, textu, widgetů, ...) pro všechny obrazovky. Dostatečná čitelnost a velikost je samozřejmě také důležitá. Rychlá odezva a plynulost při interakci jsou klíčové pro pozitivní uživatelský zážitek. Zpoždění a záseky mohou vést k frustraci uživatelů a neočekávanému chování aplikace. Možnost nastavit si prostředí podle své preference bývá v dnešní době nutností, proto tato kapitola obsahuje také UI pro nastavení, které musí být hlavně jednoduché.

Všechny návrhy byli tvořené v UI design nástroji Figma. Jak bylo již řečeno, UI bude odlišné pro uživatele, trenéry a admina, ikdyž si budou velice podobné. Nejčastější užívání aplikace bude ovšem z pohledu uživatele. Aplikace ovšem bude obsahovat desítky obrazovek, tudíž se zaměříme na ty z mého pohledu nejdůležitější.

Obsah aplikace (data) jsou získaná ze stránky Choketopusu. [8]

Také je zde možnost připojení se na jejich eshop. [9]

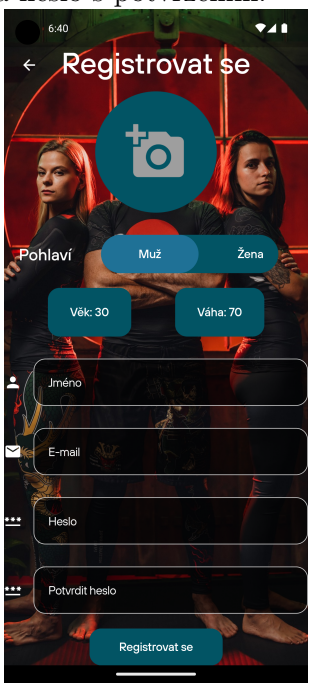
4.3.1 Uvítací obrazovka

Uvítací obrazovka je to první co uživatel vidí při prvním otevření aplikace nebo po odhlášení. Musí být přívětivá a hlavně stručná. Umožňuje uživateli se přihlásit nebo se zaregistrovat.

Obrázek 4.4. UI k uvítací obrazovce

4.3.2 Registrační formulář

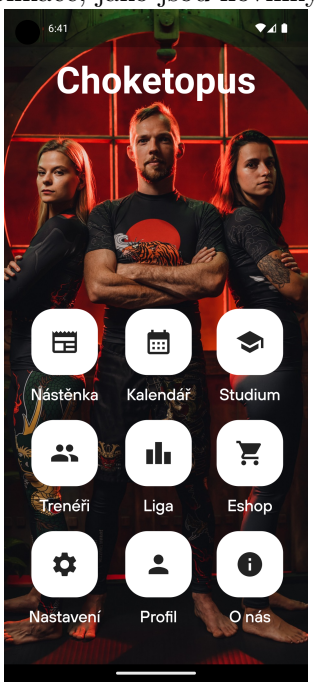
Tato obrazovka umožňuje novým uživatelům vytvořit si účet ve sportovním centru. Obsahuje formulář pro zadání osobních údajů potřebných k registraci. Uživatel musí zadat již neregistrovaný email a heslo s potvrzením.



Obrázek 4.5. UI k registračnímu formuláři

4.3.3 Hlavní nabídka

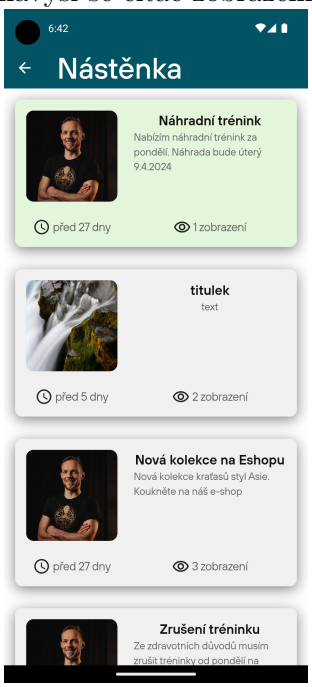
Hlavní nabídka slouží k navigaci a přístupu ke klíčovým funkcím aplikace. Zobrazuje ikony a názvy hlavních částí aplikace, jako jsou novinky, kalendář, studium atd.



Obrázek 4.6. UI k uvítací obrazovce

4.3.4 Novinky

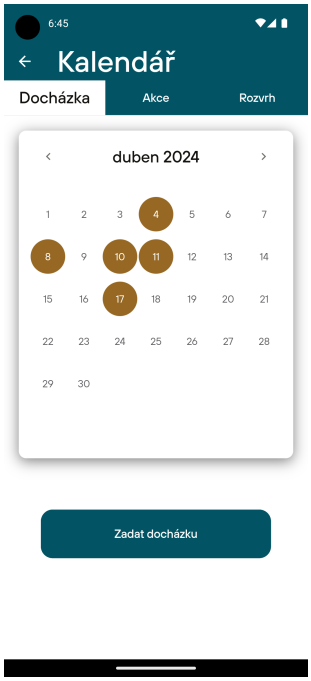
Obrazovka s newslatery poskytuje uživateli přehled o novinkách, událostech nebo důležitých informacích od sportovního centra. Nepřečtené zprávy jsou označeny zeleně. Po zakliknutí se zpráva odbarví a navýší se čítač zobrazení.



Obrázek 4.7. UI k uvítací obrazovce

4.3.5 Kalendář

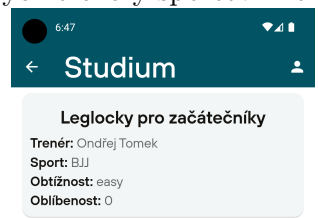
Kalendářní obrazovka slouží k zobrazení plánu událostí, lekcí nebo jiných aktivit ve sportovním centru. Umožňuje uživateli procházet události podle data a času a sledovat jeho docházku. Poskytuje detaily o jednotlivých událostech a možnost přihlášení na akce.



Obrázek 4.8. UI k uvítací obrazovce

4.3.6 Studium

Studium poskytuje materiály ke studiu jako jsou videa s detailními popisy zabalené v tréninkových plánech vytvořených trenéry sportovního centra.



Obrázek 4.9. UI k uvítací obrazovce

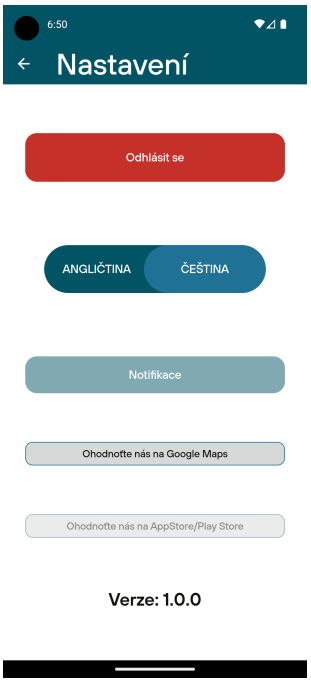
4.3.7 Trenéři

Obrazovka s trenéry umožňuje uživatelům procházet seznam trenérů ve sportovním centru a získat informace o jejich zkušenostech a nabízených službách (sportech).

Obrázek 4.10. UI k uvítací obrazovce

4.3.8 Nastavení

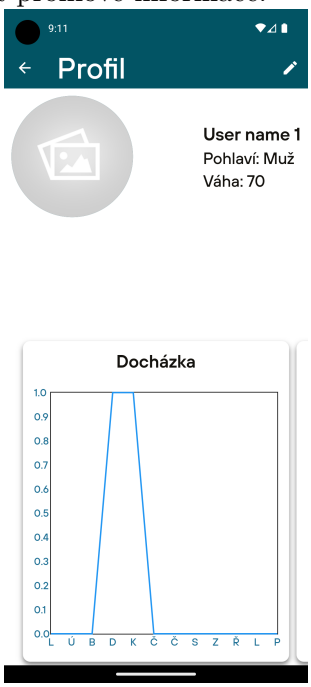
Tato obrazovka slouží k nastavení preferencí a personalizace aplikace podle potřeb uživatele. Také umožňuje uživateli se odhlásit.



Obrázek 4.11. UI k uvítací obrazovce

4.3.9 Profil

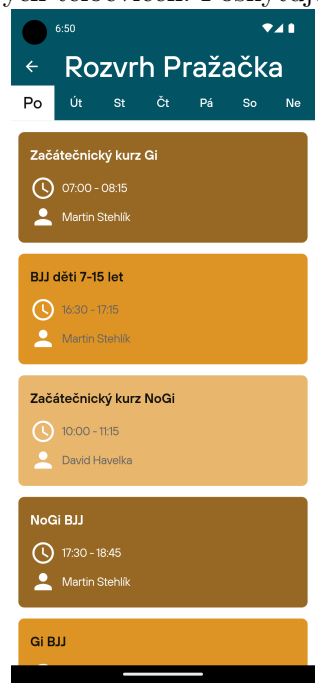
Profilová obrazovka zobrazuje informace o uživateli, včetně jména, fotky a zobrazuje statistiky nebo úspěchy spojené s aktivitami ve sportovním centru. Také samozřejmě umožňuje uživateli aktualizovat profilové informace.



Obrázek 4.12. UI k uvítací obrazovce

4.3.10 Rozvrhy

Obrazovka s rozvrhy poskytuje přehled o plánech lekcí, otevíracích hodinách a dalších časových informacích jednotlivých tělocvičen. Poskytuje přepínání mezi dny.

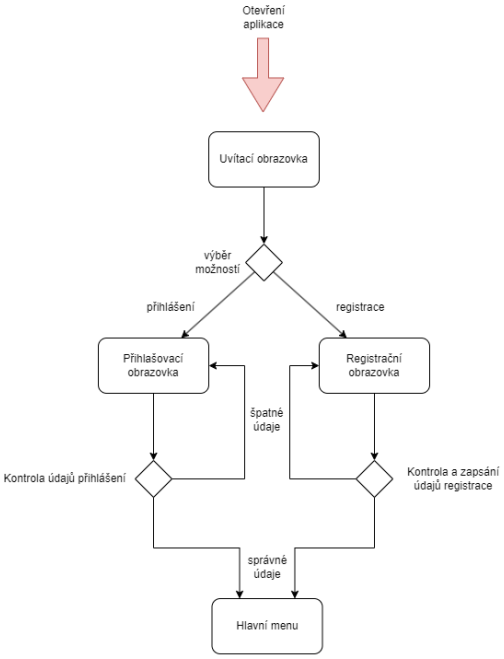


Obrázek 4.13. UI k uvítací obrazovce

4.4 Activity process diagramy

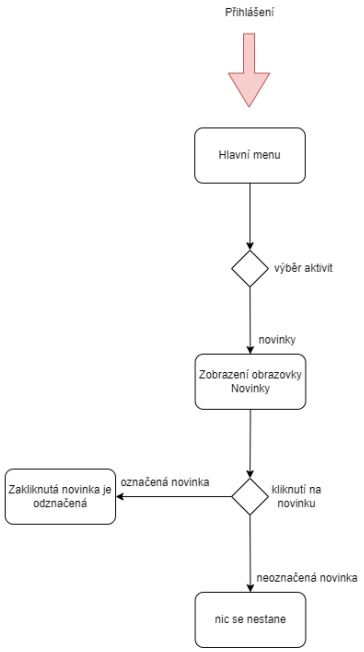
V této podkapitole si představíme Activity process diagramy mého projektu. Zaměřil jsem se na nejdůležitější a nejčastější aktivity, na které běžný uživatel narazí.

4.4.1 Přihlášení a registrace



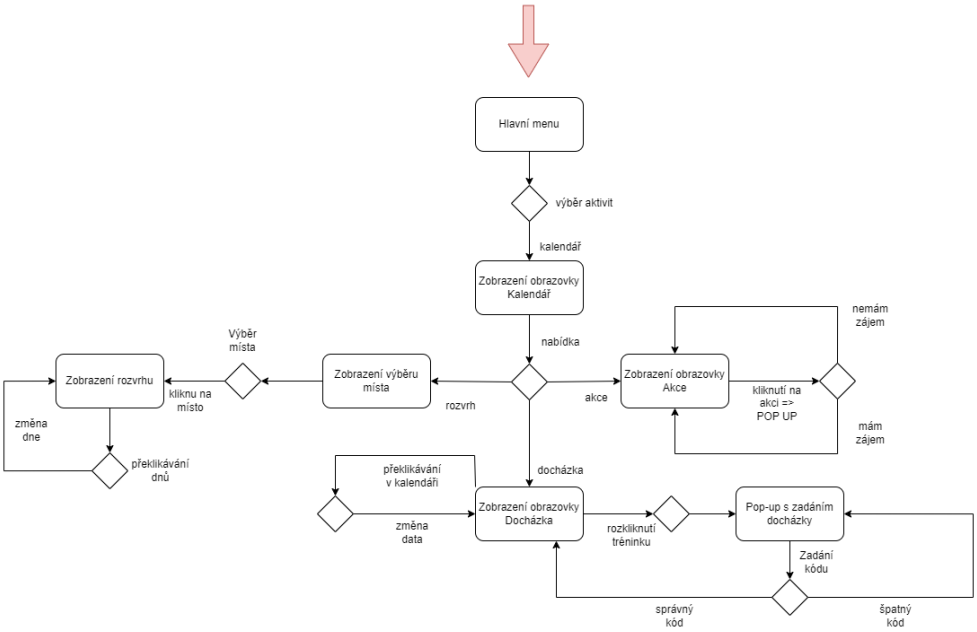
Obrázek 4.14. Activity diagram pro přihlášení a registraci

4.4.2 Zobrazení novinek



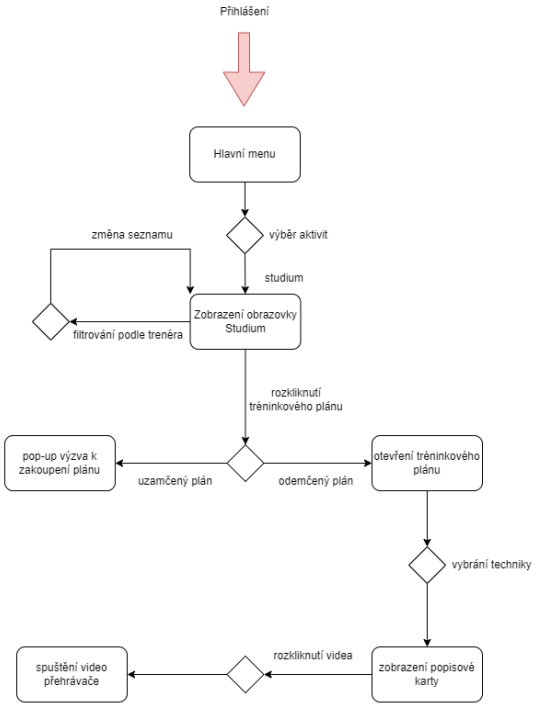
Obrázek 4.15. Activity diagram pro zobrazení newslaterů

4.4.3 Práce s kalendářem



Obrázek 4.16. Activity diagram pro práci s kalendářem

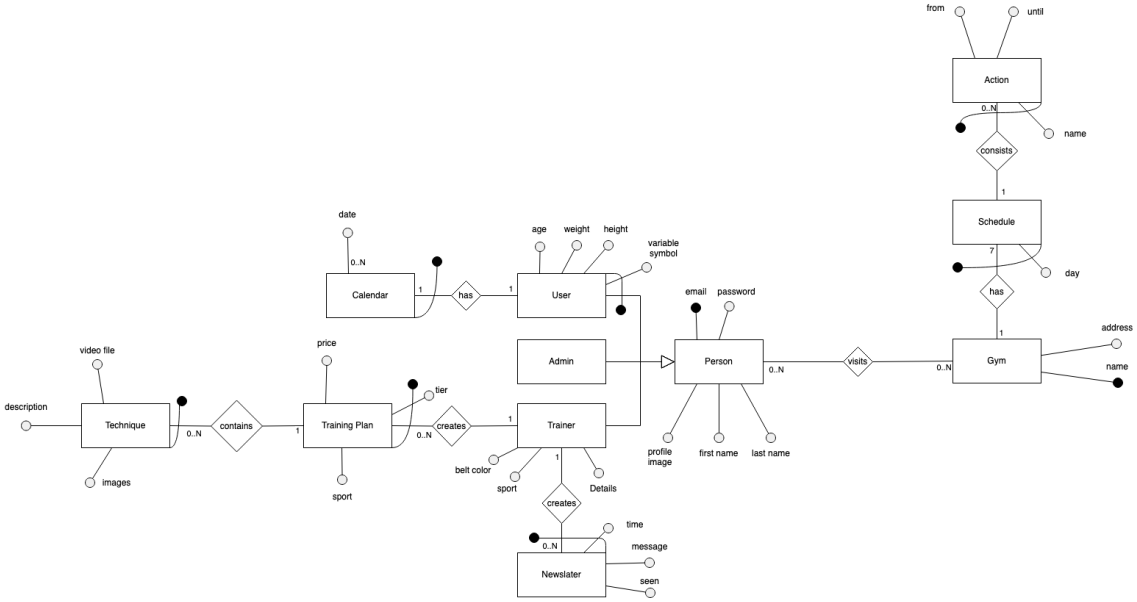
4.4.4 Spouštění výukových videí



Obrázek 4.17. Activity diagram pro zobrazování videí

4.5 Databáze

Databáze bude řešena prostřednictvím Firebase cloud.



Obrázek 4.18. Schéma databáze

4.5.1 NoSQL

V rámci projektu bude využita NoSQL databáze, konkrétně Firebase Cloud Firestore, která poskytuje flexibilní a škálovatelné úložiště dat pro naši aplikaci. NoSQL databáze jsou vhodné pro dynamická a nestrukturovaná data, což odpovídá potřebám moderních mobilních aplikací, které často pracují s různorodými typy obsahu. Firebase Cloud Firestore umožňuje ukládat data ve formátu dokumentů a kolekcí, což je intuitivní a efektivní způsob organizace informací. Díky tomu budeme schopni snadno spravovat a aktualizovat data v naší aplikaci, a to i při nárůstu uživatelského zájmu a objemu dat.

4.5.2 Struktura databáze

V rámci naší aplikace bude struktura databáze v NoSQL formátu pečlivě navržena tak, aby co nejlépe odpovídala potřebám našeho projektu. Struktura databáze bude rozdělena do logických kolekcí a dokumentů, které budou reprezentovat jednotlivé entity a relace v naší aplikaci. Díky této strukturovanosti budeme schopni efektivně organizovat a ukládat data tak, aby byla snadno dostupná a přehledná pro nás i pro naše uživatele.

V rámci struktury databáze budou jednotlivé dokumenty obsahovat atributy odpovídající konkrétním entitám naší aplikace, jako jsou uživatelé, události, tréninky nebo statistiky. Každý dokument bude identifikován unikátním klíčem a bude obsahovat různé pole s informacemi relevantními pro danou entitu. Kromě toho budou jednotlivé dokumenty navzájem propojeny pomocí referencí nebo identifikátorů, což nám umožní efektivně reprezentovat vztahy mezi jednotlivými entitami a provádět operace jako je filtrování, řazení nebo spojování dat.

4.6 Určení priorit a minimální funkční prototyp

Určení priorit a vytvoření minimálního funkčního prototypu jsou klíčové kroky v procesu vývoje mého projektu. Tato fáze je klíčová pro definování cílů, identifikaci nejvýznamnějších prvků a zajištění efektivního využití zdrojů.

V tomto shrnutí se zaměříme na důležitost určení priorit a na vytváření minimálního funkčního prototypu jako strategického nástroje pro úspěch projektu. Je potřeba určit, na co se zaměřit a co je možné nechat na později.

Všichni čtenáři mi dají jistě za pravdu, že je důležitější, aby se uživatel byl schopný v pořádku a bezpečně přihlásit než to zda se vám zobrazuje popis trenéra.

4.6.1 Priority

Určení priorit představuje proces identifikace a klasifikace klíčových prvků nebo úkolů, které mají významný dopad na úspěch projektu. Tato fáze mi jako programátorovi rozhodnout, co je nejdůležitější a co by mělo být řešeno jako první.

Prioritizace může být provedena několika způsoby. Tyto způsoby jsou důležitost pro uživatele, komplexnost implementace, strategický dopad.

Pro určení priorit vytvořím celkem 4 tříd, kde 1 je nejdůležitější a 4 není tak důležitá.

```
Třída 1: Nezbytné - prvky, které jsou nezbytné pro základní funkčnost
aplikace a musí být implementovány jako první.
Třída 2: Důležité - prvky, které mají vysoký dopad na uživatelskou
zkušenost nebo klíčové funkcionality aplikace.
Třída 3: Střední - prvky, které jsou užitečné, ale nejsou tak kritické
jako prvky ve třídě 1 a 2.
Třída 4: Neprioritní - prvky, které mohou být odloženy na později a
nemají okamžitý dopad na úspěch projektu.
```

Zde je rozložení úkolů v rámci mého projektu:

```
Třída 1:
    - Úvodní obrazovka
    - Přihlašovací do aplikace
    - Registrace do aplikace
    - Hlavní nabídka
    - Zadání a prohlížení docházky
    - Systém ligy (vyzvání, potvrzení, zadání výsledku, připsání    bodů)
    - Nastavení - odhlášení, změna jazyka aplikace
    - Studium výukových videí a jejich sledování
    - Přidávat/odebírat tréninkový plán
    - Propojení s databází/cloudem přes Firebase API

Třída 2:
    - Nástěnka novinek
    - Zobrazení událostí
    - Přidávání/odebírání trenérských účtů adminem
    - CZ/EN lokalizace
```

```
Třída 3:
    - Zobrazení rozvrhu tělocvičen
    - Profilová obrazovka
    - Obrazovka s trenéry
    - Změna hesla

Třída 4:
    - Obrazovka "O nás"
    - Informační obrazovka o sportu
    - Informační obrazovka o tělocvičnách
    - Upravit informace na profilu
```

4.6.2 Minimální funkční prototyp

Minimální funkční prototyp (MVP) je zjednodušená verze produktu, která obsahuje pouze nejnutnější funkce k splnění primárních cílů.

MVP pro mojí aplikaci vypadá takto:

```
1. uživatel je schopný do aplikace přihlásit, zaregistrovat a odhlásit
2. uživatel je schopný navigovat mezi obrazovkami
3. uživatel si je schopný zobrazit základní data z databáze
4. uživatel je schopný zadat docházku
5. uživatel je schopný spouštět a sledovat videa v aplikaci
6. uživatel může interagovat s jinými uživateli v lize
7. aplikace nepadá (nebo padá v krajních případech)
```

Kapitola 5
Implementace

Dne 23.2.2024 jsem začal vypracovávat aplikaci v rámci mé bakalářské práce. Díky možnosti seznámit se s technologií ještě před začátkem implementace jsem věděl, kde začít. Nejprve jsem vytvořil repozitář na Gitlabu a do něj nahrál prvotní založení projektu. Práce mohla začít. Nejprve se musí nastavit, základní konfigurační data v „main.dart“, jako navigace, Theme aplikace, jazyk nebo také orientace (na výšku/na šířku).

Teď když jsme nastavili aplikaci je možné se soustředit na její obsah. Obecně jsem se soustředil nejdřív vytvořit statické UI obrazovky a navigace mezi nimi ale ne moc z důvodu mé orientace v celém projektu. Zprvu se jednalo o velice jednoduché návrhy a postupně je komplikoval.

5.1 Struktura Repozitáře

Struktura repozitáře je klíčová pro efektivní správu zdrojových souborů a kódové základny naší aplikace. Rozdělení repozitáře do logických částí umožňuje lepší organizaci a správu projektu, což přispívá k rychlosti a spolehlivosti vývoje.

Ve struktuře repozitáře nalezneme několik klíčových částí, z nichž každá má svou specifickou roli a účel.

5.1.1 assets

Tato část repozitáře slouží k ukládání statických souborů, jako jsou obrázky, ikony, fonty nebo jiné multimediální soubory, které budou použity v naší aplikaci. Správné organizování těchto souborů v této části umožňuje snadnou správu a přístup k nim z kódu.

5.1.2 android

Adresář android obsahuje zdrojové soubory a konfigurace specifické pro platformu Android. Sem patří například soubory manifestu, ikony aplikace, konfigurace Gradle a další prvky nezbytné pro kompilaci a běh aplikace na platformě Android.

5.1.3 ios

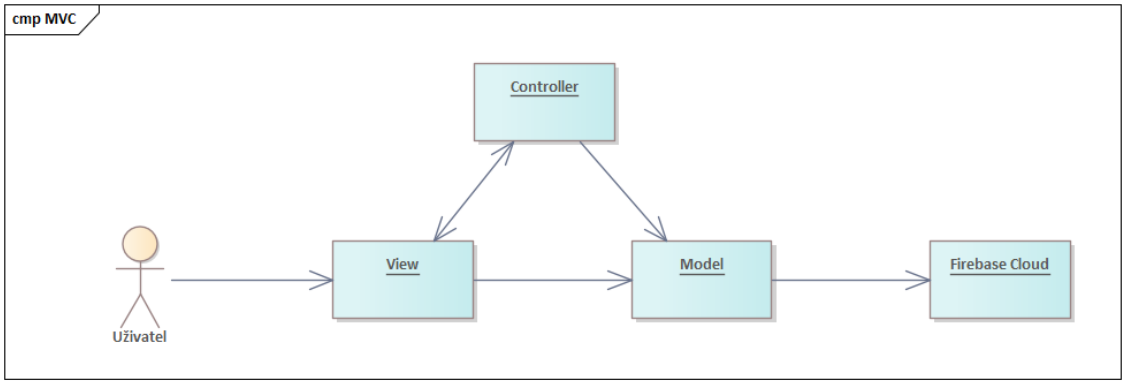
Tento adresář obsahuje soubory a konfigurace specifické pro platformu iOS. Sem patří například soubory projektu Xcode, ikony aplikace pro iOS, konfigurace pro propojení s Apple App Store a další prvky nezbytné pro kompilaci a běh aplikace na platformě iOS.

5.1.4 lib

Lib je klíčovou částí repozitáře, kde se nachází veškerý zdrojový kód naší aplikace napsaný v programovacím jazyce Dart. Zde jsou umístěny veškeré soubory s logikou aplikace, definice UI komponent, služby a další součásti naší aplikace. Správná organizace kódu v této části repozitáře je klíčová pro přehlednost a udržitelnost našeho projektu.

5.2 Architektura aplikace

Architektura aplikace je základním stavebním kamenem pro vytvoření robustní a udržitelné mobilní aplikace. Jedním z populárních architektonických vzorů pro organizaci kódu je Model-View-Controller (MVC).



Obrázek 5.1. Grafické znázornění použité architektury

Tento vzor rozděluje aplikaci do tří základních komponent:

5.2.1 Model

Model je zodpovědný za reprezentaci dat a podnikové logiky aplikace. Komunikuje s databází a zajišťuje bezproblémové předání dat. Obsahuje tedy datové struktury, logiku zpracování dat a operace prováděné nad daty. Model je nezávislý na uživatelském rozhraní a neobsahuje žádnou prezentaci dat.

5.2.2 View

View je zodpovědný za zobrazení uživatelského rozhraní a interakci s uživatelem. Jeho hlavní funkcí je prezentovat data uživateli a zajišťovat uživatelskou interaktivitu. View obvykle neobsahuje žádnou podnikovou logiku, ale pouze přímé zobrazení dat poskytnutých modelem.

5.2.3 Controller

Controller slouží jako prostředník mezi modelem a pohledem. Zpracovává uživatelské vstupy a rozhoduje, jaká akce se má provést na základě těchto vstupů. Controller také komunikuje s modelem pro získání nebo aktualizaci dat a aktualizuje pohled na základě stavu aplikace.

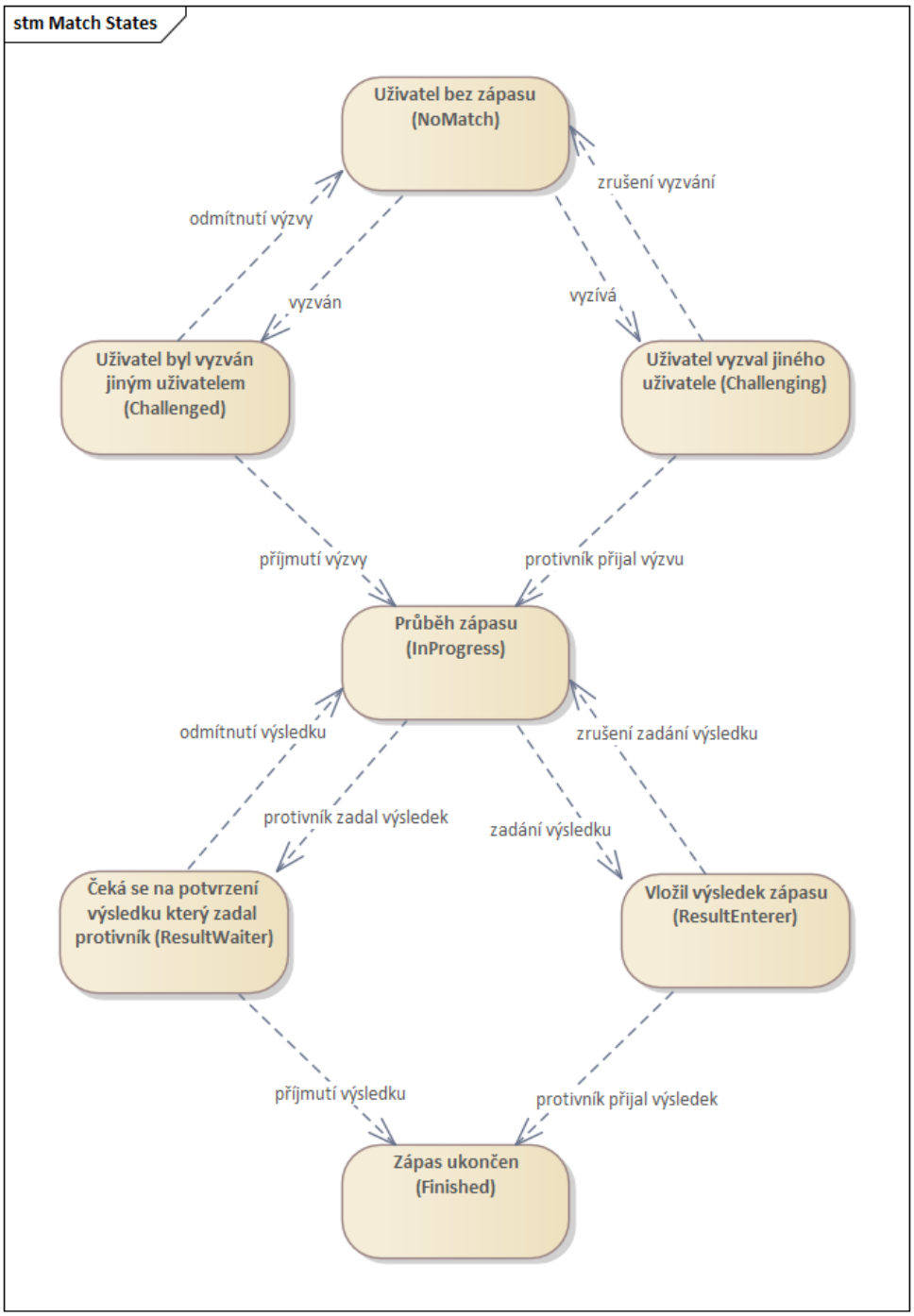
Vzor Model-View-Controller (MVC) poskytuje jasnou separaci mezi daty, prezentací a řízením aplikace, což usnadňuje správu a údržbu kódu. Díky této architektuře je možné snadněji rozdělit odpovědnosti mezi různé části aplikace a získat vyšší úroveň modularizace a znovupoužitelnosti kódu.

5.3 Implementace předem určených funkcionalit

Při implementaci předem určených funkcionalit je dbát úvodního designu, požadavků klienta a strukturovat proces vývoje, aby bylo dosaženo požadovaných cílů a výsledků. Následující podkapitoly se zaměřují na 3 konkrétní hlavní funkcionality, které byly předem definovány v rámci požadavků na aplikaci.

5.3.1 Zápasení v Lize

Zápasení v lize která umožňuje uživatelům soutěžit v rámci ligové struktury. Implementace tohoto prvku zahrnuje vytvoření uživatelského rozhraní pro prohlížení momentálního stavu ligy, registraci do ligy, zápasy proti ostatním uživatelům v lize a sledování výsledků zápasů a umístění v ligové tabulce. Je nutné se ujistit, že interakce mezi uživateli je plynulá a intuitivní. Také je důležité zajistit stav jednotlivých částí zápasů a jejich konzistenci mezi 2 uživateli, kteří spolu zápasí.
Zde je stavový diagram, který popisuje právě průběh celého zápasu:



Obrázek 5.2. Stavový diagram pro průběh zápasu

5.3.2 Zadání docházky

Zadání docházky je klíčovou funkcionalitou aplikace, která umožňuje uživatelům zaznamenávat svou docházku na tréninky a události. Implementace tohoto prvku vyžaduje vytvoření uživatelského rozhraní pro zadávání docházky, ukládání docházky do databáze a její zobrazení. Dále je třeba zajistit synchronizaci těchto dat mezi mobilní aplikací a cloudovým úložištěm, aby byla zaručena dostupnost a konzistence dat pro všechny uživatele.

5.3.3 Sledování a vytváření tréninkových plánů

Funkcionalita sledování umožňuje uživatelům sledovat výuková videa jejíž obsah vytváří trenéři. Implementace tohoto prvku zahrnuje vytvoření uživatelského rozhraní pro správu tréninkových plánů trenérům. Trenéři musí být schopni vytvářet a mazat pouze své plány a přidávat nebo odebírat do nich videa. Běžný uživatel může pouze plány sledovat.

5.4 Výsledek implementace

Po dokončení implementace projektu bylo dosaženo všech stanovených cílů. Aplikace byla úspěšně vyvinuta a obsahuje všechny plánované funkcionality. Uživatelské rozhraní bylo navrženo tak, aby bylo intuitivní a přívětivé pro uživatele. Aplikace je plně funkční a připravená k nasazení.

5.4.1 Problémy při implementaci

Během implementace se objevilo několik problémů, které bylo třeba řešit. Jedním z hlavních problémů bylo propojení aplikace s databází. Jednalo se o moji první zkušenost s cloud přístupem k datům a musel se vše naučit pokus omyl. Dalším problémem byla lokalizace aplikace mezi češtinou a angličtinou. A další závažnější problém byla organizace repozitáře pro mojí orientaci při vývoji. Tyto problémy vyžadovaly dodatečný čas a úsilí k jejich řešení.

5.4.2 Závěrečná podoba

Závěrečná podoba aplikace je dle mého názoru moderní. Je použita standardní architektura MVC. Uživatelské rozhraní je esteticky příjemné a snadno použitelné. Všechny funkcionality (až na drobné výjimky) jsou plně implementovány a otestovány, což zajišťuje spolehlivý provoz aplikace. Celkově je výsledná podoba aplikace v souladu s očekáváním a splňuje požadavky stanovené v počátečním plánu.

5.4.3 Časová náročnost

Implementace projektu byla časově náročná, přičemž celkový čas potřebný k dokončení projektu převyšoval původní odhady. Komplexita některých úkolů a nečekané problémy během implementace vedly k prodloužení plánovaného časového rámce. Nejvíce času zabral vývoj Ligy a Studium videí. Pak také zabralo dost času zaplnit cloud statickými daty (např. rozvrh tělocvičen. Jinak se vývoj aplikace ukončil 30.4.2024 a testování aplikace započalo 1.5.2024.

Kapitola 6
Testování

Při vývoji aplikace je testování jedním z nejdůležitějších aspektů, které zajistí kvalitu a spolehlivost výsledné aplikace. V rámci testování se zaměřujeme na různé oblasti aplikace, jako je funkčnost, uživatelská přívětivost, bezpečnost a výkon.

Vývojářské testování je prvním krokem v procesu ověřování kvality aplikace a slouží k identifikaci chyb a nedostatků vývoje. Výkon je u mé aplikace zanedbatelný, protože aplikace nedělá žádné náročné výpočty a škálování dat z cloudu je udržované na minimu.

6.0.1 Vývojářské testování

V rámci vývojářského testování jsou prováděny sekvenční testy, které pokrývají různé scénáře použití aplikace a jejich existencí se řídí i samotný vývoj. Tyto testy jsou zaměřeny na ověření správné funkcionality jednotlivých částí aplikace a zajištění, že uživatelé budou schopni provádět požadované akce bez problémů.

Sekvenční testy zahrnují scénáře jako přihlášení, registraci, odhlášení, správu profilu, manipulaci s daty a další interakce s aplikací. Jejich cílem je zajistit, že aplikace funguje přesně tak, jak bylo navrženo, a že uživatelé budou spokojeni s uživatelským zážitkem.

6.0.2 Sekvenční testy

- Scénář 1: Uživatel se přihlásí.
- Scénář 2: Uživatel se zaregistruje.
- Scénář 3: Uživatel se odhlásí.
- Scénář 4: Uživatel může uložit přihlašovací údaje na příště.
- Scénář 5: Uživatel při zadání špatných údajů se nedostane do aplikace.
- Scénář 6: Uživatel si zobrazí novinky a odklikne ho jako přečtený.
- Scénář 7: Uživatel zadá docházku a vidí ji po zadání na obrazovce.
- Scénář 8: Uživatel si může zobrazit nadcházející události.
- Scénář 9: Uživatel si může zobrazit rozvrh tělocvičen.
- Scénář 10: Uživatel si může pustit video z tréninku.
- Scénář 11: Uživatel si může přečíst informace o trenérovi.
- Scénář 12: Uživatel se přihlásí do ligy.
- Scénář 13: Uživatel vyzve jiného uživatele nebo přijme vyzvání.
- Scénář 14: Uživatel zadá výsledek nebo ho přijme.
- Scénář 15: Uživatel se po kliknutí na eshop přesměruje na webový prohlížeč.
- Scénář 16: Uživatel je schopný změnit jazyk aplikace v nastavení.
- Scénář 17: Uživatel si může zobrazit profil a upravit ho.
- Scénář 18: Uživatel si může zobrazit rozvrh jednotlivých tělocvičen.
- Scénář 19: Uživatel si může přečíst informace o jednotlivých sportech a podívat na ukázku sportu.
- Scénář 20: Uživatel si může přečíst informace o tělocvičnách a otevřít ji v navigaci.
- Scénář 21: Uživatel si může přečíst informace o hlavních kontaktech.
- Scénář 22: Trenér může odesílat novinky.

- Scénář 23: Trenér může smazat svoje novinky.
- Scénář 24: Trenér může vytvořit tréninkový plán.
- Scénář 25: Trenér může smazat tréninkový plán.
- Scénář 26: Trenér může přidat videa do tréninkového plánu.
- Scénář 27: Trenér může odebrat videa z tréninkového plánu.
- Scénář 28: Trenér může upravit detaily o sobě.
- Scénář 29: Trenér může vytvořit akci.
- Scénář 30: Trenér může zrušit svojí akci.
- Scénář 31: Admin může vytvářet novinky.
- Scénář 32: Admin může mazat všechny novinky.
- Scénář 33: Admin může vytvořit trenérský účet.
- Scénář 34: Admin může smazat trenérský účet.

6.1 Uživatelské testování

Uživatelské testování je fáze jenž započala 1.5.2024 a zaměřuje se hlavně na ověření požadované funkcionality a uživatelské přívětivosti aplikace. Během této fáze jsou testovací scénáře prováděny reálnými uživateli, kteří poskytují zpětnou vazbu a hodnocení aplikace z hlediska uživatelského zážitku. Co by se jim na aplikaci líbí (značeno „+“), co se jim na aplikaci nelíbí (značeno „-“), problémy při scénáři (značeno „#“) a nápady resp. poznámky (značeno „*“).

6.1.1 Tester 1

Tester 1 je můj hlavní kontakt v Choketopus Gymu. Jeho zpětná vazba je pro můj vývoj jedna z nejdůležitějších. Testuje aplikaci z pohledu admina, trenéra a běžného uživatele.

Seznam scénářů s poznámkami:

```
Scénář 7: Uživatel zadá docházku a vidí ji po zadání na obrazovce.
    * přál bych si vidět docházku účastníků
    - vidět množství lidí co již zadali docházku

Scénář 10: Uživatel si může pustit video z tréninku.
    * přidat platební bránu na některé plány

Scénář 13: Uživatel vyzve jiného uživatele nebo přijme vyzvání.
    + Liga je za mě super a dostačující pro to co potřebuji v ní dělat

Scénář 17: Uživatel si může zobrazit profil a upravit ho.
    * doplnit profil o variabilní symbol pro platby

Bonusové poznámky:
    * vidět záznam plateb v rámci aplikace až budou platby implementovány
    z pohledu uživatele i admina
    - funkce trenérů mi přijde zbytečná
```

6.1.2 Tester 2

Tester 2 je 22letý fotbalista, policista a můj nejlepší přítel, který se nabídl, že aplikaci otestuje. Jeho zkušenost s mobilními technologiemi je nadprůměrná oproti běžnému uživateli. Testuje aplikaci z pohledu běžného uživatele.

Seznam scénářů s poznámkami:

```
Scénář 6: Uživatel si zobrazí novinky a odklikne ho jako přečtený
    * na hlavní obrazovce zobrazit informaci kolik nových karet není
    zobrazeno

Scénář 8: Uživatel si může zobrazit nadcházející události.
    # prošlé události se nemažou automaticky

Scénář 10: Uživatel si může pustit video z tréninku
    * popisky jednotlivých videí namísto Video 1, Video 2, atd.

Scénář 19: Uživatel si může přečíst informace o jednotlivých sportech a
podívat na ukázku sportu.
    * přál bych si vidět nějaké shrnutí a představení společnosti
    - ve sportech jsou všude stejní trenéři
```

6.1.3 Tester 3

Tester 3 je můj 27letý bratr, který používá převážně Android zařízení a je zvyklí na nativní aplikace pro Android. Testuje aplikaci z pohledu běžného uživatele.

Seznam scénářů s poznámkami:

```
Scénář 1: Uživatel se přihlásí.
    - nutnost se přihlásit při každém spuštění aplikace
    * zapamatovat uživatele a rovnou při spuštění přejít do hlavní
    nabídky

Scénář 1, 2 a 3:Uživatel se přihlásí, zaregistruje a odhlásí.
    - po registraci přemazat již uložené informace o přihlášeném
    předchozím uživateli

Scénář 11: Uživatel si může přečíst informace o trenérovi.
    + příjemné zobrazení informací

Bonusové poznámky:
    * poznámkový blok pro trenéra na rychlé poznámky při tréninku
    * na hlavním menu přidat klikatelnou lištu s nejnovější akcí
    # šipka zpět na Androidu by měla vracet na úvodní obrazovku a ne na
    registrační formulář
    * ,,Nemáte ještě účet?" nahradit něčím jako ,,Chcete se přidat k nám
    do týmu?", působí to příjemněji a navozuje pocit přátelskosti
    * menší tlačítka na hlavní obrazovce
    + krásné zpracování designu (UI)
```

6.1.4 Tester 4

Tester 4 je můj vrstevník a kolega programátor. Jeho zpětná vazba bude více z technického hlediska. Testuje aplikaci z pohledu běžného uživatele.

Seznam scénářů s poznámkami:

```
Scénář 3: Uživatel se odhlásí.
    - odhlášení uživatele je v nastavení, preferoval bych ho někde v
    Draweru

Scénář 7: Uživatel zadá docházku a vidí ji po zadání na obrazovce
    - na kalendář se dá klikat ale nic to nedělá, zbytečná funkce

Scénář 16: Uživatel je schopný změnit jazyk aplikace v nastavení
    + chválím práci navíc s českou a anglickou lokalizací aplikace

Scénář 20: Uživatel si může přečíst informace o tělocvičnách a otevřít ji
v navigaci.
    + líbí se mi možnost vyhledat tělocvičnu přes navigaci rovnou z
    aplikace
```

6.1.5 Tester 5

Tester 5 je 23letý student, který má nadhled do vývoje ve Flutteru a používá hlavně iOS zařízení. Testuje aplikaci z pohledu admina a běžného uživatele.

Seznam scénářů s poznámkami:

```
Scénář 1: Uživatel se přihlásí.
    # po přihlášení je statická velikost tlačítek na hlavní obrazovce
    (problém s různými rozlišeními mobilních zařízení)

Scénář 10: Uživatel si může pustit video z tréninku.
    + spouštění videa rovnou z aplikace
    # po ukončení videa v horizontálním režimu a jeho ukončení, zůstane
    aplikace v horizontálním režimu


Scénář 17: Uživatel si může zobrazit profil a upravit ho.
    # informace o lize uživatele je statická a není propojená s
    uživatelem
```

6.2 Výsledky testování

Ve fázi uživatelského testování byla aplikace podrobena zkoumání reálnými uživateli, kteří poskytli cennou zpětnou vazbu a nahlédli do uživatelského zážitku z používání aplikace. Během testování byly identifikovány různé problémy a nedostatky, které je třeba vyřešit, stejně jako pozitivní aspekty, které bylo vhodné zachovat a zdokonalit.

Níže je uvedena tabulka, která shrnuje identifikované problémy během testování a navrhovaná řešení:

Tyto identifikované problémy a navrhovaná řešení budou prioritou pro další vývoj aplikace s cílem zlepšit uživatelský zážitek a celkovou funkčnost aplikace. Díky úzké

Problém	Řešení
Šipka zpět na Androidu	Upravit navigaci aplikace
Opětovná nutnost přihlášení	Přesměrovat rovnou na hlavní nabídku
Prošlé události	Mazání prošlých událostí ve Firebase.
Generické názvy videí	Udělat z videí kolekci a přidat jim název
Názvy trenérů ve sportech	Editovat proměnné v projektu
Velikost tlačítek na hlavním menu	Responzivita podle rozlišení obrazovky.
Horizontální režim po ukončení videa	Po ukončení videa nastavit vertikální režim.
Statické informace o lize	Propojit informace o lize s cloudem.

Tabulka 6.1. Tabulka identifikovaných problémů a jejich řešení.

spolupráci s uživateli a jejich zpětné vazbě mám cenné informace pro další zdokonalení aplikace.

Kapitola 7
Závěr

V této práci jsme se zabývali vývojem aplikace pro sportovní centrum zaměřené na bojová umění (sebeobranu). Proces vývoje aplikace zahrnoval analýzu požadavků, návrh architektury, implementaci a uživatelské testování. Během této práce jsem se toho spoustu naučil o vývoji pro mobilní zařízení. Vývoj ve Flutteru mě uchvátil a samotná práce v něm mě bavila. Poprvé jsem měl i tu možnost pracovat s cloud NoSQL databází Firebase.

Aplikace umožňuje zadávat docházku, sledovat tréninková videa, připojovat se do ligy, zápasit a mnoho dalšího. Po dohodě s vedoucím práce nebyly implementovány dvě funcionality:

- První funkcionalita je poskytnout trenérovi možnost nahlížet do docházky svých studentů. Tato funkcionalita je velice užitečná a její implementace je vítaná v budoucích verzích aplikace.
- Druhá funkcionalita se týká možnosti hodnocení trenérem a tréninkových plánů. Tato funkcionalita hodnocení trenéra se zatím jeví zbytečná, neboť zpětnou vazbu dostane zákazník na tréninku. Ovšem hodnocení tréninkových plánů, dává z implementačního hlediska smysl a bude uskutečněno v další versi aplikace.

Nejvíce času zabralo vypracování systému ligy, pro kterou jsem neměl žádnou předlohu ani popis, jak by měla fungovat. To je veliká nevýhoda z pohledu implementace, protože musím celý systém vymyslet sám. S konečným výsledkem jsem ovšem spokojený nejvíce ze všech funkcionalit aplikace.

7.1 Budoucnost aplikace

Budoucí vývoj a komerční využití aplikace je nadějný. Aplikace má potenciál k dalšímu rozvoji a zdokonalení. V budoucnosti by bylo vhodné zaměřit se na následující oblasti:

- Opravit již implementované funkcionality na jejichž problémy se narazilo při testování aplikace (uvedené v tabulce 6.1)
- Zdokonalit design uživatelské rozhraní s uživatelem, které bylo místy urychlené, pro příjemnější použití aplikace.
- Rozšíření možností správy událostí centra pro sdělení konkrétních informacích o akci. Například otevření navigace na místo konání akce, přidat cenu akce, možnost přihlásit se na akce a zaplatit za akci rovnou z aplikace.
- Přidat ke Controlleru Servis a Repository design patterny pro lepší udržitelnost aplikace.

Aplikace je přínosem pro společnost Choketopus Gym a její zákazníky. Sportovní centrum má nyní platformu pro poskytování nových služeb, které dříve nemohly poskytovat (liga, sledování výukových videí). Uživatelé se nyní mohou vzdělávat z pohodlí doma nebo vlastní tělocvičny, sledovat různé akce a zdokonalovat se ve sportu, který je baví.

Literatura

[1] Inc. Figma. *Figma*. 2024.
https://www.figma.com/.

[2] Microsoft. *Visual Studio Code*. 2024.
https://code.visualstudio.com/.

[3] Google. *Firebase*. 2024.
https://firebase.google.com/.

[4] Rap Payne. *Beginning App Development with Flutter* . Apress , 2019 . ISBN 978-1-4842-5181-2.

[5] Google. *Flutter Dokumentace*. 2024.
https://docs.flutter.dev/.

[6] Pooja Bhaumik. *Linkedin Learning Flutter Course*. 2022.
https://www.linkedin.com/learning/flutter-essential-training-build-for-multiple-platforms.

[7] *Flutter widgets documentation*. 2024.
https://docs.flutter.dev/reference/widgets.

[8] *Choketopus*. 2024.
https://choketopusgym.cz/.

[9] *Choketopus eshop*. 2024.
https://choketopus.com/.

Příloha A
Slovník

API	Aplikační rozhraní
Android	Operační systém pro některá mobilní zařízení
Cloud	Databázové uložiště, které vlastní nějaká společnost
Figma	Webová technologie pro tvorbu uživatelského rozhraní
Firebase	Cloud poskytován společností Google
Framework	Sada nástrojů již vytvořená jinými programátory
Gitlab	Verzovací systém
HTTP	Hypertext Transfer Protocol
IDE	Integrované vývojové prostředí
iOS	Operační systém pro Apple mobilní zařízení
Kotlin	Programovací jazyk pro Android
MacOS	Operační systém pro Apple počítačová zařízení
MVP	Minimální funkční prototyp
Open-source	Software s odhaleným zdrojovým kódem veřejnosti
UI	Uživatelské rozhraní
VSCode	Visual Studio Code
XCode	Vývojové prostředí pro iOS