Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Služba na prezentaci menu

Tomáš Kratochvíl

Vedoucí práce: Ing. Josef Gattermayer

11. května 2015

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Josefu Gattermayerovi za cenné rady, připomínky a odborné vedení při zpracování této bakalářské práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Kratochvíl, Tomáš. *Služba na prezentaci menu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Bakalářská práce se zabývá tvorbou aplikace pro systém Android, která poskytuje uživatelům informace o restauracích v bezprostřední blízkosti. Unikátností aplikace je využití technologie Bluetooth Beacon k rozpoznání restaurací v okolí. Práce pojednává o analýze, návrhu, implementaci a testování aplikace a webového serveru se kterým aplikace komunikuje.

Klíčová slova Bluetooth Beacon, Android, aplikace, restaurace, návrh, implementace, testování

Abstract

Bachelor's thesis delas with creation of application for an Android platform, which provides informations about nearest restaurants for its users. The application uses the Bluetooth Beacon unique technology to recognize and identify restaurants around the user. This thesis refers about analysis, proposal, implementation and testing the aplication and web serve, which communicate with the application.

Keywords Bluetooth Beacon, Android, aplication, restaurant, layout, implementation, testing

Obsah

Úvod **1**

1 Analýza **3**
1.1 Bluetooth Beacon 3
1.2 Rešerše podobných aplikací 6
1.3 Analýza požadavků 8

2 Návrh **11**
2.1 Případy užití 11
2.2 Drátěný model 11
2.3 Databázový model 13
2.4 Komunikační protokol 16
2.5 Grafický návrh 17
2.6 Microsite 17

3 Implementace **19**
3.1 Implementace serveru 19
3.2 Implementace aplikace 25

4 Testování **31**
4.1 Testování serveru 31
4.2 Testování aplikace 32

5 Závěr **35**

Literatura **37**

A Doplňky **41**

B Seznam použitých zkratek **49**

C Obsah přiloženého CD **51**

Seznam obrázků

1.1 Beacon koncept 4
1.2 Estimote Beacony 5
1.3 Aplikace Meníčka 6
1.4 Aplikace Zomato 7
1.5 Nákupní centra 8

2.1 Diagram případů užití 12
2.2 Drátěný model aplikace - hlavní obrazovka 13
2.3 Databázový model 14

3.1 Struktura UI 25
3.2 Zobrazení notifikace při běhu aplikace na pozadí 29

A.1 Grafický návrh aplikace 42
A.2 Grafický návrh microsite 43
A.3 Drátěný model aplikace 44
A.4 Drátěný model serveru 45
A.5 Drátěný model serveru 46
A.6 Výsledná aplikace 47

Úvod

Pro identifikaci předmětů se v dnešní době používají známé technologie, jakými jsou třeba NFC štítky, QR kódy a další technologie. Pro určení polohy, zda se uživatel nachází na konkrétním místě, je hlavně využíván polohovací systém GPS. V posledních letech se s vyvinutím technologie Bluetooth Low Energy začíná rozšiřovat další zařízení sloužící k těmto účelům. Toto zařízení je označováno Bluetooth beacon. Tato práce popisuje tuto technologii a tvorbu aplikace komunikující s beacony.

Cílem práce bylo vytvořit aplikaci, která by umožnila zákazníkům zobrazit jídelní lístek a další informace o restauraci, v jejíž blízkosti se nachází. Aplikace zjistí blízkost restaurace podle beaconu, který se v ní nachází. Podobná aplikace se stejným zaměřením a s určováním polohy uživatele pomocí beaconů dosud neexistuje. Součástí práce bylo provést analýzu řešení, navrhnout aplikaci a webový server s kterým aplikace komunikuje, obě tyto komponenty implementovat a otestovat. Požadavkem na aplikaci byl také běh na pozadí a v případě vstoupení aplikace do blízkosti restaurace, upozornění uživatele v podobě notifikace. Webový server slouží jako administrační prostředí pro majitele restaurací, kteří zde mohou spravovat informace o restauracích a nahrávat aktuální jídelní lístky. Server dále poskytuje API pro komunikaci s aplikací, jenž poskytuje data nahraná uživateli administračního prostředí serveru.

Struktura práce je dělena do několika následujících kapitol

- Úvod - uvedení do problematiky
- Analýza - popis zařízení bluetooth beacon, rešerše podoných aplikací, analýza požadavků
- Návrh - popis návrhu aplikace a webového serveru - případy užití, drátěné modely, databázový model, komunikační protokol, grafický návrh a popis propagační microsite
- Implementace - popis implementace aplikace a webového serveru

- Testování - automatické testy webového serveru a aplikační testy aplikace
- Závěr - shrnutí práce

Kapitola 1

Analýza

1.1 Bluetooth Beacon

Jedním z cílů práce bylo uživatele upozornit na blízkost restaurace pomocí Bluetooth beaconů. Bluetooth beacon je malé komerčně vyráběné zařízení velikosti několika centimetrů. Beacony jsou vysílače, které vysílají skrze Bluetooth malé množství dat v opakujících se intervalech. Jejich využití je majoritně k identifikaci předmětů, u kterých jsou umístěny, nebo k geolokaci míst při rozmístění v prostoru. Pokud se zařízení, které je schopno přijímat signál z beaconu, dostane do jeho dosahu, beacon jej na sebe upozorní zasláním těchto dat informujícím o svých vlastnostech. Tyto data obsahují identifikační číslo beaconu a mohou obsahovat další údaje jako například aktuální teplotu, naklonění beaconu atd. Tyto dodatečná data závisí na konkrétním beaconu a jeho osazení snímacími senzory.Beacony nevyužívají klasickou technologii Bluetooth, ale její modifikaci Bluetooth Low Energy.[6]

1.1.1 Bluetooth Low Energy

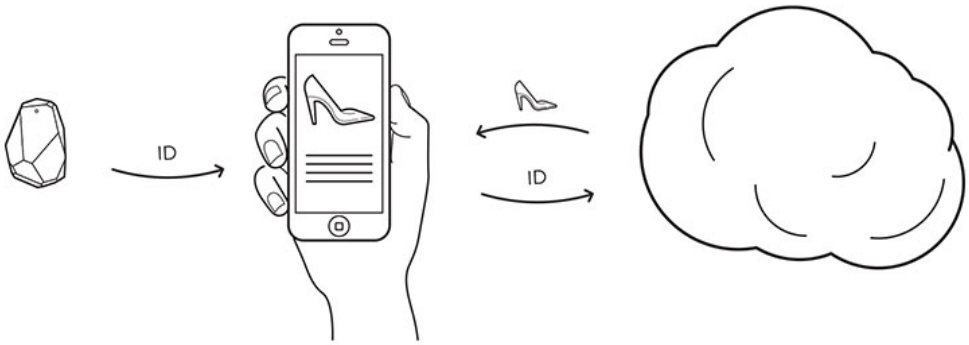
Bluetooth Low Energy (BLE) je nová technologie pro bezdrátový přenos dat známá také pod komerčním označením Bluetooth Smart. Technologie je vyvíjena společností Bluetooth Special Interest Group a je zaměřena speciálně pro nízkoenergetická zařízení, která jsou v chodu po delší časový úsek. Specifikace technologie umožňuje dva typy implementace. Prvním typem je takzvaný dual-mode. Zařízení s dual-mode specifikací obsahují klasický Bluetooth ovladač a ovladač pro BLE (počítače, mobilní telefony, tablety, ...). Druhou specifikací je single-mode. Zařízení se single-mode, jenž jsou osazeny i Bluetooh Beacony, obsahují pouze ovladač pro BLE, a tím šetří spotřebu energie zařízení.[5] Dalším příkladem může být pás pro měření srdečního tepu Polar H7[7], pomocí single-mode čipu vysílá přijímajícímu zařízení hodnotu tepu uživatele.

Původně byla tato technologie vyvinuta v roce 2006 finskou firmou Nokia pod názvem Wibree. Ještě než se technologie Wibree stala populární, Nokia ji

přesunula pod správu již zmiňované Bluetooth Special Interest Group, která kontroluje a standardizuje Bluetooth technologie. BLE je součástí standardu Bluetooth 4.0.

Velkou výhodou této technologie je její energetická nenáročnost. Zařízení, které ji využívají, jsou tak schopny fungovat přes rok pouze na knoflíkovou baterii. Na rozdíl od klasické technologie Bluetooth není koncipována na přenos většího množství dat jako jsou multimediální soubor. Je to kvůli menší datové propustnosti, která je zapříčiněna již zmiňovanou energetickou nenáročností. Na rozdíl od konkurenční technologie NFC, u které se počítá s těsným kontaktem přijímače a vysílače, lze pomocí technologie BLE propojit zařízení na vzdálenost přesahující 70m. [6]

1.1.2 Beacon koncept



Obrázek 1.1: Ukázka bluetooth beacon konceptu [8]

Pouhý Bluetooth beacon a zařízení přijímající jeho signál je pro většinu uživatelů nedostačující. Přijímací zařízení by po vstupu do vysílací zóny beaconu obdrželo jeho identifikační číslo a případně další naměřené údaje. Identifikační číslo je 128-bitová hodnota, a běžný uživatel není schopen z této hodnoty nic vyčíst a je tedy pro něj bezcenná. Z tohoto důvodu výrobci beaconů počítají s vytvořením konceptu, který se skládá ze tří částí:

1. Vysílač - Bluetooth beacon
2. Přijímač - zařízení s podporou Bluetooth 4.0 (mobilní telefon, tablet, počítač...)
3. Cloud - server komunikující s přijímačem

Princip tohoto konceptu je založen na komunikaci těchto tří výše uvedených zařízení. Bluetooth beacon je umístěn na určité pozici a v opakujících se intervalech vysílá své identifikační číslo. Přijímač přijímající signál z beaconu musí mít spuštěnou aplikaci, která komunikuje s beaconem a zároveň být připojený k internetu pro komunikaci s cloudem. V okamžiku, kdy se přijímač dostane do vysílací zóny beaconu, je mu doručeno ono identifikační číslo beaconu. Aplikace číslo zpracuje a pošle požadavek cloudu, který obratem přijímači pošle zpět další informace o konkrétním místu, kde se beacon nachází. Umístění beacnou zjistí cloud právě pomocí tohoto čísla. Přijímač poté informace od cloudu přijme, zformátuje do vhodné formy a zobrazí uživateli. Tento model je zobrazen na obrázku 1.1. [8]

1.1.3 Estimote Beacon

Na výrobu beaconů se specializují různé firmy. Některé firmy vyrábějí beacony pouze pro komunikaci se systémem iOS, jiné se snaží o multiplatformní podporu. Většina firem nabízí dva typy svých beaconů. Prvním typem bývá větší beacon, který je koncipován pro určení polohy uživatele a bývá osazen dalšími senzory a bateriemi s vyšší kapacitou slibující větší výdrž beaconu. Druhým typem je jeho menší verze napájená knoflíkovou bateriií, která slouží hlavně pro identifikaci předmětů. Zdůvodu své menší velikosti lze tyto beacony například nalepit na konkrétní předměty. Protože si tyto firmy navzájem konkurují, ceny jejich beaconů jsou velmi podobné.

Pro účely této práce byl vybrán beacon od polské firmy Estimote. Beacon obsahuje 32-bitový ARM procesor s 256 kB flash pamětí, akcelerometr a senzor teploty. Velkou výhodou je, že SDK od Estimote obsahuje jak verzi pro iOS, tak verzi pro zařízení s OS Android, a splňuje tak podmínku multiplatformní podpory. Estimote dokonce poskytuje aplikaci pro Android a iOS, která dokáže simulovat jejich beacon, což je dobře využitelné při vývoji a testování aplikací. Cena beaconů je nastavena na $99 za 3 kusy. Při objednání většího množství se na objednávku vztahuje množstevní sleva.[9]

Obrázek 1.2: Estimote Beacony [10]

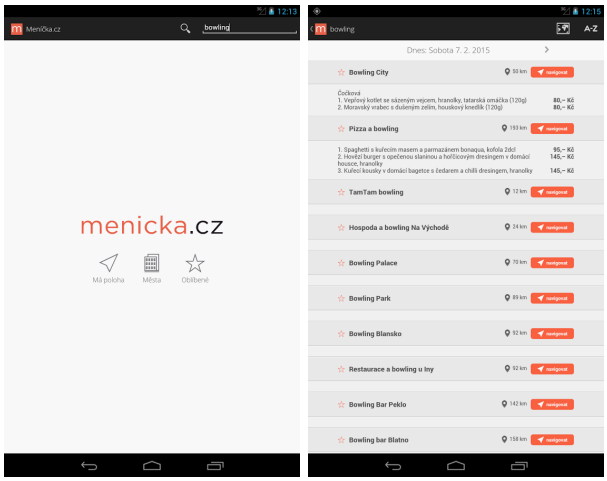
1.2 Rešerše podobných aplikací

Z důvodů neexistence aplikace, která by uživateli zobrazovala jídelní lístek restaurace a zároveň by využívala technologie Bluetooth beconů, je analýza podobných aplikací zaměřena na aplikace, které si za cíl kladou zobrazování jídelních lístků, ale nevyužívají zmíněnou technologii.

1.2.1 Meníčka

Meníčka je česká aplikace od Jana Veleckého, která zobrazuje denní nabídky restaurací. Aplikace nevyžaduje přihlášení, a tak stačí aplikaci zapnout a zobrazit si příslušnou restauraci. V úvodním menu má uživatel na výběr ze tří možností. První možností je „Má poloha“, která pomocí GPS zjistí polohu uživatele a podle toho mu nabídne restaurace v okolí. Drhou možností je „Města“, jenž je využitelná při nedostupnosti GPS a umožňuje manuální vyhledání restaurace podle výběru města. Třetí možností je „Oblíbené“. Uživatel si může restaurace přidávat do oblíbených a skrze tuto sekci je následně zobrazit. Aplikace nabízí uživateli další funkcionality v podobě například využití navigace skrze API Google Maps pro navigaci do restaurace.

Při uživatelském testování aplikace byla zjištěna nepříjemnost při manuálním hledání restaurací. Pří výběru města dochází k dlouhému načítání a aplikace nedává uživateli nijak najevo, že dochází k načítání příslušné restaurace. Uživatel tak neví, zda došlo k výběru města.



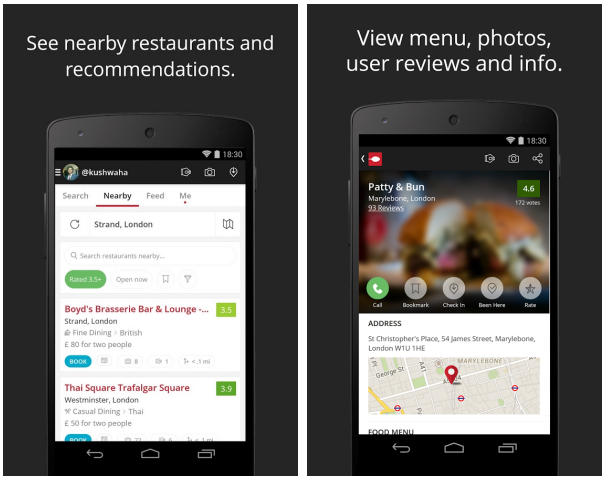
Obrázek 1.3: Aplikace Meníčka [3]

1.2.2 Zomato

Zomato je populární globální indická služba, jenž si celosvětově stáhlo z obchodu Google Play přes milion uživatelů.[4] Aplikace nabízí komplexní infor-

mace o restauracích v okolí. Aplikace vytváří malou sociální síť, a tak je nutno se v úvodu do aplikace přihlásit. Tato sociální síť nabízí uživateli hodnocení podniků, přidávání přátel a přidávání fotografií restauracím. Stejně jako aplikace Meníčka, Zomato využíva k určení polohy uživatele GPS.

Zomato dříve nahrazovala česká populární aplikace LunchTime.cz, jenž byla v roce 2014 Zomatem odkoupena. Aplikace je opravdu komplexní a odpovídá standartu globální aplikace. Z toho důvodu jsou na aplikaci stěží nalezitelné větší nedostatky.



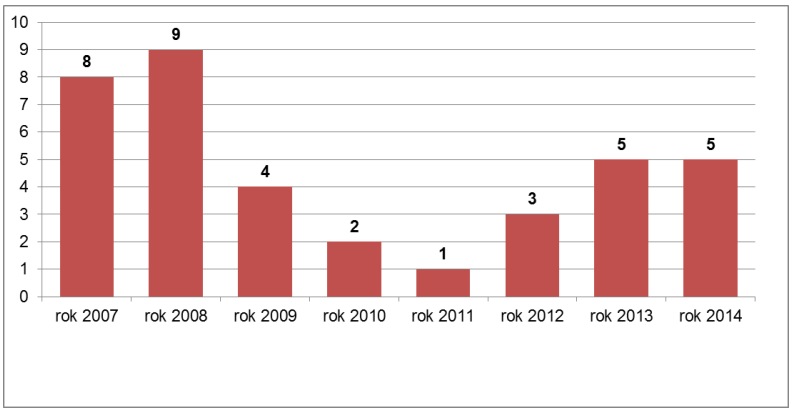
Obrázek 1.4: Aplikace Zomato [4]

1.2.3 Srovnání GPS a Bluetooh Beacon technologie

Obě předchozí aplikace, jak už bylo řečeno, používají k určení polohy GPS. Výhodou je, že aplikace jsou díky GPS schopny určit přesnou polohu uživatele s přesností na jednotky metrů. Takováto aplikace je schopna uživateli zobrazit nejbližší restaurace v okolí a sdělit mu tak aktuální vzdálenost. Nevýhodou je potřeba zapnuté GPS, což má za následek rychlé vybíjení baterie zařízení, na němž je aplikace spuštěna. Zásadní problém nastává ve chvíli, kdy uživatel je mimo dosah signálu GPS. K přerušení signálu dochází při návštěvě většiny interiérů.

Mnoho restaurací a kaváren se nachází v zastřešených pasážích a v nákupních centrech. Na konci roku 2014 bylo v České republice podle výzkumů společnosti INCOMA GfK rozmístěno 497 moderních multisortimentních obchodních komplexů (nákupních center, retail parků, outlet center, hypermarketů s malými pasážemi a obchodních domů). [2] Další výzkum stejné společnosti uvádí, že od roku 2007 jsou v České republice průměrně vybudovány přes 4 nákupní centra ročně. [1] Při úvaze průměrně tří restaurací a kaváren na nákupní centrum se dostáváme na počet 1497 restaurací a kaváren v ČR,

které nejsou pokryty signálem GPS, a tedy uváděné aplikace nemohou určit, zda se uživatel nachází v jejich bezprostřední blízkosti.



Obrázek 1.5: Vývoj počtu nově zprovozněných nákupních center od roku 2007[1]

Při rozmístění Bluetooh Beaconů v blízkosti restaurací je aplikace schopna určit zda se uživatel nachází v její blízkosti, ať už se jedná o interiér, nebo exteriér. Nutností je zapnuté Bluetooth na mobilním zařízení s aplikací. Oproti GPS velkou předností Bluetooth low enregy je její nízká energetická náročnost. Hlavní nevýhodou Bluetooth Beconů je neschopnost určit polohu, pokud se uživatel zrovna nenachází u konkrétní restaurace.

1.3 Analýza požadavků

Analýza požadavků je nezbytná pro definování konkrétních požadavků na aplikaci a webový server. Návrh a vývoj tohoto softwaru vychází z těchto požadavků. Analýza je rozdělena na dvě skupiny, první skupinou jsou funkční požadavky, které definují potřeby uživatele na software, druhou skupinou jsou nefunkční požadavky, které definují další potřebné vlastnosti aplikace a webového serveru, jako je bezpečnost, použité technologie atd. Požadavky jsou dále rozděleny na požadavky na aplikaci a na požadavky na webový server.

1.3.1 Funkční požadavky

- Aplikace
  - zobrazení detailu resturace u niž se uživatel nachází
  - zobrazení jídelních lístků resturace
  - zobrazení jídel v konkrétním jídelním lístku
  - zobrazení probíhající akce restaurace

  - push-up notifikace upozorňující na přítomnost restaurace při přiblížení k restauraci
  - push-up notfikace zobrazující se při běhu aplikace na pozadí
- Webový server
  - přihlášení uživatele
  - zobrazení restaurací uživatele
  - editace a mazání restaurací
  - editace a mazání jídelních lístků
  - editace a mazání sekcí
  - editace a mazání jídel

1.3.2 Nefunkční požadavky

- Aplikace
  - platforma OS Android
  - využití Bluetooth beaconu
  - asynchronní stahování dat
  - využití služeb OS Android pro zobrazování notifikací
- Aplikace
  - zabezpečení hesel uživatelů pomocí hašovací funkce

Kapitola 2

Návrh

2.1 Případy užití

Diagram případů užití demonstruje využití administrační části webového serveru. Zobrazuje, kdo a jak bude systém využívat, slouží pro ověření požadavků z analýzy požadavků a lze ho využít pro tvorbu testovacích scénářů.

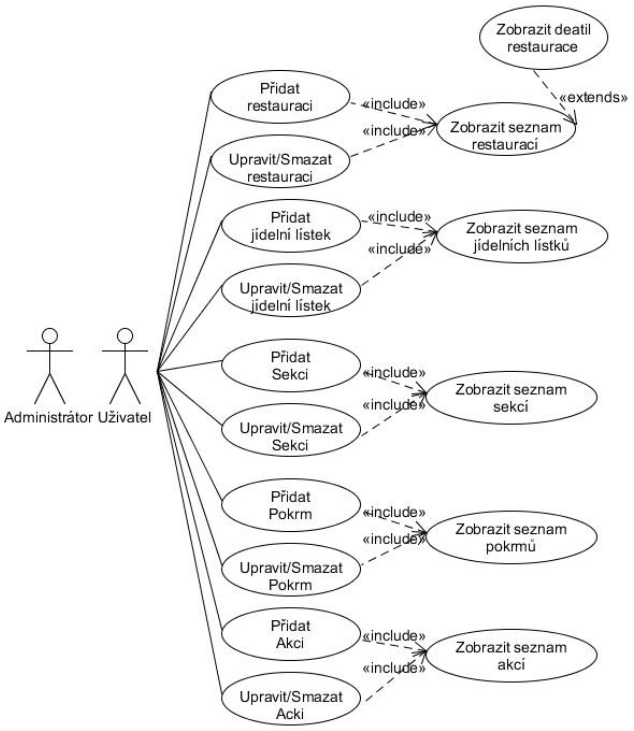
Osoby přistupující do administrační části serveru se řadí do dvou skupin. První skupinou jsou uživatelé a druhou správci. Jediný rozdíl těchto dvou skupin je právo pro přístup ke konkrétním restauracím. Uživatelům se zobrazují jen jimi vytvořené restaurace a správcům se zobrazují všechny restaurace.

Po přihlášení do administrační části je uživateli zobrazen seznam restaurací s tlačítkem pro přidání nové restaurace. Uživatel může konkrétní restaurace zobrazovat, editovat a mazat. Zároveň si může prohlédnout konkrétní jídelní lístky a slevové akce dané restaurace. Jídelní lístky obsahují sekce a ty obsahují konkrétní pokrmy. Všechny tyto vyjmenované entity (akce, jídelní lístky, sekce, pokrmy) lze v administračním prostředí přidávat, upravovat a mazat.

2.2 Drátěný model

Pro lepší představu o výsledné podobě aplikace a webového serveru byl vytvořen drátěný model. Tento model definuje jednotlivé obrazovky a popisuje umístění prvků na nich. Model volně vyplývá z funkčních požadavků definovaných v kapitole analýza.

Drátěný model je vytvářen pro hrubý návrh aplikace. Při zakázkové výrobě je nejdříve vytvořen právě tento model, který je posléze předložen zákazníkovi. Zákazník tak získá představu o výsledné aplikaci a může poté lépe dodefinovat obrazovky aplikace podle svých představ. Při použití specializovaných programů je předělání modelu časově nenáročné.



Obrázek 2.1: Diagram případů užití

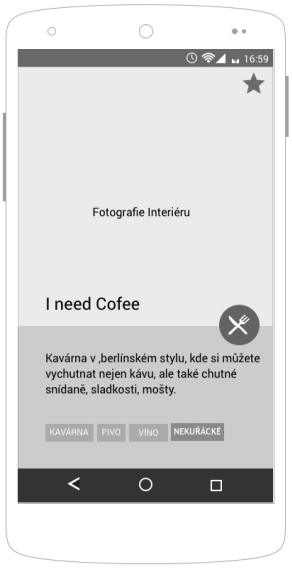
2.2.1 UXPin

Pro tvorbu modelu byl použit nástroj UXPin. UXPin je online aplikace, jenž nabízí mnoho předdefinovaných ovládacích prvků pro tvorbu drátěného modelu. Prvky jsou děleny do kategorií podle platforem, a tak lze vybírat z nabídky pro konkrétní platformu. UXPin nabízí tvorbu modelu jak pro web, tak i pro aplikace běžící na iOS, Android a Windows Phone.[11] Z tohoto důvodu byl UXPin vybrán, jelikož požadavkem na práci byl model pro web i pro mobilní aplikaci.

2.2.2 Drátěný model aplikace

Na obrázku A.3 je vidět část modelu mobilní aplikace. První obrazovka zobrazuje úvodní obrazovku po uživatelském kliknutí na push-up notifikaci. Jsou zde základní informace o restauraci, štítky a fotografie interiéru. V pravém horním rohu aplikace se nachází tlačítko v podobě hvězdice, které zobrazuje vyskakovací okno akce, jenž je zobrazeno na třetí obrazovce. Dále je zde tlačítko s ikonou příboru, které zobrazuje jídelní lístek a po kliknutí uživatele ho přesune na obrazovku s výběrem sekce. Po výběru sekce je uživatel přesunut na druhou obrazovku, kde již vidí výsledné pokrmy vybrané sekce. Čtvrtá

obrazovka znázorňuje obrazovku po zapnutí aplikace v případě, že se uživatel nenachází v dosahu žádného Bluetooth beaconu.



Obrázek 2.2: Drátěný model aplikace - hlavní obrazovka

2.2.3 Drátěný model serveru

Při tvorbě tohoto modelu byly vytvořeny z důvodu úspory času jen čtyři obrazovky. Administrační prostředí restaurací, sekcí, jídelních lístků a jídel má stejné rozmístění prvků na obrazovce. Wireframe zachycuje detail restaurace, přihlašovací obrazovku, seznam pokrm a přidání nového pokrmu.

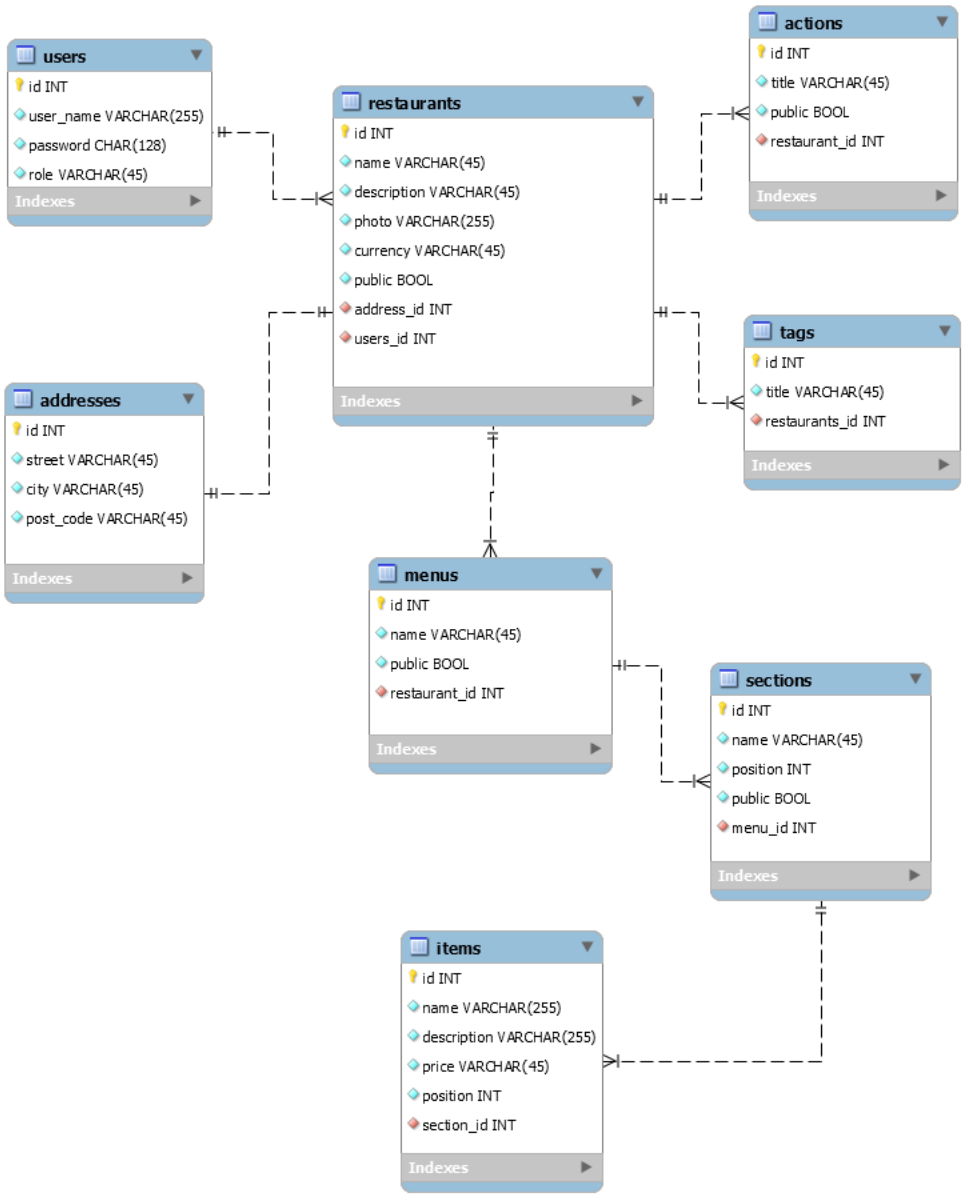
2.3 Databázový model

Databázový model složí pro definování relačního modelu databáze běžící na webovém serveru. Model popisuje jednotlivé tabulky, příslušné sloupce a jejich datové typy. Zobrazeny jsou rovněž primární klíče tabulek a vztahy mezi tabulkami.

2.3.1 MySQL Workbench

Pro návrh databáze byl použit nástroj MySQL Workbench, který umožňuje navrhnout pomocí svého grafického rozhraní relační model databáze. MySQL Workbench je přímo určen pro databázi MySQL, a je tak s ní plně kompatibilní. Nástroj umožňuje uživateli i export výsledného SQL kódu.[12]

Následující databázový model zobrazuje schéma databáze použité u webového serveru.



Obrázek 2.3: Databázový model

2.3.2 restaurants, address

Tabulka restaurants uchovává údaje o restauracích. Pro lepší názornost jsou informace o adrese vloženy do nové tabulky address a spojeny vztahem 1:1.

- name (restaurants) - jméno restaurace
- description (restaurants) - popis restaurace
- beacon (restaurants) - identifikační číslo beaconu
- photo (restaurants) - cesta k fotografii interiéru restaurace
- currency (restaurants) - udává měnu jídelních lístků (Kč, $, ...)
- public (restaurants) - udává zda se restaurace bude zobrazovat v aplikaci
- street (address) - ulice
- city (address) - město
- post_code (address) - poštovní směrovací číslo

2.3.3 users

V tabulce users jsou uloženy informace o uživatelích webového serveru, jenž slouží pro jejich autentizaci. Tabulka je spojena vztahem 1:N s tabulkou restaurants, což umožňuje uživateli si v systému vytvořit více než jednu restauraci.

- user_name - uživatelské jméno
- password - heslo
- role - udává uživatelské oprávnění jenž nabývá dvou stavů:
  - admin - uživateli se zobrazí všechny restaurace
  - user - uživateli se zobrazí jen restaurace, které sám vytvořil

2.3.4 tags

Tabulka uchovává informace o štítcích restaurace. Štítky slouží k rychlému informování zákazníka o vlastnostech restaurace. (nekuřácká, vegetariánská, italská kuchyně, ...)

- title - nápis na štítku

2.3.5 actions

Informace o právě probíhající akci restaurace.

- title - nadpis akce
- public - značí, jestli se akce zobrazí v aplikaci

2.3.6 menus, sections, items

Tyto tabulky uchovávají informace o daném menu restaurace. Tabulka menu definuje konkrétní jídelní lístky. V aplikaci se zobrazuje pouze jeden jídelní lístek, k čemuž slouží řádek public, jenž může být jen u jednoho menu na hodnotě TRUE. Tabulka sections uchovává údaje o kategoriích menu a tabulka items o konkrétních pokrmech, jenž jsou obsahem sekcí.

- name (menus, sections, items) - název menu, sekce, pokrmu
- public (menus, sections) - značí, jestli se menu/sekce zobrazí v aplikaci
- position (sections, items) - pozice sekce v jídelním lístku a pozice pokrmu v sekci
- price (items) - cena pokrmu

2.4 Komunikační protokol

Jelikož Bluetooth beacon posílá aplikaci jen své identifikační číslo, je třeba, aby aplikace stahovala další data z webového serveru. Pro komunikaci mezi aplikací a webovým serverem byl zvolen protokol REST. Protokol byl vybrán z důvodu jeho konkrétního zaměření na webové API a kvůli jeho jednoduchému rozhraní. REST sice implementuje čtyři metody pro přístup ke zdrojům (GET, POST, PUT, DELETE), ale v této práci byla využita jen metoda GET. Ostatní metody by bylo potřeba implementovat v případě, že by aplikace měla měnit, vkládat nebo mazat data. Aplikace vyžaduje tři dotazy: dotaz na data o restauraci u niž se nachází, akci této restaurace a jídelní lístek. URL API byla navrhnuta následovně:

```
/api/restaurant/8492e75f-4fd6-469d-b132-043fe94921d8
```

První část /api značí, že se jedná o API webového serveru. Druhá část určuje, jaký ze tří dotazů je dotazován. Při použití řetězce /restaurant jsou vráceny informace o restauraci. Řetězec /action vrací informace o akci a řetězec /menu vrací jídelní lístek. Poslední část je UUID Beaconu. Pří správném tvaru dotazu a existujícím UUID Beaconu, který je uložen v databázi serveru,

jsou vráceny data ve formátu JSON a stavový kód 200 značící úspěšné odeslání dat. Pokud je zadaná špatná URL nebo neexistující UUID, server vrací stavový kód 404.

2.4.1 Apiari

Pro návrh a testování komunikace byla použita služba Apiary. Jedná se o mock server, který odpovídá na dotazy pomocí uživatelsky předdefinovaných odpovědí. Lze tedy provádět testování komunikace aplikace proti tomuto mock serveru bez potřeby funkčnosti vlastního serveru. Apiary udržuje dokumentaci a uchovává API podle návrhu uživatele. Umožněno je také sdílení návrhu mezi členy týmu. Služba také obsahuje komponentu „Traffic insector“, která zaznamenává veškerou komunikaci s mock serverem.[13]

2.5 Grafický návrh

Grafický návrh aplikace vychází z drátěného modelu. Při návrhu byl kladen důraz na Material Design. Material Design je souhrn pravidel vytvořených společností Google, jenž definuje výsledný vzhled aplikace. Google se tak snaží o sjednocení vzhledu aplikací na Google Play pro lepší uživatelskou orientaci. Tyto pravidla například definují kolekce barev, ovládací prvky, font, rozmístění prvků a další vlastnosti uživatelského rozhraní aplikace.

Ke stránkám administračního rozhraní serveru nemají přístup běžní uživatelé používající aplikaci, ale pouze oprávněné osoby, které spravují informace o restauracích. Z tohoto důvodu nebyl vytvářen grafický návrh, ale byla použita základní šablona frameworku Bootstrap.

2.6 Microsite

Součástí práce bylo také vytvoření microsite propagující produkt. Microsite obsahuje pouze jednu HTML stránku, a z tohoto důvodu nebyl vytvářen drátěný model, ale pouze grafický návrh stránky, viz obrázek A.2. Kvůli správnému vykreslení na různých rozlišeních monitorů a displejů byl stejně jako pro administrační rozhraní webového serveru použit framework Bootstrap, jenž se snaží implicitně vytvářet responzivní weby.

Kapitola 3

Implementace

3.1 Implementace serveru

Webový server byl implementován v jazyce PHP. Administrační vrstva webového servu je zabezpečené prostředí s množstvím formulářů a validačních pravidel, jenž využívá relační databázi k ukládání dat. Pro toto se přímo nabízelo využití MVC frameworku. V moderních MVC frameworcích je již připraveno rozhraní pro všechny tyto operace. Framework taktéž webové stránky automaticky zabezpečuje proti útokům, a tak zvyšuje produktivitu programátora, který na tyto věci nemusí myslet a může se soustředit na psaní vlastní aplikace.

3.1.1 MVC architektura

Model-view-controller (MVC) architektura rozděluje aplikaci na tři oddělené vrstvy, tak aby šlo upravovat každou vrstvu zvlášť a dopad úprav měl na ostatní co nejmenší vliv. Zároveň jsou tak odděleny části kódu od sebe a kód se stává přehlednějším. Tyto tři části jsou Model, Controller a View. Model reprezentuje data aplikace. Controller zpracovává události vyvolané uživatelem a zastupuje funkční logiku aplikace. View je vrstva, která zastupuje uživatelské rozhraní.

3.1.2 Použité technologie

3.1.2.1 Git

Pro správu verzí práce byl použit systém Git. Program Git se hodí hlavně pro týmový vývoj aplikací, protože napomáhá vývojářům se spojováním jednotlivých částí programu. Přestože tato práce byla vytvořena jedním člověkem, systém Git byl použit, a to hlavně z důvodu zaznamenávání změn v jednotlivých verzích práce. V průběhu vývoje se lze tedy jednoduše vracet do stavu aplikace v konkrétní verzi. Vývoj je rozdělen do logických celků podle verzí, které lze komentovat, a poté zpětně dohledávat. Vývojář má přehled o kódu a

může jej spojovat s konkrétní funkcionalitou aplikace. Git informace o verzích systému ukládá do takzvaných repositářů.

Pro uložení a správu repositáře byl použit systém Bitbucket, který nabízí zdarma pro open-sorce projekty cloudové uložiště repositářů. V důsledku kopírování práce na vzdálený server, tak dochází i současně k zálohování práce.

3.1.2.2 Nette framework

Pro implementaci byl použit Nette framework, jenž je postaven na architektuře MVC. Nette je český objektový PHP framework vytvořený Davidem Grudlem.

3.1.2.3 Bootstrap

Boostrap je framework pro HTML, CSS a JS. Bootstrap obsahuje HTML a CSS šablony sloužící pro vytváření a stylování prvků na stránce. Obsahem frameworku jsou i JavaScriptové komponenty, které lze vložit na webové stránky. Velkou předností Bootstrapu je kompatibilnost ve všech prohlížečích a schopnost přizpůsobit se starším prohlížečům. Další silnou stránkou je podpora responzivního designu, což zajišťuje, že se stránka dynamicky formátuje s ohledem na použité rozlišení zobrazujícího zařízení.

3.1.2.4 jQuery

Při implementaci uživatelského rozhraní serveru byla použita knihovna pro JavaSCript jQuery. jQuery obsahuje mnoho funkcí pro práci s HTML elementy a CSS styly. Tato knihovna taktéž obsahuje propracovaný systém selektorů, pomocí nichž je možné vybrat HTML element pomocí libovolných CSS selektorů.

3.1.3 Model

Model je v MVC architektuře struktura, která se stará o data aplikace. Soubory obsahující model jsou umístěny v adresáři app\model.

3.1.3.1 MySQL

Jako databázový systém byla použita populární databáze MySQL. MySQL využívá licneci open-source, a proto je její použití zcela zdarma. Pro svou velkou popularitu existuje mnoho knihoven a aplikací pro práci s touto databází. Příkladem je aplikace phpMyAdmin, která obsahuje grafické rozhraní pro správu MySQL databáze. Nevýhodou této databáze může být její nedostatečný výkon při tvorbě enterprise aplikací, které kladou velké nároky právě na výkon. Tato nevýhoda je ovšem v kontextu této práce irelevantní, a proto byla vybrána právě tato databáze.

3.1.3.2 PDO

Databázové třídy v Nette využívají PHP nadstavbu PDO. PDO je rozhraní, které bylo přidáno do PHP ve verzi 5.1. PDO umožňuje přistupovat k rozdílným databázím jednotným přístupem. Vytváří abstraktní vrstvu, ve které definuje metody pro práci s databází. Toto umožňuje vývojářům vytvořit jednotný kód, který je použitelný u velkého množství databází. Změna databáze se provede pouhým přepisem řetězce s informacemi o připojení.[14]

```
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

Konstruktor třídy PDO přijímá v prvním parametru právě zmiňovaný řetězec s informací o připojení.

3.1.3.3 Databáze v Nette

V adresáři app\model\tables je pro každou tabulku v databázi vytvořena třída v které jsou poté definovány jednotlivé metody pro manipulaci s danou tabulkou. Všechny tyto třídy dědí od třídy Table, která definuje společné chování. Je zde například metoda *findAll*, která v návratové hodnotě vrací všechny záznamy z tabulky nebo metoda *findByID*, která v parametru přijímá identifikační číslo (ID) a poté vrací jediný řádek s konkretním ID. Velkou výhodou databázového prostředí v Nette je automatické spojování tabulek. Odpadá tak potřeba psaní dlouhých JOINů. Pro výběr dat z více tabulek stačí pouze vydefinovat konkrétní sloupce a Nette pomocí cizích klíčů zjistí relaci mezi tabulkami a samo vytvoří požadovaný dotaz.

Nette obsahuje databázovou knihovnu *Nette\Dattabase*.[15] Knihovna obsahuje třídu *Nette\Database\Connection*, která tvoří obálku nad rozhraním PDO. Pro dotazování do databáze slouží třída *Nette\Database\Table\Selection*, která vrací pole instancí *ActiveRow*, jež reprezentuje jeden řádek výsledku dotazu. K sloupcům konkrétního řádku pak lze přistupovat pomocí klíče pole, nebo lze využít přístup přes členskou proměnnou.

3.1.3.4 Přihlašování a oprávnění uživatelů

Přihlašování uživatelů v Nette zajišťuje třída *Nette\Security\User*.[16] Autentizace uživatelů, tedy proces, který určí zda je člověk oprávněn ke vstupu, je spuštěna zavoláním metody *login* této třídy, která volá autentifikátor. Autentifikátor je implementací rozhraní třídy frameworku Nette *IAuthenticator*, jenž má jedinou metodu *authenticate* a je umístěn v adresáři *app\model\auth*. Metoda vrací skrze návratovou hodnotu instanci třídy *Nette\Security\IIdentity*, pokud uživatele dohledá v databázi, která obsahuje data o uživately z databáze. V opačném případě je vyvolána výjimka *Nette\Security\AuthenticationException* a v příslušném presenteru je tato výjimka odchycena a vypsána uživateli chybová hláška informující o zadání chybných přihlašovacích údajích. Třída

Nette\Security\IIdentity dále obsahuje metodu *getRoles*, která vrací role uživatele. V této práci byly použity dvě role. Role admin, jenž uživateli s tímto oprávněním vypíše všechny restaurace z databáze a role user, která vypíše uživateli jen jeho vlastní restaurace. Pro zvýšení bezpečnosti je při ukládání hesel do databáze použita hašovací funkce SHA-512 a do databáze se ukládá pouze haš hesla o délce 128 znaků.

3.1.4 Controller

Controller je dělící vrstva komunikující s modelem a s view. Vrstva zachytává akce od uživatele a poté tyto akce zpracovává a případně upravuje model. V Nette tuto vrstvu zastupují třídy s názvem Presenter, které jsou k nalezení v adresáři app\presenters. Stejně jako u modelu je zde vytvořena třída, od které ostatní presetery dědí s názvem *BasePresenter*. Jednotlivé presentery rozdělují aplikaci na logické celky. Jeden presenter může tedy obsluhovat více šablon a komunikovat s více částmi modelu. Příkladem je například *HomepagePresenter* obsahující hned tři šablony (výpis restaurací, detail restaurace a editaci restaurace).

3.1.5 Formuláře

Hlavní část administračního rozhraní tvoří formuláře. Vyskytují se všude, kde je potřeba vkládat, nebo upravovat nějaké informace, které jsou následně ukládány do databáze. Jsou umístěny na stránkách s úpravou a vkládáním nových restaurací, akcí, menu, sekcí a jídel. Pro obsluhu formulářů implementuje Nette třídu *Nette\Application\UI\Form*.[17] Tato třída obsahuje metody pro vytváření, validaci a nastavení formuláře. Nette klade velký důraz na bezpečnost a své formuláře automaticky zabezpečuje proti útokům Cross Site Scripting (XSS) a Cross-Site Request Forgery (CSRF). Nette také disponuje dvojitou validací ověřující nejdříve formulář na straně klienta s využitím JavaScripu a poté na straně serveru.

Následující kód demonstruje vytvoření formuláře pro editaci položky v menu. Formulářová metoda *setRequired* přidává pravidlo pro nutnost vyplnění pole, druhá metoda *setDefaultValue* nastavuje výchozí hodnotu v poli formuláře. Další metodou je metoda *onSuccess* definující, jaká metoda se bude volat po odeslání formuláře.

```
public function createComponentAddItemForm() {

    $form = new UI\Form;
    $form->addText('name', 'Název:')
          ->setRequired('Zadejte prosím název')
          ->setDefaultValue($this->item->name);
    $form->addTextArea('description', 'Popis:')
```

```
            ->setRequired('Zadejte prosím popis');
        $form->addText('price', 'Cena:')
            ->setRequired('Zadejte prosím cenu');
        $form->addSubmit('submit', 'Uložit');
        $form->onSuccess[] = array($this, 'addItemFormSucceeded');
        return $form;
    }
```

3.1.6 Továrnička

Formuláře v Nette se vytvářejí jako komponenta pomocí takzvané továrničky. Továrnička je vytvořena metodou *createComponent<Name>*, kde <Name> je název vytvářené komponenty. Metoda se nikde v kódu nevolá přímo, ale je zavolána až v případě její první potřeby. Tento jev se dá například využít při použití AJAXu, kdy se přenáší jen část stránky. Pro vykreslení komponentu v Latte šabloně stačí napsat příkaz *{control <Name>}*.

3.1.7 Routování URL

Routování je mapování URL na konkrétní akci presenteru. Používá se zejména pro lépe zapamatovatelnější URL a pro SEO. Pro administrační rozhraní toto není zapotřebí řešit, ale pro API, které musí být ve tvaru \api\akce\UUID_Beaconu, je třeba přesný tvar URL. V Nette routování představuje samostatnou vrstvu, jenž může být v průběhu vývoje změněna. Znamená to, že pokud programátor není s výsledným URL spokojen, může jej kdykoliv měnit. S využitím routování se nemusí v šabloně psát odkazy pomocí cesty, ale stačí odkaz definovat pomocí Latte makra v následujícím tvaru:

```
<a n:href="Section:default $restaurant->id,$item->id">
```

Tento kód odkazuje na akci default presenteru *Section* které předává dva parametry. Ekvivalentem je poté použití v presenteru:

```
$this->redirect("Section:default",$restaurant->id,$item->id);
```

V Nette se k tomu využívá třída *Nette\Application\Routers\RouteList*, jenž je kolekcí tříd *Nette\Application\Routers\Route*. Zavolním konstruktoru třídy Route lze definovat tvar URL.

```
new Route('<presenter>/<action>[/<id>]', 'Homepage:default');
```

Prvním parametrem je tvar URL a druhým parametrem je název výchozího presenteru. Při požadavku na server s konkrétní URL Nette iteruje touto kolekcí a při první shodě zavolá příslušný presenter. Při nenalezení požadované routy je vyvolána výjimka *BadRequestException*, a uživateli se zobrazí chybová stránka 404.

3.1.8 View

View převádí data do podoby zobrazitelné uživatelem. Vrstva view je v Nette řešena pomocí šablon nazývaných Latte které se nacházejí v adresáři app\templates. Právě šablonovací systém je velkou předností frameworku.

3.1.9 Latte makra

Do šablony se dají vkládat předdefinovaná makra, která zpřehledňují výsledný kód šablony a nahrazují nativní PHP kód vkládaný do šablony.[18]

```
//Kód napsaný čistým php
<?php if ($items): ?>
    <?php $counter = 1 ?>
    <ul>
    <?php foreach ($items as $item): ?>
        <li id="item-<?php echo $counter++ ?>"><?php
        echo htmlSpecialChars(mb_convert_case($item, MB_CASE_TITLE)) ?>
        </li>
    <?php endforeach ?>
    </ul>
<?php endif?>
```

```
//Kód s využitím maker šablony Latte
<ul n:if="$items">
{foreach $items as $item}
    <li id="item-{$iterator->counter}">{$item|capitalize}</li>
{/foreach}
</ul>
```

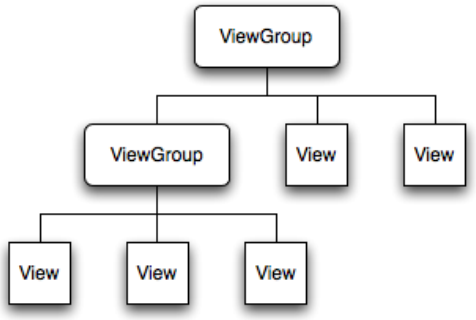
Kód výše demonstruje použití maker. Makra definujeme pomocí složených závorek. Každé párové makro jako například {if}{/if} se dá zapsat pomocí takzvaného n:makro, které se přímo vkládá do HTML elementu jako jeho atribut. Mimo zpřehlednění kódu makra escapují výpis proměnných, což zabraňuje výše zmiňovanému Cross Site Scripting útoku, a tak není třeba každý výpis proměnné vkládat jako argument PHP funkce *htmlSpecialChars*. Nette obsahuje mnoho předdefinovaných maker, která lze využívat, nebo si lze vytvořit vlastní makro za pomoci Latte API.

V této práci byly všechny výskyty PHP v šablonách nahrazeny makry, a to bez ztráty rychlosti. Nette totiž všechny makra překládá do nativního PHP kódu a ukládá na disk.

3.2 Implementace aplikace

3.2.1 Uživatelské rozhraní

Základním prvkem pro tvorbu uživatelského rozhraní je třída View. Z View vychází velké množství podtříd nazívající se Widgety. Widgety jsou interaktivní prvky uživatelského rozhraní jako např. textové pole, tlačítko, popisek, seznam atd. Dalším prvkem uživatelského rozhraní je abstraktní třída ViewGroup, jejímž obsahem mohou být třídy View a další třídy ViewGroup, jako je naznačeno na obrázku 3.1. Z ViewGroup dědí třídy nazývané rozvržení.



Obrázek 3.1: Struktura UI[19]

Android SDK obsahuje několik typů rozvržení. Každé z nich definuje jiný typ pozicování tříd View v uživatelském rozhraní. V této práci jsou požity dva typy rozvržení pro docílení správného uspořádání Widgetů. Prvním typem je lineární rozvržení, které objekty zarovnává do řádků. Druhým použitým typem je relativní rozvržení, které Widgety pozicuje relativně vůči ostatním Widgetům, rozvržením, nebo vůči okrajům obrazovky.

Pro vytváření a definování objektů uživatelského rozhraní slouží XML soubory umístěné v adresáři /res/layout. Objekty lze definovat i přímo v kódu aplikace. První řešení je však lepší, protože odděluje objekty uživatelského rozhraní od logiky aplikace.

3.2.2 Aktivita

Aktivita je základní vizuální komponenta reprezentující jednu obrazovku aplikace. V této práci je použito více obrazovek, k nimž bylo vytvořeno odpovídající množství aktivit. Tyto aktivity jsou mezi sebou provázány a vzájemně se mohou volat. Při zavolání jiné aktivity je činnost současné aktivity pozastavena a uložen na zásobník.[20] Stisknutím tlačítka zpět je vybrána aktivita z vrcholu zásobníku a její činnost je obnovena. Při spuštění aplikace je zavolána hlavní aktivita, která se v kontextu této aplikace jmenuje Main a uživateli oznamuje o nutnosti přiblížení se k Beaconu.

3.2.3 AndroidManifest

AndroidManifext.xml je konfigurační soubor uložený v kořenovém adresáři projektu a udávající operačnímu systému informace o komponentách a oprávněních, jež aplikace vyžaduje. V této práci je zde uveden název Java balíčku, výčet aktivit, deklarace služby pro vyhledávání Beaconu na pozadí, minimální verze Android API (16 - Android 4.1 - vyžaduje Estimote SDK) a oprávnění pro přístup aplikace k internetu a k Bluetooth.

3.2.4 AsyncTask

Abstraktní třída *AsyncTask* využívá vlákna pro asynchronní spuštění určité části kódu, která posléze běží na pozadí. V aplikaci je využita pro stahování dat a stahování obrázku interiéru restaurace. Při stahování dat ze serveru je uživateli zobrazena animace značící probíhající stahování. Pokud by nebylo využito vláken, aplikace by nemohla zároveň stahovat a zobrazovat animaci a došlo by tak k chvilkovému „zamrznutí“ aplikace, dokud by nedošlo k úspěšnému stažení dat.

Třídy pro asynchronní procesy v aplikaci jsou potomky právě abstraktní třídy AsyncTask, která implementuje čtyři metody. První metoda *onPreExecute* je volána před započetím asynchronního procesu a nastavuje například animaci pro načítání. Druhou metodou *doInBackground* je volána po dokončení metody *onPreExecute* a definuje proces, který se má vykonávat na pozadí. V metodě nelze nastavovat prvky uživatelského rozhraní, k čemuž jsou používány ostatní tři metody této třídy. Parametry předávané při volání asynchronní události jsou předávány metodě *doInBackground*. V popsané metodě lze využít volání *publishProgress*, jenž volá metodu *onProgressUpdate*, jejíž cílem je změna prvků v uživatelském rozhraní například pro zobrazení postupu stahování. Po dokončení činnosti na pozadí je zavolána metoda *onPostExecute*, je v parametru předán výsledek činnosti na pozadí.[21] Metoda poté může uživateli zobrazit výsledek daného procesu.

3.2.5 JSON

Formát, ve kterém aplikace přijímá data ze serveru, je JSON. Java obsahuje pro práci s JSON objekty třídu JSONObject. Tato třída napomáhá při parsování dat pomocí svých metod, a tak nebylo zapotřebí v této práci vytvářet vlastní parser.

3.2.6 Cachování

Z důvodu zmenšení objemu stahovaných dat jsou tato data ukládána do paměti. Data se tak stáhnou jednou, a poté se již opakovaně nestahují při listování aktivitami aplikace.

Při tvorbě větších aplikací je běžnou praxí tvorba databáze na straně aplikace, která je postupně naplňována daty získávanými z webového serveru. V prostředí android se k tomuto využívá například databáze SQLite. Webový server si do své databáze ukládá i údaj o verzi databáze, kterou zasílá spolu s daty aplikaci a při změně dat verzi navyšuje. Při dotazu aplikace na webový server je serveru skrze API předána verze databáze v aplikaci. Server aplikaci vrátí jen ty změny, které odpovídají rozdílu verzí databází, to znamená, že pokud jsou verze shodné, server aplikaci nepošle žádné data.

V této práci tento koncept nebyl použit. Důvodem je, že aplikace neobsahuje historii nalezených restauraci, ale pouze aktuálně nalezenou restauraci. Výsledkem by byly dotazy na databázi vracející vždy jen jeden řádek (informace o aktuální restauraci, aktuální jídelní lístek a aktuální akci). Z tohoto důvodu byl implementován jednodušší systém ukládání dat, a to do objektu na místo do databáze.

V aplikaci je vytvořena třída *RestaurantData* obsahující tři proměnné datového typu JSONObject, které si udržují JSON data stažené ze serveru a tři proměnné datového typu bool, značící, zda byly tyto objekty už naplněny daty. Pro implementaci této třídy byl použit návrhový vzor Singleton. Tento návrhový vzor se používá v případech, kdy vyžadujeme v celém programu jen jednu instanci konkrétní třídy. Metodou *getInstance* je vrácena právě tato jediná instance třídy *RestaurantData*. Při požadavku na stažení JSONu je instance dotázána, jestli již došlo v minulosti ke stažení dat, a pokud ano, data jsou použita z této instance. V opačném případě dojde k požadavku na server a data jsou stažena.

3.2.7 Komunikace s Beaconem

Pro komunikaci s Estimote Beaconem se používá knihovna Estimote SDK. Knihovna vyžaduje Android ve verzi 4.3 a vyšší a podporu Bluetooth Low Energy. Tato knihovna nabízí tři funkcionality pro práci s Estimote Beaconem: ranging, monitoring a metody pro čtení a přepis vlastností Beaconu.

3.2.7.1 Beacon ranging

Ranging umožňuje aplikaci zjištění relativní vzdálenosti mezi zařízením a beaconem. K zjištění této vzdálenosti využívá Estimote SDK údaj o síle signálu.[22] Součástí Estimote SDK je třída BeaconManager, obsahující metodu setRangingListener, která přijímá instatnci třídy RangingListener, jenž vyžaduje implementování metody onBeaconsDiscovered. Tato metoda je volána každou sekundu (tento interval lze změnit). V parametru je ji předáván aktuální list beaconů, jenž se nachází v blízkosti zařízení seřazený podle síly signálů. V práci je tento postup implementován a z listu je vždy vybírán beacon na pozici 0. Protože je pole řazeno podle síly signálu, tento beacon by měl být nejblíže aplikaci. UUID tohoto beaconu je zapisováno do globální proměnné, a pokud

je v dalším volání metody onBeaconsDiscovered shodné s UUID beaconu na pozici nula, aplikace jej ignoruje a neprovádí žádnou akci. Toto zabraňuje neustálému informování uživatele o nalezení beaconu. V opačném případě, pokud se hodnoty neshodují byl objeven nový beacon a nachází se blíže zařízení. V takovéto situaci je uživatel upozorněn push-up notifikací a pokud aplikace běží na popředí, je její aktivita změněna na aktivitu zobrazující informace o aktuálně nejbližší restauraci.

3.2.7.2 Beacon monitoring

Monitoring sleduje, zda aplikace vstoupila nebo vystoupila z regionu beaconů. Region beaconů je definován následujícími hodnotami:[23]

- UUID - 128 bitový jednoznačný identifikátor
- major - 16 bitová hodnota typu unsigned integer pro rozlišení beaconů se stejným UUID
- minor - 16 bitová hodnota typu unsigned integer pro rozlišení beaconů se stejným UUID a major hodnotou

Toto lze využívat pro dělení beaconů do skupin. Například máme-li obchod ve kterém je zboží řazeno do sekcí a ve kterém chceme, aby byl uživatel upozorněn při vstupu do jiné sekce obchodu. V takovémto případě nastavíme všem beaconům stejné UUID, aby bylo zabráněno odchytávání signálů z konkurenčních beaconů, jenž mohou být umístěny ve vedlejších obchodech. Beaconům ve stejné sekci poté nastavíme stejnou hodnotu major a rozlišnou hodnotu minor. Poté mají všechny beacony v obchodě stejné UUID, hodnotu major podle sekce, kde se nacházejí a hodnotu minor podle konkrétního produktu v dané sekci. Při nastavení monitoringu na událost zachycení beaconu s jinou major hodnotou je uživatel upozorněn při přesunu do jiné sekce obchodu.

Estimote SDK tak umožňuje pro jedno 128 bitové UUID nastavení 65 536 různých major hodnot a pro každou tuto hodnotu dalších 65 536 minor hodnot. Po vynásobení těchto čísel získáváme přes 4 miliardy kombinací, což je pro běžné účely více než dostačující.

3.2.7.3 Čtení a přepis vlastností Beaconu

Estimote SDK obsahuje soubor metod pro zápis a čtení vlastností beaconu. Lze takto měnit UUID, major a minor hodnotu, číst tyto hodnoty a další, jako například sílu signálu.

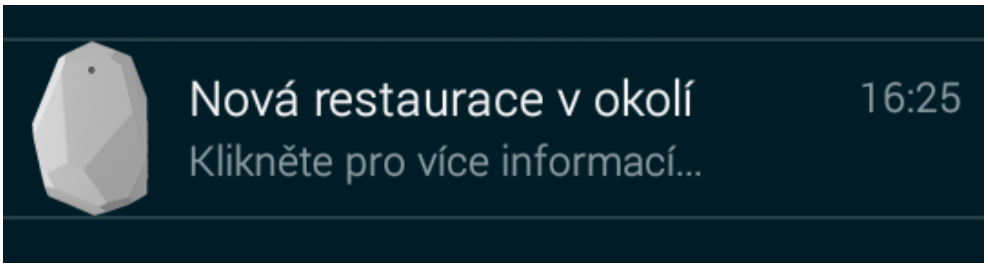
3.2.8 Běh na pozadí

Aby nemusel mít uživatel stále puštěnou aplikaci a mohl využívat jiné funkcionality telefonu, je aplikace schopna běžet na pozadí a uživatele upozornit na blízkost beaconu pomocí zobrazení upozornění ve stavovém řádku.

Běh na pozadí v Androidu obstarávají služby. Výhodou Estimote SDK je, že pro vyhledávání beaconu implicitně využívá službu, která je deklarována v souboru AndroidManifest na řádku 14:

```
<service android:name="com.estimote.sdk.service.BeaconService"
         android:exported="false"/>
```

To znamená, že vyhledávání beaconu probíhá nezávisle na aplikaci a běží, i když je aplikace na pozadí. Při nalezení Beaconu je uživateli zobrazeno upozornění ve stavovém řádku. Android SDK umožňuje nastavení textu, ikony a zvuku notifikace, jak je zobrazeno na obrázku 3.2.



Obrázek 3.2: Zobrazení notifikace při běhu aplikace na pozadí

Po kliknutí na tuto notifikaci je otevřena aplikace na aktivitě s informacemi o nalezené restauraci.

Kapitola 4

Testování

4.1 Testování serveru

Testování administračního rozhraní webového serveru probíhalo formou automatických testů. Velkou výhodou oproti manuálnímu testování je velká úspora času u větších aplikací. Dalším plusem automatických testů je jejich možné spouštění integračním serverem při průběžné integraci. Integrační server provádí build aplikace a spouštění automatických testů během vývoje při průběžné integraci. Pokud některý z testů neprojde, integrační server uloží chybové hlášení do logu. Častou chybou pří vývoji software je narušení staré funkcionality při vývoji nové komponenty aplikace. Při klasickém uživatelském testování toto nemusí být odhaleno, protože uživatel většinou netestuje již otestované části aplikace. Automatické testy jsou toto schopny odhalit.

4.1.1 Selenium

Selenium je komplexní nástroj pro automatické testování webových aplikací skládající se z několika komponent (Selenium IDE, Selenium RC, Selenium WebDrive, Selenium Grid).[24]

- Selenium IDE je plugin do internetového prohlížeče Firefox. Způsob vytváření testů je podobný nahrávání maker - uživatel provádí sekvenci příkazů (klikání, zapisování textu do formulářů, ...). Tyto příkazy jsou nahrávány a poté je lze opakovaně spouštět a testovat webovou aplikaci. Tento způsob je velice snadný a rychlý, ale u některých složitějších testů nemusí dostačovat.

- Selenium RC se skládá ze dvou částí. První částí je server, který automaticky spouští prohlížeče a slouží jako proxy server mezi webovými požadavky a prohlížeči. Druhou částí je sada knihoven pro několik programovacích jazyků, které slouží pro vytváření automatických testů. Vytváření těchto testů je na rozdíl od Selenium IDE náročnější, ale využívá

struktur programovacích jazyku, a lze tak vytvářet složitější a komplexnější testy.

- Selenium WebDrive taktéž využívá struktur programovacího jazyka, ale na rozdíl od Selenium RC volá každý prohlížeč zvlášt a není tak potřeba proxy server.
- Selenium Grid umožňuje testování na více platformách a prohlížečích současně.

4.1.2 Testovací plán

Testovací plán je dokument skládající se z testovacích případů (test cases) a byl vytvořen pro definici testů této práce. Testovací případ slouží pro zdokumentování jednoho konkrétního testu. Je zde uveden název testu, nutné podmínky před spuštěním testu, postup kroků testu a očekávaný výsledek. Testovací plán v této práci se skládá z osmi testovacích případů.

4.1.3 Automatické testování

Automatické testy byly vytvořeny podle testovacího plánu. Testy jsou napsány v jazyce Java a byly spouštěny pomocí komponenty SeleniumWebdrive. Při psaní těchto testů bylo využito dvou návrhových vzorů:

- Page Object - Tento návrhový vzor definuje strukturu kódu testu pro lepší orientaci. Pro každou samostatnou stránku webu je vytvářena třída. Jednotlivé metody třídy jsou události vykonávané na dané stránce. Příkladem je přihlašovací stránka. Byla vytvořena třída LoginPage, která reprezentuje stránku pro přihlášení. Třída obsahuje metody *setUserName* - vyplní uživatelské jméno přijaté v parametru, *setPassword* - vyplní heslo přijaté v parametru a metodu signIn - provádí klik na přihlašovací tlačítko.
- Page Factory - Tento vzor slouží ke zjednodušení tvorby webových elementů. Definuje anotaci @FindBy.

4.2 Testování aplikace

Pro testování aplikace bylo použito uživatelské testování. Princip tohoto testování spočívá ve sledování samotných uživatelů. Pro komfortnější distribuci a sledování uživatelů slouží v dnešní době několik služeb jako například známá služba TestFlight, jenž je vyhrazena výhradně pro vývojáře iOS aplikací. Z tohoto důvodu nemohla být použita a naskytovalo se několik alternativ. Další známou službou je služba HockeyApp od Micrososftu. Tato služba se pyšní podporou pro iOS, Android a WindowsPhone. Služba poskytuje bezplatné

testování po dobu jednoho měsíce, poté je vývojář nucen platit měsíční částky začínající na 10$/měsíc.[25] Pro tuto práci stačil časový úsek jednoho měsíce. HockeyApp byla zvolena pro testování.

4.2.1 HockeyApp

Služba nabízí webovou aplikaci, kde po úvodní registraci může vývojář nahrávat beta verze svých aplikací. Přidávání testerů probíhá zasláním pozvánek do jejich emailových schránek. Po přijetí pozvánek a registraci testerů do aplikace může vývojář testery rozdělovat do skupin. Těmto skupinám pak lze přidělovat aplikace, ke kterým mají přístup. Testeři si poté stáhnou do svého zařízení aplikaci HockeyApp a skrze ni si stáhnou požadované aplikace, které poté testují. Vývojář může nahrávat v průběhu testování nové verze aplikace, kdy při nahrání dojde k upozornění testerů o nové verzi. Webová aplikace nabízí souhrnné statistky, kde vývojář vidí, kdo si stáhnul jaké aplikace, kdo z testerů má jaké zařízení, počet stažení, data stažení atd. Hlavním cílem uživatelského testování je zpětná vazba testerů. V HockeyApp pro toto slouží záložka „Feedback“, do které může tester napsat své připomínky. Velkou sílou služby je SDK pro všechny podporované platformy (Android, iOS, WindowsPhone). HockeyApp SDK pro android je knihovna v Javě, kterou lze přidat do projektu vyvíjené aplikace. Po přidání několika řádek kódu k současné testované aplikaci je vývojář schopen zaznamenávat chybová hlášení při „pádu“ aplikace. Pokud tedy dojde při testování některým z testerů k vyvolání chyby a následnému „pádu“ aplikace, služba HockeyApp zašle automaticky tyto informace vývojáři, a ten si je může posléze zobrazit ve webovém rozhraní HockeyApp.

4.2.2 Průběh testování

Pro testovací účely byla vytvořena modifikace originální verze, kde se na hlavní obrazovce nachází tlačítko "Test it", po jehož zmáčknutí dojde k simulaci navázaní spojení s Beaconem. Toto bylo provedeno z důvodu, že žádný tester neměl k dispozici Estimote Beacon.

Po založení účtu byla tato práce vložena do služby HockeyApp. Byla vytvořena skupina testerů, kterým byla přidána tato modifikovaná aplikace pro testování. Při testování první verze docházelo u většiny testerů k pádům aplikace při spuštění. Důvodem byla špatná hodnota verze v AndroidManifest.xml, kde byla nastavena minimální verze systému Android na hodnotu 4.1. Verze nižší než Android 4.3 neobsahují standard Bluetooth 4.0 a při spuštění hledání Beaconů v okolí došlo k „pádu“. Po opravení chyby a nahrání nové verze nebylo testerům s nízkými verzemi Android stažení umožněno. Ostatní testeři po otestování nové verze hlásili bezproblémový provoz aplikace.

Kapitola 5

Závěr

Cílem této práce bylo vytvoření aplikace pro sytém Android. Práce splnila cíle, které si kladla a tento dokument popisuje vývoj aplikace od analýzy, přes návrh, implementaci až po finální testování.

Samotná aplikace je plně funkční se svou základní funkcionalitou. Je zde samozřejmě velký prostor pro rozšíření. Za zmínku stojí třeba systém uživatelského hodnocení restaurací, na který jsme zvyklí u konkurenčních aplikací. Velkou slabinou celého konceptu je systém zadávání jídelních lístků. Uživatel je nucen jídla vypisovat ručně do předpřipravených formulářů. Dobrým vylepšením aplikace by bylo umožnění exportu a importu jídelních lístků v různých formátech.

Hlavní zajímavostí práce byly Bluetooth Beacony, s kterými aplikace komunikovala. Práce se snaží nastínit fungování a princip této technologie. Technologie je jen několik let stará a v tuzemsku dosud skoro nevyužívaná. Toto otevírá velké možnosti pro začínající tuzemské start-upy, které by chtěli vytvořit inovativní řešení v oblasti geolokace a identifikace předmětů s využitím mobilních aplikací. Oborů využití je mnoho a nemusí se jednat konkrétně o oblast pohostinství. Příkladem může být například Brooklinské muzeum, které ke svým exponátům umístilo Beacony a vytvořilo pro zákazníky aplikaci, jež jim umožňuje dozvědět se bližší informace o nejbližším exponátu.[26]

Literatura

[1] Expanze nákupních center v ČR. Incoma GfK [online]. 2015 [cit. 2015-04-23]. Dostupné z: http://incoma.cz/expanze-nakupnich-center-v-cr/

[2] Velkoplošné obchodní komplexy již mají v České republice 4,5 milionů m2. Incoma GfK [online]. 2015 [cit. 2015-04-23]. Dostupné z: http://incoma.cz/velkoplosne-obchodni-komplexy-jiz-maji-v-ceske-republice-45-milionu-m2/

[3] Meníčka. Google play [online]. 2015 [cit. 2015-04-23]. Dostupné z: https://play.google.com/store/apps/details?id=cz.Menicka.Menicka

[4] Zomato. Google play [online]. 2015 [cit. 2015-04-23]. Dostupné z: https://play.google.com/store/apps/details?id=com.application.zomato

[5] Official Bluetooth technology website [online]. 2015 [cit. 2015-04-19]. Dostupné z: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx

[6] Beecon - The iBeacon Automation App [online]. 2015 [cit. 2015-04-19]. Dostupné z: http://www.beaconsandwich.com/what-is-ibeacon.html

[7] Měření tepu s bluetooth pásem Polar H7. Web o měření tepové frekvence při sportu [online]. 2015 [cit. 2015-04-20]. Dostupné z: http://sporttester.info/2014/mereni-tepu-s-bluetooth-pasem-polar-h7/

[8] API Documentation. Estimote Beacon [online]. 2015 [cit. 2015-04-20]. Dostupné z: http://estimote.com/api/

[9] API Documentation. Estimote Beacon [online]. 2015 [cit. 2015-04-20]. Dostupné z: http://estimote.com/

[10] Estimote's Explosive Growth: 10,000 iBeacon Developers. BEEKn | Beacons, brands and culture on the Internet of Things [online]. 2015 [cit. 2015-04-20]. Dostupné z: http://beekn.net/2013/12/estimote-startup-growth/

[11] Tour. 2015. UXPin [online]. [cit. 2015-05-10]. Dostupné z: http://www.uxpin.com/tour.html

[12] MySQL :: MySQL Workbench. 2015. MySQL [online]. [cit. 2015-05-10]. Dostupné z: https://www.mysql.com/products/workbench/

[13] How it works. 2015. Apiary [online]. [cit. 2015-05-10]. Dostupné z: https://apiary.io/how-it-works

[14] PHP. PDO Drivers [online]. 2015 [cit. 2015-04-21]. Dostupné z: http://php.net/manual/en/pdo.drivers.php

[15] Databáze. 2015. Nette [online]. [cit. 2015-05-11]. Dostupné z: http://doc.nette.org/cs/2.3/database

[16] Přihlašování & oprávnění uživatelů. 2015. Nette [online]. [cit. 2015-05-11]. Dostupné z: http://doc.nette.org/cs/2.3/access-control

[17] Formuláře. 2015. Nette [online]. [cit. 2015-05-11]. Dostupné z: http://doc.nette.org/cs/2.3/forms

[18] Šablony. 2015. Nette [online]. [cit. 2015-05-11]. Dostupné z: http://doc.nette.org/cs/2.3/templating

[19] Lesson 16. Creating layout programmatically. LayoutParams. 2015. Android tutorial. Lessons for beginners [online]. [cit. 2015-05-11]. Dostupné z: http://startandroid.ru/en/lessons/complete-list/220-lesson-16-creating-layout-programmatically-layoutparams.html

[20] Programujeme pro Android. 2013. Praha: Grada Publishing, a.s. ISBN 978-80-247-4863-4.

[21] AsyncTask. 2015. Android Developers [online]. [cit. 2015-05-11]. Dostupné z: http://developer.android.com/reference/android/os/AsyncTask.html

[22] What are Broadcasting Power, RSSI and Measured Power? 2015. Estimote [online]. [cit. 2015-05-11]. Dostupné z: https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI-and-Measured-Power-

[23] What is a beacon region? 2015. Estimote [online]. [cit. 2015-05-11]. Dostupné z: https://community.estimote.com/hc/en-us/articles/203776266-What-is-a-beacon-region-

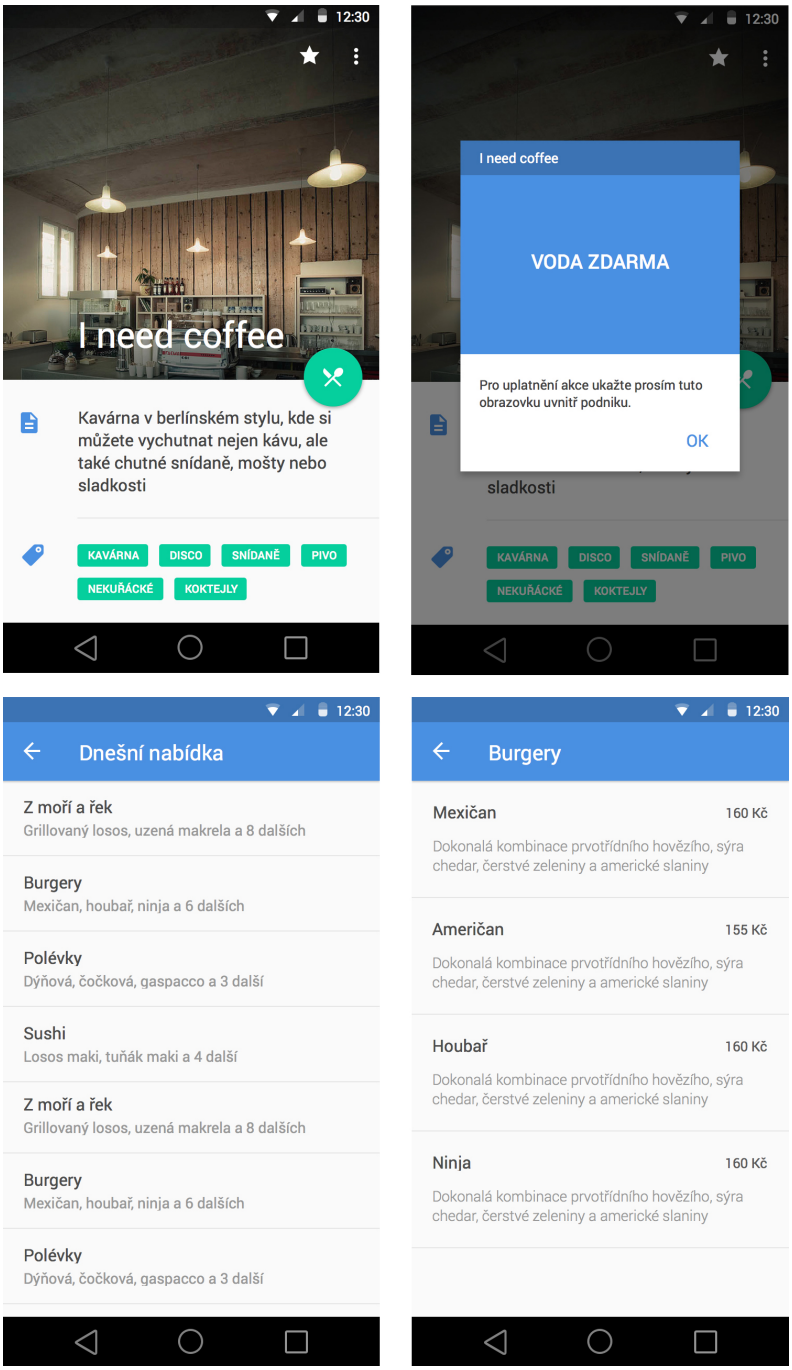
[24] Selenium Documentation. 2015. Selenium - Web Browser Automation [online]. [cit. 2015-05-11]. Dostupné z: http://www.seleniumhq.org/docs/

[25] Our plans and prices. 2015. HockeyApp - The Platform for Your Apps [online]. [cit. 2015-05-11]. Dostupné z: http://hockeyapp.net/pricing/

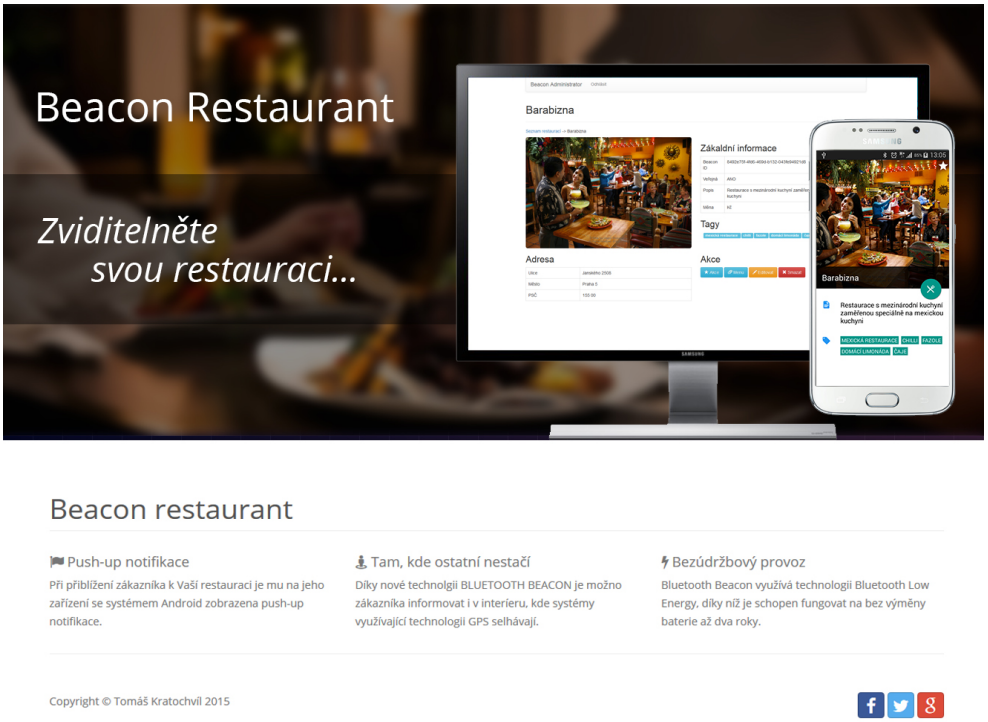
[26] The Realities of Installing iBeacon to Scale. 2015. Technology blog of the Brooklyn Museum [online]. [cit. 2015-05-08]. Dostupné z: http://www.brooklynmuseum.org/community/blogosphere/2015/02/04/the-realities-of-installing-ibeacon-to-scale/

Příloha A

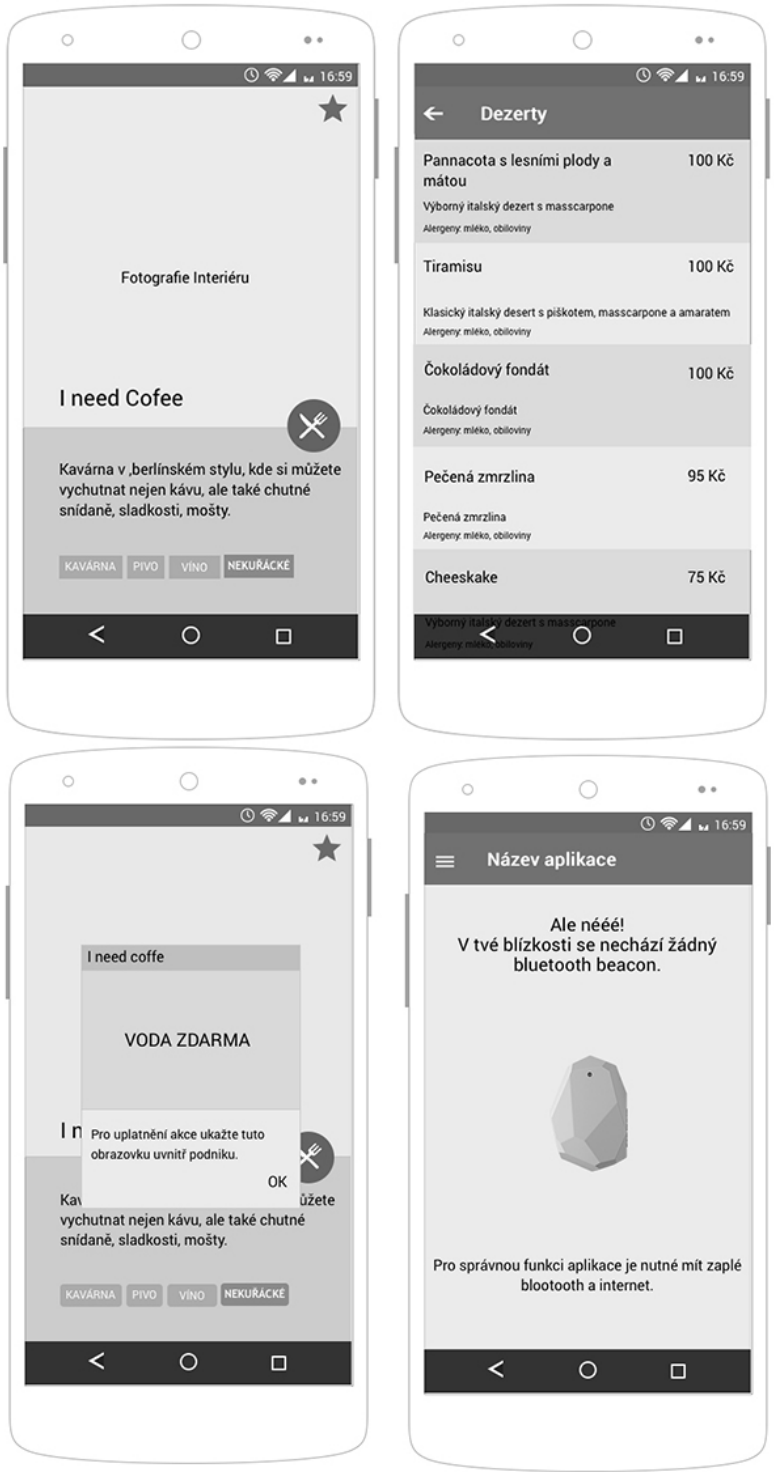
Doplňky



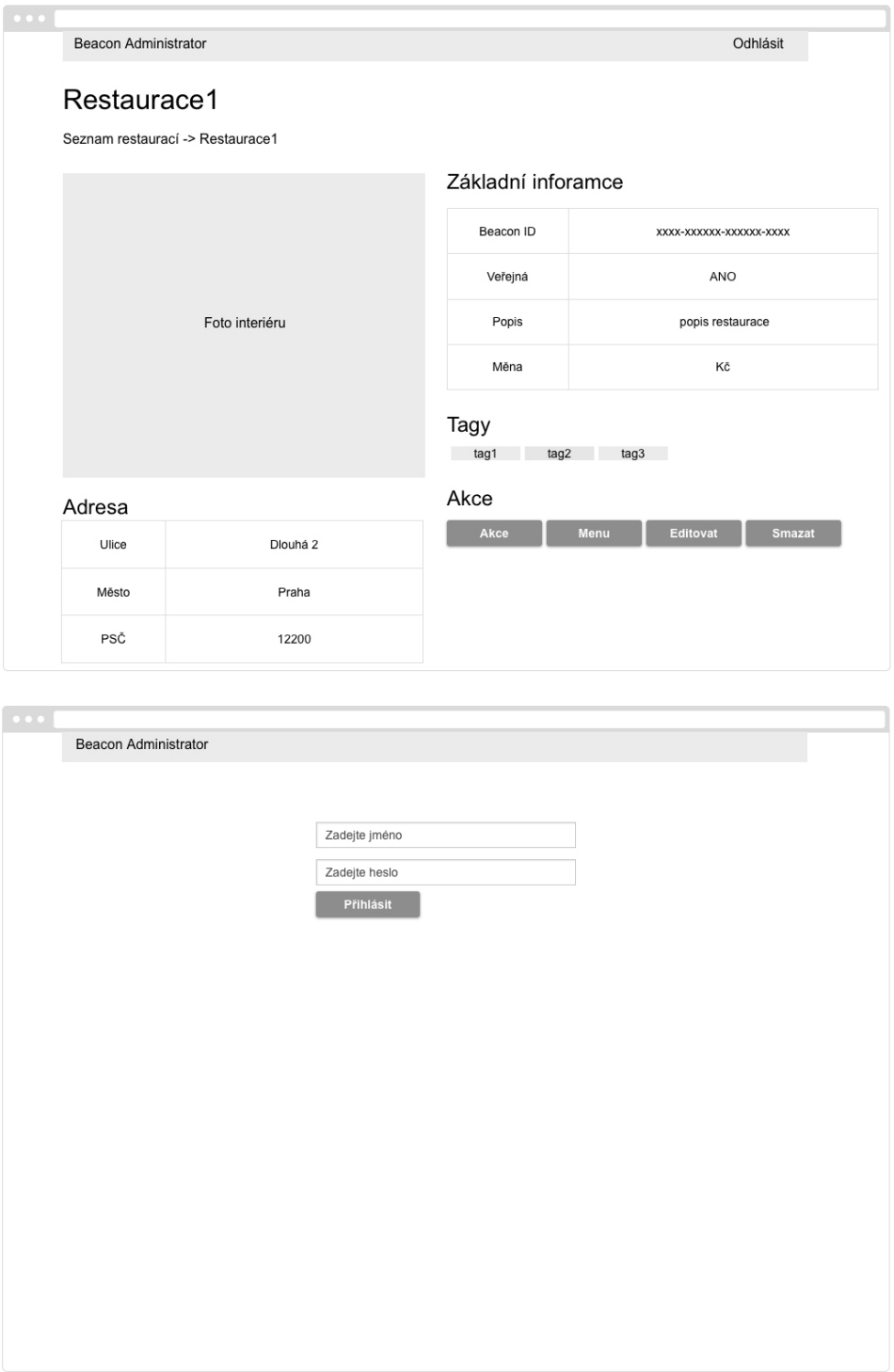
Obrázek A.1: Grafický návrh aplikace



Obrázek A.2: Grafický návrh microsite

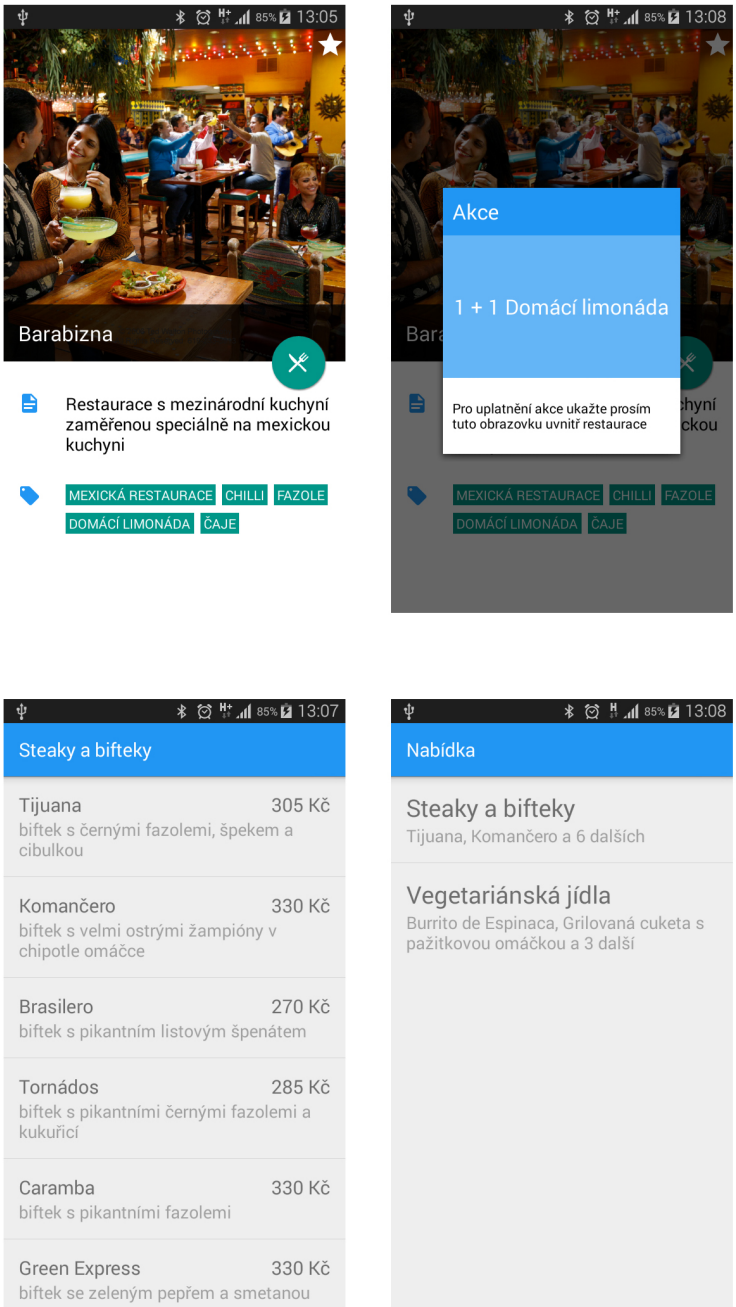


Obrázek A.3: Drátěný model aplikace



Obrázek A.4: Drátěný model serveru

Obrázek A.5: Drátěný model serveru



Obrázek A.6: Výsledná aplikace

Příloha B

Seznam použitých zkratek

GPS Global Positioning System

BLE Bluetooth Low Energy

NFC Near Field Communication

ARM Advanced RISC Machine

API Application Programming Interface

REST Representational State Transfer

SQL Structured Query Language

REST Bluetooth Low Energy

PHP Hypertext Preprocessor

HTML HyperText Markup Language

CSS Cascading Style Sheets

JS JavaScript

MVC Model-view-controller

ID IDentification

SDK Software Development Kit

XML Extensible Markup Language

JSON JavaScript Object Notation

UUID Universally unique identifier

IDE Integrated Development Environment

RC Remote Control

HTTP Hypertext Transfer Protocol

QR Quick Response

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `Code`
  - `Aplication` ..... zdrojové kódy implementace aplikace
  - `Server` ..... zdrojové kódy implementace webového serveru
    - `Database`
      - `db.sql` ..... skript pro tvorbu databáze ve formátu SQL
      - `db.png` ..... obrázek relačního modelu databáze
  - `Microsite` ..... zdrojové kódy microsite
- `Design`
  - `Aplication` ..... design aplikace
  - `Microsite` ..... design microsite
- `Testing`
  - `Server` ..... testy webového serveru
    - `Selenium` ..... testy v Seleniu WebDrive
    - `TestPlan.txt` ..... testovací plán
- `Wireframes`
  - `aplication.pdf` ..... drátěný model aplikace
  - `server.pdf` ..... drátěný model serveru
- `Text`
  - `thesis.tex` ..... text práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu pdf