FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace Life is Skill pro iOS
Student:	Rostislav Babáček
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Ústav Life is Skill vznikl jako nezisková organizace se zaměřením na děti a mládež ve školním věku.
Jeho hlavní náplní je provoz projektu LiS, který motivuje děti a mládež k volnočasovým aktivitám. Cílem práce je navrhnout a implementovat aplikaci pro mobilní operační systém iOS. Aplikace bude umožňovat účast ve hře Life is Skill.
Při realizaci postupujte podle těchto kroků:

1. Popište existující backendové API
2. Specifikujte požadavky na aplikaci a proveďte vhodný softwarový návrh aplikace, využívající API z bodu 1.
3. Implementujte za použití vhodných technologií první verzi aplikace.
4. Implementaci podrobte vhodným testů. Vhodnost testů řádně podložte.
5. Funkční aplikaci nasaďte a zajistěte možnost použití, alespoň uzavřené skupině uživatelů.
6. Zhodnoťte stávající verzi aplikace, navrhněte a diskutujte možnosti rozšíření aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 3. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Aplikace Life is Skill pro iOS

Rostislav Babáček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Glazar

2. června 2020

Poděkování

Tímto bych rád poděkoval svému vedoucímu práce panu Ing. Filipu Glazarovi nejen za cenné rady, ale také za velice vstřícný a trpělivý přístup. Děkuji také zástupcům neziskové organizace Life is Skill, již mi umožnili vypracování zajímavé aplikace v rámci této práce. Speciální poděkování patří mé rodině za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 2. června 2020 . . . . . .

Odkaz na tuto práci

Babáček, Rostislav. *Aplikace Life is Skill pro iOS*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem, implementací, testováním a nasazením mobilní aplikace Life is Skill pro iOS platformu, díky níž se mohou děti zapojit do soutěže s volnočasovými aktivitami. Pro plnohodnotnou účast v soutěži musí aplikace uživatelům nabízet funkcionality, mezi něž patří možnost vyhledat soutěžní aktivity, načítat body za splněné volnočasové aktivity pomocí NFC a QR technologií, zobrazit si přehled načtených bodů a také porovnat své výsledky s ostatními soutěžícími. Hra je implementována v programovacím jazyku Swift s využitím moderních technologií pro vývoj iOS aplikací.

Výsledkem práce je mobilní aplikace, jež je dostupná v obchodu App Store a umožňuje uživatelům s iOS zařízením plnohodnotnou účast ve hře Life is Skill.

Klíčová slova hra Life is Skill, mobilní aplikace, iOS, aplikace pro děti, Swift, App Store

Abstract

This bachelor thesis deals with an analysis, proposal, implementation, testing and deployment of a mobile application Life is Skill for iOS platform, thanks to which children are able to join a competition with leisure time activities. To gain full access to the competition, the application has to offer functionalities such as a possibility to look out competitive activities, to load the credits for fulfilled leisure time activities through NFC and QR technologies, to display a review of loaded credits and also to compare one´s results with other competitors. The game is implemented in a programming language Swift together with modern technologies for iOS applications development.

The result of the thesis is a mobile application available in App Store and it enables the users with iOS equipment full access to the game Life is Skill.

Keywords Life is Skill game, mobile app, iOS, app for kids, Swift, App Store

Obsah

Seznam obrázků

Úvod

Dnešní doba moderních technologií dává mládeži velké množství příležitostí. Hry, sociální sítě a seriály shledávají jako daleko atraktivnější než aktivity, kterými se v mládí bavili naši rodiče. Děti a mladiství svůj volný čas tráví s mobilními zařízeními, u počítače nebo u televize a volnočasové aktivity jako například návštěva kina, kulturních památek, sportovišť či vycházky do přírody jsou pro mnohé výjimečnou záležitostí. Mládež žijící virtuální realitou často nezná ani okolí svého bydliště. S tímto se rozhodli skoncovat dva muži z Chomutova. Vymysleli soutěž, ve které účastníci pomocí mobilních telefonů sbírají bodíky ze soutěžních destiček umístěných na vybraných volnočasových aktivitách. Nasbírané body jsou započítány do několika soutěží, které jsou po předem specifikovaném časovém období, tzv. sezónně, vyhodnoceny. Výherci jsou nejen ti nejlepší, ale také náhodně vybraní hráči, kteří se dané soutěže zúčastnili.

Byl jsem osloven s prosbou o pomoc při vytváření iOS aplikace pro tuto hru, která má doplnit již dostupnou Android a webovou aplikaci. Myšlenka hry a implementace iOS aplikace od úplného základu mě natolik zaujaly, že jsem se rozhodl neziskové organizaci Life is Skill pomoci a vyvinul jsem pro ni plně funkční aplikaci.

V následujícím textu se budu věnovat celému softwarovému procesu vývoje mobilní aplikace neboli analýze, návrhu, implementaci, testování a nasazení. Funkční a nefunkční požadavky zformuluji podle analýzy grafického návrhu a již existujících systémů hry, tedy webové aplikace, Android aplikace a také backend API. Při návrhu a implementaci aplikace se zaměřím na použití moderních postupů a technologií pro tvorbu iOS aplikací. V neposlední řadě navrhnu vhodné strategie testování aplikace a nasadím aplikaci do obchodu App Store.

Kapitola 1

Cíl práce

Cílem bakalářské práce je vytvoření první verze aplikace pro mobilní operační systém iOS, jež bude umožňovat účast ve hře Life is Skill.

Prvotním cílem je analýza všech již existujících softwarových systémů hry Life is Skill, mezi které patří backend API, webová aplikace určená pro správu uživatelského účtu a již dostupná Android aplikace. Provedena bude také analýza grafického návrhu vytvořeného pro tuto aplikaci. Z analýzy specifikuji funkční a nefunkční požadavky na aplikaci, vytvořím vhodný softwarový návrh a následně za pomoci moderních technologií implementuji aplikaci obsahující všechny vydefinované funkcionality pro první verzi, která bude jednoduše rozšiřitelná. V neposlední řadě je cílem nasazení a zpřístupnění iOS aplikace alespoň uzavřené skupině uživatelů. Nasazení musí předcházet vhodné otestování všech funkcionalit.

Přínosem práce bude vytvoření a zpřístupnění iOS aplikace Life is Skill, díky níž se mohou do soutěže s volnočasovými aktivitami zapojit také majitelé iPhone zařízení.

Kapitola 2

Představení hry a cílové platformy

V této kapitole se věnuji představení hry *Life is Skill*, cílové platformy a rozdílných přístupů vývoje aplikací pro danou platformu.

2.1 Organizace Life is Skill

„*Ústav Life is Skill vznikl jako nezisková organizace se zaměřením na děti a mládež ve školním věku.*“ Hlavním zakladatelem je Mgr. Petr Káš, jenž přišel s myšlenkou hry Life is Skill, která motivuje děti a mládež k volnočasovým aktivitám, již v roce 2015.

Příprava této hry, jež je hlavním projektem, byla spuštěna v roce 2017. Mezi vedlejší projekty patří přednášky pro děti z dětských domovů s názvem „Chci opravdu do krimu? Která cesta je pro mě vysněná?“.[1]

2.2 Hra Life is Skill

Hlavní podstatou hry je evidence dětí a mládeže ve věku od 6 do 18 let na všech volnočasových aktivitách registrovaných v této hře, následné vyhodnocení a odměnění účastníků.

Každý účastník prokazuje svoji účast na aktivitě načtením virtuálních bodíků pomocí mobilního telefonu ze speciální soutěžní destičky. Přehled lokalit všech destiček je dostupný ve webové a mobilní aplikaci. Volnočasové aktivity se dělí do tří kategorií: sport, kultura a ekologie. Motivací pro účastníky hry není pouze dobrý pocit ze splněných aktivit a vzpomínky, ale v rámci projektu jsou vyhlašovány soutěže trvající určité časové období, tzv. sezóny. Po ukončení a vyhodnocení každé sezóny získávají účastníci hodnotné ceny od sponzorů projektu. Odměňováni ale nejsou pouze nejlepší hráči. První po-

lovinu cen získávají soutěžící s nejvyšším počtem nasbíraných bodů a druhá polovina cen je náhodně rozdělena mezi ostatní soutěžící.

V nulté sezóně, která probíhala od května do září 2019, bylo mezi účastníky rozděleno 50 cen. Mezi výhrami se nacházela jízdní kola, sportovní oblečení, poukázky na let letadlem či volné vstupné na sportoviště. Momentálně probíhá první sezóna soutěže a ke dni 22. 5. 2020 je rozmístěno 250 soutěžních destiček a hra má 431 aktivních uživatelů.[2]

Zisk bodů

Body se načítají pomocí mobilní aplikace z destičky osazené NFC čipem nebo QR kódem. Tyto destičky jsou umístěny na registrovaných zájmových místech. Přehled všech lokací destiček je v mobilní i webové aplikaci.

Bod lze načíst po splnění volnočasové aktivity nebo pouhým navštívením místa s destičkou. Dle [3] lze ve hře získat:

- **Bod za pravidelnou docházku do klubu:** „*Člen jakéhokoliv sportovního či zájmového klubu získává bod za účast na pravidelném tréninku nebo schůzce. Bod si načte na určeném místě pomocí svého mobilu.*“
- **Bod za sportovní aktivitu:** „*Bod lze získat za návštěvu sportoviště mimo členství v klubu. Bod je získán po vstupu na sportoviště a načtení pomocí mobilního telefonu. Z jednoho sportoviště lze získat pouze 1 bod za den. Pokud jsou na sportovišti různé sporty a je-li to technicky možné, může být umožněno získání bodů z různých sportů jednoho sportoviště.*“
- **Bod za kulturní aktivitu:** „*Bod za návštěvu kulturní akce. Je možné získat 1 bod z jedné akce za den.*“
- **Bod za ekologickou aktivitu:** „*Bod za účast na ekologické akci. Je možné získat 1 bod z jedné akce za den.*“
- **Bod za zájmové místo:** „*Bod je získán za návštěvu kulturní památky nebo turistické atrakce. Jedná se například o hrad, zámek, tvrz, rozhlednu apod. Lze získat 1 bod za den. U vybraných zájmových míst je možné získání bodu až po uplynutí určitého časového období od předchozího načtení bodu v rozmezí 1 týden - 6 měsíců. Tato informace je u každého takového bodu uvedena.*“
- **Bod za naučný prostor:** „*Bod je získán za návštěvu určitého naučného místa jako celku.*“
- **Bod speciální:** „*V případě pořádání jednorázové akce jsou přiděleny speciální body. Jejich hodnota je dána náročností akce a je vždy zřetelně vyznačena u jejího vyhlášení na portálu Life is Skill.*“

Pravidla načítání bodů

- **Poloha:** Aby byl bod uznán, musí být načten v blízkosti jeho umístění. Body, které nebudou načteny v těsné blízkosti destičky, nebudou uznány.
- **Čas:** Každá aktivita má své specifické časové omezení. Pokud uživatel načte volnočasovou aktivitu vícekrát v tomto limitu, bod nebude uznán.

2.3 Cílová platforma

Cílová platforma aplikace je operační systém iOS vyvíjený společností Apple, které patří druhá pozice (obr. D.1) mezi prodejci mobilních telefonů a tabletů na světě. Tento operační systém UNIXového typu běží na zařízeních iPhone, iPad a iPod touch. Hlavním rysem iOS je jednoduchost a bezpečnost, na kterou Apple primárně cílí. Apple iOS je po Androidu druhý nejpoužívanější operační systém pro mobilní zařízení na světě (obr. D.2).

„*Systém iOS tvoří základ mobilní platformy společnosti Apple. Skutečnost, že Apple ovládá celý ekosystém svých produktů, a to včetně hardwaru i softwaru, znamená, že může nabídnout co nejlepší zážitek vzájemným propojením těchto produktů a získat nejlepší výkon z hardwaru díky chytré integraci se softwarem. Rovněž umožňuje společnosti Apple nabízet pravidelné aktualizace softwaru pro všechna zařízení, aniž by vyžadovala testování a schválení od různých výrobců nebo poskytovatelů mobilních služeb.*“[4]

Velkým benefitem zmiňované integrace je dlouhodobější podpora starších zařízení novými verzemi operačního systému. Díky tomu na rozdíl od Android platformy běží většina zařízení na poslední verzi systému, která obsahuje nejnovější funkcionality a bezpečnostní aktualizace (obr. D.4).

Tento operační systém je také známý takzvanou uzavřeností. Systém lze oproti Android platformě velmi málo přizpůsobovat, aplikace žijí ve vlastní schránce a mají omezenou přímou komunikaci s ostatními aplikacemi. Díky tomuto omezení Apple dosahuje větší bezpečnosti své platformy a zabraňuje šíření virů mezi aplikacemi.[5]

Nativní aplikace pro platformu iOS lze vyvíjet pomocí jazyků Objective-C a Swift. Další variantou je multiplatformní vývoj. Možnostmi vývoje iOS aplikací se věnuji v sekci 2.5.

2.4 Průzkum verzí cílové platformy

V sekci 2.3 jsem zmínil rychlou adaptaci uživatelů na nejnovější verzi systému a dlouhodobou dostupnost nejnovějších aktualizací pro starší zařízení. Podle dostupných dat k únoru 2020 běží přibližně 70 % zařízení (obr. D.4) na nejnovějším iOS. Podle dostupných informací pro český trh (obr. D.3) má 81,5 % uživatelů nainstalovanou nejnovější verzi iOS a přibližně 96 % iPhone zařízení běží na verzi 12+. Z této statistiky a přehledu podpory iOS pro jednotlivé

modely iPhone (viz. [6]) vyplývá, že 96 % uživatelů vlastní modely iPhone 5S, iPhone 6, iPhone 6 Plus a novější.

2.5 Programovací jazyk

V této sekci se zabývám nejčastěji používaným přístupům pro implementaci mobilních aplikací, kterými jsou nativní a multiplatformní vývoj. Mezi nativní iOS programovací jazyky patří Objective-C a jazyk Swift, jenž je nástupce Objective-C a postupně jej nahrazuje. Populárními multiplatformními frameworky pro vytváření iOS aplikací jsou například ReactNative a Flutter.

2.5.1 Multiplatformní přístup

„*Aplikace jsou psány pomocí jednoho programovacího jazyka. Zdrojový kód je při vytváření distribučního balíčku přeložen do nativního kódu dané platformy. Přeložena je logika aplikace i šablony definující vzhled aplikace. Kód je přibližně z 80 % sdílený mezi platformami. Zbytek kódu je potřeba napsat v nativním kódu. Překlad kódu je prováděn samostatně pro každou platformu, kdy výsledkem jsou jednotlivé spustitelné soubory, které lze nahrát do obchodu s aplikacemi. Některé části kódu mohou být interpretovány za běhu, což může snížit rychlost aplikace. Poslední nevýhodou oproti nativním aplikacím je úprava uživatelského prostředí závislá na možnostech daného nástroje.*“ [7] Mezi populární mobilní multiplatformní programovací jazyky patří:

- **React Native:** Framework od společnosti Facebook, který umožňuje vývojářům vytvářet mobilní a webové aplikace s využitím kombinace jazyků JavaScript a XML.[8]
- **Flutter:** „*Mobilní framework pro vytváření multiplatformních nativních aplikací. K psaní Flutter aplikace se používá objektově-orientovaný programovací jazyk Dart.*“ [9]

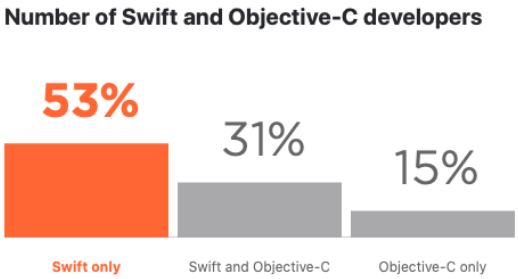
2.5.2 Nativní přístup

Aplikace jsou navrženy a implementovány pouze pro jednu specifickou mobilní platformu. Části zdrojového kódu nejsou sdíleny mezi platformami jako u multiplatformního přístupu. Nativní přístup je vhodnějxší pro komplexní aplikace řešící složitější úkoly.[10]

Objective-C

Objective-C je univerzální, objektově orientovaný programovací jazyk, který do programovacího jazyka C přidává výměnu zpráv mezi objekty ve stylu jazyka Smalltalk.[11] Tento jazyk byl hlavním programovacím jazykem společnosti Apple pro operační systémy OS X a iOS. V roce 2014 byl společností

Apple představen programovací jazyk Swift, jenž postupně Objective-C nahrazuje (obr. 2.1).



Obrázek 2.1: Swift & Objective-C v roce 2019 [12]

Swift

Programovací jazyk Swift je nástupce Objective-C určený pro iOS, macOS, watchOS, tvOS a servery. Tento výkonný a intuitivní open-source programovací jazyk je vyvíjený společností Apple a je navržen tak, aby maximálně využíval nejnovější hardware.[13] Swift je multi-paradigmatický jazyk, který zahrnuje aspekty jazyků Objective-C, Ruby, Python, Haskell, Rust, C#, D a dalších.[14]

Apple se snaží implementovat Swift tak, aby vývojářům usnadnil nejen vývoj, ale i údržbu programů. Pro Swift byly dle [13] definovány tři základní hodnoty:

- **Safe:** „*Nedefinované chování je nepřítelem bezpečnosti a chyby vývojářů by měly být zachyceny dříve, než bude software nasazen do produkce. Rozhodnutí se pro bezpečnost znamená, že Swift bude v některých momentech příliš striktní.*“
- **Fast:** „*Swift je náhrada jazyků z rodiny C, mezi které patří C, C++ a Objective-C. Proto musí být výkonnost Swiftu pro většinu úloh srovnatelná s těmito jazyky. Výkon musí být předvídatelný a konzistentní.*“
- **Expressive:** „*Swift těží z dekád pokroku v oblasti informatiky a nabízí syntax, kterou je radost používat, s moderními funkcemi, které vývojáři očekávají. Jazyk se bude neustále vyvíjet s jazykovými pokroky, aby byl ještě lepší.*“

Swift obsahuje mnoho moderních programovacích vzorů. Níže uvádím funkcionality, které mě nejvíce zaujaly:

- Proměnné jsou vždy před použitím inicializovány.
- Tzv. optionals zajišťují explicitní zpracování nil hodnoty.

- Obsahuje funkce vyššího řádu jako map, flatMap, filter a reduce.
- Typy konstant nebo proměnných jsou odvozovány kompilátorem.
- Objekty podporují extensions a protokoly.
- Swift je type safe jazyk. Přiřazení hodnot jednoho typu do jiné proměnné jiného typu je detekováno při kompilaci.
- Paměť je spravována automaticky pomocí automatického počítání referencí, tzv. ARC. Uživatel se tedy nemusí podílet na počítání referencí, ale musí důkladně zvážit závislosti mezi objekty, aby se vyhnul úniku paměti, tzv. memory leak.[15]

2.6 Vývojové prostředí

Pro implementaci iOS aplikací vyvíjí Apple vlastní IDE nazvané Xcode. Xcode nabízí vše potřebné pro implementaci, debugging a vydání aplikace. Alternativou k Xcode může být IDE AppCode od JetBrains, postavené na IntelliJ IDEA platform, které na pozadí využívá Xcode. AppCode používá Xcode simulátor, nenabízí možnost vydání aplikace a je placené.[16]

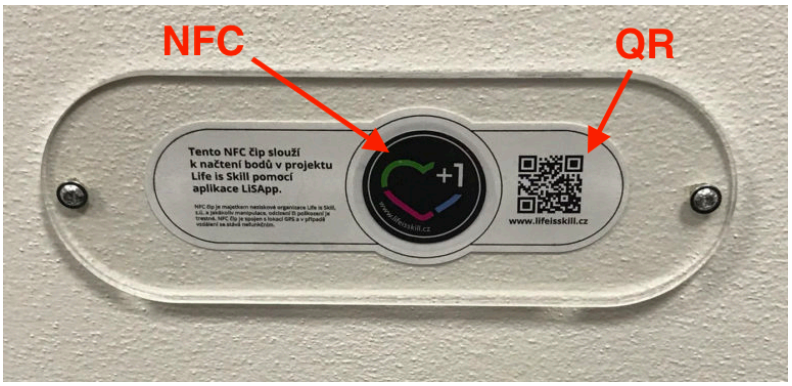
Kapitola 3

Analýza

Cílem práce je implementace první verze aplikace, která bude obsahovat funkcionality vydefinované dle konzultace s neziskovou organizací. V této kapitole analyzuji technologie pro načítání bodů, Android aplikaci, webovou aplikaci, REST API a grafický návrh pro tuto práci. Funkcionality, které jsou mimo definovaný rozsah, v analýze záměrně vynechávám. Na konci kapitoly za pomoci výsledků analýz a definovaného rozsahu specifikuji funkční a nefunkční požadavky pro první verzi aplikace.

3.1 Technologie pro načítání bodů

Body za splnění volnočasové aktivity uživatelé načítají z plastové destičky (obr. 3.1), jež je osazena NFC čipem a QR kódem.



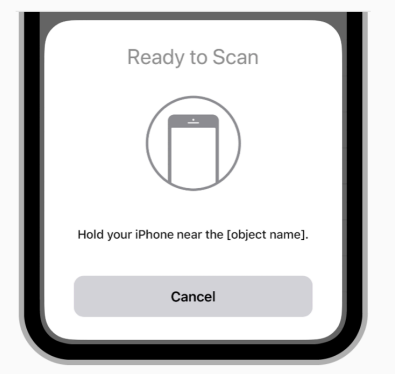
Obrázek 3.1: Life is Skill destička s bodem

QR code - Quick Response code

QR kód je dvoudimenzionální čtvercový čárový kód, který umožňuje zakódovat až 4000 znaků. Nejběžnější využití je uložení URL adresy nebo vizitky ve formátu vCard.[17] Pro načítání QR kódů je potřeba vlastnit zařízení s funkčním fotoaparátem a podporou čtení této technologie. Implementaci čtení QR kódu v iOS umožňuje AVFoundation framework od Applu, který je podporovaný od iOS 7.0.[18] Tento čárový kód není pro účely hry bezpečný, protože jej lze zkopírovat vyfocením a načíst pouhým dosažením lokality destičky bez splnění volnočasové aktivity.

NFC - Near Field Communication

NFC je bezdrátová technologie, fungující na bázi krátkých rádiových vln, umožňující rychlou a zabezpečenou výměnu dat na vzdálenost do 4 cm. V soutěžní destičce se nachází programovatelné NFC tagy, které při blízkém kontaktu s aktivním vysílačem vytvářejí elektromagnetické pole, a proto nepotřebují napájení.[19] Některé typy NFC tagů lze zabezpečit proti klonování za pomoci přístupu založeného na autentizaci. Díky zabezpečení proti klonování jsou NFC čipy pro tuto hru vhodnější. Čtení NFC je podporováno telefony od modelu iPhone 7 a lze jej na iOS implementovat pomocí Apple frameworku Core NFC, který je dostupný od iOS 11.0. Čtení NFC musí být vyvoláno akcí, která otevře tzv. NFC sheet a aktivuje čtení. Poté stačí přiložit telefon horní stranou k čipu.[20] IPhone XS, iPhone XS Max, iPhone XR a novější modely podporují čtení NFC na pozadí. Zařízení po naskenování NFC tagu zobrazí notifikaci, která po rozkliknutí vyvolá požadovanou akci jako například otevření aplikace, otevření webové stránky nebo zahájení hovoru.[21] Pro implementaci frameworku je potřeba placený developerský účet.

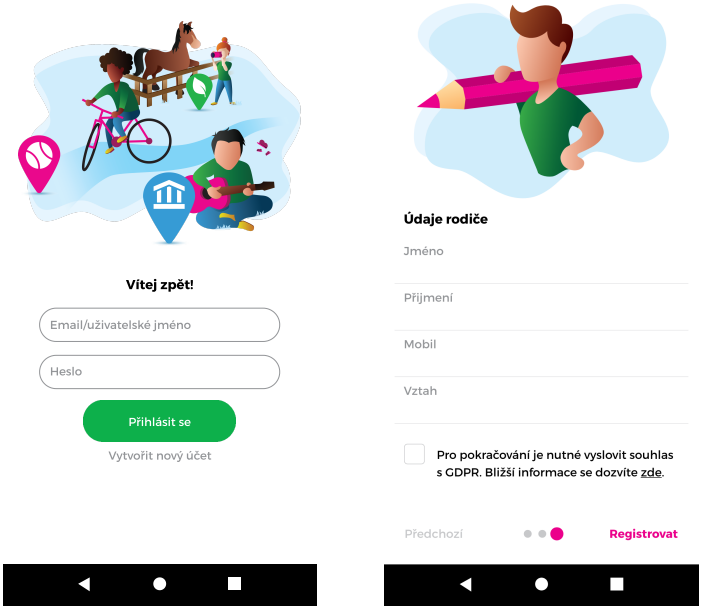


Obrázek 3.2: iOS NFC sheet [20]

3.2 Univerzální grafický návrh pro mobilní aplikace

K aplikaci byl dodán univerzální grafický návrh pro mobilní aplikaci, který vytvořil student Vysoké školy kreativní komunikace Jiří Chyla. Návrh byl vytvořen v grafickém editoru AdobeXD. Kromě designu lze také zobrazit prototyp aplikace. Některé obrazovky budu analyzovat samostatně, některé po logických celcích pro zachování kontextu funkcionalit.

Všechny obrazovky z grafického návrhu jsou umístěny v příloze B.



(a) Přihlašovací obrazovka (b) Registrace

Obrázek 3.3: Grafický návrh - Přihlášení a registrace

Přihlašovací obrazovka

Přihlašovací obrazovka (obr. 3.3) nabízí možnost přihlášení a registrace uživatele. Na obrazovce se nachází dvě textová pole, jedno pro uživatelské jméno a druhé pro heslo. Přihlášení uživatele proběhne po vyplnění těchto polí a stisknutí tlačítka *Přihlásit se*. Tlačítko *Vytvořit nový účet* se nachází pod tlačítkem pro přihlášení a otevírá obrazovku s registrací.

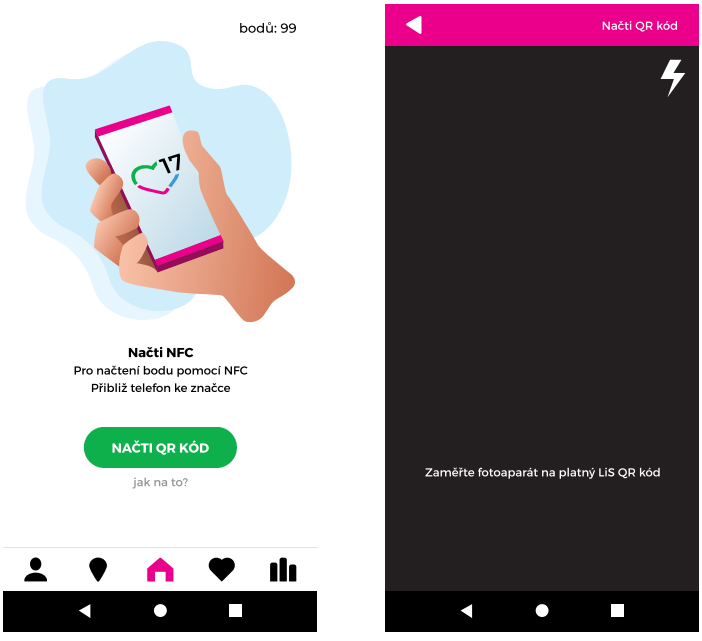
Registrace

Registrace uživatele se skládá ze tří stránek formulářů (obr. 3.3 a B.3). Pohyb mezi stránkami probíhá pomocí tlačítek na spodní straně obrazovky. První

stránka obsahuje políčka *Jméno*, *Příjmení*, *Přezdívka*, *Heslo* a *Heslo znovu*. Druhý formulář slouží k získání kontaktních a doplňujících informací o uživateli. Políčka druhého formuláře jsou: *Email*, *Mobil*, *PSČ (Poštovní směrovací číslo)*, *Datum narození* a *Pohlaví*. Po vyplnění údajů o uživateli je nutné vyplnit na poslední stránce údaje o rodiči a vyjádřit souhlas s GDPR zaškrtnutím check boxu. Formulář s údaji o rodiči obsahuje následující pole: *Jméno*, *Příjmení*, *Mobil*, *Vztah*.

Hlavní obrazovka a fotoaparát

Domovská obrazovka (obr. 3.4) se uživateli objeví po úspěšném přihlášení. Na spodní straně obrazovky je menu, které slouží k pohybu mezi jednotlivými obrazovkami aplikace. Z popisku umístěného uprostřed obrazovky vyplývá, že načítání pomocí NFC je aktivní při zobrazení domovské obrazovky. Pod tímto popiskem se nachází tlačítko, které slouží ke skenování QR kódu. Aplikace po stisknutí tlačítka *načti QR kód* přesměruje uživatele na obrazovku s kamerou. Pod tlačítkem umožňujícím čtení QR kódu se nachází tlačítko s nápovědou. V pravém horním rohu obrazovky je umístěna informace o počtu bodů, které uživatel již nasbíral.



(a) Hlavní obrazovka (b) Kamera

Obrázek 3.4: Grafický návrh - Domovská obrazovka

Mapa registrovaných bodů v soutěži

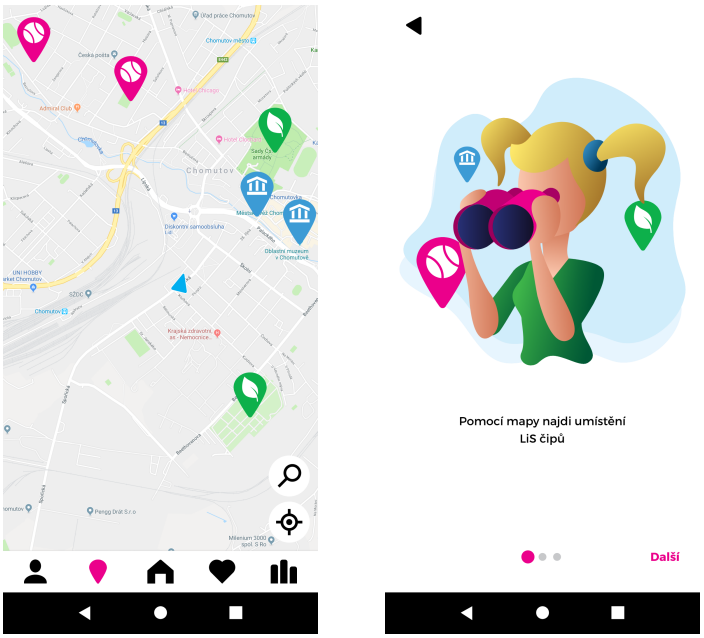
Druhou položkou ve spodním menu je přehled lokalit volnočasových aktivit (obr. 3.5). Celou obrazovku pokrývá mapa s body zájmu, tzv. piny, pro jednotlivé aktivity. Každý pin má barvu a ikonu dle druhu volnočasové aktivity.

- **Sportovní aktivita:** Růžové pozadí a ikona tenisového míčku.
- **Kulturní aktivita:** Modré pozadí a ikona chrámu.
- **Ekologická aktivita:** Zelené pozadí a ikona listu.

Na mapě je také zobrazena aktuální poloha uživatele pomocí modrého trojúhelníku. Uživatel může zaměřit svou polohu pomocí příslušného tlačítka.

Nápověda

Obrazovka s nápovědou (obr. 3.5 a B.2) se skládá ze tří stránek, mezi nimiž se uživatel posouvá pomocí tlačítek umístěných na spodní straně obrazovky. Nápověda má uživatele seznámit s principy soutěže. V dolní polovině obrazovky je pro každou stránku umístěn text s nápovědou a nad ním se nachází grafika, která je s tímto textem tematicky spojena. První obrazovka se věnuje nalezení bodu, druhá načtení bodu a třetí dané aktivitě.



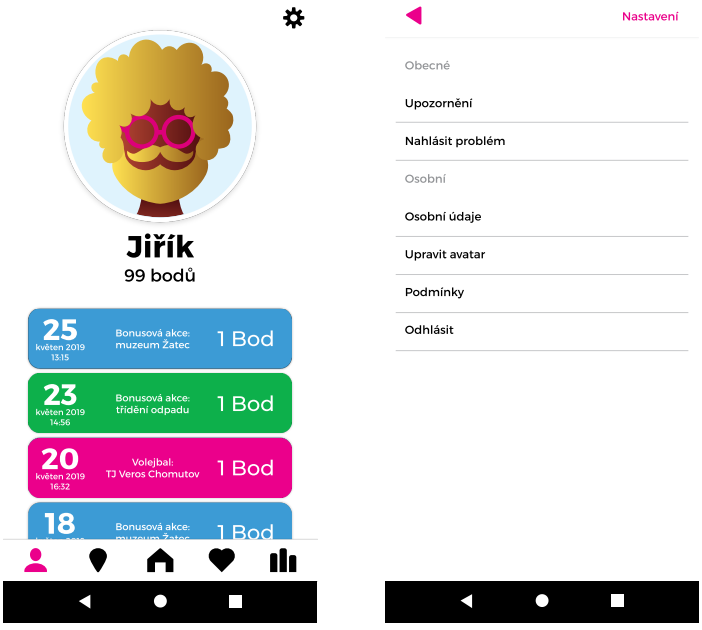
(a) Mapa (b) Nápověda

Obrázek 3.5: Grafický návrh - Mapa a nápověda

Přehled načtených bodů a nastavení

První stránkou z menu je přehled uživatelského účtu (obr. 3.6), který poskytuje hráči informace o počtu nasbíraných bodů a historii jejich načítání. Historie načítání je zobrazena pomocí tabulky, ve které každá buňka představuje načtený bod. Buňka obsahuje informaci o datu načtení, název a hodnotu bodu. Pozadí buňky odpovídá kategorii bodíku. V hlavičce tabulky se nachází avatar hráče a údaj o celkovém počtu nasbíraných bodů. V pravém horním rohu je umístěno tlačítko, s jehož pomocí si může uživatel otevřít nastavení.

Nastavení se skládá z tabulky se dvěma sekcemi. Sekce *Obecné* obsahuje položky *Upozornění* a *Nahlásit problém*. Druhá sekce, nazvaná *Osobní*, se skládá z polí *Osobní údaje*, *Upravit avatar*, *Podmínky* a *Odhlásit*.

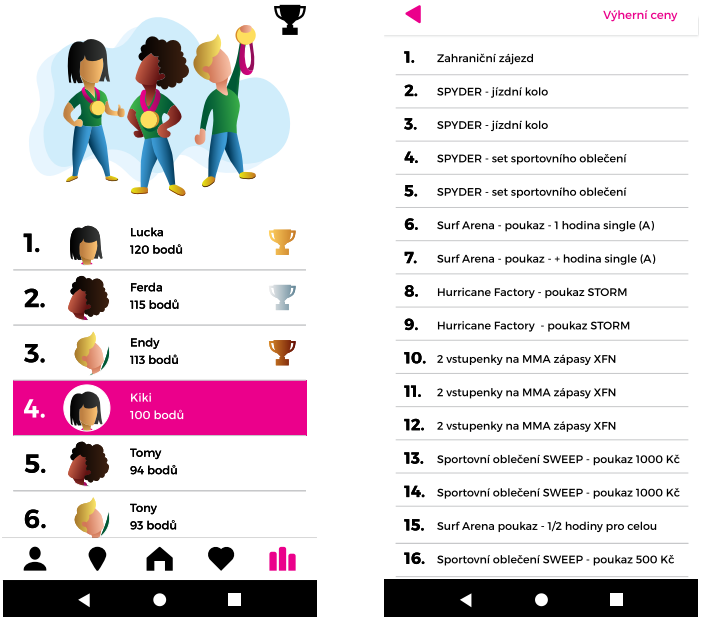


(a) Přehled bodů (b) Nastavení

Obrázek 3.6: Grafický návrh - Přehled načtených bodů

Žebříček a přehled výher

Žebříček soutěže (obr. 3.7) je zobrazen tabulkou. Každá buňka tabulky obsahuje pořadí, avatar, jméno a počet bodů uživatele. Buňky pro první tři místa obsahují obrázek poháru na pravé straně buňky, jenž odpovídá pozici hráče. Buňka s umístěním přihlášeného uživatele je zvýrazněna růžově. V pravém horním rohu se nachází tlačítko s obrázkem poháru, který odkazuje na obrazovku s tabulkou výher.



(a) Žebříček (b) Přehled výher

Obrázek 3.7: Grafický návrh - Žebříček a přehled výher

Mé zhodnocení grafického návrhu

Grafický návrh obsahuje pestrou grafiku a hodí se k tématu aplikace. Některé obrazovky budou muset být přizpůsobeny pokynům pro uživatelské rozhraní, tzv. Human Interface Guidelines, společnosti Apple.[22] Buňky v tabulce historie načítání bodů obsahují mnoho informací a budou muset být upraveny tak, aby se vešly na displej menších zařízení a jednotlivá textová pole se nepřekrývala. Dle mého názoru aplikaci chybí podrobnější představení hry pro nové uživatele a možnost zobrazení pravidel před registrací. Na úvodní stránce by mohl být umístěn odkaz na webové stránky.

Hlavním nedostatkem grafického návrhu je absence uživatelského testování.

3.3 Webová aplikace

Webová aplikace hry Life is Skill (dále jen LiS) je určena pro správu uživatelského účtu. Grafické rozhraní vychází ze stejného grafického základu jako grafický návrh pro mobilní aplikace (viz 3.2). Aplikace se skládá ze dvou hlavních obrazovek.

Přihlašovací obrazovka nabízí uživateli možnost přihlášení nebo registrace. Oproti grafickému návrhu pro mobilní aplikace je nepřihlášenému uživateli do-

stupný přehled lokalit bodů, pravidla soutěže, přehled partnerů soutěže a kontaktní údaje s kontaktním formulářem.

Po kliknutí na tlačítko *Vytvořit nový účet* se otevře modální okno s jedním registračním formulářem. Validace jednotlivých polí se provede po stisknutí tlačítka *Registrovat*. Chybové hlášky se zobrazí na pravé straně od jednotlivých textových polí.

Po přihlášení se uživateli zobrazí hlavní stránka (obr. D.7), která obsahuje všechny potřebné informace k účasti v soutěži. V horní části této stránky se nachází avatar uživatele s přezdívkou, počtem bodů a pozicí v aktuální sezóně. Stránka je rozdělena na tři sloupce. První polovinu levého sloupce vyplňuje tabulka s přehledem získaných bodů z aktuální a minulé sezóny. Buňky tabulky nejsou konzistentní s grafickým návrhem pro mobilní aplikace. Ve spodní části se nachází žebříček aktuální a minulé sezóny. Prostřední, dominantní část stránky vyplňuje přehled bonusových akcí, mapa s lokalitami bodů a plovoucí přehled partnerů. Lokality bodů lze filtrovat podle kategorie. Pravá část stránky zobrazuje přehled výher.

Webová aplikace je po grafické stránce konzistentní s grafickým návrhem pro mobilní aplikaci. Jedinou zásadní inkonzistenci vidím v možnosti výběru bodů z aktuální a minulé sezóny a také ve filtrování bodů na mapě podle kategorie.

3.4 Android aplikace

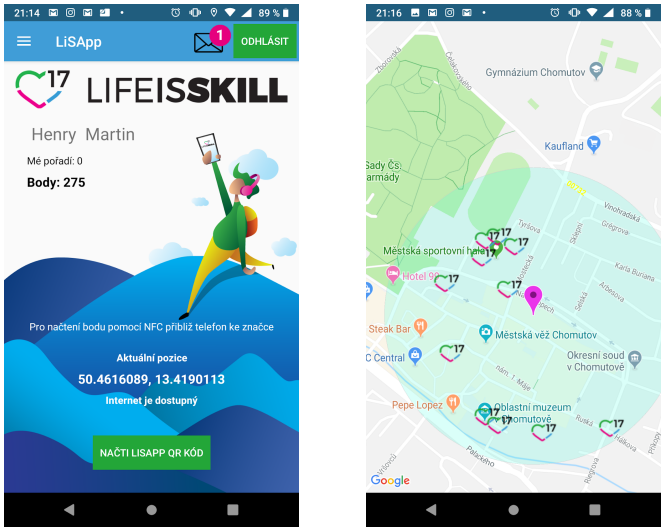
Android aplikaci (obr. 3.8) lze stáhnout na Google Play od května roku 2019, kdy startovala nultá sezóna hry. Dostupná je všem uživatelům s Android platformou od verze 4.4. Jedná se o pilotní verzi, ve které uživatelské rozhraní nevychází z dodaného grafického návrhu pro mobilní aplikace a v době psaní této práce se pracuje na redesignu.

Hlavní funkcí aplikace je načítání bodíků ze soutěžních destiček. Body lze načítat pomocí NFC a QR technologií. U zařízení podporujícího čtení NFC je ihned po přihlášení aktivní NFC čtečka a načtení bodu lze dosáhnout pouhým přiblížením k čipu. Pro čtení QR kódu musí být vyvoláno spuštění fotoaparátu stisknutím příslušného tlačítka.

Mezi další funkcionality aplikace patří:

- Zobrazení základních údajů o uživatelském účtu - počet bodů, umístění, přezdívka
- Registrace
- Zobrazení mapy s lokalitami soutěžních bodů
- Zobrazení načtených bodů
- Zobrazení zpráv

- Zobrazení bonusových akcí
- Možnost kontaktování organizátora soutěže přímo z aplikace



(a) Hlavní obrazovka (b) Mapa

Obrázek 3.8: Ukázka android aplikace

3.5 Backend REST API

REST je architektura API, která nám umožňuje pracovat s daty. „*Tato architektura implementuje čtyři základní metody, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.*“[23]

HTTP je aplikační protokol, jenž je založen na principu požadavek/odpověď. Klient pošle požadavek a server mu odpoví. Důležitou informací obsaženou v požadavku je metoda, která říká, jaký typ operace klient po serveru požaduje.[24] Mezi základní operace patří:

- GET - získání objektu
- POST - vytvoření nového objektu
- PUT - aktualizování objektu
- DELETE - smazání objektu

K existujícímu REST API neexistuje dokumentace. Byly mi poskytnuty ukázkové requesty a odpovědi. K dispozici jsou dvě prostředí, aby vývoj aplikace a změny v backendu neovlivnily probíhající sezónu. První, tzv. stage, slouží k vývoji. Druhé, označované jako produkce, je určeno pro verzi aplikace, jež je přístupná uživatelům. Backend API obsahuje dvě adresy pro zasílání požadavků, tzv. endpointy.

Základní popis endpointů

První endpoint poskytuje zdroj dat pro mapu. Přehled všech umístěných bodíků získám zavoláním *GET* metody na příslušný endpoint. API vrací kolekci všech bodů v XML formátu. Odpověď obsahuje kolekci *marker* elementů znázorňující pin na mapě. Všechny informace daného prvku jsou obsaženy v atributech elementu (obr. 3.9).

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<markers>
        <marker ID="289" pCode="190720082558" Value="1" Lat="50.631701" Lng="13.57232"
        Name="Sport areál Klíny" Type="1" Address="Klíny 27, 436 01 Klíny" Spec="0"
        Cluster="" IsValid="1" InGroup="1"/>
</markers>
```

Obrázek 3.9: REST API XML

Druhý endpoint umožňuje zavolání specifických úloh pomocí hodnoty klíče *task*. Mezi dostupné úlohy patří:

- Přihlašování - *login*
- Registrace uživatele - *register*
- Kontrola existence uživatelského jména - *checknick*
- Kontrola existence uživatelského e-mailu - *checkemail*
- Kontrola platného PSČ - *checkpsc*
- Odeslání načteného bodu - *point*
- Historie načítání - *listpoints*
- Žebříček - *rank*

Obsah požadavků i odpovědí je ve formátu JSON. Formát klíčů requestů je konzistentní na rozdíl od klíčů odpovědí, kde klíče začínají na velká, či malá písmena. Některé klíče jsou v anglickém jazyce, jiné pak v jazyce českém.

Validace platnosti načteného bodu probíhá na API a není potřebné v aplikaci implementovat danou logiku.

Autentizace

Autentizace probíhá pomocí uživatelského jména a příjmení. Odpověď vrací identifikátor uživatele, který se používá v requestech jako položka dictionary. Každý request požaduje kromě identifikátoru uživatele také položku značící verzi aplikace a unikátní identifikátor konkrétní aplikace.

Zhodnocení

Všechny požadavky jsou srozumitelné a plní úlohy, které jsou očekávány. Některé odpovědi vracejí zbytečné položky, které mohou být zavádějící. Příkladem může být hodnota klíče *Valid* obsažená v odpovědi na požadavek validace bodu, která však nemá nic společného s výsledkem validace.

Odpověď requestu pro kontrolu uživatelského jména (obr. 3.10) vrací položky *Task*, *Message*, *Result*, *UserValid*, *Valid*. Položka *Valid* vyjadřuje výsledek provedené operace. Očekával bych, že výsledek bude vracet hodnotu *true* pro volné uživatelské jméno, ale vrací hodnotu *false*. Položky *Resul* a *UserValid* nehrají žádnou roli a vždy nabývají stejných hodnot. *Message* vrací řetězec s popisem výsledku. Kontrola poštovního směrovacího čísla naopak vrací hodnotu *true* klíče *Valid* pro úspěch.


```
{
  "Task" : "checknick",
  "Message" : "Uživatelské jméno je k dispozici",
  "Result" : "OK",
  "UserValid" : true,
  "Valid" : false
}
```


(a) Je k dispozici


```
{
   "Task" : "checknick",
   "Message" : "Uživatelské jméno je již používáno",
   "Result" : "OK",
   "UserValid" : true,
   "Valid" : true
}
```


(b) Již používáno

Obrázek 3.10: Kontrola uživatelského jména

Odpověď validace bodu obsahuje položku *Valid*, která ale pro uznaný i neuznaný bod vrací hodnotu *true*. Informace o výsledku validace je obsažena v hodnotě klíče *Message*, který obsahuje doprovodný text o výsledku validace v několika větách. Bez této zprávy není možné zjistit výsledek.

API chybí jakékoliv handlování problémů pomocí *status code*. Všechny výsledky požadavků jsou předávány pomocí položek odpovědi.

Časový údaj o načtení bodu se posílá jako dva samostatné řetězce pro datum a čas ve formátu *dd.MM.yyyy* (např. 06.09.2019) pro datum a *HH:mm:ss* (např. 11:56:29) pro čas. Požadavek na získání historie načítání bodů vrací časový údaj v ISO 8601 formátu *yyyy-MM-dd HH:mm:ss* (např. 2019-09-05 11:56:29). Tato inkonzistence přináší do aplikace složitou logiku náchylnou na chyby.

Za nešikovné řešení považuji použití XML pro jeden endpoint a JSON pro endpoint druhý. Změna z XML jazyka na JSON by sjednotila práci s parsováním odpovědí serveru.

Navrhovaná zlepšení

Navrhuji několik změn týkajících se backend API, které by vedly ke zjednodušení logiky síťové vrstvy aplikace:

- API přijímá a vrací stringové datum a čas v několika formátech. Doporučuji použít tzv. *Timestamp*, který vyjadřuje počet sekund od počátku roku 1970. Tento formát je univerzální pro všechny regiony a z aplikace by odpadla logika s převodem několika různých formátů času.

- Neúspěchy requestů by mohly být vraceny errorem s příslušným statusovým kódem a případnou detailní informací v těle. Kontrola uživatelského jména by mohla kopírovat dotaz do tabulky a vracet status code *200* pro již používanou hodnotu a *400* pro volnou hodnotu. Neplatné načtení bodu by mohlo vracet error se specifickým status code a v těle vracet informace o chybě.

- Klíče odpovědí serveru by měly být konzistentní. Navrhuji změnit formát klíče tak, aby každý klíč začínal malým písmenem a byl vždy v anglickém jazyce.

- Odpovědi requestu pro validaci bodu detekují správnost načtení bodu pomocí řetězce znaků. Navrhuji přidat položky pro detekci chyby validace místa a času načteného bodu.

- Odpovědi obou endopointů bych zasílal ve stejném formátu.

3.6 Analýza požadavků aplikace

Funkční a nefunkční požadavky aplikace byly specifikovány z analýz provedených v této kapitole. Následně byly vybrány pouze ty, které jsou potřebné pro první verzi aplikace.

3.6.1 Funkční požadavky první verze aplikace

Funkční požadavky definují základní chování systému. V zásadě jde o definici, co systém dělá, nebo nesmí dělat.[25] Požadavky musí být jednoznačné, splnitelné a ověřitelné.

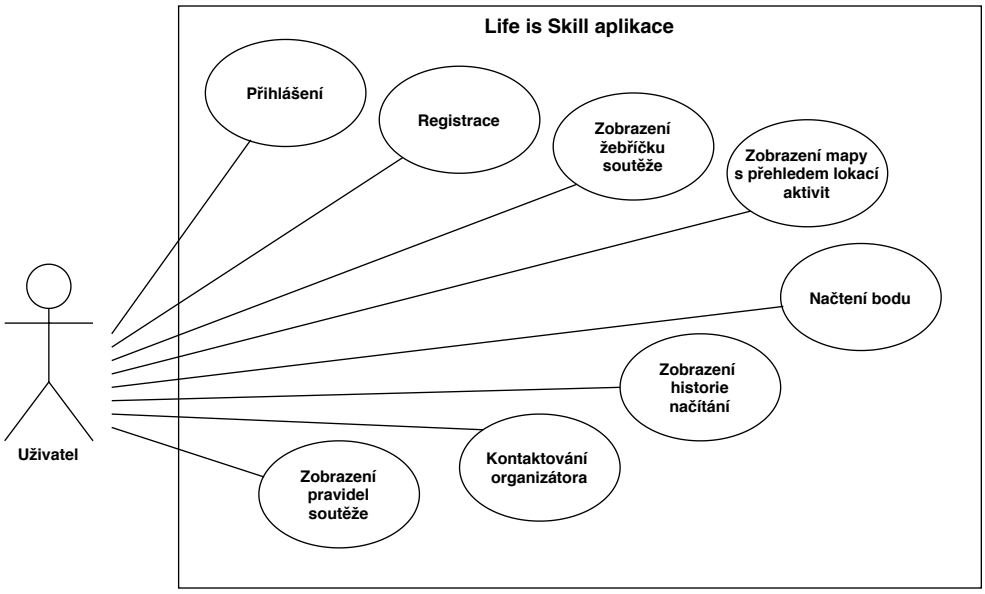
- **F1 Přihlášení:** Uživatelům je umožněno přihlášení pomocí uživatelského jména a hesla.
- **F2 Odhlášení:** Uživatel se může odhlásit ze svého uživatelského účtu.
- **F3 Registrace:** Uživatelé mohou v aplikaci vytvořit uživatelský účet, který je podmínkou pro účast ve hře. Registrovat se mohou pouze hráči ve věku od 6 do 18 let. Vytvoření účtu není umožněno hráčům, kteří neodsouhlasí GDPR.
- **F4 Načtení bodu:** Aplikace umožňuje načtení bodíku pomocí technologie dostupné na daném zařízení. Bodíky lze načítat čtením NFC čipu nebo naskenováním QR kódu.
- **F5 Zobrazení historie uživatelem načtených bodíků:** Uživatel má možnost zobrazit historii bodů, které úspěšně načetl během aktuální sezóny.
- **F6 Zobrazení mapy s přehledem lokací bodů:** Účastník soutěže si může pomocí mapy zobrazit umístnění a doprovodné informace všech dostupných bodů. Mapa zobrazuje polohu uživatele. Uživatel nevidí polohu jiných uživatelů.
- **F7 Zobrazení žebříčku soutěže:** Aplikace umožňuje všem uživatelům zobrazit žebříček právě probíhající sezóny s pořadím všech hráčů zapojených do této sezóny.
- **F9 Zobrazení pravidel soutěže:** V aplikaci budou dostupná aktuální pravidla soutěže.
- **F10 Kontaktování organizátora:** Uživatel může přímo z aplikace kontaktovat organizátora.

3.6.2 Nefunkční požadavky aplikace

„Nefunkční požadavky jsou doplněk funkčních požadavků. Popisují další nezbytné vlastnosti systému. Jedná se například o požadavky na spolehlivost, bezpečnost, výkonnost nebo dostupnost.“[26]

- **N1 Dostupnost:** Aplikace bude dostupná přes obchod AppStore.
- **N2 Responzivita:** Uživatelské rozhraní bude responzivní pro všechny velikosti obrazovek zařízení iPhone.

- **N4 Stabilita aplikace:** Aplikace bude stabilní a nebude docházet k neočekávaným pádům.
- **N5 Software bude rozšiřitelný:** Software bude možné jednoduše rozšířit o další funkcionality.
- **N6 Udržitelnost:** Udržování aplikace nebude náročné. Kód bude zdokumentovaný a nebude používat zastaralé technologie.



Obrázek 3.11: Use case diagram aplikace

Kapitola 4

Návrh

V této kapitole definuji na základě analýzy vhodné technologie pro implementaci této práce. Nejdříve volím programovací jazyk, ve kterém budu aplikaci vyvíjet, a také vhodné vývojové prostředí. Sekce 4.2 se zabývá minimální podporovanou verzí iOS. V sekci 4.4 představuji funkcionálně reaktivní programování, jež se dle výsledků analýzy hodí pro tento druh aplikace, a na závěr volím vhodnou architekturu.

4.1 Zvolený jazyk a vývojové prostředí

Android aplikace je pro hru Life is Skill již implementována. Nativní jazyky Objective-C a Swift jsou určeny přímo pro implementaci iOS aplikací, a proto jsou pro tuto platformu optimalizovány a mají okamžitou podporu nových funkcí systému. Z nativních jazyků jsem si vybral jazyk Swift, protože je nástupce Objective-C a obsahuje funkcionality současných moderních a populárních jazyků (viz 2.5.2).

Jako vývojové prostředí jsem zvolil Xcode, jenž obsahuje všechny potřebné funkcionality a nástroje pro vývoj iOS aplikací. Dalším důvodem je oproti AppCode bezplatná dostupnost Xcode IDE pro uživatele Mac zařízení.

4.2 Podporovaná zařízení

Při výběru minimální verze iOS, dále deployment target, je potřeba brát v úvahu cílovou skupinu LiS aplikace, umístění bodíků na území České republiky a rozšiřitelnost či stabilitu aplikace. Nastavením deployment targetu na iOS 11.0 bude podle průzkumu z 2.4 aplikace dostupná pro 96 % uživatelů, ale odeberu podporu pro zařízení iPhone5 a iPhone 5C, která uživatelé stále používají, a dá se předpokládat, že půjde převážně o mladší vlastníky hlavně z důvodu finanční náročnosti Apple produktů. Těchto zařízení je na českém

trhu maximálně okolo 4 %. Čím nižší deployment target nastavím, tím těžší bude přechod na nejnovější SDK (soubor nástrojů pro vývoj softwaru).

Deployment target byl po zvážení výše zmíněných faktorů stanoven na iOS 10.3. Aplikace bude dostupná pro více než 96 % uživatelů se zařízeními iPhone (obr. D.3) od modelů iPhone 5 a iPhone 5C. Jsem si vědom faktu, že přechod na nejnovější budoucí SDK může znamenat odebrání podpory větší skupině zařízení iPhone, než by tomu bylo po nastavení minimální verze na iOS 11.0.

4.3 Podpora technologií načítání bodů

Zařízení iPhone do modelu iPhone 6S včetně (viz 3.1) nepodporují čtení NFC. Těmto zařízením bude umožněno načítání bodů QR kódem. Novější modely podporují načítání bodů pomocí bezpečnější technologie NFC, která bude pro tyto modely hlavním způsobem načítání. Čtení QR kódů bude v aplikaci dostupné jako alternativa k NFC.

4.4 Funkcionálně reaktivní programování

Mobilní aplikace reagující na akce uživatelů a komunikující s API vyžadují paralelní zpracování úkolů neboli zpracování několika úkolů současně. Dále také transformaci dat z API na modelové třídy používané v aplikaci a předávání velkého množství callbacků neboli funkce, jež je vykonána právě tehdy, když je jiná funkce ukončena. V aplikaci použiji knihovnu kombinující reaktivní a funkcionální programování, která usnadní asynchronní zpracování událostí a propagaci výsledku úloh.

„***Reaktivní programování*** *je programovací paradigma, které se zabývá asynchronními datovými toky (sledy událostí) a šířením změn, které jsou prováděny v určeném pořadí.*“ [27]

Funkcionální programování je programovací paradigma, ve kterém je základní metodou výpočtu aplikace funkcí na argumenty.[28] Hlavními znaky funkcionálního programování jsou výpočty pomocí matematických funkcí, neměnitelnost a minimalizování použití proměnných a rozdílných stavů.

Funkcionálně reaktivní programování kombinuje koncept reaktivního a funkcionálního programování. Mezi nejpopulárnější funkcionálně reaktivní knihovny v iOS patří RxSwift, ReactiveCocoa & ReactiveSwift či Combine. Ve své práci jsem si vybral ReactiveCocoa a ReactiveSwift (obr. 4.1).

4.5 Architektura

Softwarová architektura popisuje rozdělení aplikace na komponenty a vztahy mezi nimi.[29] Pro správný návrh aplikace je důležité zvolit vhodnou architekturu. Její výběr představuje klíčové rozhodnutí v počáteční fázi projektu,

```
1  // Jakmile se změní `user`, pak zobraz počet bodů uživatele v `pointsLabel`
2  pointsLabel.reactive.text <~ user.producer.map { "Počet bodů: " + $0.points }
3
4  // Do `invalidPoints` vyber všechny body, které
5  // nejsou validní a seřaď je podle data načtení.
6  //
7  // Provede se pokaždé, když se změní hodnota `points`
8  invalidPoints <~ pointsRepository.points.producer
9      .map { $0.filter { point -> Bool in
10         point.isValid == false
11     }}
12     .map {
13         $0.sorted { (lhs, rhs) -> Bool in
14             lhs.timestamp < rhs.timestamp
15     }}
```

Obrázek 4.1: Ukázka reaktivně funkcionálního programování

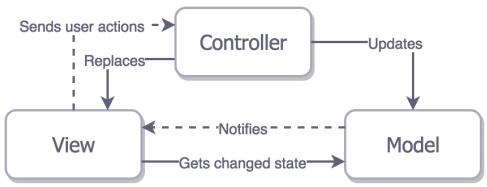
jelikož pozdější změny jsou velmi náročné. Mezi nejpopulárnější architektury pro vývoj iOS aplikací patří MVC (*Model-View-Controller*) a MVVM (*Model-View-ViewModel*). Mezi další oblíbené architektury patří MVP (*Model-View-Presenter*) a VIPER.

4.5.1 MVC (Model-View-Controller)

Tradiční MVC

„*MVC architektura (obr. 4.2) byla navržena v sedmdesátých letech vývojáři Smalltalku. Jelikož Smalltalk silně ovlivnil programovací jazyk Objective-C ve firmě NeXSTEP, která se později spojila s Applem, softwarová architektura MVC se stala de facto standardem pro vývoj na Apple platformě.*“[30]

MVC odděluje zodpovědnosti architektury, tudíž každá komponenta vykonává pouze jednu činnost. *Model* ukládá data, *View* zobrazuje data a *Controller* vhodně manipuluje s modelem a reaguje na události uživatele.[30]

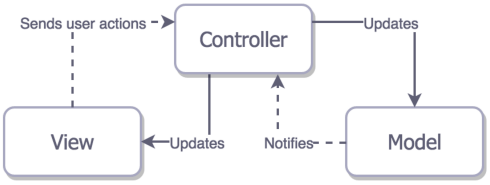


Obrázek 4.2: Schéma MVC [31]

Apple MVC

V tradičním MVC jsou všechny tři entity blízce svázané a každá entita ví o dalších dvou. Z důvodu blízkého provázání entit, které způsobuje špatnou znovupoužitelnost jednotlivých entit, je MVC v moderním iOS developmentu upraveno.[31] Schéma Apple MVC se nachází na obrázku 4.3. Controller je

prostředník mezi View a Modelem, a tak View a Model o sobě neví. Nejméně znovupoužitelný je Controller, který obsahuje většinu business logiky aplikace.



Obrázek 4.3: Schéma Apple MVC [31]

Výhody Apple MVC

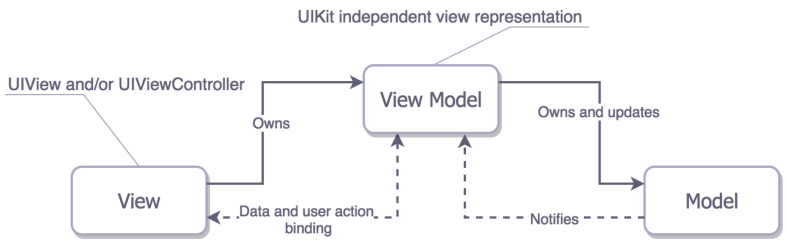
- MVC architektura je iOS standardem. Většina Apple frameworků jako UIKit, AppKit, and WatchKit tuto architekturu podporuje.[30]
- Odděluje zodpovědnosti do samostatných komponent. View a Model nejsou vůbec svázány.
- Modely jsou lehce testovatelné.
- Modely a View jsou snadno znovu použitelné.

Nevýhody Apple MVC

- Controller a View jsou často blízce svázány a je těžké je oddělit. Aplikace velmi často obsahují masivní Controllery (obr. D.5).
- V některých případech je těžké rozhodnout, jestli daná zodpovědnost patří do View nebo Controlleru.
- View a Controllery je obtížné testovat, protože obsahují prvky knihovny UIKit.[31]

4.5.2 MVVM (Model-View-ViewMode)

MVVM (obr. 4.4) je architektura, která byla navržena jako variace MVC. Tato architektura vychází z MVC a MVP (Model-View-Presenter). Martin Fowler v roce 2004 přišel s MVP alternativou k MVC. Představil nový objekt nazvaný *Presentation model*, který ukládá stav aplikace odděleně od uživatelského rozhraní.[30] Tento objekt je využíván v MVVM. *Model* v MVVM architektuře ukládá data. *ViewModel* obsahuje business logiku aplikace a *View* zpracovává interakci uživatele a zobrazuje data. V iOS MVVM je *View* označováno jako *ViewController*.[31]



Obrázek 4.4: Schéma MVVM [31]

Výhody MVVM

- Velkou výhodou MVVM je testovatelnost. Žádné dvě komponenty nejsou vzájemně úzce svázány. V aplikaci s MVVM architekturou můžeme lehce testovat ViewModel bez Controlleru. Pro otestování Controlleru nám stačí pouze mock, tzn. náhražka za reálný objekt, v tomto případě ViewModel.
- V aplikaci nevznikají masivní Controllery.
- Zodpovědnosti komponent jsou rozděleny rozumně, tudíž jim dokáže porozumět i začátečník.

Nevýhody MVVM

- Největší nevýhodou MVVM architektury je složité debbugování bindings, jež nám umožňují propagovat změnu dat na obrazovku a opačně. Příčina chyby se hledá složitěji než u klasického imperativního jazyka, kde stačí nastavit breakpointy.
- U malých projektů s použitím MVVM narůstá složitost kódu.[32]

4.5.3 Zvolená architektura

Pro projekt jsem zvolil MVVM architekturu hlavně z důvodu odstínění business logiky do samostatného ViewModelu a jeho snazší testovatelnosti. Kombinace reaktivního programování a MVVM se přímo nabízí, jelikož MVVM požaduje prvky obsažené v reaktivním programování, jako jsou bindings mezi komponentami ViewModel a View.

Pro usnadnění založení projektu použiji open-souce šablonu *iOS MVVM Project Template* od programátorů společnosti *AckeeCZ*. Šablona obsahuje počáteční nastavení projektu, vzorové rozdělení komponent podle MVVM a dále například obecnou síťovou vrstvu. Díky této počáteční konfiguraci nemusím nastavovat projekt a mohu se rovnou pustit do programování.

Kapitola 5

Realizace

V této kapitole popisuji zavedené postupy či principy, které jsem při návrhu aplikace neplánoval, ale bylo vhodné či nutné zavést je při implementaci.

5.1 Síťová vrstva

Aplikace je závislá na síťové komunikaci s API. Jak jsem v 3.5 nastínil, komunikace s API není jednoduchá, protože je potřeba využívat až devět zdrojů dat. API používá XML a JSON formát a je nutné provádět složitější mapování inkonzistentních odpovědí na objekty, které využívám v aplikaci.

Proto jsem se držel šablony zmíněné v 4.5.3 a oddělil síťovou vrstvu. Logika sestavení requestů, komunikace s API a mapování odpovědí na objekty používané aplikací probíhá v *APIService* objektech. Metody těchto objektů tedy přijímají a vrací objekty, které se používají v aplikaci. Díky této vrstvě odpadá ViewModel objektům zodpovědnost za jejich mapování.

5.2 Repository objekty

Informace o počtu bodů uživatele je zobrazena na obrazovce určené k načítání čipů a také na přehledu načtených bodů. Všechny *ViewModel* objekty musí znát identifikátor uživatele, jenž je získán z odpovědi requestu pro přihlášení, aby mohly komunikovat skrz síťovou vrstvu s API. Refresh historie načtených bodů může vyvolat sám uživatel, nebo je vyvolán automaticky po úspěšném načtení a ověření bodu.

Všechny tyto operace vyžadují stejné akce nebo stejný zdroj dat, a proto jsem v aplikaci vytvořil singleton objekty nazvané *Repository*, jež sjednocují duplicitní operace, komunikují se síťovou vrstvou a slouží také jako úložiště sdílených dat. Celkem byly vytvořeny čtyři tyto objekty: *UserRepository*, *PointRepository*, *CompetitionRepository* a *LocationRepository*.

ViewModel objekty využívají reaktivní programování, zmiňované v 4.4, k řízené propagaci změn sdílených hodnot z repository objektů, s nimiž dále pracují.

Jako příklad bych rád uvedl *UserRepository* objekt, jenž byl vytvořen pro správu dat o uživateli. Tento objekt řídí přihlašování a udržuje informace o uživatelském účtu, počtu bodů uživatele, pozici uživatele a nasbíraných bodech uživatele. Obrazovky zobrazující informaci o počtu bodů uživatele pozorují proměnnou s počtem bodů na *UserRepository* a hodnota se při jakékoliv změně propaguje na tyto obrazovky.

Přidáním *Repository* objektů jsem snížil počet zodpovědností *ViewModel* objektů. Také jsem odstranil duplicitní operace, jelikož transformace, např. seřazení či změna formátu, stejných proměnných by musela proběhnout v každém *ViewModel* objektu zvlášť.

5.3 Protocol oriented programming

Všechny *ViewModel*, *Repository* a *APIService* objekty mají vlastní protokol definující rozhraní, jež řídí přístup k objektu (obr. D.6). Při implementaci je nejdříve vytvořen protokol a až poté se vytváří třída z něj dědící. Protokol obsahuje v názvu příponu *-ing*, například *PointViewModeling*.

5.4 Dependency management

V aplikaci používám knihovny třetích stran, mezi něž patří například ReactiveSwift a SnapKit (viz 5.6). Pro koordinaci integrace závislostí třetích stran do softwaru se používá tzv. dependency manager neboli správce závislostí. Mezi populární dependency managery v iOS patří CocoaPods, Carthage a Swift Package Manager. V tomto projektu jsem pro správu závislostí použil CocoaPods a Carthage.

CocoaPods

CocoaPods je open-source dependency manager pro Swift a Objective-C Cocoa projekty, který obsahuje přes 70 tisíc knihoven. CocoaPods mají centralizovaný repozitář všech knihoven, které mohou být použity v Xcode projektech. Jsou velmi oblíbené kvůli rozmanitosti knihoven, které obsahují, ale také kvůli snadnému přidání nových závislostí. Mezi hlavní nevýhody patří build time. Při každém sestavení aplikace se musí sestavit každá závislost, což často vede k delšímu build timu.[33]

Carthage

Carthage stejně jako CococaPods je také open-source dependency manager pro Swift a Objective-C projekty. Carthage nemá centrální úložiště, je tedy

decentralizovaný. Hlavní výhodou oproti CocoaPods je build time, který je výrazně kratší, protože sestavuje frameworky pouze jednou při přidání závislosti a poté pouze při aktualizaci. Mezi hlavní nevýhody patří složitější přidání nové závislosti oproti CocoaPods, menší podpora frameworků a počáteční kompilace závislostí, která často trvá v řádu jednotek minut.[33]

Závislosti použité v aplikaci

- ReactiveSwift a ReactiveCocoa - reaktivní knihovny
- Alamofire - síťová knihovna poskytující elegantní rozhraní
- SnapKit - auto layout DSL knihovna
- SwiftLint - knihovna pro statickou analýzu kódu
- ACKReactiveExtensions - rozšíření pro ReactiveSwift
- ACKategories - swift nástroje, cocoa podtřídy a rozšíření
- Codextended - rozšíření pro elegantnější transformaci JSON objektů do Swift objektů
- Crashlytics - podrobné zprávy o pádech aplikace
- Firebase Analytics - knihovna pro statistiky aplikace
- Google Maps a Google Places - google mapa

5.5 Verzování

Uchovávání historie vyvíjeného softwaru je zásadní pro zálohu starších verzí, vývoj nových funkcionalit bez ztráty původní verze a také pro distribuci souborů mezi vícečlenný tým. Pro snadné verzování se používá správce verzí.[34] Správce verzí je systém, který zaznamenává změny souboru nebo sady souborů v čase tak, abychom se mohli později k určité verzi vrátit. Správce verzí umožňuje nejen vrátit software do předchozího stavu bez ztráty nových funkcionalit, ale také porovnat změny souborů provedených v čase či uchovávat záznam o tom, kdo a kdy jakou část upravil.[35] Mezi známé verzovací systémy patří Git, Subversion a Mercurian.

Verzování LiS aplikace

Pro správu verzí jsem použil GitHub, který poskytuje hosting a rozhraní pro verzovací systém Git. Všechny verze jsou uchovány v GitHub repozitáři, ve kterém jsem vytvořil tři stálé branche:

- **Development:** Branch sloužící pro vývoj nových funkcionalit.

- **Stage:** V této větvi se nachází aplikace, která je testována před vydáním na AppStore.
- **Master:** Větev udržující verze aplikace, které byly vydané a jsou dostupné uživatelům.

Verzování v systému Git probíhá obvykle pomocí příkazové řádky. Pro pohodlnější proces verzování používám SourceTree software, který vytváří grafické uživatelské prostředí pro Git.

Proces verzování při vývoji nové funkcionality

1. Z development branche vytvořím novou větev s názvem *feature/názevNovéFunkcionality*
2. Novou funkcionalitu vyvíjím ve vytvořené větvi. Pokud je funkcionalita rozsáhlá, vytvořím si další větev pro část vyvíjené funkcionality.
3. Po dokončení vývoje funkcionality zamerguji development do aktuální verze a spustím testy.
4. Vytvořím pull request z aktuální branche do development branche. Zkontroluji nově přidané funkcionality a opravím případné syntaktické chyby.
5. Po potvrzení pull requestu pokračuji prvním bodem pro vývoj nové funkcionality. V případě dokončení všech funkcionalit aktuální verze aplikace vytvořím pull request development větve do stage větve.
6. Zkontroluji pull request, zamerguji development větev a zpřístupním verzi pro testování.
7. Pro opravu chyb v testované verzi vytvořím ze větev *bugfix/názevChyby*. Chybu v této větvi opravím a vytvořím pull request do stage větve.
8. Po otestování a opravení všech chyb zamerguji stage větev do master větve.
9. Přidám tag aktuální verze ve formátu *číslo Verze/počet Commitů*.
10. Verzi aplikace z master větve nasadím.

5.6 Implementace uživatelského rozhraní

Xcode nabízí pro implementaci uživatelského rozhraní Interface Builder (obr. D.8). „*Editor uživatelského rozhraní Interface Builder v Xcode usnadňuje vytvoření plného uživatelského rozhraní bez nutnosti psaní kódu. Jednoduše přetáhněte okna, tlačítka, textová pole a další objekty na návrhové plátno a vytvořte funkční uživatelské rozhraní.*“ [36] Velkou výhodou Interface Builderu je okamžitý náhled vloženého UI prvku.

Alternativou ke storyboards používaných v Interface Builderu je implementace uživatelského rozhraní pomocí kódu. Znovupoužití částí UI a verzování aplikace patří mezi výhody implementace uživatelského rozhraní kódem.

Uživatelské rozhraní v LiS aplikaci je implementováno pomocí kódu. Pro definici vztahů mezi jednotlivými prvky UI používám SnapKit, jednoduché DSL pro Auto Layout.

```
1  // Without SnapKit
2  lblTimer.translatesAutoresizingMaskIntoConstraints = false
3  NSLayoutConstraint.activate([
4    lblTimer.widthAnchor.constraint(equalTo: view.widthAnchor, multiplier: 0.45),
5    lblTimer.heightAnchor.constraint(equalToConstant: 45),
6    lblTimer.topAnchor.constraint(equalTo: viewProgress.bottomAnchor, constant: 32),
7    lblTimer.centerXAnchor.constraint(equalTo: view.centerXAnchor)
8  ])
9
10 // With SnapKit
11 lblTimer.snp.makeConstraints { make in
12   make.width.equalToSuperview().multipliedBy(0.45) // 1
13   make.height.equalTo(45) // 2
14   make.top.equalTo(viewProgress.snp.bottom).offset(32) // 3
15   make.centerX.equalToSuperview() // 4
16 }
```

Obrázek 5.1: Ukázka SnapKit knihovny [37]

5.6.1 Úskalí návrhu a implementace UI

Při návrhu a implementaci uživatelského rozhraní je potřeba myslet na různé velikosti obrazovek. Apple nabízí produkty ve velikosti od 4”do 6,5”. Pokud bereme při implementaci v potaz pouze jednu velikost, často dochází k překrývání či úplnému skrytí prvků, které se nevejdou na menší obrazovku. Na větší obrazovce mohou vznikat velké mezery mezi prvky uživatelského rozhraní.

Nejen velikost obrazovky může způsobit problémy. Každá platforma má své standardy pro návrh uživatelského rozhraní. Google používá na Android platformě tzv. „Material Design“, Apple má své pokyny pro vytváření UI, tzv. „Human Interface Guidelines“, které sjednocují všechny Apple platformy. Úplné odklonění od těchto pokynů může vést k zamítnutí distribuce aplikace do obchodu App Store.

5.6.2 Implementace uživatelského rozhraní v LiS App

Návrh uživatelského rozhraní dodaný k této aplikaci byl vytvořen ve formátu univerzálního designu pro mobilní platformy. Některé prvky proto musely být přizpůsobeny Human Interface Guidelines od společnosti Apple. Projekt se při implementaci také vyvíjel a bylo potřeba reagovat na změny přidáním, či odebráním některých prvků.

S organizací Life is Skill jsem byl domluven, že se v některých případech mohu mírně odklonit od návrhu a navrhnout změny, které budou dále konzultovány s tvůrcem grafického návrhu.

V této podsekci popisuji změny grafického uživatelského rozhraní, jež bylo vhodné či nutné provést. Screenshoty finální podoby hlavních obrazovek se nachází v příloze C.

5.6.2.1 Přihlášení

Obrazovka přihlášení (obr. C.1) až na jednu výjimku odpovídá grafickému návrhu. Do pravého horního rohu jsem přidal ikonu, jež po stisknutí otevře modální okno s nápovědou (obr. C.2).

5.6.2.2 Nápověda

Nápověda zachycená na obrázcích C.1 a C.2 se stejně jako v grafickém návrhu skládá ze tří stránek. Okno nápovědy se zobrazuje zespodu nahoru jako modální okno, a proto jsem tlačítko *trojúhelníku* pro akci zpět nahradil ikonou *křížku* pro zavření. Posun mezi jednotlivými stránkami je zajištěn pomocí swipe gesta, proto není potřeba implementovat speciální tlačítka, jež tento posun zajišťují.

5.6.2.3 Tab Bar a Navigation Bar

Tab Bar i Navigation Bar (obr. 5.2) slouží pro navigaci v aplikaci. Tab Bar je fixní komponenta, tzn. tato komponenta nereaguje například na posun tabulky, která se nachází ve spodní části obrazovky a skládá se z tzv. záložek, jež představují obrazovky aplikace. Navigation Bar je umístěn naopak úplně nahoře a fixní být nemusí. V této práci se na většině obrazovek Navigation Bar hýbe s tabulkou. Pro tyto komponenty jsem oproti grafickému návrhu (obr. 3.7) navrhl redesing neboli změnu návrhu.

Grafický návrh na obrazovkách přehledu bodů, žebříčku a načítání bodíků nebere v potaz nativní Navigation Bar, jenž je vizuálně oddělen a prvky v této komponentě mají stejnou barvu jako písmo použité v aplikaci. Tlačítka umístěná v Navigation Bar komponentě splývají s textem a není zřejmé, že se jedná o prvky, které po kliknutí vyvolají akci. Proto jsem změnil barvu těchto prvků na šedou, abych vizuálně oddělil obsah obrazovky od navigační lišty.

Tab Bar je typická komponenta pro iOS aplikace. Bez podrobnější znalosti grafického návrhu (obr. 3.7) bych nebyl schopen rozpoznat, k jakému účelu slouží dvě z pěti ikonek. Z tohoto důvodu jsem navrhl kompletní redesign této komponenty. Za prvé jsem přidal každé záložce co nejkratší doprovodný text, který jasně definuje obrazovku, již zastupuje. Následně jsem k textům vybral vhodné ikony, které jasně definují, na jakou obrazovku záložka odkazuje. Ikony jsem volil tak, aby měly téměř totožné proporce. Barvu položek jsem volil

ekvivalentně k Navigation Bar komponentě, abych sjednotil navigační prvky a zároveň je oddělil od obsahu obrazovky.

Změny designu těchto komponent byly konzultovány jak se zástupci Life is Skill, tak také autorem grafického návrhu. Změny byly zapracovány a při plánovaném uživatelském testování dojde k vyhodnocení, která varianta je pro uživatele přívětivější.

(a) Navigation Bar (b) Tab Bar

Obrázek 5.2: Navigation Bar a Tab Bar

5.6.2.4 Skenování bodu

Jak jsem zmínil v sekci 3.1, technologie NFC je dostupná pouze pro některé verze iPhone zařízení, a proto jsem vytvořil dva stavy obrazovky pro skenování bodů (obr. C.5).

Platforma iOS neumožňuje automatické skenování NFC na pozadí, a proto jsem vytvořil tlačítko pro spuštění čtení NFC, které chybí v grafickém návrhu. Tímto tlačítkem jsem nahradil tlačítko pro skenování QR kódu, jež jsem umístil místo nápovědy pod nové NFC tlačítko. Následně jsem vytvořil nové tlačítko pro nápovědu a umístil jsem jej do pravé části navigační lišty. Informaci o počtu bodů jsem přesunul na levou stranu navigační lišty a přidal akci, jež po stisknutí přesměruje uživatele na přehled načtených bodů. Pro zařízení, která podporují pouze QR technologii, se zobrazuje tlačítko pro čtení dle UI návrhu.

Po stisknutí tlačítka pro čtení QR kódu se otevře obrazovka s kamerou. Po kliknutí na NFC tlačítko ztmavne pozadí a objeví se NFC sheet zmiňovaný v sekci 3.1. Oba případy jsou zachyceny na obrázku C.6.

5.6.2.5 Žebříček

Obsah stránky žebříčku, jenž můžeme vidět na obrázku C.10, je totožný s grafickým návrhem. Na levou stranu od Navigation Bar komponenty jsem přidal údaj o pořadí uživatele, jenž po stisknutí posune tabulku soutěže na pozici uživatele (obr. C.10).

5.6.2.6 Profil hráče s přehledem bodů

Finální podoba obrazovky profilu hráče s přehledem načtených bodů se nachází na obrázku C.7.

Buňky tabulky načtených bodů byly přepracovány podle webové aplikace (obr. D.7). Nad avatara uživatele bylo stejně jako ve webové aplikaci přidáno

růžové kolečko obsahující pořadí uživatele v soutěži, jež po stisknutí přesměruje uživatele na obrazovku s žebříčkem.

5.6.2.7 Nastavení

Snímek obrazovky nastavení se nachází na obrázku C.7. Titulek je pro nativní Navigation Bar umístěn uprostřed této komponenty, proto se oproti grafickému návrhu nachází uprostřed. Tab Bar není schován, jelikož Apple doporučuje neskrývat Tab Bar manuálně a ponechat nativní chování. Tlačítko *zpět* bylo implementováno pomocí nativního prvku.

Do zápatí tabulky jsem umístil nové ikony pro sociální sítě, jež uživatele přesměrují na konkrétní Life is Skill profil či stránku.

Grafický návrh obsahuje položky, které nejsou zahrnuty do této verze aplikace. Naopak v průběhu implementace byly přidány nové položky: *Pravidla*, *Partneři*, *Nápověda*, *Sdílej nás kamarádům* a *O aplikaci*. Nastavení jsem na základě těchto změn rozdělil na sekce *Hra LiS* a *Obecné*.

5.6.2.8 Registrace

Registrace byla implementována dle grafického návrhu. Pro indikátor stránky formuláře, výběr data narození a výběr pohlaví byly použity nativní komponenty. Finální podoba registrace se nachází na obrázcích C.3 a C.4.

Pole vyžadující odeslání requestu pro validaci jsou validována po stisknutí tlačítka pro posun na další krok registrace. Ostatní textová pole se validují po změně obsahu. Chybová hláška validace se zobrazuje červeným písmem namísto nadpisu políčka.

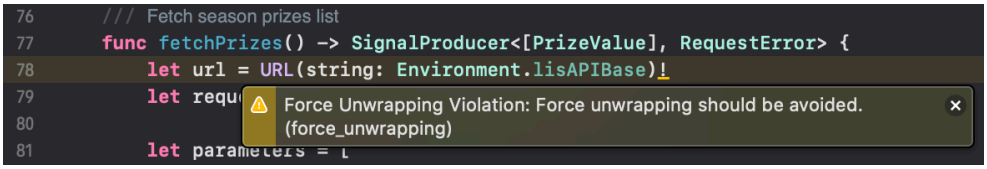
Kapitola 6

Testování

Každá aplikace musí být před zpřístupněním uživatelům řádně otestována. V této kapitole se věnuji třem druhům testování, které byly provedeny před vydáním každé verze aplikace.

6.1 Statická analýza kódu

Statická analýza kódu nevyžaduje spuštění programu, a proto je možné testovat i fragmenty aplikace, které jsou stále v procesu vývoje a nejsou zatím spustitelné.[38] Pro statickou analýzu kódu byla použita knihovna *SwiftLint*. Tato knihovna před každou kompilací testuje zdrojový kód pomocí desítek pravidel, která si může uživatel upravovat či přidávat. Mezi kontrolované aspekty patří například syntax a konvence jazyka Swift, nepoužité importy a parametry bloků, přebytečné bílé znaky a vynucené přetypování. Za skvělou funkcionalitu považuji také označení *todo* komentářů. SwiftLint nabízí možnost zvolit si, které prohřešky by měly přerušit kompilaci nebo které by měly být označeny pouze jako upozornění, tzv. warning.



Obrázek 6.1: Statická analýza kódu

6.2 Unit testy

Unit testy jsou malé izolované automatizované testy, které vykonávají jednu úlohu a kontrolují její výsledek. Pokud se výsledek neshoduje s referenčním

řešením, test skončí s chybou. Unit testy testujeme struktury od malých metod až po třídy.[39]

K unit testování bylo využito testovací prostředí, které nabízí Xcode. Pro každý testovaný objekt jsem vytvořil vlastní testovací třídu.

Díky vhodnému výběru architektury a struktury projektu bylo možné podrobit testování všechny objekty s business logikou od jednoduchých metod až po metody využívající závislosti. Pro testování metod závislých na jiných objektech jsem vytvořil *Mock* objekty závislostí.

Při testování je důležité zvolit vhodnou strategii a testovat správné i nesprávné chování. V testování jsem se zaměřil na:

- různé datové formáty zmiňované v 3.5
- validace v registraci
- transformace API odpovědí na modelové třídy aplikace
- metody ViewModel objektů
- metody Repository objektů
- metody APIService objektů
- propagaci změn u reaktivního kódu

Unit testování byla podrobena celá business logika aplikace. Vytvořil jsem 104 unit testů s pokrytím 23 %. Xcode počítá pokrytí jako počet procent řádků kódu z celku, které byly podrobeny testování. Tato informace není příliš vypovídající, jelikož více jak polovina všech řádků kódu je spojena s implementací prvků uživatelského rozhraní, které není možné testovat unit testy.

6.2.1 Testování reaktivního kódu

Reaktivní programování (viz. 4.4) přináší mnoho benefitů, mezi které patří usnadnění vytvoření asynchronních operací. Asynchronní operace vykonávají úlohu na více vláknech současně a tomuto chování je potřeba přizpůsobit testy.

```
1  func testGetStats() {
2      let expectation = self.expectation(description: "testGetStatsExpectation")
3      // Nastartuj pozorování hodnot proměnné `stats`
4      competitionRepository.stats.producer.skip(first: 1).startWithValues { _ in
5          // Splň očekávání, jestliže přijde hodnota
6          expectation.fulfill()
7      }
8      // Spusť akci k získání statistiky
9      competitionRepository.getStats.apply().start()
10     // Počkej na splnění `expectation` po dobu 5 sekund
11     wait(for: [expectation], timeout: 5)
12     XCTAssertEqual(competitionRepository.stats.value.count, statsAPIService.statsArray.count)
13 }
```

Obrázek 6.2: Appstore Screenshots

Pro testování asynchronních operací je nutné zajistit, aby funkce, jež má testovat danou funkcionalitu, neskončila dříve, než se provede úloha zavolaná na jiném vlákně. Pro testování asynchronních operací má v sobě Swift implementovány funkce a speciální proměnné. Nejdříve je potřeba vytvořit proměnnou (obr. 6.2), tzv. očekávání, které bude čekat na splnění asynchronního kódu. Do bloku, ve kterém je možné pozorovat výsledek úlohy spuštěné na jiném vlákně, je potřeba přidat akci, která očekávání splní. Dále je potřeba před vyhodnocením správnosti výsledku asynchronní operace přidat příkaz, který čeká po námi stanovenou dobu na splnění proměnné očekávání. Pokud není očekávání v definovaném časovém limitu splněno, testovací scénář selže. Pokud splnění proběhne v námi stanovém časovém okamžiku, pak se čekání odblokuje a mohou být spuštěny testy ověřující výsledek testované funkcionality.

6.3 Beta testování

Aplikace byla vydána ve čtrnácti verzích podle předem definovaných skupin funkcionalit. Jednotlivé verze musely před vydáním projít testováním v reálném prostředí. Na testování jsem se nepodílel pouze já, ale také osoby z organizace Life is Skill. Jednotlivé buildy verzí určené k beta testování jsem vydával pomocí Xcode do TestFlight prostředí (obr. D.9), které je součástí Apple Store Connect portálu. V TestFlight prostředí je možné určit testery, zobrazit jim informace o buildu, získávat feedback a také zobrazit data o pádech aplikace, jež vzniknou při testování. Testerům jsou jednotlivé buildy přístupné pomocí TestFlight iOS aplikace (obr. D.10).

Beta testování pomocí TestFlight odhalilo několik chyb a ukázalo se jako velmi účinné. Odhaleny byly pády aplikace při čtení QR kódu, které nejde nasimulovat na simulátoru z důvodu absence podpory fotoaparátu pro simulátor. Beta testování také odhalilo chyby při reálné komunikaci s API, které není možné odhalit mockováním.

6.4 Firebase analytics a crashlytics

V aplikaci používám Firebase analytics a crashlytics nástroje, které mi dodávají důležité informace a statistiky o aplikaci a uživatelích.

Crashlytics je nástroj pro shromažďování dat o pádech aplikace. V případě pádu aplikace je možné získat informace o zařízení, operačním systému a verzi aplikace. Nástroj nabízí také výpis zásobníku operací jednotlivých vláken, takže je možné zjistit, jaké operace předcházely pádu aplikace, na jakém řádku kódu a z jakého důvodu aplikace havarovala. Díky tomuto nástroji je možné okamžitě a efektivně reagovat na případné pády aplikace. Crashlytics jsem integroval do komunikační aplikace Slack. Díky této integraci dostanu

pomocí zprávy okamžité upozornění se základními informace a odkazem na detailní informace.

Analytics jsou nástroje pro sledování aktivity uživatelů. Hlavními zajímavými informacemi jsou statistiky aktivity uživatelů, adaptace uživatelů na nové verze, přehled zařízení uživatelů a statistika využívanosti jednotlivých funkcionalit aplikace.

Kapitola 7

Nasazení a budoucnost aplikace

Cílem práce byla nejen implementace aplikace, ale také nasazení a zpřístupnění uživatelům, aby se soutěže Life is Skill mohli účastnit uživatelé s iOS platformou. S organizací Life is Skill jsme si předem stanovili rozsah jednotlivých verzí tak, abychom nejdůležitější funkcionality zpřístupnili co nejdříve.

7.1 Appstore

Jediným oficiálním místem pro stažení aplikací do iOS zařízení je *AppStore*, obchod s aplikacemi, jenž je známý vysokými nároky na aplikace. „*Apple je extrémně vybíravý v tom, které z aplikací pustí do svého App Store. Nerad by si totiž pokazil svou dobrou reputaci, a proto se tam dostanou jen kvalitní a dostatečně otestované aplikace.*“ [40]

7.2 Vydání nové verze aplikace

Před vydáním aplikace je potřeba založit Apple developerský účet, připravit ikonu aplikace, zkontrolovat, jestli aplikace splňuje podmínky App Store (viz 7.2.1), a vytvořit ukázkové video či snímky obrazovky k představení aplikace (viz 7.2.2).

7.2.1 App Store Review Guidelines

„*Hlavní princip obchodu App Store je jednoduchý. Uživatelům chceme poskytnout bezpečný zážitek a všem vývojářům příležitost, aby byli úspěšní.*“ [22]

Vydání nové verze aplikace předchází kontrola, která je prováděna Apple testerem, jenž je náhodně vybrán. Aby aplikace prošla kontrolou, musí splňovat zásady této kontroly, tzv. *App Store Review Guidelines*. Guidelines jsou rozdělené do pěti kategorií: bezpečnost, výkonnost, business, design a právo. Jednotlivé body těchto kategorií jsou pečlivě kontrolovány, a pokud dojde

k náznaku porušení těchto zásad, vydání nové verze aplikace je okamžitě zamítnuto. Vývojář má následně šanci nesrovnalosti obhájit, nebo opravit.

Apple na svých stránkách uvádí (viz [41]), že schválí 60 % všech aplikací poslaných ke kontrole.

7.2.2 App Previews and Screenshots

Pro novou verzi je potřeba vytvořit video nebo obrázky, jež mají za úkol upoutat a představit aplikaci uživatelům (obr. 7.1). Screenshots jsou povinné a je potřeba nahrát obrázky zařízení s velikostí displeje 6.5" pro variantu s výřezem, tzv. notch, a 5.5" pro variantu bez výřezu.

Vytvořil jsem celkem 7 obrázků zasazených do rámečku telefonu s doprovodným textem. Obrázky provádějí návštěvníka aplikace na AppStore procesem načtení bodu od vyhledání na mapě přes načtení až po zobrazení žebříčku.

Obrázek 7.1: Appstore Screenshots

7.2.3 Samotné vydání aplikace

1. Prvním krokem k vydání aplikace je její nahrání na App Store Connect portál.

2. Vytvořit novou verzi v záložce *App Store* na tomto portálu.
3. Vybrat build, který chceme nahrát na obchod.
4. Nahrát připravená previews či screenshots.
5. Vyplnit přihlašovací údaje, kterými se tester dostane do aplikace.
6. Vyplnit release notes neboli poznámky s informacemi o novinkách ve vydávané verzi a další dodatečné údaje, jako jsou kontaktní údaje, popis aplikace a klíčová slova.
7. Zvolit mód vydání aplikace. K dispozici je automatické vydání ihned po schválení aplikace, automatické vydání po určitém časovém údaji nebo manuální vydání vývojářem.
8. Odeslat verzi ke schválení.
9. Po úspěšném schválení aplikace vydat aplikaci podle zvoleného módu.

7.3 Problémy při schvalování

Vydání první verze s registrací, obrazovkami pro načítání bodů a historií bodů proběhla hladce bez jediné výtky testera, jenž prováděl review.

V aplikaci chyběl check box pro GDPR, a proto jsem se rozhodl vydat následující den aktualizaci. Vydání aktualizace bylo zamítnuto a Apple požadoval zvýšení ohodnocení spodní věkové hranice pro stažení aplikace ze 4 let na 17 let (obr. D.10). Aplikace podle vyjádření testera obsahuje soutěž nebo loterii, a proto není vhodná pro určenou věkovou skupinu. „*Hodnocení aplikace 4+, jež jsem vybral, není konzistentní s obsahem aplikace. Dokud aplikace obsahuje soutěže, sázky nebo sázky s reálnými penězi, musíte vybrat „Yes“ pro Gambling a Contests v App Store Connect.*“

Společně se zástupci Life is Skill jsme Applu odpověděli, vysvětlili jsme všechny náležitosti a verzi jsem odeslal k překontrolování. Po týdnu nepřetržitých výměn zpráv Apple aplikaci zařadil do námi požadované věkové kategorie, ale aplikace byla znovu vrácena kvůli porušení pravidla č. 5.3.3. *„V aplikaci musí být uvedena oficiální pravidla pro loterie, soutěže a tomboly, které jasně uvádějí, že Apple není sponzorem ani jakýmkoliv způsobem zapojen do aktivity.“* Tuto informaci jsem do aplikace uvedl a aplikaci poslal znovu ke kontrole. Vydání aplikace se z průměrné doby schválení jeden den prodloužilo na celý týden, kdy jsme měli naplánované vydání verze s mapou. Apple ale i přes splnění pravidla *č. 5.3.3.* aplikaci znova odmítl (obr. D.11).

„Zjistili jsme, že aplikace během registrace po uživateli požaduje osobní informace, které nejsou přímo relevantní s hlavní funkcionalitou aplikace. Následující políčka jsou povinná, ale nezdají se relevantní k hlavní funkcionalitě aplikace: Mobil, PSC, Pohlavi, Mobil rodice“. Apple požaduje, aby vývojáři

pomocí aplikace neshromažďovali nepotřebné osobní údaje. Jedná se o porucení pravidla č. 5.1.1. „*Data Collection and Storage (iii) Data Minimization: Aplikace by měly vyžadovat pouze přístup k datům důležitým k základní funkčnosti aplikace a měly by shromažďovat a používat data, která jsou nezbytná pro splnění příslušného úkolu.*“ [22]

Aplikaci jsem poslal k opětovnému schválení s odpovědí „*(...) všechna políčka zahrnutá v registraci jsou relevantní se základní funkcionalitou aplikace, protože jsou důležitá k účasti v soutěži. Pohlaví je požadované kvůli obrázku uživatele a statistikám hry. Poštovní směrovací číslo je povinné, aby se účastníci mohli účasnit speciálních soutěží v místě bydliště. Tyto soutěže jsou více popsány v pravidlech. Telefonní číslo uživatele a rodiče jsou způsob, kterým kontaktujeme výherce hry a řešíme problémy spojené s načítáním bodů.*“ [22] Verze byla znovu zamítnuta kvůli povinnému poli *pohlaví*.

Obrázky uživatelů podle pohlaví byly v plánu, ale zatím nebyly implementovány. Do aplikace jsem přidal k původnímu univerzálnímu obrázku chlapce i obrázek dívky a aplikaci opět odeslal ke schválení. K odpovědi jsem přiložil snímek dvou obrazovek s odlišnými profilovými obrázky. Apple tuto verzi přijal a po více jak 14 dnech jsme vydali aktualizaci.

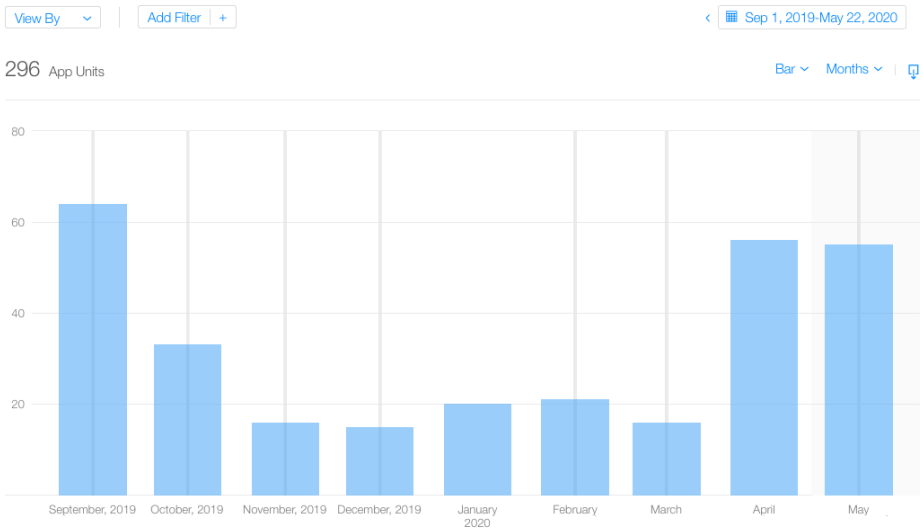
Vydání aplikace na AppStore není tak jednoduché, jak se zdálo po vydání první verze. Apple velmi pečlivě kontroluje všechna pravidla, a jelikož aplikaci netestuje strojově, tak přes testera nemusí projít funkcionality, které byly již dříve schváleny. Jak jsem již zmiňoval, v první verzi úplně chyběla zmínka o GDPR a aplikace prošla kontrolou, i když je GDPR požadováno v bodě č. 5.1.1. Data Collection and Storage (ii) Permission: *„Ujistěte se, že vaše purpose strings jasně a úplně popisují použití dat. Aplikace, které shromažďují údaje pro legitimní zájem bez souhlasu a spoléhají se na podmínky obecného nařízení Evropské unie o ochraně údajů (GDPR) nebo podobného statutu, musí splňovat všechny podmínky tohoto zákona“*[22]

7.4 Statistiky

Nejen Firebase analytics zmiňované v 6.4, ale také App Store Connect portál, poskytuje řadu statistik o aplikaci. Statistiky zmiňované v této sekci jsou z období od 1. 9. 2019 do 22. 5. 2020.

Počet stažení

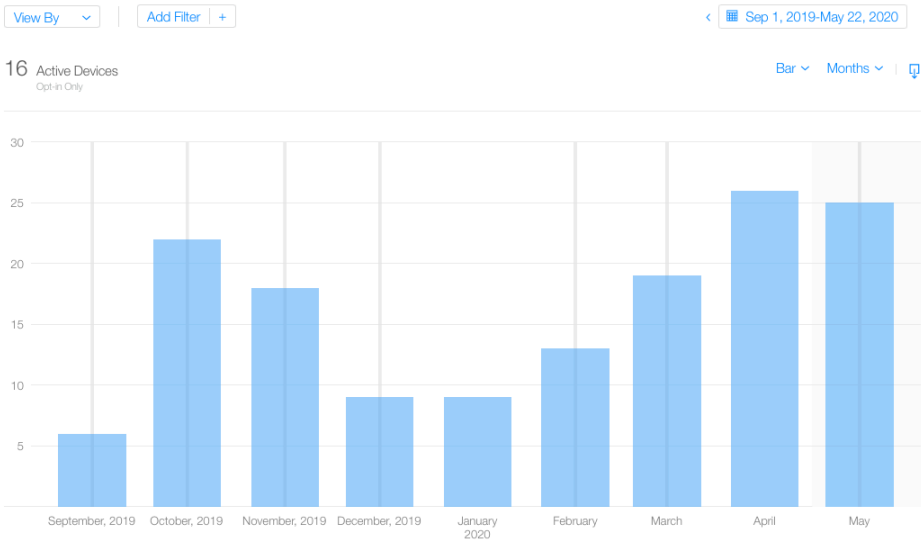
Aplikace má 296 unikátních stažení (obr. 7.2). Největší počet stažení byl zaznamenán v září, kdy byla zpřístupněna první beta verze s možností načítání a zobrazení přehledu načtených bodů. Zájem o aplikaci ke konci roku klesal a znovu se obnovil na počátku nového roku, kdy měl počet stažení rostoucí tendenci a v dubnu i květnu si aplikaci dohromady stáhlo více jak 100 uživatelů.



Obrázek 7.2: Počet stažení [42]

Aktivita uživatelů

Největší průměrná denní aktivita uživatelů byla zaznamenána v dubnu (obr. 7.3). V tomto měsíci byl denní průměr aktivních uživatelů 26. Podobně jako počet stažení i aktivita uživatelů ke konci roku klesala. V únoru a březnu začala aktivita uživatelů postupně růst a svého vrcholu nabyla v dubnu a květnu.



Obrázek 7.3: Aktivita uživatelů [42]

7.5 Budoucnost

Cílem práce bylo vytvoření první verze aplikace pro hru Life is Skill, jež obsahuje funkcionality s nejvyšší prioritou. U aplikace se počítá s dalším rozšířením.

První část plánovaných rozšíření se týká soutěže. Již v dodaném grafickém návrhu byl nástin několika nových funkcionalit, mezi které patří přehled událostí a zpráv pro hráče, notifikace a možnost vytvoření vlastního avatara z částí obličeje a doplňků, které by se hráčům zpřístupňovaly za získané body. První verze aplikace podporuje pouze jednu soutěž. V dalším rozšíření se počítá s podporou více soutěží pro jednoho uživatele. Mělo by se jednat o soutěže pro nové hráče, soutěže kategorií bodů a jiné.

Aplikace momentálně umožňuje načítání bodů, pouze pokud je uživatel připojen k internetu. S potřebným připojením k internetu může mít uživatel v lesích či uvnitř historických budov problém, a načítání bodů by proto mělo být umožněno i v offline režimu a po připojení k internetu by mělo dojít k synchronizaci. Aplikace by také měla umět v offline módu předběžně zvalidovat bod, tedy zobrazit uživateli, jestli je načtení bodu platné. V aplikaci mi chybí přehled chybně načtených bodů, proto navrhuji přidat novou obrazovku, na které by děti viděly alespoň posledních pár nevalidních načtení.

Platforma iOS nabízí také další vychytávky, které by zpříjemnily používání aplikace. Navrhuji přidání podpory *FaceID* a *TouchID* pro přihlašování a vytvoření *Today widgetu* s denní aktivitou uživatele. Přidání podpory pro iPad by rozšířilo portfolio zařízení, s jejichž pomocí je možné se účastnit hry.

Zařízení od modelu iPhoneXS a iPhoneXR umožňují čtení NFC na pozadí. Toho by bylo možné využít a po detekci Life is Skill čipu na pozadí otevřít aplikaci.

Mobilní aplikace je klíčovým prvkem celého systému. Bez ní není umožněno načítání čipů. Proto doporučuji v budoucnu nasadit i další metody testování. Šlo například o snapshot testy, které vytvoří snímek obrazovky naplněné daty a při každém spuštění porovnávají referenci s aktuálním snímkem, nebo také UI testy, s jejichž pomocí lze kontrolovat zobrazení prvků na obrazovkách nebo jestli jsou všechny obrazovky dostupné. Všechny testy by měly být spouštěny automaticky. Proto navrhuji vytvoření CI pipeline, která před každým pull requestem automaticky spustí build aplikace a poté všechny testy. Výsledky těchto testů by mohly být odesílány do e-mailu nebo komunikačního kanálu Slack.

Závěr

V rámci bakalářské práce jsem vytvořil plně funkční iOS aplikaci pro soutěž Life is Skill. Aplikaci jsem také nasadil do obchodu App Store a nyní je již plně využívána soutěžícími.

Aplikace umožňuje uživateli vytvoření nového účtu, načítání bodů pomocí NFC a QR technologií, zobrazení mapy s aktuální polohou a lokacemi dostupných volnočasových aktivit, profilu s přehledem historie načtených bodů. Zobrazuje také statistiky soutěže společně s výhrami, kontaktování organizátorů soutěže rovnou z aplikace a zobrazení pravidel soutěže.

Při tvorbě této práce jsem postupoval podle vytyčených cílů. Pro správný návrh aplikace bylo důležité se seznámit se hrou Life is Skill, jejími pravidly a již existujícími softwarovými systémy hry, tj. Android aplikací, webovou aplikací a REST API. Důkladná analýza mi pomohla při výběru architektury a technologií, jež jsem použil k realizaci. Zvolená MVVM architektura společně s funkcionálně reaktivním programováním se ukázaly jako správně zvolená kombinace, jelikož mi umožnila snadnou rozšiřitelnost jednotlivých verzí aplikace o nové funkcionality, oddělit odpovědnosti jednotlivých komponent a zvolit vhodnou strategii pro testování business logiky pomocí unit testů. Po dokončení vývoje a otestování každé z dílčích verzí bylo potřeba provést proces nasazení aplikace do obchodu App Store. Během tohoto procesu jsem se potýkal s řadou komplikací a ukázalo se, že nasazení aplikace není vždy tak jednoduché, jak by se mohlo zdát.

Celý proces vývoje iOS aplikace byl pro mne přínosem a natolik mne nadchl, že jsme se s neziskovou organizací domluvili na pokračující spolupráci při údržbě aplikace a vývoji dalších funkcionalit.

V budoucnu by bylo možné aplikaci rozšířit o funkcionality zmiňované v sekci 7.5, mezi něž patří například načítání bodů bez připojení k internetu či přidání podpory pro iPad zařízení. Zvolený proces verzování zabíral mnoho času a spíše mě zdržoval, jelikož jsem si pull requesty kontroloval sám. Tento proces bych v budoucnu vylepšil vytvořením tzv. CI pipeline s automatizovaným spouštěním testů a nasazením k testování po každém pull requestu.

Bibliografie

1. LIFE IS SKILL. *O projektu* [online] [cit. 2020-02-21]. Dostupné z: `http://www.lifeisskill.cz/?task=about`.
2. LIFE IS SKILL. *Úvod* [online]. 2019 [cit. 2020-05-22]. Dostupné z: `http://www.lifeisskill.cz/`.
3. LIFE IS SKILL. *Všeobecná pravidla soutěží LiS* [online] [cit. 2020-02-21]. Dostupné z: `http://muj.lifeisskill.cz/`.
4. BROUGHALL, Nick. iOS: What you need to know about Apple's mobile OS. *Finder* [online]. 2020 [cit. 2020-02-21]. Dostupné z: `https://www.finder.com/ios-operating-system`.
5. DIGITAL TECH AKSHAY. What Is The iOS Operating System? *Medium* [online]. 2019 [cit. 2020-02-21]. Dostupné z: `https://medium.com/@digitaltechakshay/what-is-the-ios-operating-system-b19c5d19f5bc`.
6. List of iOS devices. *Wikipedia.com* [online] [cit. 2020-02-22]. Dostupné z: `https://en.wikipedia.org/wiki/List_of_iOS_devices`.
7. DOMKAŘ, Bc. Petr. *Multiplatformní mobilní aplikace: Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí diplomové práce RNDr. Vít Rusňák, Ph.D.* [online]. Brno, 2018 [cit. 2020-02-21]. Dostupné z: `https://is.muni.cz/th/u09n4/multiplatformni-mobilni-aplikace-text.pdf`.
8. EISENMAN, Bonnie. What Is React Native? - Learning React Native. *O'Reilly* [online] [cit. 2020-02-21]. Dostupné z: `https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html`.
9. FILIP, Šmíd. Vyvíjíme mobilní aplikaci ve Flutteru. *eMan blog* [online]. 2019, č. 1/3 [cit. 2020-02-21]. Dostupné z: `https://www.eman.cz/blog/vyvijime-mobilni-aplikaci-ve-flutteru-1-3/`.

10. RAKSHIT SORAL. React Native vs Swift – A Side-by-Side Comparison for iOS Application Development. *SIMFORM* [online]. 2019 [cit. 2020-04-02]. Dostupné z: https://www.simform.com/react-native-vs-swift-ios-application-development/.

11. TUTORIALSPOINT. *Objective-C Tutorial* [online] [cit. 2020-02-21]. Dostupné z: https://www.tutorialspoint.com/objective%7B%5C_%7Dc/index.htm.

12. JETBRAINS. *Swift & Objective-C 2019 - The state of Developer Ecosystem in 2019 Infographic* [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2019/swift-objc/.

13. APPLE INC. *About Swift — The Swift Programming Language* [online] [cit. 2020-02-22]. Dostupné z: https://docs.swift.org/swift-book/.

14. SHEPHERD, Adam. What is the Swift programming language, and why should I learn it? *IT Pro* [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.itpro.co.uk/development/34417/what-is-the-swift-programming-language-and-why-should-i-learn-it.

15. STRUZINKI, Mark. ARC and Memory Management in Swift. *raywenderlich.com* [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.raywenderlich.com/966538-arc-and-memory-management-in-swift.

16. HSKYTECH. Xcode Vs. AppCode – How To Choose The IDE Right Way? *SkyTechGeek* [online]. 2018 [cit. 2020-04-09]. Dostupné z: https://skytechgeek.com/2018/09/xcode-vs-appcode-choose-ide-right-way/.

17. UNITAG. *What is a QR Code* [online] [cit. 2020-02-22]. Dostupné z: https://www.unitag.io/qrcode/what-is-a-qrcode.

18. APPLE INC. *Apple Developer Documentation*. qr - AVMetadataObject [online] [cit. 2020-02-22]. Dostupné z: https://developer.apple.com/documentation/avfoundation/avmetadataobject/objecttype/1618819-qr.

19. ALZA.CZ A.S. Co je NFC? [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.alza.cz/co-je-nfc.

20. APPLE INC. *Apple Developer Documentation*. Near Field Communication [online] [cit. 2020-02-21]. Dostupné z: https://apple.co/2y4K22i.

21. APPLE INC. *Apple Developer Documentation*. Adding Support for Background Tag Reading [online] [cit. 2020-02-22]. Dostupné z: https://developer.apple.com/documentation/corenfc/adding_support_for_background_tag_reading.

22. APPLE INC. *Apple Developer Documentation*. App Store Review Guidelines [online]. 2019 [cit. 2020-02-25]. Dostupné z: https://developer.apple.com/app-store/review/guidelines.

23. MARTIN MALÝ. REST: architektura pro webové API. *zdrojak.cz* [online]. 2009 [cit. 2020-04-09]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.

24. FRANTIŠEK KUČERA. Protokol HTTP. *zdrojak.cz* [online]. 2011 [cit. 2020-04-09]. Dostupné z: https://www.zdrojak.cz/clanky/protokol-http/.

25. WIEGERS, Karl; BEATTY, Joy. *Software Requirements* [online]. 3. vyd. Redmond, Washington: Microsoft Press, 2013 [cit. 2020-02-23]. ISBN 978-0-7356-7966-5. Dostupné z: https://bit.ly/2JR9KtM.

26. CONSULTING, PM. *Nefunkční požadavky - PM Consulting* [online] [cit. 2020-02-22]. Dostupné z: https://www.pmconsulting.cz/slovnikovy-pojem/nefunkcni-pozadavky/.

27. SINKEVICH, Vladimir. The essence of reactive programming in Java. *ScienceSoft* [online]. 2018 [cit. 2020-02-23]. Dostupné z: https://www.scnsoft.com/blog/java-reactive-programming.

28. GRAHAM, Hutton. *Programming in Haskell* [online]. NY, Cambridge University Press, 2007 [cit. 2020-02-23]. ISBN 978-0-521-87172-3.

29. MAREK, Rychlý. *Ústav informačních systémů. Fakulta informačních technologií, Vysoké učení technické v Brně.* Formální specifikace architektur informačních systémů [online] [cit. 2020-02-23]. Dostupné z: http://www.fit.vutbr.cz/~rychly/public/docs/formal-architectures/xhtml/formarch.xhtml.

30. HUDSON, Paul. *Swift Design Patterns* [online book]. 20181029. vyd. [cit. 2020-02-23]. 29.10.2018.

31. BOHDAN, Orlov. iOS Architecture Patterns - iOS App Development. *Medium* [online]. 2015 [cit. 2020-02-21]. Dostupné z: https://bit.ly/3e4QmHA.

32. MICROSOFT. *Microsoft Docs*. Advantages and disadvantages of M-V-VM [online]. 2006 [cit. 2020-02-23]. Dostupné z: https://docs.microsoft.com/en-us/archive/blogs/johngossman/advantages-and-disadvantages-of-m-v-vm.

33. FABRIZIO, Brancati. Swift Package Manager vs CocoaPods vs Carthage for All Platforms. *Codementor Blog* [online]. 2018 [cit. 2020-02-23]. Dostupné z: https://www.codementor.io/blog/swift-package-manager-5f85eqvygj.

34. FAIGL, Jan. *Úvod do verzovacích systémů* [online prezentace]. 2016 [cit. 2020-02-23]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a0b36pr2/lectures/lecture11-slides.pdf.

35. CHACON, Scott; STRAUB, Ben. *Pro Git* [online]. 2nd. Apress, 2014 [cit. 2020-02-24]. ISBN 978-1484200773. Dostupné z: https://git-scm.com/book/cs/v2.

36. APPLE INC. *Interface Builder Built-In*. Dostupné také z: https://developer.apple.com/xcode/interface-builder/. 2020-02-24.

37. MISHALI, Shai. SnapKit for iOS: Constraints in a Snap. *raywenderlich.com* [[online]]. 2019 [cit. 2020-03-19]. Dostupné z: https://www.raywenderlich.com/3225401-snapkit-for-ios-constraints-in-a-snap.

38. FORMÁNEK, David. *Nástroje pro statickou a dynamickou analýzu kódu se zaměřením na bezpečnostní chyby: Bakalářská práce. Masarykova univerzita, Fakulta informatiky.* [online]. Brno, 2014 [cit. 2020-03-29]. Dostupné z: https://is.muni.cz/th/pngje/bp.pdf.

39. OSHEROVE, R. *The Art of Unit Testing: with examples in C#* [online]. Manning Publications, 2013 [cit. 2020-02-25]. ISBN 9781617290893. Dostupné z: https://livebook.manning.com/book/the-art-of-unit-testing-second-edition/list-of-figures/.

40. MILÁČEK, Ondřej. Google Play vs. App Store. *SMARTmania.cz* [online]. 2017 [cit. 2020-02-25]. Dostupné z: https://smartmania.cz/google-play-vs-app-store-vyhody-aplikace/.

41. APPLE INC. *App Store principy a postupy* [online] [cit. 2020-03-15]. Dostupné z: https://www.apple.com/cz/ios/app-store/principles-practices/.

42. APPLE. *App Store Connect* [online] [cit. 2020-05-23]. Dostupné z: https://appstoreconnect.apple.com.

43. STATCOUNTER. *StatCounter Global Stats* [online]. 2020 [cit. 2020-02-19]. Dostupné z: https://www.gs.statcounter.com.

44. FERDINI, Matteo. *iOS Storyboards in Xcode: The Ultimate Guide* [online] [cit. 2020-04-10]. Dostupné z: https://matteomanferdini.com/ios-storyboards-xcode/.

Příloha A

Seznam použitých zkratek

XML eXtensible Markup Language

iOS iPhone Operation System

NFC Near Field Communication

QR Quick Response

JSON JavaScript Object Notation

DSL Domain-Specific Language

UI User Interface

CI Continuous Integration

API Application Programming Interface

REST Representational State Transfer

MVC Model View Controller

MVP Model View Presenter

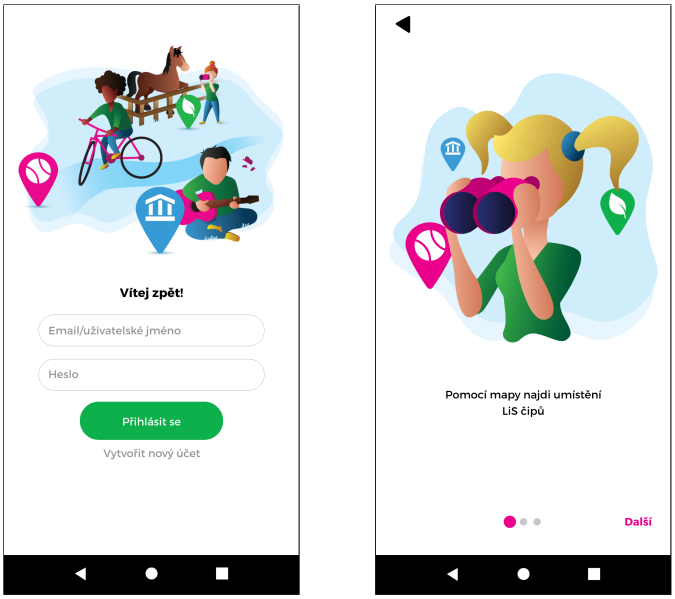
MVVM Model View ViewModel

IDE Integrated Development Environment

LiS Life is Skill

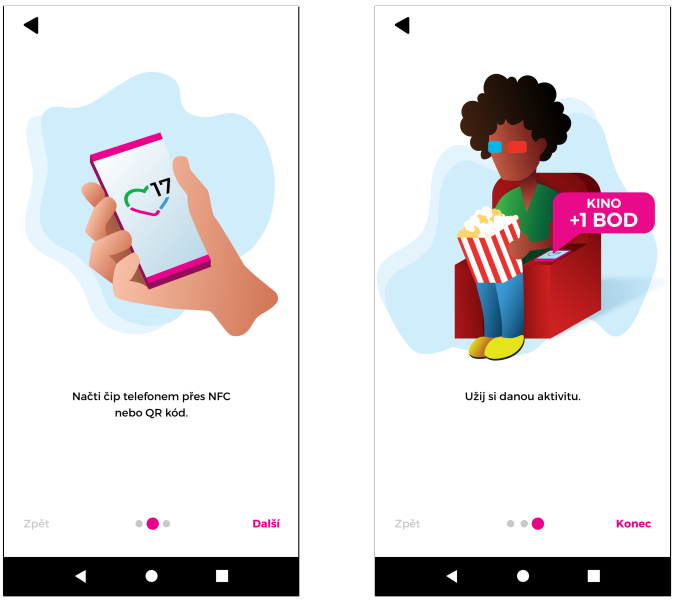
Příloha B

Grafický návrh mobilní aplikace



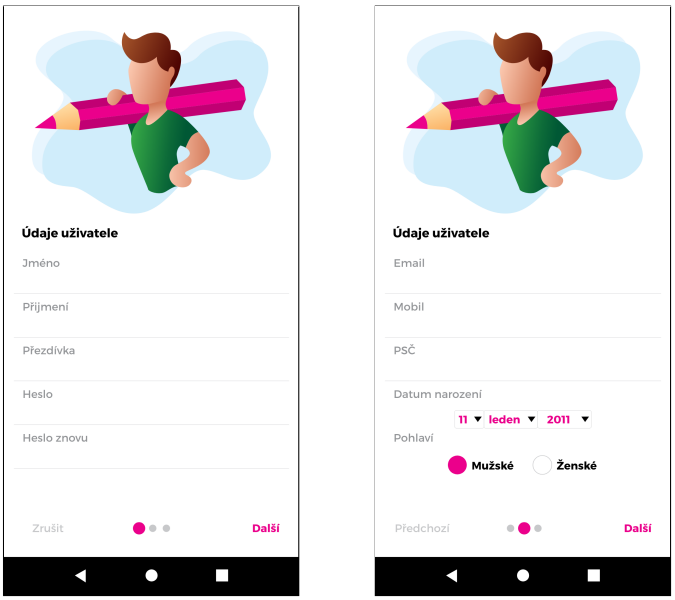
(a) Přihlašovací obrazovka (b) Nápověda strana č. 1

Obrázek B.1: Přihlašování a nápověda



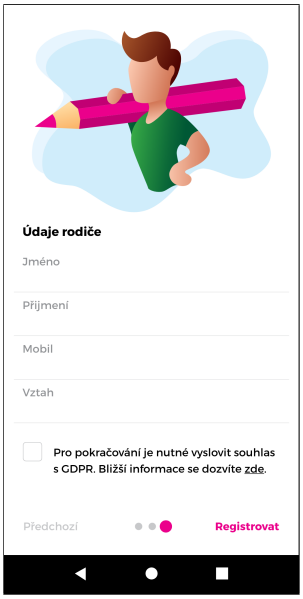
(a) Strana č.2 (b) Strana č.3

Obrázek B.2: Nápověda

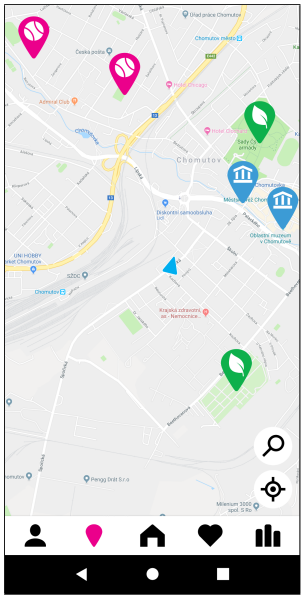


(a) Údaje o účtu (b) Osobní údaje hráče

Obrázek B.3: Registrace

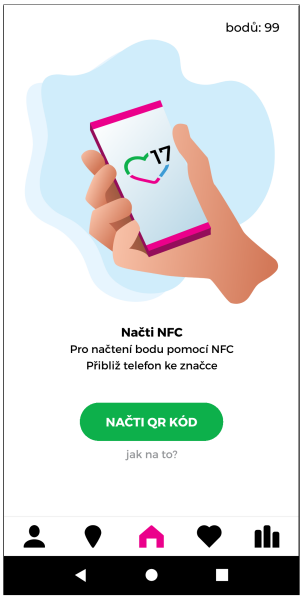


(a) Osobní údaje rodiče

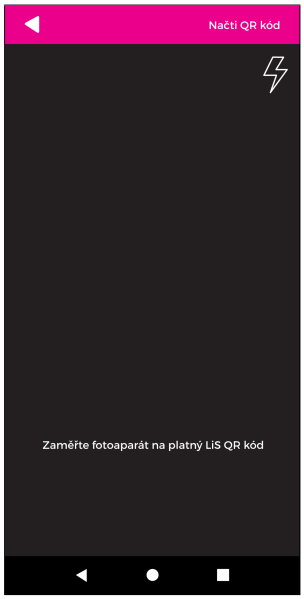


(b) Mapa

Obrázek B.4: Registrace rodiče a Mapa

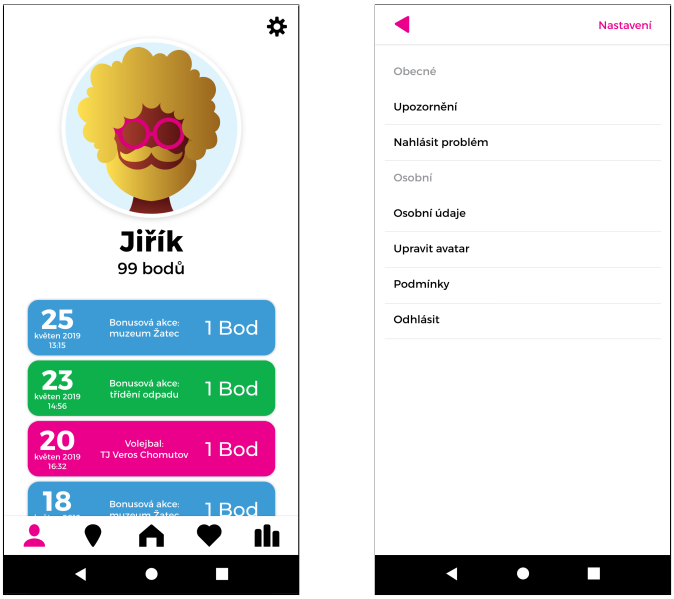


(a) NFC i QR



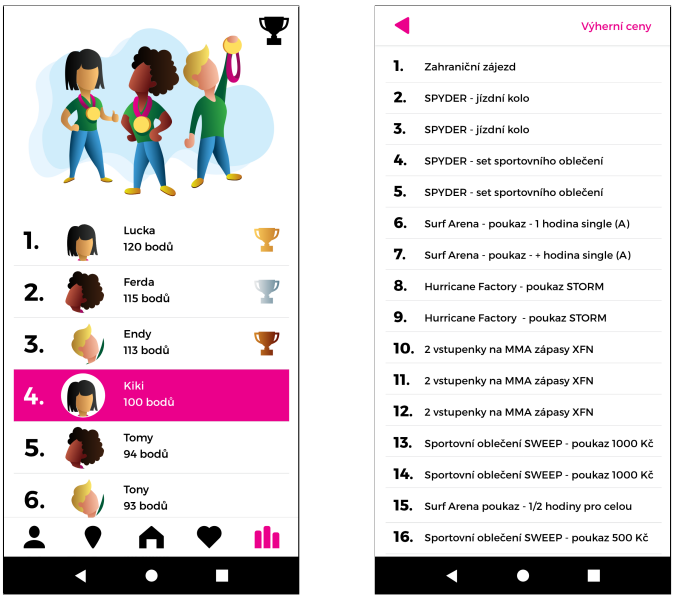
(b) Pouze QR

Obrázek B.5: Skenování bodů



(a) Přehled bodů (b) Nastavení

Obrázek B.6: Přehled bodů a nastavení

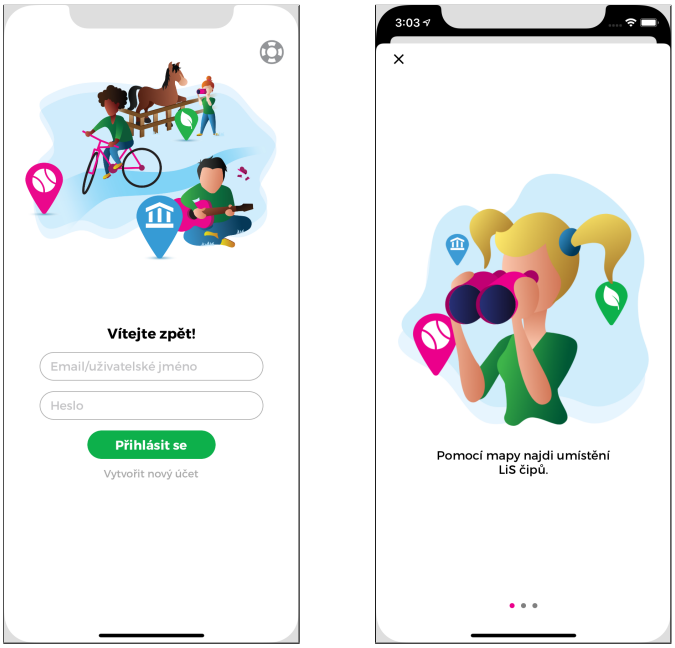


(a) Žebříček soutěže (b) Výhry

Obrázek B.7: Žebříček soutěže a výhry

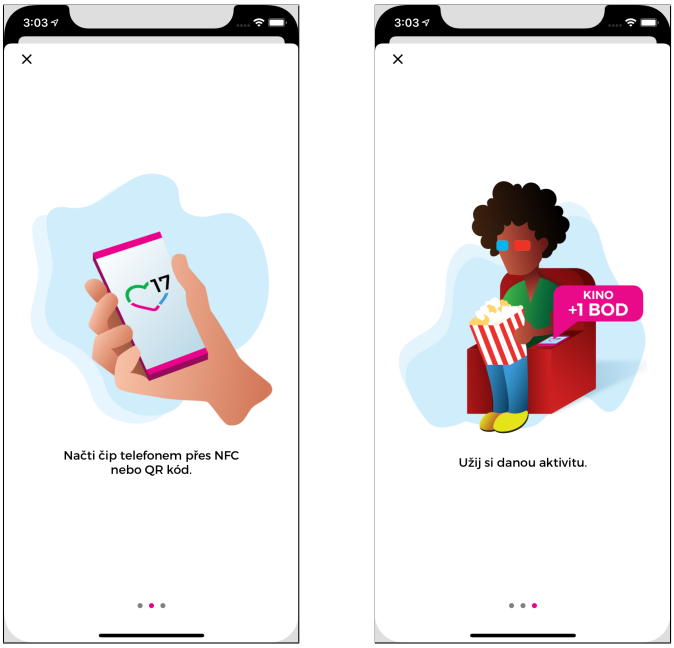
Příloha C

Finální podoba aplikace



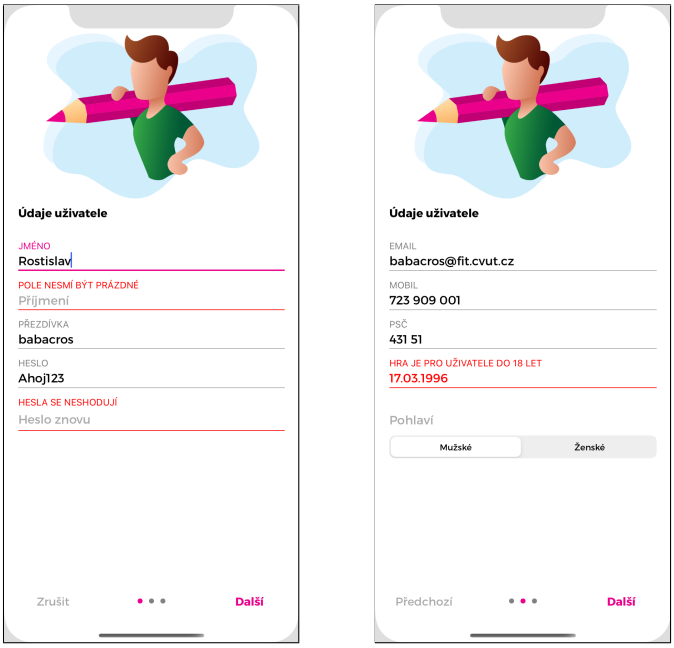
(a) Přihlašovací obrazovka (b) Nápověda strana č.1

Obrázek C.1: Přihlašování a nápověda



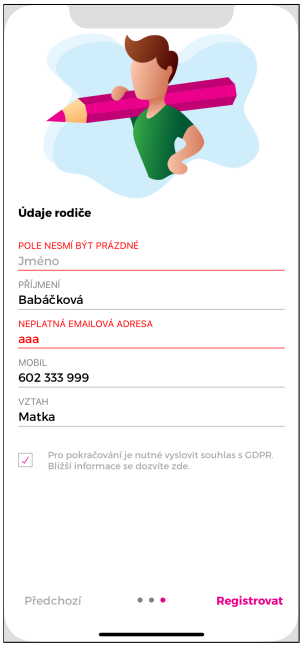
(a) Strana č.2 (b) Strana č.3

Obrázek C.2: Nápověda

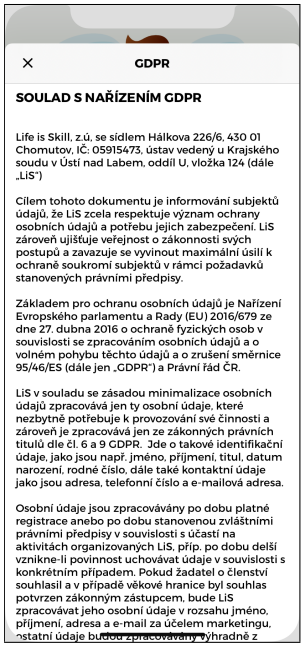


(a) Údaje o účtu (b) Osobní údaje hráče

Obrázek C.3: Registrace

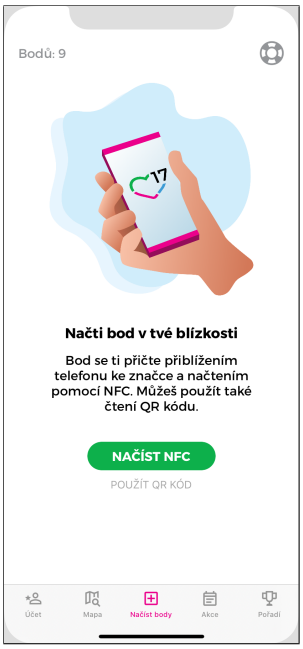


(a) Osobní údaje rodiče



(b) GDPR

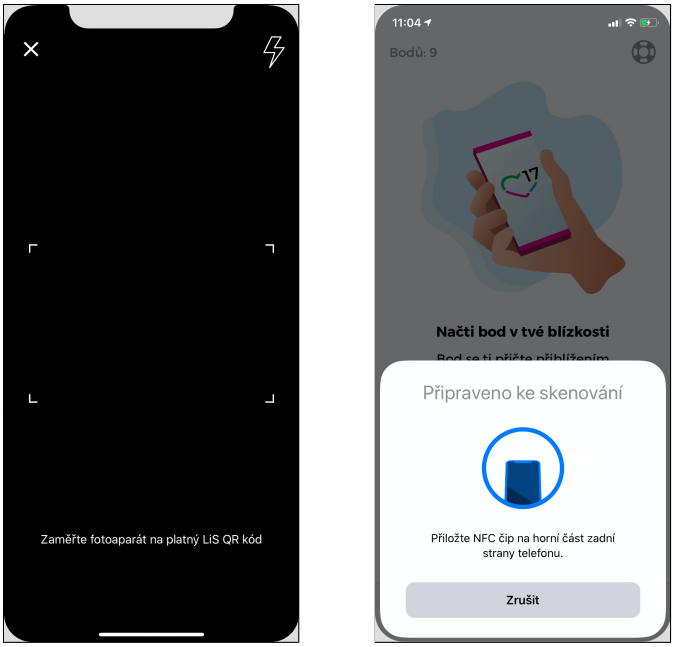
Obrázek C.4: Registrace rodiče a GDPR



(a) NFC i QR

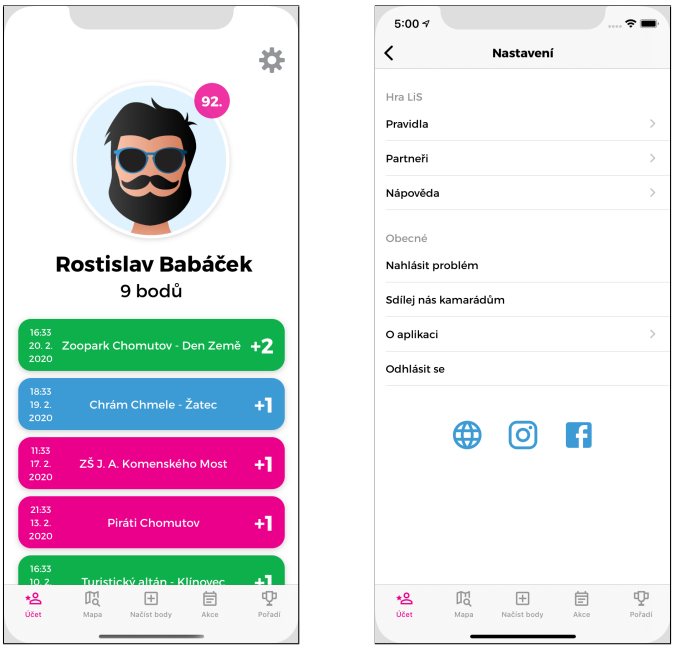
(b) Pouze QR

Obrázek C.5: Skenování bodů



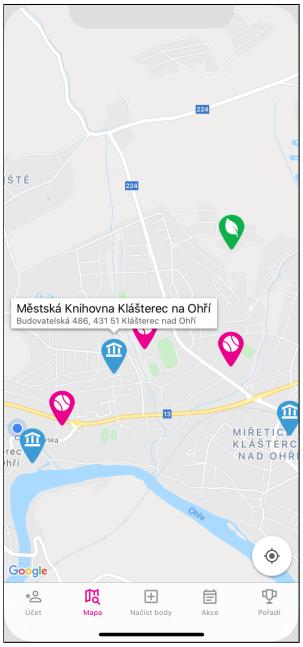
(a) Skenování QR (b) Skenování NFC

Obrázek C.6: Skenování QR a NFC

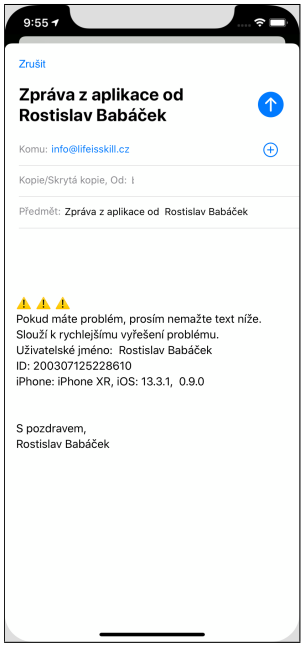


(a) Přehled bodů (b) Nastavení

Obrázek C.7: Přehled bodů a nastavení



(a) Mapa



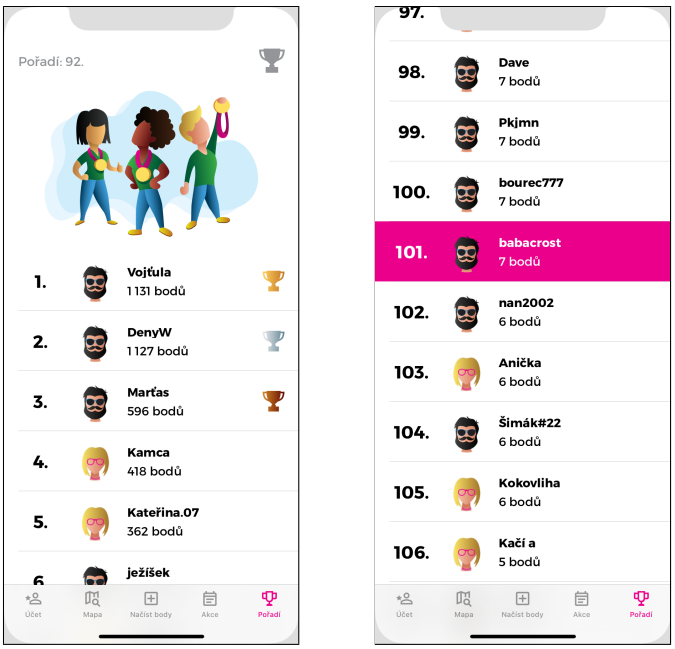
(b) Odeslání chyby

Obrázek C.8: Mapa a odeslání chyby

(a) Pravidla soutěže

(b) Výhry

Obrázek C.9: Pravidla soutěže a výhry

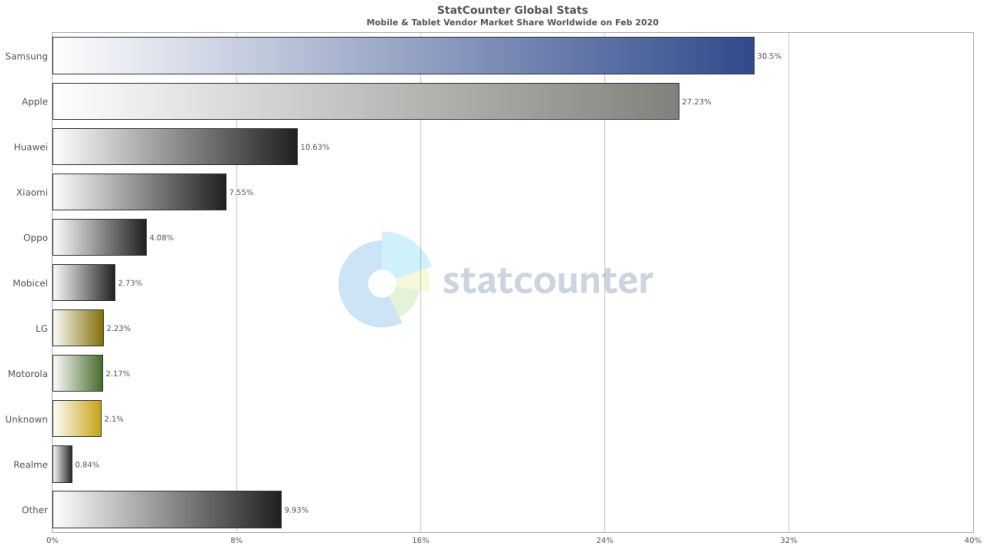


(a) Žebříček (b) Pozice hráče

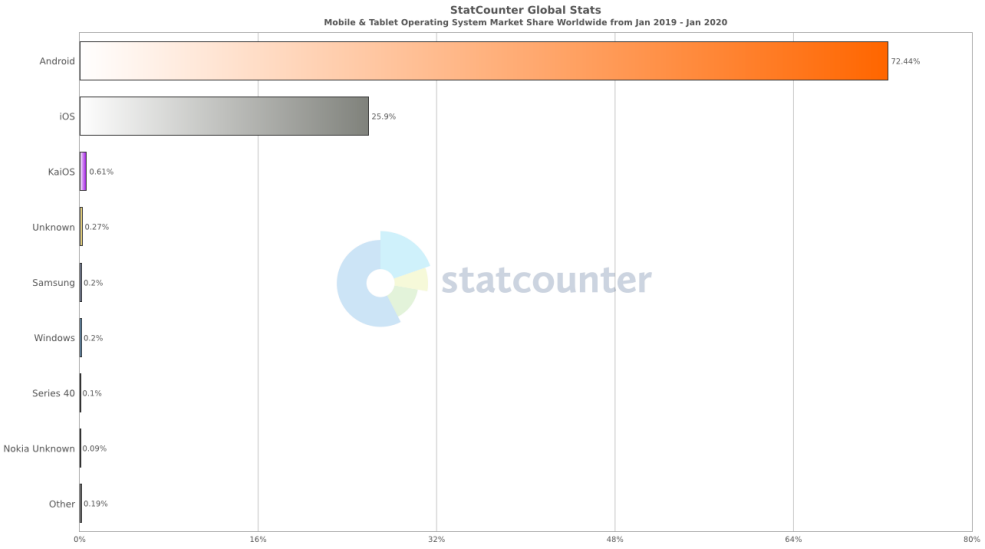
Obrázek C.10: Žebříček soutěže

Příloha D

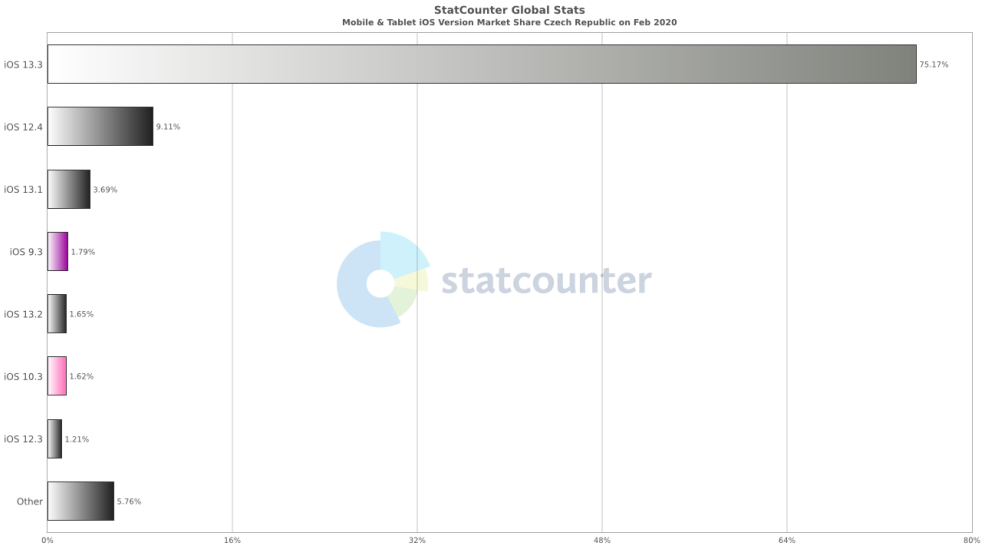
Obrázky



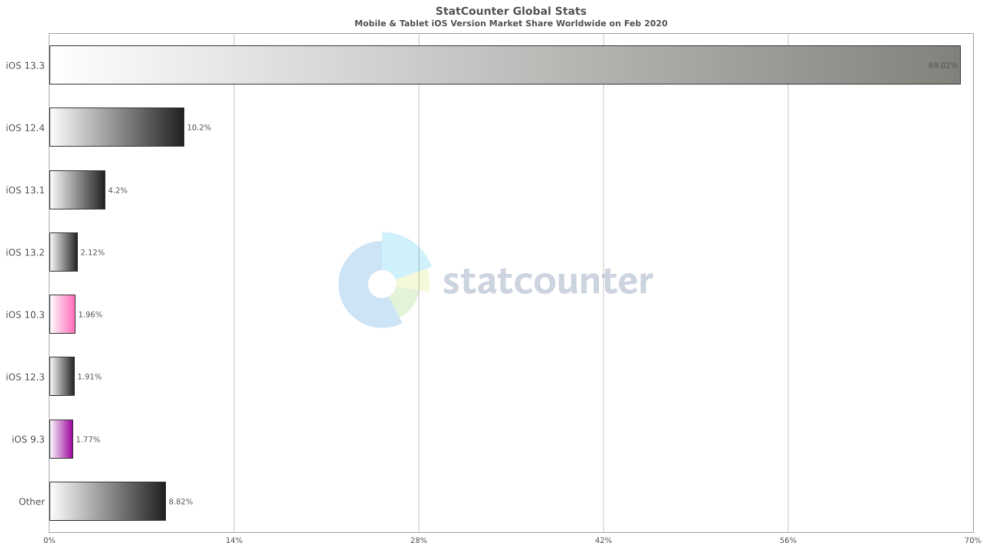
Obrázek D.1: Porovnání prodejů mobilních telefonů [43]



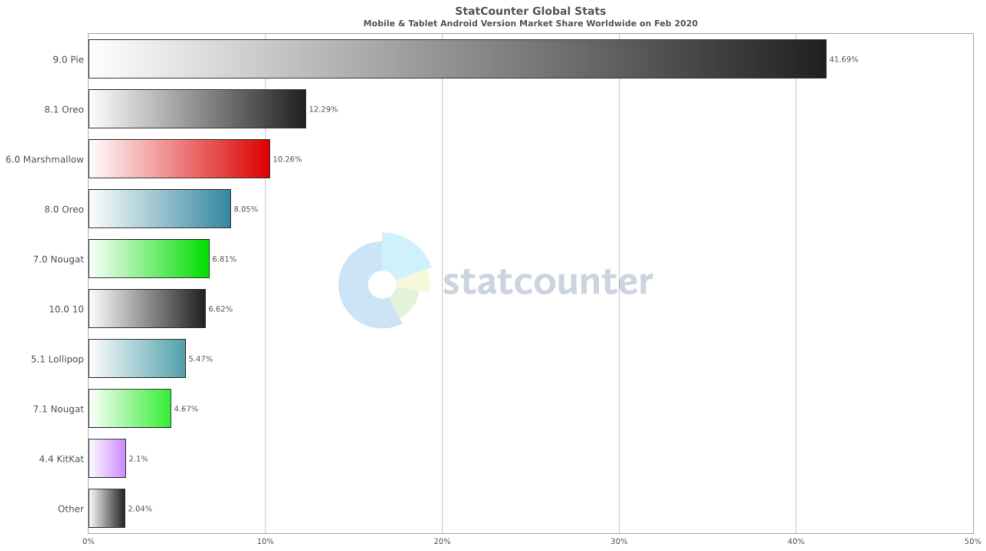
Obrázek D.2: Porovnání popularity operačních systémů pro mobilní telefony [43]



Obrázek D.3: iOS verze pro Český trh [43]

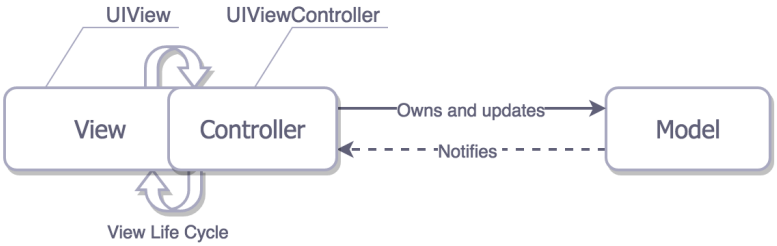


(a) iOS verze



(b) Android verze

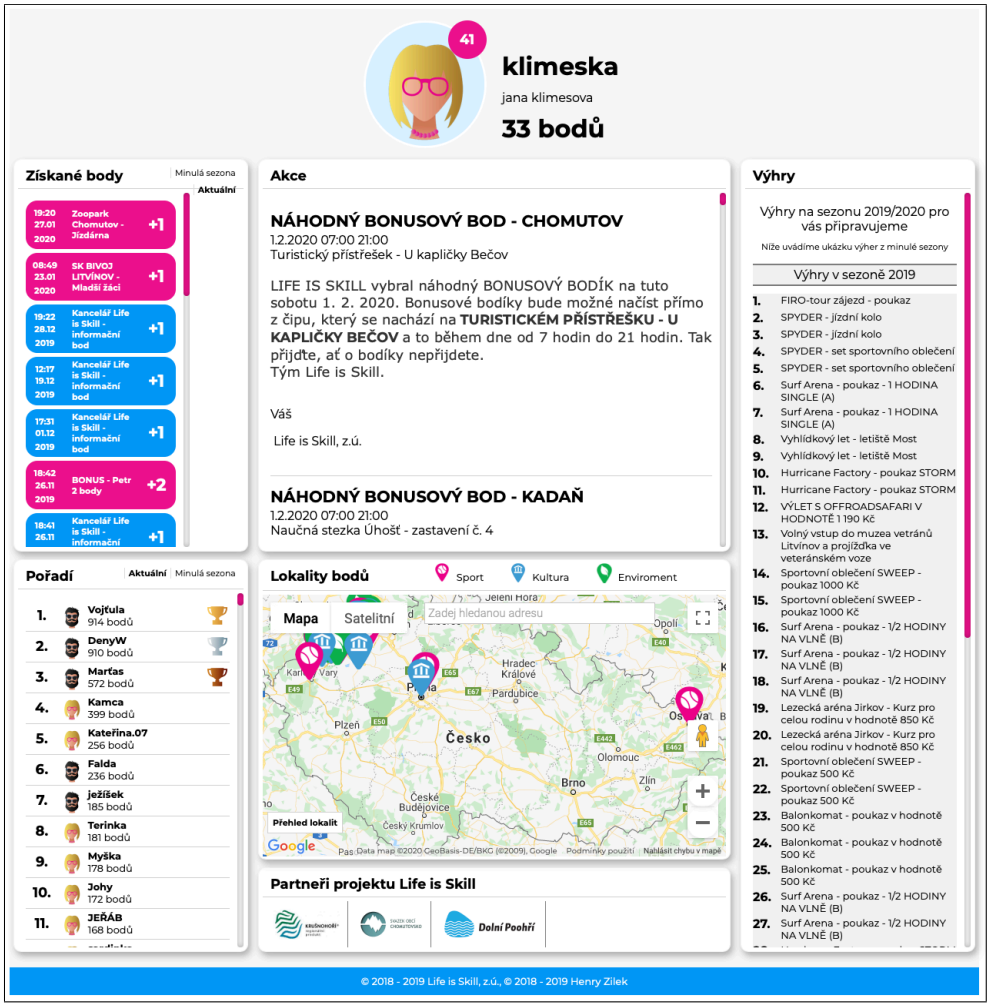
Obrázek D.4: Porovnání adaptace na nové verze mezi iOS a Android [43]



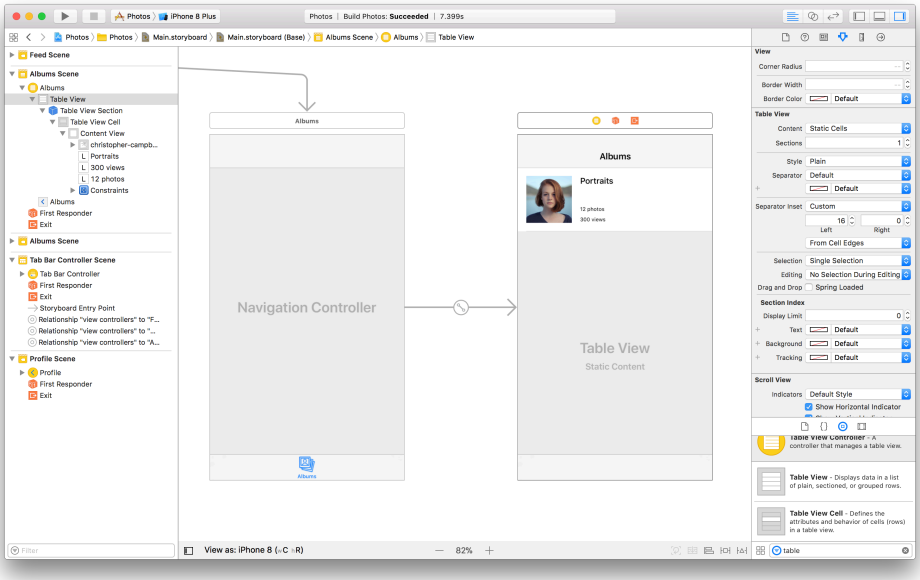
Obrázek D.5: Schéma MVC - masivní Controller [31]

```
protocol PointsListViewModeling {
    var pointsHistory: MutableProperty<[PointListPoint]> { get }

    var getPointsList: Action<Void, [PointListPoint], RequestError> { get }
    var refreshPointsList: Action<Void, [PointListPoint], RequestError> { get }

    func cleanPointsHistory()
}

final class PointsListViewModel: BaseViewModel, PointsListViewModeling {
    let getPointsList: Action<Void, [PointListPoint], RequestError>
    let refreshPointsList: Action<Void, [PointListPoint], RequestError>
    let pointsHistory: MutableProperty<[PointListPoint]>
    let outboxHidden: Property<Int>

    private func setupBindings() {

    }

    func cleanPointsHistory() {

    }
}
```

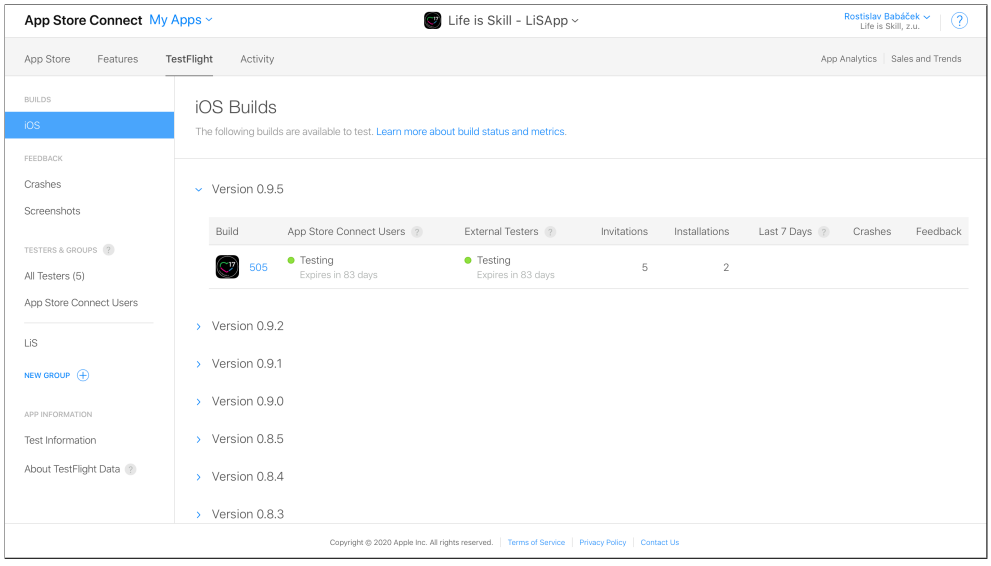
Obrázek D.6: Protocol oriented programming



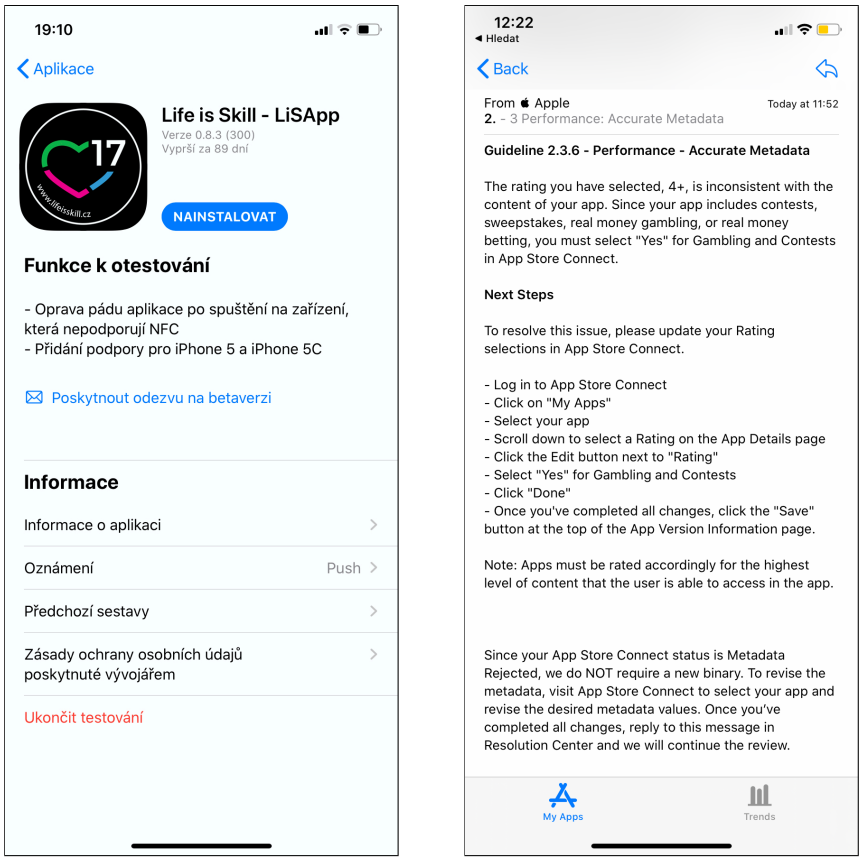
Obrázek D.7: Life is Skill webová aplikace [3]



Obrázek D.8: Xcode Interface Builder [44]

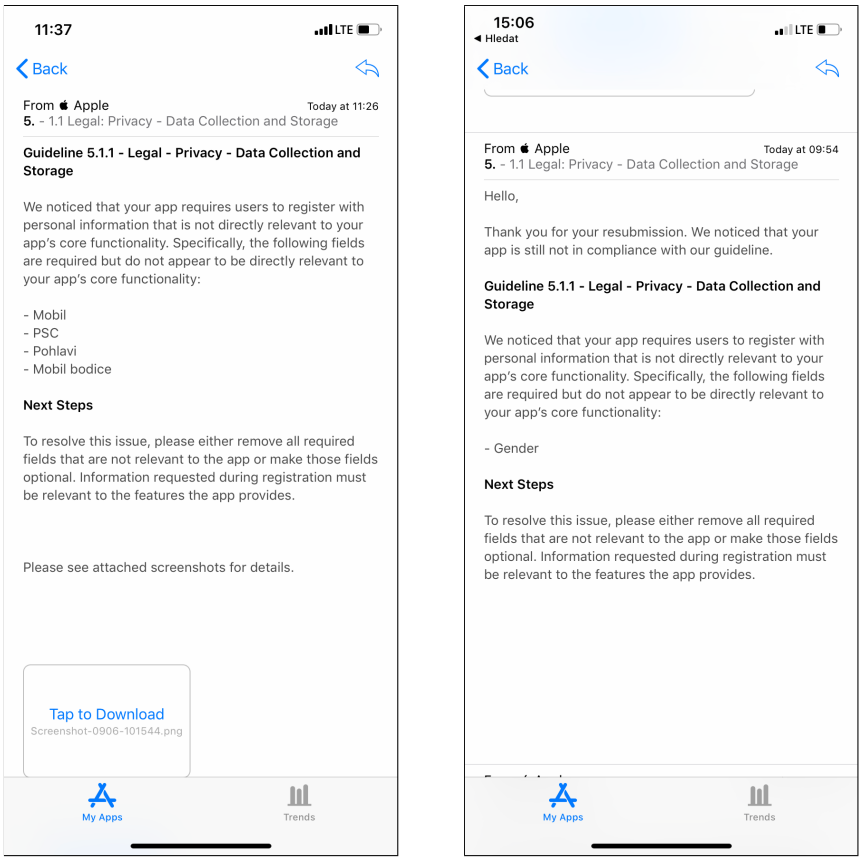


Obrázek D.9: Webové TestFlight prostředí [42]



(a) TestFlight aplikace (b) Zamítnutí věkové hranice

Obrázek D.10: TestFlight aplikace a zamítnutí věkové hranice



(a) (b)

Obrázek D.11: Zamítnutí vydání aktualizace aplikace

Příloha E

Obsah přiložené paměťové karty

- `readme.txt` ..... stručný popis obsahu paměťové karty
- `src`
  - `readme.txt` ..... instalační příručka
  - `lifeisskill` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF