VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MOTIVAČNÍ MOBILNÍ HRA PRO ELEKTRONICKÉ ZDRAVOTNICTVÍ

MOTIVATIONAL CROSS PLATFORM MOBILE APPLICATION FOR E-HEALTH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVEL POHNER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. OLENA PASTUSHENKO**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/20894/2017/xpohne01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Pohner Pavel**
Obor: Informační technologie
Téma: **Motivační mobilní hra pro elektronické zdravotnictví**
Motivational Cross Platform Mobile Application for e-Health
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s nástroji pro vývoj multiplatformních (Android / iOS / web) aplikací v jazyce JavaScript.
2. Prostudujte základní principy gamifikace a metodiku její aplikace.
3. Prozkoumejte moderní nástroje a metody elektronického zdravotnictví a mHealth.
4. Na základě analýzy a s využitím gamifikačních technik navrhněte prototyp aplikace, jejímž cílem je motivovat uživatele k zdravému chování.
5. Implementujte aplikaci na základě výzkumu (1-4).
6. Proveďte testování použitelnosti, vyhodnoťte výsledky a navrhněte další rozšíření.

Literatura:

- Norman, D.: Emotional Design: Why We Love (or Hate) Everyday Things. Basic Books, 2005.
- Bogost, I.: Persuasive Games: the Expressive Power of Videogames. MIT Press, 2010.
- Deterding, S.; Dixon, D.; Khaled, R.; Nacke, L.: From game design elements to gamefulness: defining gamification. Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek '11, 2011.
- Geuens, J.; D'haeseleer, I.; Geurts, L.; Vanden Abeele, V.: Lenses of Motivational Design for MHealth. Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts - CHI PLAY Companion '16, 2016.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Pastushenko Olena, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací cross-platform aplikace pro podporu a rozvoj pohybových aktivit s využitím gamifikace. Na počátku práce byla provedena analýza existujících aplikací, následovala analýza jednotlivých gamifikačních prvů a také nástrojů pro tvorbu cross-platform aplikací a uživatelských rozhraní. Na základě získaných informací byla navržena výsledná aplikace. Výsledkem této práce je cross-platform aplikace pro platformy Android, iOS a web, která je implementována v jazycích HTML, CSS a JavaScript s využitím frameworků Cordova pro nasazení aplikace a Ionic pro nativní vzhled aplikace. Výsledná aplikace za využití gamifikačních prvků motivuje k plnění pohybových úkolů a tréninků, čímž rozvíjí uživatelovi pohybové schopnosti a udržuje ho v kondici.

Abstract

This bachelor's thesis focuses on design and implementation of cross-platform application, which will support and extend user's physical activities and capabilities with usage of gamification. In the beginning of this work some of the existing applications were analyzed. Subsequently analysis of gamifiaction elements and tools for implementing cross-platform applications and user interfaces was performed. Based on acquired informations the application was designed. The result of this thesis is cross-platform application running on platforms Android, iOS and web, which is implemented in HTML, CSS and JavaScript with usage of Cordova framework for deploying application on target platform and Ionic framework for application's native appearance. The application with usage of gamifiaction elements motivates user to fulfill given physical tasks and trainings, which should result in improved physical abilities and better physical condition.

Klíčová slova

gamifikace, cross-platform aplikace, e-health, grafické uživatelské rozhraní, HTML, CSS, JavaScript, Cordova, Ionic

Keywords

gamifiaction, cross-platform application, e-health, graphical user interface, HTML, CSS, JavaScript, Cordova, Ionic

Citace

POHNER, Pavel. *Motivační mobilní hra pro elektronické zdravotnictví*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Olena Pastushenko

Motivační mobilní hra pro elektronické zdravotnictví

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Oleny Pastushenko. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Pohner
8. května 2018

Poděkování

Touto cestou děkuji vedoucí mé práce Ing. Oleně Pastushenko, za poskytnuté rady, ochotu a čas věnovaný konzultacím a konstruktivní kritice mé práce.

Obsah

Kapitola 1

Úvod

Hlavním cílem této práce je návrh a následná implementace motivační cross-platform aplikace pro podporu a rozvoj pohybových aktivit. Aplikace by měla uživatele motivovat k pohybu využitím prvků, typických pro počítačové hry (tzv. gamifikace). Právě tyto prvky by měly uživatele motivovat k tomu, aby se k aplikaci vracel a používal ji pravidelně. Aplikace bude umožňovat plnit různé úkoly, založené zejména na fyzické aktivitě a pohybu, monitorovat zlepšení a pokrok uživatele, odměňovat uživatele různými bodovými ohodnoceními a achievementy. To vše bude zabaleno v jednoduchém responzivním uživatelském prostředí a poběží jak na mobilních zařízeních (Android, iOS), tak i ve webových prohlížečích. Aplikace by se měla zásadně odlišovat od ostatních aplikací mírou použité gamifikace jakožto i faktem, že by měla fungovat na všech třech platformách. Aplikace bude obsahovat rozsáhlejší systém gamifikace, než je obvyklé u ostatních aplikací, čímž by měla docílit lepších výsledků.

Na začátku této práce bude vysvětlen pojem gamifikace a popsáno její využití v aplikaci. Následovat bude kapitola věnovaná existujícím řešením a konkurenčním aplikacím, s analýzou jejich výhod a nevýhod. V další kapitole budou stanoveny požadavky na navrhovanou aplikaci, dále zde bude popsáno, co by mělo splňovat uživatelské prostředí a aplikační logika. Následně proběhne analýza dostupných technologií a knihoven, které se hodí k implementaci navrhované aplikace. V další kapitole bude následovat již samotný návrh uživatelského prostředí, jakožto i výpis vybraných technologií a nástrojů, které budou použity pro vývoj a implementaci aplikace. Jedna z posledních kapitol bude věnována implementaci samotné, bude zde popsáno jak věci fungují, jak jsou jednotlivé funkce aplikační logiky navázány na prvky uživatelského prostředí a budou zde představeny zajímavé části řešení. Poslední kapitola bude věnována uživatelskému testování a závěrům z něho vyvozených, taktéž zde budou navrženy vylepšení a náměty pro budoucí vývoj aplikace.

Kapitola 2

Gamifikace

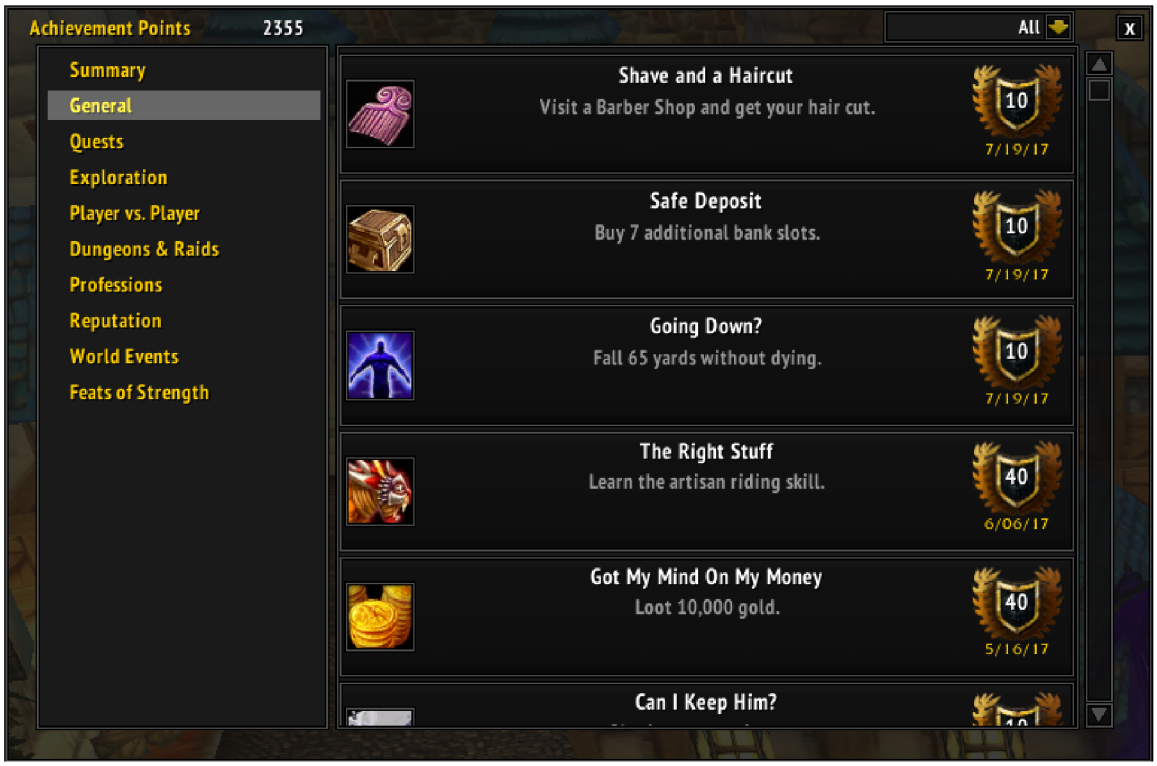
Gamifikace je využití prvků typických pro počítačové hry v aplikacích, které nejsou primárně určeny pro zábavní průmysl [6]. Obvykle se jedná o aplikaci vhodných gamifikačních prvků, jejichž cílem je zvýšit zájem uživatelů o výslednou aplikaci a také motivovat uživatele k jejímu častějšímu používání. Gamifikace tím pádem stojí na stejné filozofii jako počítačové hry - tedy využívá toho, že se uživatel chce bavit, chce být nějakým způsobem odměňován za to, čeho dosáhl nebo porovnávat svůj výkon s ostatními uživateli. Z toho vyplývá, že hlavní myšlenkou gamifikace je poskytování odměny za vykonávání různých aktivit podobně, jako to funguje v počítačových hrách. Zde se hodí poznamenat, že je někdy nemožné rozpoznat, zda je daná aplikace „gamifikována“, nebo je již hrou v pravém slova smyslu [6].

2.1 Použití gamifikace v mobilních aplikacích

Drtivá většina testovaných aplikací v určité formě využívá gamifikaci. Nejčastěji se jedná o nějakou formu získávání zkušeností a zvyšování úrovně uživatelského profilu [15]. Mezi často využívané gamifikační prvky patří však i achievementy, různé progress bary, odznaky, žebříčky nejlepších a tak podobně. Další možností je implementace virtuální měny, kterou lze získat opět plněním různých aktivit v aplikaci a za kterou si uživatel může zakoupit různé kosmetické prvky. Jsou však i aplikace, které zachází dál, než k výše zmíněným prvkům (například *Zombies, RUN!* viz 3.1), některé dokonce využívají příběhu, ve kterém hraje uživatel ústřední roli a umocňují tak motivaci se k aplikaci vracet a posunout se v příběhu dále. Některé z výše zmíněných gamifikačích prvků budou v následujících podkapitolách blíže vysvětleny.

2.1.1 Achievementy, odznaky a trofeje

Tyto gamifikační prvky jsou založeny na skutečnosti, že uživatel obvykle rád objevuje a sbírá nové věci. Všechny tyto prvky spojuje systém jejich získávání, typicky je jich v aplikaci k dispozici celá řada a bývají odstupňovány podle obtížnosti. Každý achievement nese nějaké zpravidla vtipné nebo chytlavé jméno a úkol, který je třeba pro jeho získání splnit [11]. V kontextu navrhované aplikace si například můžeme představit achievement, který by se jmenoval „Zrození nového běžce“ a pro jeho získání by bylo nutné uběhnout 5 kilometrů. Důležitým faktorem je dát seznam všech získatelných achievementů k dispozici uživateli, to pak může zafungovat jako seznam dílčích cílů, které budou uživatele motivovat k dalšímu používání aplikace.



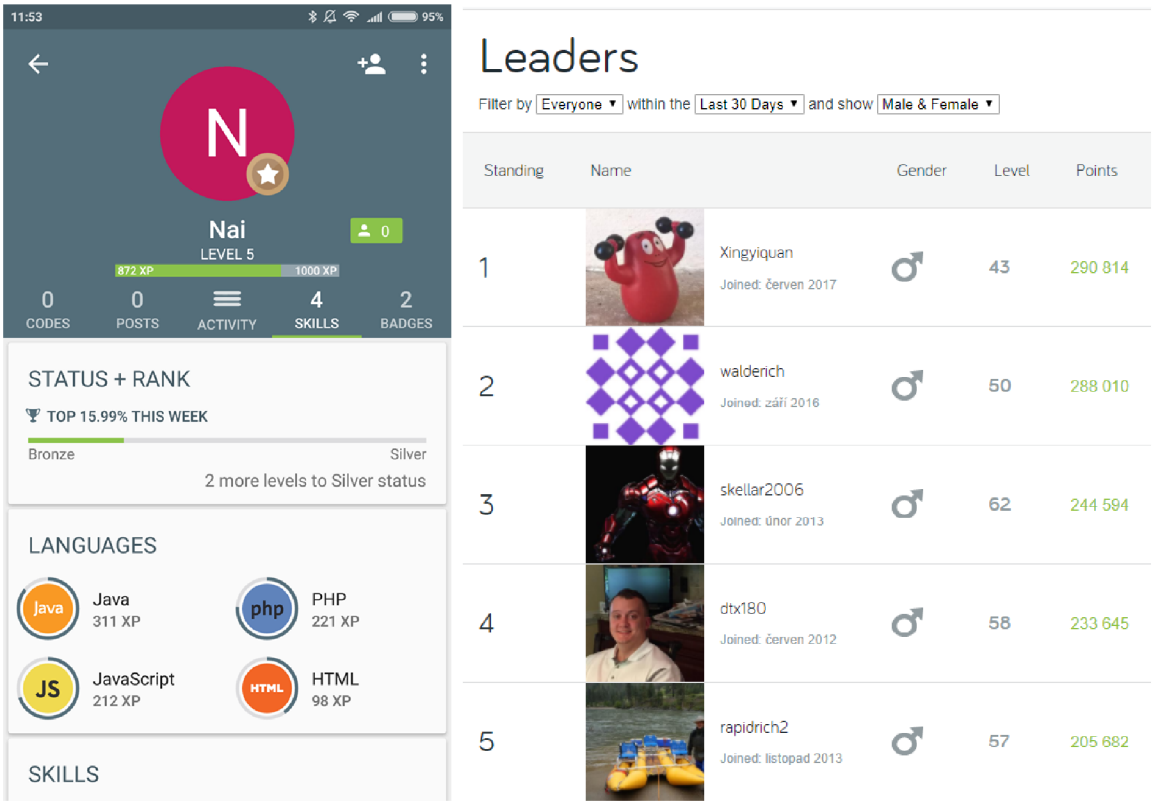
Obrázek 2.1: Příklad achievementů ze hry World of Warcraft

2.1.2 Úrovně, získávání zkušeností a úkoly

Úrovně a získávání zkušeností symbolizují postup v kontextu dané aplikace. Pro uživatele to většinou signalizuje, že se zdokonaluje a postupuje v aplikaci správným směrem. Získávání zkušenostních bodů bývá spojeno s všelijakými aktivitami v aplikaci, jednou z těchto aktivit může být například plnění úkolů. Úkoly hrají roli malých dílčích cílů, které by měly uživatele posouvat k jeho kýženému cíli. Úkoly by neměly být příliš snadné, ale ani příliš obtížné, mělo by se jednat o nástroj, který uživatele někam posune, ale zároveň ho neodradí svou obtížností. Důležité je, vhodně ukázat uživateli, co splněním tohoto úkolů získá. Zpravidla to bývají zkušeností body, ale odměnou mohou být i jiné věci, třeba zmiňovaná virtuální měna [15].

2.1.3 Žebříčky, týmy a klany

Žebříčky nejlepších staví na soutěživosti jednotlivých uživatelů, umožňují porovnávat dosažené výsledky s ostatními uživateli aplikace a zároveň motivují uživatele dostat se mezi nejlepší, aby v žebříčku figurovali na co možná nejlepším místě [15]. Na druhou stranu týmy, klany, gildy a jiné kolektivní celky spoléhají na uživatelovu chuť se socializovat, navazovat kontakty s ostatními hráči, bavit se s nimi a samozřejmě přátelsky soutěžit. Tyto prvky mají velmi pozitivní vliv na uživatelovu motivaci používat aplikaci pravidelně, setkávat se v ní se svými týmovými kolegy a přáteli a soutěžit s nimi, protože s využitím těchto prvků to už přestává být jen o uživatelovi jako jednotlivci, ale o celé komunitě uživatelů, kteří spolu soutěží.



Obrázek 2.2: Zleva: Příklad úrovní a zkušeností z mobilní aplikace SoloLearn, příklad žebříčku nejlepších z aplikace Fitocracy

2.2 Využití gamifikace ve vyvíjené aplikaci

V podkapitole 2.1 bylo zmíněno, že většina aplikací využívá gamifikační prvky jen na úrovni úrovní, zkušeností a nanejvýš nějaké míry socializace. Navrhovaná aplikace však zachází dále než k základním gamifikačním prvkům. Stěžejní myšlenka gamifikace v navrhované aplikaci stojí na využití prvků známých zejména z MMORPG (Massive Multiplayer Online Role-Playing Game) her, ve kterých si každý uživatel vytváří svou postavu (tedy virtuální obraz sebe sama ve světě oné hry). Stejný přístup využívá i naše aplikace. Uživatel si vytvoří postavu, která bude mít různé atributy a samozřejmě úroveň a plněním různých úkolů bude uživatel postupně atributy i úroveň postavy zvyšovat.

Atributy postavy by měly reflektovat reálné vlastnosti uživatele a míru jeho pokročilosti v těchto atributech. Úkoly budou společně s achievementy plnit roli dílčích cílů, které by měl uživatel plnit a tím se postupně zlepšovat, aby dosáhl svého cíle. Výzvy jsou inspirovány tzv. *boss fighty*, které můžeme nalézt v počítačových hrách, jsou symbolem souboje všeho, čeho hráč dosáhl za celý čas hraní proti nějakému velmi silnému protivníkovi. V navrhované aplikaci budou mít výzvy podobu nějakého těžkého úkolu, který bude jakýmsi podúkolem na cestě k uživatelem vytyčenému cíli. Pro příklad aplikaci bude používat uživatel, jenž se chce začít věnovat běhání, postupnými úkoly se bude postupně zlepšovat a časem bude schopný překonat výzvu, která bude mít například podobu úkolu ve formě: „Uběhni 8 kilometrů pod 40 minut.“ Výzvy budou v navrhované aplikaci tedy dlouhodobé cíle, které by uživatele měly posunout na další úroveň výkonnosti.

Kapitola 3

Analýza existujících aplikací

Mobilních e-health aplikací je celá řada, z toho důvodu je relativně těžké vybrat takové, které by se daly označit za podobné navrhované aplikaci. Nicméně níže zmíněné aplikace sdílí s navrhovanou aplikací alespoň některé z vlastností, ať to je gamifikace a nebo zaměření aplikace. Snaha byla vybrat takové aplikace, které jsou v něčem jiné a v něčem vyčnívají.

3.1 Zombies, RUN!

První vybranou aplikací je aplikace s názvem *Zombies, RUN!*[1], jedná se o nápaditou aplikaci, která vyčnívá z řady podobných aplikací velmi povedeným systémem gamifikace. Aplikace uživatele uvede do příběhu, ve kterém je jeden z přeživších zombie apokalypsy a jeho cílem je nejen přežít, ale i zachránit svět. Jak již druhá část názvu napovídá, hlavní náplní i přes veškeré gamifikační prvky zůstává samozřejmě běh. Pohyb je zasazen do příběhu takovým způsobem, že uživatel jakožto přeživší, vyráží na různé výpravy na nichž nachází suroviny, výbavu a různé další užitečné věci, které následně po úspěšném ukončení výpravy může využít ke stavbě svého městečka. Aby toho nebylo málo, každá výprava sama o sobě obsahuje namluvený příběh (*voice acting*), který uživatele vtahuje do děje. Měření vzdálenosti probíhá na základě GPS, nebo pomocí krokoměru, ve který se změní smartphone.

3.1.1 Výhody

Hlavní výhodou této aplikace je především rozsáhlý systém gamifikace, který téměř přiměje uživatele myslet si, že to ve skutečnosti není e-health aplikace, ale již hra, ve které se pohybuje. Aplikace nabízí různé aktivity spojené s během (např. běh do určité lokace za účelem získání různých odměn, samotné příběhové mise, stavbu vlastního města, atp.).

Její další nespornou výhodou je fakt, že i přes všechny gamifikační prvky zůstává v jádru e-healthovou aplikací a donutí uživatele se hýbat, skrz sledování GPS nedává uživateli jinou možnost, než běžet a nebo virtuálně doslova zemřít. Což je věc, které v naší aplikaci budeme dosahovat jen těžko, vzhledem k tomu, že neexistuje cesta jak opravdu zkontrolovat, zda uživatel daný pohyb vykonal či nikoliv.

Jednou z dalších výhod je taktéž nezávislost na internetovém připojení, aplikace si stáhne veškerá data při své instalaci a poté již nepotřebuje ke své funkcionalitě připojení, což je v dnešní době u spousty aplikací opomíjeno.

[1] Dostupné z: https://zombiesrungame.com/

V neposlední řadě je třeba zmínit její jednoduchý, ale velmi poutavý vzhled, který perfektně pasuje k tématice aplikace. Zejména obrazovky zobrazující výběr misí a pak samotný průběh mise jsou obzvlášť povedené - příjemně jednoduché a zároveň funkční.

3.1.2 Nevýhody

Za hlavní nevýhodu této aplikace by se dal označit fakt, že běží pouze na smartphonech (iOS, Android) a neexistuje tedy její desktopová verze. Nicméně vzhledem k povaze aplikace, lze považovat tuto nevýhodu za zcela irelevantní, protože by na desktopu bylo jen těžko možné tuto aplikaci využívat. Za nevýhodu bych však již považoval omezenost pouze na běh, samozřejmě aplikace takto skvěle funguje, nicméně by se dal zajisté rozšířit koncept i na jiné pohybové prvky, čehož se snaží docílit naše aplikace.

Další nevýhodou je nedostatečné sledování postupu uživatele, mám za to, že aplikace dostatečně neodměňuje uživatele za odvedený výkon, mise se dají zvládnout v docela „ležérním“ tempu a nebýt povedeného příběhu, tak by to uživatele nejspíš brzy omrzelo.

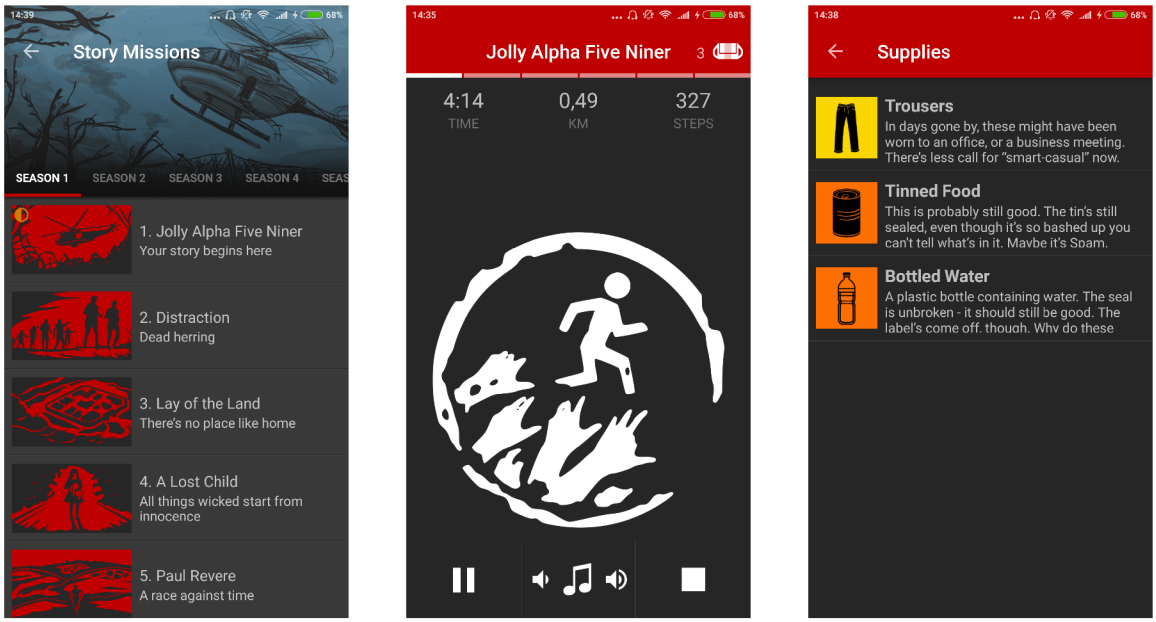
Poslední nevýhodou jsou relativně velké požadavky na paměťový prostor, velikost aplikace se šplhá do stovek MB, což je ovšem daň za to, že ke svému běhu nepotřebuje internetové připojení. V tomhle ohledu to lze aplikaci prominout, protože je bezesporu lepší stáhnout data dopředu, než pak využívat datového připojení v průběhu cvičení.

V čem splňuje naše požadavky:

- Je to e-health aplikace motivující k pohybu,
- obsahuje gamifikaci,
- má jednoduché, tematicky barveně sladěné a přehledné uživatelské prostředí,
- vhodná pro uživatele bez zkušeností s e-health aplikacemi - aplikace je vtáhne a nepustí, čímž je donutí se hýbat pravidelně.

V čem nesplňuje naše požadavky:

- Není cross-platform, protože nepodporuje web,
- dostatečně neodměňuje uživatele za odvedený výkon,
- omezena pouze na běh.



Obrázek 3.1: Zleva: Nabídka misí, průběh mise, procházení získaných zdrojů

3.2 Freelectics bodyweight

Druhou vybranou aplikací je aplikace s názvem *Freelectics bodyweight*[2], která je již klasickým typem fitness aplikace. Uživatel si zde může vytvářet tréninkové plány nebo plnit již připravené tréninkové plány, které jsou zaměřené na jednotlivé části těla, případně na zdokonalení konkrétních vlastností (síla, vytrvalost atp.). Za každý provedený cvik získává uživatel body a zvyšuje úroveň svého profilu, což je snad jediným gamifikačním prvkem této aplikace. Nesmím tedy opomenout žebříčky, kde se uživatel může porovnávat s ostatními jak v rychlosti tak i v počtu zvládnutých opakování daného cviku. Aplikace taktéž obsahuje sociální část, kam uživatel může sdílet své výkony a sledovat zároveň výkony svých přátel. Cviky jsou buď měřeny časovačem nebo tlačítkem, kterého se musí uživatel dotknout po každém odvedení jednoho cviku (tzv. *live count*).

3.2.1 Výhody

Největší výhodou této aplikace je skutečně rozsáhla nabídka cvičebních plánů a tréninků, navíc má uživatel možnost si z databáze cviků vybrat takové, které mu vyhovují a poskládat si je do svého vlastního plánu. Důležité je podotknout, že každý cvik v databázi má popis, na který sval je zaměřený, úroveň obtížnosti a videoukázku, jak ho správně technicky provádět.

Další výhodou je sociální propojení s přáteli, nic nemotivuje uživatele víc, než porovnávání se s lidmi, které zná. Navíc je tu, již zmiňovaný systém získávání bodů za jednotlivé cviky a zvyšování úrovně svého profilu, což uživateli ve výsledku nepřinese žádný užitek, na druhou stranu je však zajisté příjemné dosáhnout vysoké úrovně a velkého množství bodů.

Poslední výhodou je, stejně jako u předešlé aplikace, přehledné uživatelské prostředí, které je prosté velkého množství nastavení. V menu aplikace lze najít v podstatě jen pět položek, ve kterých se snado a rychle orientuje a k samotnému cvičení lze přejít po pár kliknutích.

[2]Dostupné z: https://www.freeletics.com/

3.2.2 Nevýhody

Největší nevýhodou této aplikace je její závislost na internetovém připojení, bez kterého v aplikaci nefunguje prakticky nic. Sice lze běžně absolvovat trénink, nicméně nejsou v něm dostupné obrázky ani videoukázky cviků a pochopitelně nefunguje ani sociální část aplikace.

Za další nevýhodu bych označil nevstřícnost k uživatelům, kteří se cvičením jako takovým nemají zkušenosti, takoví uživatelé maji tendenci cvičení po pár pokusech zanechat a aplikace nemá nic, čím by je přitáhla zpět, protože je postavená spíše na poměřování se s přáteli. Bodování a úrovně jsou tu spíše okrajová záležitost.

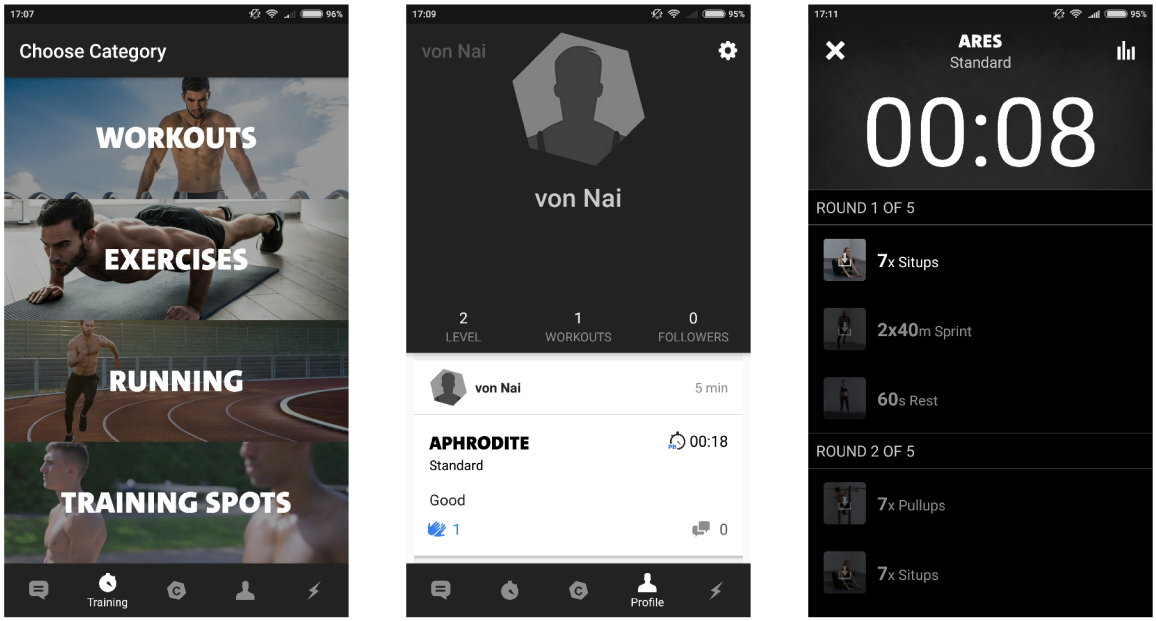
Ani tato aplikace nenabízí svou webovou podobnu a soustředí se pouze na mobilní platformy (Android, iOS).

V čem splňuje naše požadavky:

- Je to e-healtch aplikce,
- obsahuje určitou část gamifikace,
- má jednoduché moderní prostředí, podporující to, co má aplikace vykonávat,
- široký výběr cviků, každý uživatel si najde svoje.

V čem nesplňuje naše požadavky:

- Nevhodné pro nezkušené uživatele, aplikace nemá motivační prvky k tomu, aby udržela pravidelnost používání,
- není cross-platform, protože nepodporuje web,
- dostatečně neodměňuje uživatele za odvedený výkon – nedostatečná gamifikace.



Obrázek 3.2: Zleva: Hlavní nabídka, profil uživatele, průběh tréninku

3.3 Fitocracy

Třetí a poslední vybranou aplikací je aplikace s názvem *Fitocracy*[3], která se vydala svou vlastní cestou. V jádru je to klasická fitness aplikace, nabízející opět spoustu tréninků a cviků, k posílení všech částí těla. Nicméně navenek vystupuje jako sociální síť ne nepodobná Facebooku. Umožňuje sdílet dokončené tréninky, získané achievementy, fotky, stejně tak jako prohlížet příspěvky jiných uživatelů a přátel. Aplikace navíc obsahuje opět systém úrovní, kde uživatel získává za dokončené tréninky zkušenosti a tím zvyšuje úroveň svého profilu, taktéž jsou dostupné úkoly, které může uživatel plnit a následně sdílet na své zdi.

3.3.1 Výhody

Hlavní výhodou této aplikace je bezesporu vysoký počet motivačních prvků – ať jsou to již prvky gamifikační (úrovně, achievmenty, úkoly), nebo prvky sociální sítě (sdílení výkonů, psaní příspěvků a komentářů, atd.). To vše působí motivujícím dojmem a vzbuzuje pocit, že si aplikace své uživatele udrží.

Další výhodou je fakt, že aplikace běží jak na mobilních telefonech, tak i na desktop zařízeních, má totiž svou web verzi. Data jsou potom automaticky synchronizována mezi webovou a mobilní verzí.

Za poslední výhodu považuji vzhled webové aplikace, která vypadá jako opravdová sociální síť, kterou tedy ve skutečnosti je. Nicméně vzhled desktopové aplikace je skutečně povedený, prostředí působí intuitivně, jsou zachovány veškeré prvky, které jsou uživateli dobře známé například z Twitteru, či Facebooku.

3.3.2 Nevýhody

Hlavní nevýhodou této aplikace je omezenost mobilní verze, která sice umožňuje plánovat a provádět tréninky a sdílet své výsledky, ale o proti webové verzi je funkcionalita značně osekaná. Navíc, ani vzhled této aplikace není příliš povedený, uživatelské prostředí je zastaralé a zmatené.

Další nevýhodou je znovu a opět závislost na internetovém připojení. V mobilní aplikaci nejsou dostupné ani obrázky jednotlivých cviků, samozřejmě nefunguje ani sociální část aplikace, nicméně toto nelze ovlivnit. Naštěstí aplikace alespoň ukládá veškerý postup a při znovudostupnosti internetového připojení probíhá synchronizace.

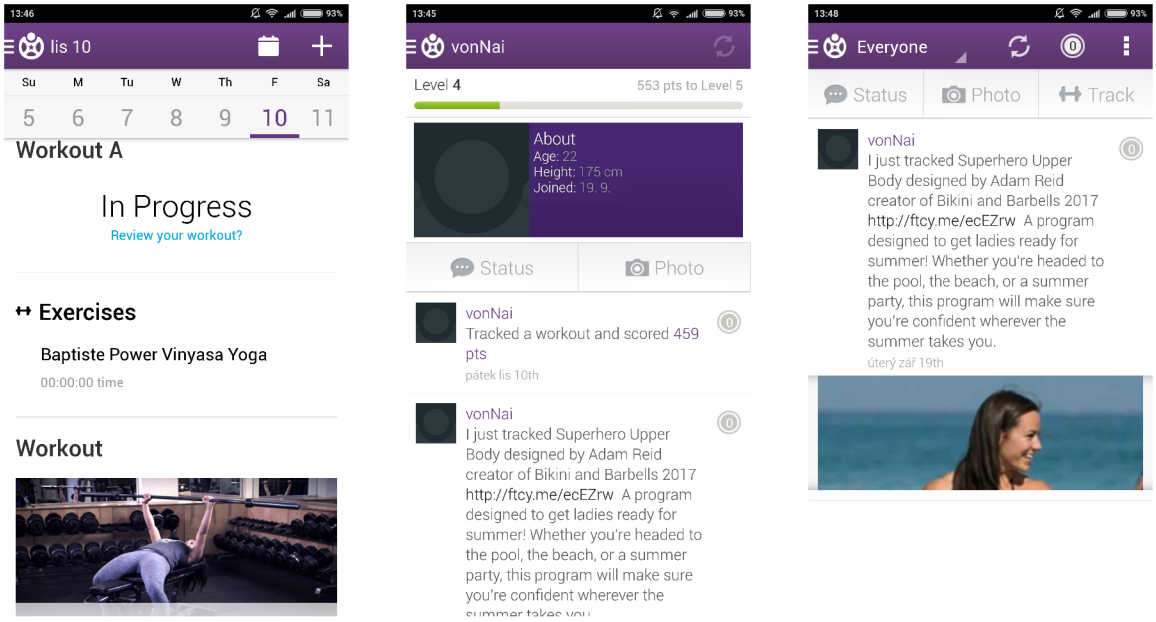
V čem splňuje naše požadavky:

- Je to e-health aplikace motivující k pohybu,
- obsahuje gamifikaci,
- cross-platform (Android, iOS, web),
- pro kohokoliv.

V čem nesplňuje naše požadavky:

- Omezená funkcionalita mobilní verze,
- nehezký vzhled mobilní verze.

[3]Dostupné z: https://www.fitocracy.com



Obrázek 3.3: Zleva: Plánování tréninku, profil uživatele, zeď uživatele

Kapitola 4

Analýza požadavků

Aplikace bude určena pro kohokoliv, kdo se bude chtít bavit a pohybovat. Zejména však bude aplikace cílit na ty uživatele, kteří se pravidelně nepohybují nebo nemají motivaci se pohybovat. Nelze tedy přímo určit konkrétní skupinu uživatelů, nicméně lze říci, že aplikace bude přístupnější uživatelům, kteří nesportují (viz Podkapitola 4.1). Vzhledem k povaze aplikace a zadání práce, byl zvolen cross-platform přístup. Aplikace by tedy měla mít svoji web, Android i iOS verzi. Uživatelské rozhraní by mělo být jednoduché, poutavě barevně sladěné, v základu zobrazovat pouze základní funkcionalitu. Tedy mělo by být v každém okamžiku jasné, k čemu je aplikace určena a jak se k této funkci jednoduše dostat. Velký důraz musí být taktéž kladen na responsivní design, aplikace musí vypadat slušně na desktopu stejně jako na chytrém telefonu. Uživatelské prostředí by mělo vhodně, ale výrazně zobrazovat odměnu za odvedený výkon, aby měl uživatel pocit, že se mu vyplatilo vynaložit takové úsilí. To stejné platí pro zobrazování postupu.

Je nutné na straně uživatele ukládat veškerá data spojená s jeho postupem, ale i data, které by byly za normálních okolností uloženy v serverové databázi.

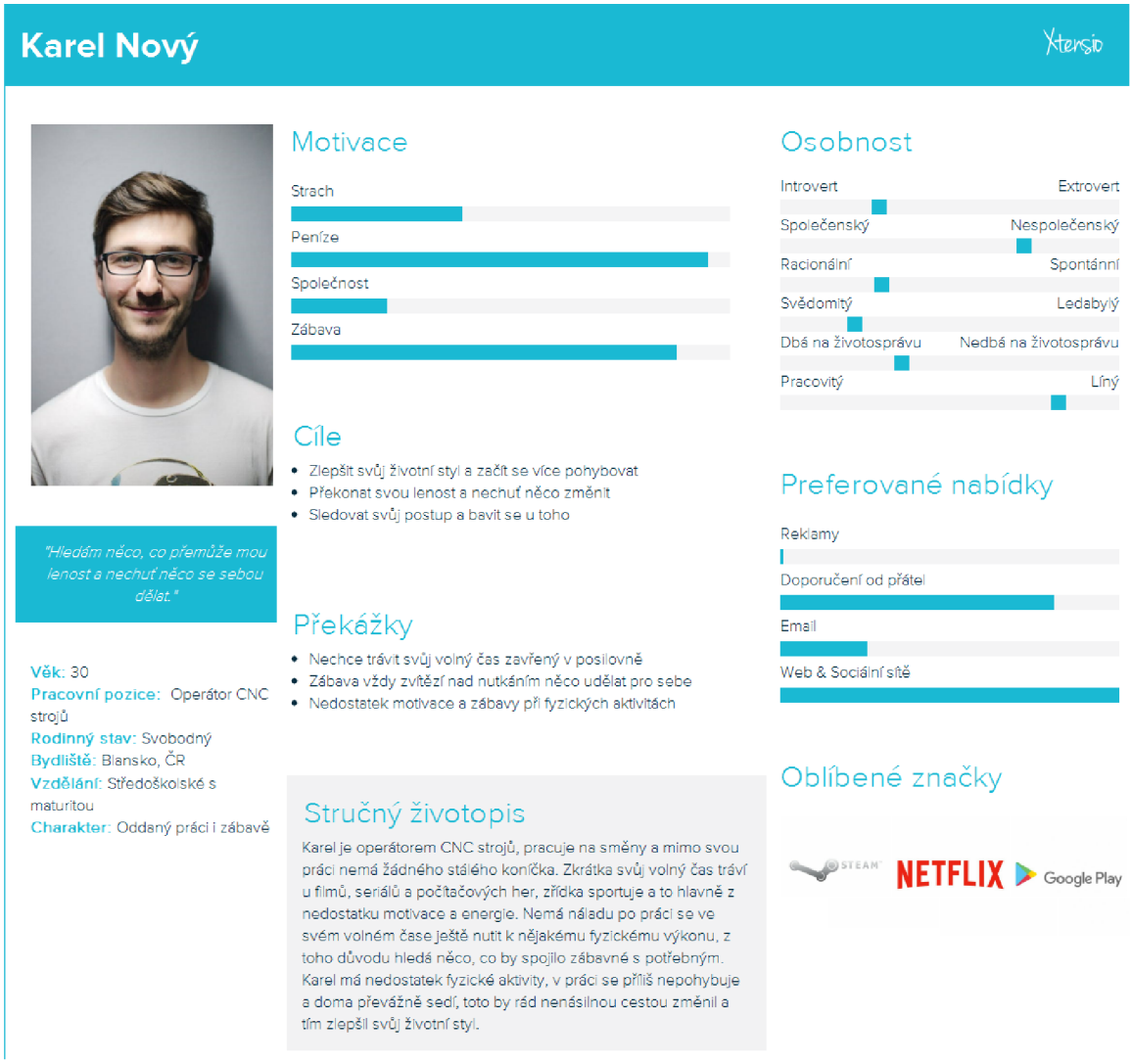
Veškerá data, které bude aplikace zobrazovat (tréninky/cvičební plány, atp.) je nutné navrhnout a ručně zapsat do aplikace.

Shrnutí požadavků:

- Web, Android, iOS,
- responzivní design,
- jednoduché a přívětivé uživatelské prostředí,
- ukládání dat na straně uživatele - využití local storage/indexedDB,
- prvky gamifikace, motivující uživatele se k aplikaci vracet,
- zobrazování postupu a získaných odměn,
- pohybové úkoly a tréninkové plány, které by měl uživatel plnit,
- vizualizace měření času při každém úkolu,
- aplikace by měla měla být pro kohokoliv, ať už je to atlet, nebo běžný člověk, co se chce pohybovat a bavit.

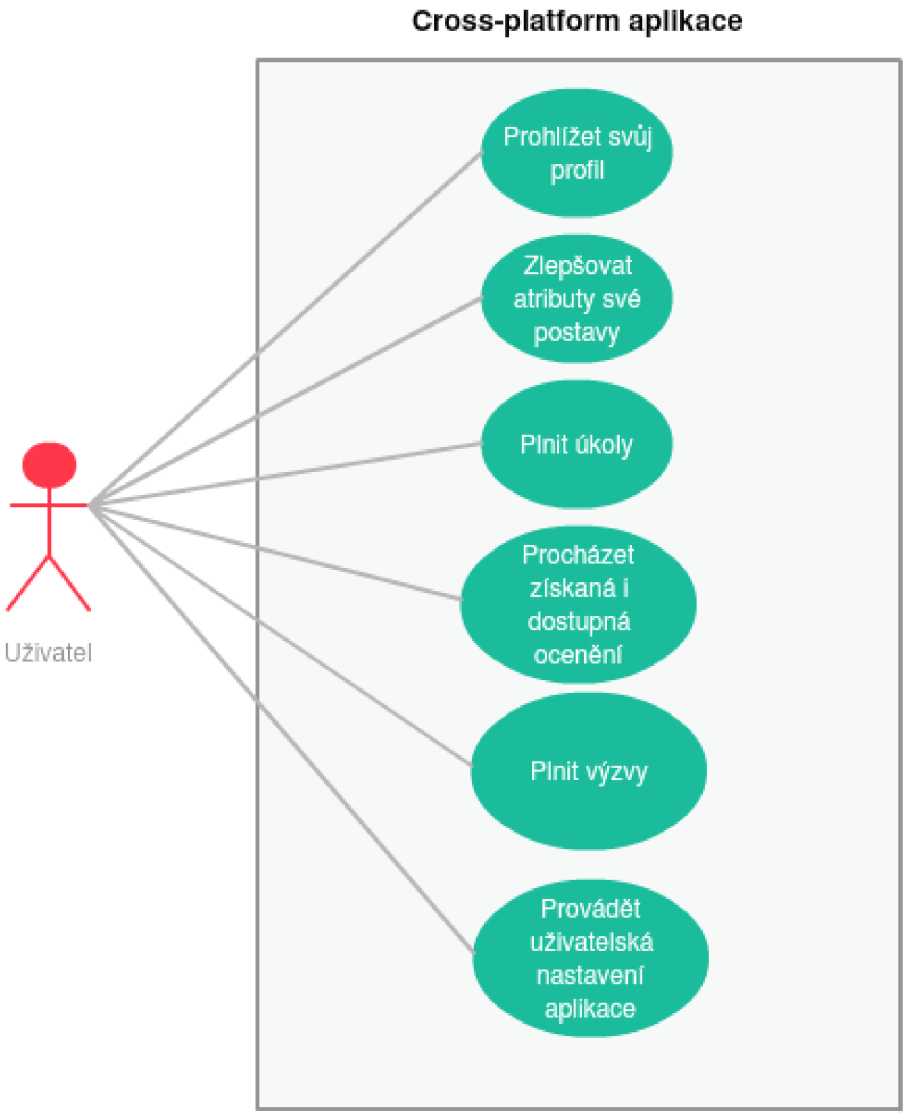
4.1 Uživatelská persona

Uživatelská persona detailně popisuje typického uživatele navrhované aplikace [3]. Zachycuje uživatelovi typické vlastnosti relevnantní pro výslednou aplikaci, stejně jako jeho cíle a problémy, pro které hledá řešešní. V tomto konkrétní případě lze vidět (obrázek 4.1), že typický uživatel navrhované aplikace je relativně mladý člověk (na pohlaví v tomto případě nezáleží), který hledá motivaci pro zvýšení své pohybové aktivity. Z persony lze vyčíst, že uživatel je spíše nespolečenský člověk, který dbá více na zábavu než na fyzickou aktivitu a mimo svou práci nedělá nic pro sebe. Toto by rád změnil, nemá však dostatečně silnou vůli, aby dokázal vyměnit zábavu za fyzickou aktivitu, z toho důvodu by rád našel řešení, které tyto dvě věci spojuje.



Obrázek 4.1: Uživatelská persona

4.2 Diagram příkladu užití



Obrázek 4.2: Use case diagram

Kapitola 5

Analýza dostupných technologií a knihoven

JavaScriptových frameworků a nástrojů pro vytváření cross-platform aplikací existuje celá řada, většinu z nich však spojuje fakt, že využívají služeb frameworku Cordova pro nasazení aplikace do nativního prostředí (angl. *deployment*). Existují i frameworky, které překládají výchozí JavaScriptový kód do nativních jazyků jednotlivých platforem, případně používají JavaScriptové interprety, těmi se však zabývat nebudeme. Všechny níže uvedené (s výjimkou Cordova) jsou ve své podstatě front-end frameworky interně využívající Cordova.

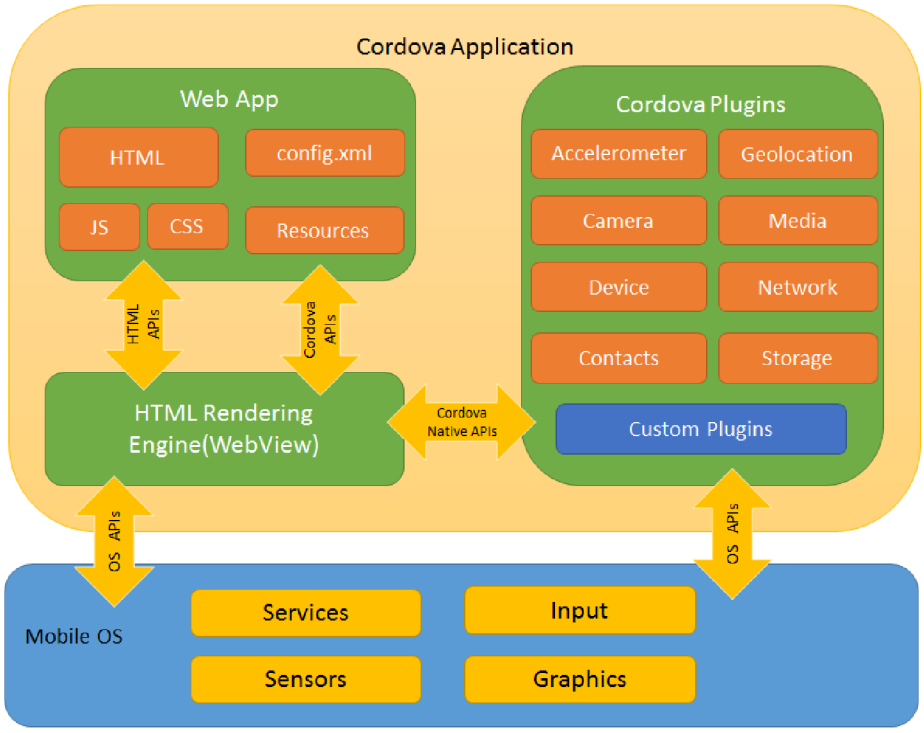
5.1 Frameworky a nástroje pro tvorbu cross-platform aplikací

5.1.1 Cordova

Je framework pro vývoj cross-platform aplikací pro platformy: Android, iOS, Web, Windows. Umožňuje implementovat aplikace pro mobilní zařízení za použití klasických webových technologií HTML5, CSS3, JavaScript. Jinými slovy, dokáže zapouzdřit typickou webovou aplikaci do nativního prostředí cílové platformy s využitím veškerých nástrojů a funkcí dostupných na oné platformě [7].

Ve své podstatě Cordova zapouzdřuje implementované uživatelské prostředí (webovou aplikaci) do prostředí označovaného jako WebView, kolem kterého jsou postaveny jednotlivé pluginy komunikující s danou platformou (viz Obrázek 5.1). Tyto pluginy umožňují ovládání systémových nástrojů na cílovém zažízení (typicky kamera, GPS, síť, atp.). Díky tomuto přístupu stačí vytvořit webovou aplikaci a framework ji zaobalí tak, aby následně fungovala na všech výše zmíněných platformách. Z toho vyplývá, že veškerá aplikační logika musí být implementována jazykem JavaScript, což nebývá typické pro většinu webových aplikací, které využívají JavaScript pouze pro front-end. Zde je však nutné použít JavaScript i pro back-end a veškerou funkcionalitu. Uživatelské prostředí je potom klasicky definováno prostředky jazyka HTML a CSS [1].

Mnoho nástrojů a frameworků je nadstavěno na Cordova a využívá jeho funkcionality.



Obrázek 5.1: Schéma aplikace s využitím Cordova (zdroj: [8]).

5.1.2 Ionic

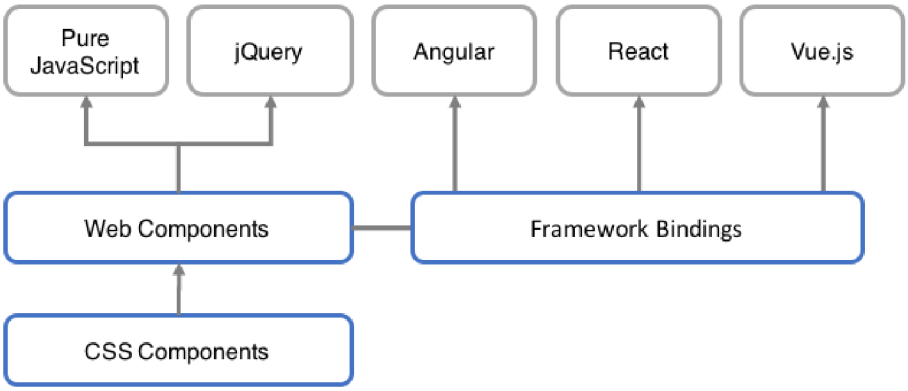
Je prvním z vybraných frameworků. Jedná se primárně o HTML5 framework, pro nasazení aplikace je tedy nutné využít služeb výše uvedeného Cordova. Ionic je k dispozici pod open-source licencí, implementován je v jazyku JavaScript, a jeho hlavním cílem je zjednodušit vývoj cross-platform aplikací. Ionic stejně jako mnoho dalších frameworků nabízí řadu implementovaných responzivních elementů, které usnadňují vývoj aplikace (mezi takové elementy patří například různá menu, záložky, navigační lišty, animace atp.). Narozdíl od ostatních frameworků se však Ionic snaží, aby aplikace vypadala nativně pro dané prostředí, tomu jsou uzpůsobeny všechny elementy a animace. Vše v případě Ionicu stojí nad JavaScriptovým front-end frameworkem AngularJS. Jak již bylo zmíněno výše, pro nasazení aplikace na cílovou platformu využívá služby frameworku Cordova a implementačními jazyky jsou HTML, CSS, JS a případně Sass [16].

Mezi hlavní výhody tohoto frameworku jednoznačně patří nativní vzhled aplikací, rozvinutá vývojářská komunita a využití známého a úspěšného frameworku pro web development - Angularu. Taktéž je možné zmínit relativně podstatnou výhodu, kterou je podpora ve vývojovém prostředí Visual Studio, ve kterém lze velmi snadno emulovat cílový operační systém a taktéž usnadňuje *release* (vydání) aplikace. Na druhou stranu vyšlo při testování najevo, že Ionic aplikace mají mírné problémy s výkonem, jevily se jako pomalejší ve srovnání například s aplikacemi konkurenčního OnsenUI (5.1.3), který je značně odlehčený a díky tomu rychlejší.

Ionic je frameworkem srovnatelným s nativním SDK (*Software Development Kit*), je skvělou volbou pro malé i velké cross-platform aplikace, nabízí mnoho implementovaných funkcí a knihoven. Obsahuje celou řadu volně použitelných ikon aplikací a ovládacích prvků.

5.1.3 OnsenUI

Je velmi podobným JavaScript frameworkem jako výše uvedený Ionic. Je k dispozici pod open-source licencní a je volně použitelný i pro komerční aplikace. OnsenUI se však zásadně liší od Ionic frameworku v podpoře ostatních frameworků, tam kde je Ionic nadstaven pouze nad AngularJS, podporuje OnsenUI využití všech rozšířených frameworků a umožňuje dokonce i vývoj v čistém JavaScriptu bez použití jakéhokoliv frameworku. Lze volně přepínat mezi následujícími frameworky: Angular, React a Vue.js [19]. Podobně jako Ionic obsahuje i Onsen mnoho implementovaných responsivních elementů a knihoven, které zjednodušují přenos aplikace na mobilní zařízení. Přenos aplikace na mobilní zařízení opět zařizuje Cordova. Na rozdíl od Ionicu však neklade takový důraz na nativní vzhled aplikací a lze ho použít i pro vývoj mobilních web aplikací. Mezi hlavní výhody tohoto frameworku patří jeho jednoduchost, přehlednost a flexibilita, za hlavní nevýhodu lze označit malou vývojářskou komunitu a s tím spojené nesnáze jako je nedostatečný počet výukových textů, řešení a odpovědí na otázky, které mohou vzniknout při vývoji aplikací.



Obrázek 5.2: Architektura OnsenUI (zdroj: [19]).

5.1.4 Framework7

Je dalším HTML5 open-source frameworkem implementovaným v jazyce JavaScript. Mezi jeho hlavní výhody patří rychlost a jednoduchost. Framework7 není postavený nad žádnou *third-party* (tzv. aplikace třetích stran) knihovnou pro manipulaci s DOM (*Document Object Model*), místo toho obsahuje svou vlastní knihovnu označovanou jako DOM7 [17]. Syntax je velmi podobný knihovně jQuery, z čehož plyne jednoduchost jeho použití. Ani zde nechybí celá řada responzivních komponent a snaha o co nejvěrohodnější napodobení nativních aplikací. Přesto, že Framework7 není narozdíl od Ionic postaven nad žádným *third-party* front-end frameworkem a spoléhá tak pouze na své vlastní prostředky, je možné v kombinaci s ním jednoduše využít Angular, React nebo Vue.js. Hlavní nevýhodou tohoto frameworku je omezenost platforem, na které lze vyvíjet aplikace. Nasazení aplikace opět zajišťuje Cordova, nicméně je možné vytvářet pouze aplikace pro Android a iOS. Vývojáři Framework7 jsou toho názoru, že je lepší nadřadit rychlost a jednoduchost nad plný cross-platform přístup [14].

5.2 Front-end knihovny a frameworky

Téměř ve všech výše zmíněných frameworcích se objevila zmínka o využití jiných front-end frameworků a knihoven jako jsou Angular, React nebo Vue.js. Z tohoto důvodu je důležité si některé z nich aľespoň lehce přiblížit. V následujících podkapitolách budou uvedeny zmiňované frameworky Angular, Vue.js a také knihovna React.

5.2.1 AngularJS

Je jedním z nejpopulárnějších front-end web frameworků, vyvíjený a udržovaný společností Google. Implementovaný je opět pouze v jazyce JavaScript, do HTML dokumentu ho lze přidat přes HTML tag `script` a následně lze plně využívat jeho funkcionalitu. Snahou tohoto frameworku je rozšířit využití jazyku HTML, umožňuje vytvářet vlastní HTML tagy, kterým lze přiřadit libovolnou funkcionalitu [12]. Stejně tak nabízí mnoho připravených HTML tagů, na něž je navázána celá funkcionalita Angularu jako je například *two-way data binding*, který zajišťuje automatickou synchronizaci dat mezi back-end a front-end částí aplikace. Angular implementuje návrhový vzor MVC (*Model View Controller*), kde jednotlivé HTML tagy zastupují *view* a na ně navázané JavaScript funkce potom *controller*. Provázání zajišťuje zmíněný *two-way data binding*, *controller* potom ovládá a mění obsah *model*, který se automaticky promítá do *view*. Celou zde popsanou funkcionalitu lze zajistit pouze využitím vhodných Angular HTML tagů (*ng-model*, *ng-controller* a dalších) [4].

Mezi hlavní výhody Angularu patří jeho robustnost a široké použití jakožto i implementace návrhového vzoru MVC, čímž odděluje aplikační logiku od uživatelského rozhraní. Angular umožňuje vývoj složitých aplikací s lehkostí a snad s nejmenším počtem řádků kódu. Poslední výhodou je rozsáhlá vývojářská komunita, spousta návodů, otázek a odpovědí, výukových textů a velmi dobře zpracovaná dokumentace [23].

Mezi nevýhody tohoto frameworku lze zařadit jeho netradičnost a s tím spojenou neintuitivnost. Zkrátka Angular je nutné se naučit od základu znovu, jeho využití a prostředky jsou však takové, že se to bezesporu vyplatí. Další nevýhodou je jeho výkon při velkém množství aktivních prvků, které musí sledovat *two-way binding*. Angular toto řeší sekvenční kontrolou každé proměnné, která hlídá změny jednotlivých prvků v *model* a následně je promítá do *view* (toto se označuje jako *dirty-checking*). Pokud je však těchto sledovaných prvků velké množství, projeví se to na výkonu a rychlosti odezvy výsledné aplikace [12].

5.2.2 React

React je open-source JavaScript knihovnou vyvíjenou a udržovanou společností Facebook. Tentokrát se však jedná čistě o front-end knihovnu, specializující se na vytváření uživatelských prostředí a zobrazování často se měnících dat. React nemá prostředky pro práci s daty, narozdíl od Angularu, React zastupuje pouze *View* v návrhovém vzoru MVC [18]. Knihovna umožňuje deklarovat jak má uživatelské rozhraní vypadat v jakých situacích a následně dokáže uživatelské rozhraní dynamicky měnit na základě změny dat v back-end a to bez nutnosti znovunačtení webové stránky. Toto je umožněno použitím tzv. virtuálního DOM (*Document Object Model*), který React na pozadí buduje, porovnává s aktuálně vyrenderovaným reálným DOM a následně do něho promítá změny, čímž je React osvobozen od opakovaného znovubudování celého stromu DOM při každé změně. React lze, podovně jako Angular, zapisovat přímo do HTML dokumentu a to buď čistě jako JavaScript anebo jako jeho rozšíření JSX (JavaScript XML), což je transformace standardní syntaxe JavaScript

tak, aby umožňovala zapisovat *HTML like* tagy, které jsou poté renderovány jako opravdové HTML prvky. To slouží zejména k vytváření nových komponent uživatelského rozhraní [9].

Některé z hlavních konceptů už byly vysvětleny (JSX a Virtual DOM), ještě je důležité zmínit, jakým způsobem React buduje jednotlivé prvky uživatelského prostředí. K tomu slouží tzv. *React elements*, což jsou JavaScriptové objekty reprezentující HTML prvky, a dále také tzv. *React components*, které obvykle zahrnují větší části uživatelského prostředí a definují jak strukturu tak jeho funkcionalitu. Obě tyto zmíněné entity lze velmi snadno vytvářet použitím JSX [9].

React je skvělou volbou pro tvorbu uživatelského prostředí, je vyvíjený za účelem nahradit populární knihovnu jQuery, dokáže s lehkostí zvládnout obrovský počet měnících se dat (viz Facebook), výsledné prostředí je responzivní a velmi rychlé [5]. Na druhou stranu však React neposlouží jako jediný framework nebo knihovna pro vývoj aplikací, neboť nemá prostředky pro práci s back-end, je proto nutné ho použít v kombinaci s libovolným jiným frameworkem, který toto dokáže a nechat React dělat práci, kterou dělá dobře – budovat uživatelské prostředí.

5.2.3 Vue.js

Jedná se o další z mnoha open-source JavaScript front-end frameworků. Není nepodobný výše zmíněným, s oběma sdílí několik vlastností a funkcí. Vue.js je framework primárně se zaměřující na budování uživatelského prostředí podobně jako React, nicméně je využitelný i pro komplexní SPA (*Single Page Applications*) podobně jako Angular. S knihovnou React sdílí několik funkcionálních vlastností:

- Využívá virtuální DOM,
- umožňuje definovat vlastní prvky uživatelského prostředí,
- obsahuje podporu pro JSX, i když nabízí jinou cestu, kterou lze definovat užitavelské prostředí.

Vue.js nabízí možnost namísto JSX využít tzv. šablony (*templates*), což jsou v podstatě validní HTML dokumenty se speciálními Vue.js tagy, kterých využívá například také Angular. Tento přístup usnadňuje efektivní psaní kódu a taktéž jej činí přehlednějším. Obecně ve Vue.js platí, že pro prezentační a renderovací komponenty je vhodné využít šablon, a pro logické a funkcionální komponenty potom JSX. Co se týče výkonu a rychlosti aplikací jsou Vue.js a React relativně srovnatelné a jejich přístup k renderování a práci s DOM je velmi podobný [20].

S frameworkem Angular sdílí zejména šablonový přístup, dokonce i přidané HTML tagy se jmenují podobně. Vue.js je však mnohem jednodušší co se týče syntaxe i celkového API, je taktéž funkcionálně méně rozsáhlé a tím pádem je snazší se ho naučit efektivně používat. Angular definuje pevnou strukturu aplikace a vyžaduje její dodržování, na druhé straně Vue.js je v tomto tolerantnější, za to však striktně rozlišuje mezi *Directives* a *Components*. *Directives* jsou ve Vue.js využívány výhradně pro manipulaci s DOM a *Components* jsou potom autonomní jednotky uživatelského rohzraní s definovaným vzhledem a aplikační logikou. V případě Angularu se hranice mezi těmito elementy často smývá. Co se týče rychlosti a výkonu, v případě Angularu je již zmíněný problém s využitím *dirty-checking*, v jehož důsledku je pomalejší než Vue.js, který tento přístup nevyužívá a bývá tím pádem rychlejší [20].

Vue.js je jakýmsi kompromisem mezi Angularem a Reactem, bere si silné stránky z obou zmíněných a jejich chyby se snaží napravit. Navíc je navržený tak, aby byl jednoduchý a umožňoval snadné použití s ostatními knihovnami a frameworky (tato vlastnost se v angličtině označuje jako *adoptable*). Lze ho snadno přidat do již existujících implementací a libovolně kombinovat s ostatními frameworky.

5.3 Databáze a ukládání dat

Z povahy aplikace vyplývá, že bude nutné ukládat různá data potřebná pro korektní chod. Vzhledem k tomu, že aplikace nepoběží jako klient-server, ale pouze u klienta, je třeba zde ukládat všechna data. Toho lze dosáhnout třemi způsoby. Všechny tři způsoby budou popsány v následujících podkapitolách.

5.3.1 Local storage

Je persistentní úložiště na straně klienta, kam lze ukládat libovolná data z internetového prohlížeče. Kapacita tohoto úložiště je omezená v závislosti na použitém prohlížeči zpravidla se omezení pohybuje mezi 5-10 MB, taktéž je nutné data serializovat, tedy ukládat je v párech „název - hodnota“, kde obě položky musí být datového typu string [22]. Typicky data setrvávají na svém místě i po zavření internetového prohlížeče a jejich persistence není ničím omezena. Na některých operačních systémech (typicky Android) však může dojít k jejich automatickému smazání z důvodu nedostatku místa, což je velkou nevýhodou tohoto přístupu neboť uživatel tak přijde o uložený postup a veškerá uživatelská data. Data uložená v local storage jsou přístupné pomocí JavaScriptových funkcí spouštěných v prohlížeči/aplikaci [13]. Local storage je podporován na všech platformách včetně všech webových prohlížečů [8].

5.3.2 IndexedDB

Je podobně jako local storage persistentní úložiště na straně klienta, jedná se však o objektově orientovaný databázový systém, který umožňuje ukládat JavaScriptové objekty, z čehož vyplývá, že se nejedná o relační databázi. Oproti relační databázi se zásadně liší způsobem ukládání dat, nenalezneme zde žádné tabulky a sloupce s hodnotami, nýbrž objekt, který zaručuje persistenci jiných objektů stejné třídy. IndexedDB taktéž ukládá data v párech, nicméně v tomto případě je to pár „klíč - hodnota“. Hodnota může být libovolně strukturovaný objekt a klíč může klidně být atributem tohoto objektu. Lze vytvářet indexy, které vyhledávají objekty podle zvoleného atributu. Veškeré změny i čtení se provádí za pomocí transakcí, nelze měnit stav databáze, ani nad ní provádět příkazy mimo transakce. Transakce zajišťuje konzistenci dat při přístupu z více instancí aplikace. Důležité je zmínit, že tento databázový systém nepoužívá Structured Query Language (SQL), jedná se o tzv. NoSQL databázi. Na místo totoho přístupu jsou zde dotazy na indexy, jejichž výsledkem je kurzor, pomocí kterého lze iterovat skrz navrácené hodnoty. O proti local storage má značně vyšší kapacitu, ukládaná data není třeba serializovat a lze je ukládat ve strukturované formě, což je velkou výhodou tohoto přístupu, nicméně trpí stejným problémem jako local storage, uložená data může operační systém v případě nedostatku volného místa uvolnit [13]. IndexedDB není podporována na platformě iOS, pro tuto platformu je nutné využít jiné úložiště [8].

5.3.3 Web SQL Database

Je posledním lokálním úložištěm. Z názvu lze odvodit, že se opět jedná o databázi, nicméně tentorkát jde skutečně o relační databázi, nad kterou lze provádět SQL dotazy. Hned na počátku je však důležité zmínit, že je v dnešní době označena World Wide Web Konzorciem za zastaralou a její použití není doporučeno (v prohlížeči Firefox není dokonce ani podporována) a to z důvodu nedostatečné podpory nezávislých implementací (použití jiného databázového systému než SQLite). Její nástupcem je výše zmíněná IndexedDB [21]. Přestože WebSQL není podporována ve všech prohlížečích, je stále využitelná pro mobilní platformy. Framework Cordova ji dokáže využít pro platformy Android, iOS a BlackBerry [8].

Kapitola 6

Návrh aplikace

Tato kapitola se věnuje návrhu aplikace, budou zde představeny zvolené technologie jak pro tvorbu uživatelského prostředí, tak pro ukládání dat. Následně bude představen a popsán mockup výsledné aplikace, tedy předběžný návrh uživatelského prostředí a jeho jednotlivých prvků. Závěrem této kapitoly budou navrženy základní prvky a funkce aplikační logiky a databáze.

6.1 Zvolené technologie

6.1.1 Výběr technologie pro uživatelské prostředí

Na základě provedené analýzy v předcházející kapitole je zřejmé, že bude nutné použít framework Cordova pro nasazení aplikace na cílové platformy. Otázkou však zůstává, který a jestli vůbec nějaký framework nebo knihovnu použít v kombinaci s Cordova. Na základě všech zmíněných výhod a nevýhod bylo rozhodnuto, že nejlepším řešením bude využít služby frameworku Ionic a to zejména kvůli rozšířené vývojářské komunitě, podpoře frameworku Angular a podpoře ve vývojářském prostředí Visual Studio. Co se týče pro aplikaci reletanvních vlastností, Ionic nabízí nativní vzhled uživatelského prostředí a snadné využití frameworku Cordova, což ho činí pro navrhovanou aplikaci nejvhodnější volbou.

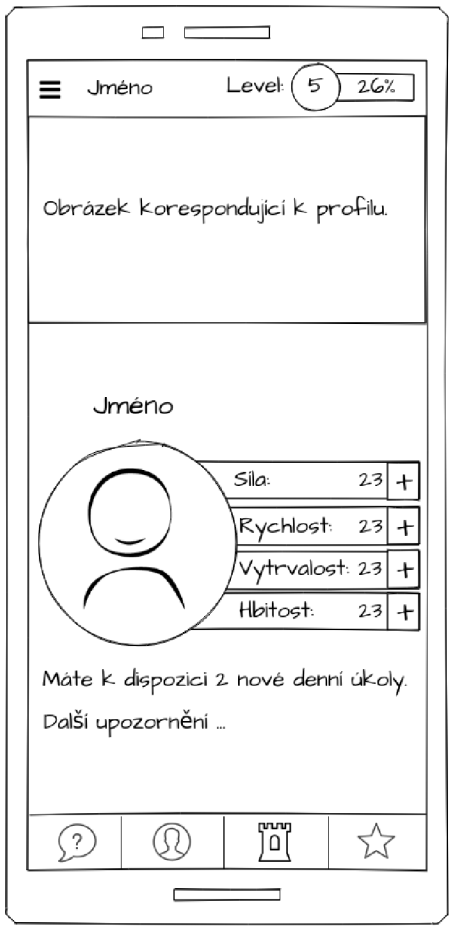
6.1.2 Výběr technologie pro ukládání dat

Zde bylo rozhodnutí jednoduché, vzhledem ke všem omezením Local storage je IndexedDB jasným favoritem. Nejen, že data není třeba serializovat, ale mít možnost data úkladat ve strukturované formě a následně vyhledávat jen ty data, které jsou skutečně relevantní v daném případě je výhodou, kterou nelze ničím vyvrátit. Zejména je-li třeba ukládat data různých kategorií, například tréninky jednotlivých atributů bude nutné ukládat odděleně, potom je vhodné, aby bylo možné je taktéž odděleně procházet a načítat z databáze, aby se zamezilo dlouhému čekání na data z databáze. Jak bylo však zmíněno v podkapitole 5.3.2, IndexedDB postrádá podporu platformy iOS, na základě této skutečnosti je nutné trochu pozměnit přístup k ukládaní dat. Aplikace běžící na webové platformě bude skutečně využívat IndexedDB z důvodů, které jsou popsané výše. Nicméně aplikace běžící v nativních prostředích jednotlivých platforem budou využívat Web SQL databázi, aby se předešlo problémům s nekompabilitou IndexedDB na platformě iOS.

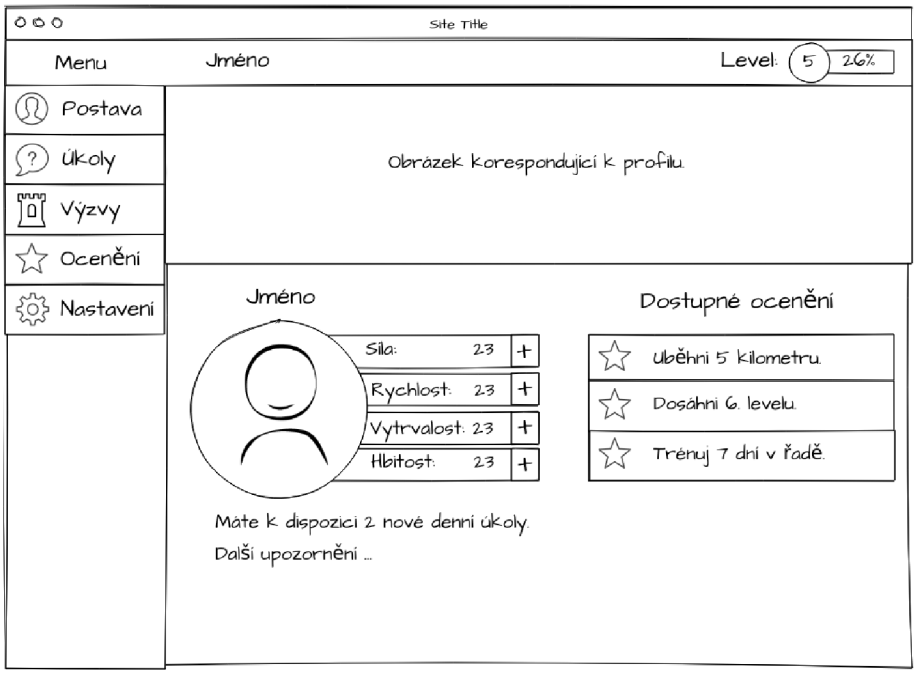
6.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (*Graphic User Interface*, dále jen GUI) je prostředkem, pomocí kterého bude uživatel komunikovat s aplikací a naopak [10]. GUI se bude starat o zobrazení veškerých informací o vytvořené postavě a vhodnou formou bude nabízet úkoly a prostředky pro zlepšení této postavy. Důležitou částí GUI je také vizualizace průběhu vykonávání tréninků (úkolů), získávání achievementů a dosahování nových úrovní.

Na obrázku 6.1 níže lze vidět navržené grafické uživatelské prostředí. Jednotlivé komponenty jsou přehledně poskládany tak, aby byly u sebe ty komponenty, které spolu nějak souvisí. Jako hlavní navigační prvek aplikace slouží lišta na spodním okraji obrazovky, kde lze přecházet mezi hlavními obrazovkami a funkcemi aplikace. Na spodní liště se tedy aktuálně nacházejí zleva odkazy na obrazovku s úkoly, profil vytvořené postavy, obrazovku s výzvami a obrazovku s achievementy. Ostatní navigační prvky jsou uschované v menu, které je skryté a lze ho otevřít tlačítkem na vrchní liště obrazovky. Mockup zobrazuje obrazovku profilu vytvořené postavy, která je zároveň výchozí obrazovkou aplikace, zde lze zvyšovat jednotlivé atributy postavy a zobrazují se zde i důležitá upozornění. Aktuální úroveň postavy a postup v aplikaci lze sledovat na vrchní liště, která bude, podobně jako lišta spodní, viditelná téměř ve všech obrazovkách aplikace. Ve skrytém menu se potom bude nacházet odkaz na uživatelské nastavení aplikace a informace o aplikaci.



Obrázek 6.1: Mockup na mobilním telefonu



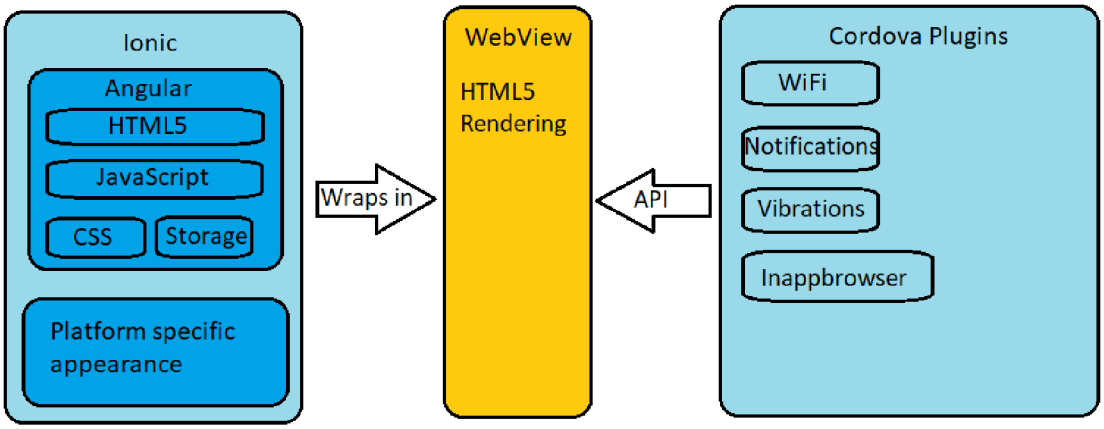
Obrázek 6.2: Mockup na desktopu

6.3 Aplikační logika a ukládání dat

Aplikační logika (*Application Programming Interface*, dále jen API) se bude starat o vstupy zadané uživatelem prostřednictvím GUI, bude spouštět žádané funkce, evidovat jejich výsledky a prezentovat výstupy uživateli prostřednictvím GUI. Hlavní částí API bude potom databáze, ze které se budou načítat jednotlivé úkoly, achievementy a výzvy. Úkolem této části aplikace bude taktéž kalkulace obtížnosti jednotlivých úkolů a cviků na základě vstupních informací od uživatele, atributů postavy a úspěšně dokončených respektive nedokončených úkolů.

6.3.1 Architektura aplikace

Architektura aplikace je naznačená na obrázku 6.3, na kterém lze vidět, že výsledná aplikace bude složena z prostředků frameworků Ionic a Cordova. Ionic zde bude zastupovat roli front-end a zároveň i některých funkcionálních věcí v back-end aplikace, jako je například práce s databází a aplikační logika. Cordova potom bude sloužit jako wrapper aplikace v nativním prostředí jednotlivých platforem. Bude zaobalovat webovou aplikaci vytvořenou pomocí Ionic do prostředí dané platformy a umožní aplikaci přístup k některým funkcím specifických pro mobilní platformy.



Obrázek 6.3: Architektura navrhované aplikace

Kapitola 7

Implementace

Tato kapitola se věnuje implementaci výsledné aplikace a jejím detailům. Bude zde rozveden způsob ukládání a zpracování dat potřebných pro korektní běh aplikace, stejně jako realizace grafického uživatelského prostředí. Nebude zde chybět ani popis hlavních funkcí a programových částí zodpovědných za veškerou funkcionalitu aplikace.

7.1 Implementační struktura aplikace

Aplikace je implementována podle návrhového vzoru **MVC** (*Model-View-Controller*), jak je obvyklé u aplikací stojících na frameworcích Ionic a Angular. Implementačními prostředky jsou jazyky moderních webových technologií – pro GUI se využívá prostředků HTML5 a CSS3 a pro všechno ostatní potom dialekt jazyka JavaScript, který nese název **TypeScript** (viz 7.1.1). Struktura aplikace je definována tak, aby co možná nejlépe vyhovovala použitému návrhovému vzoru. Pro každou obrazovku aplikace se v implementační struktuře nachází samostatné soubory pro GUI, které jsou implementované zmíněnými jazyky HTML5 a CSS3 a pak také soubor v TypeScript, který plní roli *controlleru* výsledného GUI, tedy definuje, které věci se mají zobrazit v určitém okamžiku. Na místě *modelu* vystupují potom tzv. *providery*, které jsou prostředkem známým z frameworku Angular, obecně jde o třídu, z které lze vytvořit pouze jednu instanci (tzv. *singleton*) a tuto instanci lze potom injektovat do libovolné jiné třídy, která následně může využívat veškeré atributy a metody injektované instance. *Providery* jsou tím pádem skvělým místem pro back-end logiku a pro práci s databází, neboť je lze následně injektovat do jednotlivých *controllerů* a ovládat tím přímo jednotlivé obrazovky.

7.1.1 TypeScript

TypeScript je v dnešní době hlavním implementačním jazykem pro implementaci aplikací pomocí frameworku Ionic, jeho použití bylo tedy vynuceno využitím tohoto frameworku. Úvodem je však důležité zmínit, že každý program napsaný v JavaScriptu je zároveň i TypeScript programem, jinými slovy TypeScript je **plně kompatibilní** s JavaScriptem [2].

TypeScript přináší do JavaScriptu systém programových modulů, tříd, rozhraní a jak již název napovídá taktéž datové typy. Hlavním cílem tohoto rozšíření JavaScriptu je zlepšit samotný vývoj a udržitelnost aplikací, což je v originálním JavaScriptu obtížné a nepříjemné. Zavedení ortodoxního přístupu k třídám a datovým typům přibližuje tento jazyk k běžným objektově orientovaným jazykům jako jsou Java nebo C#. TypeScript má svůj

vlasní překladač, který překládá zdrojový kód do výsledného JavaScriptu, zde se provádí právě typová kontrola proměnných a objektů jakožto i další úkony syntaktické a sémantické analýzy. Je důležité dodat, že typová kontrola není natolik striktní, aby bránila typickému použití JavaScriptu, může se tedy stát, že zdrojový kód bezpečně projde typovou kontrolou a překladem z TypeScriptu, ale následně spadne na běhové chybě způsobené použitím špatného datového typu.

Ve výsledku TypeScript přidává novou syntaxi pro deklaraci a používání datových typů pro třídní a objektové atributy, proměnné, parametry a návratové hodnoty funkcí a také typovou podporu pro výrazy. Dále přidává podporu pro moduly, třídy, rozhraní a lambda funkce. Všechno je však na závěr přeloženo překladačem zpět do JavaScript. Cílem TypeScriptu není tedy být novým převratným jazykem, ale vyplnit stávající nedostatky JavaScriptu [2].

7.2 Databáze

Dříve nežli bude představena samotná implementace aplikace, je dobré si přiblížit způsob ukládání dat a také jaká data jsou vlastně ukládána. Jak bylo zmíněno v podkapitole 6.1.2, data jsou ukládána na straně uživatele v lokálním úložišti, tím pádem je může uživatel bohužel kdykoliv smazat. Prostředí pro práci s úložištěm je sjednocené frameworkem Ionic, který na pozadí používá na základě cílové platformy buď IndexedDB nebo Web SQL (viz 6.1.2). Při prvním běhu aplikace, přesněji spíše při zjištění, že v úložišti nejsou žádná data, se spouští metoda `entryScript()` z objektu třídy s názvem `DatabaseWrapperProvider`, která jak již název napovídá je *providerem* zodpovědným za práci s úložištěm. Do úložiště jsou ukládány jednotlivé objekty jako dvojice „název - objekt“.

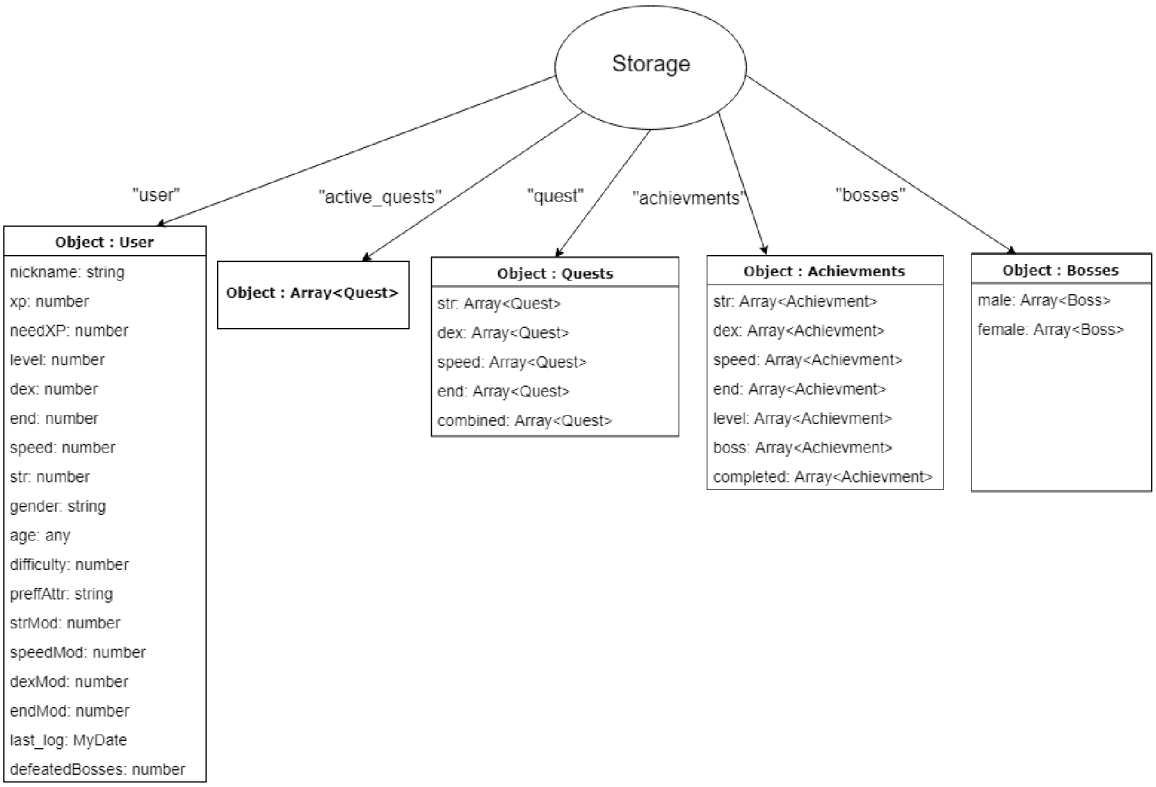
7.2.1 Inicializace databáze

Metoda `entryScript()` obsahuje zapsaná veškerá data potřebná pro běh aplikace, které uloží do úložiště pod příslušné klíče. Jsou zde vytvořeny objekty pro jednotlivé úkoly, uživatele, výzvy, achievementy a všechno ostatní, co je zapotřebí za běhu aplikace ukládat anebo načítat. Metoda je velmi obsáhlá neboť dat a objektů k uložení je mnoho. Cílem této metody je připravit úložiště do takového stavu, aby bylo v aplikaci přímo použitelné a nebylo za potřebí do něho vkládat žádné nové položky, pouze číst a upravovat stávající. Schéma inicializovaného úložiště lze vidět na obrázku 7.1.

7.2.2 Struktura dat

Z obrázku 7.1 je patrné, že se do databáze ukládají objekty se všemi svými atributy vyjma metod. Metody nejsou při ukládání do úložiště zachovány, proto je nutné u objektů, obsahujících metody, při načítání z databáze vytvořit novou instanci dané třídy a předat ji data z načteného objektu, vznikne tak nový objekt s načtenými daty z databáze. Snahou při návrhu ukládání dat bylo zachovat co nejnižší počet uložených dvojic a zároveň jednoduchost načítání žádaných dat z úložiště zpět do aplikace. Z toho důvodu vznikly třídy `Quests`, `Achievments` a `Bosses` a další, všechny tyto třídy mají společný fakt, že drží pole dalších objektů logicky rozdělených dle určitých kategorií. Taktéž obsahují metody, které umožňují snazší vybírání specifických objektů z úložiště. V praxi to potom funguje tak, že se z úložiště načte jeden z těchto obecných objektů – například instance třídy `Quests` a zavolá se na ní metoda pro získání jednoho konkrétního úkolu. Výhodou tohoto přístupu

je fakt, že v úložišti nemusí být uložený vlastní záznam pro každý jeden úkol, ale všechny úkoly jsou uložené pospolu. Nevýhodou však je, že při jakékoliv manipulaci s úkoly je nutné z úložiště načíst celý objekt obsahující všechny úkoly.



Obrázek 7.1: Schéma úložiště – objekty a klíče, pod kterými jsou uložené.

7.2.3 Ukládání a načítání dat

Ukládání a načítání dat má potom na starost tentýž *provider*, k tomu slouží zejména metody `getValue()` a `setValue()` první z nich bere jako parametr klíč, pod kterým má hledat objekt v úložišti. Vytvoří tedy požadavek na úložiště, aby mu byla vrácena položka skrývající se pod klíčem, který byl předán parametrem. Celá tato akce je asynchronní, interpret tedy nečeká na dokončení operace a pokračuje ve vykonávání zdrojového kódu. Operace je realizována za pomocí TypeScript objektu označovaného jako *promise*, což je jeden z více způsobů jak se vypořádat s asynchronními operacemi. *Promise* je objekt, který čeká na vykonání dané operace a na základě úspěchu či neúspěchu dané operace zavolá příslušnou funkci. Obdobně se služeb *promise* využívá i při zapisování hodnot do úložiště.

7.3 Back-end logika

Při spuštění aplikace jsou veškerá data načtena z úložiště do objektu třídy `GlobalVarsProvider`, který drží aktuální stav aplikace. V atributech tohoto objektu jsou uloženy téměř všechny informace potřebné pro korektní běh a zobrazování veškerých informací v aplikaci. Například se zde nachází uložený objekt aktuální uživatelské postavy, její atributy, úroveň a tak podobně. Dále tento objekt obsahuje mnoho metod pro distribuci těchto informací do všech

částí aplikace stejně jako pro aktualizaci těchto informací na základě různých akcí, které uživatel vykonává v aplikaci. Tento *provider* je tedy injektovaný do téměř každé části aplikace. Předává jednotlivým *controllerům* informace, které je nutné zobrazit v daném UI a naopak ukládá změny těchto informací a zároveň je propaguje zpět do databáze tak, aby bylo zaručeno, že se žádná data neztratí při vypnutí aplikace v kterémkoliv bodě jejího běhu.

7.3.1 Komunikace mezi třídami

Během implementace vyvstala otázka, jak zajistit, aby všechny části aplikace měly vždy aktuální informaci z `GlobalVarsProvider`. Bylo tedy nutné najít způsob, jak donutit jednotlivé *controllery*, aby si aktualizovaly své atributy v případě, že se některé z nich staly neaktuálními. Této funkcionality bylo dosaženo využitím třídy původně z frameworku Angular, která se nazývá *Events*. Podobně jako u *providerů* lze vytvořit pouze jednu instanci této třídy, která je pak předávána do všech objektů, v jejichž konstruktoru se nachází její deklarace. Třída *Events* nabízí možnost, jak se přihlásit k odběru událostí se specifickým názvem a přiřadit k této události příslušnou operaci. Zároveň lze i publikovat takové události a dokonce k nim přidat nějakou informaci (například proměnnou, která nese aktuální hodnotu atributu). V praxi to tedy potom funguje tak, že při změně nějaké informace v `GlobalVarsProvider` dojde k publikování události, která říká, že se tento atribut změnil. Zároveň se k této události přidává aktuální hodnota změněného atributu. Všechny ostatní objekty, které jsou přihlášeny k odběru události s tímto názvem jsou ihned notifikovány a následně jsou zavolány jejich přiřazené operace. Tímto způsobem je zajištěno, že veškerá zobrazovaná data jsou aktuální a totožná s těmi, které jsou uložené v `GlobalVarsProvider`.

7.4 Grafické uživatelské rozhraní

Ve frameworku Angular na kterém stojí použitý framework Ionic, je každá obrazovka tzv. komponentou, jinými slovy obsahuje definici samotného grafického prostředí (v HTML/CSS) a pak také zmiňovaný *controller*. Komponenty lze znovupoužívat, vkládat je do jiných komponent a tak podobně. Vzhled každé komponenty aplikace je tedy defninován v HTML a CSS souboru, data k zobrazení, animace a komunikace uživatelského rozhraní s aplikací je potom řízena právě oním *controllerem* každé komponenty. Každý *controller* je definovaný svou vlastní třídou, jejíž instance je vytvořena při načítání obrazovky, na které komponenta vystupuje. Téměř každý *controller* si ve svém konstruktoru vyžádá data z `GlobalVarsProvider` a zároveň se přihlásí k odběru událostí týkajících se těchto dat.

7.4.1 Zobrazení dat v uživatelském rozhraní

Zobrazování dat z *controlleru* do uživatelského prostředí zajišťuje tzv. *data-binding*. Ten umožňuje pomocí speciální HTML direktivy svázat proměnnou nacházející se v *controlleru* dané komponenty s uživatelským rozhraním a *data-binding* následně zajistí, aby byla tato proměnná zobrazena na místo právě této HTML direktivy. Ve frameworku Angular, jehož služeb využívá framework Ionic v této záležitosti, se pro to využívá následujícího zápisu „{{`název-proměnné`}}“. Důležité je zmínit fakt, že zobrazení dat z *controlleru* do výsledného uživatelského rozhraní probíhá automaticky a v každém okamžiku je zaručeno, že se zobrazují aktuální data, která jsou právě uložena v *controlleru*.

7.4.2 Navigace mezi obrazovkami

Framework Ionic řeší navigaci mezi obrazovkami jednoduchým způsobem – implementuje zásobník jednotlivých obrazovek označovaných jako *views*, na který jsou postupně vkládány a odebírány jednotlivé obrazovky. Aktuálně zobrazená obrazovka je vždy na vrcholu zásobníku, tj. byla vložena jako poslední. Mezi obrazovkami lze tímto způsobem plynule přecházet, pouze se na zásobník vkládají nové, případně odebírají stávající. Jak již je pro zásobník typické, vkládá a odebírá se vždy vrchol zásobníku. *Views* uložené na zásobníku není potřeba znovu vytvářet, na druhou stranu, pokud se vkládá nové *view* na zásobník, je nutné ho nejprve vytvořit tzn. zavolat jeho konstruktor a nechat vyrenderovat HTML stránku. V praxi to funguje tak, že je zde vždy jedna kořenová obrazovka, ve vyvíjené aplikaci je to obrazovka profilu postavy, která tvoří dno zásobníku. Následně jsou na zásobník postupně vkládány další obrazovky, které uživatel otevírá a odebírány ty, které opět zavírá případně se z nich vrací zpět.

7.5 Hlavní obrazovky aplikace

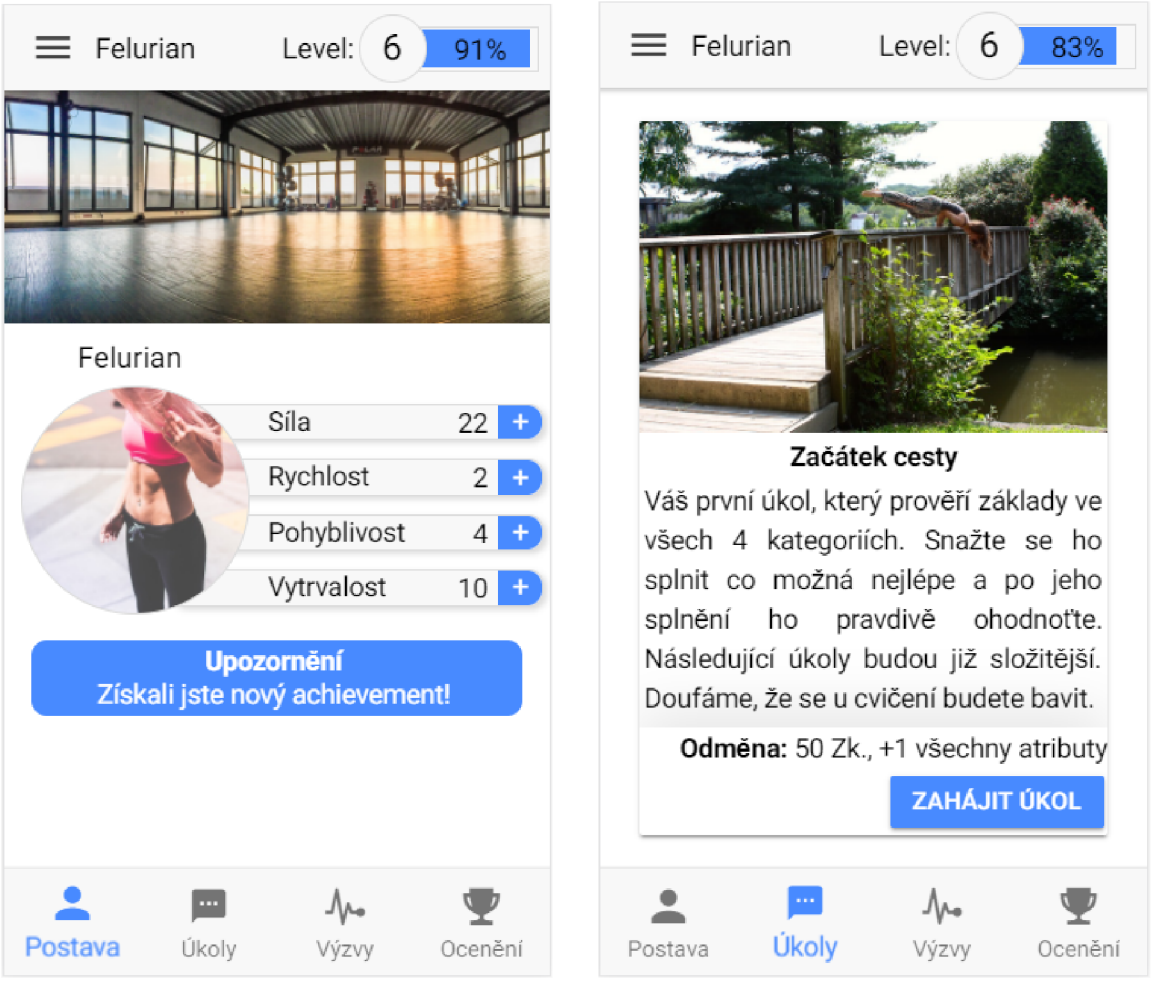
Tato podkapitola se bude věnovat hlavním obrazovkám aplikace, co nabízí a jakým způsobem jsou implementované. Bude zde rozveden způsob komunikace mezi uživatelským rozhraním, back-end logikou a databází. Taktéž zde budou k nahlédnutí screenshoty z implementované aplikace.

7.5.1 Obrazovka postavy

Hlavní obrazovka postavy je k vidění na obrázku 7.2 vlevo, tvoří ji v hlavičce horizontální lišta, která je v této podobně společná pro většinu obrazovek v aplikaci. Vpravo v této liště je zasazena vytvořená komponenta, která se nazývá *progress-bar*. Slouží pro zobrazování úrovně a získaných zkušeností uživatelské postavy. Co se týče implementace této komponenty, při vytvoření je jí do konstruktoru předána z `GlobalVarsProvider` aktuální úroveň postavy a procento získaných zkušeností. Úroveň je přímo navázána pomocí *databindingu* na uživatelské rozhraní a je tím pádem přímo zobrazena. Se zkušenostmi je to trochu složitější, zde se proměnná váže na CSS atribut šířky vnitřní části komponenty (na obrázku zobrazený modře), což elegantně zobrazí procentuální postup do další úrovně. Mimo *progress-baru* se ve vrchní liště nachází ještě jméno postavy a tlačítko hlavního menu.

Pod vrchní lištou se již nachází samotný obsah obrazovky postavy. Obrazovce postavy dominuje tematický obrázek, pod nímž se nachází další vytvořená komponenta, která nese název *character-head*. Kromě jména postavy zobrazuje profilový obrázek a hlavní atributy postavy. Jméno a atributy postavy jsou komponentě opět předané při konstrukci objektu z `GlobalVarsProvider` a nechybí zde ani přihlášení k odběru událostí, aby byla zajištěna aktuálnost atributů. Kromě toho, obsahuje komponenta ještě tlačítka na zvyšování příslušných atributů postavy. Kliknutí na jedno z těchto tlačítek má za následek navigaci na obrazovku tréninku daného atributu a to vložením obrazovky tréninku na zásobník obrazovek.

Jako tlačítko slouží též obrázek postavy, který lze kdykoliv změnit. Toho je docílené pomocí HTML tagu *input* typu soubor, který inicializuje formulář na výběr libovolného souboru ze souborového systému uživatele. Vybraný obrázek se následně v bitové formě uloží do databáze tak, aby se při každém dalším spuštění aplikace mohl správně načíst. Pod celou touto komponentou se nachází ještě jedna a to komponenta starající se o zobrazení upozornění, její implementace je poněkud jednoduchá. Komponenta pouze svazuje



Obrázek 7.2: Zleva: Obrazovka postavy, obrazovka úkolů.

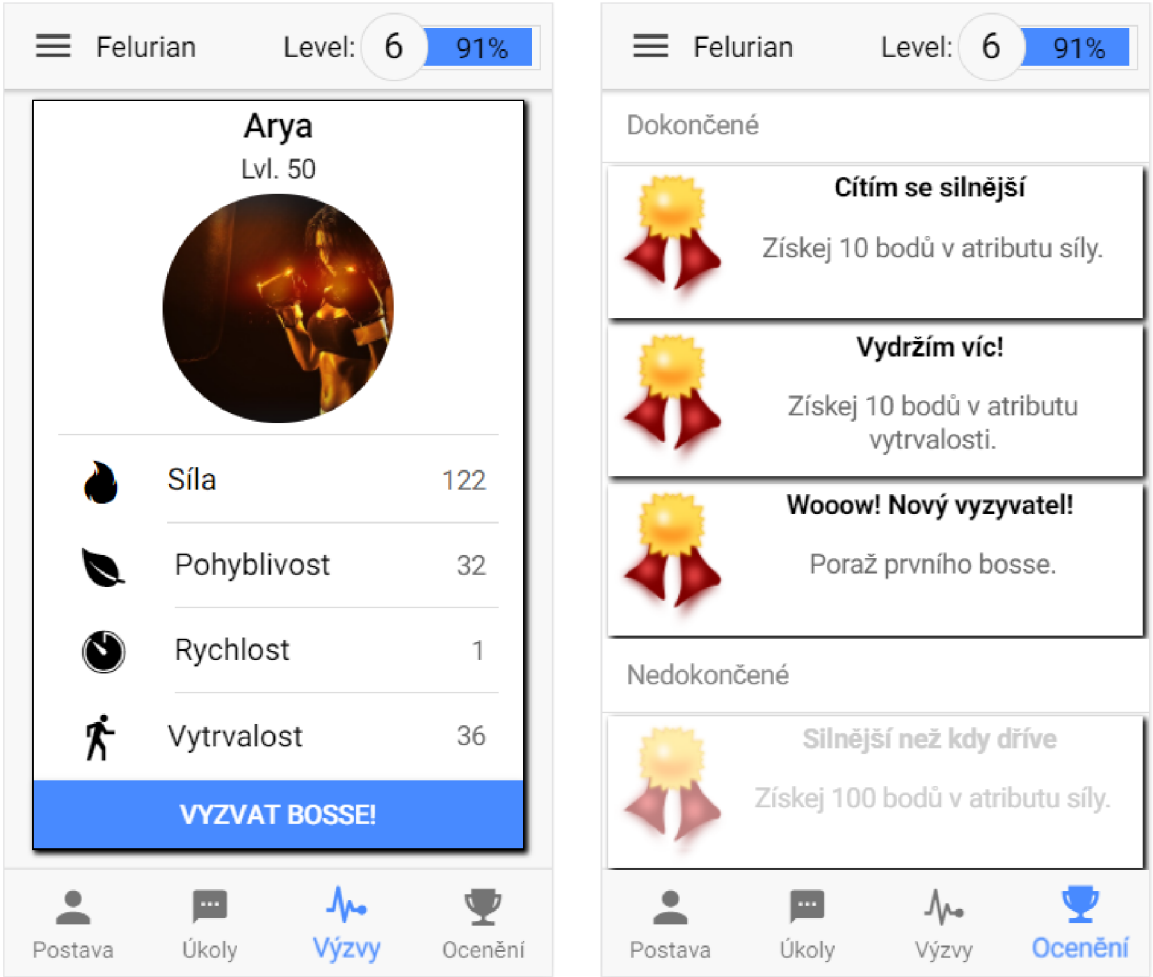
své instanční proměnné, které představují jednotlivá upozornění s uživatelským rozhraním a taktéž přihlašuje odběr události týkajících se těchto upozornění, aby byly automaticky změněny na aktuální podobu. Pod touto komponentou se nachází již pouze spodní horizontální lišta, která je na mobilních zařízení hlavním navigačním nástrojem aplikace.

7.5.2 Obrazovka úkolů

Na obrazovce úkolů (obrázek 7.2 vpravo) lze vidět, stejně jako v případě obrazovky postavy, vrchní a spodní lištu. Hlavním obsahem této obrazovky je seznam obsahující až 5 úkolů z kterých může uživatel vybírat. Seznam je implementovaný pomocí speciálního HTML tagu, který s sebou přináší Ionic framework, `ion-list`. Tento seznam je plně responzivní a velmi snadno upravitelný, vkládají se do něj jednotlivé položky pomocí tagu `ion-item`. Položkou seznamu může být cokoliv, v případě implementované aplikace je to tematický obrázek úkolu, název úkolu, jeho popis, odměna a konečně tlačítko pro zahájení úkolu. Data k zobrazení jsou vyžádána v konstruktoru z `GlobalVarsProvider` a jsou uložena v instanční proměnné. Zejména se jedná o pole aktivních úkolů a čas do získání dalšího úkolu. Jednotlivé úkoly jsou opět za pomocí *data-bindingu* navázány na HTML dokument a výsledné grafické rozhraní, nicméně tentokrát je využita HTML direktiva frameworku

Angular, která se označuje jako **ngFor* a předchází duplicitám v HTML dokumentu. Jak již název direktivy napovídá, jedná se o for cyklus, ve kterém pro každý prvek svázané proměnné z *controlleru* vytvoří úsek HTML dokumentu, který je uveden uvnitř tagu s touto direktivou. Vázanou proměnnou v této direktivě je právě pole aktivních úkolů a tedy pro každý úkol je vytvořen nový `ion-item` tag s celým svým obsahem, aniž by bylo nutné každý úkol manuálně definovat v HTML dokumentu.

Kontrola, zda již neuplynul časový limit pro přidělení dalšího úkolu, probíhá vždy při vstupu na tuto obrazovku, v TypeScript je za to zodpovědná funkce `ionViewDidEnter`, která je spuštěna vždy při vstupu na obrazovku, nehledě na to, zda byla obrazovka načtena ze zásobníku, nebo byla nově zkonstruována. Tlačítko pro zahájení úkolu má podobnou funkcionalitu jako výše popsané tlačítko pro zvýšení konkrétního atributu na obrazovce postavy, vytvoří novou obrazovku tréninku, zároveň dojde i k předání informací o úkolu z aktuální obrazovky a vložení nové obrazovky na zásobník.



Obrázek 7.3: Zleva: Obrazovka výzev, obrazovka ocenění.

7.5.3 Obrazovka výzev

Tato obrazovka (obrázek 7.3 vlevo) zobrazuje jednotlivé bosse, které má uživatel za úkol porazit. Implementačně na ní není příliš mnoho zajímavého, data o jednotlivých bossech

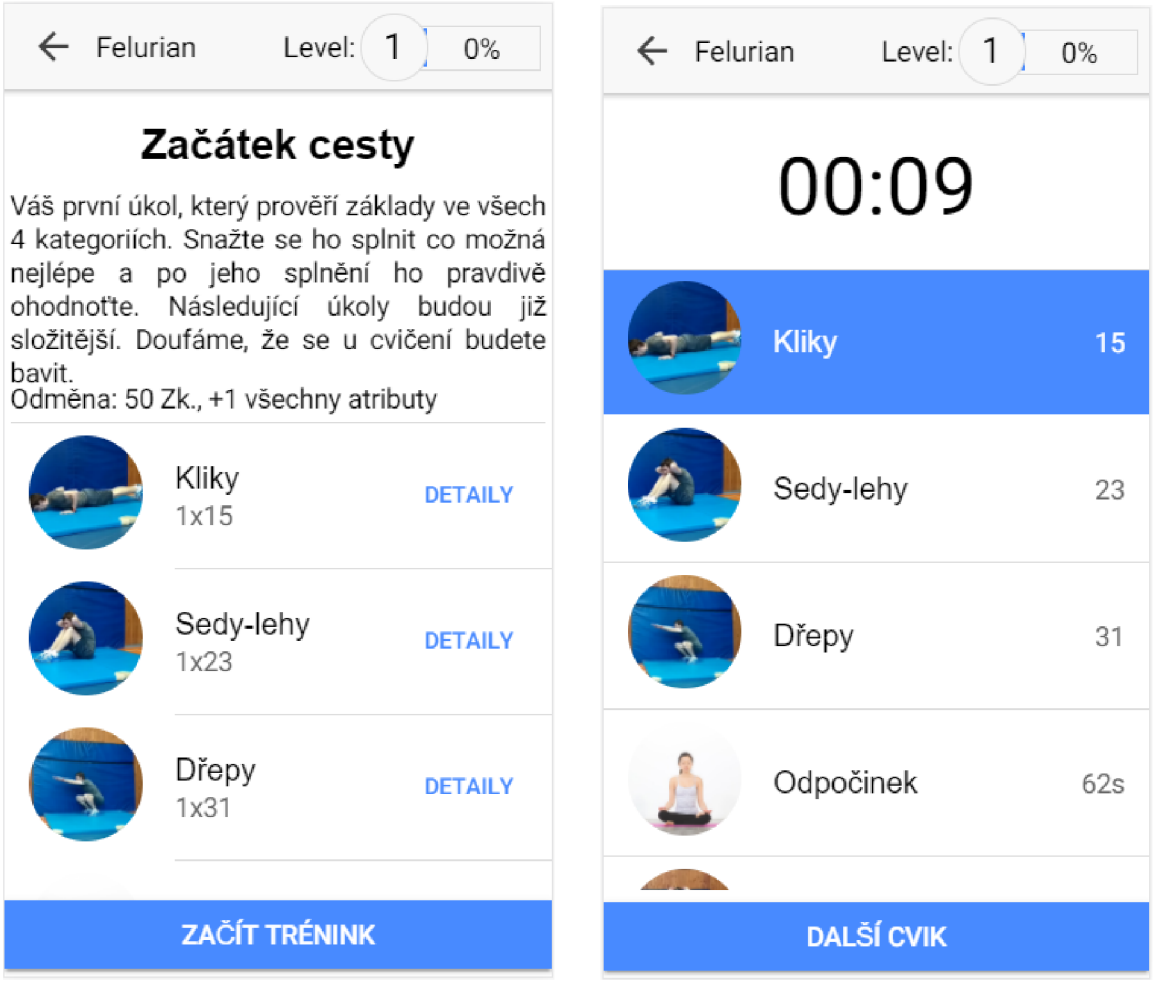
jsou opět předány do konstruktoru obrazovky z `GlobalVarsProvider`, pro tyto data je tu připravena instanční proměnná třídy `Boss`, která obsahuje vlastní proměnné – cestu k obrázku, jméno, úroveň, jednotlivé atributy a také úkol, který je nutný splnit k poražení bosse. Jednotlivé atributy této instanční proměnné jsou navázány na HTML dokument pomocí *data-bindingu*. Ve spodní části obrazovky se nachází tlačítko na vyzvání onoho bosse. Kliknutí na toto tlačítko má za následek opět zkonstruování nové obrazovky tréninku a její vložení na zásobík. Samotná obrazovka tréninku a jeho průběh bude popsán v podkapitole 7.5.5.

7.5.4 Obrazovka ocenění

Obrazovka ocenění (obrázek 7.3 vpravo) slouží k zobrazování dokončených, ale i nedokončených ocenění. Jednotlivá ocenění se nenačítají z `GlobalVarsProvider` jak je v implementované aplikaci obvyklé, ale přímo z úložiště při konstrukci této obrazovky a to hlavně proto, aby nebylo nutné udržovat v paměti všechny ocenění a ještě je tam aktualizovat, pokud uživatel některé z nich získá. Toto probíhá přímo v úložišti, pokud uživatel provede nějakou akci, která by mohla potenciálně mít za následek získání ocenění, je zavolána funkce z *provideru* `GlobalVarsProvider` s názvem `checkAchievment()`, které se předává kategorie ocenění a hodnota, která se následně porovnává s podmínkou prvního ocenění v této kategorii (ocenění jsou ve svých kategoriích seřazeny od nejjednodušších po nejsložitější, proto stačí porovnávat hodnotu pouze s prvním ocenením). Pokud je hodnota vyšší než podmínka prvního ocenění, znamená to, že uživatel dosáhl tohoto ocenění a následně je toto dosažené ocenění přesunuto do kategorie splněných ocenění a je publikována událost, která říká, že se stav ocenění změnil. Na tuto událost reaguje obrazovka ocenění a vyžádá si z úložiště aktuální stav všech ocenění a aktualizuje jím své instanční proměnné. Instančními proměnnými jsou pole splněných a nesplněných ocenění, která jsou navázána pomocí *data-bindingu* na HTML dokument, kde se pomocí HTML direktivy **ngFor* jednotlivá ocenění vygenerují.

7.5.5 Trénink a jeho průběh

Poslední obrazovky, kterých se zde dotkneme, se týkají samotného tréninku (obrázek 7.4 vlevo) a jeho průběhu (obrázek 7.4 vpravo). Na obrazovku tréninku vede jakékoliv zahájení úkolu, zvýšení jednoho z atributů anebo vyzvání bosse. Při navigaci na tuto obrazovku je vždy předáván objekt úkolu (instance třídy `Quest`), který krom jiného obsahuje pole se seřazenými cviky, které bude uživatel následně vykonávat. V konstruktoru této obrazovky je však nejprve nutné vytvořit hlubokou kopii celého pole cviků, neboť bude v průběhu tréninku modifikováno a je nežádoucí, aby tyto modifikace měly vliv na ostatní části aplikace, kde tento objekt úkolu vystupuje. Následně je potřeba vytvořit souhrn jednotlivých cviků, aby si uživatel mohl prohlédnout co ho čeká. K tomu slouží metoda `sumarrizeExercises()` vlastní třídy této obrazovky. Tato metoda prochází zmiňované pole cviků a podle ID jednotlivých cviků sčítá jejich série a ukládá je do nového pole, které je následně navázáno na HTML dokument a vykresleno pomocí **ngFor*. Výsledkem této metody je tedy nové pole, ve kterém se nachází každý cvik jen jednou a je k němu přiřazena informace o celkovém počtu sérií a o počtu opakování v každé sérii. Díky tomu má uživatel přehled o tom, kolik opakování celkem musí zvládnout. Z obrazovky tréninku lze taktéž přejít na obrazovku detailu jednotlivých cviků a to kliknutím na tlačítko „Detaily“. Tato obrazovka obsahuje podrobný popis cviku, nahrávku správného provedení cviku a seznam bodů, na co se zaměřit při provádění onoho cviku.



Obrázek 7.4: Zleva: Obrazovka tréninku, obrazovka průběhu tréninku.

Pomocí tlačítka „Začít trénink“ je provedena navigace na obrazovku průběhu tréninku, do které se opět při navigaci předává celý objekt úkolu. Z tohoto objektu je zde využito pouze pole cviků, které je zde zobrazeno jako seznam. Seznam cviků je implementovaný podobně jako popsaný seznam úkolů v kapitole 7.5.2, opět je pro jeho vytvoření použito HTML tagů `ion-list` a `ion-item` v kombinaci s **ngFor*. Aktuálně prováděný cvik, tj. první v onom seznamu je podbarvený modře, klikáním na tlačítko ve spodní části obrazovky se přechází na další cvik a to tím způsobem, že je aktuálně první cvik ze seznamu odebrán a jeho místo zaujme cvik následující. Z tohoto důvodu bylo nutné v předchozí obrazovce vytvořit hlubokou kopii tohoto pole, neboť hrozilo, že při přerušení průběhu cviku a navrácení se zpět v aplikaci, bude odcvičená část cviků nenávratně ztracena. Posledním nepopsaným elementem této obrazovky je potom časový čítač nacházející se ve vrchní části obrazovky. Tento čítač je implementovaný za pomocí objektu třídy *TimerObservable*, který v zadaném intervalu vyvolává událost. K odběru této události je možné se samozřejmě přihlásit, čehož je využito právě u tohoto čítače. Při každém zaregistrování události od tohoto objektu proběhne iterace proměnné symbolizující počet uplynulých sekund. Po uplynutí šedesáti sekund, je tato proměnná opět nastavena na nulu a zároveň je zvýšena proměnná, symbolizující počet uplynulých minut, o jedna.

Kapitola 8

Testování

V průběhu návrhu a hlavně implementace byla aplikace testována s cílem odhalit bugy a chyby, které mohly během implementace vzniknout. Taktéž byly poskytovány obrázky z vývoje některých obrazovek skupině potenciálních uživatelů a dle jejich hodnocení a názorů bylo uživatelské prostředí upraveno, aby vyhovovalo co možná nejvíce uživatelům. Po dokončení samotné implementace byla aplikace testována skupinou uživatelů s cílem zjistit jejich názor na aplikaci jako celek. Na základě výsledků testování a zpětné vazby od uživatelů byly stanoveny rozšíření a vylepšení aplikace.

8.1 Průběh testování

Testování probíhalo na webové verzi aplikace, která byla umístěna na školním serveru a byla tedy pod konkrétní URL adresou přístupná v síti internet. Pro účely testování byl využit software[1], který měří a počítá různé statistiky a také sleduje pohyby kurzoru jednotlivých uživatelů této aplikace a dokáže je následně přehrát, což je obzvlášť důležité pro testování uživatelského prostředí. Skupině uživatelů byly zadány tři úkoly, které měly otestovat jednotlivé části aplikace, její použitelnost a přehlednost. Počínání uživatelů bylo monitorováno právě přes zmiňovaný software. Po dokončení těchto úkolů bylo uživatelům umožněno podat zpětnou vazbu písemnou formou, kde dostali prostor ke shrnutí celého testování, ohodnocení aplikace a k vyjádření svého názoru na testovaný produkt. Uživatelé prováděli testování ze svých domovů za podmínek, které jim co možná nejlépe vyhovovaly a ve svém volném čase. Uživatelům nebyly podány žádné další informace ani vysvětlení, byl jim předložen pouze smysl a cíl aplikace. Účelem tohoto přístupu bylo napodobit reálné použití aplikace, kterou budou uživatelé také používat za podmínek jim pohodlných a ve svém volném čase.

8.1.1 Vytvoření postavy a seznámení se s aplikací

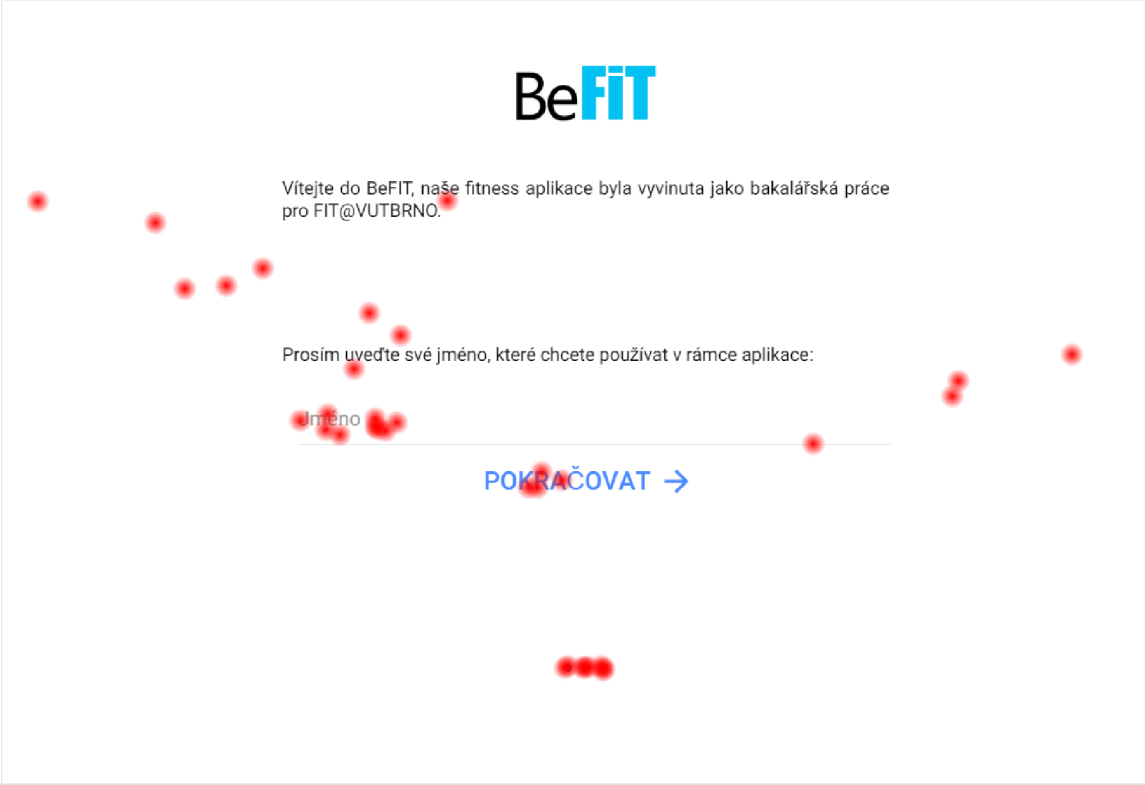
První úkol byl zaměřený hlavně na seznámení se s aplikací, jak ostatně vyplývá z jeho názvu. Uživatel měl projít úvodním intrem, vyplnit příslušné informace a spustit samotnou aplikaci. Dále měl projít každou z hlavních obrazovek a nakonec změnit obrázek své postavy. Tento úkol měl za cíl přiblížit uživateli ovládání a navigaci v aplikaci a zároveň otestovat uživatelské prostředí úvodního intra, které je navržené spíše pro mobilní platformy.

[1] https://metrica.yandex.com/about?

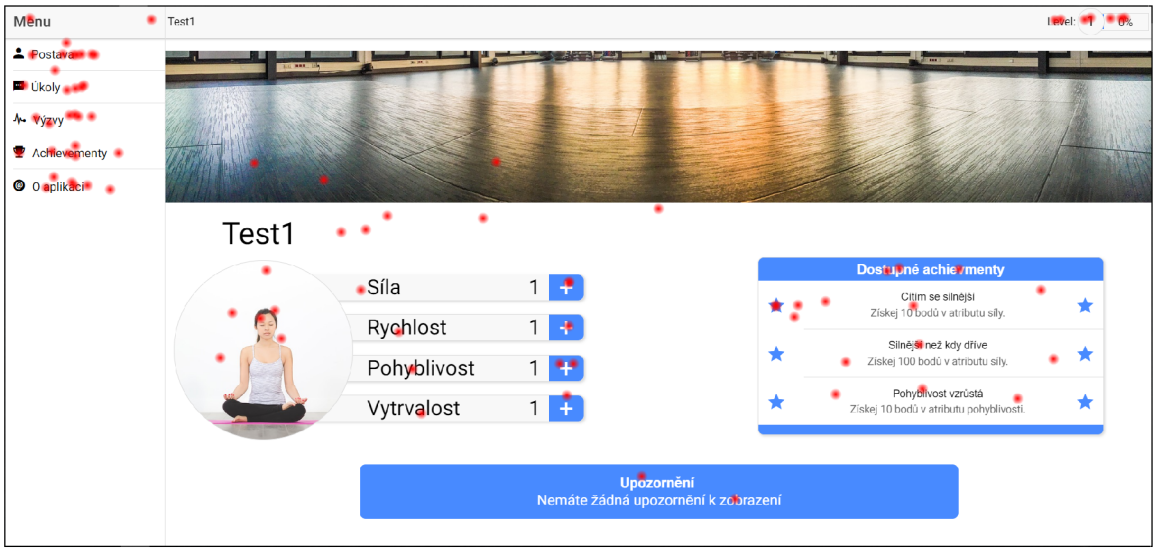
Z následujících obrázků (obrázek 8.1, 8.2) je patrné, že se uživatelé zhostili tohoto úkolu vícenež dobře. Z obrázku 8.1 zachycujícího jednotlivé kliky všech uživatelů na obrazovce intra je patrné, že pro navigaci používali hlavně tlačítko. Méně potom přímý přechod mezi slajdy v podobě tří bodů ve spodní části obrazovky a téměř minimálně přechod stisknutím a potáhnutím doprava, respektive doleva, který je typický spíše pro dotyková zařízení. Dále si lze povšimnout, že uživatelé při zadávání jména klikali výhradně na tu část textového vstupu, kde se nacházel jeho název, přestože lze kliknout kamkoliv jinam na tento element.

Na obrázku 8.2 jsou potom zachyceny všechny kliknutí uživatelů na hlavní obrazovce aplikace. Nejvíce kliknutí se vyskytuje v části hlavního menu, což je logické vzhledem k zadání tohoto úkolu. Zajímavější jsou pokusy o kliknutí na level a loading bar na vrchní liště obrazovky. Lze předpokládat, že uživatelé očekávali nějakou odezvu na tyto kliknutí, v aplikaci však žádná odezva u těchto elementů implementována není. Podobná situace je u boxu s dostupnými achievementy, zde lze předpokládat, že uživatelé očekávali přechod na obrazovku achievementů.

První úkol proběhl poměrně úspěšně, podařilo se otestovat prvotní kontakt uživatelů s aplikací a jak je vidět z uvedených obrázků, uživatelé příliš nebloudili. Je však důležité počítat i s nedostatky, které uživatelé odkryli v tomto úkolu. Jedná se o zmiňovaný level a progress bar, který by zřejmě měl po kliknutí zobrazit aktuální počet zkušeností a potřebný počet k dosažení dalšího levelu, a box s achievementy, které by měly umožnit prokliknutí na obrazovku achievementů.



Obrázek 8.1: Intro - mapa kliknutí



Obrázek 8.2: Obrazovka postavy - mapa kliknutí

8.1.2 Dokončení prvního úkolu v aplikaci

Druhý úkol byl zaměřený na hlavní funkcionalitu aplikace. Zde si uživatelé měli vyzkoušet spuštění úkolu a jeho průběh. Uživatel měl tedy za úkol přejít na obrazovku úkolů, spustit první úkol, projít tréninkem, zdárně ho dokončit a převzít odměnu. Úkol si kladl za cíl otestovat uživatelské prostředí úkolů a průběhu tréninku.

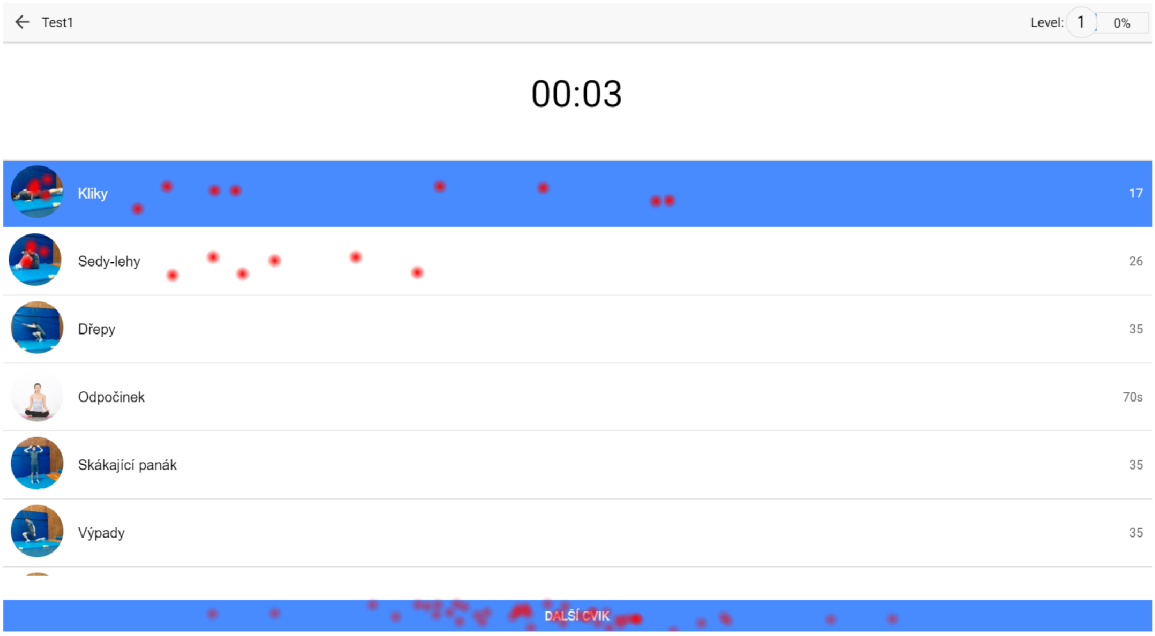
Na obrázku 8.3 je k vidění mapa všech kliknutí na obrazovce průběhu tréninku. Lze vidět, že na konec všichni uživatelé pochopili, že přechod na další cvik se provádí tlačítkem ve spodní části aplikace, nicméně evidentně tomu předcházelo několik zmatených kliků na jednotlivé cviky. Někteří uživatelé dokonce klikali na obrázek cviku, je možné, že předpokládali, že tím otevřou detail cviku, kde se dozví jak tento cvik provádět. Bohužel takto to v aplikaci není implementováno, detaily cviku se dají prohlížet pouze před zahájením samotného tréninku a to v obrazovce tréninku.

I druhý úkol lze hodnotit jako úspěšný, uživatelé dosáhli stanoveného cíle bez větších obtíží. Lehké nedostatky, které tento test odhalil je třeba vzít v úvahu v návrhu dalšího vývoje aplikace.

8.1.3 Výzva bosse a detaily jednotlivých cviků

V posledním úkolu si měli uživatelé vyzkoušet výzvu bosse včetně všech nutných kroků. Uživatelům bylo obzvláště zdůrazněno, aby se podívali na detaily jednotlivých cviků. Uživatelé měli tedy za úkol přejít na obrazovku výzev, vyzvat bosse, postupně si projít detaily jednotlivých cviků a následně je provést při samotném tréninku. Po tomto výkonu byli aplikací instruováni, aby zadali své výsledky do formuláře, který je porovná s výsledky vyzvaného bosse. Tento poslední úkol měl uživatele dostat do částí aplikací, které v předešlých úkolech ještě neměli šanci navštívit.

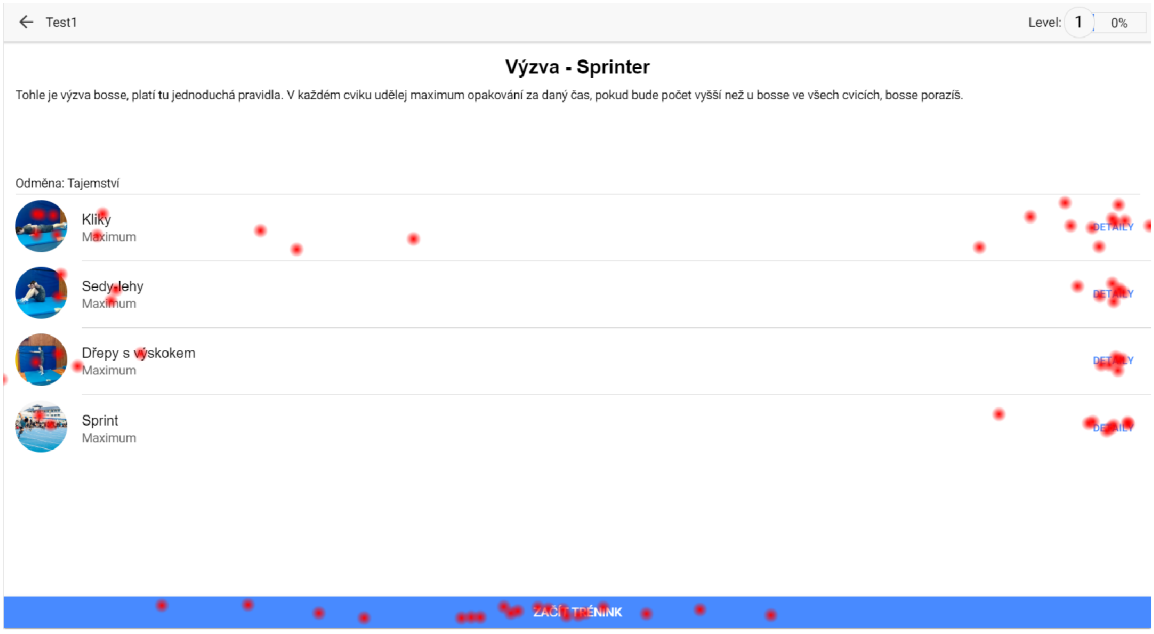
Na obrázku 8.4 lze vidět výsledek testování na obrazovce, následující po hlavní obrazovce výzev. Uživatelské prostředí je zde stejné jako v případě klasických úkolů, jedním z nich již uživatelé prošli v předchozím úkolu. Tato zkušenost, se zde velmi výrazně projevila, není tu takový počet zmatených kliknutí, většina kliknutí směřuje tam, kam má. Tento fakt vrhá



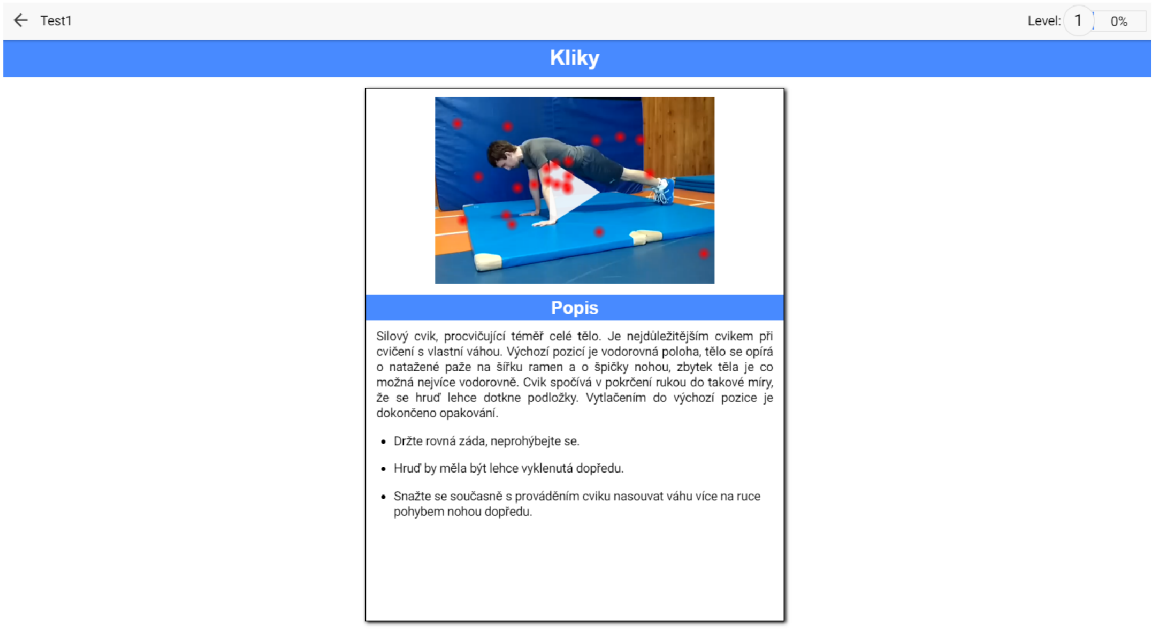
Obrázek 8.3: Průběh tréninku - mapa kliknutí

pozitivní světlo na rozhodnutí ponechat uživatelské prostředí jednotlivých výzev totožné jako u úkolů. Na obrázku 8.5 je zobrazen výsledek testování detailu cviku, který nemohl dopadnout lépe, všichni uživatelé plně pochopili, že jediný aktivní element na obrazovce je videoukázka cviku, kromě kliknutí na tento element zde tedy nenajdeme žádná jiná.

Třetí a poslední úkol neodhalil již žádné nedostatky, uživatelé využili zkušeností získaných v předchozích dvou úkolech a velmi dobře zvládli i tento poslední úkol.



Obrázek 8.4: Výzva - mapa kliknutí



Obrázek 8.5: Detail cviku - mapa kliknutí

8.2 Zpětná vazba

Po dokončení testování dostali uživatelé prostor vyjádřit svůj názor na testovanou aplikaci. Jak tomu obvykle bývá, ne každý uživatel byl ochotný podat zpětnou vazbu, ale i přesto se sešlo relevantní množství zajímavých názorů. Zpětnou vazbu od uživatelů lze volně rozdělit na takové tři skupiny, v první z nich se uživatelé zaměřili na hodnocení uživatelského prostředí, v druhé potom na funkcionalitu a to, co jim zde chybí a třetí skupina hodnotila aplikaci tak, jak ji testovala. Tedy texty uživatelů ze třetí skupiny popisovaly převážně kam klikali a co očekávali, že se stane. Na základě této zpětné vazby budou v podkapitole 8.3 stanoveny možné rozšíření implementované aplikace.

8.2.1 Uživatelské prostředí

Zde se většina názorů shodovala v tom, že uživatelské prostředí aplikace je intuitivní, jednoduché a funkční. Někteří uživatelé aplikaci vytýkali zvolenou barevnou kombinaci, dle jejich názorů, je místy až příliš bílého prostoru a není zde možnost si uživatelské prostředí barevně jakkoliv přizpůsobit. Uživatelé taktéž ocenili možnost výběru vlastního obrázku postavy, na druhou stranu se některým uživatelům nelíbil úvodní obrázek, který se nachází na obrazovce postavy nahoře, podle těchto názorů by měl být obrázek buď plně přizpůsobitelný uživatelem, obdobně jako obrázek postavy anebo by aplikace měla nabízet výběr z několika úvodních obrázků.

8.2.2 Obsah a funkcionalita

Zde se jednotlivé názory poněkud rozcházely, některým uživatelům přišly úkoly v aplikaci zbytečně repetitivní, s čímž se nedá nesouhlasit, neboť aplikace aktuálně obsahuje databázi 22 cviků, což není dostatečné množství k vytvoření většího množství rozdílných úkolů, to by se však mělo do budoucna změnit. Na druhou stranu uživatelům se velmi líbil koncept souboje s bossem, kde museli nejprve vykonat maximální množství opakování jednotlivých cviků a následně svůj výkon porovnat s bossem aniž by dopředu věděli, jakých hodnot musí dosáhnout. Někteří uživatelé dokonce uvedli, že by rádi podobnou funkci viděli v multiplayerovém provedení, ve kterém by na stejném principu fungoval souboj mezi dvěma uživateli. Jediná věc, ve které se názory uživatelů téměř ve všech případech shodovaly, byla absence jakéhokoliv porovnání s jinými uživateli.

8.2.3 Testování

Zde uživatelé vyjadřovali své názory k absolvovanému testování, pokusili se vysvětlit co od aplikace očekávali a kde byli na druhou stranu překvapeni jejím chováním. Z velké části zde uživatelé psali o nedostatcích, které již odkrylo testování – v podkapitole 8.1.1 nebylo ještě úplně jasné, co uživatelé očekávali od progress baru, na který opakovaně klikali. Zde však někteří uživatelé situaci vysvětlili. Skutečně očekávali, že jim bude nějakým způsobem podána informace kolik zkušeností mají a kolik ještě potřebují. Jsou toho názoru, že je nedostačující zobrazovat postup pouhými procenty. Co se týče nedostatku, který se projevil v druhém úkolu popsaném v podkapitole 8.1.2, uživatelé vysvětlili, že se snažili otevřít detail cviku přímo z průběhu tréninku. Nicméně většina názorů se v tomto nedostatku shoduje v tom, že stačí si jednotlivé cviky projít před samotným tréninkem a pouze zkoušeli, zda se zde objeví nějaká reakce aplikace.

8.3 Možnosti vývoje aplikace

Rozšíření popsána v této podkapitole vznikla na základě předešlého testování. Z téměř každého názoru jednotlivých uživatelů byly vybrány ty nejvhodnější myšlenky a nápady pro budoucí vývoj a vylepšení implementované aplikace. Tato rozšíření se budou týkat převážně uživatelského prostředí, funkcionality a obsahu aplikace.

8.3.1 Uživatelské prostředí

Testování prokázalo, že uživatelské prostředí je navrženo a implementováno vhodným způsobem, a proto se rozšíření v tomto ohledu zaměřuje na nápravu nedostatků, popsaných v sekci 8.2.1. Dále by bylo možné implementovat změnu barev uživatelského prostředí formou nějaké palety předvolených barev, ze které by si uživatelé mohli vybrat barvu, která by jim co možná nejlépe vyhovovala a aplikace by se pak přebarvila do odstínů této barvy. Samozřejmě, pokud by došlo k přidání nového obsahu a funkcionality, muselo by se uživatelské prostředí vhodně rozšířit, aby pokrylo všechny funkce aplikace.

8.3.2 Funkcionalita a obsah

Uživateli nejvíce vytýkaný nedostatek aplikace se týkal možnosti soutěžení a socializace s jinými uživateli, z tohoto důvodu bylo navrženo rozšíření, ve kterém by spolu uživatelé mohli soutěžit podobným způsobem, jako to nyní funguje při souboji s bossem. Toho by se dalo docílit zavedením nové obrazovky s názvem „Aréna“, kde by proběhlo vyhledání protivníka, následně by byl oboum uživatelům zadán stejný úkol, na jehož konci by museli oba zadat svůj výkon a ten lepší by zvítězil. Tato funkcionalita by ovšem již potřebovala podporu serveru, se kterou by aplikace komunikovala, jednalo by se tedy o znatelný zásah do aktuální implementace. Pokud by ovšem došlo k implementaci *client-server* přístupu, nebyl by problém zavést i další funkcionalitu, jako jsou globální žebříčky uživatelů seřazených například podle úrovně anebo systém týmů a přátel, kde by se uživatelé mohli sdružovat do týmů a být v kontaktu s jinými uživateli. Vrcholem tohoto rozšíření by potom byly tréninky pro více uživatelů současně, čímž by trénování dostalo úplně nový rozměr.

Co se týče obsahu, zde by bylo vyhodné rozšířit databázi cviků a navrhnout nové úkoly a s tím související tréninky, aby byla zajištěna větší rozmanitost a tím pádem i dosah celé aplikace.

8.4 Vydání aplikace

Po dokončení testování byla aplikace připravena k vydání. Na základě pozitivních výsledků testování bylo vhodné vypustit aplikaci do světa, proto byla umístěna do obchodu pro platformu Android – **Google Play**[2]. Aplikace byla v průběhu pár hodin obchodem schválena a publikována pro kohokoliv, kdo bude ochotný si ji nainstalovat. Zájem o aplikaci zpočátku nebyl takový, aktuálně má pouze dva aktivní uživatele, nicméně oba ohodnotili aplikaci vysokými známkami.

[2] https://play.google.com/store/apps/details?id=befit.release

Kapitola 9

Závěr

Hlavním cílem této práce byl návrh a následná implementace motivační cross-platform aplikace pro podporu a rozvoj pohybových aktivit s využitím gamifikace. Na základě těchto požadavků bylo nutné nejprve nastudovat metody a techniky gamifikace, knihovny a frameworky pro tvorbu cross-platform aplikací v jazyce JavaScript. Následně bylo nutné připravit návrh nejen uživatelského prostředí, ale i aplikace jako celku. Dle připraveného návrhu byla aplikace implementována, byly vytvořeny jednotlivé cviky a tréninky, které hrají v aplikaci důležitou roli. Na závěr této práce byla implementovaná aplikace otestována a následně vydána na Google Play. Na základě zpětné vazby z tohoto testování byly stanoveny další možnosti vývoje aplikace.

Výsledkem této práce je cross-plaftorm aplikace pro platformy web, Android a iOS, která s využitím gamifikace motivuje uživatele k pohybu prostřednictvím různých pohybových aktivit a cviků. Od ostatních aplikací se odlišuje mírou použité gamifikace, je v tomto směru více hrou, nežli běžnou sportovní aplikací. Pro implementaci této aplikace byl využit framework Ionic, který je zodpovědný za celkový vzhled a funkcionalitu a pak také framework Cordova, který umožňuje nasazení aplikace na všechny tři zmíněné platformy.

Pro další možnosti vývoje aplikace lze doporučit implementaci *client-server* přístupu a následné zavedení možností soutěžení jednotlivých hráčů mezi sebou v podobě arény a systému týmů a přátel. V oblasti uživatelského prostředí by bylo možné zavést barevnou paletu s předvolenými barvami ze kterých by si mohli uživatelé vybírat v jakém barevném schematu chtějí mít svou aplikaci.

Literatura

[1] Ancheta, W.-B.: *An Introduction to Cordova: Basics*. [Online; navštíveno 15.12.2017].
URL https://code.tutsplus.com/tutorials/an-introduction-to-cordova-basics--cms-25146

[2] Bierman, G.; Abadi, M.; Torgersen, M.: Understanding TypeScript. In *ECOOP 2014 – Object-Oriented Programming*, editace R. Jones, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, ISBN 978-3-662-44202-9, s. 257–281.

[3] Blomkvist, S.: Persona–an overview. *Retrieved November*, ročník 22, 2002.

[4] Branas, R.: *AngularJS Essentials*. Packt Publishing, 2014, ISBN 978-1-78398-008-6.

[5] Das, S.: *What is React.js and Why I recommend it to other JavaScript Developers?* [Online; navštíveno 19.1.2018].
URL https://www.linkedin.com/pulse/what-reactjs-why-i-recommend-other-javascript-sandip-das

[6] Deterding, S.; Dixon, D.; Khaled, R.; aj.: *From Game Design Elements to Gamefulness: Defining Gamification*. In *MindTrek '11 Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, Tampere, Finland: ACN New York, 2011, ISBN 978-1-4503-0816-8.

[7] Elrom, E.: *Pro MEAN Stack Development*. Apress, 2016, ISBN 978-1-4842-2044-3.

[8] Foundation, T. A. S.: *Cordova Apache Documentation: Overview*. [Online; navštíveno 15.12.2017].
URL https://cordova.apache.org/docs/en/latest/guide/overview/

[9] Gackenheimer, C.: *Introduction to React*. Apress, 2015, ISBN 978-1-4842-1245-5.

[10] Galitz, W. O.: *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. New York, NY, USA: John Wiley & Sons, Inc., 1997, ISBN 0-471-15755-4.

[11] Hamari, J.: Do badges increase user activity? A field experiment on effects of gamification. *Computers in Human Behavior*, ročník 71, 01 2017: s. 469–478.

[12] Jain, N.; Mangal, P.; Mehta, D.; aj.: AngularJS: A Modern MVC Framework in JavaScript. *Journal of Global Research in Computer Science*, ročník 5, č. 12, 2014: s. 17–23.

[13] Kessin, Z.: *Programming HTML5 Applications*. O'Reilly, 2011, ISBN 978-1-449-39908-5.

[14] Kharlampidi, V.: *Framework7 Documentation: Introduction*. [Online; navštíveno 18.1.2018].
URL https://framework7.io/docs/introduction.html

[15] Mekler, E. D.; Brühlmann, F.; Opwis, K.; aj.: *Do points, levels and leaderboards harm intrinsic motivation?: an empirical analysis of common gamification elements*. In *Gamification '13 Proceedings of the First International Conference on Gameful Design, Research, and Applications*, Toronto, Ontario, Canada: ACN New York, 2013, ISBN 978-1-4503-2815-9, s. 66–73.

[16] Ravulavaru, A.: *Learning Ionic*. Packt Publishing, 2015, ISBN 978-1-78355-260-3.

[17] Schinsky, H.: *Intro to Framework7*. [Online; navštíveno 18.1.2018].
URL https://phonegap.com/blog/2015/11/30/framework7/

[18] Simons, E.: *React in plain english*. [Online; navštíveno 19.1.2018].
URL https://thinkster.io/tutorials/what-exactly-is-react

[19] Team, O. U.: *Onsen UI ver. 2 Documentation: Architecture*. [Online; navštíveno 18.12.2017].
URL https://onsen.io/v2/guide/architecture.html#architecture

[20] Vue.js: *Comparison with Other Frameworks*. [Online; navštíveno 19.1.2018].
URL https://vuejs.org/v2/guide/comparison.html

[21] W3C: *Web SQL Database*. [Online; navštíveno 16.12.2017].
URL https://www.w3.org/TR/webdatabase/

[22] W3Schools: *HTML5 Web Storage*. [Online; navštíveno 16.12.2017].
URL https://www.w3schools.com/html/html5_webstorage.asp

[23] Wodehouse, C.: *AngularJS: A Powerful JavaScript Framework*. [Online; navštíveno 18.1.2018].
URL https://www.upwork.com/hiring/development/angularjs-basics/

Příloha A

Manuál

Na přiloženém paměťovém médiu se nachází složky *Web build* a *Android build*, které obsahují spustitelné soubory aplikace. V první jmenované složce lze nalézt soubor `index.html`, jehož otevření v libovolném internetovém prohlížeči spustí webovou verzi aplikace. Ve složce *Android build* se nachází jediný soubor s názvem `BeFIT.apk`, což je spustitelný soubor aplikace pro platformu Android. Tento soubor je potřeba přenést na libovolné zařízení se systémem Android a zde ho spustit. Proběhne bezobslužná instalace a následně bude aplikace na tomto zařízení připravena k použití. Přestože je aplikace digitálně podepsána, bude potřeba v nastavení tohoto zařízení nejprve povolit instalaci aplikací z neznámých zdrojů. Instalaci pro Android zařízení je taktéž možné provést z obchodu **Google Play**[1].

V kořenovém adresáři tohoto média se pak nachází ještě tři další složky, složka *TZ src* obsahuje zdrojové kódy technické zprávy včetně všech obrázků a jiných nezbytných věcí. Vygenerování práce ve formátu pdf probíhá zadáním příkazu `make` ve zmíněném adresáři. Dalši složkou je složka *App src*, která obsahuje zdrojové soubory aplikace logicky rozdělené do jednotlivých adresářů. Poslední složkou v kořenovém adresáři paměťového média je složka *App dependencies*, ve které se nachází kompletní projekt vytvořený frameworkem Ionic. V rámci tohoto projektu lze přeložit zdrojové kódy do spustitelné formy pro konkrétní platformu.

A.1 Překlad aplikace

Pro úspěšný překlad aplikace je nejprve nutné mít nainstalovány frameworky Cordova a Ionic. Ty lze jednoduše nainstalovat z `npm` (je potřeba mít i `Node.js`). Instalaci frameworků potom lze spustit příkazem `npm install -g ionic cordova`. Po dokončení instalace frameworků lze již v příkazové řádce navigovat do kořenové složky projektu umístěného na paměťovém médiu a zde zadat příkaz `ionic serve` pro překlad zdrojových kódů pro webovou plaftormu. Příkaz `ionic cordova build android` potom přeloží soubory pro Android a vytvoří spustitelný soubor s příponou `.apk`, zde je však nezbytné mít nainstalované vývojové prostředí nazývající se *Android Studio*. Pro překlad zdrojových kódů pro plaftormu iOS lze potom použít příkaz `ionic cordova build ios`, aby však překlad proběhl bez problémů, je třeba ho spouštět na legálně získaném operačním systému firmy Apple (např. OS X).

[1] https://play.google.com/store/apps/details?id=befit.release