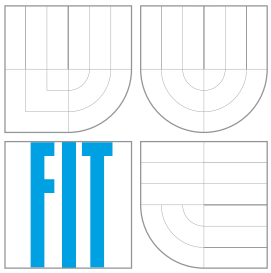
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO VYHLEDÁVÁNÍ PARTNERŮ DO POSILOVNY (IOS)

APPLICATION FOR SEARCHING FOR FITNESS BUDDIES (IOS)

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DOMINIK PLŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Ing. ADAM HEROUT, Ph.D.

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Plšek Dominik**
Obor: Informační technologie
Téma: **Aplikace pro vyhledávání partnerů do posilovny (iOS)**
Application for Searching for Fitness Buddies (iOS)
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte problematiku tvorby mobilních aplikací.
2. Prostudujte a popište principy existujících aplikací řešících problém podobný tomu v zadání.
3. Navrhněte základní principy uživatelského rozhraní vytvářené aplikace.
4. Navrhněte alternativní přístupy k řešení jednotlivých prvků uživatelského rozhraní.
5. Implementujte prototyp vytvářené aplikace - zaměřte se pouze na funkčnost spadající do "minimal viable product".
6. Na základě zpětné vazby uživatelů a statistiky používání rozvíjejte prototyp aplikace.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3,
- značné rozpracování bodů 4 a 5.

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: **Herout Adam, doc. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Využitím mobilních aplikací pracujících na principu geografické blízkosti a sdílené ekonomiky vznikly velmi úspěšné projekty, například v oblasti taxi služby. Uvedené principy se dají aplikovat i na jiné oblasti lidského života. V rámci své práce jsem se zaměřil na využití zmíněných metod v oblasti fitness. Značná část potenciálních zájemců o fitness aktivity je odrazována skutečností, že nemají s kým tyto aktivity sdílet. Cílem práce je vytvoření multiplatformní mobilní aplikace Gymber, která napomáhá nalezení partnera nebo trenéra k běhání, posilování či jiné fitness aktivitě. Gymber umožní uživatelům sdílet a reagovat na nabídky k fitness aktivitám a zobrazí seznam míst, která jsou k těmto účelům vhodná, informuje o případných otevíracích hodinách, speciálních akcích a novinkách ve fitness centrech. Práce předkládá témata návrhu uživatelského rozhraní, implementaci na platformě iOS a návrh serverové části služby a API podporujícího multiplatformnost mobilní aplikace Gymber.

Abstract

Many successful projects are based on mobile applications using geographic position of the user and shared economy, mostly in transport market. These principles can be used in other fields of human life as well. As part of my thesis, I focused on applying these methods to the fitness market. A considerable part of potential customers is discouraged from exercising regularly by the fact that they do not have anyone to share these activities with. The purpose of the thesis is to create a mulitplatform mobile application Gymber which helps users searching for a partner or a trainer for running, weight-lifting or other desired fitness activity. Gymber allows users to share and respond to offers for fitness activity, displays a feed of appropriate places for these activities, informs users about places' opening hours, special events and news. The thesis also describes topics about user interface design, iOS application implementation and designing backend API for supporting other mobile platforms, too.

Klíčová slova

mobilní aplikace, iOS, uživatelské rozhraní, Ruby on Rails, aplikační rozhraní

Keywords

mobile application, iOS, user interface, Ruby on Rails, application programming interface

Citace

PLŠEK, Dominik. *Aplikace pro vyhledávání partnerů do posilovny (iOS)*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Herout Adam.

Aplikace pro vyhledávání partnerů do posilovny (iOS)

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal. Paralelně byla vytvářena bakalářská práce Filipa Kalouse, která řeší obdobnou aplikaci se zaměřením pro mobilní platformu Android. Práce využívají sdílenou serverovou část a aplikační rozhraní.

.....
Dominik Plšek
17. května 2016

Poděkování

Rád bych poděkoval za odbornou pomoc vedoucímu práce, prof. Ing. Adamu Heroutovi, Ph.D.

Obsah

Kapitola 1

Úvod

Celosvětově je přes 131 700 000[1] lidí, kteří si zakoupili členství v posilovně nebo fitness klubu. Z uvedeného počtu lidí přibližně 67%[1] nikdy své členství nevyužije. Co vede tyto lidi k tomu, aby zaplatili poměrně drahé členství v posilovně nebo fitness klubu a následně vůbec zakoupené permanentky nevyužili?

Při hledání odpovědi na tuto otázku lze specifikovat nejčastěji se opakující důvody, proč lidé nenavštěvují fitness kluby a posilovny nebo ruší své členství v těchto zařízeních. Pokud pomineme překážky ze strany rodiny a pracovního nasazení, zjistíme, že jedním z hlavních důvodů je fakt, že aktivity nemají s kým sdílet a cena za členství ve fitness zařízeních je příliš vysoká[2]. V České republice, až na výjimky, obnáší členství v klubu pouze vstup do fitness zařízení a zákazník vyžadující pomoc odborného trenéra, je nucen doplácet další poplatky za hodiny s lektorem. Problém lze v dnešní době chytrých zařízení a platformách založených na sdílené ekonomice řešit.

Mobilní aplikace Gymber vychází z konceptu, že každý člověk se zkušenostmi s cvičením se může stát trenérem. Nabízí možnost vytvoření neplacené i placené nabídky ke cvičení. Zároveň slouží jako reklamní prostor pro zařízení určená k fitness aktivitám, například prostřednictvím zasílání novinek a pozvánek na pořádané akce. Uživateli nabízí přehled míst v geografické blízkosti s účelem motivovat jej k fitness aktivitě.

Gymber podporuje společné aktivity uživatelů ve cvičení, běhání, bruslení a jiných. Využitím konceptu začlenění uživatele se zkušenostmi z fitness do role trenéra dochází k potencionálnímu snížení ceny za lekci a zpřístupnění placených lekcí širší veřejnosti. Díky silnému sociálnímu prvku Gymber spojuje lidi, kteří nemají s kým cvičit a přivádí je společně k fitness aktivitám. Zároveň se staví do pozice agregátoru lokalit vhodných k fitness aktivitám a prezentuje je uživatelům za účelem motivovat ke zdravému životnímu stylu.

Práce představuje návrh a implementaci mobilní aplikace Gymber. Jednotlivé kapitoly popisují mobilní operační systémy, programování pro platformu iOS a návrh uživatelského rozhraní, serverové části služby a aplikačního rozhraní. Část práce zahrnuje implementaci uživatelského rozhraní, persistenci dat a komunikaci mobilní aplikace se vzdáleným serverem na systému iOS.

[1] http://www.statisticbrain.com/gym-membership-statistics/
[2] http://www.fitnessforweightloss.com/gym-statistics-members-equipment-and-cancellations/

Kapitola 2

Mobilní operační systémy

Mobilní operační systém je operační systém, optimalizovaný pro chytré telefony, tablety, PDA a jiná mobilní zařízení. Hlavním rozdílem mezi operačním systémem určeným pro stolní počítače a mobilním operačním systémem, je podpora mobilních funkcí – ovládání pomocí dotyku, GPS navigace, přístup k fotoaparátu a videokameře, Bluetooth, rozpoznávání řeči a jiné. Zařízení s podporou komunikačních prostředků – volání, posílání SMS zpráv, obsahují více operačních systémů. Operační systém reálného času (real-time operating system, RTOS) je specializovaný nízkoúrovňový systém spravující přijímač signálů a hardware zaměřený na komunikační schopnosti zařízení. Grafické uživatelské rozhraní, spravování souborového systému a další funkce jsou řízeny mobilním operačním systémem, např. iOS, Android a další. Přehled a popis jednotlivých mobilních operačních systémů obsahuje kapitola 2.1.

2.1 Přehled mobilních operačních systémů

Kapitola se zabývá přehledem mobilních operačních systémů a shrnuje jejich rozdíly a specifika. V přehledu jsou zahrnuty pouze významnější systémy, které na trhu mobilních operačních systémů zaujímají přední pozice. V přehledu nejsou uvedeny operační systémy s nevýznamným podílem na trhu.

Android je mobilní operační systém založený na jádře Linux distribuovaný jako open–source. Vznik operačního systému Android započal ve firmě Android Inc., která byla odkoupena společností Google v roce 2005. Roku 2007 vzniklo konsorcium technologických firem Open Handset Aliance, které představilo společně s mobilním operačním systémem Android otevřený standard pro mobilní zařízení. I přes to, že je Android open–source, v mnoha případech mobilní zařízení obsahují proprietární aplikace. Ke stažení aplikací slouží na platformě Android internetový obchod Google Play. Podíl Androidu na trhu mobilních operačních systémů činí 80.7%[1] a tím je nejrozšířenějším operačním systém na mobilních zařízeních.

iOS, původně iPhone OS, je mobilní operační systém od firmy Apple. Systém je podporován na zařízení iPhone a iPad. iOS vychází z operačního systému Mac OS X pro stolní počítače. iOS je pouze podmnožinou systému Mac OS X, ale přidává podporu pro dotykové ovládání a další funkce pro mobilní zařízení. S operačním systémem Mac OS X sdílí systém iOS jádro **XNU** a některé knihovny. Architektura systému iOS je blíže popsána v kapitole 3.1. iOS je druhým nejrozšířenějším mobilním operačním systémem s podílem na trhu 17.7%[1]. I přes to, že je systém Android rozšířenější a počet stažení aplikací převyšuje sta-

[1] http://www.macrumors.com/2016/02/18/ios-android-market-share-q4-15-gartner/

žení na platformě iOS o více jak 90%, roční příjem z aplikací v App Store, internetovém obchodě na operačním systému iOS, převyšuje Google Play o 80%[2][3].

Windows Phone, mobilní operační systém od firmy Microsoft, je nástupce systémů Windows Mobile a Zune. Uživatelské rozhraní operačního systému Windows Phone je založeno na systému Modern UI, které využívá i operační systém Windows pro stolní počítače. Pro systém Metro je typický dlaždicový design. Windows Phone 7 byl představen v roce 2010 a po verzi 8.1 přejmenován a nahrazen univerzálním systémem Windows 10. Na trhu s mobilními operačními systémy zaujímá systém Windows 1.1%[1].

BlackBerry je operační systém pro stejnojmenná mobilní zařízení, vyvíjený firmou BlackBerry Limited, dříve Research In Motion Limited (RIM). Mobilní zařízení BlackBerry byla známá svými fyzickými QWERTY klávesnicemi a důrazem na šifrovanou komunikaci. Nové generace mobilních zařízení od BlackBery obsahují dotykové virtuální klávesnice a vybrané modely jsou distribuované s operačním systémem Android. BlackBerry je po Windows čtvrtým nejrozšířenějším mobilním operačním systémem s podílem 0.2%[1].

[2] https://ia.net/writer/updates/the-best-things-in-android-are-freewith-in-app-purchases
[3] http://9to5mac.com/2016/01/20/app-store-ios-downloads-vs-android-revenue/

Kapitola 3

Programování mobilních aplikací na platformě iOS

Od svého uvedení v roce 2007 se iOS stal mobilním operačním systémem s více než 5,5 miliony[1] vývojářů, kteří vytvořili přes 1,5 milionu aplikací[2]. Počet zařízení, která pracují na celkově čtyřech operačních systémech – OS X, iOS, watchOS a tvOS od společnosti Apple přesáhl 1 miliardu[3]. Zmíněný počet zařízení představuje obrovský potenciál pro vytváření aplikací, které mohou komunikovat napříč spektrem produktů společnosti Apple. V této kapitole bude představena architektura a základní vývojové prostředky pro systém iOS. Zmíněné informace nejsou specifické pouze pro mobilní operační systém iOS, ale jsou aplikovatelné i pro ostatní operační systémy společnosti Apple.

3.1 Architektura systému iOS

Operační systém iOS lze nalézt v mobilních zařízeních společnosti Apple – *iPhone*, *iPad* a *iPod Touch*, kde obsluhuje hardware zařízení a poskytuje technologie nutné k implementování nativních aplikací [4]. Obrázek 3.1 znázorňuje jednotlivé úrovně operačního systému iOS.

Nejnižší vrstva **Core OS** obsahuje jádro operačního systému *Mach 3.0* a obsluhuje nízko úrovňové operace [5]. Důležitou částí Core OS vrstvy je zabezpečení a řízení spotřeby. Od sedmé generace mobilního telefonu iPhone, verze iPhone 5S, ve vrstvě Core OS dochází k autentizaci uživatele pomocí čtečky otisků prstů *Touch ID*. Vrstva zodpovídá za práci se souborovým systémem, certifikáty a BSD schránkami.

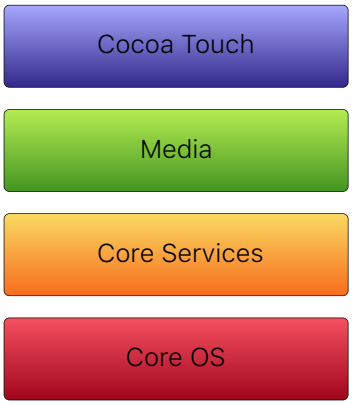
Vyšší vrstva **Core Services** se stará o síťovou komunikaci a podporu vláken. Core Services obsahují základní aplikační rámce systému iOS – Core Foundation a Foundation, které definují elementární datové typy, využívané všemi mobilními aplikacemi [6]. Kromě podpory síťové komunikace se Core Services starají o lokační služby a práci s databází *SQLite*.

Ve vrstvě **Media** se nachází podpora pro audiovizuální technologie. Vývojář má možnost využívat aplikačních rámců z vrstvy Media – Core Audio a Core Animation, pro vytváření zvukových efektů a animací ve své aplikaci. Ve vrstvě Media je zakomponovaná podpora

[1] http://www.visionmobile.com/blog/2014/12/can-app-stores-sustain-5-5-million-developers/

[2] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[3] http://www.apple.com/pr/library/2016/04/18Apples-Worldwide-Developers-Conference-Kicks-Off-June-13-in-San-Francisco.html



Obrázek 3.1: Jednotlivé části systému iOS. [4]

pro vykreslování 2D objektů pomocí aplikačního rozhraní *Quartz* a *OpenGL* pro 2D, resp. 3D vykreslování [7].

Nejvyšší vrstva **Cocoa Touch** obsahuje klíčové aplikační rámce, pomocí kterých programátor definuje vzhled aplikace. Rámce poskytují podporu pro technologie, například podporu víceúlohového (multitasking) systému, zadávání vstupu prostřednictvím dotykové obrazovky, notifikací a jiné.

3.2 Použití návrhových vzorů v systému iOS

K vývoji mobilních aplikací na platformě iOS je nezbytně nutné pochopení problematiky objektově orientovaného programování. Aplikační rozhraní, se kterými přijde vývojář do styku, jsou implementována v objektově orientovaných jazycích a dodržují objektově orientované paradigma. I při návrhu uživatelského rozhraní v Interface Builderu, jenž je součástí vývojového prostředí Xcode, dochází na pozadí bez vědomí programátora k vytváření objektů a zasílání zpráv těmto objektům k jejich konfiguraci. Právě díky silnému důrazu na objektové paradigma je možné tuto funkcionalitu rozšiřovat a definovat vlastnosti jednotlivých prvků. Nejpoužívanějším návrhovým vzorem při vývoji na platformě iOS je Model–View–Controller [2].

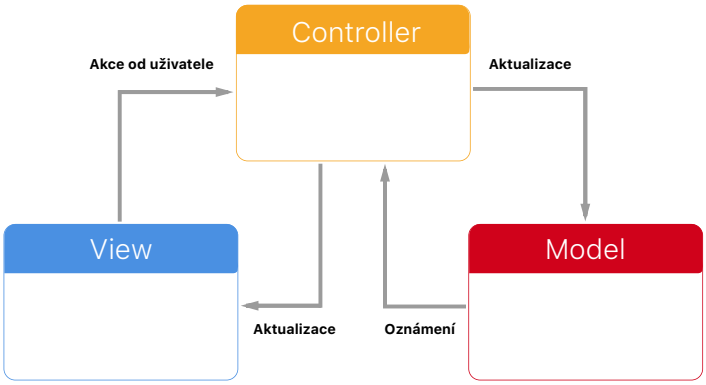
Model–View–Controller, zkráceně „*MVC*“, je návrhový vzor, který rozděluje objekty v aplikaci do tří rolí – modelu, pohledu a řadiče. Každá role je nezávislá a změna v ní má minimální vliv na ostatní role [9].

Model zapouzdřuje aplikační data, obsahuje logiku a výpočty k manipulaci a zpracování dat. Ve správném návrhu aplikace model neobsahuje informace o reprezentaci dat. Model zahrnuje doménově specifické informace a je znovupoužitelný k řešení problémů, ke kterým byl navrhnut. Obsahuje-li aplikace data, vyžadující persistenci, pak by měly být tyto informace obsaženy v modelu.

Pohled je objekt, který vidí přímo uživatel a může reagovat na jeho interakci. Úkolem pohledu je prezentovat data poskytnutá modelem a umožnit jejich editaci.

Řadič je prostředník mezi pohledem a modelem. Zasílá zprávy pohledu, resp. modelu, ke změně svého stavu. Řadič může řídit životní cyklus objektů v aplikaci a provádět založení a konfiguraci úloh.

Na obrázku 3.2 je prezentován diagram znázorňující komunikaci mezi jednotlivými rolemi. Pokud uživatel zadá v pohledu pokyn ke změně hodnoty objektu, pohled informuje



Obrázek 3.2: Diagram znázorňující spolupráci komponent v návrhovém vzoru MVC. [9]

o této skutečnosti řadič, který zašle zprávu k aktualizaci modelu a provede změnu hodnoty. V případě, že aplikace používá databázi, provede model persistenci změny. Po provedení dílčích činností informuje model řadič a ten aktualizuje pohled v závislosti na úspěšnosti celkové operace. Z obrázku 3.2 je zřejmé, že v žádném případě nekomunikuje pohled přímo s modelem a model nekomunikuje s pohledem. Veškerou komunikaci pohled – model zajišťuje řadič. Pokud aplikace pracuje s databází, model zajišťuje přístup k datům a může provádět jejich validaci před samotnou validací na úrovni databází.

Pochopení principu návrhového vzoru Model–View–Controller je naprosto zásadní pro vývoj mobilních aplikací na platformě iOS [2]. V základních aplikačních rozhraních se vývojář nejčastěji setká s konceptem *delegáta* a s ním související implementací *protokolů*. Jazyk Swift využívá protokolového programování. Více informací o programovacím jazyce Swift popisuje kapitola 3.3.

3.3 Programovací jazyk Swift

Programovací jazyk *Objective–C* byl od uvedení operačního systému iOS v roce 2007 až do roku 2014 hlavním jazykem, ve kterém vývojáři programovali aplikace nejen pro mobilní operační systém iOS, ale i počítačové aplikace pro systém OS X. Swift, programovací jazyk představený v roce 2014, se rychle etabloval mezi vývojáři a stal se oblíbeným programovacím jazykem k vytváření aplikací pro operační systémy iOS, OS X, watchOS a tvOS[4].

Jazyk Swift je kompilovaný a *multi-paradigmatický* programovací jazyk se zaměřením na intuitivnost a rychlost. Se svou silnou typovostí zabraňuje přiřazení špatného typu do proměnných, v základu nepracuje s ukazateli a zavádí typ proměnné Optional. U konstant a proměnných v jazyce Swift není explicitně udáván datový typ. Programovací jazyk Swift sám rozpozná datový typ podle přiřazované hodnoty a za jednotlivými příkazy není nutné psát středník.

Optional je speciální datový typ, který má dva stavy – *Set*, resp. Nastaveno a *Not set*, resp. Nenastaveno. Pokud je proměnná typu Optional ve stavu Set, proměnná obsahuje přidruženou hodnotu určitého datového typu. Je-li proměnná typu Optional ve stavu Not Set, přidružená hodnota není nastavená a pokus o čtení skončí chybou za běhu programu a k pádu aplikace. Swift obsahuje konstrukce k získání stavu proměnné Optional a umožňuje před samotným čtením zkontrolovat stav a zabránit tak chybě. V žádném ze svých stavů

[4] http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/

proměnná Optional nevrací tzv. *null pointer* ani adresu do paměti [2]. Důležitou vlastností, kterou musí programátor při používání Optional proměnných brát v potaz je stav rozbalená, resp. nerozbalená proměnná typu Optional. Je-li proměnná deklarována jako explicitně rozbalená a program se pokusí přečíst hodnotu jejího přidruženého typu, vznikne chyba za běhu programu, která vede ke spadnutí celého programu. Čtení z nerozbalené proměnné typu Optional chybou neskončí, program vypíše stav – Set, resp. Not Set. Práce s tímto druhem proměnných je prezentována v ukázce 3.1.

Ve zdrojovém kódu aplikace lze libovolně kombinovat programovací jazyky Swift a Objective–C. Zmíněné jazyky jsou mezi sebou kompatibilní a Swift využívá mnoho aplikačních rozhraní jazyka Objective–C. Swift, vzhledem k tomu, že je multi-paradigmatický, umožňuje výběr libovolného programovacího paradigmatu, např. objektově orientovaného, funkcionálního a jiných. Na rozdíl od jazyka Objective–C, kde bylo nutné vkládání hlavičkových souborů a knihoven, můžeme kód v programovacím jazyce Swift psát rovnou [8].

Ukázka 3.1: Programovací jazyk Swift

```
1  print("Hello, World!")
2  let x = 1
3  var y = "Hello"
4  var z: Int?
5  var w: Int!
6
7  z = 42
8  w = nil
9
10 guard let i = z else {
11 fatalError("Promenna neni definovana")
12 }
13
14 print(i)
15 print(w)
16
17 z = nil
18 print(z)
```

Ukázka kódu 3.1 prezentuje základní konstrukce v jazyce Swift a práci s typem Optional. Druhý řádek předkládá deklaraci konstanty datového typu Integer **x** s hodnotou 1. Třetí řádek obsahuje deklaraci proměnné s hodnotou „Hello“ a datovým typem String. Definice proměnné **z** je ukázána na řádku 4. Proměnná je datového typu Optional a její přidružená hodnota je typu Int. V základním stavu je hodnota proměnné **z** Not Set. Řádek pět zahrnuje definici proměnné **w**, která je datového typu Optional, ale na rozdíl od proměnné **z** je implicitně rozbalena. Po přiřazení hodnot do proměnné **z**, resp. **w**, na řádku 7, resp. 8 je ukázána konstrukce guard, sloužící k podmíněnému rozbalení proměnné datového typu Optional. Je-li hodnota proměnné Optional rovna *nil* (no value), přiřazení v konstrukci guard selže a vypíše se chyba. Oproti řádku 14, na kterém dojde k validnímu výpisu hodnoty proměnné **i**, řádek 15 skončí chybou za běhu programu. Chyba je podmíněna implicitním rozbalením proměnné datového typu Optional, která neobsahuje přidruženou hodnotu a program přistupuje k hodnotě *nil*. Pokud by proměnná nebyla implicitně rozbalena (použití operátoru ? namísto !), program by přistupoval pouze k stavu proměnné Optional. Při tomto přístupu ukázaném na řádku 18, dojde k vypsání stavu – Not Set.

3.4 Vývojové nástroje

K vývoji aplikací a programů, nejen pro systémy OS X, iOS, watchOS a tvOS, Apple vytvořil vývojové prostředí **Xcode**. V tomto integrovaném vývojovém prostředí se nachází textový editor, překladač, simulátor, nástroje pro hledání chyb a systém pro správu verzí. **Interface Builder** – nástroj k vytváření grafického rozhraní, je rovněž integrován do vývojového prostředí Xcode. Bližší popis nástroje Interface Builder je uveden v kapitole 6.1. Další důležitou komponentu při vývoji tvoří program **Instruments**, sloužící ke sledování aktivity aplikace, alokování zdrojů, spotřeby, síťové komunikace a jiných. Instruments je dostupný po instalaci integrovaného vývojového prostředí Xcode. Alternativu k prostředí Xcode představuje vývojové prostředí AppCode od firmy JetBrains.

AppCode nabízí pokročilejší možnosti zpracování zdrojového kódu a možnost instalace zásuvných modulů. Pokud programátor pracuje při kódování zároveň s grafickým prostředím, otevře se program Interface Builder, který spustí vývojové prostředí Xcode. Při těchto činnostech dochází často ke změně kontextu mezi dvěma odlišnými vývojovými nástroji. Stejná situace nastává v případě, že vývojář pracuje s datovým modelem pro aplikační rámec Core Data.

Kapitola 4

Mobilní aplikace využívající geografické blízkosti uživatelů

S vývojem chytrých mobilních telefonů a aplikačních rozhraní je snadné požádat uživatele o zpřístupnění polohy. Dříve byla geografická lokace uživatele používaná především k navigaci, většinou pomocí specializovaného zařízení. Dnešní chytrá mobilní zařízení umožňují vývojářům získat polohu uživatele, a se stále se rozšiřujícím internetovým připojením může uživatel využívat vytvořené aplikace v průběhu celého dne. Použitím moderních technologií vznikly nové produkty a došlo k výrazné digitalizaci služeb již existujících.

V kapitole jsou představeny vybrané mobilní aplikace, které využívají geografické lokace uživatele inovativním způsobem.

4.1 Uber

Společnost Uber vznikla v roce 2009, původně pod názvem UberCab, jako online dopravní společnost. V roce založení firma obdržela investici v hodnotě $200 000, rok poté $1.25 milionu a v roce 2012 $48 milionů[1]. Služba byla spuštěna ve městě San Francisco v roce 2011. Uber rychle expandoval, v roce 2011 působil ve městech New York, Chicago a Washington, D.C. Prvním městě neležícím na území Spojených Států, ve kterém byla služba zavedena, se stala Paříž. Společnost expandovala do více měst na různých kontinentech. Po třech letech od spuštění služby byla firma v roce 2014 oceněna na částku $18.2 miliardy[2]. Během agresivního růstu provázela společnost Uber kontroverze. V mnoha zemích je snaha zakázat služby nabízené společností Uber. Častým argumentem kritiků je fakt, že společnost neplatí daně ani licenční poplatky za provozování taxi služby. Obsahem stížností se stala i bezpečnost cestujících, vzhledem k tomu, že řidiči využívající mobilní aplikaci Uber nemají na rozdíl od řidičů taxi požadovanou kvalifikaci, licenci ani pojištění. Na trzích, kde legislativa dovoluje vytvoření dohody o pronájmu vozidla, je po splnění minimálního věku, odpovídajícího zdravotního stavu a řidičského oprávnění, jedinou podmínkou absolvování praktického testu. Ze zmíněného vyplývá, že nároky na řidiče jsou diametrálně odlišné od náročnosti získání oficiálního statusu řidiče taxi.

I přes kritiku se obchodní model společnosti Uber stal vzorem pro jiné firmy a zavedl pojem uberifikace [14]. Aplikace Uber funguje jako služba na požádání – uživatel pomocí nabízené mobilní aplikace pro chytré telefony vytvoří nabídku na odvoz. Aplikace lokalizuje

[1] https://www.crunchbase.com/organization/uber/entity
[2] http://www.wsj.com/articles/uber-gets-uber-valuation-of-18-2-billion-1402073876

polohu uživatele, který upřesňuje místo vyzvednutí. Po přijmutí nabídky řidičem má uživatel možnost sledovat polohu vozidla. Platba probíhá automaticky stržením částky z kreditní karty uživatele a účet je zaslán na e-mailovou adresu uživatele. Účtovaná cena není počítána pomocí taxametru, veškeré zpracování provádí Uber. Uživatel a řidič mají možnost se navzájem hodnotit. Má-li uživatel špatné hodnocení, služba se pro něj může stát méně vyhovující (např. prodloužení doby čekání), resp. může dojít k vyřazení ze služby Uber.

4.2 Foursquare

Koncept aplikace Foursquare byl vytvořen v roce 2008 a služba byla spuštěna v roce 2009. Původní zaměření aplikace Foursquare bylo vytvoření gamifikovaného prostředí se sociálními prvky, kde se uživatelé pomocí aplikace označovali v geografickém místě za účelem získání bodů, odznaků a případných výhod. Pokud se uživatel nacházel v místě, které podporovalo službu Foursquare, mohl pomocí mobilní aplikace lokalizující jeho polohu provést označení vybraného místa. Navštěvoval-li uživatel často místa určitého zaměření, např. fitness centra, byl odměněn speciálními odznaky. Využívání aplikace Foursquare k pravidelnému a častému označování bylo odměněno titulem virtuálního starosty. Titul využívaly podniky k poskytování slev a jiných benefitů pravidelným zákazníkům. Nenachází-li se místo v aplikaci Foursquare, je možné lokaci přidat z aplikace nebo pomocí webového rozhraní služby. K místům, vyskytujícím se v aplikaci Foursquare, lze psát recenze, tipy a sdílet fotografie.

V roce 2014 byla vydána osmá verze aplikace Foursquare, která odstranila možnost označování se a sbírání bodů, resp. odznaků. V aplikaci lze vyhledávat podniky v geografické blízkosti uživatele, psát recenze k podnikům a hodnotit je. Aplikaci je možné personalizovat, uživatel aplikace může určit kategorie podniků, oblíbené kuchyně a jiná kritéria. Z poskytnutých dat doporučuje aplikace uživateli zajímavá místa k návštěvě. Funkcionalita, kterou aplikace Foursquare pozbyla, byla přenesa do zcela nové mobilní aplikace **Swarm**.

4.3 Swarm

Rozštěpením aplikace Foursquare vznikla v květnu roku 2014 mobilní aplikace Swarm. Funkce, odstraněné z aplikace Foursquare byly přesunuty do zcela nové aplikace Swarm. Zaměřením aplikace Foursquare se stalo vyhledávání vhodných podniků dle preferencí uživatele, zanechávání recenzí a fotografií k danému podniku. V aplikaci Swarm má uživatel možnost označovat lokaci, sbírat body a odznaky. Na rozdíl od soupeření se všemi uživateli aplikace, jak tomu bylo u služby Foursquare, uživatel soupeří pouze se svými přáteli.

Swarm je mnohem více zaměřený na sdílení polohy. Uživateli se zobrazuje poloha přátel a aplikace vytváří kompetitivní prostředí mezi přáteli uživatele a uživatelem samotným. Jednotlivé označování je odměňováno body, které se počítají do celkového hodnocení, zaznamenávaného napříč přidanými uživateli. Body slouží jako virtuální měna, určená např. k nákupu nálepek zobrazujících se při označení v lokalitě.

Aplikace Swarm a Foursquare se navzájem doplňují. Data získaná od uživatele při používání aplikace Swarm, jsou použita ve službě Foursquare. Foursquare na základě poskytnutých dat může odhadnout místa relevantní pro uživatele a doporučit je k návštěvě.

4.4 Tinder

Tinder je služba využívající geografické blízkosti uživatelů k seznámení. Prostřednictvím sociální sítě Facebook se uživatel registruje do aplikace Tinder a jsou získány informace k vytvoření profilu uživatele. Podle společných zájmů, geografické blízkosti a počtu vzájemných přátel jsou určeni pravděpodobní kandidáti k seznámení. Kandidáti jsou následně prezentováni uživateli, který může pomocí gesta „švihnutí“ (swipe) na chytrém telefonu doleva nebo doprava určit, zda kandidát nevyhovuje, resp. vyhovuje jeho kritériím. Tinder lze propojit s aplikací Instagram, přičemž po propojení profil uživatele obsahuje více fotografií a vypovídá přesněji o jeho zájmech.

V případě, kdy dojde k vzájemnému souhlasnému označení mezi uživatelem a kandidátem, zpřístupní se konverzace. U bezplatného účtu je limitován počet možností označení vyhovujících uživatelů. Funkce je zpoplatněna pomocí konceptu předplatného, které je uživateli opakovaně účtováno v daném časovém intervalu.

Kapitola 5

Návrh mobilní aplikace Gymber

Před samotným začátkem vytváření návrhů a prototypů bylo nutné definovat, co má obsahovat MVP (Minimal Viable Product) aplikace [16]. V případě Gymberu byly navrženy tři rozdílné případy užití, modelující potenciální chování uživatele aplikace. Při tvorbě návrhů případů užití byly vybírány funkce, které mají přidanou hodnotu pro uživatele a odlišují aplikaci od konkurence s tím, že v pozdějších verzích bude do aplikace postupně přidávána další funkcionalita [12].

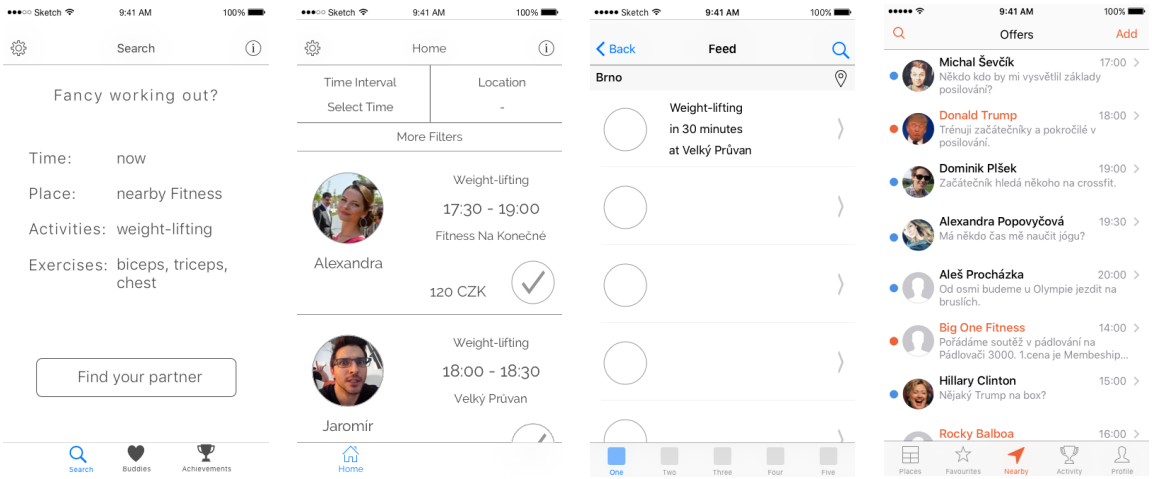
První případ užití (příloha A.1) modeluje aplikaci, která slouží k vyhledávání a objevování fitness podniků v okolí. Uživatel má možnost prohlížet v seznamu jednotlivá fitness centra a procházet jejich hodnocení, případně rezervovat hodinu lekce s trenérem nebo jinou placenou fitness aktivitu nabízenou zařízením. Pro fitness centra by aplikace nabízela prostor k inzerci a možnosti nalákat uživatele na poskytované placené lekce.

Druhý diagram zobrazený v příloze A.2 byl vytvořen k aplikaci, kde by uživatel přímo reagoval na nabídky trenérů. Trenéři nabízí své lekce a uživatelé mají možnost přes aplikaci objednat jejich služby. Jednotlivé nabídky jsou řazeny podle hodnocení trenérů. Aplikace by přinášela trenérům z fitness center i nezávislým trenérům možnost zajištění si zákazníků. Uživatelé mohou využít výběr ze širšího spektra nabídek a absolvovat placené lekce s trenérem za nižší ceny.

Třetí návrh umožňuje uživateli vyhledávat partnery nebo trenéry ke cvičení, nabízí přehled míst vhodných k vykonávání fitness aktivit a zobrazuje jejich případné otevírací hodiny, kontakty a webové stránky. Uživatel může v aplikaci přidat ostatní uživatele mezi oblíbené a sledovat jejich aktivitu. Trenéři i uživatelé mohou vypisovat placené a neplacené nabídky. Oproti druhému návrhu aplikace se obyčejný uživatel může stát trenérem. Z tohoto návrhu vychází specifikace MVP aplikace Gymber, kde implementované funkce odpovídají případům užití aplikace. Návrh specifikuje příloha A.3.

5.1 Validace myšlenky

Před vývojem aplikace byla provedena analýza, která měla zjistit, zda je o připravovaný produkt zájem. Ke zjištění názoru na vytvářenou aplikaci byl vytvořen internetový dotazník distribuovaný pomocí sociálních sítí. Celkově dotazník vyplnilo 34 respondentů. Hlavním cílem bylo zjištění, zda by lidé vyzkoušeli vyvíjenou aplikaci a jestli by využívali funkce k nalezení partnera na cvičení a sjednání si lekcí s trenérem. Při vyhodnocení výsledků bylo zjištěno, že 39% dotázaných by vyzkoušelo službu k nalezení partnera a 29% by ji využívalo. K možnosti sjednání si lekcí s trenérem pomocí aplikace 48% dotázaných odpovědělo, že by



(a) První prototyp. (b) Druhý prototyp. (c) Třetí prototyp. (d) Čtvrtý prototyp.

Obrázek 5.1: Vizualizace jednotlivých prototypů aplikace Gymber. Návrh prvního prototypu se skládá z hlavní obrazovky, kde pomocí formulářových prvků uživatel zadá filtry k vyhledání vyhovující nabídky. Od druhého prototypu aplikace je uživateli k dispozici seznam nabídek, na které může reagovat. Nadcházející prototypy optimalizují prostor v seznamu nabídek a rozvíjí funkčnost mobilní aplikace.

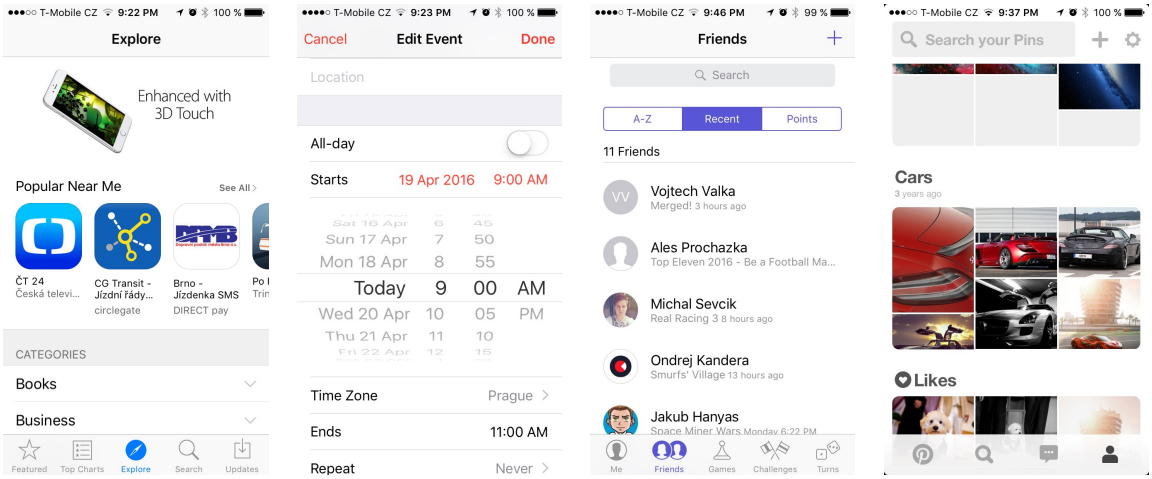
službu vyzkoušeli a 23% by službu využívalo. Ze zjištěných výsledků vyplynulo, že uživatelé vyjadřují podobný zájem o obě dvě funkce, tedy nalezení partnera ke cvičení, resp. sjednání si lekce s trenérem pomocí mobilní aplikace. Z výše zmíněného důvodu byl MVP navrhován pomocí diagramu případu užití, který lze nalézt v příloze A.3. Pokud by byl projeven menší zájem o vyhledávání partnerů, bylo by možné definovat MVP pomocí diagramu v příloze A.2, který modeloval aplikaci zaměřenou pouze na sjednání placených lekcí s trenérem.

5.2 Návrh uživatelského rozhraní pro minimal viable product

Každé uživatelské rozhraní má splňovat princip přehlednosti a jednoduchého ovládání [10]. Při návrhu uživatelského rozhraní pro Gymber bylo zapotřebí brát v potaz i to, že aplikace bude cílena na dvě rozdílné platformy – iOS a Android. Cílem bylo navrhnout vizuální styl, který je možné využít na obou dvou platformách s tím, že na každou z platforem se uživatelské prostředí optimalizuje dle konvencí dané platformy. Tato práce představuje výsledné uživatelské rozhraní pro platformu iOS. Na systému iOS je využito co nejvíce základních systémových prvků, které jsou mírně upraveny s cílem vyhnout se vytváření vlastních grafických elementů, které by mohly být pro uživatele matoucí [10]. Jako příklad je možné uvést tzv. „hamburger menu“, které je na platformě iOS atypické, na rozdíl od platformy Android, kde se využívá velmi často.

Při navrhování aplikace Gymber byly vytvořeny celkově čtyři různé návrhy prototypů, z nichž poslední návrh byl finalizován do stavu, aby mohl být použit jako podklad k vytvoření mobilní aplikace. Jednotlivé prototypy jsou vyobrazené na obrázku 5.1.

U prvního prototypu (obrázek 5.1a) je po spuštění aplikace uživateli prezentována obrazovka s vyhledávacím dialogem. V dialogu uživatel vybírá čas, místo a aktivitu. Problémem návrhu byla nepoužitelnost aplikace k vyhledávání trenérů. Uživatel neměl možnost pro-



(a) App Store (b) Calendar (c) Game Center (d) Pinterest

Obrázek 5.2: Obrazovky vybraných aplikací, které byly předlohou pro komponenty aplikace Gymber. App *Store* obsahuje prvek „Explore“ zobrazující populární mobilní aplikace v blízkosti uživatele. *Calendar* využívá prvku „date picker“ pro komfortní zadávání času v mobilním zařízení. *Game Center* představuje vhodné rozmístění prvků v buňce. *Pinterest* zobrazuje různě velké obrázky pomocí rozdílných vertikálně dlouhých buněk.

cházet existující nabídky, pouze v nich vyhledávat pomocí dialogu. Je nutné zmínit fakt, že vyplnění dialogu bylo časově náročné a návrh působil neintuitivně.

Hlavní obrazovkou druhého prototypu je seznam nabídek. V horní části obrazovky je umístěn filtr, kde lze vybrat vhodný časový interval nabídky, polohu a další možnosti řazení. Jednotlivá buňka s nabídkou zabírá většinu obrazovky a zobrazuje velké množství informací. V návrhu druhého prototypu aplikace lze přijímat nabídky ze stejné obrazovky. Problémem návrhu je velikost buňky zobrazující informace k jednotlivé nabídce. Při zachování stejného počtu zobrazovaných informací a dodržení čitelnosti je buňka příliš velká a uživatel je nucen často rolovat obrazovku. Přijmutí nabídky ze stejné obrazovky se může jevit přínosné pro uživatele – není nutný přechod na detail nabídky. Drží-li uživatel chytrý telefon v pravé ruce a roluje palcem, pak se nachází tlačítko na přijmutí v části, kde pokládá palec k rolování a může nastat nechtěné přijmutí nabídky.

U návrhu třetího prototypu bylo cílem zmenšit buňku s informacemi o nabídce. Tlačítko pro přijmutí nabídky bylo odstraněno a přesunuto do detailu nabídky. Filtr, který byl umístěn v horní části obrazovky druhého návrhu byl odstraněn a přesunut do vlastní obrazovky, která se zpřístupní po stisknutí tlačítka s lupou. Horní část obrazovky tvoří návěští s lokací uživatele. Ve výpisu má uživatel možnost vidět profilovou fotku zadávajícího, aktivitu, čas a lokaci.

Čtvrtý prototyp vychází z poznatků zjištěných u předchozích prototypů, především z návrhu třetího. Zmenšením profilové fotky a organizací jednotlivých prvků bylo dosaženo menšího rozměru buňky. Výsledkem je více nabídek na obrazovce a uživatel není nucen zbytečně rolovat. U návrhu bylo docíleno rozlišení placených a neplacených nabídek, resp. trenérů od uživatelů a zjednodušení zobrazovaných informací uživatelům.

Na obrázku 5.2 jsou prezentovány obrazovky ze čtyř různých aplikací – *App Store*, *Kalendář*, *Game Center* a *Pinterest*. Obrázky slouží jako prostředek k naznačení procesu vývoje uživatelského rozhraní, přívětivého pro koncového uživatele, s využitím známých vzorů,

na které je uživatel zvyklý. Hlavní navigaci tvoří spodní menu, které lze nalézt například u nativní aplikace systému iOS – *App Store*. Aktivní prvek „*Explore*“ – zvýrazněný modře – zobrazuje aplikace populární v blízkosti uživatele. U aplikace Gymber prostřední prvek zobrazí uživateli seznam nabídek ke cvičení v jeho geografické blízkosti. V nativní aplikaci *Kalendář* na systému iOS je pro zadávání časových údajů využito prvku *date picker*, který slouží pro komfortní zadávání času na mobilním zařízení. Tento přístup odstraňuje ruční vypisování času uživatelem a je využit v aplikaci Gymber pro usnadnění zadávání časových údajů. Ve výpisu nabídek jsou jednotlivé nabídky složeny z profilové fotky uživatele, statusu, který poskytl při vytváření nabídky a času začátku aktivity. Zmíněné rozvržení prvků se inspiruje nativní aplikací *Game Center*. Ukázka z aplikace *Pinterest* zobrazuje různě velké obrázky pomocí rozdílných vertikálně dlouhých buněk. Při řešení problému vykreslení různě velkých buněk obsahujících fotografii místa vhodného k fitness aktivitě a dalších informací, bylo využito podobného principu, ale v horizontální rovině. Řešení umožňuje do budoucna využívat velikost zobrazované plochy v uživatelově telefonu jako monetizačního nástroje aplikace.

Na obrázku 5.3 je představena obrazovka z aplikace, která uživateli umožňuje prohlížet lokality vhodné k fitness aktivitě. Převážně grafická data, která se skládají z obrázku zobrazujícího dané místo a název, umožňují rychlý přehled o lokalitách v blízkosti. V pravém horním rohu je uvedena vzdálenost uživatele od lokality, v levém horním rohu počet ostatních uživatelů aplikace, kteří se momentálně nachází v zobrazeném místě.

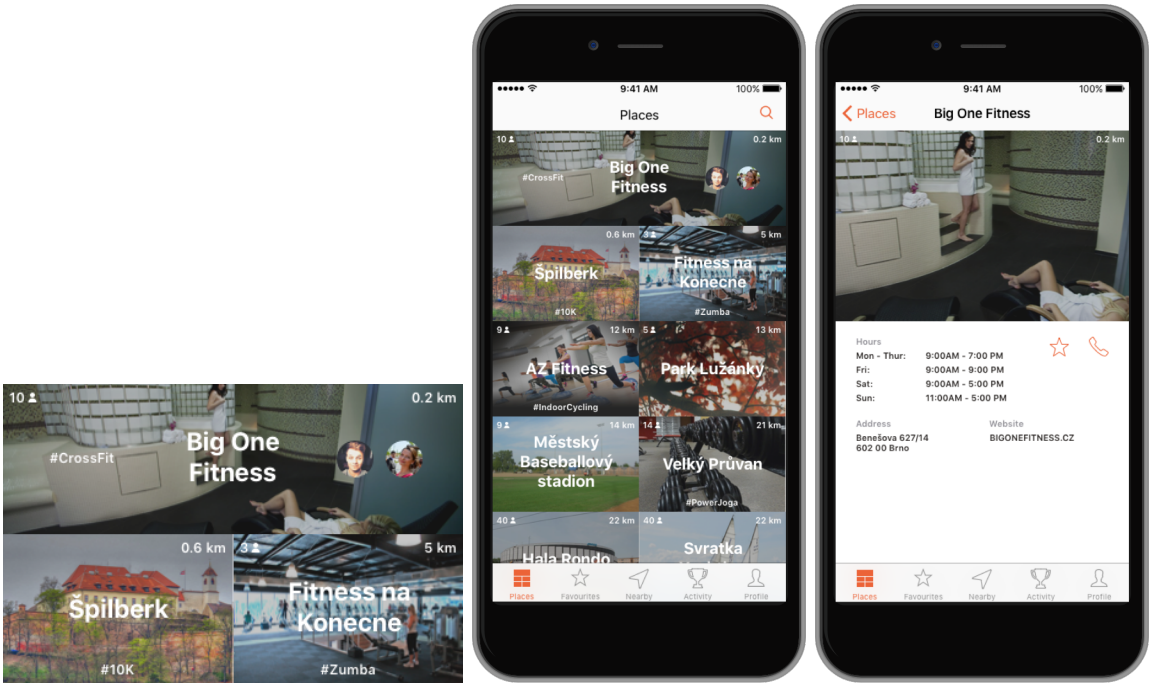
V buňce místa se nachází profilové obrázky uživatelů, kteří jsou mezi oblíbenými a v danou chvíli se nacházejí v zobrazeném místě. Tento sociální prvek má podporovat uživatele aplikace k tomu, aby vykonával fitness aktivitu se svými přáteli, případně vyzkoušel nová místa, která již navštívili někteří z jeho přátel. Nachází se zde štítek (hash tag), který informuje uživatele o zajímavých aktivitách nebo zdůrazňuje popularitu místa vyjádřenou počtem uživatelů aplikace, kteří se na daném místě nacházeli. Štítky vycházejí z populární sociální aplikace *Twitter*. Vzhledem k potenciálně velkému počtu míst je k přehlednému zobrazení dat použit tabulkový výpis. Na podobný styl mohou být uživatelé zvyklí například z aplikace *Pinterest* a jiných aplikací zobrazujících převážně grafické objekty.

Pokud má uživatel zájem dozvědět se něco více o lokalitě, může poklepnout na požadované místo a aplikace zobrazí detail. Detail obsahuje například adresu, webové stránky a otevírací hodiny fitness centra. V případě možné rezervace lze využít tlačítka se symbolem telefonu. Tlačítko se symbolem hvězdy umožní uživateli přidat lokalitu mezi oblíbené. V tomto případě se v nabídce na obrázku 5.4 budou v závislosti na typu místa vypisovat informace k vybrané lokalitě, například novinky o slevových akcích a speciálních nabídkách.

Obrázek 5.4 prezentuje obrazovku z aplikace, zpřístupňující výpis nabídek s fitness aktivitami od ostatních uživatelů. Do tabulky situovaný výpis obsahuje buňky s celým jménem uživatele zadávajícího nabídku, požadovaný čas konání aktivity a krátký popis. Placené a neplacené nabídky jsou rozděleny podle barvy kruhu, který se nachází na levé straně od profilové fotky zadávajícího uživatele.

Pro placené nabídky byla zvolena oranžová barva. Oranžovou barvou je vybarveno i jméno zadávajícího, pokud se jedná o trenéra fitness aktivit. V opačném případě je u neplacených nabídek kruh vybarven modře a jméno zadávajícího má černou barvu. Nová nabídka je přidávána v horní liště pomocí tlačítka **Add**.

Obrazovka pro přidání nové nabídky je zobrazena na obrázku 5.5. Pomocí přepínače lze vybrat, zda jde o placenou nebo neplacenou nabídku. Pokud je vybrána placená (symboly dolaru), zobrazí se textové pole, do kterého uživatel může uvést cenu. Do pole s návěštím *popis* lze zadat libovolný popis k nabídce. Časový interval se upřesňuje pomocí dvou texto-



Obrázek 5.3: Vizualizace obrazovky s výpisem a detailem míst vhodných k fitness.

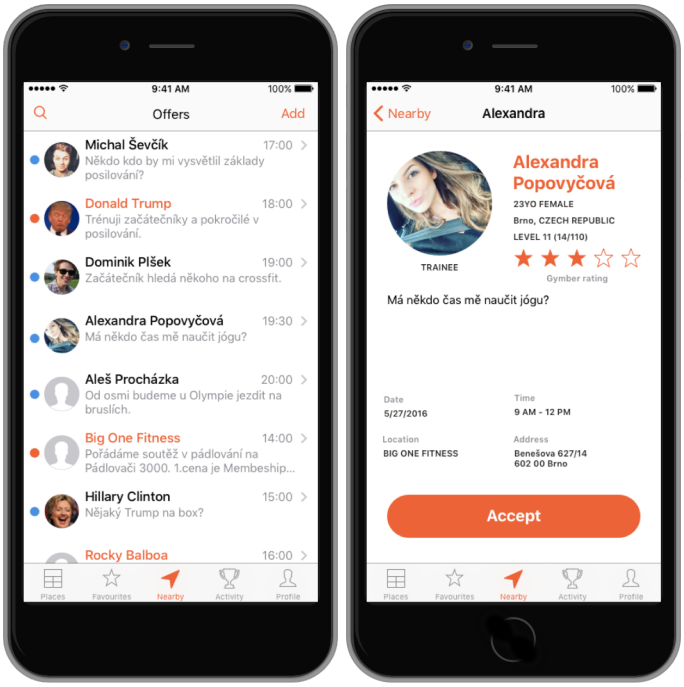
vých polí, která zobrazí optimalizovanou softwarovou klávesnici pro určení časového údaje. V posledním textovém poli je možné vyznačit lokaci pro vytvářenou nabídku. Po přidání je nabídka zobrazena v seznamu a zpřístupněna zadávajícímu uživateli ke sledování stavu.

Po poklepání na příslušnou nabídku uživatel uvidí detail, který obsahuje jméno a příjmení, věk, pohlaví a město, ve kterém zadávající bydlí. Používáním aplikace uživatel získává body a zvyšuje se mu úroveň v rámci aplikace. Podle klasifikace od ostatních se uživateli vypočítává hodnocení. Tyto dva parametry budou ovlivňovat pořadí nabídky ve výpisu na obrázku 5.4 a pomohou ostatním uživatelům k rozhodování, kterou nabídku ke cvičení přijmout. U uživatelů evidovaných jako trenéři, jsou tyto parametry důležité k získání zákazníků. Princip hodnocení vychází z aplikace Uber[1], kde dochází k vzájemnému hodnocení mezi řidičem a pasažérem. V aplikaci Gymber budou placené lekce podporovány a zobrazovány na vyšších místech ve výpisu.

Informace o datu, času a lokaci jsou umístěny v dolní části u potvrzujícího tlačítka. Umístění umožňuje uživateli vidět tyto důležité informace bez nutnosti rolování obrazovky a slouží k zamezení chybného zvolení nevyhovující nabídky.

Na obrázku 5.5 je ukázaná obrazovka zobrazující se po přijmutí nabídky uživatelem, který může reagovat na nabídku vlastní zprávou. Zpráva se po odeslání zobrazí uživateli, který nabídku vytvořil v detailu vytvořených nabídek. Při akceptování nabídky lze upřesnit časový interval aktivity. Po odeslání se nabídka se zprávou a ostatními informacemi zobrazí v seznamu přijatých, resp. vytvořených nabídek. Případná finanční kompenzace a další domluva je v kompetenci uživatelů.

[1] http://uber.com



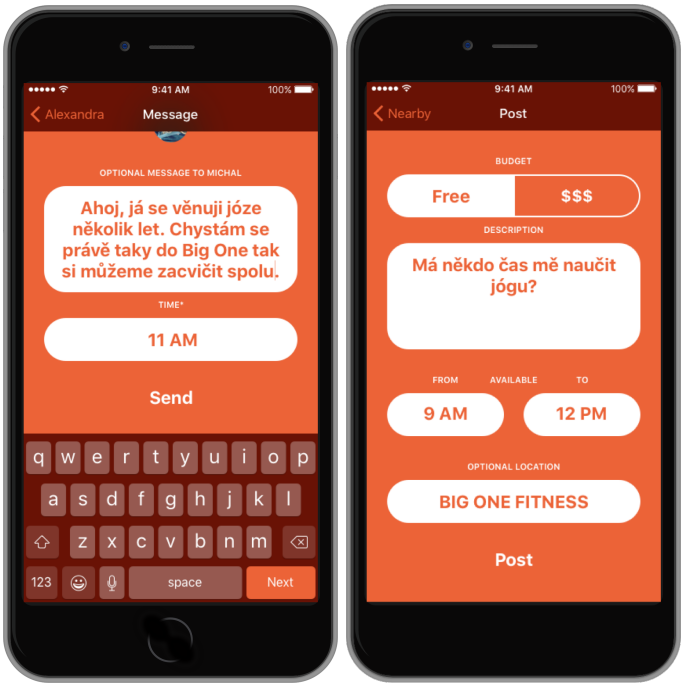
Obrázek 5.4: Vizualizace obrazovky s výpisem a detailem nabídek ke cvičení.

5.3 Srovnání aplikace Gymber s aplikacemi podobného zaměření

Ke srovnání mobilní aplikace Gymber je potřeba definovat kategorie, do kterých spadá a z nich vybrat aplikace podobného zaměření. Při pohledu na navržené funkce zjistíme, že aplikaci nelze zařadit pouze do kategorie fitness. Primárním účelem aplikace je přivést uživatele ke společnému cvičení – vykonávání fitness aktivity. Na rozdíl od známých fitness aplikací jako *Runtastic*, *Runkeeper*, *Nike+ Running*, *Endomondo*, *Strava* a dalších, neslouží Gymber k zaznamenávání a sdílení dosažených výsledků. K nalezení partnera ke cvičení slouží aplikace WellSquad. Uživatel má možnost vytvořit vlastní tým a s členy týmu vykonávat a zaznamenávat fitness aktivity. Vedoucími týmu v aplikaci *WellSquad* mohou být ověření fitness trenéři a dietologové.

Gymber nepodporuje vytváření týmu, ale umožňuje uživatelům sledovat se navzájem. Fitness trenéři nespolupracují s uživateli v týmu, ale vypisují nabídky na placené lekce. Uživatelé mohou na nabídky reagovat v reálném čase podobně jako v aplikaci Uber, kde uživatel vytvoří nabídku na odvoz, u které upřesní svoji polohu a registrovaný řidič ve službě Uber má možnost nabídnout přepravu svým automobilem za stanovenou taxu. K nalezení vhodného místa k fitness aktivitě může být použita služba Foursquare, resp. služba Swarm. Zmíněné služby slouží ke sdílení recenzí a k označení se v dané lokalitě, ale nespecializují se na fitness.

V aplikaci Gymber může uživatel procházet seznam míst vhodných ke cvičení od velkých fitness center po veřejné parky a lokality, které jsou vhodné k běhání a ostatním fitness aktivitám. Gymber umožňuje uživateli sdílet status, podobně jako na sociální síti *Facebook*. Může sloužit i k seznámení se s ostatními uživateli podobně jako aplikace *Tinder*.



Obrázek 5.5: Vizualizace obrazovky k zaslání zprávy a vytvoření nové nabídky.

5.4 Multiplatformnost služby

K zajištění multiplatformního chodu služby bylo zapotřebí definovat společné aplikační rozhraní, pomocí kterého bude komunikovat server s aplikací. Aplikace komunikující podle předepsaného protokolu se serverem obdrží data z databáze ve formátu vhodném pro přenos, zpracuje je a interpretuje v aplikaci. Pro přenášení dat byl zvolen formát *JSON* (JavaScript Object Notation), který se vyznačuje svou jednoduchostí, srozumitelností a platformovou nezávislostí. Oproti formátu XML obsahuje JSON mnohem méně značek. Nevýhodou formátu JSON je velikost přenášených dat. V návrhu jsou přenášená data před samotným přenosem serializována.

Z hlediska přístupu a zabezpečení uživatelských dat bylo nutné navrhnout aplikační rozhraní tak, aby fungovalo jako privátní a podporovalo autentizaci. K zajištění, že API využívá pouze klient s nainstalovanou aplikací je využito šifrovaného klíče, který je uložen v chráněné sekci aplikace. Při každém dotazu dojde k ověření, zda server komunikuje s ověřeným klientem. K tomu, aby mohl uživatel plynule využívat aplikaci na obou platformách, je důležité používat pro obě platformy stejnou databázi. Je nutné zajistit autentizaci uživatelů, která je v případě aplikace Gymber založena na ověřování tokenů (pešek, žeton). Při vytvoření nového uživatelského účtu se vygeneruje unikátní autentizační token, který je uložen s uživatelskými daty. Klient zašle autentizační token uživatele společně s globálním ověřovacím tokenem v každé zprávě předávané na server. V opačném případě nedojde k ověření a server vrací chybu – neautorizovaný přístup.

Protokol, který využívá klient a server ke komunikaci je shodný na obou platformách. Veškerá komunikace probíhá přes zabezpečený protokol *HTTPS* k ochraně autentizačních a ověřovacích klíčů a uživatelských dat. Výše zmíněné řešení umožňuje bezstavový provoz na serveru s podporou autentizace a využívá společný protokol ke komunikaci na straně klienta.

5.5 Návrh serverové části služby

V kapitole 5.4 byly definovány nároky na návrh multiplatformního aplikačního rozhraní. Pokud klient, resp. mobilní aplikace, zašle validní zprávu dle protokolu z kapitoly 5.4, dojde ke zpracování zprávy, vyhodnocení a zaslání odpovědi zpět klientovi. Zmíněné činnosti implementuje server postavený na architektonickém stylu *REST* (*Representational State Transfer*). REST je koncept návrhu komunikace a serveru, který využívá metody *HTTP* protokolu ke čtení, vytváření, editaci a mazání zdrojů. Zdrojem v případě aplikace Gymber je uživatel, místo a další.

K REST službám se vážou metody HTTP, které jsou označovány jako „*RESTful*“. Metody *GET*, *POST*, *PATCH* a *DESTROY* protokolu HTTP jsou zpracovávány na serveru a dále interpretovány jako odpovídající akce nad modelem zdrojů. Odpověď serveru se skládá z HTTP hlavičky a těla zprávy. Autentizace REST služeb se zpravidla realizuje bezstavovým způsobem v rámci každého požadavku, nedochází tedy k vytváření relací mezi serverem a klientem. Často je využívaná tzv. autentizace pomocí tokenů, kdy je vytvořen nebo využíván unikátní klíč k ověření identity a úrovně přístupu klienta.

Implementací REST architektury je vytvořeno jednoduché a rozšiřitelné rozhraní. Na straně klienta je zapotřebí implementovat a interpretovat pouze omezený počet operací. Výhodou je možnost ukládat zdroje do mezipaměti serveru a zkrátit tak dobu odezvy. Navržená RESTful služba implementovaná podle zmíněných koncepcí podporuje verzování aplikačního rozhraní, využívání mezipaměti ke snížení doby odezvy a přístupů do databáze a bezstavovou autentizaci založenou na výměně tokenů.

Kapitola 6

Implementace

Kapitola 5 představila návrh aplikace z pohledu uživatelského rozhraní, návrh aplikačního rozhraní a serverovou část služby. V této kapitole jsou jednotlivé části popsány z hlediska implementace, informuje o použitých technikách, podpůrných službách a dalších nástrojích, které umožní vývoj mobilní aplikace pro platformu iOS využívající podobné principy jako služba Gymber.

6.1 Implementace uživatelského rozhraní pro platformu iOS

Návrh představený v kapitole 5.2 obsahuje široké spektrum elementů, které je potřeba implementovat. I přes důraz na využití základních prvků, které jsou systémové, je zapotřebí tyto prvky správně nakonfigurovat a případně upravit. K implementaci uživatelského rozhraní na platformě iOS bylo z velké části využito nástroje **Interface Builder**. Tento nástroj je v současné době součástí vývojového prostředí Xcode a slouží ke grafickému rozmístění a základnímu nastavení prvků. Interface Builder nemusí být použit pouze na základní rozmístění, lze v něm vytvářet i komplikovaná uživatelská rozhraní. Nástroj využívá konceptu *Storyboard*, což je kompozice scén, kde každá scéna je reprezentována svým řadičem a pohledem. Scény jsou propojeny přechody, které slouží k navigaci v uživatelském rozhraní. Interface Builder s rozšířenou funkcionalitou získanou od vydání sedmé verze vývojového nástroje Xcode, umožňuje i *refactoring* Storyboardů, který je vhodný při návrhu komplikovanějšího uživatelského rozhraní a byl použit při vývoji aplikace Gymber.

Při implementaci aplikace Gymber byly všechny statické prvky vytvářeny za podpory Interface Builderu. V tomto nástroji nelze pracovat s dynamickými prvky, pouze jejich prototypy. Z toho důvodu byly vytvářeny prototypy dynamických buněk v případě návrhu obrazovky obsahující výpis nabídek a výpis míst vhodných k fitness aktivitě, které jsou prezentovány na obrázku 5.4, resp. obrázku 5.3. Při implementaci obrazovky sloužící k výpisu míst bylo zapotřebí dynamicky určovat velikost buňky. Při využití systémového prvku *UICollectionView* lze přepsat metody určující velikost právě vytvářené buňky v tabulce. Pomocí užitých metod bylo docíleno návrhu zobrazeného na obrázku 5.3.

I přes úskalí tvorby dynamicky se měnících prvků, které musí být vytvářeny za běhu aplikace v kódu, je vhodné při vývoji mobilních aplikací pro platformu iOS využívat nástroje Interface Builder k tvorbě uživatelských rozhraní. Výhodou je jednodušší vytváření prototypů uživatelských rozhraní a možnost jejich refaktoringu. Nevýhodou je skutečnost, že využití Interface Builderu a práce pomocí Storyboardů může být obtížnější ve větším týmu vývojářů a klade nároky na znalost nástroje.

6.2 Použití služby Apiary k definici společného API

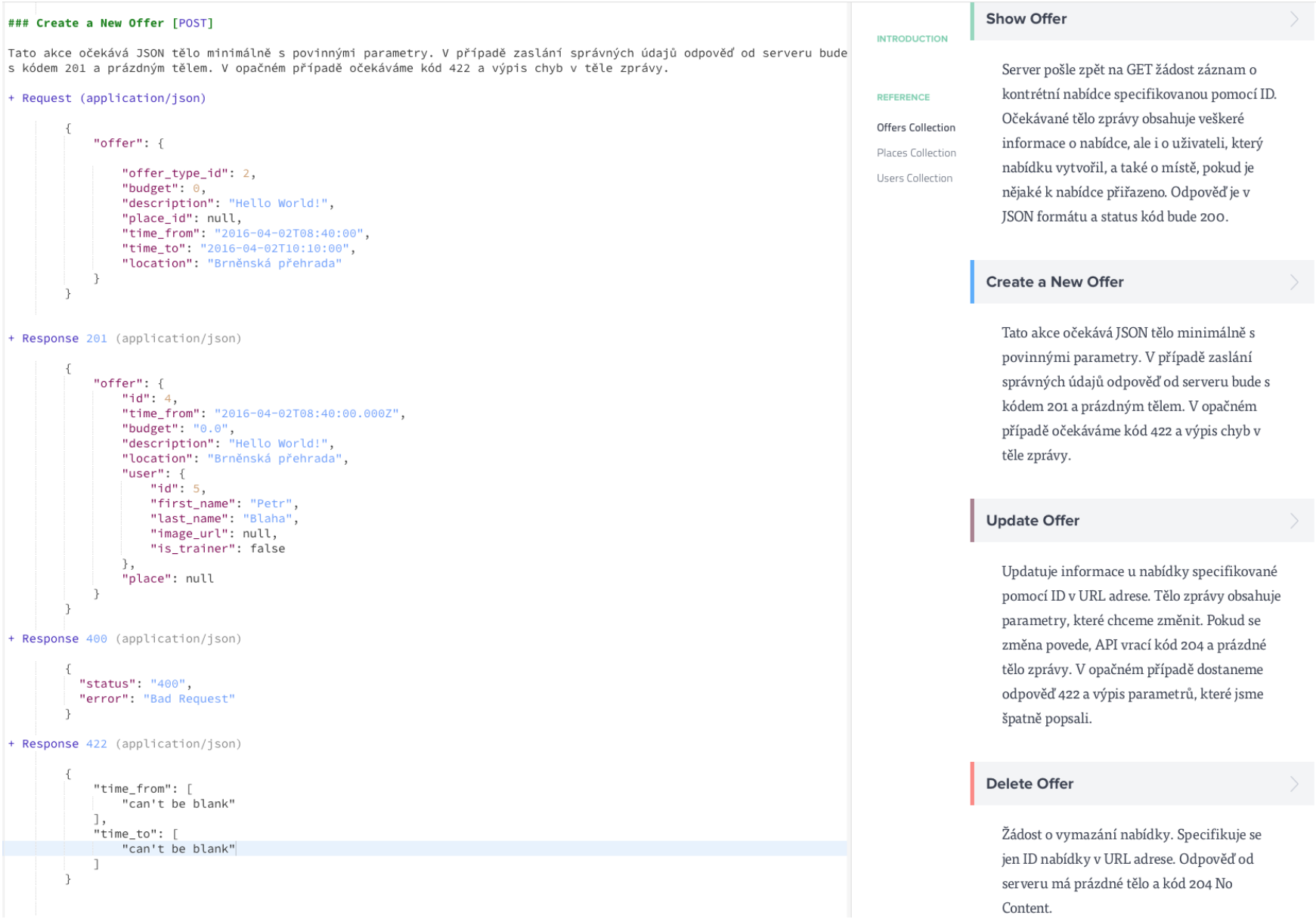
Definice API může být složitý a časově náročný proces, při kterém musí návrhář brát v potaz mnoho aspektů, například velikost přenášených dat, počet dotazů na server, formát zpráv, používaný protokol a další. Dobře navržené API nesmí postrádat dokumentaci a přesnou a jednoznačnou definici metod. K návrhu aplikačního rozhraní, které je využito v aplikaci Gymber, byla použita webová služba Apiary, která slouží k návrhu REST aplikačních rozhraní. V Apiary návrhář pracuje s tzv. *návrhem* (Blueprint), kde lze pomocí předem definované syntaxe navrhovat a zároveň dokumentovat jednotlivé metody.

Apiary umožňuje náhled vytvořeného návrhu s HTTP hlavičkou a obsahem zprávy z testovacího serveru. Každému projektu ve službě Apiary je přiřazena unikátní adresa umožňující testování API s předdefinovanými odpověďmi i z vlastní aplikace. Projekt vytvořený v dané službě lze propojit s repozitářem umístěným na serveru Github. Apiary poskytuje vygenerování základního kódu k vytvoření dotazu na aplikační rozhraní v mnoha programovacích jazycích, umožňuje kontrolovat zaslané dotazy a pomáhá tak k testování a ladění API.

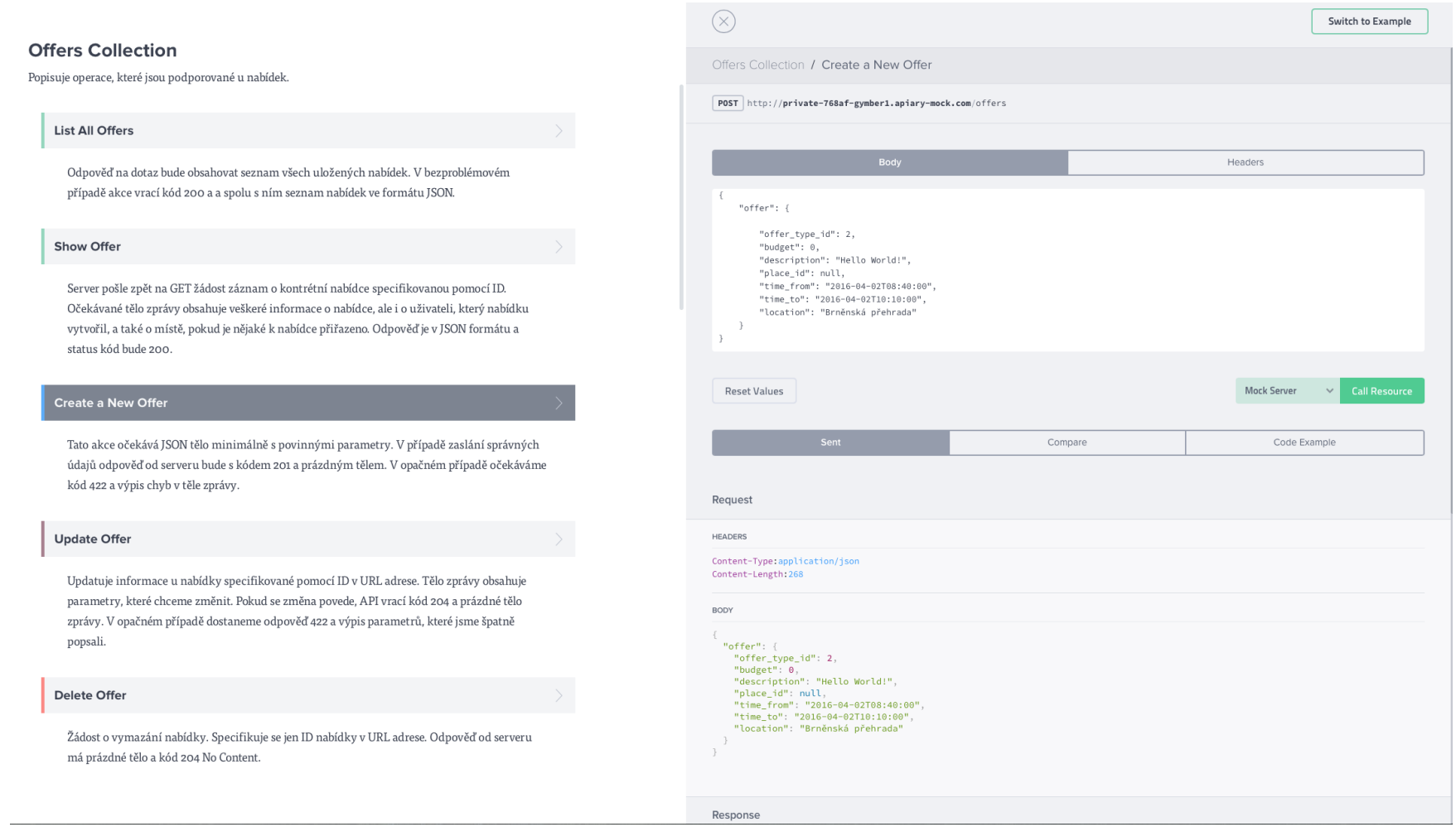
Při návrhu aplikačního rozhraní bylo využíváno především možnosti definice metod v plánu s odpovídající dokumentací obsahující popis právě definované metody. Uvedený postup umožnil spolupráci při tvorbě aplikačního rozhraní použitého na platformách iOS a Android. Aplikace na obou platformách využívaly testovacího serveru Apiary k ladění protokolu a zpracování přenášených dat. Důvodem užití Apiary může být podpora týmové spolupráce s různou úrovní přístupů. Po ustanovení vlastního stylu a protokolu v týmu lze souběžně pracovat na definici a dokumentaci, což vede k urychlení procesu návrhu a dokumentace aplikačního rozhraní.

Na obrázku 6.1 je zobrazena ukázka návrhu aplikačního rozhraní pomocí služby Apiary. V levé části obrazovky se nachází editor, ve kterém pomocí syntaxe definované službou dochází k samotnému navrhování aplikačního rozhraní a jeho dokumentování. Pravá část obsahuje dokumentaci k vybrané metodě. Obrázek 6.2 prezentuje analýzu vytvořené metody v aplikačním rozhraní. Dokumentace v levé části obrazovky obsahuje popis jednotlivých metod. Po vybrání metody se v pravé části obrazovky zobrazí nástroj pro testování. Pomocí tlačítka *Call Resource* je vytvořená metoda zaslána na testovací server nacházející se na službě Apiary a dojde k vygenerování odpovědi, analyzovatelné návrhářem aplikačního rozhraní.

Výsledný návrh na službě Apiary byl implementován na vlastním serveru. Více informací k implementaci aplikačního rozhraní na serverové části služby je uvedeno v kapitole 6.3.



Obrázek 6.1: Ukázka návrhu aplikačního rozhraní pomocí služby Apiary.



Obrázek 6.2: Analyzování aplikačního rozhraní pomocí služby Apiary.

6.3 Aplikace frameworku Ruby on Rails na serverové části služby

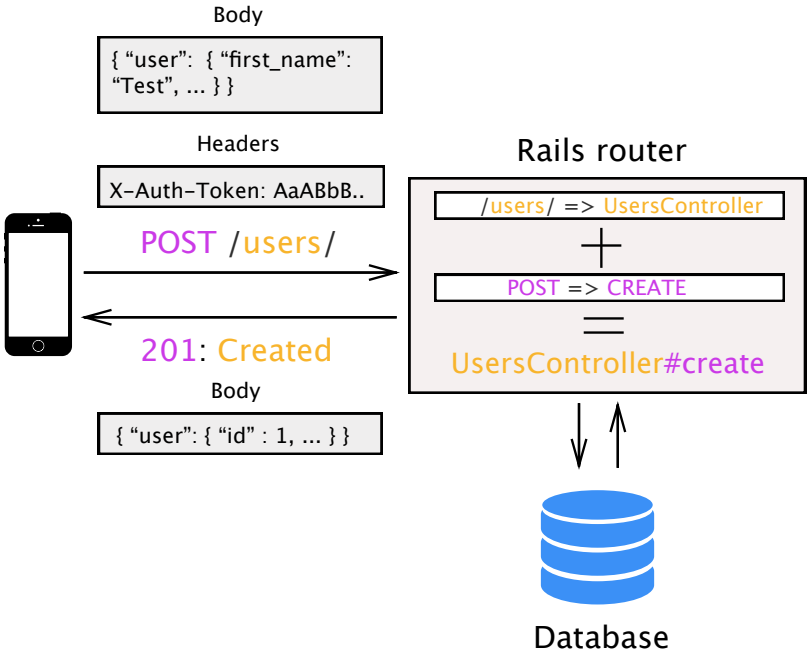
Na serverové části je umístěn webový server, který komunikuje s klientem a databází. Pokud klient vyžaduje data z databáze, zašle dotaz na server. Server se připojí k databázi a vrátí případná získaná data klientovi. Kapitoly 5.4 a 5.5 vznesly nároky na aplikační rozhraní a nastínily implementační nároky na server. Server musí implementovat REST službu, komunikovat s vhodně zvolenou databází, která je společná pro klienta na dvou rozdílných platformách, serializovat získaná data a podporovat autentizaci. U aplikačního rozhraní je vhodné ukládání dat do mezipaměti vzhledem k možnosti velkého počtu dotazů na server. Samotnou serverovou službu lze implementovat v různých programovacích jazycích a aplikačních rámcích nad těmito jazyky.

Při vývoji serverové části u aplikace Gymber byl zvolen aplikační rámec *Ruby on Rails*, který je založen na jazyce *Ruby*. Aplikační rámec Ruby on Rails stojí na architektuře Model – View – Controller (*MVC*) [13]. Architektura MVC rozděluje jednotlivé odpovědnosti práce s daty mezi zmíněné komponenty aplikace a byla blíže popsána v kapitole 3.2. Při implementaci aplikačního rozhraní, které bylo představeno v kapitole 5.4, je hlavní částí serverové služby model, řadič a databáze. Pohled u aplikačního rozhraní tvoří serializovaná data zasílaná řadičem klientovi. V Rails je známý koncept „konvence má přednost před konfigurací“, kdy při vhodně zvolených názvech řadičů, modelů a tabulek v databázi není nutná žádná konfigurace a aplikační rámec dle svých konvencí zajistí komunikaci mezi komponentami [11].

Na obrázku 6.3 je zobrazeno schéma procesu komunikace klienta – mobilní aplikace a serveru. Klient zašle zprávu obsahující ve svém těle nutné informace k úspěšné registraci uživatele a data jsou posílána ve formátu JSON. V hlavičce každé zprávy, která je odesílána na server, je nutné vkládat autentizační token. Vzhledem k tomu, že služba využívá REST konceptu, je při registraci uživatele nastavena HTTP metoda *POST* a cílová adresa na serveru bude ve formátu */users/*. Zpráva dorazí na server, kde je zpracována pomocí směrovače aplikačního rámce Ruby on Rails. Směrovač zjistí, že cílová adresa souvisí s uživateli a použije řadič uživatelů. Podle metody HTTP protokolu se upřesní funkce, kterou bude řadič dotaz zpracovávat. V uvedeném příkladě byla využita metoda POST. Směrovač zprávu přepošle řadiči uživatelů, který provede metodu *create*. Metoda create na řadiči uživatelů je zodpovědná za vytvoření nového uživatele. V této metodě dojde k volání modelu, kde je provedena validace zaslaných dat. Pokud jsou data validní, jsou zapsána do databáze. Při zápisu se vygeneruje unikátní identifikační číslo uživatele a autorizační token. Informace jsou při úspěšné registraci nového uživatele zaslány zpět klientovi a je nastaven kód v hlavičce HTTP protokolu na číslo *201*, které odpovídá úspěšnému vytvoření uživatelského účtu. Na straně klienta dojde k persistenci autorizačního tokenu, který je poté používán k autentizaci uživatele.

Při procesu registrace uživatele se mohou vyskytnout následující chyby:

- Z důvodu privátnosti aplikačního rozhraní je první akcí, kterou server vykoná při obdržení zprávy autorizace klienta – mobilní aplikace. Pokud je poskytnutý autorizační token nevalidní, server vrací zprávu informující klienta o neúspěšné autorizaci.
- V případě poskytnutí špatné kombinace adresy a metody HTTP protokolu ze strany klienta, zachytí směrovač aplikačního rámce Ruby on Rails chybu a vrací odpověď nesoucí informaci o tom, že není definovaná metoda pro zaslanou kombinaci adresy a metody HTTP protokolu.



Obrázek 6.3: Schéma REST služby běžící nad aplikačním rámcem Ruby on Rails. Obrázek zachycuje komunikaci mobilní aplikace se vzdáleným serverem a zobrazuje přenášené hlavičky HTTP protokolu sloužící k autentizaci uživatele. Na serverové části znázorňuje způsob zpracování přijatého dotazu aplikačním rámcem Ruby on Rails a komunikaci s databází.

- Při validaci dat a jejich následnému ukládání do databáze může dojít k zjištění nesrovnalostí v zaslaných datech. Chyba je interpretovaná klientovi a nastane zrušení procesu registrace uživatele. Aplikační rámec Ruby on Rails v případě výskytu chyby generuje čitelný popis a chybový kód, který je zasílán zpět klientovi. V mobilní aplikaci je tedy možné ze získaných odpovědí informovat uživatele a zajistit tak úspěšnou registraci.

Stejný princip komunikace znázorněný na obrázku 6.3 a popsaný výše, využívají i další části aplikace nejen na platformě iOS, ale i na systému Android.

6.4 Datový model aplikace na serverové části služby

Tato kapitola se zaměřuje na řešení, jak uložit prezentované informace poskytované uživateli v mobilní aplikaci. V kapitole 5.2 byl představen návrh mobilní aplikace, kde jsou zobrazeny informace o lokalitách vhodných ke cvičení, nabídky k fitness, uživatelské profily a jiné. Všechny zmíněné údaje je nutné uchovávat v centrálním úložišti – databázi spolupracující se serverem. Na obrázku 6.4 je zobrazen ER diagram databáze na vzdáleném serveru.

Diagram zahrnuje jedenáct entit, z nichž čtyři řadíme mezi hlavní entity – **Uživatel**, **Nabídka**, **Místo** a **Administrátor**. **Administrátor** vystupuje jako osamocená entita, nepřístupná z aplikačního rozhraní. Entita **Uživatel** obsahuje aktivační a autorizační tokeny, které byly blíže popsány v kapitole 5.4. Podpora přihlašování pomocí služby *Facebook* je zajištěna uchováváním identifikačního čísla a tokenu, poskytnutých z aplikačního rozhraní služby. Profilové obrázky jsou uchovávány na cloudovou službu **Amazon Web Services S3**.

Uživatelé, obdobně jako na sociální síti Twitter, se mohou vzájemně „sledovat“. Databáze zmíněnou funkci podporuje pomocí entity **Relationship**, která představuje vazbu mezi jednotlivými uživateli a obsahuje identifikační číslo sledujícího a sledovaného. Z návrhu je patrné, že databáze je denormalizovaná. Pomocné entity udávají typ místa nebo nabídky, zeměpisnou lokalitu a další.

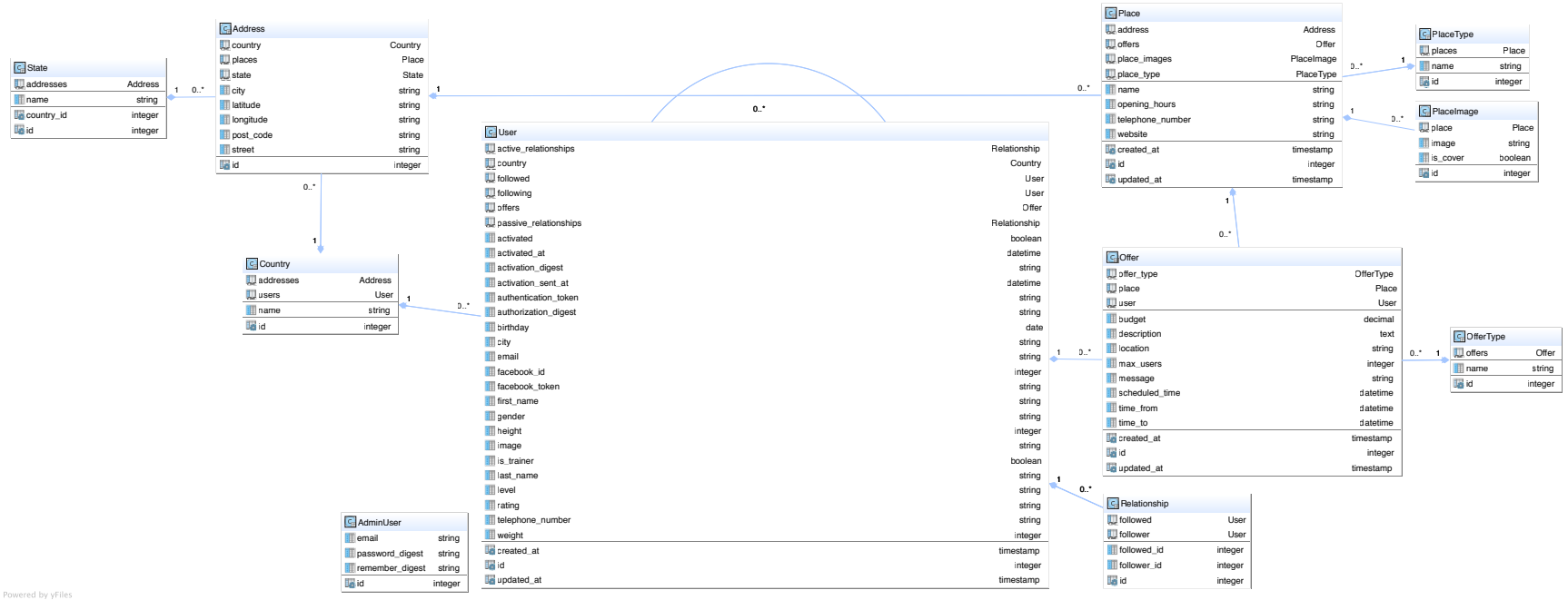
Entita **PlaceImage** byla vytvořena pro ukládání více fotografií vztahujících se k jedné lokalitě. Obdobně jako u profilových obrázků uživatelů jsou fotografie míst ukládány do cloudové služby.

Atribut **opening_hours** entity **Place** obsahuje informace o případných otevíracích hodinách fitness lokality. Údaje jsou serializované do jednoho řetězce pomocí aplikačního rámce Ruby on Rails. K této serializaci dochází na modelu Place a atribut umožňuje efektivní uložení otevíracích hodin a jejich následné zpracování na straně klienta.

Ve vývojovém a testovacím prostředí webové aplikace byl použit relační databázový systém **SQLite**. SQLite na rozdíl od jiných relačních databázových systémů nepracuje na principu klient–server, ale databáze je distribuována společně s aplikací. Databáze je ukládána jako jeden soubor na zařízení. Vzhledem k absenci architektury klient–server, aplikace využívající databáze SQLite vyžadují méně konfigurace [15]. Řízení přístupu je řešeno pomocí oprávnění souborového systému, udělených databázovému souboru.

V produkčním prostředí je nasazen relační databázový systém **PostgreSQL**. Oproti SQLite, relační databázové systémy PostgreSQL a MySQL pracují na principu klient–server. Na rozdíl od MySQL je PostgreSQL objektový relační databázový systém. Skutečnost, že PostgreSQL je distribuován jako open–source projekt, vedla k vytvoření silné komunity vývojářů a vzniklo mnoho rozšíření pro zmíněný databázový systém. PostgreSQL je spolehlivý a rozšiřitelný relační databázový systém implementovaný podle standardu SQL s důrazem na integritu dat [17]. Nevýhodou PostgreSQL oproti MySQL je pomalejší operace čtení dat a menší podpora databázového systému ze strany hostingových služeb.

Webová aplikace je společně s databází hostována na službě Heroku, která plně podporuje aplikační rámec Ruby on Rails a relační databázový systém PostgreSQL.



Obrázek 6.4: ER diagram databáze použité v serverové části služby.

6.5 Persistence dat v aplikaci

V mobilní aplikaci je potřeba ukládat uživatelská data k zajištění omezené funkčnosti aplikace i bez fungujícího internetového připojení. Uživatel by měl mít možnost procházet profil, přijaté a naposledy stažené nabídky a místa k fitness aktivitě. Akce jako přijmutí či odmítnutí nabídky nelze bez internetové konektivity provést, ale uživatel má možnost zkontrolovat dříve aktualizovaný stav. K uložení dat byl na platformě iOS použit aplikační rámec Core Data.

Core Data je aplikační rámec pro ukládání dat v systémech iOS, OS X, watchOS a tvOS. Objekty jsou načítány z datového skladu pomocí Core Data a je z nich vytvářen graf objektů. Aplikační rámec Core Data je zodpovědný za správu grafu objektů a životního cyklu objektů v něm. Vzhledem k tomu, že programátor nepracuje přímo s relační databází nebo jiným datovým skladem, ale s vytvořeným grafem objektů, jsou umožněny vysokoúrovňové operace nad daty, které jsou nezávislé na datovém skladu. Práce s grafem objektů nese i nevýhody, a to především pokud upravujeme velké množství dat zároveň. Pokud bychom měli entitu *Auto* s vlastností *Barva* a chtěli všem existujícím záznamům aut změnit barvu na červenou, aplikační rámec Core Data by načetl všechny objekty do grafu objektů a v něm prováděl změnu barvy. Zmíněné řešení je vysoce neefektivní, na rozdíl od odpovídající operace v relační databázi. K zamezení vytváření grafu objektů při dávkovém zpracování lze využít aplikační rozhraní Core Data a použít dedikované funkce pro práci s dávkovými soubory.

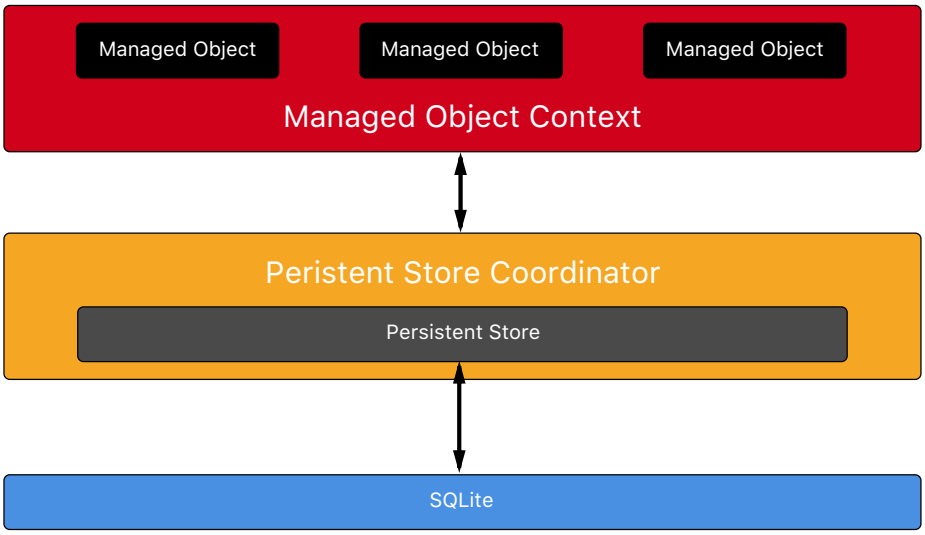
Core Data umožňuje vytvářet vazby mezi entitami a automaticky aktualizuje inverzní relace. K vytváření entit a vazeb mezi nimi lze použít tabulkového nebo grafického editoru, který je součástí vývojového prostředí Xcode. Na rozdíl od relačních databází není nutné v Core Data vytvářet M:N tabulky, stačí pouze definovat M:N vazbu mezi požadovanými entitami [1].

Obrázek 6.5 prezentuje základní vrstvu aplikačního rámce Core Data. Vrstva je složená ze čtyř hlavních komponent – spravovaných objektů (managed objects), kontextu spravovaných objektů (managed object context), koordinátoru datového skladu (persistent store coordinator) a datového skladu (persistent store). Datový sklad představuje persistentní místo k uložení dat – např. databázi SQLite.

Spravovaný objekt (**Managed Object**) je objekt, který představuje záznam z datového skladu [3]. Spravovaný objekt spadá do kontextu spravovaných objektů a obsahuje referenci na popisný objekt entity, která udává, jakou entitu objekt reprezentuje. Tímto způsobem může spravovaný objekt fungovat jako model pro entitu a programátor nemusí vytvářet dvě různé třídy pro stejnou entitu v datovém modelu.

Kontext spravovaných objektů (**Managed Object Context**) zodpovídá za správu a shromažďování spravovaných objektů. Kontext ovlivňuje životní cyklus spravovaných objektů, jejich validaci, udržuje inverzní vazby mezi entitami a zajišťuje operace undo, resp. redo. Kontext spravovaných objektů je propojený s nadřazeným objektem datového skladu. Většinou tímto objektem bývá koordinátor datového skladu, ale může se jednat i o jiný kontext spravovaných objektů. Při zaslání dotazu k získání dat, kontext spravovaných dat deleguje zprávu svému nadřízenému objektu, který vrátí objekty odpovídající požadovanému dotazu. Všechny změny provedené nad kontextem spravovaných objektů nejsou uloženy, dokud programátor explicitně nezašle zprávu kontextu k jeho uložení.

Koordinátor datového skladu (**Persistent Store Coordinator**) představuje centrální objekt ve vrstvě Core Data a spolupracuje s datovým skladem a spravovanými objekty. Pro kontext spravovaných objektů zapouzdřuje skupinu datových skladů jako jeden spojený



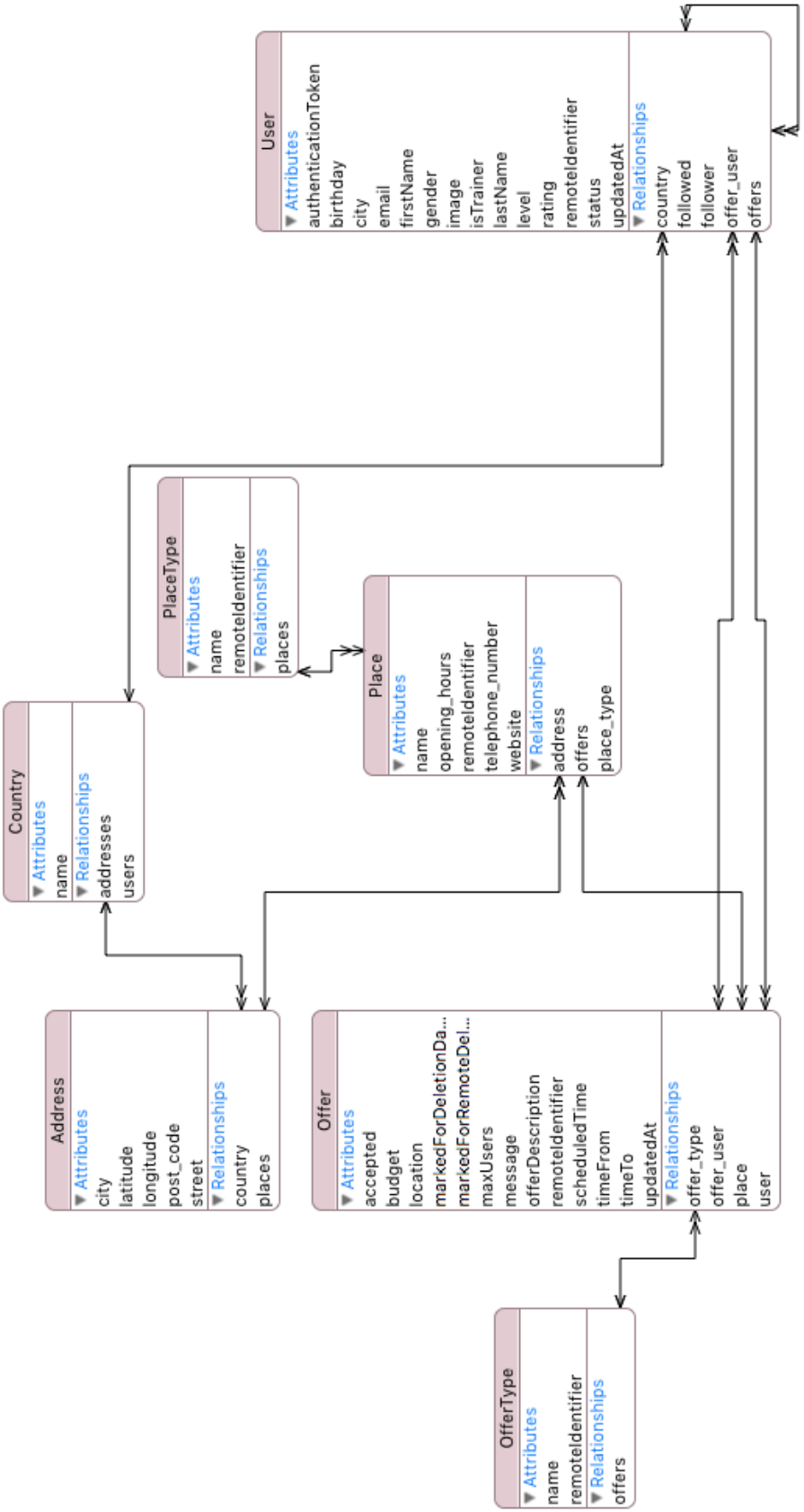
Obrázek 6.5: Jednotlivé komponenty v základní vrstvě aplikačního rámce Core Data. [3]

sklad. Obsahuje reference mezi spravovaným objektem a entitou v datovém skladu.

Datový sklad (**Persistent Store**) je úložiště obsahující spravované objekty. Core Data nativně podporuje perzistenci do tří souborových typů – binární, XML a SQLite. Vývojářům je umožněno implementovat vlastní datový sklad. Data je možné ukládat i v paměti, kde po skončení procesu zaniknou. Entity v datovém skladu jsou definovány spravovanými objekty, které je vytvořily. Dojde-li ke změně modelu v aplikaci, žádný z vytvořených datových skladů patřících původnímu modelu, nelze otevřít. Řešením je provedení migrace obsahu datového skladu do nového formátu.

Na obrázku 6.6 je vizualizován datový model mobilní aplikace Gymber. Na rozdíl od datového návrhu, který je využit na serverové části služby a byl zmíněn v kapitole 6.4 obsahují jednotlivé entity atributy nutné k implementaci synchronizace mezi lokální databází v mobilní aplikaci a centrální databází umístěnou na serveru. Takovým atributem je *remoteIdentifier*, který nese informaci o identifikačním čísle objektu v databázi na serveru. Při načítání dat ze vzdálené databáze pomocí aplikačního rozhraní jsou jednotlivé objekty analyzovány. Po získání všech požadovaných atributů dojde k vyhledání záznamu v lokální databázi. Je-li hledaný záznam nalezen, objekt je již obsažen v lokální databázi a je nutné zjistit, zda je shodný s objektem uloženým ve vzdálené databázi. Zmíněný krok je proveden prostřednictvím atributu *updated_at*. Je-li datové razítko v lokální databázi starší než razítko přijaté, dojde k aktualizaci objektu v lokální databázi. V opačném případě je vytvořen PATCH dotaz na vzdálený server. V obou případech se objekty sesynchronizují. Nenastane-li při načítání dat nalezení objektu pomocí vzdáleného unikátního čísla, lokální databáze neobsahuje analyzovaný objekt a dojde k jeho vložení.

Na rozdíl od návrhu, který je využit na serveru, neobsahuje datový model mobilní aplikace spojové tabulky a jednotlivé entity obsahují pouze atributy, potřebné pro fungování aplikace.



Obrázek 6.6: Datový návrh pro mobilní aplikaci.

Kapitola 7

Testování

Ke zjištění správné funkčnosti serverové části, mobilní aplikace a použitelnosti uživatelského rozhraní byly jednotlivé komponenty testovány. V kapitole 7.1 je popsán postup a výsledky testování grafického uživatelského rozhraní mobilní aplikace. Kapitola 7.2 se zaměřuje na testování serverové části služby, především řeší dobu odezvy aplikace. Paměťová a energetická náročnost mobilní aplikace je diskutována v kapitole 7.3.

7.1 Analýza uživatelského rozhraní

Výsledný návrh, zmíněný v kapitole 5.2 a implementovaný na platformě iOS, bylo nutné konfrontovat s uživateli. K zjištění zpětné vazby na uživatelské rozhraní od uživatelů byly vypracovány úkoly a strukturovaný dotazník.

Úkoly se skládaly ze zvládnutí základních úkonů, a to:

1. Vytvoření uživatelského účtu
2. Přihlášení se do aplikace
3. Vyhledání nabídky v seznamu nabídek
4. Zjištění informací o nabídce
5. Přijmutí nabídky

Strukturovaný dotazník obsahoval otázky, které měly za cíl zjistit názor uživatele na grafické rozhraní.

1. Působily na Vás některé prvky nejednoznačně? Pokud ano, které?
2. Vyhovuje Vám, jak byly jednotlivé informace v seznamu nabídek zobrazeny? Pokud ne, co by Vám vyhovovalo více?
3. Byly v seznamu nabídek obsaženy všechny pro Vás důležité informace k dané nabídce? Pokud ne, co Vám chybělo?
4. Narazili jste na problém při zadávání nové nabídky? Pokud ano, na jaký?
5. Byl pro Vás proces přijmutí nabídky vyhovující? Pokud ne, co Vám nevyhovovalo?

Tabulka 7.1: Registrace a přihlášení do aplikace.

	Vytvoření účtu	Přihlášení se
Respondent 1	26s	42s
Respondent 2	22s	15s
Respondent 3	40s	27s
Respondent 4	26s	17s

Tabulka 7.2: Operace s nabídkami.

	Vyhledání nabídky	Detail nabídky	Přijmutí nabídky
Respondent 1	8s	3s	20s
Respondent 2	3s	1s	7s
Respondent 3	2s	2s	19s
Respondent 4	3s	1s	10s

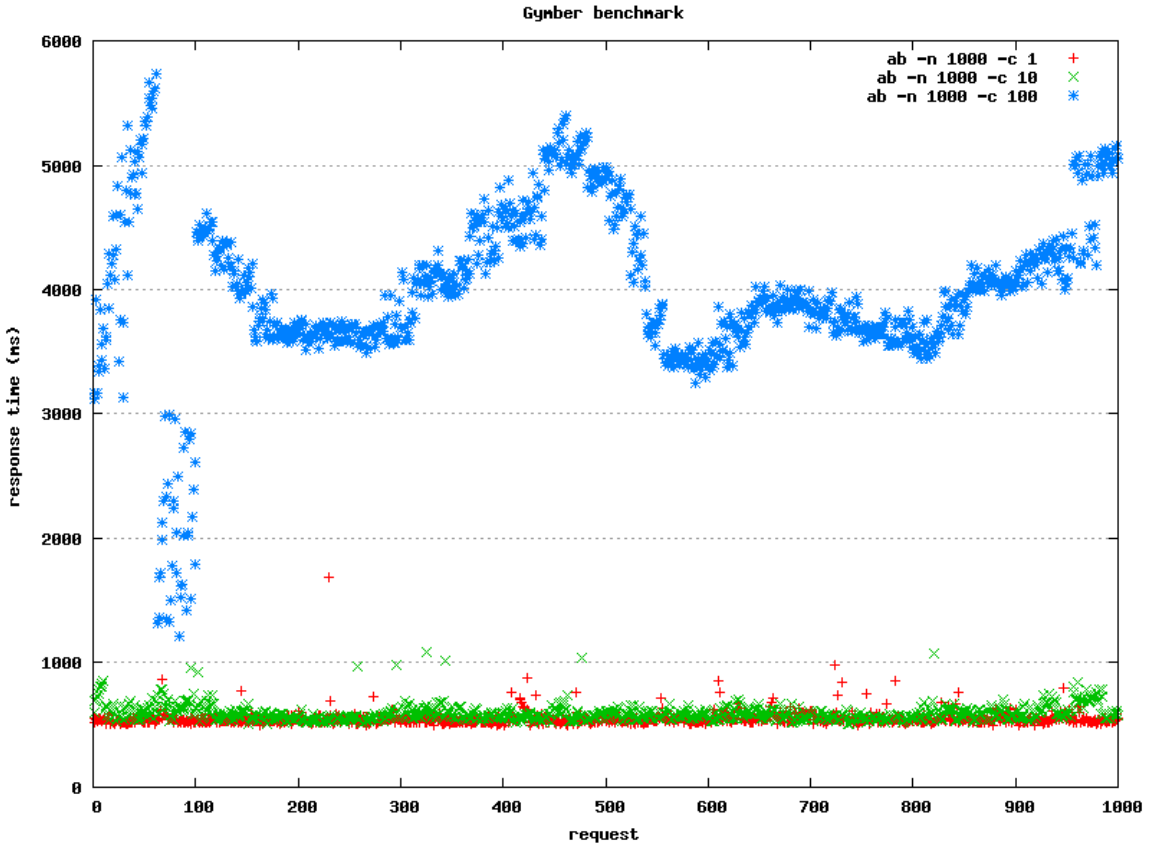
Testovacím zařízením byl mobilní telefon iPhone 5S s aktuální verzí iOS 9.3.1. Respondentům byly zadány jednotlivé úkoly a byl měřen čas jejich vypracování. Po splnění úkolu následovala krátká diskuze o provedeném procesu. U vybraných respondentů bylo testováno více částí aplikace.

Tabulka 7.1, resp. 7.2, zobrazuje naměřené časy jednotlivých respondentů, které byly zaokrouhleny na celé sekundy. První tabulka obsahuje časové údaje respondentů při vytváření účtu a přihlašování se do aplikace. Proces registrace a přihlášení byl úspěšný u všech respondentů. První respondent při přihlašování vícekrát zadal špatný email a naměřený čas převyšuje časy ostatních. Registrace byla pro respondenty přehledná, všechny naměřené časy jsou měřeny i se samotným vypisování údajů pomocí softwarové klávesnice. Při registraci, resp. přihlášení, uživatel nezadává heslo, ale na poskytnutý email je zaslán aktivační kód. První respondent zmínil, že místo posílání aktivačního kódu na email by raději obdržel kód pomocí SMS zprávy.

Vyhodnocení operací s nabídkami v mobilní aplikaci řeší tabulka 7.2. Respondenti měli za úkol z náhodné obrazovky v aplikaci přejít na seznam nabídek a vyhledat zadanou nabídku. Průměrně byl tento úkon bezproblémový. Přijmutí nabídky bylo velmi závislé na rychlosti psaní respondenta na softwarové klávesnici mobilního zařízení. Při přijímání nabídky první respondent jako problém uvedl jazykovou bariéru (aplikace byla prezentována v angličtině). Ze zmíněného vznikl podnět k vypracování lokalizované aplikace. Třetí respondent dal podnět k zasílání notifikací.

7.2 Zkoušení serverové části služby

Aplikační rozhraní a server jsou nedílnou součástí platformy. API bylo při návrhu dokumentováno, ale při implementování jednotlivých částí mobilní aplikace dochází k úpravám a vylepšování vybraných částí, které nebyly při návrhu zřejmé. Implementace aplikačního rozhraní na serverové části byla snadno testovatelná – pomocí REST klienta *Postman* byly zasílány dotazy a analyzovány odpovědi. Při spojení serverové části a mobilní aplikace je důležitá rychlost vzájemné komunikace a výměny dat. Na serverové části dochází k optimalizaci pomocí ukládání dat z databáze do vyrovnávací paměti, využití indexů v databázi a vyhovujícímu návrhu aplikačního rozhraní. Kapitola se věnuje zhodnocení serverové části



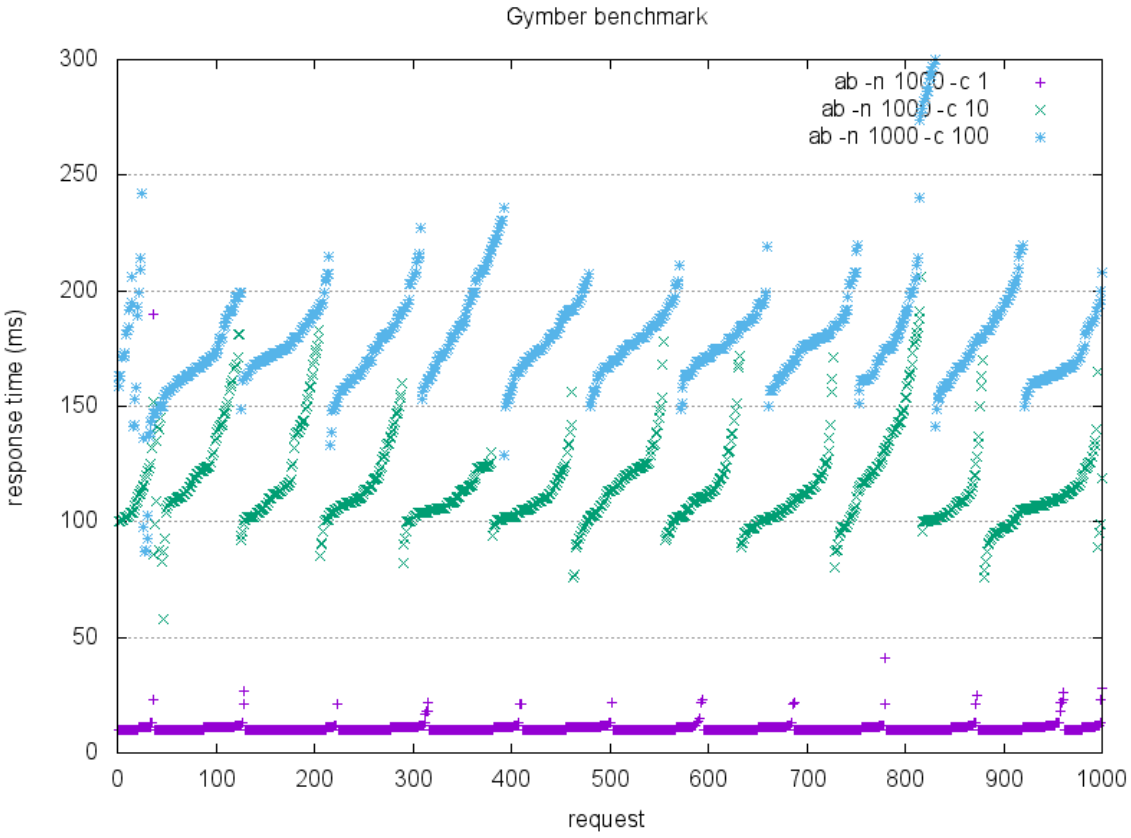
Obrázek 7.1: Průběh testování aplikace umístěné na službě Heroku.

a zkoumá její použitelnost v provozu. Ve zmíněných případech docházelo k měření serverové části pomocí volně dostupného nástroje **Apache Benchmark**[1]. Jednotlivé testy se skládaly ze zasílání 1000 dotazů. K vytvoření zátěže byl parametr, udávající počet souběžně zasílaných dotazů, nastavován v rozmezí 1, 10 a 100 dotazů. Pomocí zmíněného testovacího nástroje probíhalo zasílání dotazů s HTTP metodou GET na výpis všech nabídek z databáze – adresa [https://gymber.herokuapp.com | localhost:3000]/api/v1/offers/. Data získaná pomocí nástroje Apache Benchmark byla následně vizualizována volně dostupným programem **gnuplot**[2].

Graf zobrazený na obrázku 7.1 znázorňuje dobu odezvy na jednotlivé dotazy. Při měření byla serverová část umístěna na službě Heroku. Doba odezvy při měření s parametrem udávajícím počet souběžně zasílaných dotazů v rozmezí 1 a 10 dotazů, se pohybovala do jedné sekundy. Výkyvy jsou ojedinělé, doba odezvy byla průměrně stejná. Při zatížení, kdy bylo na server zasíláno souběžně 100 dotazů každou sekundu, se doba odezvy rapidně zvýšila. V nejhorším případě byla doba odezvy až 6 sekund.

Ze zjištěných výsledků je zřejmé, že využití neplaceného vzdáleného hostingu nabízejícího minimální paměťový a procesorový výkon je nevhodné pro nasazení aplikace do provozu. Samotný server, na kterém je služba provozována, je dle analýzy sítě pomocí nástroje **traceroute**[3] až 25. uzlem v síti. Změřená doba odezvy pomocí nástroje traceroute, který

[1] https://httpd.apache.org/docs/2.2/programs/ab.html
[2] http://www.gnuplot.info
[3] http://www.freebsd.org/cgi/man.cgi?query=traceroute



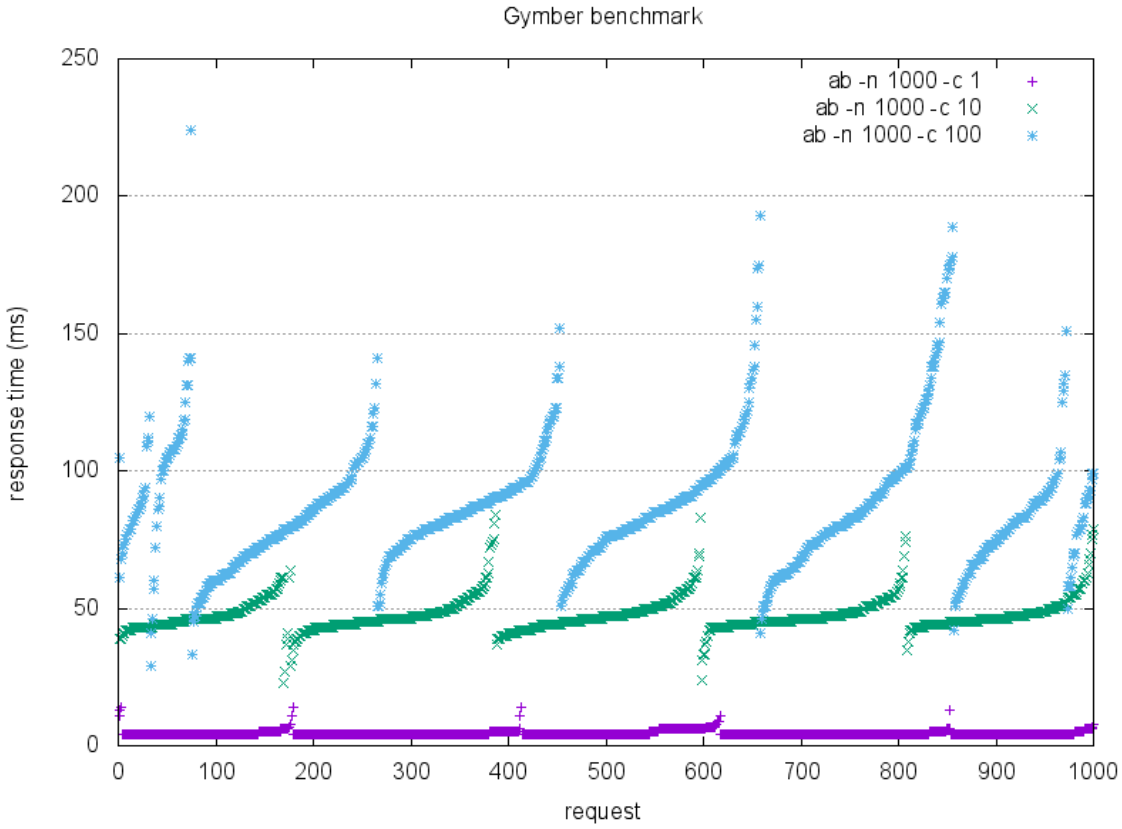
Obrázek 7.2: Průběh testování aplikace umístěné na lokálním stroji se sníženým výkonem CPU.

zasílal dotaz na server umístěný na službě Heroku, v nejlepším případě neklesla pod 135 ms, průměrná doba byla 164 ms. Často se doba odezvy pohybovala v rozmezí 250 ms, v nejhorším případě byla doba odezvy až 2,2 sekundy. Průměrné hodnoty byly zjištěny ze 100 měření.

Při nasazení mobilní aplikace do reálného provozu nelze využít neplaceného programu na službě Heroku. Pokud by byla webová aplikace hostována na Heroku, bylo by nutné využití placených programů s možností výběru lokality serveru a zvýšení paměťového a procesorového výkonu nebo použití jiné placené služby, případně vlastního serveru s odpovídající konektivitou.

Obrázek 7.2 obsahuje graf znázorňující dobu odezvy serverové části umístěné na lokálním stroji. Za účelem simulace větší zátěže serveru vlivem sdílení s jinými aplikacemi byl uměle snížen výkon procesoru nastavením do úsporného režimu. Snížení výkonu procesoru vedlo k prodloužení doby odezvy. Na rozdíl od výsledků, prezentovaných v grafu na obrázku 7.1, doba odezvy serveru umístěného na lokálním stroji při stejné zátěži, se sníženým výkonem procesoru, nepřesahuje 300 ms.

Na obrázku 7.3 je zobrazen graf, který obsahuje výsledky výkonnostního měření na lokálním stroji. Až na jeden výkyv je doba odezvy pod 200 ms. Při souběžném zasílání 100 dotazů za sekundu, je průměrná doba odezvy 83 ms. Z naměřených výsledků na lokálním stroji lze usoudit, že serverová část je poměrně dobře optimalizovaná. Všechny metody v aplikačním rámci Ruby on Rails zasílající data klientovi využívají ukládání do vyrovná-



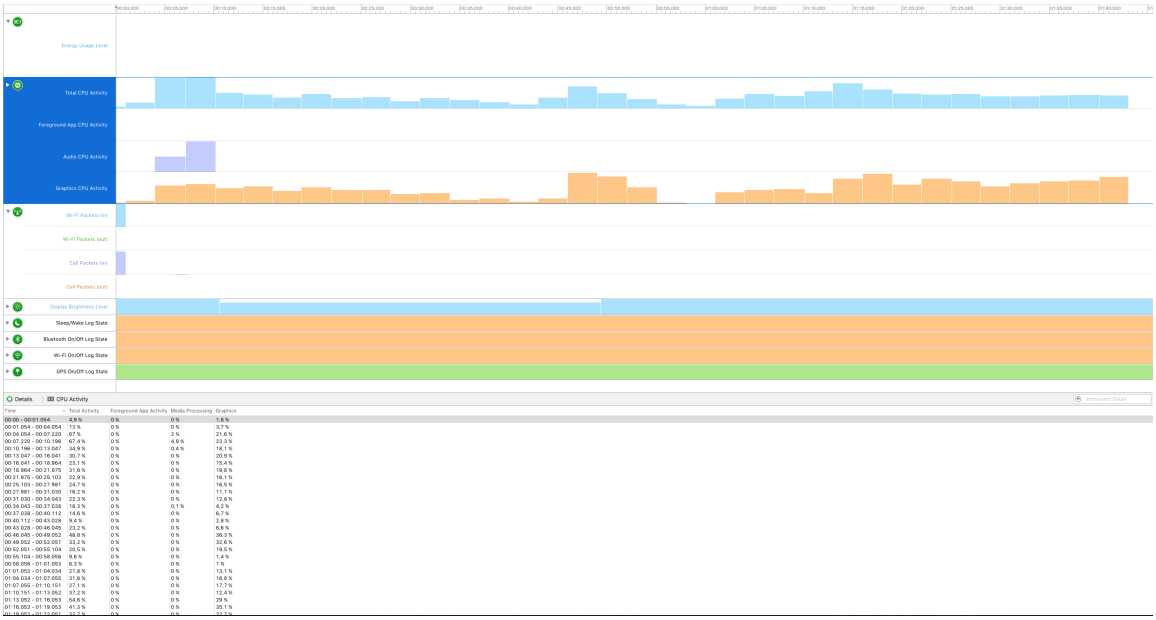
Obrázek 7.3: Průběh výkonového testování aplikace umístěné na lokálním stroji.

vací paměti. V databázi jsou vytvořené indexy na sloupce, podle kterých dochází k častému vyhledávání, např. **authentication_token** u entity **User** a další.

7.3 Testování mobilní aplikace

Při testování serverové části byla zmiňována doba odezvy. Na mobilním zařízení jsou nároky na mobilní aplikaci odlišné. V současné době není na mobilních zařízeních problém výpočetní výkon zařízení, ale programátor je limitován paměťovou a energetickou náročností aplikace.

Snaha výrobců je zvyšovat datová úložiště na mobilních zařízeních, ale problémem zůstává velikost aplikací. Vzhledem k tomu, že mobilní zařízení uživatel využívá i jako fotoaparát a videokameru, rychle vyčerpá již tak omezenou kapacitu paměti a je nucen odstraňovat aplikace. Druhým problémem je samotná instalace aplikace stahované z internetového obchodu dané platformy. Pokud je aplikace datově náročná, může instalace v závislosti na typu a kvalitě internetového připojení trvat neúměrně dlouho. Mobilní zařízení disponují omezenou velikostí operační paměti RAM a operační systém může násilně ukončit proces paměťově náročné aplikace. Posledním faktorem, který musí programátor zohlednit, je životnost baterie mobilního zařízení. Aplikace by měla být optimalizovaná a energeticky nenáročná. V opačném případě dochází k odstranění z důvodu znatelného zkracování životnosti baterie mobilního zařízení. Tato kapitola se zabývá analýzou zmíněné problematiky v aplikaci Gymber pro platformu iOS.



Obrázek 7.4: Testování mobilní aplikace pomocí nástroje Instruments. Obrázek slouží k ilustraci procesu testování energetické, paměťové a procesorové náročnosti. Instruments nepodporuje exportování provedeného měření ve větším rozlišení nebo ve formátu CVS.

Testování mobilní aplikace bylo provedeno pomocí nástroje Instruments, který lze použít po nainstalování prostředí Xcode. Referenčním strojem je iPhone 5S s operačním systémem iOS ve verzi 9.3.1. Velikost mobilní aplikace po kompilaci na zařízení byla 18,5 MB. Při testování se v aplikaci nenacházely žádné přídavné aplikační rámce. Před načtením dat ze serveru jsou lokálně uloženy jen obrázky použité v aplikaci a profilové fotky pro obě pohlaví používané v případech, kdy uživatel nenahrál vlastní fotografii. Velikost mobilní aplikace na cílovém zařízení lze zmenšit optimalizací překladače pro produkční verzi aplikace. Pro testování energetické náročnosti aplikace Instruments používá interval 0 – 20, kde 20 značí energeticky velmi náročnou aplikaci. Z výsledků měření získaných programem Instruments vyplývá, že spotřeba mobilní aplikace Gymber je zanedbatelná. Aplikace byla testována při běžném používání – registrace, přihlášení, prohlížení seznamu nabídek a lokalit, vytváření nových nabídek. Maximální využití procesoru bylo 43%, resp. 20% v případě grafického procesoru. Naměřené hodnoty využití procesorů byly vesměs stejné napříč měřeními, v jednom z měření se energetická náročnost rovnala 1. V průběhu testování byla zjištěna i maximální energetická náročnost 3. Zmíněné výsledky se nepodařilo reprodukovat. Síťová komunikace měla minimální vliv na chod aplikace. Vzhledem k umístění webové aplikace na službě Heroku docházelo při získávání dat ze vzdáleného serveru ke znatelné prodlevě. Problém doby odezvy byl zmíněn v kapitole 7.2.

Kapitola 8

Závěr

Tato práce představila postup návrhu mobilní aplikace Gymber, jejího uživatelského a aplikačního rozhraní a serverové části služby. Text uvádí čtenáře do problematiky mobilních operačních systémů a programování mobilních aplikací na platformu iOS. Shrnuje poznatky z analýzy podobných mobilních aplikací využívajících geografické blízkosti uživatelů a jejich srovnání.

Cílem práce bylo vytvoření prototypu mobilní aplikace zaměřené na vyhledávání partnerů do posilovny. V rámci vývoje bylo zapotřebí vytvořit serverovou část implementující navržené aplikační rozhraní a mobilní aplikaci komunikující se serverovou částí. Uživatelské rozhraní aplikace bylo programováno s důrazem na uživatelskou přívětivost. Výsledný prototyp umožňuje uživateli vytvořit nabídku ke cvičení, reagovat na nabídky ostatních uživatelů a procházet seznam lokalit vhodných k fitness aktivitě.

Provedená měření dokládají, že implementovaná serverová část je schopná obsloužit požadovaný počet uživatelů. Testování prokázalo, že aplikace je paměťově a energeticky nenáročná a uživatelsky přívětivá. Z vyhodnocení uživatelských odpovědí vyplynulo, že pro český trh je vhodné mobilní aplikaci lokalizovat do českého jazyka.

Prototyp poskytuje základ pro plnohodnotnou aplikaci. K zařazení mobilní aplikace do reálného provozu je nutná změna hostingové služby. Současně používaná služba Heroku vykazuje nepřiměřeně dlouhé doby odezvy. Pro další vývoj lze konečnou funkcionalitu aplikace rozšiřovat o podněty uživatelů z reálného prostředí.

Literatura

[1] Florian Kugler, D. E.: *Core Data*. CreateSpace Independent Publishing Platform, 2015, iSBN: 1518602649.

[2] Hegarty, P.: Developing iOS 9 Apps with Swift. lecture, 2016, https://itunes.apple.com/cz/course/1.-course-overview-introduction/id1104579961?i=367106842.

[3] Inc., A.: Core Data Core Competencies. documentation (english), October 2011, https://developer.apple.com/library/ios/documentation/DataManagement/Devpedia-CoreData/.

[4] Inc., A.: About the iOS Technologies. documentation (english), September 2014, https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html.

[5] Inc., A.: Core OS layer. documentation (english), September 2014, https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html.

[6] Inc., A.: Core Services layer. documentation (english), September 2014, https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html.

[7] Inc., A.: Media layer. documentation (english), September 2014, https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html.

[8] Inc., A.: *The Swift Programming Language*. Apple Inc., 2014, https://itunes.apple.com/us/book/swift-programming-language/id881256329.

[9] Inc., A.: Model–View–Controller. documentation (english), October 2015, https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html.

[10] Inc., A.: iOS Human Interface Guidelines. manual (english), March 2016, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/.

[11] RailsGuides: Rails Routing from the Outside In. documentation (english), http://guides.rubyonrails.org/routing.html.

[12] Ries, E.: *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011, iSBN: 9780307887894.

[13] Sam Ruby, D. H. H., Dave Thomas: *Agile Web Development with Rails 4*. Pragmatic Bookshelf, 2011, iSBN: 1937785564.

[14] Schlafman, S.: Uberification of the US Service Economy. blogpost (english), April 2014, http://schlaf.me/post/81679927670.

[15] SQLite: Distinctive Features Of SQLite. documentation (english), https://sqlite.org/different.html.

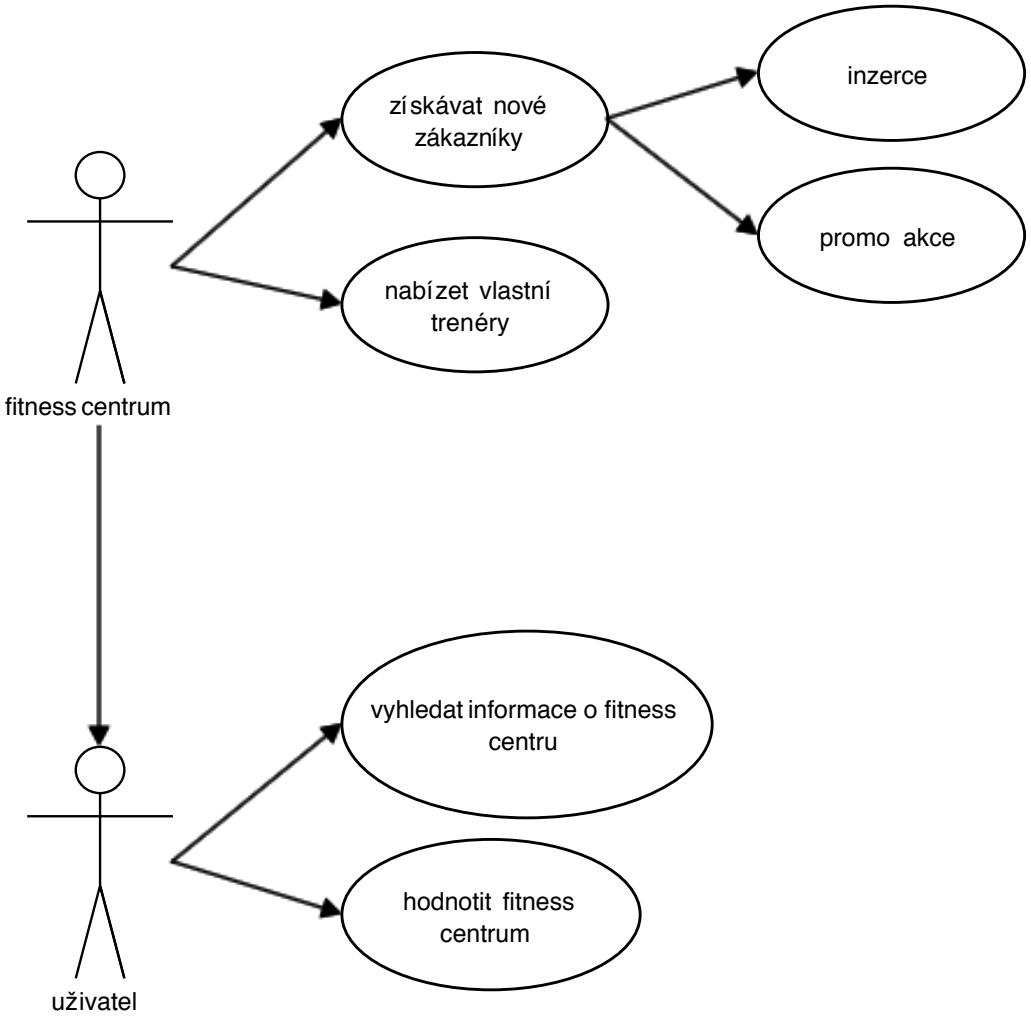
[16] Steve Blank, B. D.: *The Startup Ownerś Manual: The Step–By–Step Guide for Building a Great Company*. K & S Ranch, 2012, iSBN: 0984999302.

[17] Tezer, O.: SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. February 2014, blogpost on Digital Ocean (english).

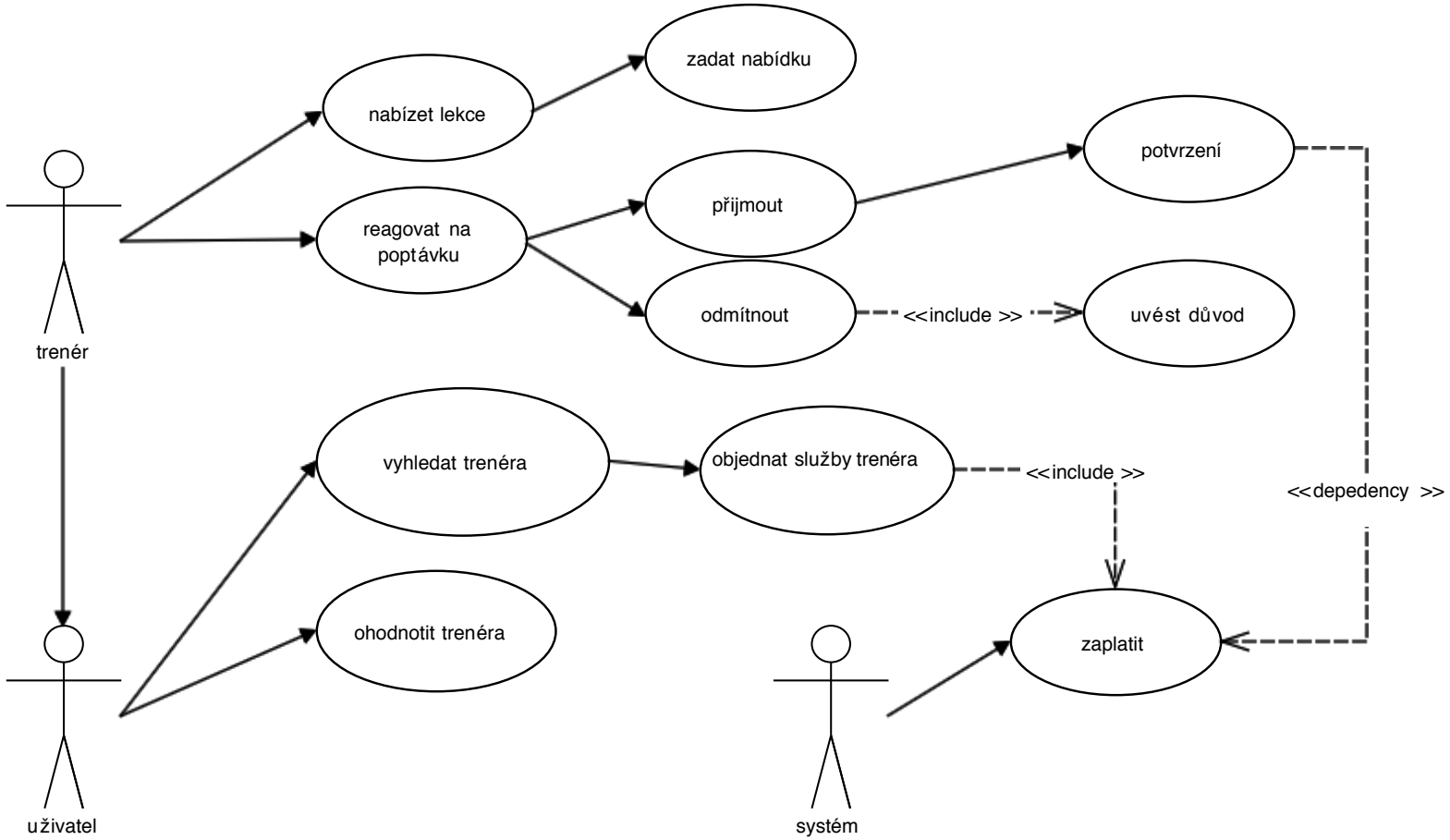
Přílohy

Příloha A

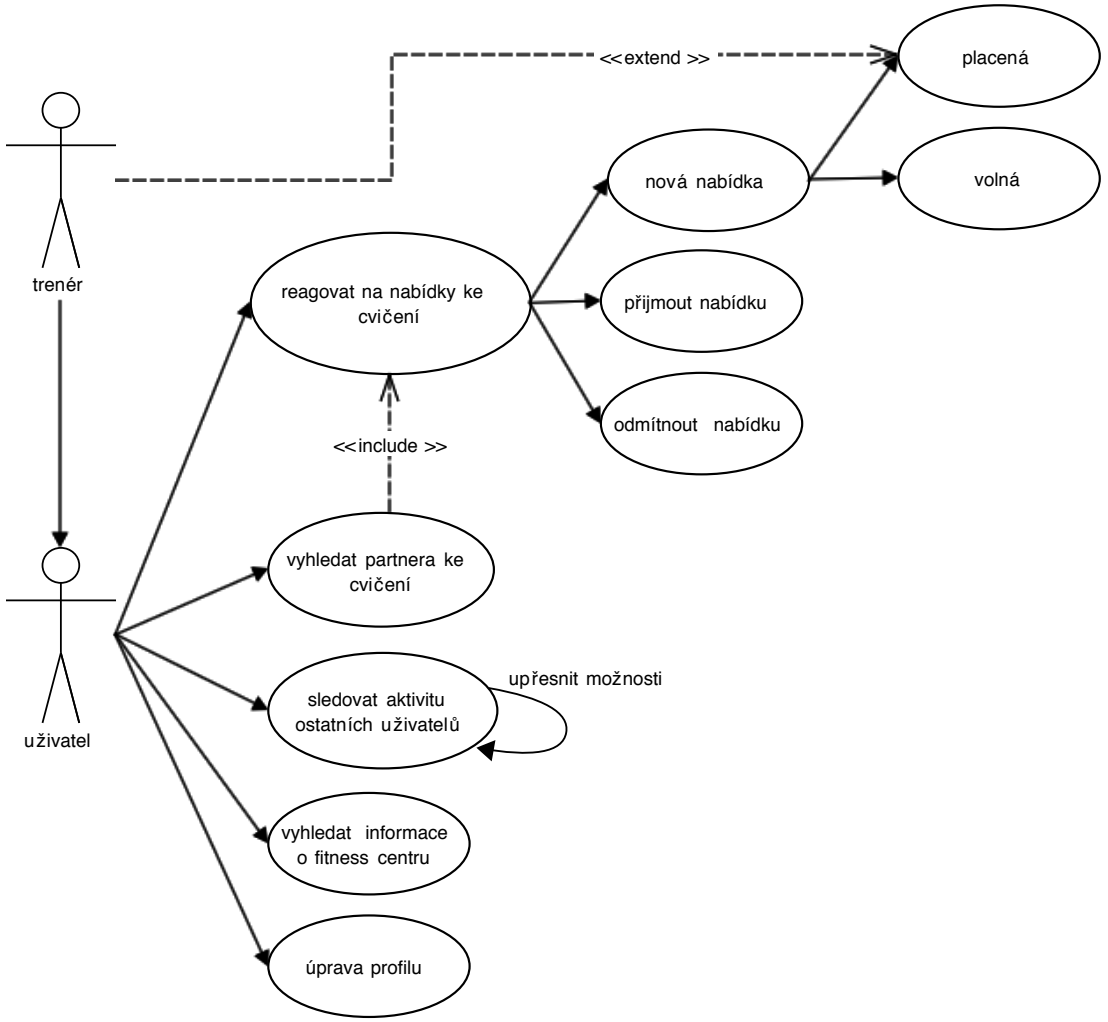
Diagramy případů užití



Obrázek A.1: První diagram případu užití – vyhledávání a hodnocení fitness center.



Obrázek A.2: Druhý diagram případu užití – platforma pro objednání placených lekcí s trenérem.



Obrázek A.3: Výsledný diagram případu užití – aplikace Gymber.