České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Naim Ashhab**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Mobilní aplikace Trampingo**

Pokyny pro vypracování:

Cílem práce je je vytvoření fungující mobilní aplikace pro tablety s operačním systémem iOS, která slouží jako sportovní deník pro skoky na trampolíně. Aplikace má sloužit jako statistická pomůcka pro skokany i trenéry. Do aplikace se budou moci zadávat tréninky se všemi potřebnými informacemi. Aplikace bude ukazovat statistiky pro vybrané filtry a vybraným časovým odstupem. Aplikace bude jak pro trenéra tak pro skokana. Analyzujte potřeby takového systému a porovnejte s existujícími systémy. Navrhněte serverové API pro mobilní aplikace, dále pak jednotlivé aplikace (trenérskou, skokanskou). Implementujte a řešení otestujte.

Seznam odborné literatury:

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Vedoucí: Ing. Tomáš Černý, MSc.

Platnost zadání: do konce letního semestru 2015/2016

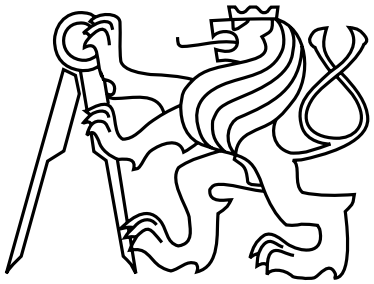
L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 3. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Mobilní aplikace Trampingo

Naim Ashhab

Vedoucí práce: Ing. Tomáš Černý, Ph.D

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

23. května 2016

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Tomáši Černému, PhD. za vedení a věcné rady při psaní bakalářské práce.

Prohlášení

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

V Praze dne 24. 5. 2016 .....

Abstract

This thesis describes process of creating Trampingo application prototype. Trampingo is an application for recording and analysing trampoline trainings. The application is composed of server back-end and two mobile applications, one for coach and one for gymnast. The application allow coach to track gymnast ongoing training and analyze it in real time. Further more it allows both coach and gymnast analyze trainings one by one or by group of trainings in specific time range.
Trampingo is giving needed insight into trainings, helps better understand training trends and make training reporting easier.

Abstrakt

Práce popisuje postupy při vytváření prototypu aplikace Trampingo. Trampingo je aplikace pro zaznamenávání a analýzu tréninků skoků na trampolíně. Aplikace je složena ze serverového back-endu a dvou mobilních aplikací, jedné pro skokana a jedné pro trenéra. Aplikace umožňuje trenérovi v reálném čase sledovat průběh tréninků svých skokanů a analyzovat jej. Dále umožňuje analyzovat všechny v minulosti ukončené tréninky jak po jednom, tak po více, ve vybraném časovém úseku.
Trampingo pomáhá trenérům i skokanům získat potřebný nadhled nad tréninky, lépe pochopit tendenci tréninků a zjednodušit skokanům vykazování tréninků.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Skoky na trampolíně jsou v dnešní době takřka neznámý sport. I když se v poslední době těší větší pozornosti ze strany medií i široké veřejnosti, stále je mnoho lidí, kteří nevěří, že se tento sport dá dělat na vrcholové úrovní nebo že je to od roku 2000 olympijský sport.

Možná proto se v něm, až na malé výjimky, nikdo nesnaží inovovat. Věřím, že tak jako se firma v dnešní době neobejde bez IT podpory, tak i sport by měl svoji IT podporu mít. To je také důvod, proč jsem si zvolil toto téma, a proč bych chtěl přispět svojí prací ke zlepšení stavu v tomto sportu.

1.1 Základní pojmy

Protože v práci používám pojmy spojené se skoky na trampolíně, které čtenář nemusí znát, popíšu je hned ze začátku.

Trénink Tréninkem se v této práci myslí čistě cvičení na trampolíně. Do tréninku není zahrnuto rozcvičení před a protažení po tréninku ani žádné jiné kondiční nebo kompenzační cvičení. Trénink je složen z několika výstupů.

Výstup Skokan na tréninku neskáče na trampolíně nepřetržitě, ale opakovaně nastupuje a slézá z trampolíny. Pro představu to můžeme přirovnat ke sprinterovi, který doběhne do cíle a krokem se vrací na start, aby znovu běžel do cíle. Jeden takovýto cyklus se nazývá jeden výstup. Výstup je složen z prvků, které skokan skáče.

Prvek Prvky jsou ve skocích na trampolíně všechny skoky kromě obyčejného výskoku. Tedy na to, aby se skok mohl označit za prvek, musí skokan provést minimálně čtvrt salta, půl vrut nebo polohu.

Poloha Poloha skoku je postavení těla v průběhu skoku definované pravidly. Polohy jsou ve skocích na trampolíně tři. Poloha schylmo, skrčmo a toporně. Každý prvek musí mít přiřazenou polohu ve které je proveden. Každá poloha přidává různé bonusové body ke koeficientu obtížnosti prvku.

Koeficient obtížnosti Každý prvek na trampolíně má přiřazenou obtížnost. Tato obtížnost je vypočítaná z počtu čtvrt salt, půl vrutů v prvku a dalších bonusových bodů.

1.2 Vize

V této práci chci vytvořit prototyp aplikace pro skoky na trampolíně. Má to být prototyp služby, kterou budou denně používat skokani i trenéři. Má to být prototyp nástroje, který usnadní evidenci a vykazování, vnese nadhled a pomůže při plánování i zpětném hodnocení tréninků.

1.3 Motivace

Já sám jsem skokanem na trampolíně a věřím, že prototyp bude prvním krokem k aplikaci, která bude mít reálné využití ve světě, a kterou budu sám využívat pro své potřeby skokana. Aplikace by měla zlepšit dohled nad tréninky a tím pomoct odhalit skryté problémy ve vývoji skokanských výkonů. Doufám, že nakonec bude Trampingo pomáhat skokanům i trenérům na celém světě. Rád bych svojí prací odstartoval novou éru vykazování a evidenci tréninků na trampolíně a ukončil éru zaznamenávání výkonů na papír a přepisování do excelové tabulky.

Kapitola 2

Analýza

Analýza je prvním krokem vývoje projektu. V analýze popisuji současný stav, cíle projektu a potřeby serveru a klientů. Dále vytvářím model doménových tříd a nasazení.

2.1 Současný stav

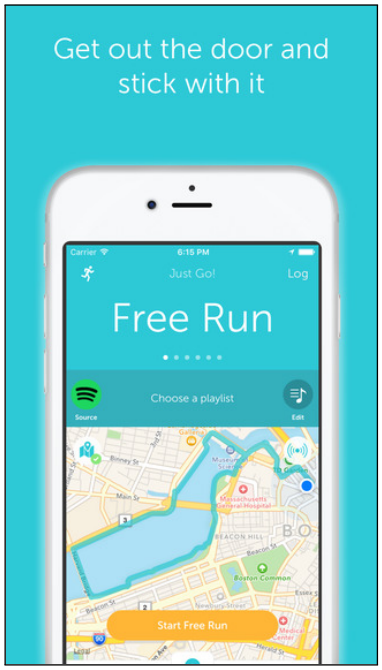
V celém světe dnes není žádná služba, která by se specializovala na vedení a vykazování tréninků ve skocích na trampolíně nebo jiném gymnastickém sportu. Můžeme najít služby pro běh, jízdu na kole, plavání a spoustu dalších populárních sportů, ale ani jednu pro sporty gymnastické. Je pochopitelné, že gymnastické sporty široká veřejnost nedělá, a tak není poptávka po takových službách. Pokud tedy gymnasta musí nebo si chce vést tréninkový deník, nezbývá mu než se spokojit s papírem a tužkou.

Proto se budu inspirovat službami, které dnes jsou a slouží podobnému účelu, a to vykazování sportovní aktivity. Nejznámější aplikací tohoto druhu je bezpochyby Runkeeper[1]. Ten umožňuje uživateli sledovat jeho běh, chůzi, jízdu na kole, plavání, lyžování a mnoho dalších aktivit. Aplikace je v základu zdarma a při placení měsíčního paušálu je k dispozici několik užitečných rozšíření. Hezkým zpestřením je i gamifikace v podobě získávání ocenění, která motivuje uživatele stanovovat si vyšší a vyšší cíle (snímky z aplikace jsou na obrázcích 2.1, 2.2 a 2.3).

Další službou tohoto druhu je aplikace Strava[2]. Ta se obdobně zaměřuje na sledování běhu a jízdy na kole. Stejně jako Runkeeper nabízí GPS sledování aktivity, kterou následně zobrazí na mapě, analyzuje všechny aktivity a motivuje uživatele oceněními, například za uběhnuté kilometry nebo zrychlení v mezičase na kilometr (snímky z aplikace jsou na obrázcích 2.4, 2.5 a 2.6).

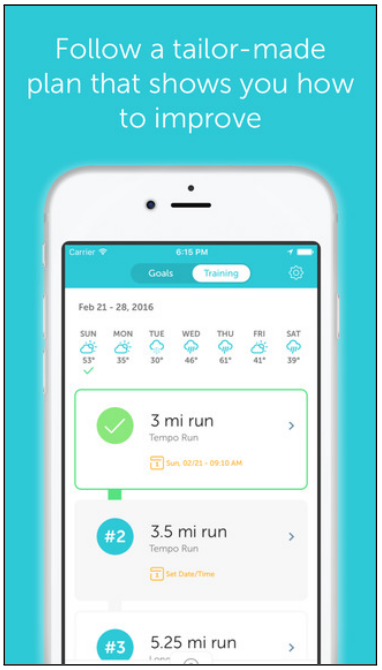
V neposlední řadě můžu zmínit aplikaci Nike+ Running[3]. Tato aplikace uživatelům dovoluje soupeřit mezi sebou, zaznamenávat běh a plnit předpřipravené tréninkové lekce, které jsou připraveny uživateli na míru. Aplikace vyžaduje přihlášení přes NikeID a tím dovoluje uživatelům spárování všech jejich Nike+ zařízení, jako FuelBank SE (snímky z aplikace jsou na obrázcích 2.7 a 2.8).

[1] <https://itunes.apple.com/us/app/runkeeper-gps-running-walk/id300235330?mt=8>
[2] <https://itunes.apple.com/us/app/strava-running-cycling-gps/id426826309?mt=8>
[3] <https://itunes.apple.com/us/app/nike+-running/id387771637?mt=8>

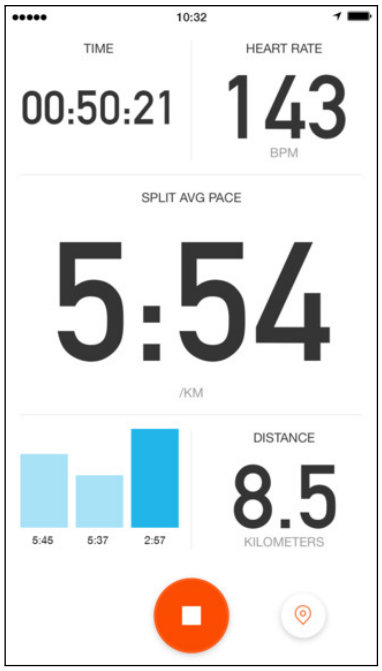


Obrázek 2.1: Snímek obrazovky aplikace Runkeeper

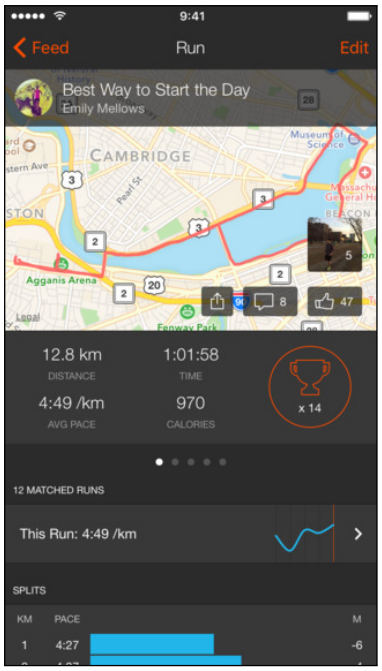
Obrázek 2.2: Snímek obrazovky aplikace Runkeeper



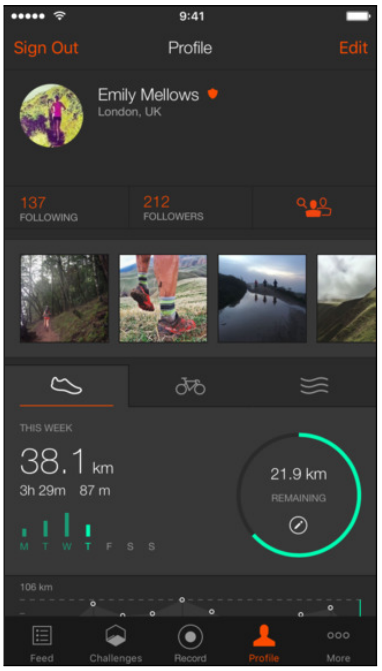
Obrázek 2.3: Snímek obrazovky aplikace Runkeeper



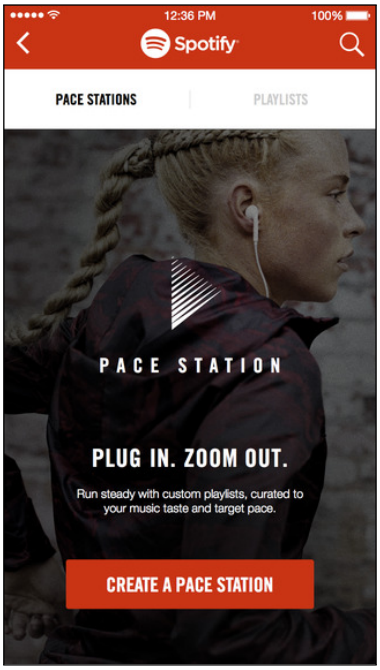
Obrázek 2.4: Snímek obrazovky aplikace Strava



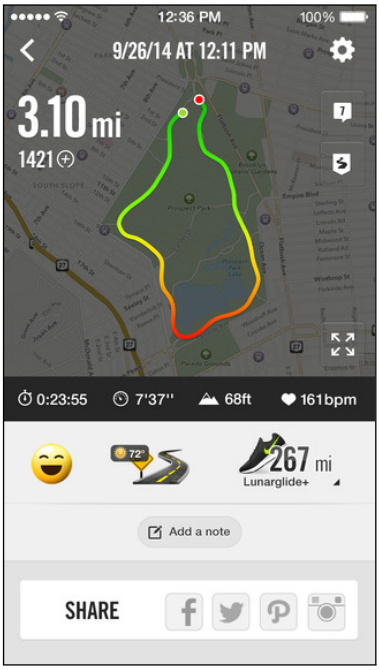
Obrázek 2.5: Snímek obrazovky aplikace Strava



Obrázek 2.6: Snímek obrazovky aplikace Strava



Obrázek 2.7: Snímek obrazovky aplikace Nike+ Running



Obrázek 2.8: Snímek obrazovky aplikace Nike+ Running

Všechny výše jmenované aplikace řeší podobný problém jako já. Zaznamenávají sportovní aktivitu a analyzují je. K tomu uživatele ke sportu motivují a nechávají ho sdílet svoje výsledky na sociálních sítích.

2.2 Role

Není nijak překvapivé, že jsou role intuitivně rozděleny na skokana a trenéra podle reality. Žádné jiné role nejsou v aplikaci potřeba.

2.3 Cíle

Na začátku projektu jsem si stanovil cíle, kterých musí projekt dosáhnout. Po diskuzi se skokany i trenéry jsem dospěl k těmto bodům:

1. Zjednodušení vyplňování tréninků
2. Zobrazování probíhajících tréninků v reálném čase trenérovi
3. Ukládání všech ukončených tréninků
4. Analýza jednoho vybraného tréninku

5. Analýza skupiny tréninků ve vybraném časovém úseku

Bez těchto cílů by projekt neměl reálný užitek.

2.3.1 Zjednodušení vyplňování tréninků

V dnešní době se všechny tréninky vyplňují na papír a u každého by měly být vyplněny minimálně tyto údaje:

- Datum a čas zahájení tréninku
- Doba tréninku
- Celkový počet výstupů
- Celkový počet skočených prvků
- Celkový koeficient obtížnosti všech prvků

Žádný z těchto údajů by neměl u vykázaného tréninku chybět. Datum a čas zahájení a doba tréninku nejsou tak náročné na vyplnění, ale zbylé tři údaje se musejí ručně dopočítávat. Když si vezmu, že typický trénink má 16 až 24 výstupů a v každém výstupu je 5 až 10 prvků, tak pro spočtení pouze celkového koeficientu obtížnosti je nutno sečíst 80 až 240 desetinných čísel. Celkem tedy musí skokan pro vypočítání všech údajů v tréninku udělat 176 až 504 součtů.

Ideální by tedy bylo, aby skokan vkládal do aplikace skočené prvky a aplikace by automaticky dopočítávala zbylé údaje. Aplikace by musela skokanovi nabízet seznam všech prvků, u kterých by byl koeficient obtížnosti. Z tohoto seznamu by skokan skládal jednotlivé výstupy a aplikace by měla všechny potřebné informace k tomu, aby dokázala všechny údaje automaticky dopočítat. Počet součtů, které musí skokan dělat ručně, by se snížil na ideálních 0.

Toto řešení by bylo jednoznačně nejlepší, ale bohužel z těchto důvodů dost náročné:

- Protože je seznam prvků na trampolíně nekonečný, musel by mít skokan možnost přidávat, editovat a mazat prvky ze svého seznamu.
- Každý skokan aktivně používá přibližně 40 prvků a zbylé neskáče tak často. Aplikace by musela řadit prvky podle používání, aby ty často skákané byly vždy dostupnější než ty méně skákané, ale aby i ty ostatní byly jednoduše dostupné.
- Opakované vybírání (80 až 240 výběrů) prvků ze seznamu, může být pro uživatele na mobilním zařízení obtížné, zvlášť když se seznam pořád mění ve snaze optimalizovat dostupnost často skákaných prvků. Proto by se muselo navrhnout ideální uživatelské rozhraní pro skládání výstupů.

Kvůli této náročnosti jsem se rozhodl implementovat do prototypu jednodušší řešení, které nepřináší takové zjednodušení skokanovi, ale pro účely prototypu poslouží dostatečně. Skokan bude muset pro vyplnění výstupu zadat počet prvků a jejich spočtený koeficient obtížnosti. Aplikace dále dopočítává celkový součet prvků a obtížnosti za trénink a navíc přidá průměrnou obtížnost na prvek. Tento poslední údaj je tam jako přidaná hodnota aplikace a je to na první pohled pěkný ukazatel náročnosti celého tréninku i jednotlivých výstupů.

2.3.2 Zobrazování probíhajících tréninků v reálném čase trenérovi

Jednou z výhod elektronické evidence oproti papírové je ta, že ji můžeme synchronizovat s jinými zařízeními. Trenér by měl být schopen vidět, kdo z jeho skokanů právě trénuje a kdo ne. Pak si jejich tréninky jednotlivě zobrazit a sledovat jejich postup. To by s papírovou evidencí nebylo možné.

Vše by mělo probíhat co nejvíce automaticky, bez potřeby trenéra neustále aktualizovat změny. Skokan přidá výstup a u trenéra v zařízení se tato změna ihned projeví. Stejně tak je to i se vším ostatním, jako smazání výstupu, editace výstupu a zrušení nebo dokončení celého tréninku. Není potřeba žádného tlačítka "aktualizovat".

Další výhodou může být situace, kdy trenér není na tréninku fyzicky přítomen. Taková situace není úplně běžná, ale může nastat. Trenér si v tomto případě jenom otevře aplikaci a odkudkoli sleduje průběh tréninků, a to právě ve zmiňovaném reálném čase.

2.3.3 Ukládání všech ukončených tréninků

Proto, aby aplikace mohla nahradit papírovou evidenci, musí jako evidence fungovat. Kvůli tomu je nezbytné, aby všechny tréninky byly ukládány a v budoucnu k nim byl umožněn přístup. Tento cíl je vcelku intuitivní, tak ho není potřeba dále rozepisovat.

2.3.4 Analýza jednoho vybraného tréninku

Jedním z hlavních přínosů aplikace má být jednodušší analýza tréninků. Analýza je rozbor celku na menší části ve snaze mu lépe porozumět. V tomto případně rozebíráme trénink na jednotlivé výstupy. Proto budeme analyzovat údaje, které se v každém výstupu mohou měnit. Například dobu tréninku není třeba analyzovat, neboť je to jeden údaj, který je pro jeden trénink konstantní. Toto je seznam analyzovatelných údajů v tréninku na jeden výstup:

- Počet prvků
- Celkový koeficient obtížnosti
- Průměrný koeficient obtížnosti na prvek

Do prototypu jsem zvolil počet prvků za výstup a průměrný koeficient obtížnosti na prvek. Protože má každý výstup různý počet prvků, není celkový koeficient obtížnosti na výstup tak směrodatný jako průměrný koeficient obtížnosti na prvek. Ten nám dokáže na první pohled lépe říci, jestli se obtížnost prvků ve výstupech zvyšuje nebo snižuje.

2.3.5 Analýza skupiny tréninků ve vybraném časovém úseku

Tak jako je jedním z cílů analýza jednoho tréninku, je cílem i analýza skupiny tréninků. Analýzou nebudeme rozebírat trénink na jednotlivé výstupy, ale skupinu tréninků na jednotlivé tréninky. Díky tomu se seznam analyzovatelných údajů zvětšuje:

- Doba tréninku
- Celkový počet výstupů
- Celkový počet prvků
- Celkový koeficient obtížnosti
- Průměrný počet prvků na výstup
- Průměrný koeficient obtížnosti na výstup
- Průměrný koeficient obtížnosti prvku za trénink

2.4 Požadavky

Se stanovenými cíli mohu snadno získat seznam požadavků na systém. Požadavky z cílů jsou převážně funkční a tak je ještě rozšířím o požadavky nefunkční.

2.4.1 Funkční požadavky

Ještě než se pustím do vytváření požadavků z cílů, můžu si vytvořil úplně základní požadavky pro všechny systémy s neveřejným obsahem. Z podstaty toho, že v systému existují pravidla na dostupnost obsahu, potřebuji aby v něm byli uživatelé, kteří by se do systému přihlašovali. K tomu, aby se mohl uživatel přihlásit, je nutné, aby se nejdříve registroval. Tak my vzniknou první dva požadavky *Přihlášení do aplikace* a *Registrace uživatele*. K těm ještě přidám požadavky *Změna hesla* a *Obnova ztraceného hesla*, které by neměli u žádného systému s hesly chybět.

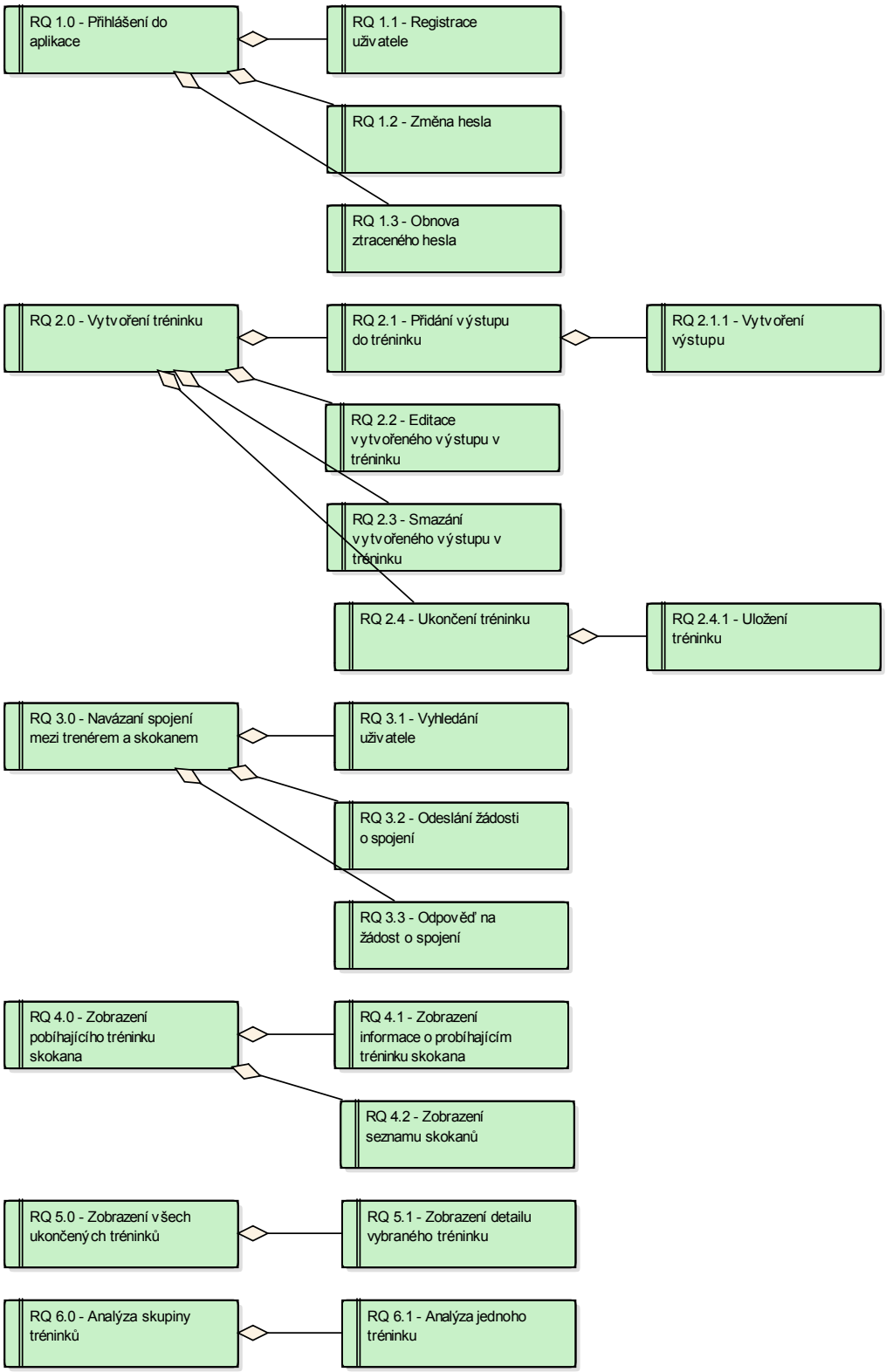
Z cíle zjednodušení vyplňování tréninku si vytvořím požadavek na *Vytvoření tréninku*. Ten složím z požadavků *Přidání výstupu do tréninku*, *Vytvoření výstupu*, *Editace vytvořeného výstupu v tréninku*, *Smazání vytvořeného výstupu v tréninku*, *Ukončení tréninku* a *Uložení tréninku*. Více požadavků z tohoto cíle nedokážu vytvořit.

Další cíl je zobrazovat probíhající trénink v reálném čase trenérovi. K tomu potřebuji, aby bylo vůbec možné nějaké *Navázaní spojení mezi trenérem a skokanem*. To zahrnuje *Vyhledání uživatele*, *Odeslání žádosti o spojení* a *Odpověď na žádost o spojení*. Když už je spojení navázáno, mohu požadovat *Zobrazení pobíhajícího tréninku skokana* a to díky *Zobrazení seznamu skokanů* a *Zobrazení informace o probíhajícím tréninku skokana*.

Díky tomu, že všechny tréninky ukládám, mohu požadovat *Zobrazení všech ukončených tréninků* a *Zobrazení detailu vybraného tréninku*.

A abych naplnil všechny ostatní stanovené cíle, přidám poslední dva funkční požadavky *Analýza skupiny tréninků* a *Analýza jednoho tréninku*.

Diagram funkčních požadavků je na obrázku 2.9.

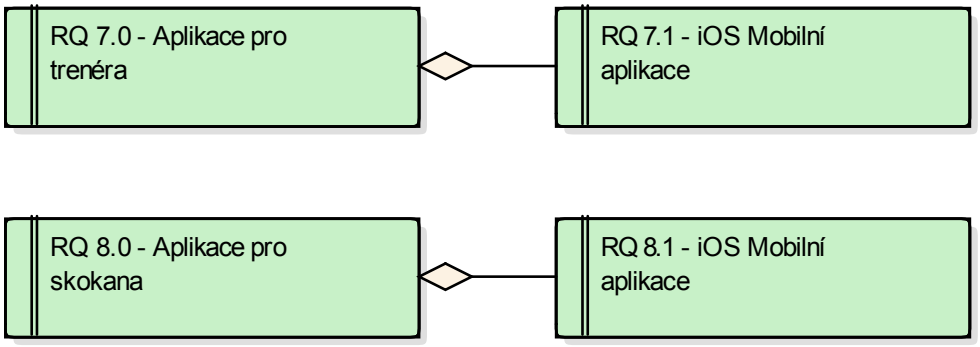


Obrázek 2.9: Diagram funkčních požadavků

2.4.2 Nefunkční požadavky

Abych doplnil funkční požadavky vycházejících z cílů, přidám nefunkční požadavky *Aplikace pro trenéra* a *Aplikace pro skokana*, které jsou rozšířené o požadavek *iOS Mobilní aplikace*, který je pod oběma.

Diagram nefunkčních požadavků je na obrázku 2.10.



Obrázek 2.10: Diagram nefunkčních požadavků

2.5 Uživatelské aktivity

Obě aplikace jsou si velice podobné, a tak budu popisovat aktivity obou aplikací zároveň. Aktivity které si zaslouží pozornost v této sekci, jsou pouze přihlášení/registrace a vytvoření tréninku. Ostatní jsou natolik intuitivní a jednoduché, že je není třeba dále popisovat.

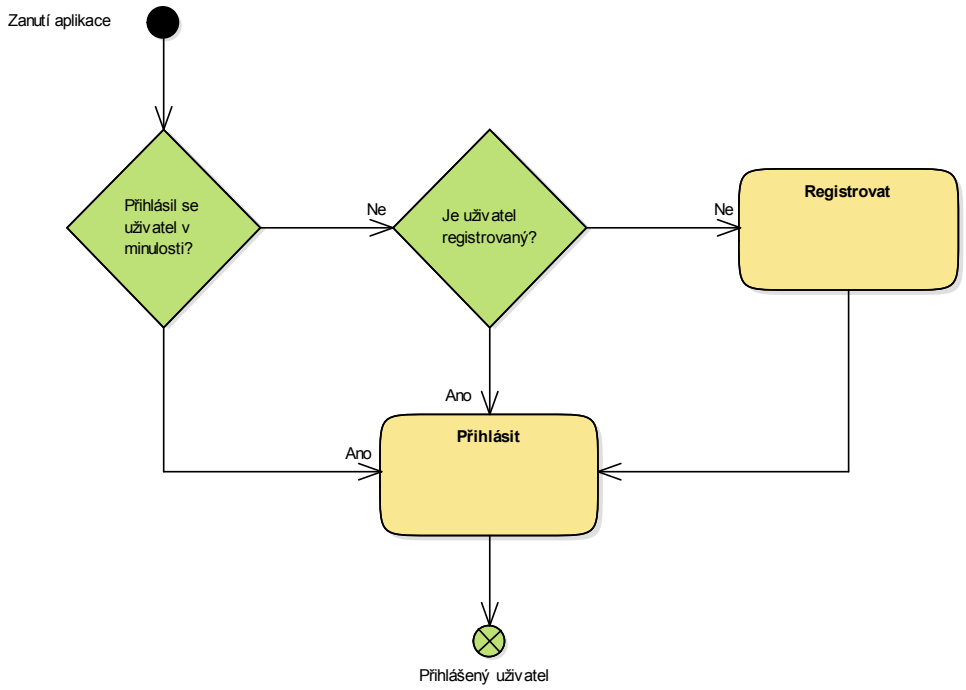
2.5.1 Přihlášení/Registrace

Jako první věc, kterou uživatel uvidí při zapnutí aplikace, bude obrazovka pro přihlášení/registraci. Přes tuto obrazovku musí každý uživatel projít, aby si vytvořil účet, přihlásil se a mohl začít aplikaci používat. Pokud se uživatel v minulosti již přihlásil, po zapnutí ho aplikace automaticky přihlásí znovu. Průchod je zaznamenán na obrázku 2.11 na straně 11.

Po přihlášení se uživatel dostane na hlavní obrazovku. Obě role zde mají možnost zobrazit seznam tréninků a analýzu. Skokan má navíc možnost zahájit trénink.

2.5.2 Vytvoření tréninku

Zahájit trénink může pouze skokan, který tak učiní z hlavní obrazovky aplikace. Po zahájení se skokanovi automaticky začne měřit čas do ukončení, tak aby ho později nemusel sám zaznamenávat. Z této obrazovky může skokan trénink ukončit, zrušit a přidat, odebrat nebo změnit výstup. Není stanoven žádný horní limit počtu výstupů v tréninku, ale pro ukončení a



Obrázek 2.11: Diagram aktivity přihlášením/registrací

uložení tréninku musí trénink obsahovat alespoň jeden výstup. Trénink bez žádného výstupu by nebyl trénink. Zrušit trénink může skokan kdykoli.

Na této obrazovce vidí skokan dále i analýzu tréninku, která se s každou změnou tréninku aktualizuje.

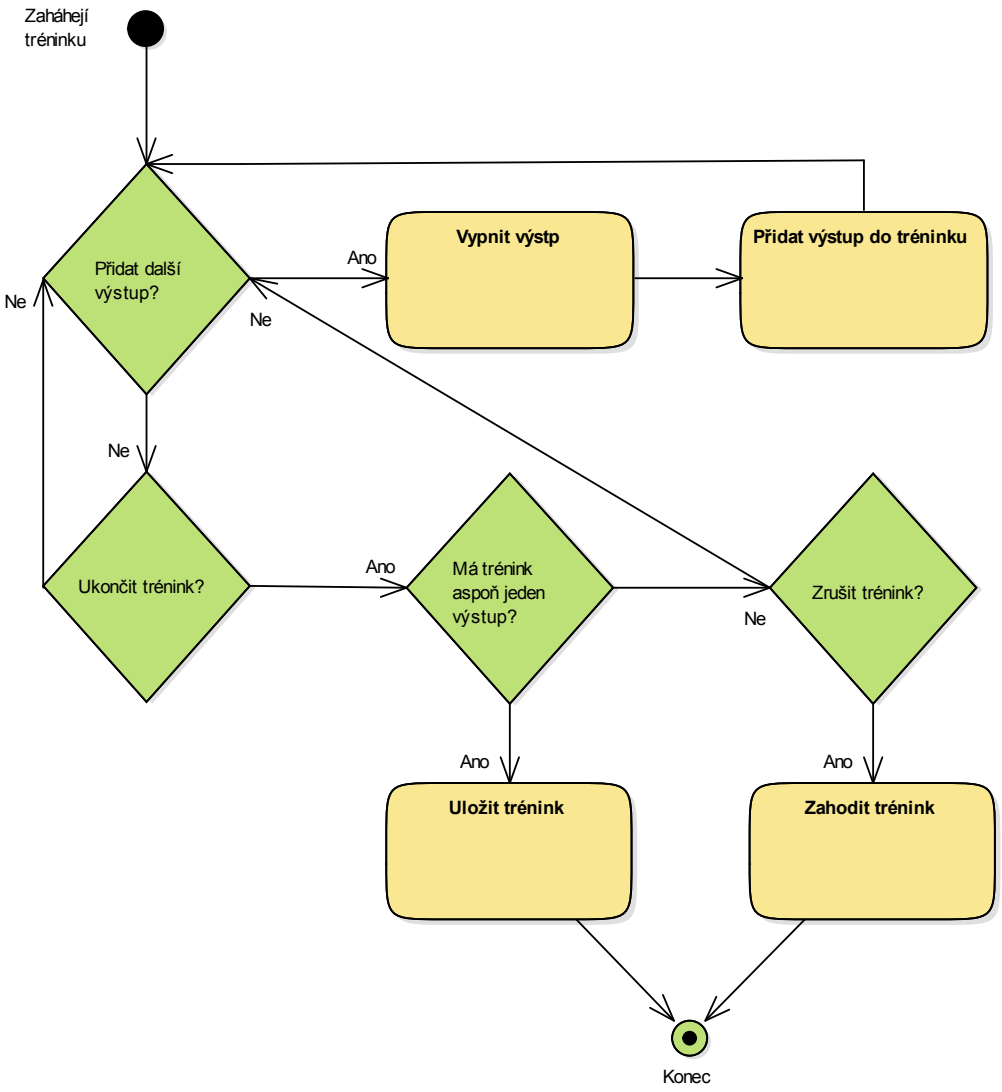
Pro představu je na obrázku 2.12 na straně 12 zaznamenán průchod vytvářením tréninku.

2.6 Analýza doménových tříd

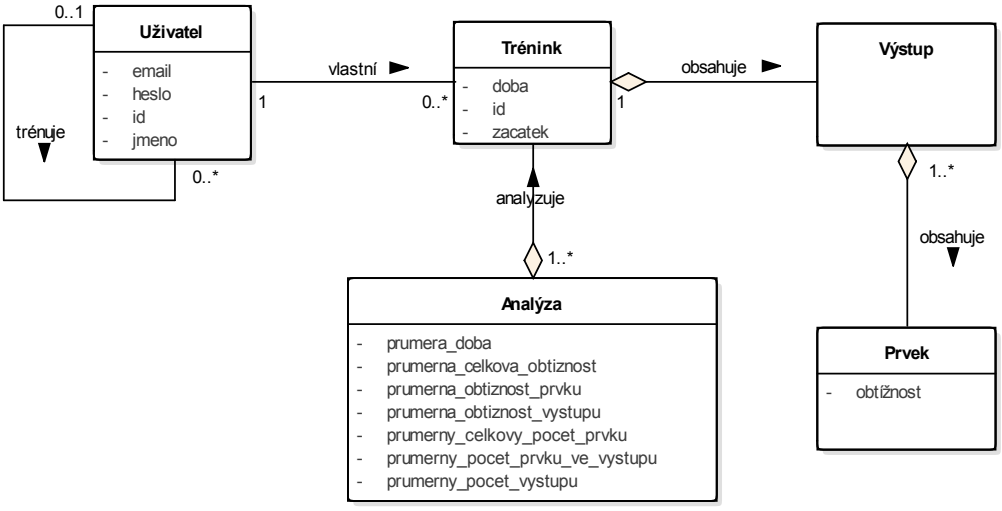
Pro abstrakci celého systému jsem si vytvořil doménový model tříd (obrázek 2.13). Model zachycuje, co by měli hlavní modelové třídy obsahovat a jaké jsou mezi nimi závislosti.

2.6.1 Popis doménového modelu

Základní třídou systému je Uživatel. Ten obsahuje atributy id (identifikátor), jméno, email, heslo a vazby spojující trenéra a jeho skokany. Uživatel může být jako trenér u více skokanů, ale jako skokan může mít nanejvýš jednoho trenéra. Uživatel u sebe nemá atribut role, protože může být jak skokan tak trenér zároveň. Roli pod kterou zrovna vystupuje se rozlišuje podle aplikace, kterou zrovna používá.



Obrázek 2.12: Diagram aktivity vytváření tréninku



Obrázek 2.13: Analytický doménový model

Další veledůležitou třídou je třída Trénink. Tu vytváří Uživatel s rolí skokan, který je dále jeho vlastníkem. Trénink nemůže existovat bez přiřazeného skokana, tak jako v reálném světě. Trénink je složen z Výstupů a ty jsou složeny z Prvků. Každý prvek si nese svůj koeficient obtížnosti.

Poslední třídou v modelu je Analýza. Analýza je celá složena z tréninků, které analyzuje. Analýza musí obsahovat alespoň jeden trénink.

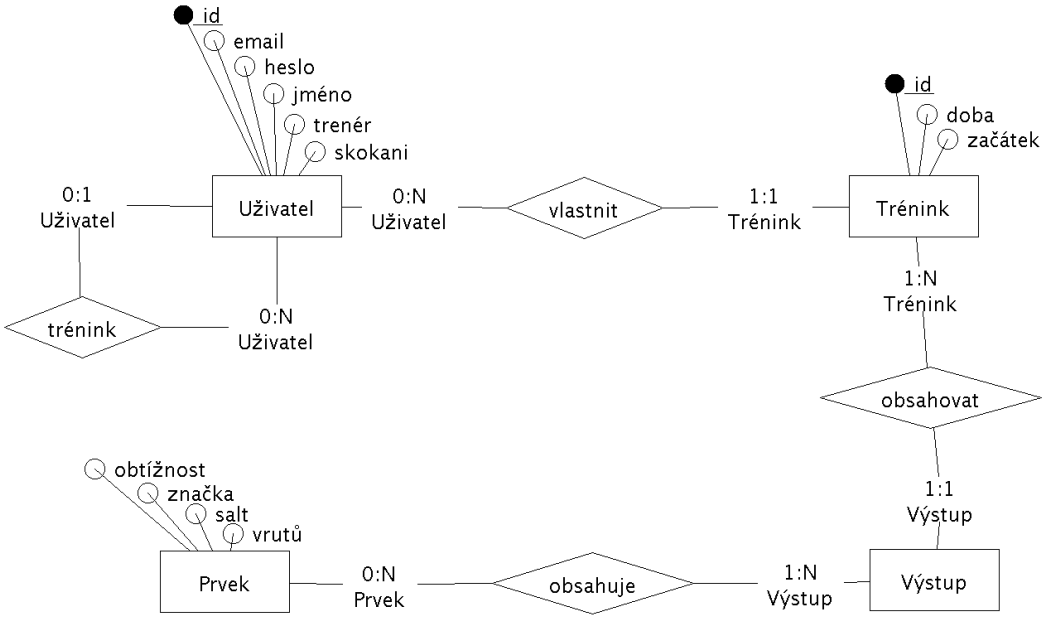
2.7 Model databáze

Když už mám vytvořený analytický doménový mode, mohu si rozmyslet co všechno budu chtít ukládat do databáze. Určitě chci mít uložené uživatele a tréninky, jinak bych nemohl žádného uživatele přihlásit a ani bych mu nemohl zobrazovat ukončené tréninky. Co ale nemusím ukládat je analýza. Tu budu vypočítávat vždy z aktuálních dat z vybraného časového úseku.

Na obrázku 2.14 je hotový diagram modelu databáze. Diagram je ER model.

2.8 Mobilní aplikace

Jednou z otázek, která při analýze vyvstává, je kolik mobilních aplikací má projekt vlastně mít. Vím, že skokan i trenér mají jak společné funkce (zobrazení a analýza tréninků), tak i funkce rozdílné (trenér nemůže začít vyplňovat trénink). Nabízejí se dvě možnosti. Jedna aplikace pro skokana i trenéra nebo dvě aplikace, každá pro jednoho.



Obrázek 2.14: ER model databáze

2.8.1 Jedna aplikace pro obě role

Jedním z možných řešení je vyvinout jednu aplikaci, do které se bude přihlašovat jak trenér, tak skokan. Při přihlašování si uživatel zvolí, jestli se chce přihlásit jako skokan nebo jako trenér a podle toho se mu aplikace přizpůsobí.

Na první pohled je to výhodné v tom, že aplikace může dát uživateli možnost změnit svou roli bez nutnosti opětovného přihlášení.

Na druhou stranu, pokud by chtěl uživatel využívat službu jako skokan i jako trenér zároveň, musel by se opakovaně vracet na místo, kde se role mění.

Nedokážu odhadnout, kolik by takových uživatelů bylo, ale toto řešení by jim určitě nezjednodušovalo práci. Možná by je to dokonce od používání služby úplně odradilo.

2.8.2 Dvě aplikace pro každou roli

Druhým řešením je vyvinout aplikace dvě, jednu pro každou roli, a nechat uživatele, aby používal tu, kterou potřebuje. Uživatel si zde nemusí vybírat roli. Ta je mu přiřazena podle aplikace, přes kterou se přihlašuje.

Výhoda tohoto řešení je v možném paralelním používání obou aplikací a tedy i rolí. Uživatel může mít zapnuté obě najednou, změnit aplikaci podle potřeby a nemusí opouštět kontext ani jedné z nich.

Další výhodou dvou aplikací je možnost přizpůsobit uživatelské rozhraní pro potřeby jednotlivých rolí. Co je pro jednu roli hlavní funkčnost, může být pro druhou vedlejší a naopak.

2.8.3 Volba

Při rozhodování jsem se nechal inspirovat většími firmami. Například společnost Uber[4], alternativní taxi služba, má role rozdělené velice podobně. Má řidiče (v mém případě trenéry) a zákazníky (v mém případě skokany). I když mají obě role hodně společného, mají každá vlastní mobilní aplikaci.

Proto jsem do prototypu zvolil variantu se dvěma aplikacemi.

2.9 Server

Pro mobilní aplikace je server hlavní úložiště a zdroj všech sdílených dat. Samotné aplikace by byly bez serveru velice omezené. Stále by mohly vytvářet nové tréninky, lokálně je ukládat a analyzovat je, ale prakticky by spolu nemohly komunikovat. Postrádaly by hlavní smysl.

Můj server se tedy bude skládat z databáze pro ukládání dat a veřejného API, přes které budou mobilní aplikace se serverem komunikovat.

Pro potřeby prototypu není potřeba žádného webového rozhraní. Klienti systému jsou mobilní aplikace.

2.10 Komunikace

Pro komunikaci mobilních klientů se serverem se běžně používají dva protokoly, HTTP a WebSocket. Jak popsal Arun Gupta ve svém článku [7], každý má své výhody a nevýhody.

Oba protokoly používají ke komunikaci TCP spojení [17]. Jejich hlavní rozdíl je v tom, jakou má jejich spojení životnost. HTTP si pro každý dotaz vytvoří nové spojení a po získání odpovědi ho zahodí. Oproti tomu WebSocket (dále jen WS) si ho drží po celou dobu a všechny dotazy a odpovědi vede přes toto spojení.

Díky tomu, že WS svoje spojení po připojení drží, může být komunikace obousměrná. Jak server, tak klient mohou kdykoli poslat zprávu na druhou stranu. To by přes HTTP nešlo. Tam zprávu posílá vždy jen klient a server mu na ni odpovídá. Navíc je přenášení dat přes WS znatelně rychlejší právě díky tomu, že není potřeba pokaždé navazovat nové spojení.

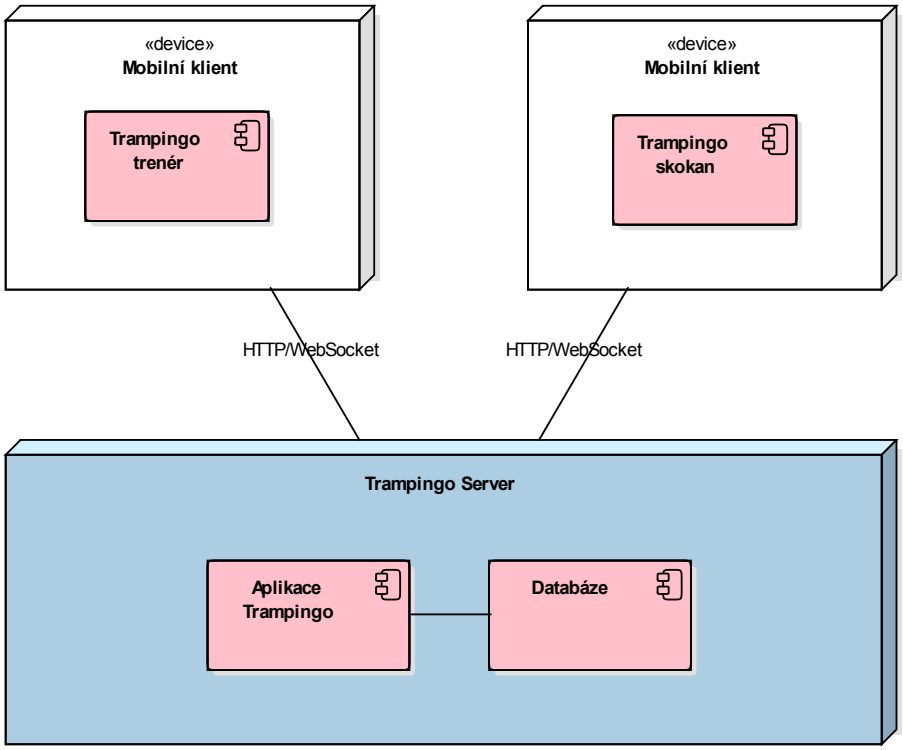
WS je tedy pro splnění cíle č. 2 (Zobrazování probíhajících tréninků v reálném čase trenérovi) nepostradatelný. Jsou i jiné možnosti, jak tohoto cíle dosáhnout. Například bych se mohl po nějakém časovém úseku ptát serveru, jestli se na straně skokana něco nezměnilo nebo ze serveru posílat notifikaci, že si má klient stáhnout nová data. WS je však nejjednodušší a nejelegantnější řešení.

Kvůli všem potřebám projektu jsem nakonec komunikaci vyřešil takto. HTTP používám pro přihlášení a na WS se připojuji vždy jako autentizovaný uživatel.

[4] <https://www.uber.com>

2.11 Model nasazení

Pro představu o tom jak budou jednotlivé komponenty projektu rozděleny, jsem vytvořil model nasazení (obrázek 2.15).



Obrázek 2.15: Model nasazení

Na serveru bude nasazena samotná aplikace s databází bez webového rozhraní. Používat aplikaci bude možné pouze přes mobilní klienty. Pro zdůraznění, že jsou mobilní aplikace dvě, každá pro jednu roli, jsem do modelu zanesl dvě zařízení, v každém jednu aplikaci.

Kapitola 3

Návrh

V této kapitole se zabývám návrhem jednotlivých komponent systému. Každou komponentu budu vyvíjet v objektově orientovaném paradigmatu.

3.1 Interakce

V této sekci chci navrhnou interakci komponent systému při vyplňování tréninku. Ostatní aktivity jsou tak triviální, že je není potřeba dále rozepisovat.

3.1.1 Vyplňování tréninku

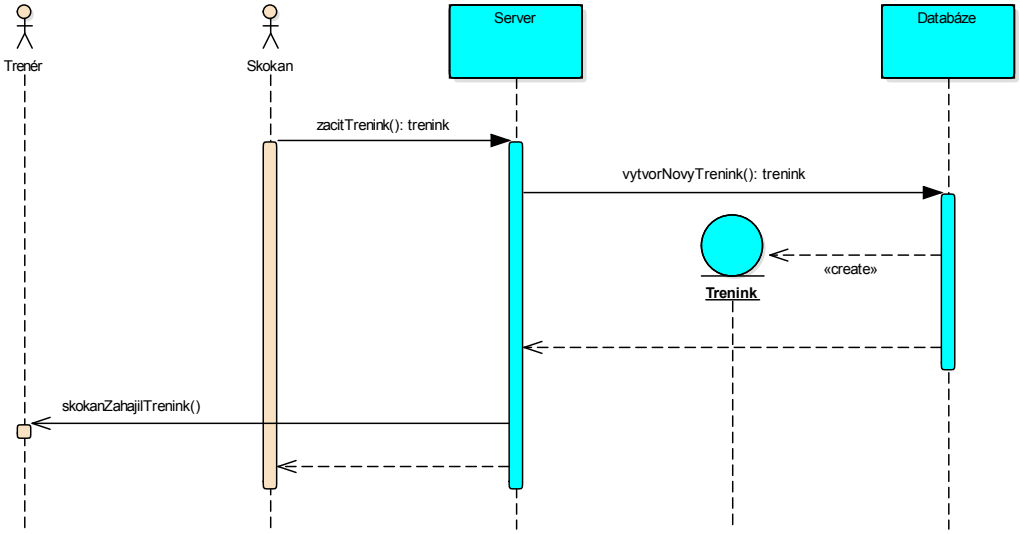
Tato interakce musí začít zahájením tréninku a k tomu musí být uživatel přihlášení ve skokanské aplikaci. To jsou základní podmínky pro tuto interakci. V tu chvíli je skokan na stránce s pobíhajícím tréninkem a trenér je o této informaci notifikován. Pro návrh budu předpokládat, že trenér sleduje tento probíhající trénink. Sekvenční diagram zahájení tréninku je na obrázku 3.1.

V další fázi interakce skokan vyplňuje trénink, tak jak ho skáče na tréninku. Tato fáze se opakuje tak často, dokud skokan trénink neukončí nebo nezruší. Skokan přidá/změní/odebere výstup, změna se zaznamená na serveru a pošle trenérovi. V diagramu na obrázku 3.2 je znázorněné celá tato interakce.

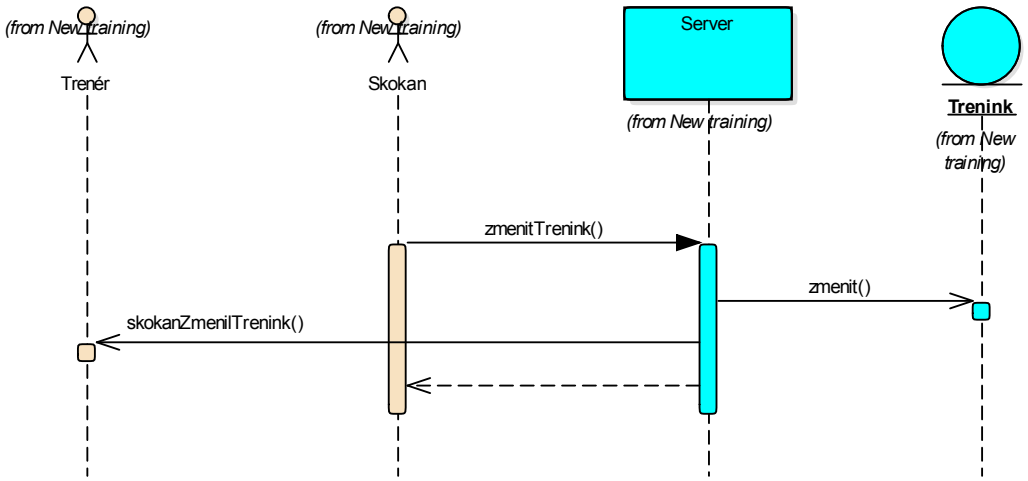
Poslední fází je ukončení/zrušení tréninku. Sekvenční diagram poslední části je na obrázku 3.3.

3.2 Entity

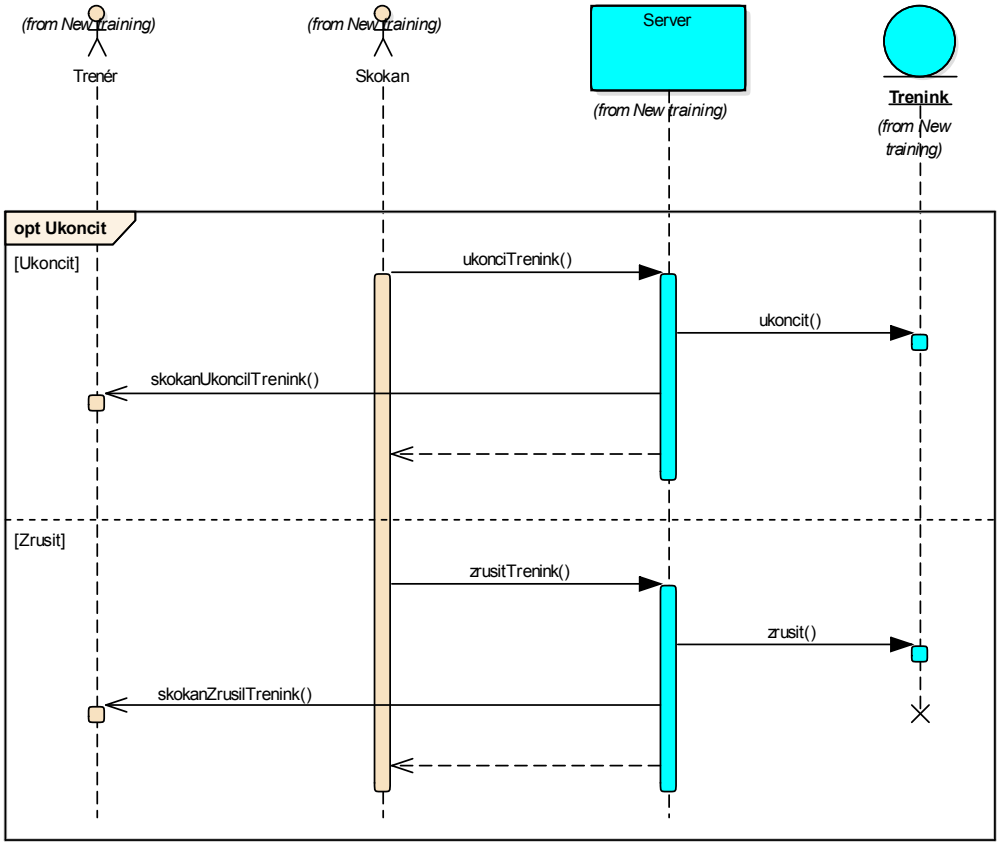
V projektu jsou dvě hlavní entity, tréninky a uživatelé. Obě entity dále popíšu.



Obrázek 3.1: Sekvenční diagram zahájení tréninku



Obrázek 3.2: Sekvenční diagram vyplňování tréninku



Obrázek 3.3: Sekvenční diagram ukončení/zrušení tréninku

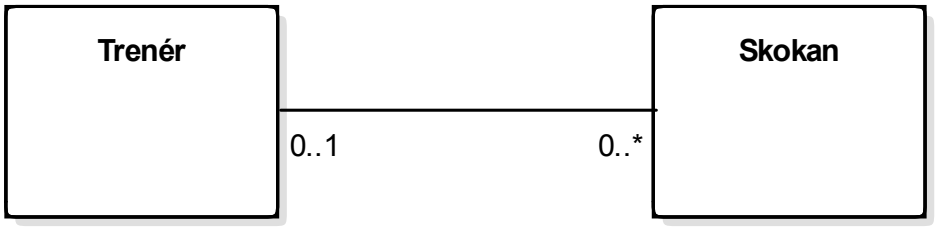
Jméno atributu	Typ atributu
id	String
jméno	String
email	String
heslo	String
role	[trenér, skokan]
trenér	Uživatel.id
accessToken	String

Tabulka 3.1: Uživatelské atributy

3.2.1 Uživatel

Entita uživatel reprezentuje skutečného uživatele, který bude aplikaci používat. Očekává se, že bude mít atributy jako unikátní identifikaci, jméno, email, heslo a roli, kterou reprezentuje (tj. skokan nebo trenér). Co už nemusí být tak zřejmé je to, jak budu ukládat navázané spojení mezi jednotlivými uživateli.

Vazby mezi rolemi mám nadefinované tak, že trenér může mít neomezeně skokanů, ale skokan může mít nanejvýše jednoho trenéra (obrázek 3.4). Díky tomu mi stačí, abych si u skokana držel odkaz na trenéra, pokud nějakého trenéra má. Pak dokážu získat všechny informace, které potřebuji. Ve chvíli, kdy potřebuji trenéra u skokana, najdu si trenéra podle identifikace. Pokud potřebuji všechny skokany k danému trenérovi, vyhledám si všechny skokany, u kterých se identifikace trenéra shoduje s identifikací daného trenéra.



Obrázek 3.4: Vazby mezi uživatelskými rolemi

Jako poslední atribut, který u uživatele potřebuji je takzvaný *access token*. Tento token mi bude sloužit jako autentizace uživatele po přihlášení. Token bude unikátní pro každého uživatele a bude se generovat při registraci nebo při změně hesla. Uživatel ho bude přikládat ke každému dotazu na server a server si tím ověří totožnost dotazujícího uživatele.

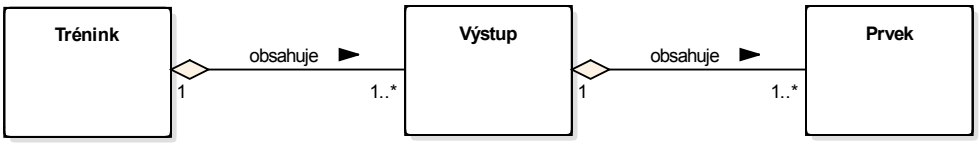
Výsledný výčet všech atributů je zanesen v tabulce 3.1.

Jméno atributu	Typ atributu
id	String
majitel	Uživatel.id
začátek	Date
doba	Int
výstupy	[[Prvek]]
ukončený	Bool

Tabulka 3.2: Tréninkové atributy

3.2.2 Trénink

V reálném světě je trénink posloupností výstupů, které jsou posloupností jednotlivých prvků. Vazby mezi tréninkem, výstupem a prvkem jsou znázorněny v obrázku 3.5. Prvek je elementární částice každého výstupu a tedy i tréninku. Dá se tedy s nadsázkou říci, že trénink je jen jiné označení pro posloupnost prvků. Trénink má ještě pár svých atributů, které trénink doplňují, aby dokázal kopírovat reálný svět a to celkovou dobu tréninku a jeho začátek.



Obrázek 3.5: Model Tréninku

V neposlední řadě mi chybí přidat k tréninku ještě atribut majitel. Je to uživatel, který ho vytvořil. Jako poslední atributy přidám identifikátor a informaci, jestli je trénink ukončený.

Výsledný výčet atributů je pak zanesen v tabulce 3.2.

3.3 Databáze

Při návrhu databáze jsem se potřeboval rozhodnout, jakou databázovou technologii zvolím. Vybíral jsem z možností SQL databází a NoSQL, konkrétně tedy dokumentově orientovaných databází.

Do databáze potřebuji ukládat dvě třídy, uživatele a tréninky. Ukládání uživatelů je u obou možností velice podobné, tam je více méně jedno, kterou možnost zvolím. Rozhodující bylo ukládání tréninků. Tak jak je ukázáno na obrázku 3.5, trénink je složen z řady výstupů a výstupy jsou složeny z řady prvků. V SQL bych pro uložení tréninků potřeboval 3 různé tabulky. Jednu pro samotné tréninky, jednu pro výstupy a jednu pro prvky. Pro načtení tréninku z databáze bych tedy musel použít dvakrát funkci *JOIN*.

Oproti tomu dokumentová NoSQL databáze nemá tabulkovou strukturu dat. Dokumenty mají strukturu takřka stejnou jako JSON a to jim dovoluje denormalizovat jejich data (vkládat do sebe další dokumenty). Díky tomu mohu uložit jeden trénink jako složený JSON. To vede ke zrychlení načítání dat. Proto byla mojí volbou právě dokumentová NoSQL databáze.

3.4 Server

Už vím, že server bude muset umět komunikovat přes HTTP zprávy a WebSocket. Z toho vyvozuji i existenci rozhraní pro oba typy komunikace. Dále vím, že je potřeba někde ukládat data a že potřebuji místo, kde si budu párovat právě připojené sockety a uživatele, kteří se přes socket autentizovali. Z toho si můžeme udělat výčet vlastností, které server bude mít:

- HTTP API
- Komponentu pro zpracování HTTP zpráv
- WebSocket API
- Komponentu pro zpracování WebSocket zpráv
- Komponentu pro spravování databáze
- Slovník pro spárování uživatelů a připojených socketů

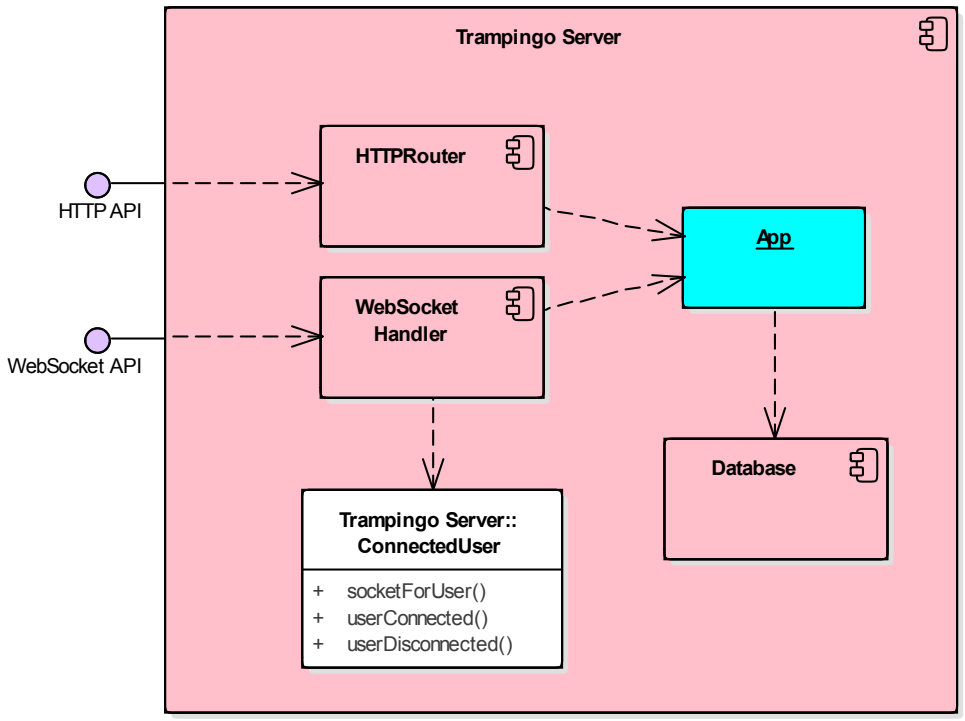
Na obrázku 3.6 je vidět abstrakce serveru jako spojení všech dílčích částí, reprezentujících vnitřní strukturu serveru.

3.5 API

API označuje rozhraní, přes které spolu mobilní aplikace a server budou komunikovat. Systém má dvoje API, jedno pro HTTP a jedno pro WebSocket.

Po lepší konzistenci a abstrakci odpovědí, jsem si vytvořil i minimální kostru, kterou každá odpověď musí mít. Každá odpověď musí obsahovat pole *status*, které nese informaci, jestli se dotaz vydařil nebo ne, pole *data*, ve kterém se budou posílat data odpovědi a pole *error*, které slouží k popisu chyby, pokud se dotaz nevydařil. Pole *error* je dále rozděleno na dvě podpole: *code*, podle kterého bude klient vypisovat lokalizované chybové hlášky, a *message*, pro lepší čitelnost odpovědi.

```
{
        status: true/false,
        data: {
                // Response data
        },
        error: {
                code: 1001,
                message: "User not found"
        }
}
```



Obrázek 3.6: Diagram struktury serveru

3.5.1 HTTP

Pro návrh HTTP API jsem si vybral REST architekturu. REST je architektura rozhraní, navržená pro distribuované prostředí. Je použitelné pro jednotný a snadný přístup ke zdrojům a k jejich přístupu používá 4 základní metody [12].

Celé HTTP API je v příloze.

3.5.2 WebSocket

Komunikace přes WebSocket postrádá metody pro přístup ke zdrojům, tak jak je to u REST. Proto si musím tyto metody něčím nahradit. Vyřešil jsem to tím, že ve jméně zprávy, kterou posílám, vytvářím jakési URI, podle kterého se rozhoduji, co chci dělat. Například pokud chci smazat trénink, tak jméno zprávy bude *training/delete* a pokud bych chtěl naopak přidat nový trénink, pojmenuji zprávu *training/add* a tak dále.

Celé WebSocket API je v příloze.

3.6 Mobilní aplikace

Mobilní aplikace jsou hlavní částí celého projektu. Je důležité, aby byly aplikace navrženy tak, aby se ovládaly co nejintuitivněji, byly stabilní, rychlé a splňovaly všechny cíle projektu. Na mobilních aplikacích celý projekt stojí a padá.

Pro mobilní klienty je na výběr ze tří platforem, na kterých mohu prototyp implementovat a to Windows Phone[1] od Microsoft[2], Android[3] od Google[4] nebo iOS[5] od Apple[6]. Protože již mám nějaké zkušenosti s vývojem aplikací třetích stran pro telefony s operačním systémem iOS, vybral jsem si iOS i jako platformu pro vývoj prototypu.

3.6.1 Architektura

Apple silně podporuje všechny vývojáře, aby pro své aplikace používali architektonický vzor Model-View-Controller (dále jen MVC). Originální vzor sestává ze tří objektů. *Model* představuje objekt aplikace, *View* prezentaci na obrazovce a *Controller* definuje způsob, jak uživatelské rozhraní reaguje na vstup od uživatele. MVC odděluje tyto objekty a tím zvyšuje flexibilitu a znovupoužitelnost celého řešení [14].

Tyto objekty spolu komunikují podle těchto pravidel [18]:

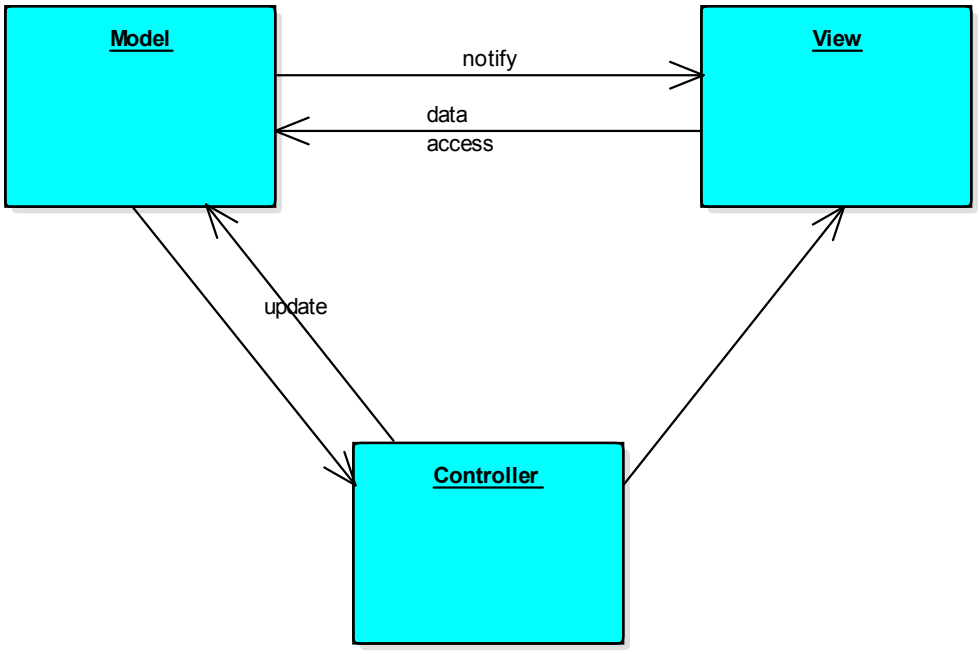
1. Controller aktualizuje Model
2. Model notifikuje View, že se změnil
3. View si bere data přímo z Modelu

Apple si však tento vzor trochu pozměnil. Hlavní rozdělení objektů nechal tak, jak je to v originálním vzoru, ale změnil způsob, jakým spolu komunikují. Podle Apple MVC vzoru jsou pravidla pro komunikaci takováto (obrázek [3]):

1. View vyvolá uživatelskou akci na Controlleru
2. Controller aktualizuje Model
3. Model notifikuje Controller, že je aktualizovaný
4. Controller aktualizuje View, podle nových dat

Pro svoje aplikace jsem použil právě takto pozměněný MVC vzor podle Apple.

[1] <https://www.windowsphone.com/>
[2] <https://www.microsoft.com/>
[3] <https://www.android.com>
[4] <https://www.google.com>
[5] <http://www.apple.com/ios>
[6] <http://www.apple.com>



Obrázek 3.7: Architektonický vzor Model-View-Controller

3.6.2 Grafické uživatelské rozhraní

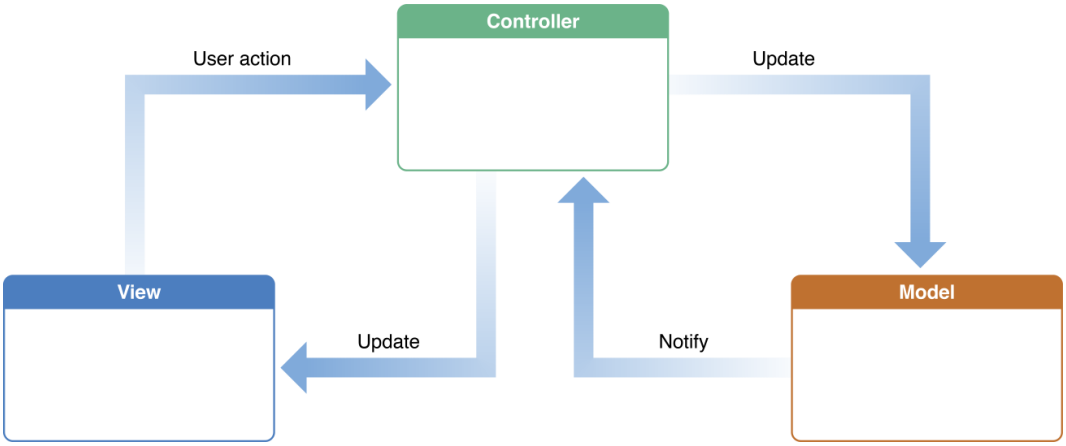
Z minulé sekce si dokážu udělat dobrý obrázek o tom, jaké obrazovky bude aplikace mít. Přihlášení, registrace, hlavní obrazovka, vyplňování tréninku, vyplňování výstupu, seznam tréninků, detail tréninku a analýza tréninků. Pak ještě obrazovku s uživatelským nastavením, o které jsem se v průchodu aplikací nezmínil, a seznam se zdá být úplný.

Protože jsou skokanská a trenérská aplikace takřka stejné, chtěl jsem je rozdělit alespoň barvou, aby je bylo možno jednoduše rozeznat. Pro skokanskou aplikaci jsem zvolil barvu modrou a pro trenérskou červenou. Ona závodní trampolína má většinou modrou barvu, tak mi to pro skokanskou aplikaci přišlo příhodné.

Jako grafický prvek jsem použil zkosenou hranu hlavních tlačítek. V aplikaci je to jedinečný prvek a působí dynamickým dojmem. Při návrhu jsem se ztotožnil s článkem *One Big Reaseon Why iOS8 Sucks* [10], který píše o trendu zvětšování mobilních obrazovek a potřebě nových grafických návrhových vzorů, kde by hlavní ovládací prvky byly na dosah palce, tedy v dolní části obrazovky. Proto jsem se snažil všechny důležité ovládací prvky umísťovat níže, aby byly pro palec snáze dostupné, tj. čím častěji je ovladač používán, tím níže je umístěn.

3.6.2.1 Přihlášení a registrace

Tyto dvě obrazovky mají několik společných vlastností a to takových, že vypadají podobně a když je uživatel opustí, tak je buď přihlášený nebo vypnul aplikaci (což snad neudělá). Proto se na tyto dvě obrazovky koukám jako na jednu. Obě obrazovky budou mít



Obrázek 3.8: Architektonický vzor Model-View-Controller podle Apple Zdroj: [3]

tlačítko pro odeslání požadavku a políčka pro vyplnění odesílaných údajů. Jako bonus jsem do obrazovek přidal možnost registrace/přihlášení pomocí sociálních sítí Facebook[7] a Twitter[8] (obrázky 3.11 a 3.10).

Pro to, abych mohl jednoduše přecházet mezi obrazovkami přihlášení a registrací, jsem vytvořil třetí obrazovku, která funguje jako rozcestník (obrázek 3.9. Nemá žádnou funkčnost, jen odkazuje na jednotlivé obrazovky.

3.6.2.2 Hlavní obrazovka

Hlavní obrazovka vypadá v každé aplikaci jinak. Skokan zde vidí svoji stručnou analýzu za posledních 30 dnů a trenér má seznam svých spojených skokanů a jejich stručnou analýzu za posledních 30 dnů (obrázky 3.12 a 3.13). Pro zjednodušení jsem dal trenérovi na hlavní obrazovku možnost spravovat svoje skokany a odpovídat na jejich žádosti o spojení.

V dolní části obrazovky pak mají oba tlačítka pro otevření seznamu tréninků a analýzy. Skokan má navíc tlačítko pro zahájení tréninku.

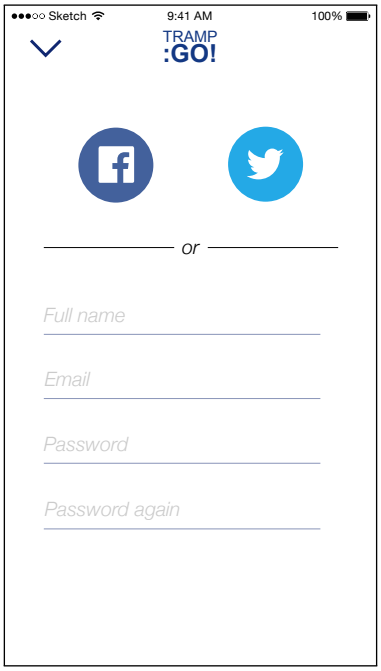
3.6.2.3 Obrazovka tréninku

Tato obrazovka slouží jak pro vyplnění tréninku, tak pro zobrazení tréninku ukončeného. Obrazovka je pro obě role velice podobná. Rozdíl je akorát v tom, že trenér nemůže do tréninku přidávat výstupy. Tím pádem se mu jeví probíhající trénink stejně jako trénink ukončený.

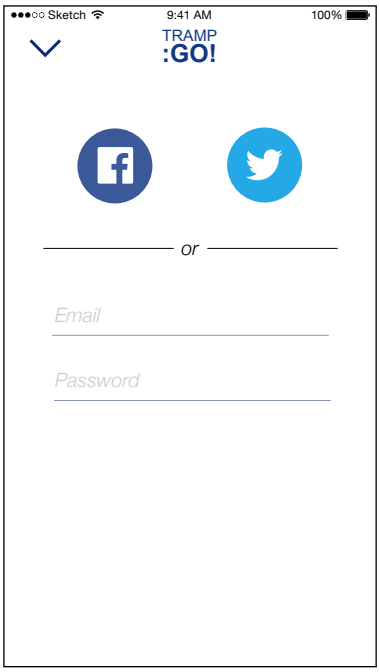
V horní části obrazovky jsou tlačítka pro ukončení nebo zrušení tréninku a základní informace o tréninku jako datum, doba tréninku, počet výstupů, obtížnost, prvky a průměrná obtížnost na prvek. Druhá část je rozdělena na dvě další části. Jednou je grafická analýza vybraného/probíhajícího tréninku (obrázek 3.14) a druhou seznam výstupů v tréninku (obrázek 3.15).

[7] <www.facebook.com>
[8] <www.twitter.com>

Obrázek 3.9: Obrazovka rozcestníku pro přihlášení a registraci



Obrázek 3.10: Obrazovka registrace



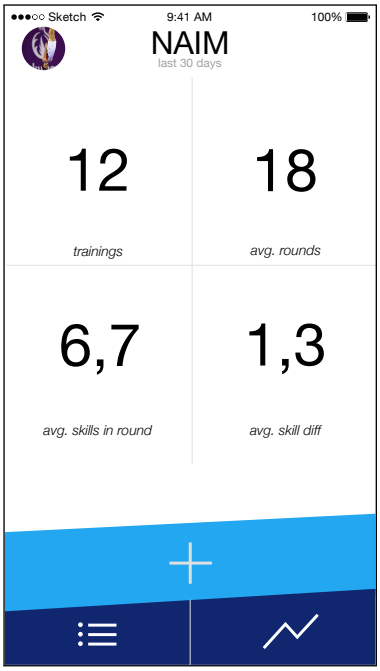
Obrázek 3.11: Obrazovka přihlášení

Skokan se ještě dostane na obrazovku pro přidání výstupu (obrázek 3.16). Proč se do této obrazovky zadávají právě tyto údaje a ne jiné popíšu v závěrečném zhodnocení cílů v sekci 6.2.1.

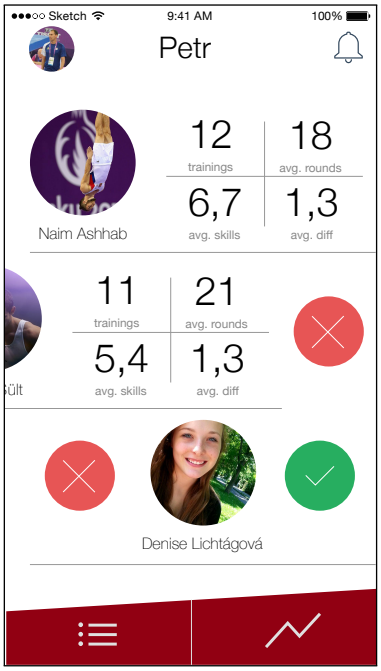
3.6.2.4 Obrazovka analýzy

Tato obrazovka slouží pro analýzu skupiny tréninků v časovém období. Nejníže jsou umístěna tlačítka pro filtrování typu hodnot v grafu. Každé barevné kolečko znázorňuje jeden typ, který se může do grafu zobrazit. Oproti tomu nejvýše jsou statické informace, se kterými uživatel nijak neinteraguje, takže je vhodné dát je na místo, kam se špatně dosahuje. Trenér má nad tlačítky pro filtr grafu ještě jedny tlačítka pro výběr skokana, kterého chce analyzovat. Uprostřed obrazovky je pak samotný graf, nad kterým je ovládání pro posun v čase, vždy však ve vybraném časovém intervalu. Ten si uživatel vybírá nad grafem. Ukázky obrazovek jsou na obrázcích 3.17 a 3.18.

Trenér nemusí vždy analyzovat pouze jednoho skokana. Může analyzovat více skokanů najednou, ale pouze s jedním typem hodnot. Například trenér může analyzovat u jednoho skokana obtížnost a počet prvků v tréninku a nebo u více skokanů počet prvků v tréninku, nikoli už obtížnost.



Obrázek 3.12: Hlavní obrazovka skokana



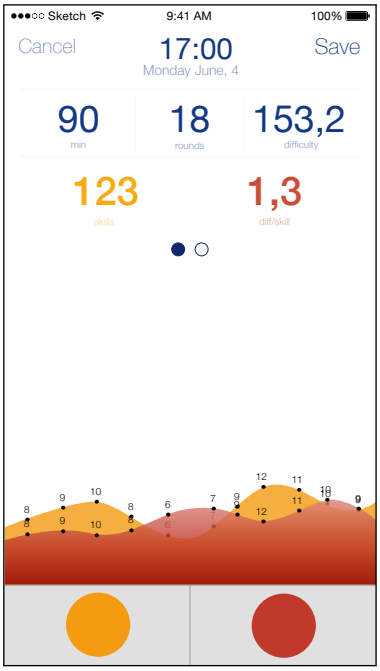
Obrázek 3.13: Hlavní obrazovka trenéra

3.6.2.5 Seznam tréninků

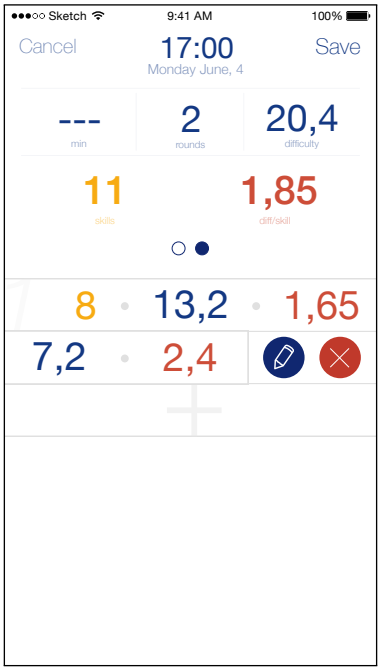
V seznamu tréninků jsem chtěl uživateli co nejvíce usnadnit práci a ukázat mu co nejvíce informací ještě dříve, než si otevře detail. Proto jsem seznam rozdělil na jednotlivé dny a v těch zobrazuji tréninky jako karty, ve kterých jsou základní informace o tréninku a analytický graf (obrázek 3.19).

Na obrazovce se tedy nejníže nachází výběr dne. Dny reprezentují kolečka a ty jsou buď prázdná nebo vyplněná, podle toho jestli v daný den proběhl alespoň jeden trénink. V pravém rohu je pak možnost rychlého vybrání dne (obrázek 3.21). Detail se otevře kliknutím na kartu tréninku. Detail pak vypadá stejně jako obrazovka tréninku.

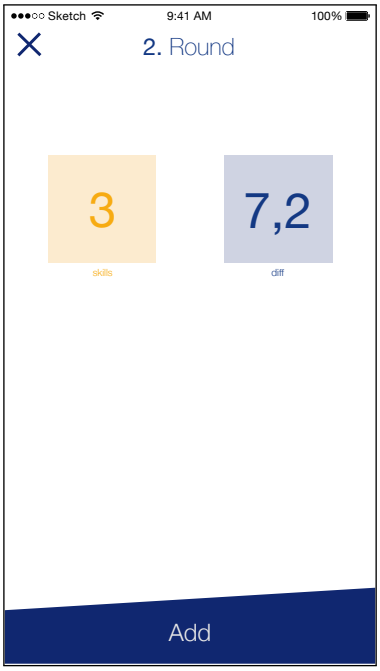
Trenér má v kartě navíc v levém horním rohu obrázek skokana, kterému trénink patří (obrázek 3.20).



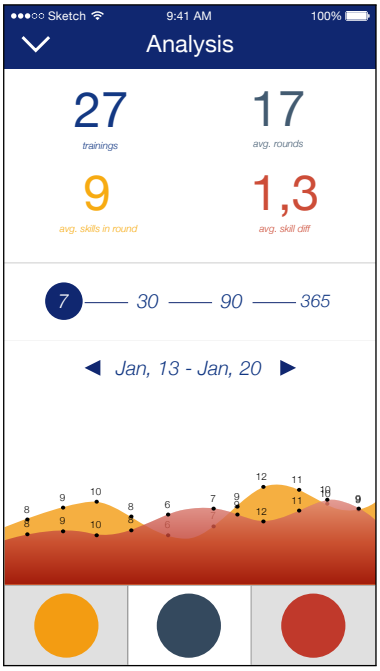
Obrázek 3.14: Obrazovka tréninku s analýzou



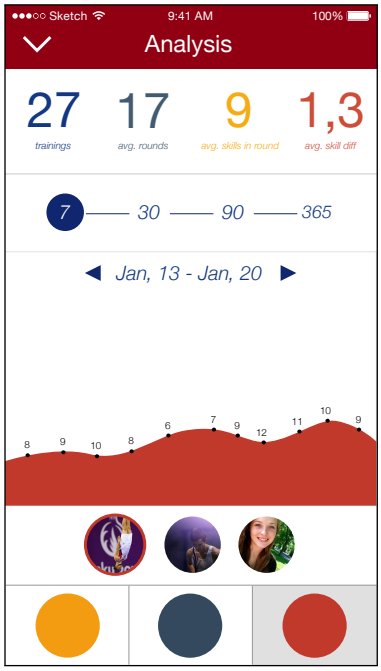
Obrázek 3.15: Obrazovka tréninku s přehledem výstupů



Obrázek 3.16: Obrazovka pro přidání výstupu



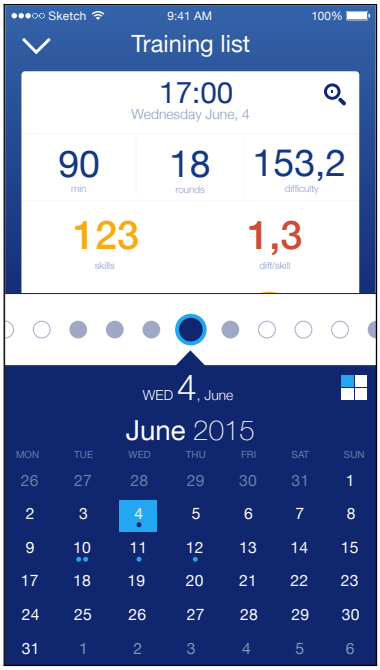
Obrázek 3.17: Obrazovka analýzy tréninků ve skokanské aplikaci



Obrázek 3.18: Obrazovka analýzy tréninků v trenérské aplikaci

Obrázek 3.19: Obrazovka seznamu tréninků ve skokanské aplikaci

Obrázek 3.20: Obrazovka seznamu tréninků v trenérské aplikaci



Obrázek 3.21: Obrazovka seznamu tréninků s rychlým výběrem data

Kapitola 4

Implementace

V této kapitole popisuji práci na implementaci prototypu. Díky kvalitní analýze a dobrému návrhu jsem nenarazil při implementaci na vážnější problém.

4.1 Mobilní aplikace

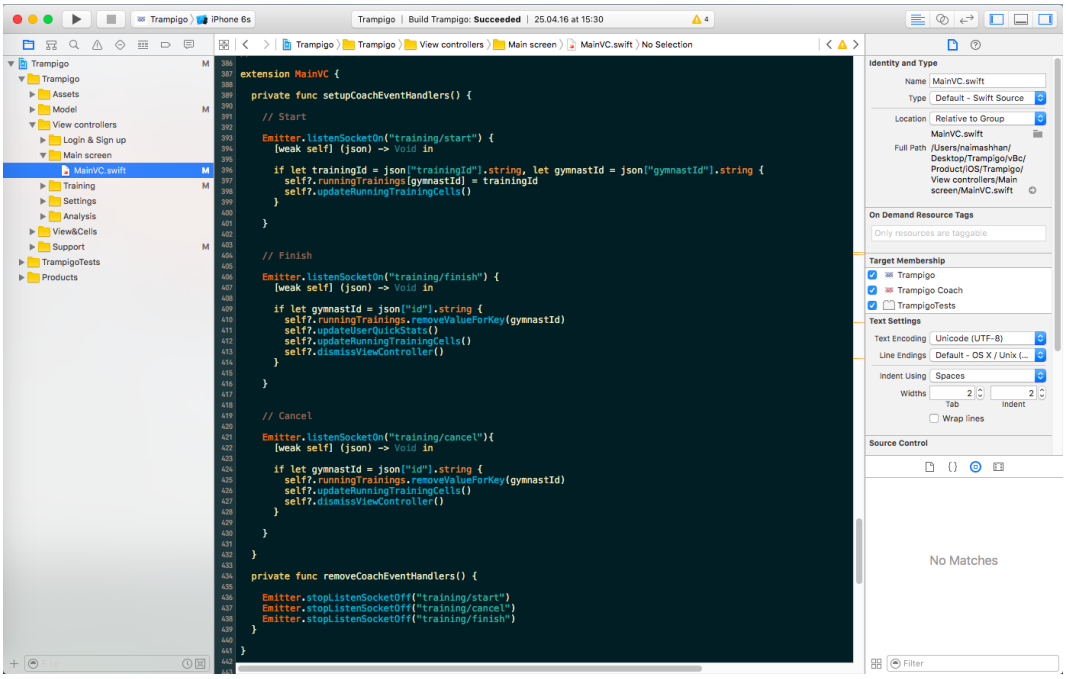
Jak už jsem se v práci zmiňoval, prototyp jsem implementoval pro operační systém iOS od společnosti Apple. Pro ně se zásadně programuje v jazycích Objective-C[1] nebo nově (od 2. Června 2014) v jazyce Swift[2]. A protože jsem fanda všeho nového, Swift jsem si vybral bez váhání. Swift, jako velice mladý jazyk, se v průběhu implementace razantně vyvíjel. Prototyp jsem začínal psát ve Swiftu 1.0 a končil ve verzi 2.1 (v době psaní práce už vyšel Swift 2.2 a byl připravený Swift 3.0).

Vývojové prostředí pro tuto platformu je aplikace Xcode[3] (prototyp byl implementován ve verzi 7.3.1, obrázek 4.1), která pomáhá vývojářům ve věcech od syntaxe jazyka, přes vytváření grafického uživatelského rozhraní, testování, napovídání a až po zjednodušování práce se systémem pro správu verzí GIT[4].

Xcode obsahuje řadu knihoven, které Apple poskytuje vývojářům třetích stran. Pro vývoj mobilních aplikací je nejzákladnější Cocoa Touch, která obsahuje Objective-C runtime, knihovnu Foundation a UIKit. Foundation poskytuje nejzákladnější třídu NSObject, která definuje chování všech dalších tříd, dále třídy které obalují primitivní typy jako NSString a NSNumber a kolekce jako NSArray, NSDictionary a NSSet. UIKit je používán pro vytváření uživatelského rozhraní a všeho co se na obrazovce vykresluje. Pomáhá zpracovávat uživatelská gesta a obsahuje základní části grafického rozhraní jako například tabulky (UITableView), tlačítka (UIButton) a textové pole (UITextField) [2].

Pro prototyp jsem využil dalších open-source knihoven, které vývojáři nabízejí pro volné používání.

[1] <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
[2] <https://developer.apple.com/swift/>
[3] <https://developer.apple.com/xcode/>
[4] <https://cs.wikipedia.org/wiki/Git>



Obrázek 4.1: Vývojové prostředí Xcode 7.3.1

Seznam používaných knihoven:

Alamofire Knihovna pro HTTP komunikaci [1]

SwiftyJSON JSON parser pro Swift [5]

SocketIO Knihovna pro WebSocket komunikaci [11]

KeychainAccess Obalová třída pro Apple Keychain [9]

Charts Knihovna pro vytváření grafů [6]

MGSwipeTableCell Knihovna přidávající funkce řádkům tabulky [4]

S prostředím Xcode je automaticky nainstalován i simulátor iOS zařízení. Simulátor dokáže simulovat zařízení iPhone, iPad a AppleWatch se všemi verzemi operačního systému. Simulátor se mi stal nepostradatelným nástrojem hlavně kvůli tomu, že Apple nedovoluje jednoduše spustit vyvíjenou aplikaci na zařízení. Pro to musí být vývojář registrovaný, mít zaplacenou vývojářskou licenci nebo být v týmu, který ji zaplacenou má a pak vytvořit mnoho certifikátů a profilů.

4.2 Server

Při výběru technologií pro serverovou část jsem měl na výběr z mnoha jazyků, např. Java, Ruby, PHP, JavaScript nebo Python. Já jsem však hledal jazyk, který by mi dovolil

jednoduše a rychle vytvořit back-end pro mé mobilní klienty.

Proto jsem se rozhodoval hlavně mezi jazyky PHP a JavaScript. Můj server nemá mít žádné webové rozhraní, takže jsem se mohl soustředit jen na technologie back-endu a front-end jsem mohl vypustit. S PHP už jsem zkušenosti měl, tak by pro mě vývoj v něm byl jednoduší. Ale nakonec jsem zvolil JavaScript a s tím i jeho serverový framework Node.js[5].

Základem Node.js je javascriptový interpret V8 z Google Chrome [13]. Node.js je s frameworkem Express[6] velice jednoduše nasaditelný a má dobrou dokumentaci. Node.js mi navíc umožňoval používat framework SocketIO[7], který jsem využil pro real-time komunikaci přes WebSocket.

Z návrhu vím, že pro ukládání dat používám NoSQL dokumentově orientovanou databázi. Jako framework jsem si vybral Mongoose[8]. Ta je postavena nad frameworkem MongoDB[9] a rozšiřuje ho.

Na serveru jsem potřeboval i rychlou paměť pro ukládání připojených uživatelů. K tomu mi dobře posloužil Redis[10], který používá rychlou paměť pro ukládání strukturovaných dat, a který jsem dostal jako doporučení od vedoucího práce.

Seznam všech používaných knihoven:

Express Webový framework pro tvorbu Node.js aplikací

SocketIO Knihovna pro WebSocket komunikaci

Redis Key-value databáze pro rychlou práci s daty [8]

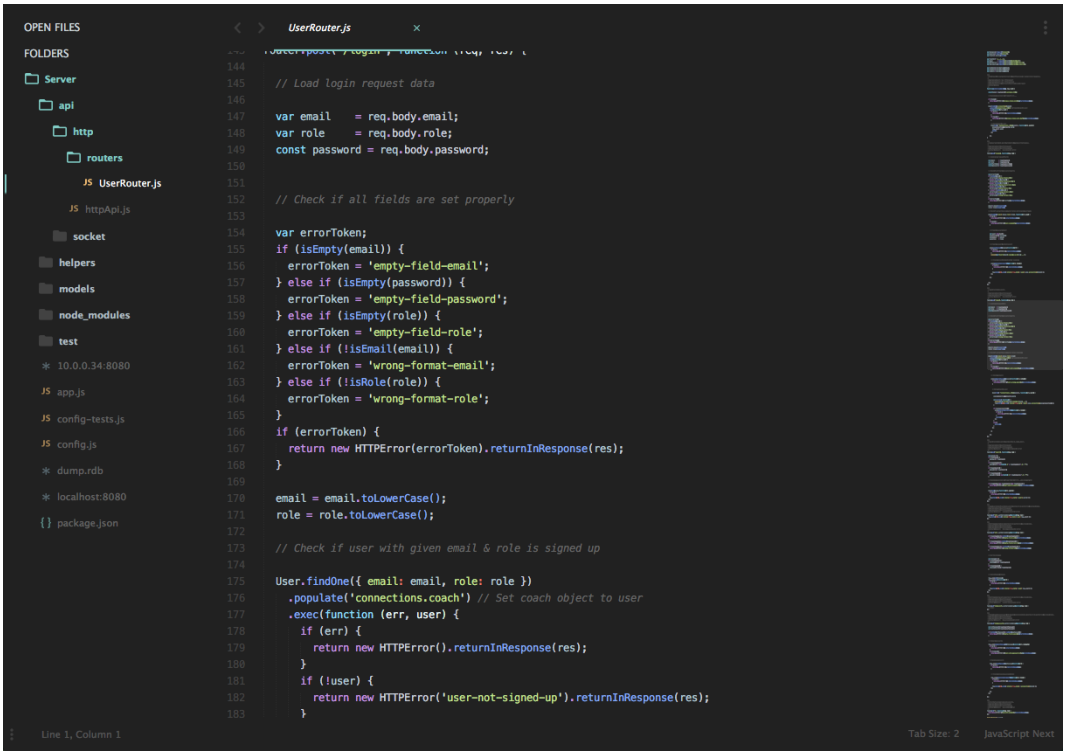
Mongoose NoSQL databáze postavená nad MongoDB

bcrypt-nodejs Node.js implementace bcryp[11] šifrovací funkci

supertest Knihovna pro testování HTTP komunikace

Pro vývoj serverové části jsem použil prostředí Sublime Text (obrázek 4.2).

[5] <https://nodejs.org/>
[6] <http://expressjs.com>
[7] <http://socket.io>
[8] <http://mongoosejs.com>
[9] <https://www.mongodb.org>
[10] <http://redis.io>
[11] <https://cs.wikipedia.org/wiki/Bcrypt>



Obrázek 4.2: Vývojové prostředí Sublime Text

Kapitola 5

Testování

V dalším kroku vývojového cyklu jsem si potřeboval ověřit, že vše funguje jak má. Proto se v této kapitole budu zabývat testováním systému.

5.1 Unit testy

Všechny modelové třídy, v mobilní aplikaci i na serveru, jsou z 90% pokryty Unit testy. Unit test je automatizované ověřování funkčnosti a korektnosti implementace aplikace [15]. Těmito testy jsem si ověřoval, že výpočty provedené modelovými třídami jsou v pořádku, že se adaptéry chovají správně ve všech případech, a že server vrací správnou odpověď na jakoukoli zprávu.

5.2 Uživatelské testy

Uživatelské testování je testováním použitelnosti aplikace. Její princip spočívá v tom, že pozoruje uživatele při používání testované aplikace [16]. Z uživatelského testování mohou vyjít cenné informace, na které samotný programátor nemusí přijít.

Protože hlavním cílem aplikace je zjednodušit vedení tréninků, potřeboval jsem i já uživatelsky otestovat prototyp aplikace a to jak skokanskou část, tak trenérskou. Proto jsem vzal hotový prototyp k nám do tělocvičny a otestoval jsem ho přímo v prostředí, ve kterém má později zjednodušovat práci.

V obou případech přeskočím část, ve které se uživatel registruje, navazuje spojení s trenérem a rovnou přejdu k práci s tréninky.

5.2.1 Uživatelské testování - Skokan

Pro testování jsem oslovil jednoho skokana z mého oddílu, kterého zde nebudu jmenovat. Vysvětlil jsem mu hlavní princip aplikace a to, že aplikace má sloužit jako elektronický deník a nástroj pro analýzu tréninků. S těmito informacemi jsem mu dal do ruky zařízení s aplikací a nechal jsem ho zapisovat si celý trénink do ní.

5.2.1.1 Vyplňování tréninku

Prvním úkolem testovaného subjektu bylo zahájit trénink. Na první obrazovce jsou 3 tlačítka, každé reprezentuje jednu logickou část aplikace. Jedno začne trénink, druhé zobrazí seznam ukončených tréninků a třetí zobrazí obrazovku analýzy. I když tlačítka nejsou popsaná, ale jejich funkčnost je reprezentovaná ikonami, skokan hned po prvním spuštění poznal, co které tlačítko dělá. Dokázal tedy bez obtíží zahájit trénink.

Po zahájení tréninku se skokanovi zobrazí obrazovka s prázdným tréninkem, který má v průběhu vyplňovat. Obrazovka je rozdělena na dvě části, v horní části jsou informace o tréninku a v dolní je analýza probíhajícího tréninku a seznam výstupů v tréninku. Nahoře na obrazovce jsou tlačítka pro zrušení a ukončení tréninku.

Skokan se nemusel ptát a sám přišel na všechna ovládání v této části aplikace. Skokan používal aplikaci celý trénink a vypisoval jednotlivé výstupy. Nedalo mu mnoho práce každé kolo vyplnit a ihned mohl vidět veškeré informace o tréninku. Skokanovi se líbala analýza na stejné obrazovce, která ho informovala o věcech jako změnu obtížnosti na prvek za výstup. Tato informace krásně ukazuje tendenci trenérů na začátku tréninku skákat jednodušší prvky než na konci.

Dále pochválil možnost ihned vidět celkový počet prvků a obtížnosti za trénink, který by normálně musel počítat ručně. Navíc je to dobrý ukazatel, jak pro skokana, tak pro trenéra, jestli byl trénink, co se týče náplně, průměrný, podprůměrný nebo nadprůměrný.

Po vyplnění celého tréninku ho skokan ukončil jediným kliknutím na tlačítko, dostal se na hlavní obrazovku a pojistkou toho, že se trénink započítal, mu byla změna hodnot v základních informacích na jeho hlavní stránce.

Poznámky od skokana jsou tedy následující:

Prázdné stavy První poznámka byla lépe znázornit stav, ve kterém nejsou žádné výstupy. Nyní je tam prázdná obrazovka, tlačítko pro vložení výstupu a graf bez dat. Bylo by vhodné do prostoru pro výstupy přidat text nebo ikonu naznačující, jak se má skokan chovat pro vytvoření výstupu.

Neustálé odemykání telefonu Jednou z věcí, které skokana obtěžovaly, bylo neustálé otevírání telefonu pro zaznamenání výstupu. Skokan měl na telefonu šestimístné číselné heslo. S tím bohužel moc neudělám, neboť iOS má velice striktní pravidla pro aplikace třetích stran, co se týče ovládání operačního systému. Je to tedy opravdu nepříjemnost, ale s tím nic neudělám.

Opakování výstupu Skokan postrádal možnost opakovat poslední výstup. Na tréninku se často výstupy opakují dvakrát až třikrát za sebou, takže tato poznámka má jistě smysl.

Nemožnost přeřazovat výstupy Skokan si stěžoval na nemožnost přeřazovat výstupy, které vytvořil. Pokud by například chtěl vložit výstup ne na poslední místo, ale na jiné, například druhé, tak by teď musel všechny výstupy, které vybranému místu následují, smazat, vložit výstup a poté znovu všechny smazané výstupy vypsat. Je to situace, kterou bych v reálu neočekával, ale musím dát skokanovi za pravdu, že aplikace by tuto možnost měla mít.

Místo barev popisky Skokan nejdříve nechápal, k čemu patří čísla v řádku výstupu. Poté si uvědomil spojitost mezi barvami informací v horní části obrazovky a barvami čísel ve výstupu. Stejné typy informací mají stejnou barvu, takže obtížnost bude mít vždy stejnou barvu, ať už se jedná o obtížnost výstupu, prvku nebo celého tréninku. Skokan poznamenal, že toto může být pro člověka se špatnou schopností rozeznávat barvy docela problém, takže by bylo lepší přidat textové popisky. To samé platí i pro typy dat v grafu. S tím můžu jedině souhlasit.

5.2.1.2 Zobrazení seznamu tréninků

Další věcí, kterou jsem testoval, bylo pohybování se v seznamu tréninků. Skokan i tento seznam našel bez problému. Seznam je tabulka všech ukončených tréninků, kde řádky tabulky jsou základními informacemi o tréninku a kliknutím na řádek se zobrazí úplný detail tréninku. Skokan s tím neměl jediný problém. Díky tomu, že je detail takřka totožný s obrazovkou pro vyplňování tréninku skokan věděl, jak se co na obrazovce ovládá.

K této části neměl žádnou poznámku.

5.2.1.3 Analýza tréninků

Skokan už si vyzkoušel analýzu jednoho tréninku nejdříve při vyplňování a poté při zobrazení detailu ze seznamu. Proto pro něj ani analýza více tréninků nebyla nic nového. Jediné co přibylo, je výběr časového úseku. Pro zjednodušení jsem dal na výběr ze čtyř možností a to 7, 14, 30 a 90 dní, počítáno vždy od dne, kdy je analýza zobrazena.

Ani k této části neměl skokan žádné nové poznámky.

5.2.2 Uživatelské testování - Trenér

Jako další subjekt testování jsem si vybral vlastního trenéra ve skocích na trampolíně a skokan jsem byl v tuto chvíli já sám. Trenérská aplikace je jiná v tom, že neumožňuje vytvářet tréninky a tím se stává jen prostředkem pro zobrazení. Trenér nemůže žádnou akcí zasahovat do průběhu tréninku. S těmito informacemi jsem dal zařízení s aplikací do ruky trenérovi a test začal.

5.2.2.1 Probíhající tréninky

Po přihlášení má trenér namísto základních informací seznam skokanů a u nich jejich základní informace. V tomto seznamu se trenérovi zobrazí i stav, kdy je skokan v právě běžícím tréninku. Trenérovi chybí tlačítko pro zahájení tréninku, ale druhá dvě mu zůstala.

Ve chvíli, kdy jsem tedy zahájil trénink, se tato informace ukázala trenérovi na zařízení jako zelený pásek u mého řádku. Samotného ho napadlo, že po kliknutí na tento označený řádek uvidí detail tréninku, který právě vytvářím. K detailu měl stejnou poznámku jako prve skokan a to, že by se měl lépe zpracovat stav, kdy nejsou vyplněné žádné výstupy. Jinak byl trenér příjemně překvapený, že se data aktualizují automaticky.

Poznámky trenéra k probíhajícímu tréninku byly tedy takovéto:

Prázdné stavy Stejně jako skokan, i trenér by uvítal jasnější zobrazení pro stav, který nemá žádné výstupy.

Měření času Ve skocích na trampolíně se výška skoku neměří na jednotky délky, ale na jednotky času. Je daleko jednoduší změřit dobu letu, než do jaké výšky skokan vyskočil. Trenér by tedy uvítal, kdyby byly v aplikaci stopky, které by rovnou přenášeli změřený čas skokanovi do aplikace a ten si ho mohl přiřadit k výstupu. Přibyla by další cenná informace.

Ukončení tréninku V prototypu je pravidlo, že když skokan ukončí nebo zruší trénink, tak se zavře okno tréninku i trenérovi, pokud ho má zobrazené. Trenér dobře poznamenal, že toto chování se hodí pro zrušený trénink, ale u řádně ukončeného by se okno mohlo nechat s poznámkou, že trénink byl dokončen. Když se například trenér zrovna nedívá na obrazovku nebo má telefon zamknutý, může být zmatený, kam se trénink poděl.

5.2.2.2 Seznam tréninků

Trenér má na své hlavní straně dvě stejná tlačítka pro zobrazení seznamu tréninků a analýzy tréninků, stejně jako skokan. Ani trenér neměl problém rozeznat, co tato dvě tlačítka reprezentují. V trenérském seznamu tréninků je jediná změna oproti skokanovi a to ta, že trenér vidí seznam všech svých skokanů. Pro rozeznání, který trénink je čí, jsou u jednotlivých tréninků fotky skokanů.

Trenér, stejně jako skokan, se v tomto seznamu dokázal jednoduše pohybovat a nečinilo mu žádný problém vybrat přesně ten trénink, který chtěl. Po zobrazení detailu viděl tutéž obrazovku, jakou pozoroval při běžícím tréninku.

Poznámky k této části:

Oddělení tréninků Pro rozdělení tréninků je u každého tréninku fotka, kterou si skokan přiřadil. Prní trenérova otázka byla, co když si skokan fotku nenastaví nebo budou mít dva skokani fotku stejnou. Tento problém v prototypu není řešený.

Filtr Trenér postrádal možnost zobrazit si seznam pouze pro jednoho skokana. Obdobně možnost pro dva a tak dále.

5.2.2.3 Analýza tréninků

Trenér může, narozdíl od skokana, analyzovat více než jednoho člověka. Proto musí mít na obrazovce analýzy další ovladač pro přepínání skokanů. Na druhou stranu je vhodné porovnávat skokany mezi sebou, proto jsem do trenérské analýzy přidal pár pravidel pro zobrazování analýzy. Trenér může u jednoho skokana analyzovat všechny vlastnosti tréninků (tak jak je to u skokana) nebo jednu vlastnost u více skokanů (například celkový počet prvků za trénink napříč všemi skokany).

Trenér na obrazovce analýzy poznal funkčnost všech ovládacích prvků, ale z pravidel pro zobrazování analýz byl zmatený. Nejdříve mu nedávalo smysl, proč nemůže do analýzy přidat dalšího skokana a až po vysvětlení mu pravidla začala dávat smysl. Určitě by na tato pravidla po chvíli používání přišel sám, ale tím, že mu aplikace nedávala žádnou odezvu, proč nechce dalšího skokana nebo vlastnost přidat, to nebylo na první pohled zřejmé.

Kapitola 6

Závěr

Na závěr shrnu celý postup práce na projektu, plnění cílů a zamyslím se nad dalšími možnostmi práce do budoucna.

6.1 Postup

I když jsem od začátku vyvíjel prototyp, tak jsem jako metodiku používal vodopádový přístup k vývoji softwaru. Ten rozděluje vývoj na 5 hlavních částí, které jdou za sebou tak, jak by tekla voda v kaskádovém vodopádu.

6.1.1 Analýza

První věcí ve vývojovém cyklu byla analýza. V analýze jsem zkoumal současný stav vykazování tréninků ve skocích na trampolíně, stanovil jsem si role, definoval jsem vlastnosti, které by měl ideální systém splňovat a tyto vlastnosti jsem převzal jako cíle pro svůj projekt. Poté jsem mohl definovat základní doménové třídy, ze kterých jsem vytvořil diagram znázorňující jejich závislosti mezi sebou. V neposlední řadě jsem se rozhodoval mezi možností vytvořit jednu mobilní aplikaci pro obě role nebo pro každou roli vlastní. Analyzoval jsem potřeby serveru a komunikace mezi serverem a mobilními klienty a nakonec jsem vytvořil model nasazení, zachycující celý systém.

6.1.2 Návrh

Dalším krokem ve vývojovém cyklu byl návrh systému. V návrhu jsem si nejdříve důkladně popsal entity, které budu v projektu používat a způsob jejich ukládání. Zabýval jsem se možnostmi databáze a pokračoval jsem návrhem serveru, který bude databázi spravovat. Pokračoval jsem navržením rozhraní, přes které budou mobilní klienti a server komunikovat, až jsem došel k návrhu samotných mobilních aplikací.

Do prototypu jsem nakonec změnil grafické zobrazení seznamu tréninků. Kvůli náročnosti navrženého řešení a faktu, že změna nemá žádný vliv na funkčnost aplikace, jsem z návrhu slevil a seznam zobrazuji jako tabulku, kde každý řádek reprezentuje jeden trénink.

6.1.3 Implementace

V implementační části jsem popisoval, jaké technologie jsem používal pro každou komponentu systému, jaké frameworky nebo externí knihovny jsem pro implementaci použil a jejich stručný popis.

6.1.4 Testování

V poslední části vývoje jsem testoval, zda to, co jsem naprogramoval, funguje a má smysl. Většinu modelových tříd jsem popsal unit testy, abych si ověřil jejich bezchybnost. Dále jsem absolvoval uživatelské testování pro lepší pochopení uživatele, jeho potřeb a chování v aplikaci. Ověřil jsem si tím, že aplikace bude v reálu užitečná.

6.2 Zhodnocení cílů

Všechny cíle stanovené v analýze jsem se snažil co nejlépe naplnit. U některých se mi to povedlo více a u některých méně.

6.2.1 Zjednodušení vyplňování tréninků

Z tohoto cíle jsem kvůli náročnosti musel nakonec slevit. Pro úplné zjednodušení vyplňování tréninků bych potřeboval, aby uživatel vytvářel výstupy vybíráním prvků. Bohužel jsem nenašel jednoduchý způsob, jak bych toho dosáhl. Kvůli tomu, že je škála prvků ve skocích na trampolíně teoreticky nekonečná, bych musel přidat možnost přidat, odebrat a měnit prvky, které má uživatel lokálně uložené v zařízení. Dále bych musel nechat uživatele, aby si z tohoto seznamu prvky vybíral. Při statickém a nijak neoptimalizovaném seznamu by to při jeho velikosti nebylo nijak velké zjednodušení. Naopak by se dalo hovořit i o ztížení vyplňování tréninků.

Proto jsem do prototypu zvolil vytváření výstupů zadáním celkového počtu skoků a obtížnosti ve výstupu. Je to dočasné řešení, které je na půl cesty tam, kam se v budoucnu potřebuje systém dostat.

6.2.2 Zobrazování probíhajících tréninků v reálném čase trenérovi

Tento cíl se mi podařilo naplnit na výbornou. Díky WebSocket připojení obou mobilních aplikací se mi podařilo změnu vyvolanou skokanem ihned propagovat do aplikace trenéra. Nejenže trenér vidí, který z jeho skokanů právě trénuje a který ne, ale při zobrazení konkrétního tréninku se mu automaticky aktualizuje i seznam výstupů, všechny hodnoty jako celkový počet skoků nebo obtížnosti a graf ukazující průběh několika vybraných atributů.

6.2.3 Ukládání všech ukončených tréninků

U tohoto cíle bylo potřeba zpracování databáze na serveru, kde se všechny tréninky ukládají. To se podařilo, a tak byl i tento cíl naplněn. Bohužel jsem musel kvůli náročnosti grafického návrhu z výsledného zpracování seznamu tréninků slevit a zobrazil jsem ho jako obyčejnou tabulku.

6.2.4 Analýza jednoho vybraného tréninku

Při naplňování tohoto cíle nebyl největší problém vytvoření analytických dat, ale způsob jejich zobrazování. Zvolil jsem způsob pomocí grafu, ve kterém se může najednou zobrazovat více typů informací. To pro lepší hledání závislostí mezi nim. Každý typ se dá vypnout tak, že se nakonec v grafu zobrazuje jen jeden.

6.2.5 Analýza skupiny tréninků ve vybraném časovém úseku

Tento cíl je jen rozšíření cíle předchozího. V podstatě zde akorát dojde k přidání několika dalších informací k analýze. Proto i tento cíl byl, po splnění cíle předchozího, splněn.

6.3 Vize do budoucna

Jsem přesvědčen o tom, že tato práce poslouží jako dobrý prototyp pro elektronické vykazování a analyzování tréninků ve skocích na trampolíně. Jsem velice rád, že jsem se v práci mohl zamyslet nad tolika problémy, které by mě na první pohled nemusely napadnout.

Do budoucna bych chtěl nahradit dočasné řešení u cíle *Zjednodušení vyplňování tréninků*. Jednou z možností, jak vylepšit řešení v prototypu, je zobrazovat optimalizovaný seznam podle nejčastěji vybíraných prvků (tento seznam by se *učil*, jaké prvky uživatel nejčastěji vybírá a podle toho by seznam upravoval) nebo uživatelského nastavení.

Další možností by mohlo být hlasové ovládání. Názvy prvků jsou na trampolíně dobře strukturované a dalo by se podle hlasového vstupu dobře rozeznat, o který prvek se jedná. Nakonec by skokan mohl celý výstup nadiktovat a systém by ho rozeznal a zaznamenal. Toto je myslím řešení, které by vykazování usnadnilo úplně nejvíce.

Literatura

[1] Alamofire Software Foundation. *Alamofire - Elegant Networking in Swift* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/Alamofire/Alamofire>.

[2] Apple Inc. *Cocoa (Touch)* [online]. 2015. Dostupné z: <https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html>.

[3] Apple Inc. *Model-View-Controller* [online]. 2015. [cit. 26. 4. 2016]. Dostupné z: <https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html>.

[4] FERNANDEZ, I. *MGSwipeTableCell* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/MortimerGoro/MGSwipeTableCell>.

[5] FU, R. *SwiftyJSON* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/SwiftyJSON/SwiftyJSON>.

[6] GINDI, D. C. – JAHODA, P. *ios-charts* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/danielgindi/ios-charts>.

[7] GUPTA, A. REST vs WebSocket Comparison and Benchmarks. Dostupné z: <http://blog.arungupta.me/rest-vs-websocket-comparison-benchmarks/>.

[8] JIRÁSEK, P. *Poznejte Redis - key-value databázi* [online]. 2013. [cit. 20. 5. 2016]. Dostupné z: <https://petrjirasek.cz/blog/poznejte-redis>.

[9] KATSUMI. *KeychainAccess* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/kishikawakatsumi/KeychainAccess>.

[10] KOVALENKO, D. *One Big Reason Why iOS8 Sucks* [online]. 2015. [cit. 5. 5. 2016]. Dostupné z: <https://medium.com/user-experience-design-1/the-big-reason-why-ios8-sucks-73ac7925c626>.

[11] LITTLE, E. *Socket.IO-Client-Swift* [online]. 2016. [cit. 20. 5. 2016]. Dostupné z: <https://github.com/socketio/socket.io-client-swift>.

[12] MALÝ, M. REST: architektura pro webové API, 2009. Dostupné z: <https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>.

[13] NEŠETŘIL, J. [online]. 2010. Dostupné z: <https://www.zdrojak.cz/clanky/javascript-na-serveru-zaciname-s-node-js/>.

[14] PECINOVSKÝ, R. *Návrhové vzory*. Computer Press, 2013.

[15] Přispěvatelé Wikipedie. *Unit testing* [online]. 2014. [cit. 24. 4. 2016]. Dostupné z: <https://cs.wikipedia.org/wiki/Unit_testing>.

[16] Přispěvatelé Wikipedie. *Uživatelské testování* [online]. 2016. [cit. 24. 4. 2016]. Dostupné z: <https://cs.wikipedia.org/wiki/Uživatelské_testování>.

[17] ROUSE, M. TCP (Transmission Control Protocol), 2014. Dostupné z: <http://searchnetworking.techtarget.com/definition/TCP>.

[18] WEIHER, M. *Model Widget Controller (MWC) aka: Apple* [online]. Dostupné z: <http://blog.metaobject.com/2015/04/model-widget-controller-mwc-aka-apple.html>.

Příloha A

Obsah přiloženého CD

- `abstract`
  - `abstract-cz.txt` ..... abstrakt práce v českém jazyce
  - `abstract-en.txt` ..... abstrakt práce v anglickém jazyce
- `readme.txt` ..... popis obsahu CD
- `src`
  - `mobile-application.zip` ..... zdrojové kódy mobilní části aplikace
  - `server.zip` ..... zdrojové kódy serverové části aplikace
  - `ashhab-thesis-2016.zip` ..... BP ve formátu LaTeX
- `text`
  - `ashhab-thesis-2016.pdf` ..... text BP ve formátu PDF

Příloha B

API

B.1 HTTP

Parametry s otazníkem jsou nepovinné.

URI	Metoda	Parametry	Popis
/user/signup	POST	fullName, email, password	Registrace nového uživatele
/user/login	POST	email, password, role	Přihlášení registrovaného uživatele
/user/search	POST	email?, id?, fullName?	Vyhledávání uživatelů
/user/me	GET	accessToken	Informace o přihlášeném uživateli
/user/me	PUT	accessToken, fullName?, email?	Editace učivatelského účtu
/user/me/password	POST	accessToken, old, new	Změna uživatelského hesla (vygeneruje nový accessToken).
/user/me/avatar	PUT	accessToken, avatar?	Editace uživatelského profilového obrázku

Tabulka B.1: Tabulka HTTP API metod

B.2 WebSocket

URI	Parametry	Popis
/training	Training.id	Vrácí konkrétní trénink podle ID. Uživatel na něj musí mít oprávnění.
/training/list	fromDate, toDate, User.id	Vrátí seznam všech tréninků, na které má uživatel oprávnění (Vlastní je nebo jsou jeho skokanů) nebo jen daného uživatele.
/training/start	date	Vytvoří a zahájí trénink
/training/cancel	Training.id	Přeruší a smaže probíhající trénink
/training/finish	Training.id	Ukončí probíhající trénink
/training/add/round	Training.id, skills, difficulty	Přidá výstup do probíhajícího tréninku
/training/delete/round	Training.id, index	Smaže výstup z probíhajícího tréninku
/training/edit/round	Training.id, index, skills, difficulty	Upraví výstup v probíhajícím tréninku
/training/running	[User.id]	Vráti seznam probíhajících tréninků u vybraných uživatelů

Tabulka B.2: Tabulka WebSocket API metod