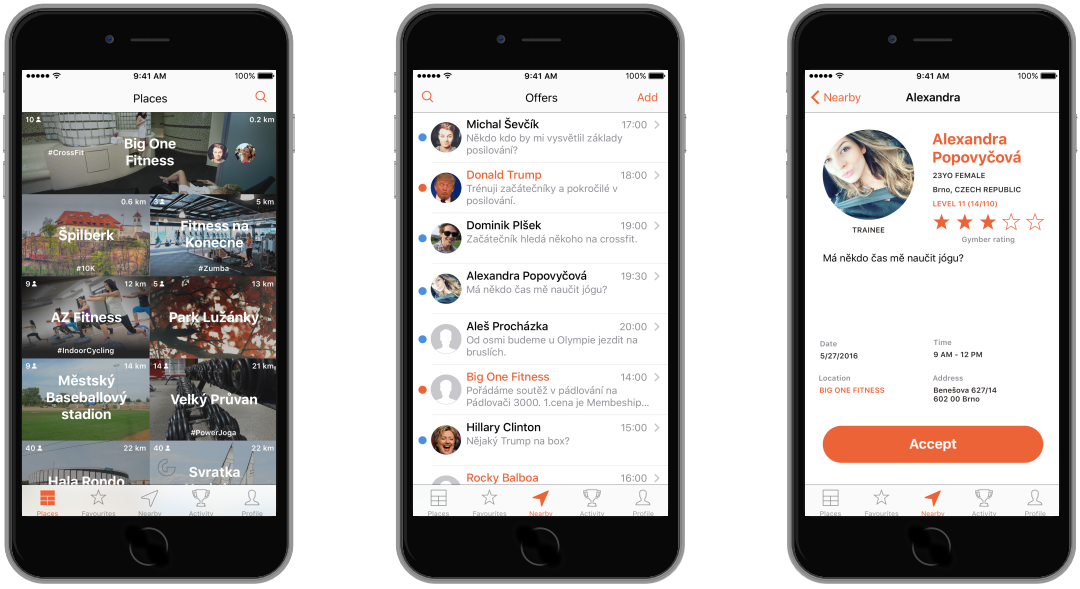
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Gymber – Mobilní aplikace pro vyhledávání partnerů do posilovny

Dominik Plšek*



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 projektu 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. Článek předkládá témata návrhu uživatelského rozhraní, implementaci na platformě iOS a návrh serverové části služby a API podporujícího multiplatformnost mobilní aplikace Gymber.

Klíčová slova: Mobilní aplikace — Startup — iOS — UI — UX

Přiložené materiály: *N/A*

*xplsek00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

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.

Příkladem řešení problému vysoké ceny za službu je provozovatel taxi služby *Uber*, který využívá platformu založenou na sdílené ekonomice v sektoru služeb. Po vzniku společnosti Uber mnoho firem využívá podobného obchodního modelu a postupně se zavedl pojem „uberifikace“ [1]. Pomocí nabízené mobilní aplikace se může každý řidič automobilu stát řidičem taxi. Platby za jednotlivé jízdy jsou automaticky zpracovávány službou a pomocí GPS lokace zákazníka v autě je vypočítáváno jízdné bez využití taxametru.

Sdílení polohy a sbírání bodů za časté návštěvy stejného místa nabízí mobilní aplikace *Swarm*, která vznikla jako *spin-off* aplikace *Foursquare*. V aplikaci Swarm se uživatel označí v místě, ve kterém se nachází a je za tuto aktivitu odměněn virtuálními body a nálepkami. Pokud uživatel navštěvuje určité místo častěji než ostatní uživatelé, stane se v daném místě starostou[3]. Mnoho podniků, především z restauračního sektoru, odměňují uživatele (starosty) slevami.

V oblasti fitness lze nalézt aplikace, které se snaží řešit problém chybějících partnerů ke cvičení. Platforma *Wellsquad* umožňuje vytvoření vlastního týmu cvičících partnerů, které lze pomocí aplikace kontaktovat a zaznamenávat aktivitu týmu. Mobilní aplikace *GymComrade* lokalizuje polohu uživatele, zobrazuje mapu uživatelů, kteří jsou v dosahu a umožní navázat kontakt s domluvou společného cvičení.

Mobilní aplikace Gymber vychází z konceptu, že každý člověk se zkušenostmi s cvičením, se může stát trenérem. Nezobrazuje na mapě uživatele v dosahu, ale 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.

Na rozdíl od zmíněných aplikací zaměřených pouze na fitness kluby, Gymber podporuje společné aktivity uživatelů i v jiných oblastech, například běhání, bruslení a další. 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.

Článek předkládá témata návrhu uživatelského rozhraní, implementaci na platformě iOS a návrh serverové části služby a API podporujícího multiplatformnost mobilní aplikace Gymber.

2. Návrh mobilní aplikace Gymber

Před samotným začátkem vytváření návrhů a prototypů bylo nutné definovat, co má obsahovat MVP[4] aplikace [2]. 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 [3]. Výsledný 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.

2.1 Uživatelské rozhraní

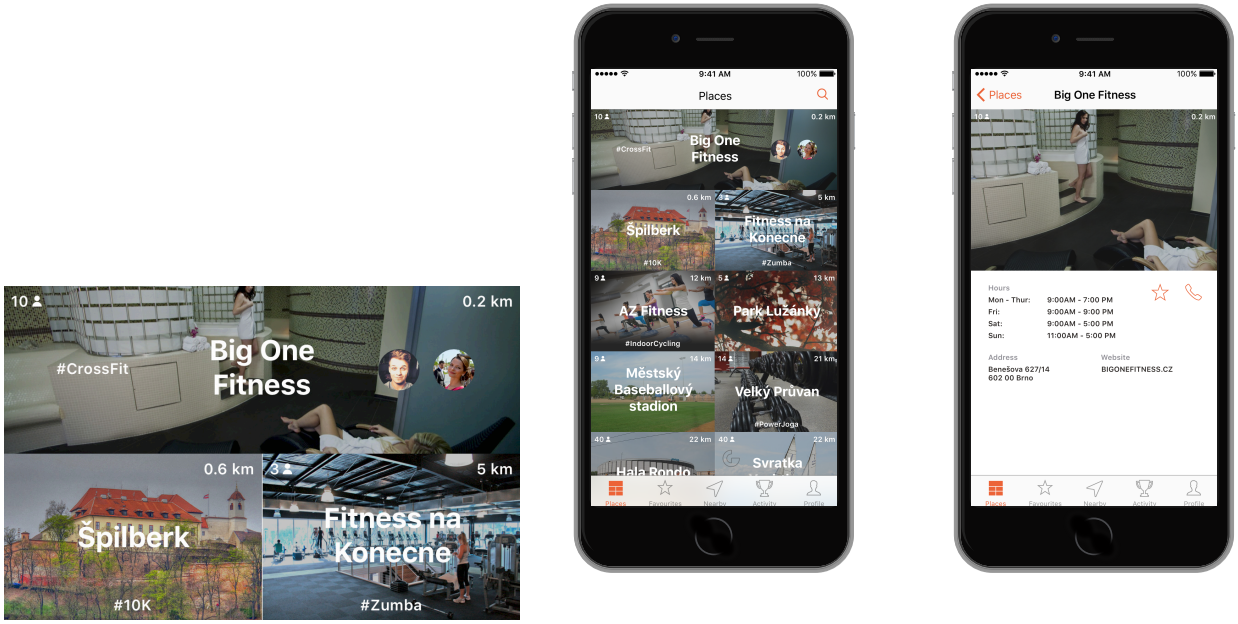
Každé uživatelské rozhraní má splňovat princip přehlednosti a jednoduchého ovládání [4]. Při návrhu uživatelského rozhraní pro Gymber bylo zapotřebí brát v potaz i to, že aplikace bude cílena na dvě rozdílné platformy – iOS a Android. Cílem bylo navrhnout vizuální styl, který je možné využít na obou dvou platformách s tím, že na každou z platforem se uživatelské prostředí optimalizuje dle konvencí dané platformy. Tento článek představuje výsledné uživatelské rozhraní pro platformu iOS. Na systému iOS je využito co nejvíce

[1] http://www.statisticbrain.com/gym-membership-statistics/

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

[3] Mayor

[4] Minimal Viable Product



Obrázek 1. Vizualizace obrazovky s výpisem a detailem míst vhodných k fitness.

základních systémových prvků, které jsou mírně upraveny s cílem vyhnout se vytváření vlastních grafických elementů, které by mohly být pro uživatele matoucí [4]. Jako příklad je možné uvést tzv. „hamburger menu“, které je na platformě iOS atypické, na rozdíl od platformy Android, kde se využívá velmi často.

Na obrázku 1 je ukázána 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.

Štítek[5], který vychází z populární sociální aplikace *Twitter*, informuje uživatele o zajímavých aktivitách nebo zdůrazňuje popularitu místa. Výpis, který využívá tabulkového zobrazení je vhodný z důvodu potencionálně velkého počtu míst a uživateli přehledně zobrazí požadovaná data. Na tento styl výpisu 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. V detailu lze nalézt například adresu, webové stránky, otevírací hodiny fitness centra. V případě možné rezervace lze využít tlačítka se symbolem telefonu. Tlačítko se symbolem hvězdy umožní uživateli přidat lokalitu mezi oblíbené.

Obrázek 2 prezentuje obrazovku z aplikace, na které může uživatel nalézt 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.

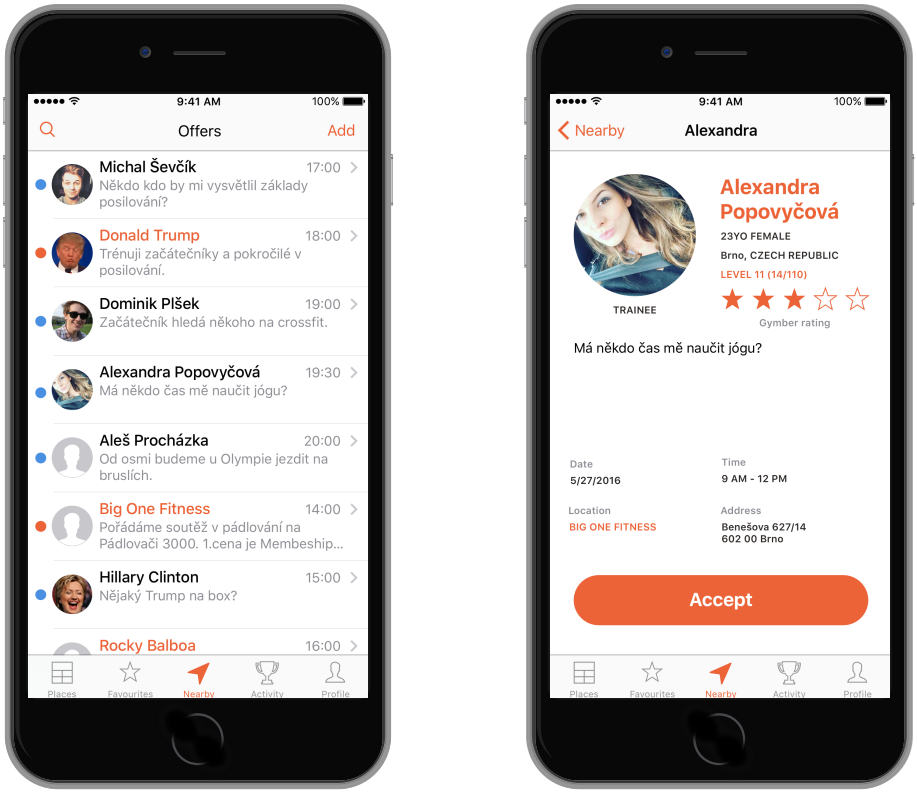
Pro placené nabídky byla zvolena oranžová barva. Oranžovou barvou je vybarveno i jméno zadávajícího, pokud se jedná o trenéra fitness aktivit. V opačném případě je u neplacených nabídek kruh vybarven modře a jméno zadávajícího má černou barvu. Nová nabídka je přidávána v horní liště pomocí tlačítka **Add**.

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 2 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[6], 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 nut-

[5] hash tag

[6] http://uber.com



Obrázek 2. Vizualizace obrazovky s výpisem a detailem nabídek ke cvičení.

nosti rolování obrazovky a slouží k zamezení chybného zvolení nevyhovující nabídky. Po potvrzení nabídky lze zadat krátkou zprávu a upřesnit čas setkání. Následná finanční kompenzace a další domluva je v kompetenci uživatelů.

2.2 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*[7], 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[8]. Při vytvoření nového uživatelského účtu se vygeneruje unikátní autentizační token, který je uložen společně 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.

2.3 Návrh serverové části služby

V kapitole 2.2 byly definovány nároky na návrh multiplatformního aplikačního rozhraní. Pokud klient, resp.

[7] JavaScript Object Notation

[8] pešek, žeton

mobilní aplikace, zašle validní zprávu dle protokolu z kapitoly 2.2, dojde k 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, která 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 a 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.

3. Implementace

Kapitola 2 seznámila čtenáře s návrhem aplikace z pohledu uživatelského rozhraní, návrhu aplikačního rozhraní a serverové části 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.

3.1 Implementace uživatelského rozhraní pro platformu iOS

Návrh představený v kapitole 2.1 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 pří-padně 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 však 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. Tyto scény jsou propojeny přechody, které slouží k navigaci v uživatelském rozhraní. Interface Builder s rozšiřující funkcionalitou, kterou nabyl 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 2, resp. obrázku 1. 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 těchto metod bylo docíleno návrhu, zobrazeného na obrázku 1.

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

3.2 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*[9], kde lze pomocí předem definované syntaxe navrhovat a zároveň dokumentovat jednotlivé metody.

[9] Blueprint

Apiary umožňuje náhled návrhu API s HTTP hlavičkou a obsahem zprávy z testovacího serveru. Každému projektu ve službě Apiary je přiřazena unikátní adresa, pomocí které lze testovat 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. Aplikace na obou platformách pak využívaly testovacího serveru Apiary k ladění protokolu a zpracování přenášených dat. Důvodem k použití Apiary může být i 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. Výsledný návrh na službě API byl poté implementován na vlastním serveru. Více informací k implementaci aplikačního rozhraní na serverové části služby lze nalézt v kapitole 3.3.

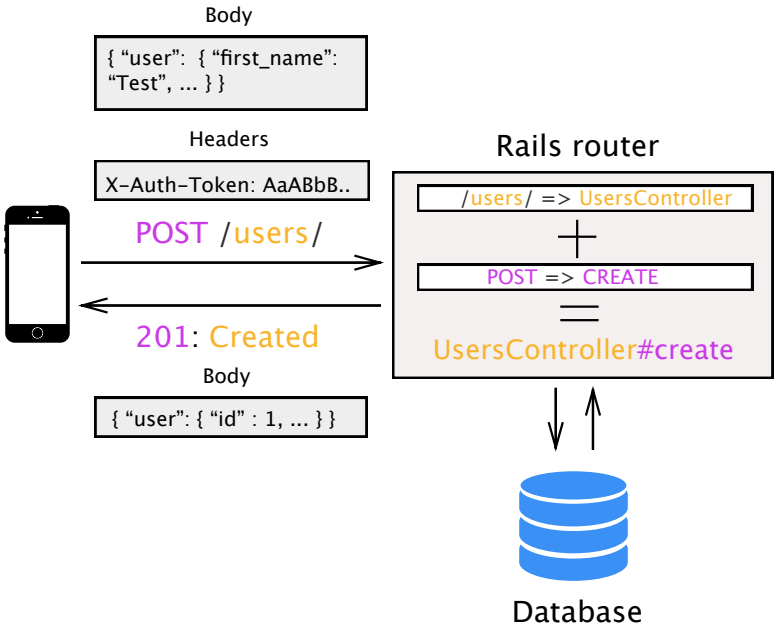
3.3 Aplikace frameworku Ruby on Rails na serverové části služby

Na serverové části je umístěn webový server, který komunikuje s klientem a databází. Pokud klient vyžaduje data z databáze, zašle dotaz na server, který se připojí k databázi a vrátí případná získaná data klientovi. Kapitoly 2.2 a 2.3 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*) [5]. Architektura MVC rozděluje jednotlivé zodpovědnosti práce s daty mezi zmíněné komponenty aplikace. Při implementaci aplikačního rozhraní, které bylo představeno v kapitole 2.2, je hlavní částí serverové služby model, řadič a databáze. Pohled u aplikačního rozhraní tvoří serializovaná data, která jsou posílána 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 těmito komponentami [6].

Na obrázku 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/*. Tato 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 dále 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. Při validaci dat a jejich následnému ukládání do databáze může dojít k zjištění nesrovnalostí v zaslaných datech. Chyba je interpretovaná klientovi a dojde ke zrušení procesu registrace uživatele. Aplikační rámec Ruby on Rails v případě výskytu chyby generuje čitelný popis a chybový kód, který je zasílán zpět klientovi.



Obrázek 3. Schéma REST služby běžící nad aplikačním rámcem Ruby on Rails.

4. Závěr

Článek představil postup návrhu uživatelského rozhraní, aplikačního rozhraní a serverové části mobilní platformy Gymber. Seznámil čtenáře s implementací uživatelského rozhraní na platformě iOS a využitím služby Apiary k definici API a popsal implementaci aplikačního rozhraní na aplikačním rámci Ruby on Rails.

Účelem je shrnutí postupu a zkušeností při vývoji mobilní aplikace na platformu iOS. Samotný vývoj mobilních aplikací je mladá část v oblasti IT, vzniklá s vydáním prvního modelu chytrého telefonu iPhone od značky Apple, který umožnil instalaci a stahování aplikací od vývojářů třetích stran pomocí internetového obchodu. Pro návrháře a vývojáře mobilní aplikace dává článek inspiraci k využití nástrojů a služeb při návrhu a implementaci vlastních mobilních aplikací.

Přínosem článku je sdílení know how vývoje mobilní aplikace, zdůraznění důležitosti vytváření prototypů aplikace před jejím samotným vývojem a definování pojmu minimal viable product. Ujasnění si, kdo je cílovým uživatelem, jaké jsou jeho problémy a co uživatel požaduje je důležitým aspektem při vývoji nejen mobilní aplikace. Po důsledném zjištění zmíněných parametrů může vývojář začít navrhovat aplikaci zaměřenou na potřeby uživatele.

Mobilní aplikace Gymber je programována s plánem reálného nasazení na obchody Apple Store a Google Play. Platforma jako celek je navrhována za účelem vyvíjení se z návrhů a požadavků od reálných uživatelů.

Poděkování

Rád bych poděkoval Prof. Ing. Adamu Heroutovi, Ph.D. za odborné vedení a motivaci při tvorbě článku.

Literatura

[1] Steve Schlafman. Uberification of the us service economy. blogpost (english), April 2014. http://schlaf.me/post/81679927670.

[2] Bob Dorf Steve Blank. *The Startup Owner's Manual: The Step–By–Step Guide for Building a Great Company*. K & S Ranch, 2012. ISBN: 0984999302.

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

[4] Inc. Apple. ios human interface guidelines. manual (english), March 2016. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/.

[5] David Heinemeier Hansson Sam Ruby, Dave Thomas. *Agile Web Development with Rails 4*. Pragmatic Bookshelf, 2011. ISBN: 1937785564.

[6] RailsGuides. Rails routing from the outside in. documentation (english). http://guides.rubyonrails.org/routing.html.