VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2017 Patrik Potoček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

SOUTĚŽ VE SPORTOVNÍM SÁZENÍ

COMPETITION IN SPORTS BETTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **Patrik Potoček**
AUTHOR

VEDOUCÍ PRÁCE **Ing. Vítězslav Beran, Ph.D.**
SUPERVISOR

BRNO 2017

Abstrakt

Táto práca se zoberá návrhom a následným vývojom mobilnej aplikácie pre platformu iOS, ktorá umožňuje jej užívateľom súťažiť v športovom stávkovaní. Výnimočnosť tejto aplikácie spočíva v novom uhle pohľadu na športové tipovanie, kde užívatelia súperia medzi sebou a bez nutnosti rizika finančnej straty. Princíp súťaže spočíva v zostavení ticketu z tipov na športové stretnutia a následnom vložení do hernej miestnosti za poplatok vo forme virtuálnych žetónov.

Abstract

This thesis describes design and implementation of mobile application for iOS, which allows its users to compete in sports betting. The uniqueness of this application rests in new point of view on sports betting where users compete with each other and do not have to face the risk of financial lost. The principle of this competition rest in creation of ticket which consists of individual bets on sports events and placing this ticket to game rooms for fee in form of virtual coins.

Kľúčové slová

iOS, mobilná aplikácia, športové stávkovanie, užívateľské rozhranie, swift,

Keywords

iOS, mobile application, sports betting, user interface, swift

Citace

Potoček Patrik: Súťaž v športovom stávkovaní, bakalářská práce, Brno, FIT VUT v Brně, 2017

Súťaž v športovom stávkovaní

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vítězslava Berana, Ph.D.. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Patrik Potoček
13. 05. 2017

Poďakovanie

Rád by som poďakoval vedúcemu práce pánu Ing. Vítězslavu Beranovi, Ph.D. za cenné rady počas konzultácii, trpezlivosť a podporu pri riešení bakalárskej práce.

Kapitola 1

1. Úvod

Cieľom tejto práce je vytvoriť návrh mobilnej aplikácie, ktorá umožňuje jej užívateľom súťažiť v športovom stávkovaní a následne realizovať tento návrh pre mobilnú platformu iOS.

Súčasný systém športového stávkovania robí z tejto aktivity gambling. Systém športového stávkovania zahŕňa dve zainteresované strany a to užívateľa a stávkovú kanceláriu. Stávková kancelária vydáva kurzy, ktoré majú reflektovať pravdepodobnosť úspechu daného tipu. Čím vyšší je kurz, tým vyššia je pravdepodobnosť neúspechu tipu, ale zároveň vyššia výhra v prípade, že je tip úspešný. Možná výhra sa s kombináciu rôznych tipov zvyšuje, ale rovnako narastá aj riziko prehry, a to práve kvôli bežnému systému stávkovania, podľa ktorého čo i len jeden neúspešný tip znamená pre užívateľa prehru. Problém v tomto systéme vidím práve v súťaži užívateľa a stávkovej spoločnosti, pričom stávková spoločnosť navrhuje systém teda pravidlá hry a zároveň aj reguluje samotné kurzy. V mojom návrhu systém slúži len na sprostredkovanie informácií a vyhodnotenie súťaže medzi samotnými užívateľmi bez záujmu výhry jednej či druhej strany. Navyše jej užívatelia nebudú súťažiť o reálne peniaze, ale virtuálne žetóny, ktoré môžu získať v aplikácii zdarma.

V kapitole č. 2 pojednávam o vývoji mobilných aplikácii prevažne z pohľadu softwarovej architektúry, tvorbe užívateľského rozhrania a jej užívateľoch a stručne popisujem technologické možnosti riešenia aplikácie. Kapitole č. 3 je o návrhu systému aplikácie a užívateľského rozhrania. Definuje podrobnú špecifikáciu celej aplikácie spolu s jej cieľovou skupinou. Obsahuje návrh informačnej štruktúry, užívateľského rozhrania a serverovej časti. Kapitola č. 4 sa zaoberá implementáciu návrhu klientskej a serverovej časti aplikácie pomocou aktuálnych technológii a následnom užívateľskom testovaní. Záver kapitoly vyhodnocuje toto testovanie a diskutuje o ďalších možnostiach vývoja.

Kapitola 2

2. Teória

2.1. Mobilné aplikácie a ich vývoj

Príchodom chytrých telefónov (z anglického “smartphone") medzi bežných ľudí sa postupne mení technologický svet. V dnešnej dobe, kedy má už väčšina ľudí počas celého dňa prístup k zariadeniu s výpočetnou silou väčšou, ako malo ľudstvo k dispozícií počas prvého letu na mesiac, sú otvorené nové možnosti vývoja softwaru, ktorý je ihneď prístupný a spája veľké množstvo užívateľov. Štatistika z Marca roku 2017 uvádza, že užívatelia platformy Android mali k dispozícií 2 800 000 aplikácií a užívatelia platformy iOS 2 200 000 aplikácií [1].

Spomedzi všetkých mobilných operačných systémov na trhu sú momentálne najdominantnejšie iOS a Android. Tieto operačné systémy používajú pre vývoj rôzne platformové technológie a programovacie jazyky. Preto aplikácia, ktorá má byť prístupná na oboch platformách nezdieľa zdrojový kód. Z tohoto dôvodu vznikli rôzne prístupy k vývoju mobilných aplikácii a to hlavne prístup natívny a multiplatformový. Multiplatformový vývoj, prevažne realizovaný pomocou webových technológii, umožňuje sprístupniť jednu aplikáciu na rôzne platformy bez nutnosti individuálneho programovania. Každý operačný systém má však svoje vlastné návrhové vzory na ktoré je jej užívateľ navyknutý a hybridné rozhranie často porušuje vzory obidvoch operačných systémov. Existujú však druhy aplikácii, ktoré nemusia dodržovať tieto návrhové vzory ako napríklad 3D hry kde je multiplatformový výhodný až žiadúci. Pre druh mojej aplikácie je multiplatformový vývoj nevhodný a volím si prístup vývoja natívnej aplikácie pre operačný systém iOS.

iOS je operačný systém UNIXového typu a vychádza z desktopového operačného systému macOS X. Vývoj aplikácií na tejto platforme sa nezaobíde bez sady nástrojov s názvom iOS SDK (iOS Software Development Kit). iOS SDK je kolekcia nástrojov pre vývoj mobilných aplikácií, ktorého najdôležitejšou komponentou je Xcode. Xcode je IDE (z anglického “integrated development environment”), ktoré združuje kompilátor, nástroje na grafickú tvorbu aplikácie, simulátory systémov a mnohé ďalšie nástroje umožňujúce vývoj.
Systém iOS sa rozdeľuje na štyri základné vrstvy z ktorých najrelevantnejšou z pohľadu mojej práce je vrstva Cocoa Touch.

Cocoa Touch - jedná sa o najvyššiu vrstvu systému. Táto vrstva okrem iného zabezpečuje odchytávanie dotykových gest a vykresľovanie obrazovky, správu vlákien a údržbu súborov.

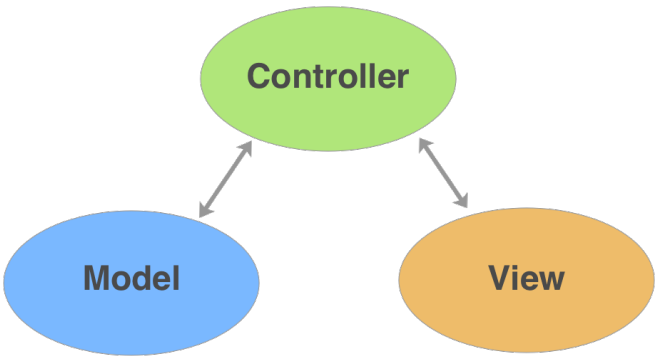
Disponuje frameworkami, ako Foundation a UIKit, ktoré sú základnými stavebnými kameňmi pri vývoji.

UIKit - framework, ktorý ponúka dôležitú infraštruktúru potrebnú pre konštrukciu a správu iOS aplikácií. Ponúka hierarchiu obrazovky a okna, ktorá je potrebná pre údržbu užívateľského rozhrania. Ďalej disponuje grafickými a interaktívnymi komponentami pomocou ktorých je užívateľské rozhranie zostavené.

2.2. Softwarová architektúra mobilnej aplikácie

Softwarová architektúra mobilných aplikácií je neoddeliteľnou súčasťou vývoja. Zvolením vhodnej architektúry sa zabezpečí udržovateľnosť, flexibilita a stabilita aplikácie. V nasledujúcej časti rozoberiem štyri druhy architektúr, použiteľných pre vývoj mobilných aplikácií.

MVC (Model, View, Controller) - architektúra najčastejšie spájaná s vývojom pre iOS. Sama firma Apple ju odporúča a prezentuje vo svojej dokumentácií. Dôvodom je jednoduchá abstrakcia a tým aj rýchle pochopenie a adaptovanie do riešení. Architektúra sa opiera o zapúzdrenie dátového modelu, užívateľského rozhrania a riadiacej logiky aplikácie. Zapúzdrené komponenty spolu komunikujú prostredníctvom zaužívaných návrhových vzorov ako je delegácia či notifikácie. Každá jedna obrazovka v aplikácii by mala mať svoj vlastný vzor MVC, ktorý je zobrazený na obrázku 2.1.



Obrázok: 2.1 MVC

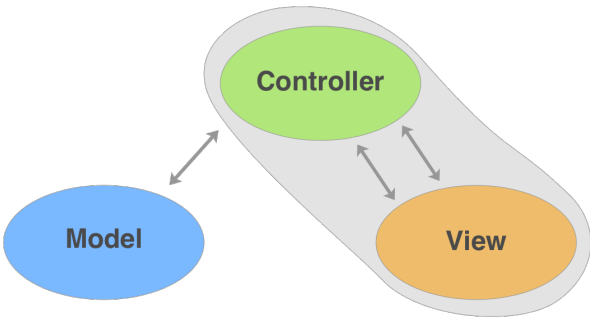
Úlohy jednotlivých zložiek MVC:

- Model slúži na správu s dátami aplikácie. Je zodpovedný za ukladanie, spracovanie a servírovanie dát riadiacej logike anglicky Controlleru. Model by mal komunikovať výhradne s riadiacou jednotkou.
- View sa stará o reprezentáciu dát aplikácie prostredníctvom užívateľského rozhrania a taktiež komunikuje výhradne s riadiacou jednotkou.

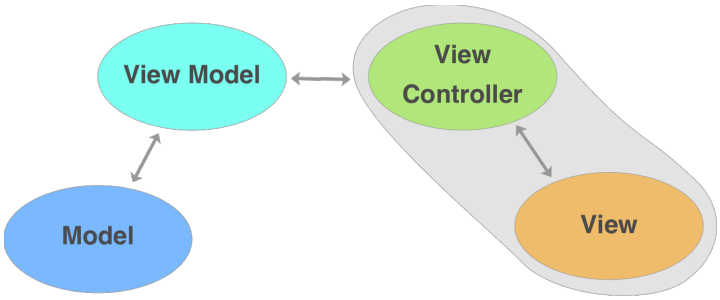
- Controller má na starosti všetko ostatné. Je to najvyťaženejší prvok v MVC a zároveň mozog, ktorý spája dátový model a rozhranie. Má na starosti odchytávanie vstupov z užívateľského rozhrania, navigáciu medzi obrazovkami a mnohé ďalšie.

V zhrnutí je MVC vstupná brána do problematiky architektúr a má využitie nielen vo vývoji mobilných aplikácií. Avšak pre moje požiadavky je nedostatočná hlavne z pohľadu takmer nemožnej testovateľnosti a malého rozloženia logiky, ktorá sa takmer celá nachádza v jednej komponente.

MVVM (Model, View, ViewModel) - softwarová architektúra MVVM priamo vychádza z MVC a rieši problém príliš veľkej zodpovednosti riadiacej jednotky. Z obrázka 2.1, ktorý znázorňuje rozdelenie jednotlivých komponent v MVC je zrejmé, že controller a view sú oddelené a nemali by do seba zasahovať. Aj napriek tejto rozdielnosti sú tieto dve komponenty úzko späté, a to z dôvodu, že controller často obsahuje tzv. prezentačnú logiku. Obrázok 2.2 zobrazuje architektúru MVC v praxi. V MVVM sú operácie ako prevádzanie hodnôt modelu do formy prezentovatelnej užívateľským rozhraním, zapúzdrené vo view modely. Obrázok 2.3 tak zobrazuje koncept architektúry MVVM, kde controller je odprostený od zobrazovacej logiky, ktorá je presunutá v novej komponente s názvom view model. Architektúra MVVM upravuje MVC a čiastočne rieši jeden z problémov, a to príliš veľkú zodpovednosť controlleru. Testovateľnosť je o málo lepšia, ale stále má controller príliš veľa zodpovednosti, ako napríklad riadenie navigácie medzi jednotlivými obrazovkami.



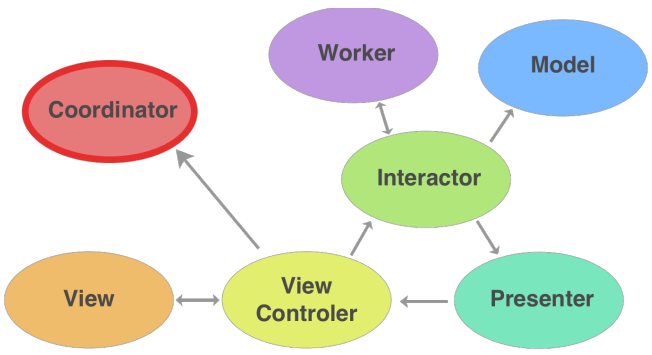
Obrázok: 2.2 MVC v praxi



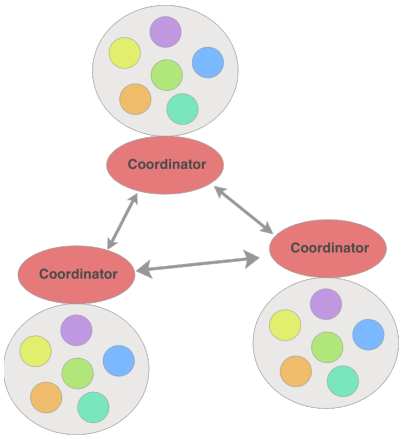
Obrázok: 2.3 MVVM

Clean swift - najmladšia softwarová architektúra spomedzi spomenutých, ktorá sa sústreďuje primárne na vývoj iOS aplikácií. Vychádza z architektúry VIPER (View, Interactor, Presenter, Entity, Routing), ktorá spadá pod typ tzv. Clean Architecture [2]. Clean swift je komplexná architektúra, ktorá presne rozdeľuje komponenty podľa ich účelu. Obsahuje sedem vrstiev, a to View, Controller, Interactor, Presenter, Model, Coordinator a Worker. Každá vrstva má jasne definované rozhranie, pomocou ktorého môže komunikovať s ostatnými. Všetkých sedem vrstiev tak spolu tvorí scénu, ktorá reprezentuje jednu obrazovku aplikácie.
Úlohy jednotlivých vrstiev:
View: tak ako v predošlých architektúrach jedná sa o užívateľské rozhranie aplikácie.
Controller: vrstva zabezpečujúca komunikáciu s užívateľským rozhraním. Je zodpovedná za prijímanie vstupov užívateľa a zobrazovanie výstupu z presentra.
Interactor: mozog celej architektúry, sú v ňom implementované všetky potrebné výpočty, a ako jediná vrstva má prístup k dátovým modelom, ktoré sprostredkovávajú workery.
Worker: je komponenta, ktorá má za úlohu servírovať dáta interactoru. Ak aplikácia potrebuje aktualizovať dáta zo serveru sú workery objekty, ktoré zastrešujú túto komunikáciu
Presenter: prijíma výstup z interactoru a prevádza ho do formy prezentovateľnej užívateľským rozhraním. Výstup z presentru sa tak dostáva späť do controlleru.
Coordinator: jedná sa o vrstvu zodpovednú za navigáciu medzi jednotlivými scénami aplikácie. Komunikuje výhradne s controllerom.

Tok dát v tejto architektúre tak pripomína akýsi uzavretý kolobeh, ktorý je znázornený na obrázku 2.4. Clean swift architektúra svojou vysokou komplexnosťou a dekompozíciou rieši problém testovateľnosti aplikácie, pretože každá vrstva má jasný vstup a výstup, dajú sa nezávisle na sebe testovať. Zároveň predstavuje koncept coordinatorov, ktoré elegantne riešia navigáciu medzi obrazovkami aplikácie, viz obrázok 2.5. Táto architektúra má veľký potenciál pre vývoj v početných tímoch. Nevýhodou je veľké množstvo vrstiev a teda komplexnejší a obšírnejší zdrojový kód.



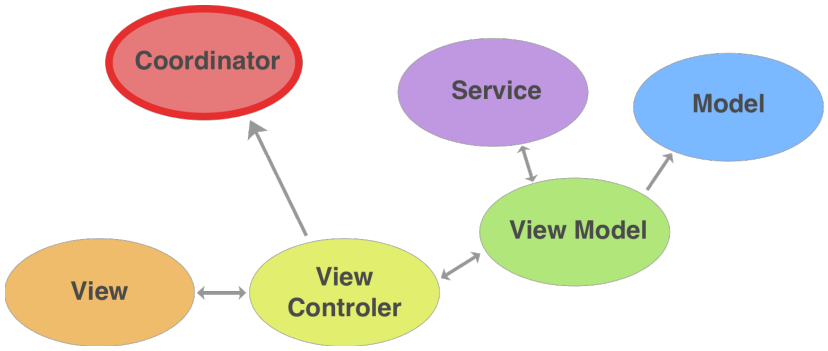
Obrázok: 2.4 Architektúra Clean



Obrázok: 2.5 Koncept coordinatorov

MVVM-C: jedná sa o koncept architektúry, ktorú som pomáhal navrhnúť a vznikla spojením všetkých troch vyššie spomenutých architektúr. Cieľom tohoto konceptu je vziať to najlepšie z každej architektúry a vytvoriť tak ľahko testovateľný a dekomponovaný systém s menej komplexnou abstrakciu ako pri architektúre clean swift. MVVM-C je rozšírením architektúry MVVM o coordinator a prerozdelením úloh niektorých z vrstiev. Coordinator je podobne ako v clean swifte zodpovedný za navigáciu medzi jednotlivými scénami, a je vždy vstupným aj výstupným bodom celej scény. View model je v tejto architektúre povýšený na úroveň riadiacej logiky celej scény a má prístup k dátovým modelom. Z controllera sa teda stáva len akási vrstva spájajúca všetky ostatné komponenty obsahujúca prezentačnú logiku. View model na prácu s dátami využíva ďalšiu komponentu s názvom servise, ktorá zaobstaráva úlohy ako napríklad sťahovanie a ukladanie dát. Tok dát a rozdelenie architektúry sú zobrazené na obrázku 2.6.

Architektúra MVVM-C svojou jednoduchou abstrakciou, udržovateľnosťou kódu a dekompozíciou spĺňa všetky moje požiadavky na architektúru, ktorú dodržujem pri vývoji mobilných aplikácií.



Obrázok: 2.6 MVVM-C

2.3. Serverové služby

BaaS (Backend as a Service)

BaaS[1] (z anglického Backend as a Service) je cloudová služba, ktorá ponúka riešenie pre zjednodušenie implementácie serverovej časti aplikácie. Väčšina moderných aplikácií ukladá dáta a komunikuje prostredníctvom internetu. Dáta o užívateľoch a ich činnostiach, zdielané dokumenty a podobné musia byť uložené na mieste prístupné všetkým užívateľom, a preto tieto aplikácie vyžadujú samostatné aplikácie bežiace na vzdialenom serveri, ktoré sprostredkujú komunikáciu a prístup k daným dátam. Implementácia týchto serverových aplikácií si vyžduje znalosti z mnohých informačných sfér. Cieľom konceptu BaaS je abstraktovať celý tento proces

[1] Backendless. Video dostupné z: https://backendless.com/what-is-backend-as-a-service.

a umožniť tak vývojárovi znovu nevymýšľať koleso, ale sústrediť sa na riešenie problém súvisiacich s jeho návrhom. BaaS tak predovšetkým ponúka riešenie správy dát a logiku serverovej časti, ale takisto aj push notifikácie, správu autorizácie užívateľov či analýzu dát v aplikácií.

Parse Server
Parse Server[2] je BaaS riešenie vyvinuté spoločnosťou Facebook, ktorá toto riešenie voľne sprístupnila. Parse server využíva MongoDB a je napísaný v NodeJS.

PaaS (Platform as a Service)

PaaS[3] niekedy označované ako aPaaS (aplication platform as a service) je cloudová služba, ktorá jej užívateľom ponúka platformu pre vývoj a údržbu aplikácií. Poskytovateľ tejto služby ponúka za mesačný poplatok hardware a software potrebný pre chod databáz, serverovej logiky, API a mnohé ďalšie nevyhnutné nástroje pre život aplikácie závislej na internete.

Heroku
Heroku[4] je jeden z PaaS poskytovateľov. Výhodou je, že mesačný poplatok je priamo úmerný prevádzke takže ponúka nástroje na vývoj zdarma.

2.4. Užívateľské rozhranie a jeho užívateľ

Každý človek je odlišný fyzicky, mentálne alebo vzdelaním. Existujú však ľudské charakteristiky, ktoré máme spoločné a pre návrh užívateľského rozhrania sú veľmi dôležité. Jedná sa o charakteristiky ako pamäť či schopnosť učenia.

Pamäť

Z pohľadu návrhu užívateľského rozhrania patria medzi jedny z najdôležitejších procesov pamäte rozpoznanie a spomínanie. Pod spomínaním rozumieme prehľadávanie spomienok za účelom nájsť konkrétnu informácie kdež to pod rozpoznávaním chápeme iba rozhodovanie či daná informácia sa už v pamäti nachádza. Skvelým príkladom rozpoznávania je napríklad výber typu fontu písma z ponuky. Človek si nemusí nutne pamätať názov fontu, ale postačí mu vidieť samotnú ukážku daného fontu. Rozpoznávanie je teda menej náročné ako spomínanie.

---

2 Parse server. Dostupné z: http://parseplatform.org.

3 Appenda. Dostupné z: https://apprenda.com/library/paas/iaas-paas-saas-explained-compared.

4 Heroku. Dostupné z: https://www.heroku.com.

Proces premeny krátkodobej spomienky na dlhodobú nazývame učenie. Učenie je pre človeka často namáhavé a preto je potrebné sa snažiť tomuto procesu vyhnúť napríklad použitím známych prvkov, ktoré užívateľ pozná z iných aplikácií. [3]

Cyklus ľudskej aktivity

Donald Norman vo svojom sedem krokovom kognitívnom modely [4] popisuje, ako ľudia narábajú s počítačovými systémami. Táto teória sa opiera o predpoklad, že ciele užívateľa motivujú k akciám a činnosti.
Ako prvé si človek sformuluje cieľ, ktorý následne transformuje na zámer previesť akciu. Tento zámer ďalej interpretuje na sekvenciu krokov pre dosiahnutie cieľa. Výsledky jeho akcie sú vnímané a pochopené. V dobrom návrhu užívateľského rozhrania je potrebné aby užívateľ čo najrýchlejšie dosiahol požadovaný cieľ. [3]

Mentálna mapa

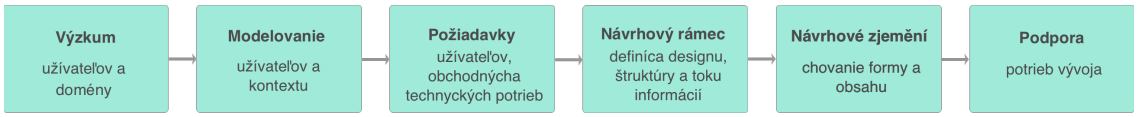
Mentálna mapa alebo aj mentálny model je vlastná interpretácia užívateľa ako chápe nejaký systém [5]. Užívateľ si vytvorí vlastný pojem o tom, ako niečo funguje. Pri návrhu rozhrania vzniká aj reprezentačný model, ktorý sa snaží pomôcť užívateľovi si ten svoj sformulovať a schovať komplexnosť systému. Tento mentálny model tak človeku pomáha pochopiť kroky k prevedeniu jeho cieľov. Inými slovami, užívateľ bude očakávať chovanie aplikácie na základe jeho vlastného mentálneho modelu. Pokiaľ sa užívateľov mentálny model zhoduje s chovaním aplikácie, môžeme označiť rozhranie za intuitívne.

2.5. Tvorba užívateľského rozhrania

Už viac ako tridsať rokov je známy návrh orientovaný na užívateľa (z anglického “User Centered Design”), no aj napriek tomu stále prevažuje vývoj zameraný na funkcionalitu programu ako na potreby užívateľa [6]. Jedným z dôvodov je, že orientácia na užívateľa je finančne a časovo náročná [7]. Užívateľský výskum, analýza, prototypovanie, testovanie použiteľnosti a podobné metódy vývoja vyžadujú mnoho času, prostriedkov a peňazí. Na druhú stranu prínos týchto metód je značný a potrebný hlavne v oblasti mobilných aplikácií kvôli vysokej konkurencií.

Návrh zameraný na ciele

Návrh zameraný na ciele užívateľa (z anglického “Goal-Directed Design”) sa zameriava na zistenie, kto bude užívateľom aplikácie, a prispôsobuje tak návrh na jeho potreby a kontext, v ktorom bude výsledný produkt používaný [6]. Dôležité je identifikovať ciele užívateľa prostredníctvom užívateľského výskumu. Návrh zameraný na ciele je možné rozdeliť do šiestich fázy zobrazených na obrázku 2.7.



Obrázok 2.7 Proces tvorby GUI podla Goal-Directed Design

UX Design

Existuje mnoho definícií UX designu. Najviac sa stotožňujem s Tomerom Sharonom, ktorý tvrdí: "*UX design je umenia a veda vytvárania pozitívnych emócií prostredníctvom interakcie s produktom.*" [8] Preto je pri návrhu UX nesmierne dôležité poznať cieľového užívateľa produktu. Na spoznanie svojho cieľového užívateľa slúži užívateľský výskum, ktorý je jeden z prvých a kľúčových krokov v návrhu rozhrania a systému. Pokiaľ nieje definovaný cieľový užívateľ, je návrh bezcieľny a často mylný. Jedným z techník na definíciu cieľového užívateľa je využitie person.

Persona - dokument, ktorý popisuje cieľového užívateľa produktu. Relevantnosť informácií závisí od produktu. V mojom prípade ma zaujímajú informácie ako vzťah k športu, sociálny status, vzdelanie alebo solventnosť. Z týchto informácií je možné získať, čo je pre užívateľa a produkt dôležité.

Testovanie

Poznanie užívateľa však ešte neznamená, že navrhnuté rozhranie či systém spĺňa potreby užívateľa. Vtedy je prevedené užívateľské testovanie. Prvý krok testovania je zadefinovať si čo vlastne chcem od užívateľa zistiť. Následne vytvoriť sadu otázok, ktorých odpovede ponúknu informácie relevantné k mojim cieľom testovania.

Ďalšou metódou je testovanie s využitím techniky rozmýšľania nahlas (z anglického "think loud"). Užívateľ dostane veľmi stručné vysvetlenie, o čom aplikácia je, a následne je sledované, ako s aplikáciu narába. Počas práce s aplikáciou nahlas rozpráva svoje dojmi a myšlienky.

Gamifikácia

Základom UX návrhu je použiteľnosť, avšak gamifikácia prináša konceptu úplne novú vrstvu, a to zábavu. Ľudia sú odjakživa poháňaný súťaživosťou a odhodlaním uspieť. Zakomponovaním gamifikácie do návrhu je možné získať potencionálnu motiváciu pre užívateľov používať daný produkt. Správny návrh gamifikácie je taký, ktorý ponúka užívateľovi ciele a zároveň motivuje užívateľa k ich dosiahnutiu, pričom vedľajším efektom jeho snahy o splnenie týchto cieľov sú záujmy tvorcov aplikácie. [9] Pomocou gamafikácie je možné motivovať užívateľa sa aj naďalej k aplikácií vraciať. Pretože môj návrh aplikácie je celý založený na súťažení medzi jej užívateľmi, zakomponovanie gamafikácie v podobe rôznych rebríčkov a odmeňovaní podporí motiváciu užívateľov naďalej aplikáciu požívať a popritom sa zabávať.

Užívateľský výskum

Moja aplikácia je o súťažení v športovom stávkovaní medzi užívateľmi, preto pri určovaní cieľovej skupiny užívateľov je potrebné sa zamerať aj na psychologický aspekt súťaženia. Mobilná aplikácia sa bude primárne orientovať na mladšiu skupinu užívateľov. Mladý ľudia majú najväčšie predpoklady pre využitie tejto aplikácie. Sú športovo aktívny alebo šport pravidelne sledujú a majú svojho obľúbeného športovca či tým, ktorému fandí. Aby som mohol dostatočne objasniť prečo práve mladí ľudia, pojmem užívateľov aplikácie ako skupinu, presnejšie neformálnu skupinu. Budem pri tom vychádzať z poznatkov sociálnej psychológie.

Skupinová príslušnosť sprevádza človeka celý život. Po celú dobu jeho života sa človek cíti byť príslušníkom nejakej skupiny a túto príslušnosť vyhľadáva. Skupinová príslušnosť je životnou nutnosťou a dáva pocit spolupatričnosti. Momentálne nemám na mysli ani skupiny veľké, ako je napríklad štát alebo škola, ani skupiny primárne, ako sú rodina, partia kamarátov, ani skupiny formálne, ako je napríklad pracovný kolektív. Pre moje potreby je podstatné fungovanie neformálnych skupiny. Malé neformálne skupiny tvoria osoby, ktoré spojuje nejaká činnosť, istým spôsobom sú spolu v interakcii alebo majú spoločný cieľ. [10] Do takýchto skupín ľudia vstupujú každý deň, môžu to byť ľudia v autobuse, alebo ľudia nachádzajúci sa na jednom vianočnom trhu, či užívatelia jednej mobilnej aplikácie.

Základnou funkciou malých skupín je, že človeku umožňujú uspokojovať sociálne potreby, tzv. socializácia. Sociálny psychológovia v tomto zmysle hovoria taktiež o porovnávacej funkcií, potrebe sa s niekým porovnávať a súťažiť. Je teda podstatné, aby mnou navrhnutá mobilná aplikácia ponúkala prostredie, kde nefiguruje žiadna druhá strana, konkrétne stávková kancelária. Užívatelia aplikácie by teda tvorili malú neformálnu skupinu, kde sú si všetci rovní, dodržujú rovnaké pravidlá a vstupujú do spoločenstva so spoločným cieľom, s cieľom vyhrávať. V neposlednom rade ich spojuje záľuba v športovom stávkovaní.

Hoci som spomenul, že z pohľadu mojej aplikácie je dôležité hlavne fungovanie neformálnej skupiny, užívatelia budú mať možnosť zakladať aj tzv. privátne miestnosti, do ktorých môžu pozvať svojich známych, a súťažiť tak vrámci primárnej skupiny, teda skupiny svojich známych. Táto funkcionalita prináša do súťaže väčšie emócie.

Kapitola 3

3. Návrh riešenia

V tejto kapitole sa venujem vlastnému návrhu aplikácie. Popisujem ciele a podrobnú špecifikáciu riešenia spolu s návrhom v závislosti na predošlej teoretickej časti.

3.1. Cieľ práce

Súčasný systém športového stávkovania zahŕňa dve zainteresované strany, a to užívateľa a stávkovú kanceláriu. Stávková kancelária vydáva kurzy, ktoré majú reflektovať pravdepodobnosť úspechu daného tipu. Čím vyšší je kurz, tým vyššia je pravdepodobnosť neúspechu tipu, ale zároveň vyššia výhra v prípade, že tip je úspešný. Možná výhra sa s kombináciu rôznych tipov zvyšuje, ale rovnako narastá aj riziko prehry, a to práve kvôli bežnému systému stávkovania, podľa ktorého čo i len jeden neúspešný tip znamená pre užívateľa prehru. Problém v tomto systéme vidím práve v súťaži užívateľa a stávkovej spoločnosti, pričom stávková spoločnosť navrhuje systém, teda pravidlá hry, a zároveň aj reguluje samotné kurzy. Navyše aplikácie stávkových spoločností ponúkajú okrem športového tipovania aj hazardné hry, ako sú poker či hracie automaty. Aj vďaka tomuto je športové tipovanie označované za gambling či hazard. V mojom návrhu systém slúži len na sprostredkovanie informácií a vyhodnotenie súťaže medzi samotnými užívateľmi bez záujmu výhry jednej či druhej strany.

Cieľom práce je vytvoriť návrh mobilnej aplikácie spolu so systémom, ktorý by bol konkurencieschopnou alternatívou pre súčastný systém športového tipovania. Systém ktorý zmení označenie športového tipovania za hazard.

3.2. Špecifikácia

Mobilná aplikácia s názvom DailyBet umožňuje jej užívateľom tipovať výsledky športových zápasov a následne prostredníctvom týchto tipov súťažiť s ostatnými užívateľmi. Súťažiť však nemusí o reálne peniaze, ale o žetóny, ktoré môže užívateľ v aplikácií získať. Pred popisom priebehu súťaže je potrebné zadefinovať pojmy, s ktorými budem ďalej pracovať.

Kurz: Hodnota vyjadrujúca pravdepodobnosť úspechu. Čím vyššia je hodnota, tým je pravdepodobnosť úspechu nižšia a naopak.

Zápas: Konkrétny zápas, ktorý sa skladá z domáceho a hosťovského týmu a kurzov pre výhru jedného z týmov poprípade remízy.

Tip: Skladá sa zo zápasu a tipu výsledku tohoto zápasu. Jeho výsledná hodnota je rovná kurzu zvoleného tipu zápasu.

Ticket: Zostavuje ho užívateľ a skladá sa zo 7 tipov a výsledného skóre, ktoré je rovné násobkom jednotlivých hodnôt tipov.

Herná miestnosť: Miesto, kde užívateľ môže vložiť svoj zostavený ticket. Herné miestnosti sa líšia svojimi parametrami:

- maximálny počet ticketov
- hodnota stávky
- príslušnosť k druhu hry.

Každá herná miestnosť má tri stavy: miestnosť otvorená, miestnosť uzavretá a miestnosť vyhodnotená.

Otvorená herná miestnosť: Užívatelia majú možnosť do tejto miestnosti vkladať svoje tickety. Jeden užívateľ môže mať v miestnosť vložený práve jeden ticket. Tento stav miestnosti je predvolený.

Uzavretá herná miestnosť: Každá herná miestnosť má svoj čas uzávierky, ktorý je rovný 00:00 GMT. Po naplnení tohoto času sa miestnosť uzavrie a žiadny užívateľ už do tejto miestnosti nemá možnosť vložiť svoj ticket. Ďalšou možnosťou uzatvorenia hernej miestnosti je naplnenie kapacity hráčov. V oboch prípadoch uzavretia sa herná miestnosť prestane zobrazovať v ponuke miestností a vygeneruje sa nová miestnosť tohoto typu.

Vyhodnotená herná miestnosť: Po vyhodnotení všetkých ticketov v miestnosti sa tickety zoradia zostupne podla ich výsledku a určí sa víťaz. Týmto sa herná miestnosť prepne do stavu vyhodnotená.

Druhy hier: Užívateľ má možnosť hrať päť druhov hier (Dailybet, Class, 50/50, Head to Head, Private). Každá hra má svoje herné miestnosti s rôznymi kombináciami ich parametrov.

- Dailybet: Miestnosti majú neobmedzený počet ticketov a rôzne hodnoty stávky. Tieto miestnosti môžu byť uzatvorené iba svojou uzávierkou.
- Class: Miestnosti sa líšia maximálnym počtom ticketov a hodnotou stávky.
- 50/50: Podobne ako v hre class sa miestnosti v 50/50 líšia maximálnym počtom ticketov a hodnotou stávky, ale vyhráva vždy polovica užívateľov, a to práve dvojnásobok hodnoty stávky. Tento druh hry prináša menšie riziko, ale zároveň menšiu výhru.
- Head to Head: Ako už napovedá názov, jedná sa o hru jedného proti jednému v rôznych druhoch stávky.
- Private: Hra, ktorú si vytvára užívateľ. Všetky parametre sú voliteľné a užívateľ má možnosť pozvať iných hráčov do miestnosti.

Žetóny: Žetóny sú prostriedok hodnoty stávok na jednotlivé tickety. Každý užívateľ po registrácií obdrží istý počet žetónov. Prostredníctvom týchto žetónov môže podávať stávky na tickety. Hneď po vložení ticketu do niektorej z miestností sa mu odráta počet žetónov rovný hodnote stávky danej miestnosti. Ak užívateľ žetóny minie, raz za týždeň má možnosť si žetóny doplniť.

Vyhodnotenie ticketu: Každý tip ticketu sa vyhodnocuje priebežne na základe odohraných zápasov týchto tipov, pričom do výsledného celkového skóre ticketu sa počítajú len úspešné tipy. Po odohraní všetkých zápasov sa určí výsledné skóre ticketu, ktoré je rovné násobkom hodnôt úspešných tipov.

Pojmy v praxi

Aplikácia DailyBet zobrazuje ponuku reálnych zápasov, hraných v blízkej dobe. Užívateľ má možnosť z týchto zápasov zostaviť vlastný ticket. Po zostavení ticketu si následne zvolý aký typ hry chce hrať a vyberie si hernú miestnosť, kde svoj zostavený ticket vloží, a užívateľovi sa odrátajú žetóny vo výške hodnoty stávky danej miestnosti. Po uzavretí tejto miestnosti užívateľ súťaží s ostatnými hráčmi, ktorý vložili svoj ticket do rovnakej miestnosti. Po vyhodnotení všetkých ticketov v miestnosti sa určia víťazi, ktorým sa ihneď pripočítajú žetóny z výhry.

3.3. Užívateľ

Z dôvodov obmedzených dátových zdrojov sa zameriam na kvalitatívny výskum ako na kvantitatívny. Na základe rozhovoru s rôznymi ľudmi, ktorý sa líšia veko, vzdelaním a záujmami, som zostavil personu cieľového užívateľa produktu.

Persona cieľového užívateľa

meno: Andrej
priezvisko: Horný
vek: 23
vzdelanie: stredoškolské

Relevantné dáta:
- bývalý športovec a stále športovo aktívny
- je pracovne činný a má mesačný príjem peňazí
- navštevuje a aktívne sleduje športové akcie
- sociálne aktívny, pravidelne sa stretáva s priateľmi
- má rád výzvy a súťaže
- mobilný telefón má celý deň po ruke a používa ho vždy keď potrebuje vyplniť volnú chvíľu.

Potreby užívateľa

Nie každý užívateľ má rovnaké skúsenosti a v kontexte s mobilnou aplikáciou existujú tri obecne známe typy užívateľov: začiatočníci, pokročilí a experti.

Z pohľadu môjho návrhu musí byť aj pokročilí či expertný užívateľ pri prvom spustení aplikácie považovaný za začiatočníka, pretože sa s veľkou pravdepodobnosťou s podobnou aplikáciou nestretol. Paradoxom je, že práve užívatelia, ktorý máju skúsenosti so softwarom určeným na športové stávkovanie, a majú teda už vytvorený vlastný mentálny model, môže byť účel aplikácie ťažšie pochopiteľný. Práve preto je potrebné užívateľovi v stručných bodoch vysvetliť, na čo aplikácia slúži. Nieje potrebné vysvetľovať jednotlivé funkcie aplikácie, stačí len, aby si aj neskúsený užívateľ vytvoril základný mentálny model: čo je účelom aplikácie, na základe ktorého by si bol schopný vytvoriť cieľ.

Užívateľ spustí aplikáciu za jedným z nasledujúcich cieľov:

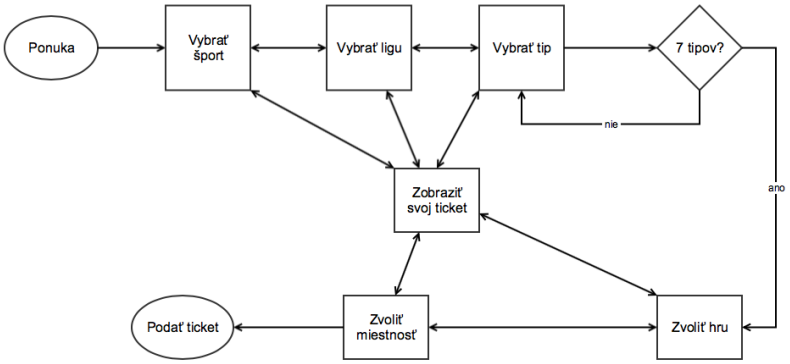
- Chce si prezrieť výsledky svojich ticketov.
- Chce si prezrieť zápasy a kurzy v ponuke.
- Chce zostaviť ticket.

3.4. Návrh užívateľského rozhrania

3.4.1. Informačná štruktúra

Pri návrhu informačnej štruktúry si najskôr rozpíšem jednotlivé kroky dôležitých procesov užívateľa. Využijem na to procesné diagramy, ktoré slúžia k názornému grafickému zobrazeniu postupnosti všetkých krokov určitého procesu.

Najzákladnejším a najkomplexnejším procesom aplikácie bude zostavenie ticketu a následné vloženie do jednej z ponúkaných miestností. Obrázok 3.1 zobrazuje procesný diagram tohoto procesu. Tento diagram mi umožňuje uvedomiť si a zohľadniť jednotlivé väzby medzi krokmi procesu v informačnej štruktúre aplikácie. Počas tohoto procesu musí užívateľ prejsť takmer celou informačnou štruktúrou aplikácie. Ako prvá je ponuka zápasov, kde si vyberie a zvolí tipy výsledkov zápasov. Následne vyberie druh hry a miestnosť, do ktorej ticket vloží. Po vytvorení a vložení ticketov do miestností má možnosť sledovať ich progres v prehľade aktuálnych ticketov. Užívateľský profil len ponúka informácie o samotnom užívateľovi, jeho činnostiach a správe svojich žetónov.



Obrázok 3.1 Procesný diagram vytvárania a vkladania ticketu

3.4.2. Ponuka zápasov

Názov DailyBet, v doslovnom preklade denná stávka, vychádza z predpokladu, že užívateľ má nielen možnosť, ale je aj vedný k tomu, aby aplikáciu denne otváral. Fakt, že ponuka zápasov bude obsahovať len zápasy konajúce sa do tridsiatichšiestich hodín, podnecuje model denného užívania. Dnes si vytvorím ticket a zajtra viem výsledok. Zároveň udržuje obsah aplikácie sústavne menný a nestály, každý deň bude iná ponuka. Zobrazovať zápasy do tridsiatichšiestich hodín má aj iný význam, ktorému sa budem venovať neskôr v časti návrhu. Kvôli prehľadnosti a zaužívanosti je ponuka roztriedená podľa športu a následne ligy. Toto roztriedenie je používané v mnohých existujúcich riešeniach športového tipovania. Ponuka zápasov je úzko spätá s vytváraním ticketu. Pri prezeraní zápasov má užívateľ možnosť voľby svojich tipov. Tento model zlučuje vytváranie a prehľad ponuky zápasov a nabáda užívateľa k vytvoreniu svojho ticketu už pri prezeraní ponuky zápasov. Dôležité je preto zobrazovať progres vytvárania ticketu prostredníctvom informácií o počte tipov, aktuálnom skóre a možnosti ticket zobraziť.

3.4.3. Druh hry a herné miestnosti

Po tom, ako užívateľ zostavý svoj ticket, nasleduje výber druhu hry, ktorú chce hrať, a hernej miestnosti, do ktorej ticket vložiť. Z procesného diagramu, viz obrázok 3.1, je zrejmé, že počas celého procesu musí mať užívateľ možnosť si svoj zostavený ticket prezrieť a upraviť. V závislosti od druhu hry by malo byť roztriedenie jednotlivých miestností rôzne. Konkrétne v hre dailybet sa miestnosti líšia iba hodnotou stávky, kdežto v ostatných hrách, napr. class, sa miestnosti rozlišujú jak hodnotou stávky, tak limitom ticketov. Kvôli prehľadnosti navrhujem užívateľovi zobraziť len miestnosti určitých hodnôt stávky. Užívateľ si zvolí hodnotu stávky a následne vyberie miestnosť s požadovaným limitom ticketov. Už na prehľade miestností by mal vidieť základné informácie, teda limit, súčastný počet ticketov a žetónov v hre. Po vybratí by mal vidieť detailnejšie informácie, napr. zoznam hráčov v miestnosti. Fakt, že užívateľ uvidí ostatných hráčov, podnecuje u hráčov súťaživosť. Počas vkladania ticketov do miestností je táto miestnosť v otvorenom stave a užívateľ nemá možnosť nahliadnuť na tickety ostatných hráčov,

avšak po uzavretí miestnosti spôsobenej naplnením kapacity či dosiahnutím uzávierky, má užívateľ možnosť prezrieť si tickety svojich súperov.

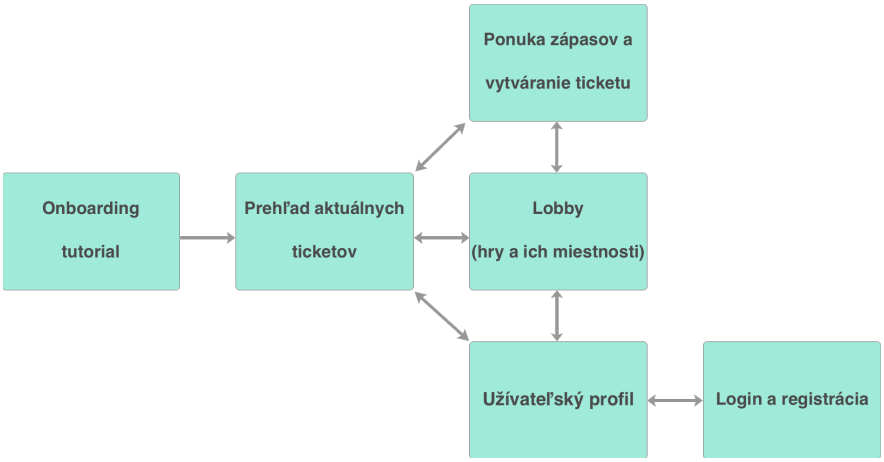
3.4.4. Prehľad aktuálnych ticketov

Zostavením a vložením ticketu do hernej miestnosti ešte jeho životný cyklus nekončí. Zobrazenie prehľadu aktuálnych ticketov a sledovanie ich progresu bude pravdepodobne najčastejším dôvodom otvorenia aplikácie. Užívateľ bude mať možnosť vidieť zoznam aktuálne hrajúcich ticketov s informáciami, ako je počet odohraných zápasov, aktuálne skóre ticketu, pozícia v miestnosti, čas do ukončenia hry, druh hry, hodnota stávky miestnosti, počet hráčov v miestnosti, počet žetónov v hre a najvyššie skóre. Po vybraní jedného z ticketov bude užívateľovi zobrazený detail ticketu spolu s miestnosťou, kde okrem spomenutých informácií uvidí aj ostatných hráčov a ich skóre s možnosťou zobrazenia ich zápasov. Možnosť zobrazenia zápasov iných hráčov bude prístupná iba ak je miestnoť uzavrená. Užívateľ takisto uvidí všetky tipy a výsledky týchto tipov.

3.4.5. Užívateľský profil

Užívateľský profil ponúka užívateľovi celkový prehlad o jeho počinoch v aplikácií. Najdôležitejšou funkcionalitou je správa jeho žetónov. Užívateľ tu má možnosť videť svoj aktuálny stav žetónov a požiadať o ich doplnenie. Ďalšou z informácií dostupných z užívateľského profilu je história všetkých hier ktoré odohral. Užívateľ má možnosť si prezrieť svoje už vyhodnotené tickety. Samozrejmosťou je zobrazenie fotky a základných informácií o užívateľovi (prezývka, e-mail) a možnosť úpravy týchto informácií.

Výsledná informačná štruktúra aplikácie zahrňujúca aj vyššie nespomenuté časti onboarding tutorial a login s registráciou je zobrazená na obrázku 3.2.



Obrázok 3.2 Informačná štruktúra aplikácie

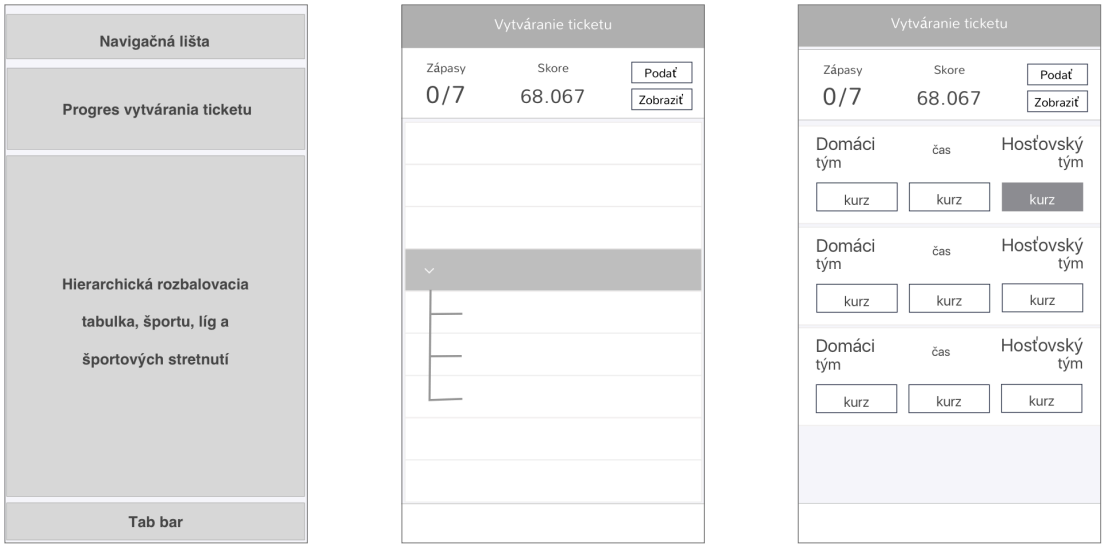
3.5. Návrh drôtených modelov

Pri návrhu drôtených modelov užívateľského rozhrania vychádzam z navrhnutej informačnej štruktúry aplikácie a keďže je návrh cielený na platformu iOS budem sa snažiť dodržiavať smernice užívateľského rozhrania, ktoré vydáva firma Apple.

Z pohľadu informačnej štruktúry dáva zmysel, aby každý z hlavných okruhov, konkrétne ponuka zápasov, výber hry a podanie stávky, prehľad aktuálnych ticketov a užívateľský profil, existovali v aplikácií paralelne. Inými slovami, užívateľ by k nim mal mať kedykoľvek prístup. iOS má pre tento návrhový vzor zaužívanú komponentu s názvom *tab bar*. V nasledujúcej časti rozoberiem informačnú štruktúru po vizuálnej stránke. Každý jeden zo štyroch už spomínaných okruh reprezentuje jednu obrazovku v TabBare.

3.5.1. Vizualizácia ponuky zápasov

Ako som už spomenul ponuka zápasov je spojená s vytváraním ticketu. Jednotlivé zápasy budú roztriedené podľa športu a ligy v ktorej sa zápas nachádza. Obrazovka by okrem zoznamov športov, líg a zápas mala obsahovať aj znázornenie progresu vytvárania ticketu spolu s tlačidlami, ktoré umožnia užívateľovi náhľad na vytváraný ticketu a pokračovať z vytvárania na výber typu hry. Znázornenie progresu a tlačidlá bude viditeľné z každej vrstvy ponuky. Ako prvý sa užívateľovi zobrazí zoznam športov. Po zvolení jedného zo športu sa zoznam zmení na ligy v tomto športe. Pre väčšiu prehľadnosť, ligy z jednej krajiny budú združené v kategórií tohoto štátu formou padajúceho zoznamu teda až po kliku na daný štát sa zobrazia jeho ligy. Po vybratí požadovanej ligy sa zoznam zmení na jednotlivé zápasy. Každý zápas bude mať pri



Obrázok 3.3 Drôtené modely ponuky zápasov

názvoch jednotlivých tímov označiteľné tlačidlá s hodnotou kurzu. Po zvolení jedného z kurzov sa automaticky aktualizuje progres vytvárania ticketu. Na obrázku 3.3 sú zobrazené drôtené modely znázorňujúce rozloženie jednotlivých prvkov.

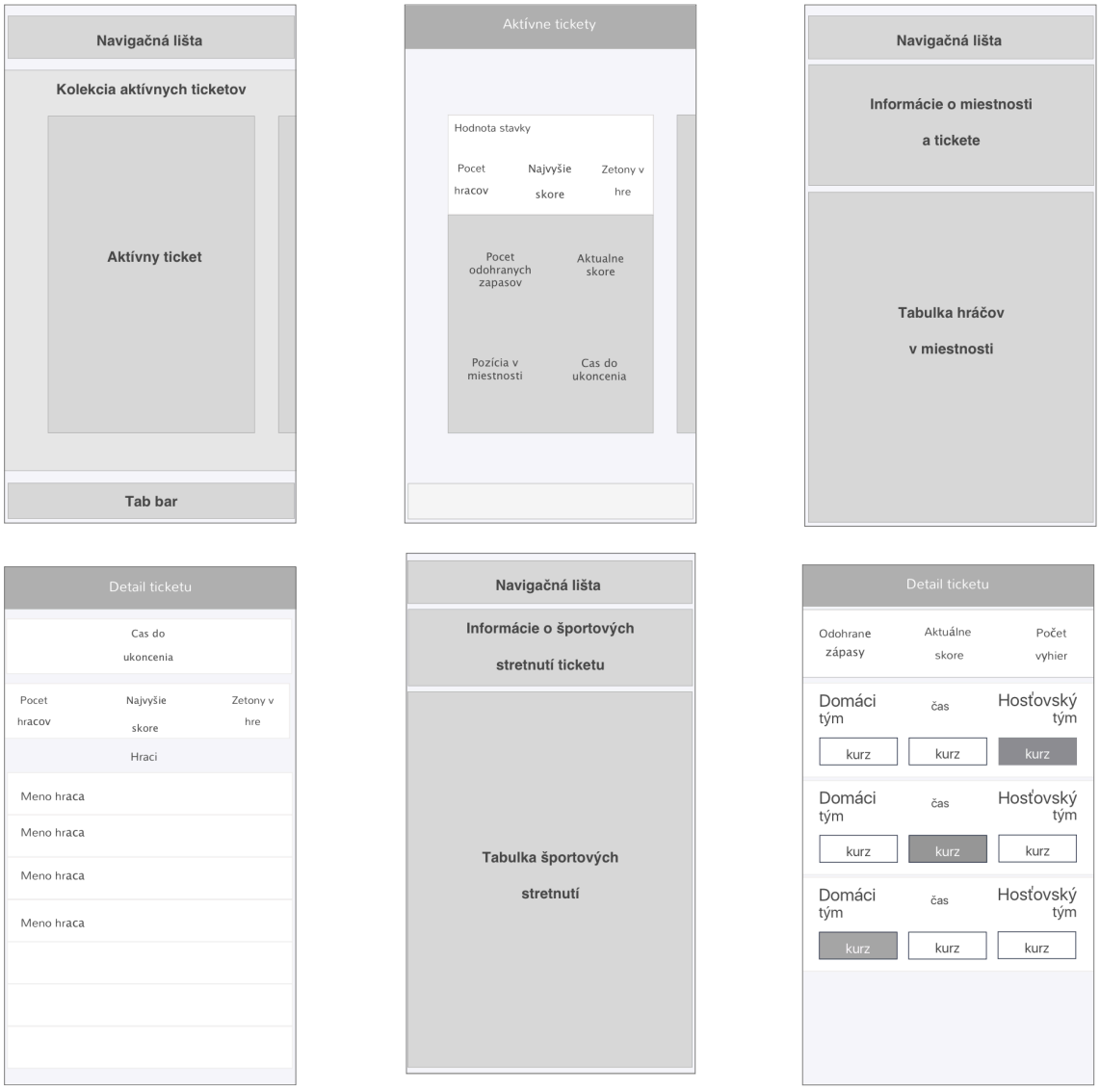
3.5.2. Vizualizácia druhu hry a herných miestnosti

Užívateľ dokončil vytváranie ticketu a chce ho vložiť do hracej miestnosti. Predtým si však musí zvoliť zo zoznamu druh hry. Po zvolení hry sa mu zobrazí zoznam miestností spolu so skórom ticketu a možnosťu jeho zobrazenia. Zobrazenie herných miestností sleduje klasický návrhový vzor zoznamu a detailu. Po kliku na jednu z miestností sa zobrazí jej detail. Detail miestnosti zobrazuje informácie o tejto miestnosti a užívateľ má možnosť vložiť svoj vytvorený ticket. Po úspešnom vložení ticketu sa zobrazí potvrdenie o podaní ticketu obsahujúce súhrn informácí o aktuálnom stave a odkazom na prehľad užívateľovho prehľadu prebiehajúcich hier. Na obrázku 3.4 sú zobrazené drôtené modely obrazoviek druhu hier a herných miestností.

Obrázok 3.4 Drôtené modely druhu hier a herných miestností

3.5.3. Vizualizácia prehľadu aktuálnych ticketov

Užívateľ chce sledovať progres svojich hraných ticketov. V návrhu tejto obrazovky sa vychádza z predpokladu, že užívateľ bude mať priemerne podaný jeden až tri tickety. Z tohoto dôvodu je obrazovka navrhnutá tak, aby zobrazovala práve jeden ticket s možnosťou swipu pre zobrazenie ďalšieho. Ná drôtenom modely viz obrázok 3.5 sa ticketu svojim výzorom snaží reflektovať ticket v reálnom svete. Obsahuje vizuálne rozdelený súhrn informácií o hernej miestnosti a samotnom tickete. Dôležitou súčasťou obrazovky prehľadu aktuálnych ticketov je vizualizácia stavu keď užívateľ nemá žiaden aktuálne hrajúci ticket. Je nezbytné, aby bol užívateľ o tomto stave informovaný a odkazovaný na vytvorenie svojho ticketu. Obrazovka preto obsahuje tlačidlo plus ktorý užívateľa navedie na ponuku zápasov, ktorá je zlúčenú s vytváraní ticketu. Po kliku na jeden z ticketov sa užívateľ dostane na detail progresu celej hry. Detail ticketu je

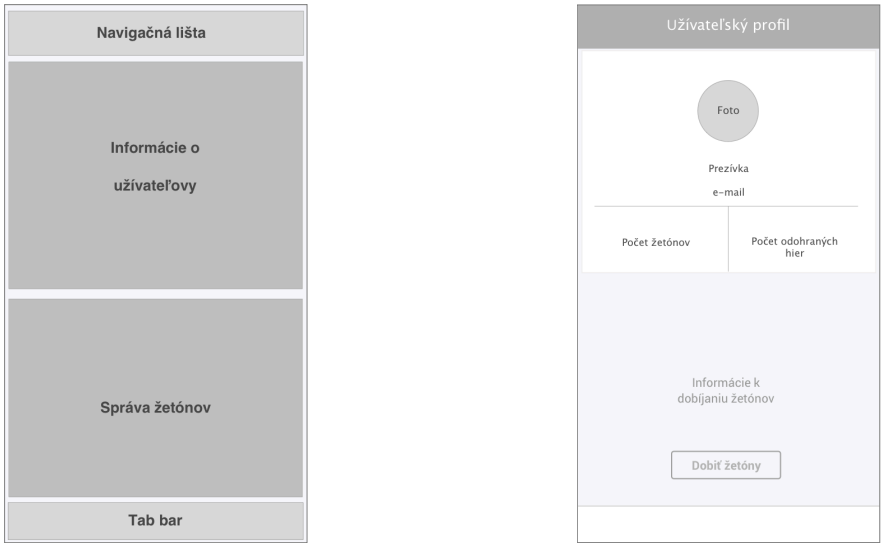


Obrázok 3.5 Vizualizácia prehľadu aktuálnych ticketov

rozdelený do dvoch obrazoviek. Prvá obrazovka s názvom hry a hodnotou stávky zobrazuje informácie týkajúce sa hernej miestnosti. Zobrazuje zoznam užívateľov v tejto miestnosti spolu s ich ticketmi. Užívateľ tak má možnosť sledovať nielen svoj progres, ale aj progres svojich súperov. Druhá obrazovka obsahuje informácie o tipoch na tomto tickete. Zápas každého tipu má farebne zvýraznené kurzy na základe stavu tohoto tipu. Môžu nastať celkom tri stavy a to, že zápas ešte neprebehol, zápas prebehol a užívateľov tip je spravý, alebo že zápas prebehol a užívateľov tip je nesprávny. Každý jeden stav bude znázornený prostredníctvom farebného označenia kurzov pri zápase. Prechod medzi jednotlivými obrazovkami bude možný prostredníctvom swipu z jednej strany obrazovky na druhú.

3.5.4. Vizualizácia užívateľského profilu

Užívateľský profil ponúka prehľad osobných informácií spolu s históriou jeho odohraných ticketov a správu žetónov. Táto obrazovka má prevažne informačný charakter a neumožňuje užívateľovi veľa interakcií. Užívateľský profil sa skladá z dvoch obrazoviek zobrazujúce osobné dáta a históriu ticketov. Užívateľ má možnosť vidieť aktuálny počeť svojich žetónov a odohraných hier. Správa žetónov ponúka informácie o tom kedy je možné si svoje žetóny doplniť a tlačidlo na samotné doplnenie. Návrh drôtených modelov je zobrazený na obrázku 3.6.

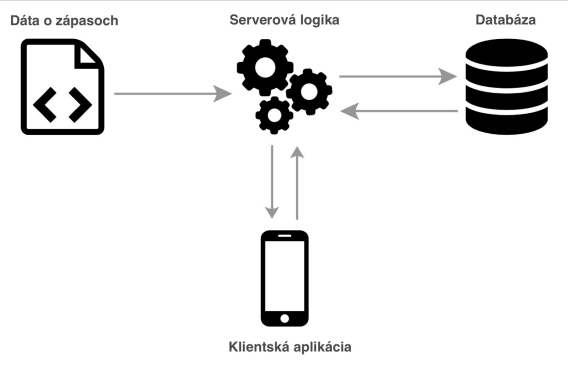


Obrázok 3.6 Drôtené modely užívateľského profilu

3.6. Návrh serverovej časti a jej komunikácie s aplikáciu

3.6.1. Serverová aplikácia

Serverová aplikácia má niekoľko úloh. Mala by sprostredkovať dáta medzi databázou a klientskou aplikáciou, či sa starať o správu ponuky zápasov a herných miestností. Je dôležité, aby sa server staral o čo najväčší počet funkcionality. Klientská aplikácia by mala slúžiť len na prijímanie dát od užívateľa. Dôvodom veľkej zodpovednosti serveru je v prvom rade flexibilita. Pri budúcej úprave nieje potrebné, aby si užívateľ aktualizoval svoju aplikáciu. Ďalším z dôvodov je podpora rôznych platforiem. Rozšírenie aplikácie napríklad na webovú platformu bude jednoduchšie a jednotnejšie. Obrázok 3.5 zobrazuje návrh architektúry systému.



Obrázok: 3.5 Architektúra systému

3.6.2. Ponuka zápasov

Server je zodpovedný za aktualizáciu ponuky zápasov, ktorú bude získavať prostredníctvom XML feedu. Aktualizácia zápasov spolu s ich kurzami prebieha dvakrát denne, pretože niektoré športové ligy vypisujú kurzy na večerné zápasy až behom dňa. Server zparsuje súbor vo formáte XML a získanými informáciami naplní databázu zápasov. Ligy a športy budú v aplikácií nemenné, takže je potrebná aktualizácia len hraných zápasov. Parsovanie týchto zápasov neobmedzuje spomínaný tridsaťšesť hodinový limit ponuky zápasov. Zobrazenie správnej ponuky zápasov má na starosti klientská aplikácia.

3.6.3. Vytvorenie ticketu

Klientská aplikácia pri vkladní ticketu do jednej z herných miestností posiela požiadavok na server o vytvorenie tohoto ticketu s potrebnými informáciami, ktorými sú identifikačné čísla

zápasov a príslušné tipy užívateľa, identifikačné číslo miestnosti a samotného užívateľa. Tento návrh umožňuje pristupovať k funkcionalite serveru z akejkoľvek platformy, pretože posiela primitívne dátové tipy. Server pred vytvorením ticketu musí dáta zvalidovať. Validácie prebieha v niekoľkých krokoch:

- Kontrola štartu zápasov. Užívateľ nemôže tipovať na zápasy, ktoré prebiehajú alebo už prebehli.
- Kontrola, či užívateľ už v danej miestnosti nemá vytvorený ticket.
- Kontrola, či užívateľ má dostatok žetónov na vytvorenie ticketu.
- Kontrola, či herná miestnosť má volné miestno pre nový ticket, inými slovami či sa nenaplnil maximálny počet ticketov danej hernej miestnosti.

Je dôležité, aby všetky úpravy vykonané na entite užívateľa či hernej miestnosti boli vykonané atomicky, pretože sa jedná o kritickú sekciu, kedy k jednej miestnosti môže pristupovať naraz niekoľko užívateľov. V priebehu vytvárania sa môže stať, že užívateľ naplní limit hráčov hernej miestnosti. V takomto prípade server vytvorí novú hernú miestnosť a naplnenú uzavrie.

3.6.4. Správa herných miestností

Ako som už spomínal, každá herná miestnosť má svoj čas uzávierky. Tento návrh má hneď niekoľko dôvodov. Pre miestnosti hry dailybet, kde je limit hráčov neobmedzený, je to nevyhnutný prvok, pretože miestnosť sa uzavrie iba prostredníctvom jej uzávierky. Pre ostatné miestnosti je možnosť uzavretia aj po naplnení limitu, avšak môže nastať problém, že sa limit hráčov nenaplní. Vzhľadom na myšlienku a princíp denného používania aplikácie je preto výhodné mát uzávierku na všetkých typoch miestností. Každá herná miestnosť je dvadsaťštyri hodín v stave, kedy je možné vkladať tickety, počínajúc časom 00:00 a končiac 23:59 UTC. Server musí každý deň v čase 00:00 UTC vykonať údržbu všetkých miestností. Údržbou sa myslí uzatvoriť stávajúce miestnosti a vytvoriť nové.

3.6.5. Komunikácia servera s klientskou aplikáciou

Pri dátach herných miestností, ktoré sú v jeden čas prístupné a frekventovane meniace všetkými užívateľmi je dôležité, aby užívateľ videl najaktuálnejšie dáta. Pri mobilných aplikáciach je bežnou praxou, že užívateľ si sám požiada o aktualizáciu zobrazených dát. Jedným z vizuálnych návrhových vzorov je tzv. pull to refresh a používa ho väčšina mobilných aplikácií pre aktualizáciu dát v zozname. Tento vzor sa hodí pre štruktúru ako je ponuka zápasov, kedy sa dáta aktualizujú dvakrát denne. Pri herných miestnostiach, kedy sa dáta môžu meniť niekoľko krát v priebehu jednej minúty, by bolo vhodné, keby server tieto dáta posielal aplikácií pri ich aktualizácií. Klientska aplikácia tak musí najskôr požiadať o odoberanie dát zodpovedajúcim istým kritériám a server by potom vždy pri pridaní, odobratí či aktualizácií posielal tieto dáta klientskej aplikácií. Tento návrh sa okrem herných miestností hodí aj na prehľad aktuálne hrajúcich ticketov a zabezpečuje, že užívateľ vždy vidí najaktuálnejšie dáta bez nutnosti akejkoľvek iniciatívy.

Kapitola 4

4. Realizácia a testovanie

V tejto kapitole popíšem, ako som postupoval pri implementácií návrhu. Z návrhu je zrejmé, že je potrebné implementovať klientsku mobilnú aplikáciu pre iOS, serverovú aplikáciu a grafický návrh drôtených modelov.

4.1. Grafický návrh

Grafický návrh vychádza z drôtených modelov, ktoré sú zobrazené v kapitole 2. Všetky obrazovky boli zostrojené pomocou programu Sketch. Pred začiatkom kreslenia som si zvolil základný farebný akord pozostávajúci z troch farieb. Na každej obrazovke som sa snažil dodržiavať jednotnosť vizuálneho štýlu. Mojou snahou bolo zostrojiť elegantné, čisté a zároveň hravé rozhranie, ktoré navyše ponúka priestor pre grafické animácie. Výsledný grafický návrh je zobrazený v prílohe **B**.

4.2. Klientska aplikácia

Aplikácia je realizovaná na mobilnej platforme iOS. Pri implementácii som používal integrované vývojové prostredie Xcode a programovací jazyk Swift.

Použité knižnice

Pri implementácii boli prevažne dostačujúce knižnice, ktoré ponúka iOS SDK, avšak nezaobišiel som sa bez knižníc tretích strán. Pre importovanie knižníc tretích strán do projektu využívam správcu závislostí s názvom Cocoapod. Jednou z použitých knižníc je PromiseKit. PromiseKit ponúka elegantné riešenie asynchrónnych funkcií spätného volania a používam ju prevažne na obalenie komunikácie so serverom. Ďalšou z použitých knižníc je FuntastyKit, ktorá obsahuje rozšírenia pre elegantné riešenie návrhových vzorov a triedy pre správne udržovanie MVVM-C architektúry. FuntastyKit nieje celkom knižnica z tretej strany, pretože som sa podieľal na jej vytváraní. Najdôležitejšou závislosťou tretej strany je Parse server. Parse server je BaaS riešenie, ktoré ponúka SDK pre iOS, Android, Jvascript a mnohé iné jazyky a platformy. Pomocou Parse je realizovaný celý backend a knižnica Parse iOS SDK sprostredkuje komunikáciu klientskej aplikácie so serverom. Navyše toto SDK ponúka riešenia caschovania dát či prihlásenie a registráciu.

MVVM-C v praxi

Kód aplikácie dodržiava softwarovú architektúru MVVM-C. V kapitole 2 som popísal výhody MVVM-C architektúry a v nasledujúcej časti popíšem, ako vypadá implementácia tejto softwarovej architektúry.

Každá jedna obrazovka v mobilnej aplikácií sa označuje ako scéna, ktorá sa skladá z minimálne štyroch zložiek a to coordinator, controller, view model a view.

Coordinator je vstupným a výstupným bodom celej scény. Samotná aplikácia má svôj vlastný coordinator, ktorý rozhoduje o tom, akú scénu zobrazí pri spustení aplikácie. Coordinator je zodpovedný za vytváranie inštancií všetkých ostatných zložiek scény. V praxi je coordinator protokol, ktorý je zobrazený na obrázku 4.1. Trieda, ktorá napĺňa tento protokol vo svojej inicializácií inicializuje aj všetky ostatné zložky scény. Po tom, čo coordinator nainicializuje celú scénu, užívateľ už má pred sebou zobrazenú obrazovku a aplikácia čaká na jeho vstupy. View controller slúži na odchytávanie týchto vstupov, ktoré sú delegované zo samotnej obrazovky. View controller je jediná trieda, ktorá má prístup k obrazovke scény a taktiež si drží odkaz na coordinator pri potrebe presunu do inej scény. View controller teda zastáva funkciu usmerňovaču toku dát. Najskôr odchytí vstupy z obrazovky a rozhoduje, ako s nimi naloží. View controller si okrem coordinatora drží odkaz na view model, v ktorom sa nachádza výpočtová logika scény. View model komunikuje s view controllerom pomocou návrhového vzoru delegáta. View model si drží odkaz na delegáta, ktorý napĺňa komunikačný protokol. Príklad tohoto protokolu je zobrazený na obrázku 4.2. Trieda view modelu si drží všetky potrebné dáta a jediný prevádza operácie nad týmito dátami. View model si drží odkaz aj na inštancie tried servisov, ktoré slúžia ako vyššia abstrakcia pre operácie ako perzistencia či

```
public protocol Coordinator {
    /// Triggers navigation to the corresponding controller
    func start()

    /// Stops corresponding controller and returns back to previous one
    func stop()

    /// Called when segue navigation form corresponding controller to different
        controller is about to start and should handle this navigation
    func navigate(from source: UIViewController, to destination:
        UIViewController, with identifier: String?, and sender: AnyObject?)
}

/// Navigate and stop methods are optional
public extension Coordinator {
    func navigate(from source: UIViewController, to destination:
        UIViewController, with identifier: String?, and sender: AnyObject?) {
    }

    func stop() {
    }
}

public protocol DefaultCoordinator: Coordinator {
    associatedtype ViewController: UIViewController
    weak var viewController: ViewController? { get set }

    var animated: Bool { get }
    weak var delegate: CoordinatorDelegate? { get }
}
```

Obrázok: 4.1 Protokol coordinatora

získavanie dát. Vo väčšine prípadov sú operácie servisov asynchrónne, takže komunikáciu servisov a view modelu sprostredkovávajú funkcie spätného volania. Spomínaná knižnica PromiseKit ponúka elegantné používanie týchto funkcii.

Spojením všetkých zložiek scény vzniká tok dát, ktorý postupne prechádza všetkými vrstvami, kde každá jedna vrstva má jasne definované úlohy. Zlúčením tohoto modelu s technikou vkladania závislostí (anglicky dependency injection), ponúka možnosť mockovať dáta jednotlivých vrstiev a testovať tak každú vrstvu samostatne.

```
protocol TicketCreationViewModelDelegate: class {
    func reloadData()
    func changedTableState(from state: TableStates)
    func addRows(at indexPaths: [IndexPath])
    func deleteRws(at indexPaths: [IndexPath])
    func updateTipsAndScore()
    func show(error: Error)
}
```

Obrázok: 4.2 Komunikačný protokol

Implementácia komunikácie so serverom

Knižnica Parse SDK pre iOS ponúka možnosť vytvárať objekty zdedené z triedy PFObject, ktorá je základná trieda reprezentujúca perzistentné dáta v parse cloud servery. Inými slovami, ak by vlastné triedy modelov aplikácie dedili z triedy PFObject, mám možnosť priamo na inštanciách týchto tried využívať funkcionalitu ukladania či sťahovania dát tejto inštancie zo servera. Ďalšou výhodou dedenia tried modelov z PFObjectu je jednoduché sťahovanie súboru dát prostredníctvom dotazov. Knižnica Parse SDK disponuje triedou PFQuery, ktorá definuje dotaz nad triedou PFObject. Spojením zdedených modelov aplikácie z triedy PFObject a funkcionality, ktorú ponúka trieda PFQuery vzniká elegantné a jednoduché získavanie dát zo servera, bez nutnosti parsovania a prevádzania získaných dát na modely v aplikácií. Na obrázku 4.3 je zobrazený príklad získavania herných miestností, ktoré reprezentuje trieda Room definovaná na obrázku 4.4.

```
func getRooms(for game: GameType, enterFeed: Int?) -> Promise<[Room]> {
    return Promise { fulfill, reject in
        guard let query = Room.query() else {
            return
        }
        query.whereKey("gameType", equalTo: game.rawValue)
        query.findObjectsInBackground { objects, error in
            if let rooms = objects as? [Room] {
                fulfill(rooms)
            } else if let error = error {
                reject(error)
            }
        }
    }
}
```

Obrázok: 4.3 Funkcia získavania herných miestností

```
class Room: PFObject, PFSubclassing {
    @NSManaged var gameType: Int
    @NSManaged var highestScore: Double
    @NSManaged var entryFee: Int
    @NSManaged var ticketLimit: Int
    @NSManaged var numberOfTickets: Int
    @NSManaged var stake: Int
    @NSManaged var state: Int
    @NSManaged var closingDate: Date

    static func parseClassName() -> String {
        return self.className
    }
}
```

Obrázok: 4.4 Definícia triedy popisujúca hernú miestnosť

Návrh komunikácie klientskej aplikácie so serverom pojednáva o konceptu, kedy server sám posiela dáta klientskej aplikácií pri zmene týchto dát. Knižnica Parse túto funkcionalitu ponúka. ParseLiveQuery umožňuje registrovať dotaz na server spolu s funkciu spätného volania, ktorá sa zavolá pri zmene dát spadajúcich pod daný dotaz. Funkcia spätného volania sa zavolá s dvoma parametrami, samotným dotazom a objektom, ktorý sa zmenil spolu s popisom o tom, aká zmena nastala. Konkrétny objekt mohol byť upravený, vymazaný, vytvorený alebo opustil rámec dotazu. Najväčším problémom tejto implementácie je, že pokiaľ prijatý objekt obsahuje odkaz na iný objekt, je potrebné odkazovaný objekt dodatočne stiahnuť. V situácii, kedy bude mať mnoho užívateľov registrovaný dotaz, pod ktorý spadajú rovnaké objekty, môže nastať priveľká záťaž na server pri zmene tohoto objektu, kedy server najskôr tento objekt každému odošle a vzápätí obdrží požiadavok na stiahnutie odkazovaných objektov.

Toto riešenie je potrebné zakomponovať do architektúry MVVM-C a využil som ho pri prehľade aktívnych ticketov. View model scény aktívnych ticketov si drží odkaz na servisu, ktorá zabezpečuje sťahovanie dát, ktoré sú relatívne k ticketom. Po načítaní celej scény view model žiada od servisy, aby stiahla všetky aktuálne hrajúce tickety, a ako parameter jej zasiela odkaz na funkciu view modelu, ktorá má za svoj parameter asynchrónnu funkciu spätného volania zabalenú do triedy Promise, ktorú poskytuje knižnica PromiseKit. Funkcie view modelu sú zobrazené na obrázku 4.5. Funkcia servisy má potom za úlohu vytvoriť a zaregistrovať dotaz na aktuálne hrajúce tickety. Následne tento dotaz vykoná a vráti výsledky. Počas registrácie dotazu sa musí definovať funkcia spätného volania, ktorá sa zavolá vždy pri operácii s dátami spadajúcimi pod dotaz. Táto funkcia zistí, o aký druh operácie ide, a vytvorí asynchrónnu funkciu, ktorú predá ako parameter funkcií, ktorá prišla servise ako parameter pri volaní z view modelu. Celá funkcia je zobrazená na obrázku 4.6. Zvolené riešenie mi umožňuje elegantne získať všetky potrebné informácie priamo do view modelu, kde rozhodujem o tom, akým spôsobom aktualizujem obrazovku. Zároveň neporušujem pravidlá MVVM-C architektúry a každá operácie prislúcha správnej vrstve.

```
func load() {
    ticketService.getActualTickets(with: handle).then { tickets -> Void in
        self.actualTckets = tickets
        self.delegate?.didUpdateTickets()
    }
}

func handle(promise: Promise<TicketEvent>) {
    _ = promise.then { ticketEvent -> Void in
        switch ticketEvent {
        case let .update(ticket):
            self.actualTckets.enumerated().forEach { index, actualTicket in
                if actualTicket.objectId == ticket.objectId {
                    ticket.tips = self.actualTckets[index].tips
                    self.actualTckets[index] = ticket
                }
            }

        case let .add(ticket):
            self.actualTckets.append(ticket)

        case let .delete(ticket):
            self.actualTckets.enumerated().forEach { index, actualTicket in
                if actualTicket.objectId == ticket.objectId {
                    self.actualTckets.remove(at: index)
                }
            }
        }
        self.delegate?.didUpdateTickets()
    }
}
```

Obrázok: 4.5 Funkcie view modelu prehľadu aktuálnych ticketov

```
func getActualTickets(with subscribeHandler: @escaping TicketsPromiseHandler) -> Promise<[Ticket]> {
    return Promise { fullfill, reject -> Void in
        guard let user = PFUser.current() else {
            return
        }
        ticketQuery.whereKey("belongsToUser", equalTo: user)
        ticketQuery.whereKey("isActive", equalTo: true)
        ticketQuery.includeKey("room")
        ticketQuery.includeKey("tips.match")
        actualTicketSubscription = liveQueryClient.subscribe(ticketQuery).handleEvent { query, event in
            var promise: Promise<TicketEvent>
            switch event {
            case .updated(let ticket):
                promise = Promise<TicketEvent> { fullfill, reject in
                    ticket.room.fetchInBackground { room, error in
                        if let room = room as? Room {
                            ticket.room = room
                            fullfill(.update(ticket))
                        } else if let error = error {
                            reject(error)
                        }
                    }
                }
            case .left(let ticket), .deleted(let ticket):
                promise = Promise { fulfill, _ in fulfill(TicketEvent.delete(ticket)) }

            case .created(let ticket), .entered(let ticket):
                promise = Promise<TicketEvent> { fullfill, reject in
                    ticket.room.fetchInBackground { room, error in
                        if let room = room as? Room {
                            ticket.room = room
                            fullfill(.add(ticket))
                        } else if let error = error {
                            reject(error)
                        }
                    }
                }
            }
            DispatchQueue.main.async {
                subscribeHandler(promise)
            }
        }
        _ = actualTicketSubscription.handleError { (query, err) in
            reject(err)
        }
        ticketQuery.findObjectsInBackground { objects, error in
            if let tickets = objects {
                fullfill(tickets)
            } else if let error = error {
                reject(error)
            }
        }
    }
}
```

Obrázok: 4.6 Funkcia zabezpečujúca sťahovanie a registráciu dotazov na tickety

4.3. Serverová aplikácia

Pre chod serverovej aplikácie som využil PaaS službu od Heroku, na ktorom som spustil BaaS riešenie Parse server. Pre lepšiu prácu s databázou parse servera som na Heroku spustil aplikáciu Parse dashboard, ktorá mi umožňuje prístup a úpravu databázy z webového prehliadača, čo výrazne zjednodušilo vývoj. Databáza je postavená na MongoDB, avšak Parse server vytvára vrstvu, ktorá umožňuje narábať s jej dátami podobne, akoby boli uložené v relačnej databáze.

Implementácia serverovej časti je realizovaná prostredníctvom parse cloud funkcii, ktoré ponúka Parse server. Funkcie sú písané v jazyku JavasScript. Počas implementácie som narazil na niekoľko problém, z ktorých najzávažnejší ból, že parse server podporuje atomické operácie iba nad číslicovými dátovými typmi. Tento problém vznikol pri snahe zobrazovať užívateľovi jeho pozíciu a najvyššie skóre v miestnosti, kým je miestnoť otvorená a užívatelia do nej môžu vkladať svoje tickety. Najvýhodnejším riešením je vyhodnotiť pozíciu užívateľa a najvyššie skóre až po uzavretí miestnosti.

4.4. Testovanie a vyhodnotenie

V tejto časti práce sa zaoberá testovaním aplikácie a vyhodnotím výsledkov. Popisujem podobu, priebeh a vyhodnotenie výsledkov z testovania.

Testovanie počas vývoja

Aplikácia bola počas vývoja priebežne testovaná rôznymi užívateľmi. Cieľom testovania bolo zistiť zrozumiteľnosť navrhnutého užívateľského rozhrania. Počas testu prebiehalo hlasné premýšľanie a následné dotazy a pripomienky k užívateľskému rozhraniu.

Pilotný test

Aplikácia obsahovala ponuku zápasov, druhy hier a herných miestností. Užívateľ mal za úlohu vytvoriť ticket a vložiť ho do jednej z herných miestností. Užívateľ bol zmetený z návrhu, ktorý kombinuje zlúčenie ponuky zápasov a vytváraním ticketu, pretože aplikácia po štarte zobrazila ponuku zápasov a užívateľ nevedel, že je zároveň aj v procese vytváraní ticketu. Testovanie prinieslo užitočné pripomienky a ukázalo sa, že prehľad aktuálnych ticketov je klúčovým prvkom v navigćií aplikácie, ktorý užívateľa nasmeruje na vytváranie a dáva informáciu o tom, že užívateľ nemá žiadne vytvorené tickety.

Testovanie s prehľadom aktuálnych ticketov

Pridanie prehľadu ticketov, ktorý sa užívateľovi zobrazil po štarte, eliminovalo problém s navigáciou a užívateľ bez väčších problémov úspešne vytvoril ticket, vložil ho do miestností a

vrátil sa späť na prehlad. Menšie problémy spojené s grafickou neprehľadnosťou niektorých tlačidiel som na základe jeho pripomienok vyriešil pozmenením grafického návrhu.

Záverečné testovanie použiteľnosti

Cieľom testovanie je zistiť použiteľnosť aplikácie pre užívateľa. Záverečné testovanie prebehlo na štyroch respondentoch s rôznymi skúsenosťami v oblasti mobilných aplikácii a stávkovania. V úvode som respondentom stručne zhrnul, čo aplikácia rieši, teda že umožňuje jej užívateľom súťažiť v športovom tipovaní s virtuálnymi žetónmi. Respondentom, ktorý nemajú skúsenosti v oblasti stávkovania, som stručne vysvetlil, ako stávkovanie funguje.

Pri užívateľoch, ktorí nemajú skúsenosti so stávkovaním, sa zameriam na to, či zvládnu vytvoriť ticket a či je rozhranie dostatočne názorné na to, aby pochopili princíp stávkovania. Pri skúsených užívateľoch, ktorý vedia ako stávkovanie funguje, sa zameriam na to, či im niečo prekáža, chýba, alebo čakajú od aplikácie iné chovanie.

Respondent č. 1

Prvý respondent má nízke skúsenosti jak v oblasti stávkovania tak i mobilných aplikácii. Navyše je to užívateľ operačného systému android.

Úvodný tutorial užívateľ zbežne prebehol, čakal len na dokončenie animácií. Pri zobrazení prehľadu najskôr niekoľkokrát stláčal informačný dialóg o tom, že nemá vytvorený ticket namiesto tlačidla plus. V ponuke zápasov sa rýchlo zorientoval a náhodne natipoval tri zápasy. Doposiaľ si užívateľ vzhľadom na jeho skúsenosti viedol dobre, ale zasekol sa v polovici vytvárania ticketu. Nesprávne pochopil, koľko zápasov musí obsahovať ticket a opakovane sa snažil prejsť do ďalšieho kroku. Bolo nutné respondentovi vysvetliť, koľko zápasov musí ticket obsahovať, aby sa posunul na ďalší krok. Pri výbere druhu hry nebolo užívateľovi jasné, aký je princíp jednotlivých hier. Pri výbere hernej miestnosti zmiatlo respondenta, že pri hodnote stávky chýba znak žetónov, aby bolo jasné, čo hodnota reprezentuje. Vloženie a prehlad ticketu prebehlo bez problémov.

Respondent č. 2

Druhý respondent má stredné skúsenosti v oblasti stávkovania a mobilných aplikácii. Jedná sa o osobu s vyšším vekom než je cieľový užívateľ.

Počas celého testovania bol najväčším problémom jazyk aplikácie. Respondent neovláda dostatočne anglický jazyk, preto bude aplikáciu potrebné lokalizovať. Vytvorenie ticketu prebehlo bez problémov. Užívateľ ihneď pochopil ako ticket vytvoriť, kde sledovať progres a upraviť svoj ticket. Menší problém nastal pri vkladaní ticketu do hernej miestnosti, kde respondentovi robilo problém pochopiť zobrazené údaje hlavne z dôvodu jazykovej bariéry. Prehľad aktívnych ticketov prebehol bez problémov. Bolo pozorovateľné, že sa užívateľ po podaní prvého ticketu kompletne zorientoval v aplikácií a nerobilo mu problém vyhľadať požadované dáta.

Respondent č. 3

Tretí respondent zodpovedá všetkým kritériám cieľového užívateľa. Má pokročilé skúsenosti v stávkovaní a je dlhodobým užívateľom mobilných aplikácii.

Priebeh testovania prebiehal s najmešími problémami zo všetkých testovaní. Počas úloh nebolo potrebné užívateľovi takmer nič vysvetľovať. Jediná nejasnosť vznikla pri zobrazení detailu aktuálneho ticketu. Konkrétne mu nebolo jasné, ktoré z jeho zápasov už prebehli a ktoré stále čakajú na vyhodnotenie.

Vyhodnotenie testovania

Testovanie prinieslo niekoľko problémov s užívateľským rozhraním aplikácie, ktoré následne popíšem. Jedná sa o problémy, ktoré sa vyskytovali najčastejšie a vychádzajú z reakcií a odpovedí na dotazy k aplikácii.

Progres vytvárania ticketu

Prvým problém je zobrazenie progresu vytvárania ticketu. Je potrebné dať užívateľovi jasne najavo, koľko tipov musí zadať, aby mohol postupovať ďalej. Jedným z riešení je zobrazovať alert dialóg, keď užívateľ klikne na tlačítko set nachádzajúce sa na obrazovke ponuky zápasov. Toto riešenie je najjednoduchšie, pretože rovnaký dialóg už zobrazujem pri snahe užívateľa vložiť nedokončený ticket do hernej miestnosti. Ďalšou možnosťou je prerobiť súčastné zobrazenie a pridať progres bar s počtom potrebných tipov, ktoré by sa postupňe vyplňovali.

Vysvetlenie princípu druhov hier

Jedná sa o problém, ktorý nastal pri všetkých respondentoch. Pri výbere druhu hry nieje jasný princíp hier. Tento problém je riešiteľný pridaním talčidla pre vysvetlivku, alebo zobrazením vysvetlivky priamo pri názve hry.

Chýbajúce informácie

Na základe hlasného rozmýšľania užívateľov počas testovania a následných otázok som zistil problém chýbajúcich informácií, ktoré užívateľ očakáva.

- Pri vložení ticketu aplikácia neposkytuje informáciu o tom, koľko žetónov bolo užívateľovi odrátaných z jeho kreditu.
- Pri vkladní a prehľade aktívnych ticketov chýba informácia o výhernej štruktúre, teda o tom, aké umiestnenie vyhráva a koľko žetónov z banku.
- Skúsený užívatelia si stávkovanie asociujú s reálnymi peniazmi, preto je potrebné užívateľovi vysvetliť princíp stávkovania so žetónmi, ktorý je bez poplatkov.

Možnosť zrušenia práve vložené ticketu

V súčasnej verzí aplikácie ticket po vložení do hernej miestnosti zostáva v procese vytvárania. Užívateľ by mal mať po vložení rozpracovaného ticketu možnosť rozhodnúť sa, či chce tento ticket vložiť do ďalšej miestnosti, alebo chce, aby sa tento rozpracovaný ticket vynuloval, teda zmazal všetky natipované športové stretnutia.

Možnosti ďalšieho vývoja

Súčasný stav aplikácie zodpovedá prototypu, ktorý možno prezentovať za účelom investície, pretože na chod produkčnej verzie aplikácie je potrebné platiť mesačné poplatky minimálne za prevádzku serveru a odober dát športových stretnutí spolu s ich výsledkami. Na základe komunikácie s rôznymi službami, ktoré ponúkajú športové dáta, sa pre začiatok zdá byť najvhodnejšou voľbou firma Trefík[5] . Spoločnosť Trefík mi ponúkla dáta v podobe XML/JSON z dvadsaťjeden športových súťaží na športové stretnutia, ich kurzy a priebežné výsledky. Pri frekvencii aktualizácie kurzov dva krát denne a aktualizácii výsledkov tridsať krát denne činí mesačný poplatok 25 EUR + DPH. Existuje mnoho služieb ktoré ponúkajú potrebné športové dáta a práve z dôvodu nejasnosti pôvodu týchto dát výslednej aplikácii chýba implementácia importu reálnych dát.

V kapitole č. 2 stručne pojednávam o gamifikácií, ktorá v produkčnej verzí aplikácie nesmie chýbať. Samotný princíp aplikácie je založený na súťažení a hre ku ktorej sa hodia rôzne druhy rebríčkov a odmien užívateľov za splnenie istých cieľov. V ďalších fázach vývoja je potrebné rozšíriť gamifikačný návrh, ktorý dodá aplikácii nový rozmer a spraví ju užívateľsky zaujímavejšiu.

Okrem spomínaných rozšírení aplikácii je potrebné implementovať aj úpravu užívateľského profilu a obrazovky všetkých druhov hier. Momentálne je implementovaná jedna z druhu hier a navrhnuté užívateľské rozhranie pre ostatné. Pri druhu hry private bude potreba navrhnúť celý proces vytvárania miestnosti a pozývania užívateľov.

---

5 Trefík. Dostupné z: https://www.trefik.cz.

Kapitola 5

5. Záver

Cieľom tejto práce bolo vytvoriť návrh mobilnej aplikácie, ktorá umožní jej užívateľom súťažiť v športovom stávkovaní a následne tento návrh implementovať pre mobilnú platformu iOS. Výnimočnosť tejto aplikácie spočíva v novom uhle pohľadu na športové tipovanie. Mnou navrhnutý systém nefiguruje v športovom stávkovaní ako súper, ako je tomu v súčasnom klasickom stávkovaní, ale ako platforma a sprostredkovateľ informácii s možnosťou užiť si športové tipovanie bez nutnosti riskovania peňažnej straty.

Pred návrhom bolo potrebné naštudovať problematiku tvorby užívateľského rozhrania a vývoja mobilných aplikácii. Následne bola na základe užívateľského výskumu definovaná cieľová skupina užívateľov, ich potreby a podrobná špecifikácia aplikácie. Ďalšou časťou bolo zostavenie informačnej štruktúry potrebnej pre návrh drôtených modelov a neskôr výsledného grafického návrhu. Poslednou časťou návrhu ból návrh serverovej časti aplikácie a komunikácie s klientskou aplikáciou. V práci ďalej pojednávam o realizácií tohoto návrhu, kde popisujem implementáciu klientskej a serverovej časti mobilnej aplikácie. Počas vývoja a na jej závere prebehlo užívateľské testovanie použiteľnosti na respondentoch rôznych skúseností v oblasti mobilných aplikácií a športového tipovania.

Výsledná aplikácia umožňuje jej užívateľovi zostrojiť si svoj ticket prostredníctvom ponuky športových stretnutí, vybrať si druh hry a vložiť zostrojený ticket do jednej z herných miestností za vstupný poplatok v podobe virtuálnych žetónov. Virtuálne žetóny má užívateľ možnosť získať v aplikácii zdarma. Následne má užívateľ možnosť sledovať výsledky nielen svojich ale aj ostatných užívateľov v miestnosti. Na základe výsledkov ticketov sa formuje výsledné umiestnenie každého z užívateľov a aj celkový víťazi jednotlivých miestností.

V budúcich fázach vývoja sa chcem zamerať na skompletizovanie implementácie serverovej časti, konkrétne importu športových dát. Zakomponovanie gamifikačných prvkov ako sú rebríčky a odmeny užívateľov a úpravu užívateľského rozhrania na základe poznatkov z testovania.

Literatúra

[1] *App stores: number of apps in leading app stores 2017 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[2] *Clean Swift iOS Architecture for Fixing Massive View Controller*. [online] Available at: http://clean-swift.com/clean-swift-ios-architecture/.

[3] Galitz, W.: The Essential Guide to User Interface Design, Wiley, 2007.

[4] Norman, D.: The Design Of Everyday Things, Basic Books, 1988.

[5] Nielsen, J.: Mental Models, http://www.jnd.org/dn.mss/human-centered. html,2005.

[6] Cooper,A.aReimann,R.aCronin,D.:AboutFace3:TheEssentialsofInterfaceDesign, Wiley, 2007.

[7] Benyon, D. a Turner, P. a Turner, S.: Designing Interactive Systems: People, Activities, Contexts,Technologies,Addison-Wesley,2005.

[8] Lanoue, S. (2017). *What is UX Design? 15 User Experience Experts Weigh In | UserTesting Blog*. [online] UserTesting Blog. Available at: https://www.usertesting.com/blog/2015/09/16/what-is-ux-design-15-user-experience-experts-weigh-in/ [Accessed 9 May 2017].

[9] Burke, B. (2014). Gamify: How Gamification Motivates People to Do Extraordinary Things. 1st ed. Bibliomotion.

[10] M. Hewstone and W. Stroebe, Sociální psychologie: moderní učebnice sociální psychologie. Praha: Portál, 2006.

Príloha A

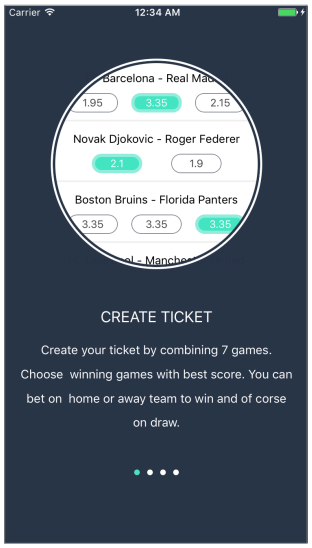
Obsah CD

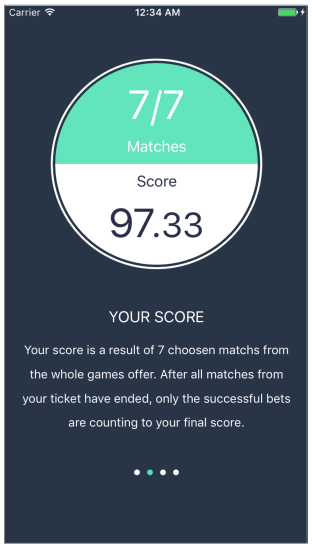
Priložené CD obsahuje:

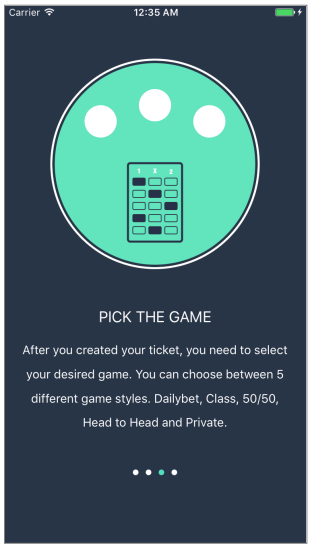
- adresár s projektom aplikácie určený pre vývojové prostredie Xcode
- technickú správu vo formáte PDF
- zdrojový tvar technickej správy
- video a plagát prezentujúci výslednú aplikáciu

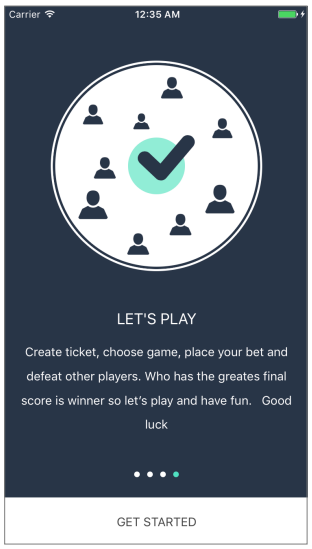
Príloha B

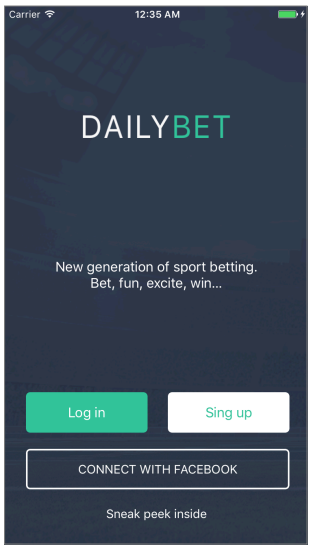
Grafický návrh

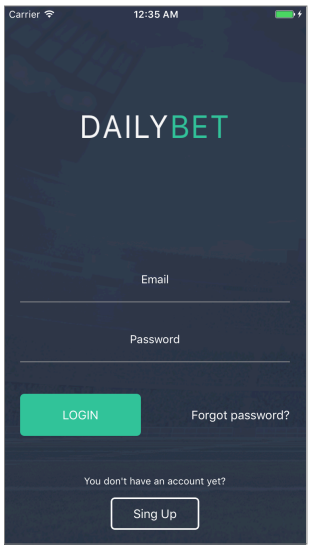


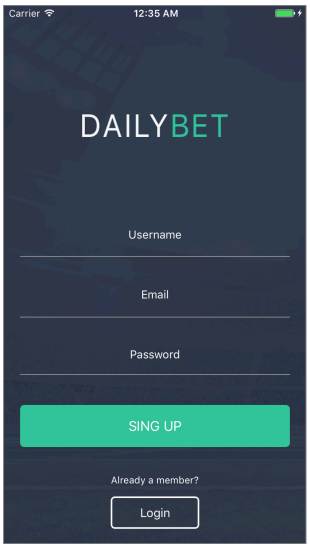


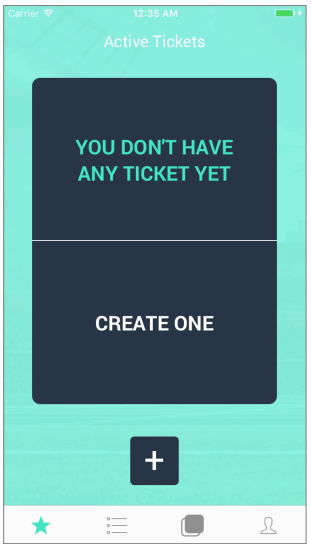


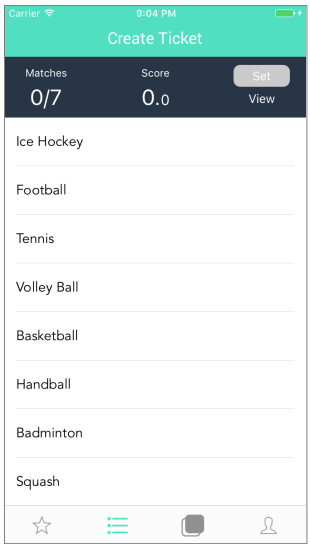


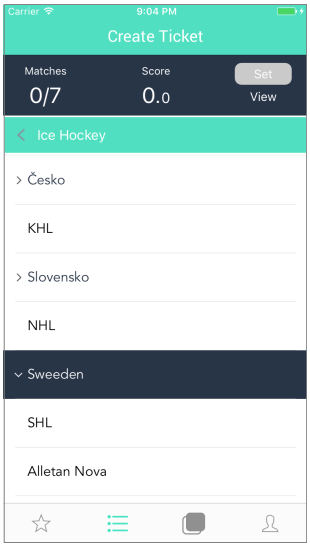


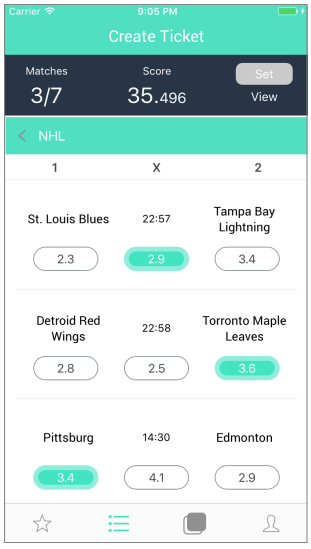


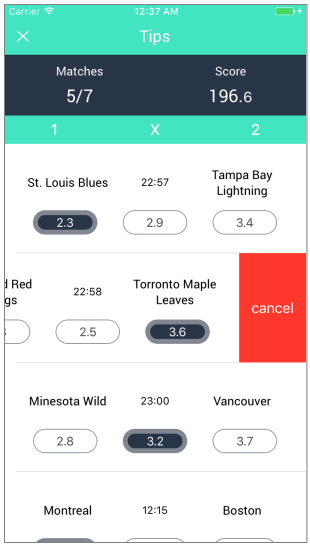

Carrier
12:35 AM
DAILYBET
Username
Email
Password
SING UP
Already a member?
Login

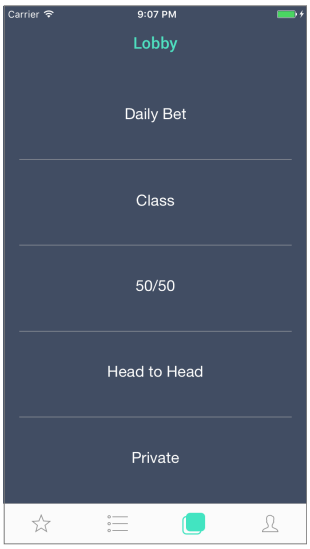

Carrier
12:35 AM
Active Tickets
YOU DON'T HAVE ANY TICKET YET
CREATE ONE

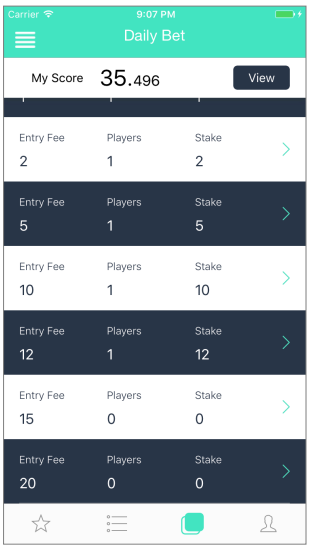

Carrier
9:04 PM
Create Ticket
Matches
0/7
Score
0.0
Set
View
Ice Hockey
Football
Tennis
Volley Ball
Basketball
Handball
Badminton
Squash

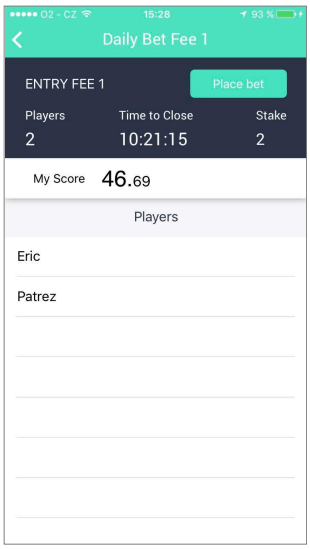

Carrier
9:04 PM
Create Ticket
Matches
0/7
Score
0.0
Set
View
Ice Hockey
Česko
KHL
Slovensko
NHL
Sweeden
SHL
Alletan Nova

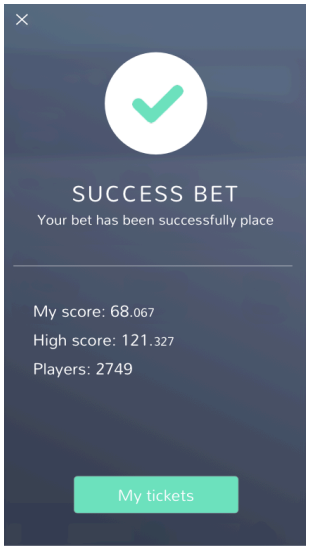

Carrier
9:05 PM
Create Ticket
Matches
3/7
Score
35.496
Set
View
NHL
1
X
2
St. Louis Blues
22:57
Tampa Bay Lightning
2.3
2.9
3.4
Detroid Red Wings
22:58
Torronto Maple Leaves
2.8
2.5
3.6
Pittsburg
14:30
Edmonton
3.4
4.1
2.9

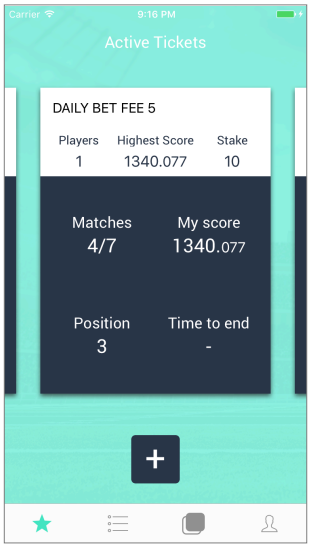

Carrier
12:37 AM
Tips
Matches
5/7
Score
196.6
1
X
2
St. Louis Blues
22:57
Tampa Bay Lightning
2.3
2.9
3.4
22:58
Torronto Maple Leaves
2.5
3.6
cancel
Minesota Wild
23:00
Vancouver
2.8
3.2
3.7
Montreal
12:15
Boston

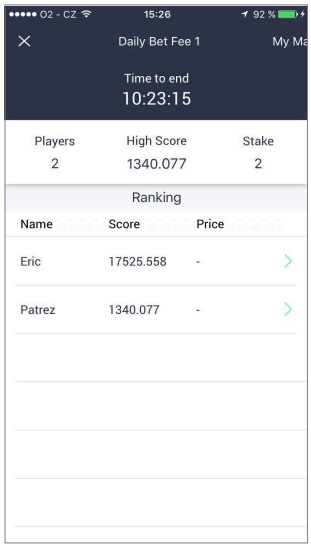

Carrier
9:07 PM
Lobby
Daily Bet
Class
50/50
Head to Head
Private

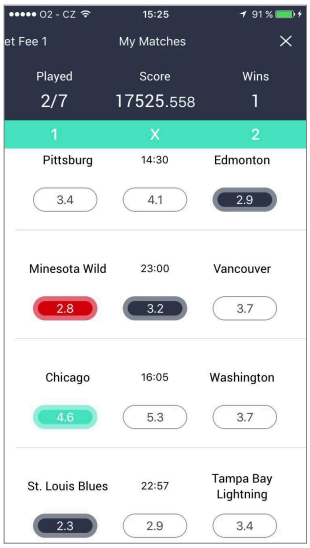

Carrier
9:07 PM
Daily Bet
My Score
35.496
View
Entry Fee
Players
Stake
2
1
2
5
1
5
10
1
10
12
1
12
15
0
0
20
0
0

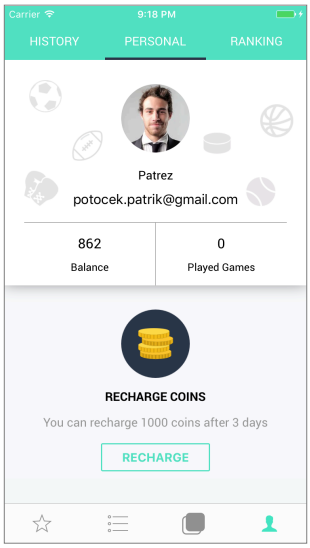

15:28
Daily Bet Fee 1
ENTRY FEE 1
Place bet
Players
2
Time to Close
10:21:15
Stake
2
My Score
46.69
Players
Eric
Patrez


SUCCESS BET
Your bet has been successfully place
My score: 68.067
High score: 121.327
Players: 2749
My tickets


Carrier
9:16 PM
Active Tickets
DAILY BET FEE 5
Players
1
Highest Score
1340.077
Stake
10
Matches
4/7
My score
1340.077
Position
3
Time to end
-


O2 - CZ
15:26
92 %
Daily Bet Fee 1
Time to end
10:23:15
Players
2
High Score
1340.077
Stake
2
Ranking
Name
Score
Price
Eric
17525.558
-
Patrez
1340.077
-


O2 - CZ
15:25
91 %
et Fee 1
My Matches
Played
2/7
Score
17525.558
Wins
1
1
X
2
Pittsburg
14:30
Edmonton
3.4
4.1
2.9
Minesota Wild
23:00
Vancouver
2.8
3.2
3.7
Chicago
16:05
Washington
4.6
5.3
3.7
St. Louis Blues
22:57
Tampa Bay Lightning
2.3
2.9
3.4


Carrier
9:18 PM
HISTORY
PERSONAL
RANKING
Patrez
potocek.patrik@gmail.com
862
Balance
0
Played Games
RECHARGE COINS
You can recharge 1000 coins after 3 days
RECHARGE

Príloha C

Testovací formulár

Sada úloh

1. Vytvoriť ticket z ponuky.
2. Vložiť ticket do jednej z miestností.
3. Zobraz si prehľad ticketov a zisti ako sa tvojmu ticketu darilo.

Otázky

Ako by ste v stručnosti aplikáciu popísali známemu/kamarátovy?

Čo vás na aplikácii zaujalo?

Čo vám v aplikácií chýba?

S čím ste mal najväčší problém?

Používali by ste túto aplikáciu a ako často?

Tvrdenia

Vždy som vedel kde sa v aplikácii nachádzam.

Vyhovuje mi spôsob zobrazenia druhov hier a ich herných miestností.

Vyhovuje mi roztriedenie ponuky športových stretnutí.

Vyhovuje mi spojenie ponuky športových stretnutí s vytváraním ticketu.

Vyhovuje mi zobrazenie prehľadu aktívnych ticketov.

Vyhovuje mi zobrazenie detailu podaného ticketu.