FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Webová aplikace pro anotace a doporučení denních menu
Student:	Bc. Michal Kváček
Vedoucí:	Ing. Jaroslav Kuchař, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je vytvoření webové aplikace poskytující aktuální přehled o denních menu vybraných restaurací. Součástí aplikace bude doporučovací systém, který na základě informací o uživatelích bude navrhovat polední menu pro aktuální den.
Požadavky:
- Proveďte průzkum existujících podobných aplikací.
- Navrhněte a implementujte modulární rozhraní pro získávání informací o denních menu restaurací.
- Zajistěte automatické rozdělení denních menu do definovaných kategorií a jejich automatickou anotaci. Takto anotovaná denní menu budou sloužit jako podklad pro doporučovací systém.
- Navrhněte, implementuje a otestujte rozhraní pro uživatele, které zajistí získání zpětné vazby a kde najdou doporučená jídla pro aktuální den.
- Aplikace bude realizována jako open-source.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Webová aplikace pro anotace a doporučení denních menu

Bc. Michal Kváček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jaroslav Kuchař, Ph.D.

9. května 2019

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 9. května 2019 .....

Odkaz na tuto práci

Kváček, Michal. *Webová aplikace pro anotace a doporučení denních menu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce popisuje návrh a následnou implementaci jednoduchého doporučovacího systému zaměřeného na denní menu. V práci je řešen problém komplexně - od získání dat až po zobrazení doporučení ve webové aplikaci. Tato webová aplikace je naprogramována jako jednostránková aplikace založená na frameworku Vue.js. Doporučovací systém je typu content based.

Klíčová slova klasifikace, zpracování jazyka, dopočučení, denní menu

Abstract

This thesis describes analysis and implementation of simple recommender system used for recommending daily menus. Text covers the problem in complex way - from scraping data about daily menu to delivering final recommendation to users via web application. The final application is build as single page application on top of Vue.js framework. The recommender system was implemented as content based.

Keywords classification, natural language processing, recommandations, daily menu

Obsah

Seznam obrázků

Úvod

Rozpoznání obsahu a jeho následné doporučování uživatelům se v poslední době stalo velmi populární a uživatel internetu se s nějakou formou doporučovacích systémů setkává prakticky na denní bázi. V této práci bych rád vyzkoušel navrhnout a implementovat systém, který bude uživatelům doporučovat denní menu na základě jejich preferencí.

V této práci provedu analýzu existujících aplikací a služeb, které se zabývají zobrazováním informací o denních nabídkách restaurací, následně provedu analýzu požadavků a představ uživatelů a navrhnu uživatelské rozhraní aplikace. Na základě těchto dat provedu technickou analýzu a pokusím se navrhnout funkční doporučovací systém.

Mým cílem není vytvořit složitý systém pro tvorbu doporučování. Místo toho se pokusím navrhnout aplikaci, která bude problém doporučování řešit komplexně - od samotného získání vstupních dat, přes jejich klasifikaci až po doporučení konkrétního jídla uživateli.

V rámci své diplomové práce bych si rád splnil i jeden svůj osobní cíl. Rád bych se seznámil s novými technologiemi, se kterými se běžně ve své praxi nesetkávám.

Kapitola 1

Doporučovací systémy

Jako doporučovací systém se obvykle označuje takový druh filtrovacího programu, který se snaží odhadovat preference (nebo hodnocení) uživatele [1]. K tomu je nutné znát informace jednak o uživateli a jednak o doporučovaném produktu.

S doporučovacími systémy se uživatelé internetu setkávají na denní bázi. Ať už se jedná o sociální sítě, video portály (YouTube), generátory playlistů (Spotify) a mnoho dalšího. Systémů existuje celá řada, v následujících odstavcích se pokusím stručně popsat dva modely. Těmi jsou *Collaborative filtering* a *Content Based system*.

1.0.1 Collaborative filtering

Tento druh doporučovacího systému je založen na myšlence, že podobní uživatelé budou mít podobné preference. Pro výpočet podobnosti mezi uživateli se často používají algoritmy typy KNN [2].

Tento systém předpokládá, že doporučovaných produktů je méně, než samotných uživatelů. V opačném případě by prostor, ve kterém je vyhledávána podobnost byl příliš řídký a doporučování by bylo prakticky náhodné.

1.0.1.1 Výhody

- Jednoduchost implementace
- Relativní transparentnost
- Variabilita

U tohoto typu doporučení je relativně snadné odhadnout, proč byl daný produkt uživateli doporučen. Mimo to je možné přístup doporučování aplikovat prakticky v jakékoliv oblasti.

Mimo uvedené výhody má samozřejmě i tento typ své nevýhody.

1.0.1.2 Nevýhody

- Problém „nového produktu“
- Nutnost velkého množství dat

Jako „new product problem“ se označuje situace, kdy je do katalogu přidán nový produkt, který zatím nebyl hodnocen, případně s ním interagovalo jen velmi málo lidí. Tím pádem nemůže být ani doporučován dalším uživatelům.

Tento druh doporučovacího systému vyžaduje rozsáhlou databázi uživatelských interakcí a obecně dat, ze kterých lze vycházet [3].

1.0.2 Content based

Content based doporučovací systém funguje na základě porovnávání rozdílnosti doporučovaného produktu a uživatelských preferencí. Tento přístup je vhodný ve chvíli, kdy má systém k dispozici informace o doporučovaných produktech ale už ne o uživatelích. Preference uživatele jsou tak rozpoznávány na základě jednotlivých vlastností produktů [4].

1.0.2.1 Výhody

- Řeší problém „nového produktu“
- Jednoduchost imlementace
- Transparentnost doporučení

Díky tomu, že content based system pracuje přímo s informacemi daného produktu, může do doporučení rovnou zahrnout i produkt, který nikdo ještě nehodnotil. Podobně jako Collaborative filtering, i v tomto případě je relativně snadné odhadnout, proč byl daný produkt uživateli doporučen.

1.0.2.2 Nevýhody

- Nutnost znalosti informací o produktu
- Doporučení mohou „uvíznout v bublině“

Zřejmou nevýhodou tohoto přístupu je fakt, že musíme znát detailně vlastnosti produktu. V případě, že tyto informace nejsou k dispozici, klasifikace může být naprosto špatná a doporučení tím pádem nerelevantní. Druhým problémem je možnost, že uživatel bude dostávat doporučení např. na produkty, které již koupil [5].

1.1 Zpětná vazba

Zpětnou vazbou v terminologii doporučovacích systémů se myslí proces získávání informací o uživateli. Tento proces se dělí na dvě metody: explicitní a implicitní.

Explicitní zpětná vazba je založena na nějaké konkrétní akci uživatele, např. hodnocení produktu.

Naopak **implicitní zpětná vazba** nevyžaduje od uživatele žádnou akci, jen sleduje jeho chování. Za tento způsob získávání informací můžeme považovat např. seznam prohlédnutých produktů.

Kapitola 2

Existující služby

Na internetu jsem se snažil najít webové aplikace, které se týkají denních menu restaurací, případně restaurací obecně. Pokoušel jsem se nalézt službu, která by dokázala řešit problém s doporučováním (případně automatickou klasifikací jídel).

V rámci této kapitoly bych rád identifikoval služby, které budou moci sloužit jako zdroj dat pro výslednou aplikaci. V této části se zaměřím na primární zdroj dat, tedy na samotná denní menu.

2.1 Způsob hodnocení

V této části bych rád popsal, jakým způsobem budu hodnotit jednotlivé existující služby. Zaměřím se na uživatelskou část a na možnosti využití dat z aplikace pro své potřeby.

V první části se budu věnovat uživatelskému rozhraní aplikace. Subjektivně zhodnotím přehlednost a stručně popíšu dostupné funkce.

V druhé části prozkoumám, zda by nešly využít data ze zkoumané služby buď pro klasifikaci, nebo pro samotné zobrazování denních menu. Vzhledem k tomu, že data vznikají ručně (jsou zadávána lidmi) budu hodnotit i jejich kvalitu - tedy zda uživatelé do denních menu zadávají opravdu jen názvy jídel či i jiné věci.

2.2 Zomato

Obrázek 2.1: Zomato logo

Zomato je dnes poměrně známá služba. Primárně slouží jako vyhledávač restaurací. Kromě hodnocení jednotlivých podniků obsahuje Zomato funkcionalitu i pro nahrání jídelního/nápojového lístku nebo i denního menu. Zomato funguje mezinárodně, konkrétně ve 24 zemích světa [6]. Mimo vyhledávání nových i známých restaurací nabízí i objednávku jídel [7]. Zda to ale funguje i v České republice jsem nezkoušel.

2.2.1 Vzhled a funkce

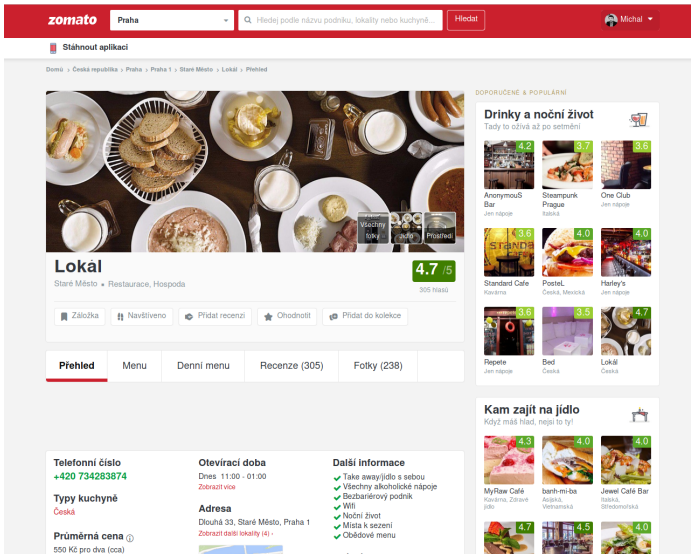
Uživatelské prostředí Zomato je intuitivní a příjemné. Na titulní straně jsou zobrazené tzv. „kolekce“, jedná se tedy o jakousi kategorizaci (Hity týdne, Happy Hours, Žijeme zdravě, ...). Samotný detail restaurace pak obsahuje informace o podniku, hodnocení, stálou nabídku, denní menu atp.

2.2.2 Možnost využití dat a jejich kvalita

Zomato obsahuje velké množství dat nejenom pro Českou republiku. Tato data jsou vkládána přímo uživateli spravujícími danou restauraci. Poměrně často se stává, že seznam denní nabídky obsahuje položky, které určitě nejsou jídla. Takovým příkladem je např. kategorizace (obrázek 2.3), případně rozdělování jednoho jídla do více řádků (obrázek 2.4).

Data o restauracích jsou zpřístupněná přes jejich REST API. Tento způsob získání dat má ovšem své omezení - běžný uživatel je limitován počtem 1000 požadavků denně [8]. Toto bohužel není jediné omezení. API nabízí i možnost vyhledávání podle kategorií, např. umožňuje získat všechny restaurace s denním menu. Bohužel toto vyhledávání poskytuje jen prvních 100 výsledků – tím pádem se příliš nedá použít pro naplnění seznamu restaurací.

Žádnou formu doporučování jsem v tomto systému nenalezl, nicméně vzhledem k existenci REST API zakomponuji tuto službu (i přes její omezení) do své následující práce.



Obrázek 2.2: Detail restaurace v Zomato.com

2.3 Restu

Restu je v mnoha ohledech podobné, jako již zmíněné Zomato. Obsahuje také informace o nabídce, hodnocení a komentáře uživatelů. Oproti Zomato však Restu nabízí navíc možnost rezervace stolu - pro určitý počet lidí na konkrétní hodinu. Uživatel tak rovnou ví, zda-li má daná restaurace v požadovaný čas volno či nikoli.

2.3.1 Vzhled a funkce

Jak již bylo zmíněno v úvodu, Restu se soustředí především na zarezervování stolu. Jednotlivé restaurace jsou taktéž kategorizovány.

Oproti Zomato jsem našel záložku „Doporučené“. Tato sekce zobrazuje přihlášenému uživateli seznam restaurací, které by ho mohly oslovit. Data služba získává pravděpodobně z předchozích rezervací a určitě i ze seznamu „Moje oblíbené“. V době psaní tohoto textu jsem však žádnou restauraci v „oblíbených“ neměl, Restu tedy muselo brát data jen na základě provedených rezervací. Subjektivně mi přijde, že doporučování v tomto případě funguje dobře, neboť tři z pěti restauracích zobrazených na obrázku 2.6 znám a mám s nimi dobré zkušenosti.

2.3.2 Možnost využití dat a jejich kvalita

Co se týče nabídky denních menu, i Restu.cz nabízí přehled poledního menu. Zdá se mi, že na zveřejňování denních menu je více používané Zomato. Tento fakt je podpořen i tím, že Restu se profiluje jako průvodce po restauracích

ČESNEČKA S KRUTONY	**35 Kč**
ŽAMPIONOVÝ KRÉM	**35 Kč**
Zvýhodněné Tříchodové menu za	**125 Kč**
ČESNEČKA S KRUTONY NEBO ŽAMPIONOVÝ KRÉM	
VEPŘOVÁ JÁTRA NA GRILU S VAŘENÝM BRAMBOREM A TATARSKOU OMÁČKOU	
DOMÁCÍ MOUČNÍK	
Hlavní chody	
200g KRŮTÍ STEAK SE ŠVESTKOVOU OMÁČKOU, DOMÁCÍ BRAMBOROVÉ KROKETY	**159 Kč**
UZENÁ KRKOVIČKA S ČERVENÝM ZELÍM A HOUSKOVÝM KNEDLÍKEM	**119 Kč**
200g STEAK Z VEPŘOVÉ KOTLETY SE SÁZENÝM VEJCEM, HRANOLKY	**139 Kč**
Fit jídlo	
200g MARINOVANÉ KUŘECÍ PRSO S KUSKUSEM SE ZELENINKOU	**139 Kč**
Vegetariánské jídlo	
ROZPEČENÝ HERMELÍN SE ZELENINOVÝM SALÁTKEM, BRUSINKOVÝM DIPEM	
A VAŘENÝM BRAMBOREM	**129 Kč**
Salát za	**129 Kč**
SALÁT S GRILOVANÝM KUŘECÍM MASEM A KUKUŘICÍ	
(TRHANÉ SALÁTY, RAJČATA, OKURKA, KUKUŘICE, KUŘECÍ MASO, JOGURTOVÝ DRESING)	

Obrázek 2.3: Zomato.com - kategorizace jídel v denní nabídce

a také jako rezervační systém [9]. Pokud se ale podaří nějakou restauraci s denním menu najít, většinou má data poměrně kvalitní.

Pro účely své práce však data z tohoto portálu nemohu použít, neboť se mi nepodařilo získat souhlas s přebíráním a publikací dat obsažených na stránkách. Restu zakazuje jakékoli publikování dat bez předchozího souhlasu [10].

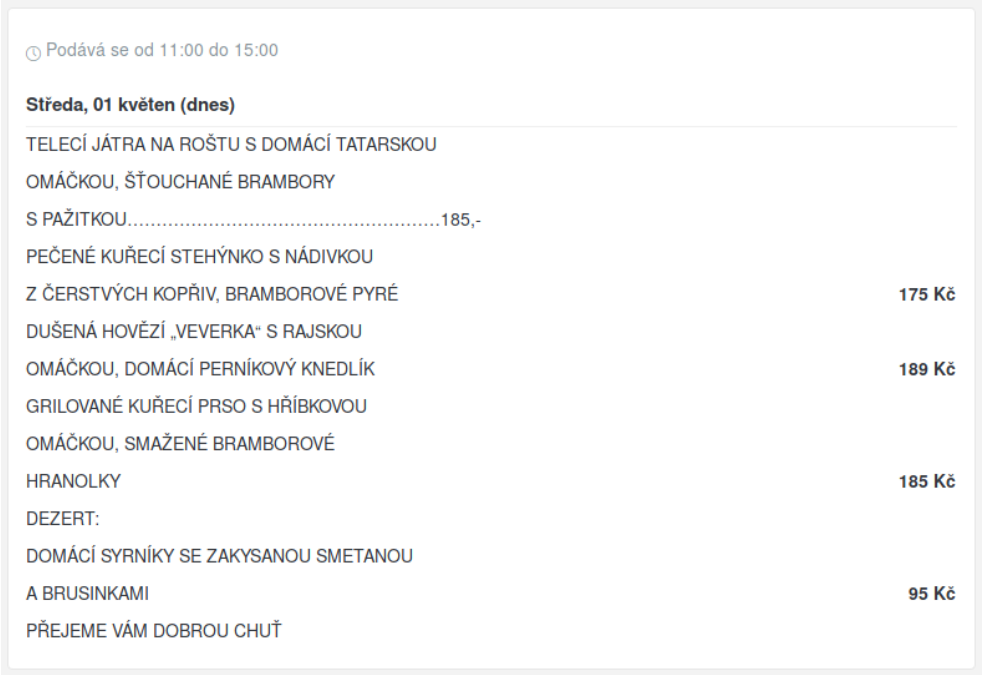
2.4 MenuPraha.cz

2.4.1 Vzhled a funkce

MenuPraha.cz se zaměřuje jen na zobrazování denních menu a informací o restauracích. Dle všeho se tedy zdá, že MenuPraha.cz žádná data neanalyzuje.

Vzhled aplikace je poměrně jednoduchý, na titulní straně se nachází seznam denních menu vybraných restaurací. Bohužel se mi nepodařilo zjistit, podle jakého klíče jsou restaurace vybírány.

Mimo zobrazení denní nabídky se dají v aplikaci najít i informace o restauraci - otevírací doba, odkaz na web, kontaktní telefon. Jednotlivé restaurace se dají hodnotit a psát k nim recenze. Příjemná funkce je „Nejbližší denní menu“. Tento seznam v pravém sidebaru obsahuje restaurace s denním menu



Podává se od 11:00 do 15:00

Středa, 01 květen (dnes)

TELECÍ JÁTRA NA ROŠTU S DOMÁCÍ TATARSKOU	
OMÁČKOU, ŠŤOUCHANÉ BRAMBORY	
S PAŽITKOU.....185,-	
PEČENÉ KUŘECÍ STEHÝNKO S NÁDIVKOU	
Z ČERSTVÝCH KOPŘIV, BRAMBOROVÉ PYRÉ	**175 Kč**
DUŠENÁ HOVĚZÍ „VEVERKA“ S RAJSKOU	
OMÁČKOU, DOMÁCÍ PERNÍKOVÝ KNEDLÍK	**189 Kč**
GRILOVANÉ KUŘECÍ PRSO S HŘÍBKOVOU	
OMÁČKOU, SMAŽENÉ BRAMBOROVÉ	
HRANOLKY	**185 Kč**
DEZERT:	
DOMÁCÍ SYRNÍKY SE ZAKYSANOU SMETANOU	
A BRUSINKAMI	**95 Kč**
PŘEJEME VÁM DOBROU CHUŤ	

Obrázek 2.4: Rozdělování jména jídla na více řádků

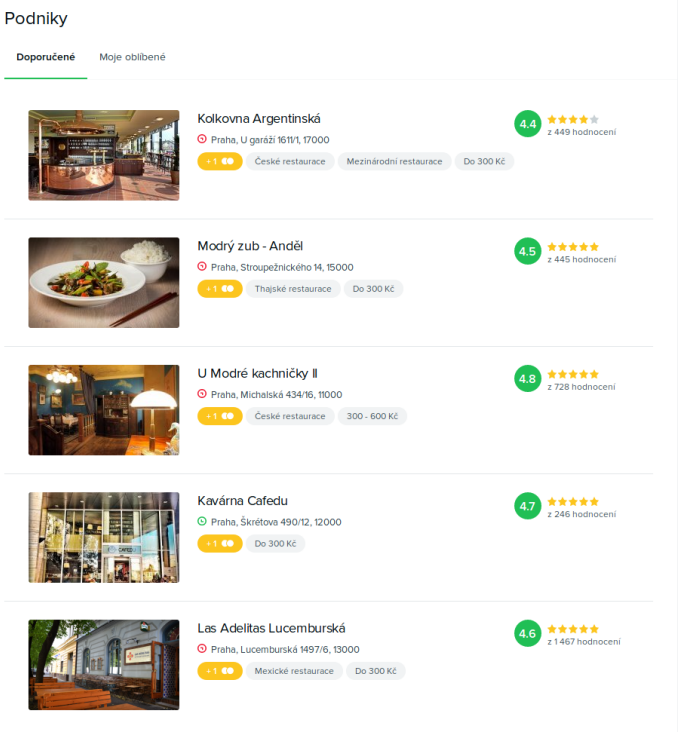
Obrázek 2.5: Restu.cz logo

v okolí dané restaurace.

2.4.2 Možnost využití dat a jejich kvalita

Tento portál obsahuje velké množství restaurací z Prahy, konkrétně jsem jich v sitemap nalezl něco málo přes 2000. Zdaleka ne všechny restaurace však mají denní menu aktualizované, resp. vyplněné. Bohužel ne všechna denní menu označují jen jídla, v datech se nachází celkem dost chyb. Pro ilustraci uvádím screenshoty z náhodně vybraných restaurací, viz obrázky 2.10, 2.11, 2.12.

MenuPraha.cz umožňuje data používat pro nekomerční účely [11], tudíž i přes občasné problémy v podobě a formátu, budu využívat jejich data pro získávání informací o denních menu.



Obrázek 2.6: Doporučování restaurací po přihlášení v Restu.cz

Obrázek 2.7: MenuPraha.cz - logo

2.5 Menicka.cz

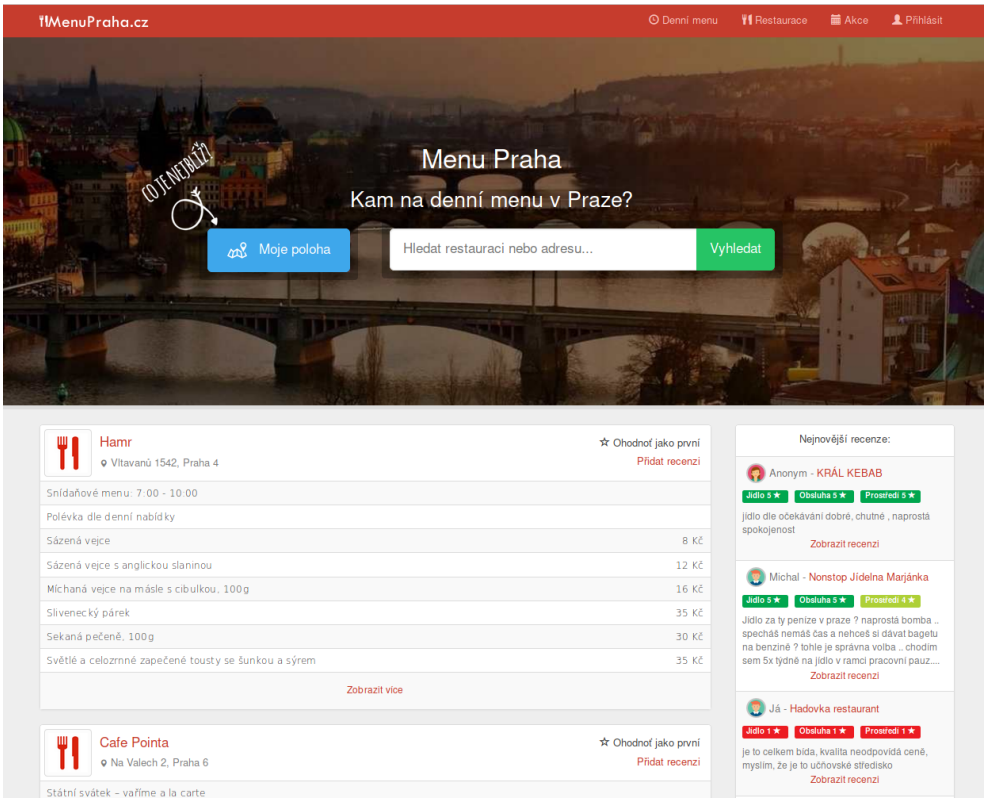
2.5.1 Vzhled a funkce

Další z českých služeb pro agregaci denních menu restaurací je služba Menicka.cz. Tato služba se zaměřuje jen na zobrazování denní nabídky. Kromě rozhraní pro uživatele, obsahuje možnost vkládání denní nabídky pro restaurace. Svou nabídkou pokrývá celou Českou republiku. Umožňuje ukládání restaurací do „oblíbených“ a zasílání denních menu na e-mail. Mimo nabídku denních menu nabízí i zobrazení jídelních lístků se stálou nabídkou restaurací.

Rozhraní je příjemně jednoduché, viz obrázky 2.14 a 2.15.

2.5.2 Možnost využití dat a jejich kvalita

Celkově je prostředí aplikace příjemné a prezentovaná data se zdají být i poměrně kvalitní – nenalezl jsem zdaleka tolik problémů, jako v datech z API



Obrázek 2.8: MenuPraha.cz - titulní strana

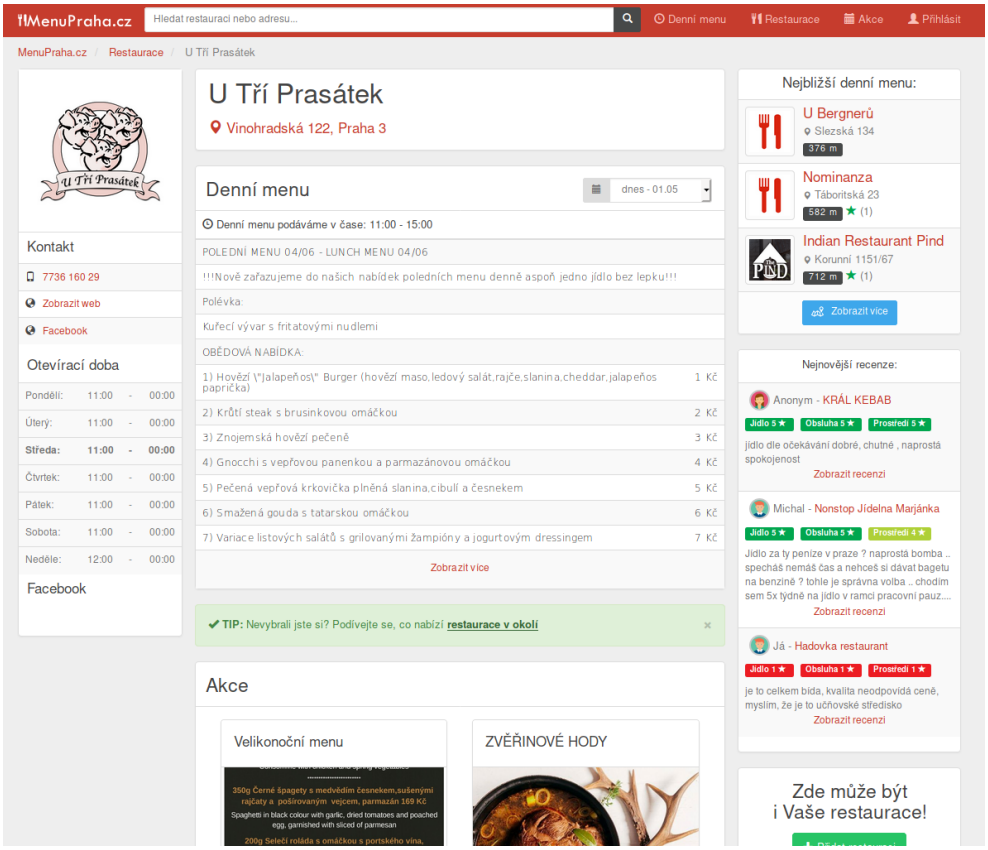
Zomato, případně v datech z MenuPraha.cz.

Bohužel se mi nepodařilo získat povolení používat data pro potřeby práce, proto s touto službou v implementaci počítat nebudu. Dle informací v patičce webu provozovatel zakazuje jakékoli použití obsahu bez předchozího písemného souhlasu.

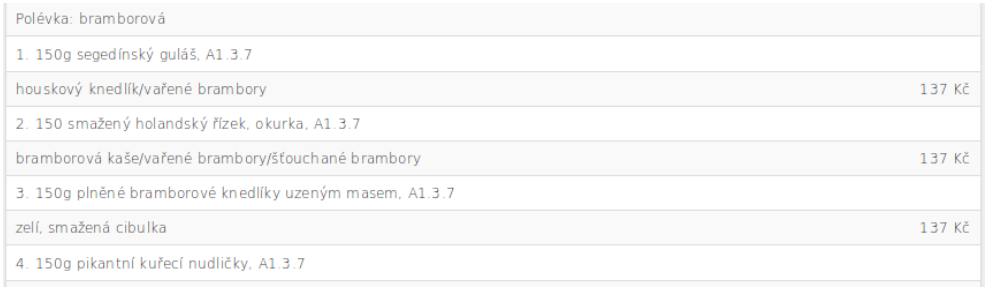
2.6 Menší projekty na doporučování jídel a jejich klasifikaci

Mimo fungující služby jsem na internetu našel ještě několik projektů zabývajících se rozpoznáváním, případně doporučováním jídel. GitHub má pro tuto kategorii projektů vlastní téma, dostupné je na URL adrese `https://github.com/topics/food-recommendation`. Projekty v této kategorii se zabývají několika aspekty doporučování případně klasifikací.

Všechny projekty však počítají buď s rozsáhlou databází jídel, případně pracují již s naučenými modely (v případě aplikací využívající neuronové sítě). Všechny tyto modely a neuronové sítě však počítají jen s jídly zadávanými



Obrázek 2.9: MenuPraha.cz - detail restaurace

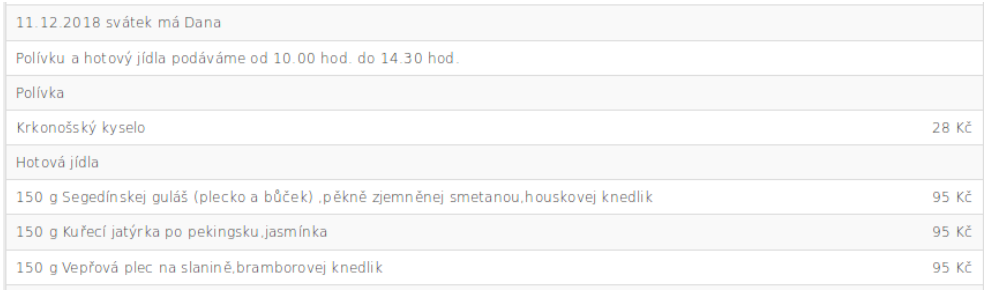


Obrázek 2.10: MenuPraha.cz - rozdělení jednoho jídla na dva řádky

v angličtině. V repozitářích jsou dostupné jen naučené modely, žádnou informaci o klasifikaci nebo analyzovaných datech se mi z toho vyčíst nepodařilo.

Mimo uvedené repozitáře na GitHubu jsem narazil ještě na článek popisující vývoj aplikace určené k doporučování jídla na základě algoritmu Apriori. Autoři v ní popsali způsob doporučování jídel v restauracích na základě předchozích dat [12]. Aplikace popisovaná v této krátké publikaci bude prav-

Obrázek 2.11: MenuPraha.cz - označení dne jako název jídla



Obrázek 2.12: MenuPraha.cz - informace o svátku, době podávání denního menu a hovorová čeština

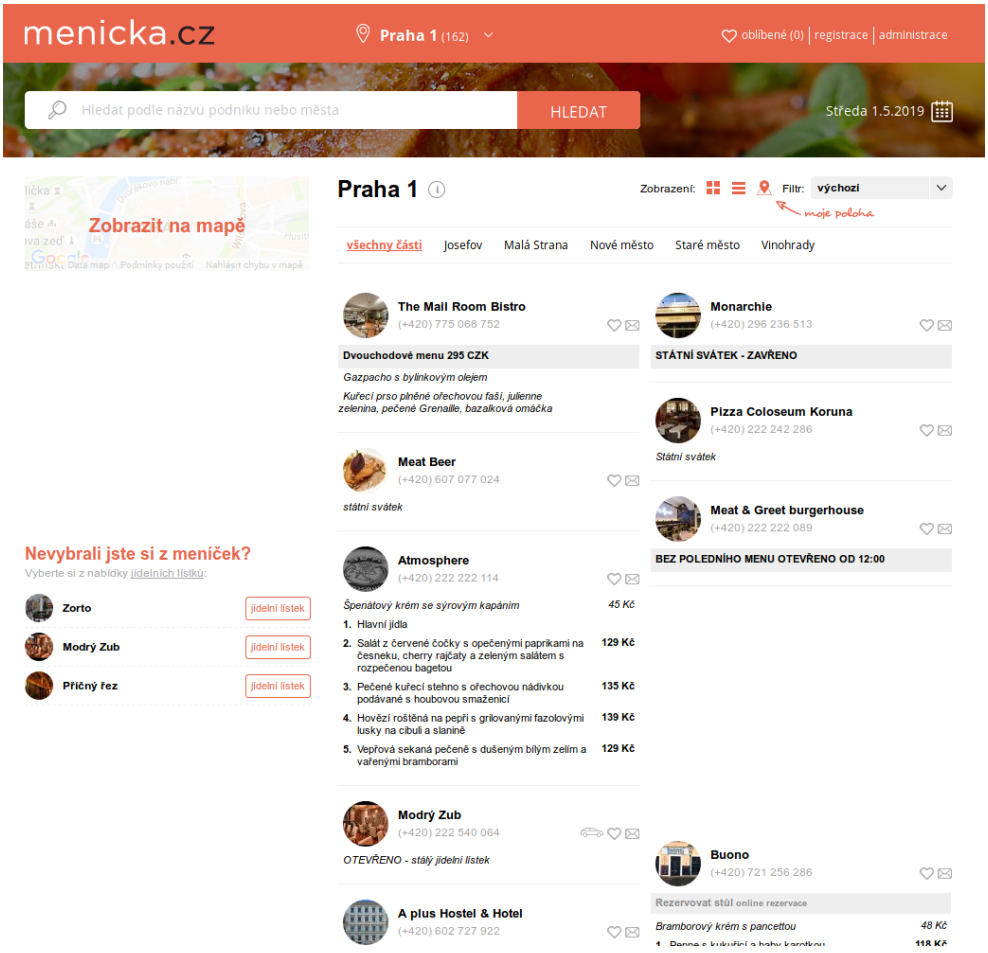
Obrázek 2.13: Menicka.cz - logo

děpodobně nějaká školní práce, nikde na internetu jsem k ní bližší informace nenašel, tudíž kvalitu doporučování nemohu posoudit.

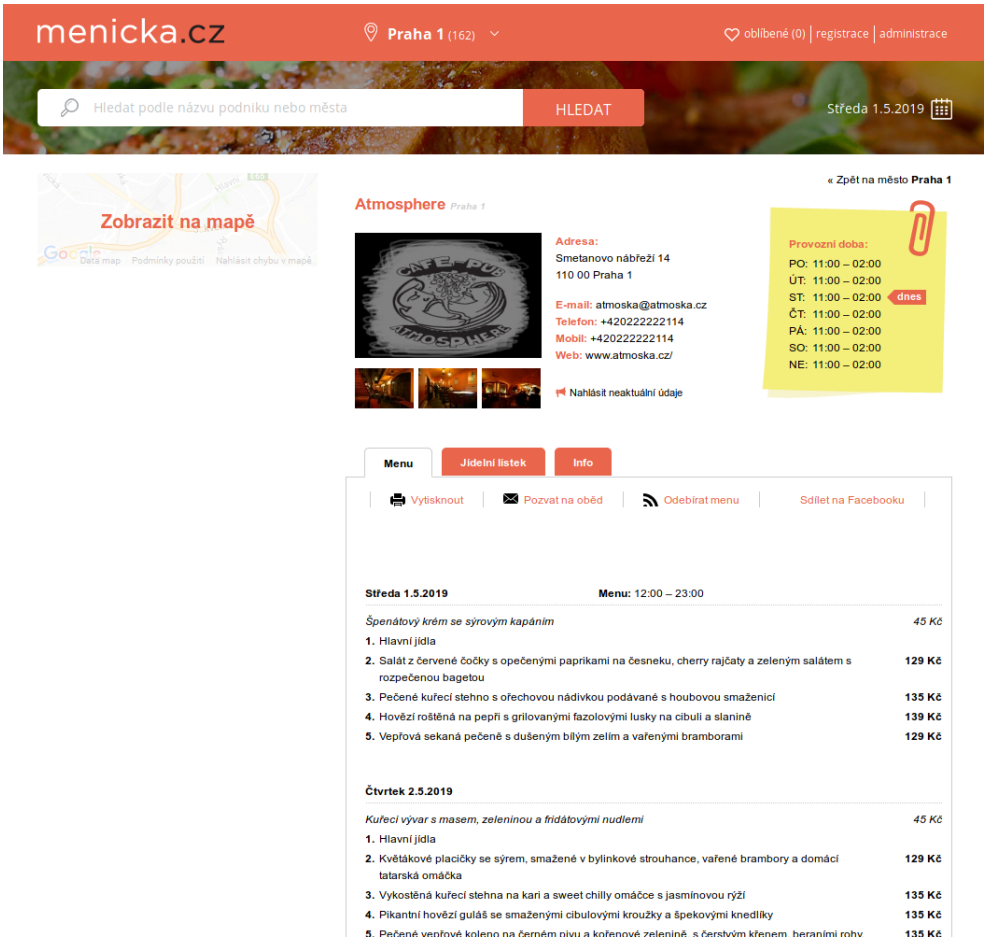
2.7 Závěr

V této kapitole byla provedena analýza existujících služeb, resp. služeb které nějakým způsobem pracují s denními menu restaurací. Mimo existující aplikace jsem prozkoumal i menší projekty, většinou výsledky hackathonů, případné naprogramované jako školní projekt. V žádné aplikaci ani menším projektu jsem nenašel žádný způsob doporučování jídel. Jediná analyzovaná služba, Restu.cz, poskytuje doporučení týkající se restaurací.

Zároveň jsem našel služby, ze kterých bude možné získávat informace o denních menu. API Zomato a obsah webu MenuPraha.cz pokrývá svými daty jen Prahu, nicméně pro ilustrační účely budou data dostačující.



Obrázek 2.14: Menicka.cz - titulní strana



Obrázek 2.15: Menicka.cz - detail restaurace

Kapitola 3

Analýza a návrh webové části aplikace

Všechny služby, které jsem při průzkumu nalezl se zabývají jen otázkou zobrazování denních menu, případně řeší prezentaci restaurace. Zaměřují se na hodnocení a řeší spíše marketingovou stránku.

V této části práce bych rád identifikoval požadavky uživatelů a navrhl aplikaci, která by umožňovala uživatelům získávat informace o denních menu.

3.1 Analýza potřeb a představ uživatelů

Pro účely získání požadavků a nápadů na funkčnost aplikace jsem oslovil celkem 5 lidí, tři z nich jsou kolegové z kanceláře, zbývající dva dotazovaní jsou přátelé mimo informatickou praxi. Společně jsme dali dohromady seznam požadavků, které by aplikace na zobrazování a doporučování denních menu měla mít. Představy jsem s nimi probíral formou rozhovoru, ve kterém jsem nastínil své představy ohledně funkcionality aplikace a poté jsem s dotazovanými řešil, jaká by byla jejich očekávání.

Hlavním a nejdůležitějším cílem je samozřejmě možnost jednoduše zobrazit denní nabídku vybraných restaurací. S tím souvisí i možnost volby, které restaurace chce uživatel vidět a které ho tolik nezajímají (např. kvůli vzdálenosti).

Mimo samotné zobrazení restaurací se většina z dotazovaných shodla na tom, že by aplikace určitě měla podporovat vyhledávání - a to jak v denní nabídce, tak i v seznamu restaurací. Tato funkce se hodí pro případ, kdy uživatel dostane chuť na něco konkrétního (vyloučíme-li možnost objednání minutkového jídla) a hledá, která restaurace dané jídlo nabízí.

Kolegové z kanceláře shodně uvedli, že by velkým přínosem byla možnost tvorby doporučení jako tip na společný oběd - pomocí seznamu přátel by

aplikace vybírala doporučení z těch restaurací, kde si pravděpodobně vyberou všichni (nebo alespoň značná část).

Při probírání funkčnosti jsem se snažil také zjistit, jakým způsobem by aplikace měla získávat od uživatele jeho preference. Na tuto věc se odpovědi celkem lišily - dva z dotazovaných uvedli, že by bylo dostatečné, kdyby aplikace umožňovala ruční nastavení těchto preferencí. Oproti tomu dvě odpovědi označily za nutnost nějakého automatického rozpoznávání chutí, ideálně bez velké námahy uživatele. Poslední z dotazovaných uvedl, že by aplikace měla podporovat oba dva přístupy.

Probírali jsme i způsob využívání aplikace, tedy z jakých zařízení budou uživatelé aplikaci pravděpodobně využívat nejvíce. Tři dotazovaní uvedli, že aplikace musí bezchybně fungovat především v mobilním prohlížeči, jednoduše z toho důvodu, že když stojí před oblíbenou restaurací (kde jim právě vyprodali jejich oblíbené jídlo), rádi by měli rychlý přístup k denní nabídce podniků v okolí. Zbylí dva uvedli, že tento požadavek není pro ně zase tolik důležitý, neboť si hledají vždy více možností, kam na oběd zajít.

3.1.1 Závěr

Z rozhovoru s dotazovanými přáteli jsem získal jiný vhled do potřeb a očekávání uživatelů, byť ve větší míře mi jen potvrdili mé vize. Během rozhovorů padlo samozřejmě i velké množství návrhů, myšlenek a zajímavých funkcí, které však přesahují rozsah práce, a proto budou uvedeny jen jako „nice to have požadavky“, případně v poslední kapitole jako nápady na rozšíření aplikace.

Na základě těchto dat se nyní pokusím sestavit seznam funkčních a nefunkčních požadavků, ze kterých budu v následující práci vycházet.

3.2 Funkční požadavky

V této části práce budou rozebrány požadavky, které musí aplikace splňovat po funkční stránce. Tyto požadavky se vyslovují přímo k samotným funkcím aplikace. V následujících bodech budou stručně popsány funkční požadavky seřazené podle důležitosti.

3.2.1 Zobrazení restaurací v okolí

Aplikace bude podporovat geolokační služby (pomocí prohlížeče) a bude zobrazovat restaurace v dostupném okolí cca 1km.

3.2.2 Přidání restaurace do oblíbených

Aplikace bude umožňovat přidávání restaurace do oblíbených, což ocení uživatel především v rámci rychlého a jednoduchého přístupu ke svému prefero-

vanému podniku. Bude pro něj tedy jednoduché získat informaci o aktuálním denním menu.

3.2.3 Doporučování jídel na základě preferencí uživatele

Aplikace bude automaticky klasifikovat denní nabídku restaurací a na základě zpětné vazby odhadne preference uživatelů. Získaná data pak budou využita pro individuální doporučování obsahu.

3.2.4 Možnost ručního nastavení preferencí

Uživatel bude mít v aplikaci možnost ručně si nastavit své vlastní preference dle přednastavených kategorií, aby měl přímou kontrolu nad obsahem doporučovaných menu.

3.2.5 Tvorba tipů na společný oběd s přáteli

Aplikace bude umožňovat vytvářet doporučení, která budou společná pro více uživatelů. Takto vygenerované doporučení bude sloužit jako tip na společný oběd.

3.2.6 Učení aplikace

Uživatel bude mít možnost přispět k lepší klasifikaci denních menu pomocí jednoduchého uživatelského rozhraní. To bude určeno k získávání dat spojených s klasifikací jídel. Data získaná od uživatelů se pak promítnou do klasifikace následující dny.

3.3 Nefunkční požadavky

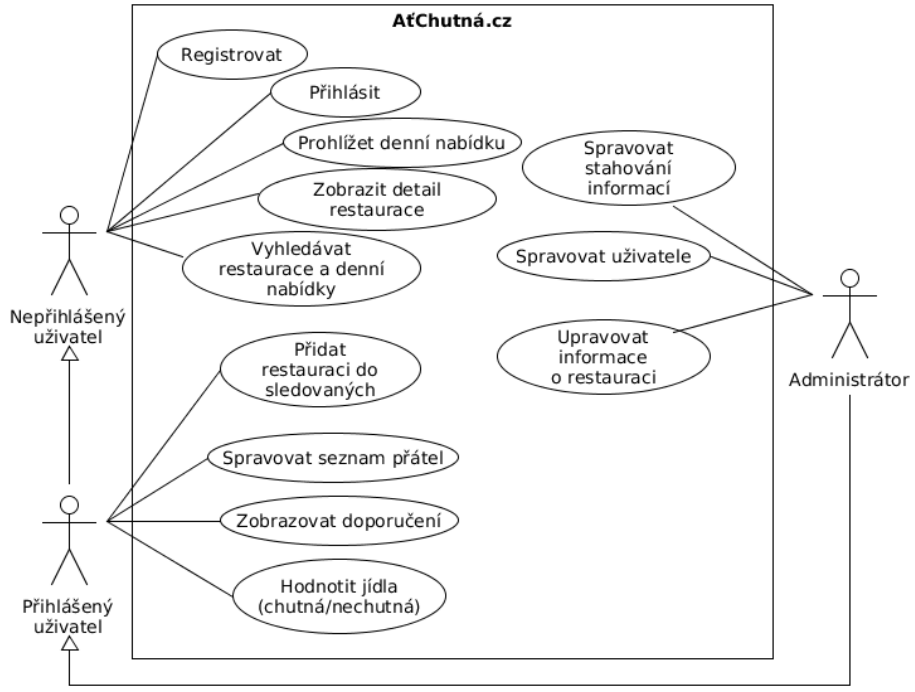
Nefunkční požadavky označují omezení kladené na systém. Tato omezení se přímo netýká funkčnosti aplikace nicméně významným způsobem ovlivňují komfort při používání výsledného softwarového produktu.

3.3.1 Přístupnost uživatelského rozhraní

Uživatelské rozhraní aplikace bude jednoduché na používání. Jednoduchost a intuitivnost používání bude posouzena po dokončení wireframů pomocí jednoduchého uživatelského testování.

3.3.2 Responzivní zobrazení

Aplikace se bude správně zobrazovat jak na monitoru počítače, tak i mobilním displeji.



Obrázek 3.1: Use case diagram

3.3.3 Výkonnost aplikace

Doba potřebná k načtení dat z API serveru (získání dat pomocí metody GET) bude vždy nižší než 0.5s.

3.4 Use case

Aplikace počítá se třemi rolemi, konkrétně s přihlášeným a nepřihlášeným uživatelem a administrátorem. Pro lepší představu o způsobech použití aplikace jsem připravil use case diagram, viz obrázek 3.1. V následující části textu popíšu jednotlivé role a vysvětlím méně zřejmé případy použití.

3.4.1 Uživatelské role

3.4.1.1 Nepřihlášený uživatel

Nepřihlášený uživatel je návštěvník webu, vidí jen seznam restaurací a denní menu. O tomto uživateli aplikace neukládá žádná data.

3.4.1.2 Přihlášený uživatel

Přihlášený uživatel je návštěvník, který se do aplikace zaregistroval. Tato role má možnost označovat denní menu podle svých preferencí a na základě toho získávat doporučení.

3.4.1.3 Administrátor

Uživatelská role „Administrátor“ neoznačuje osobu, která by mohla vkládat denní menu restaurací, jedná se o roli tzv. „super uživatele“. Tato role tedy slouží ke správě samotné aplikace a bude v celém systému zastoupena pravděpodobně jen jedním (nebo několika málo) uživateli.

3.4.2 Případy použití

V této části nebudu popisovat zřejmé aktivity (např. zobrazení denního menu restaurace), ale zaměřím se především na méně zřejmé případy používání.

3.4.2.1 Spravovat seznam přátel

Přihlášený uživatel má možnost ve svém profilu přidávat ostatní uživatele (kteří jsou také registrovanými) do seznamu přátel. Tento seznam bude sloužit k vytváření takových doporučení, aby uživatel mohl jít se svými přáteli na oběd.

3.4.2.2 Správa stahování

Administrátor může pozastavit či úplně zrušit stahování denní nabídky, např. v případě, že restaurace dlouhodobě tyto informace neposkytuje.

3.4.2.3 Správa uživatelů

Uživatelé se do aplikace registrují sami, nicméně občas je třeba řešit např. požadavky na smazání účtu. Počítám, že tento druh požadavků bude řešen individuálně, např. pomocí e-mailu.

3.4.2.4 Upravovat informace o restauraci

Informace o restauraci jsou staženy automaticky při inicializaci projektu, nicméně v případě jejich změny (např. změna URL adresy denní nabídky) aplikace toto nezaznamená. Proto je tato věc umožněna alespoň administrátorovi, který může editovat informace o dané restauraci ručně.

3.4.2.5 Hodnotit jídla (chutná/nechutná)

Každý přihlášený uživatel může ohodnotit jídlo podle svých chutí. Tím pádem zanese do systému informaci o preferencích, na základě kterých mu bude aplikace generovat doporučení. Mimo hodnocení chutná/nechutná je možné dané menu označit i volnou „Toto není jídlo“. Tato část bude podrobněji popsána v kapitole o zpracování dat.

3.5 Wireframe

Na základě vzhledu existujících služeb, dialogu s oslovenými lidmi a definovaných funkčních a nefunkčních požadavků jsem se pokusil sestavit jednoduchý wireframe, na kterém je zachycen princip fungování webové části aplikace. Pro jeho tvorbu jsem se rozhodl použít nástroj Balsamiq (`https://balsamiq.cloud`). Tím vznikly wireframy v podobě klikatelného PDF, které je dostupné na přiloženém médiu.

Takto vytvořený wireframe jsem se rozhodl otestovat na pěti přátelích. Cílem nebylo dokonale vyladit uživatelské prostředí (což by se s klikatelným PDF dělalo poměrně těžko), účelem testování bylo spíše odhalit hrubé nedostatky v návrhu a ujasnit si rozdělení webové aplikace do jednotlivých komponent.

3.6 Uživatelské testování

Scénáře pro otestování byly dva a vcelku jednoduché. Jeden se zaměřoval na jednoduchost získání informací o denním menu z oblíbené restaurace a ohodnocení jídla. Druhý se pak snažil odkrýt nedostatky v procesu ruční volby preferencí a zobrazení samotných doporučení.

Vzhledem k tomu, že testování probíhalo nad wireframem, velkou část tvořila diskuse o tom, co se stane když uživatel provede nějakou akci. Tento přístup není samozřejmě zcela korektní, nicméně pro získání prvotní zpětné vazby je, dle mého názoru, dostačující.

3.6.1 Scénář č. 1: Hodnocení jídla, tvorba klasifikace

V prvním scénáři jsem se snažil odhalit problémy, které by uživatel mohl pociťovat při procesu sestavování svých preferencí. Testeři měli za úkol nalézt svou oblíbenou restauraci a naznačit, jakým způsobem by hodnotili dané jídlo.

3.6.2 Scénář č. 2: Ruční ohodnocení, zobrazení doporučení

Druhý scénář se zaměřoval na ruční získání preferencí uživatele, jejich úpravu a následně zobrazení seznamu doporučení.

3.7 Uživatelské testování - výsledky

Uvedené scénáře jsem zvolil proto, neboť se jedná o klíčové průchody aplikací. Pokud uživatel nebude hodnotit jednotlivá jídla, případně pokud si nevytvoří ruční klasifikaci, nebude mít aplikace žádná data, podle kterých by byla schopna provádět další doporučování. Tím by se pro uživatele proměnila jen v pouhou aplikaci zaměřenou na zobrazování denní nabídky restaurací, kterých je již několik.

Pro získání zpětné vazby jsem oslovil pět lidí. Záměrně jsem vybral jinou skupinu, než se kterou jsem probíral funkcionalitu aplikace. Jednalo se o tři muže a dvě ženy. Všichni ve věku okolo 25 let.

Toto zjednodušené uživatelské testování odhalilo následující problémy (seřazeno dle závažnosti):

1. Terminologie
2. Chaotická stránka profilu
3. Zobrazení přátel, se kterými lze jít do dané restaurace
4. Seznam přátel
5. Umístění seznamu doporučení

Jako největší problém byla označena terminologie používaná ve wireframu. Pro všechny z dotazovaných přišlo slovo „Profil“ jako nejednoznačné. Všichni pod tímto odkazem očekávali spíš nastavení jména a hesla, ne však seznam oblíbených restaurací a už vůbec ne seznam doporučených jídel. Toto shodně uvedli 4 z 5 dotazovaných.

Druhým největším problémem byla samotná stránka, kterou jsem ve wireframu označil právě jako výše zmíněný profil. Pokud odhlédneme od toho, že zde většina očekávala jiný obsah, 4 z 5 testerům přišla tato stránka nepřehledná. Navíc správa přátel jim připadala zbytečně „utopená“ až po seznamu oblíbených restaurací.

Jako třetí největší problém testeři označili nemožnost zjistit, s kým by mohli jít na doporučený oběd. Takto se vyjádřili 3 z 5 testerů.

Z wireframů testeři příliš nepochopili, že doporučení budou vytvářena pro všechny lidi v seznamu přátel. Očekávali, že si budou moci vybrat, s jakými přáteli chtějí získávat tipy na společný oběd.

Jako čtvrtý, nejméně závažný problém, byl označen blok s doporučenými jídly. Dva testeři označili tento blok jako „vytržený z kontextu“, spíše ho očekávali na titulní straně. Toto očekávání bylo i podpořeno faktem, že na titulní straně aplikace by se měly zobrazovat restaurace v okolí.

Po průchodu scénáři následovala nestrukturovaná diskuze, ve které vzniklo několik zajímavých podnětů jak pro samotné doporučování, tak i „nice to have“ funkce.

3.7.1 Závěr

Díky zjednodušenému uživatelskému testování se mi podařilo získat cennou zpětnou vazbu od lidí, kteří tvoří potenciální cílovou skupinu uživatelů. Tuto zpětnou vazbu se budu snažit v maximální možné míře využít při vývoji webové aplikace.

Kapitola 4

Analýza a návrh doporučovacího systému

V minulé kapitole byla analyzována webová aplikace, tedy rozhraní sloužící pro koncové uživatele. V této části se pokusím rozebrat požadavky potřebné ke klasifikaci denních menu a jejich následnému doporučování. Zároveň bych zde rád popsal několik problémů, se kterými jsem se během procesu implementace setkal.

4.1 Požadavky

Před samotnou analýzou řešení uvedu základní požadavky a omezení kladená na systém.

4.1.1 Modularita přidávání zdrojů dat

Vzhledem k tomu, že jsem takto schopný získat informace o denních menu jen z relativně malého počtu restaurací (data ze Zomato a MenuPraha.cz pokrývají pouze Prahu), je potřeba navrhnout část aplikace určenou k získávání dat, pokud možno, co nejvíce modulárně. Přidání nového zdroje dat (např. menu vystavené na webu restaurace) bude tedy probíhat ve vytvoření potřebného parseru [1] a jeho zaregistrování do databáze.

4.1.2 Zpracování dat a jejich filtrování

Při analýze podobných služeb jsem zjistil, že data vyžadují celkem často nějakou formu zpracování. Ať už se jedná o extrakci cen z textu, sloučení více menu do jednoho nebo odstranění informací, které se sice vztahují k denní

[1] Třída sloužící pro extrakci relevantních informací z informačního zdroje, např. webové stránky.

nabídce, nicméně neoznačují žádná jídla. Tyto problémy je třeba detekovat a pokud možno i odstranit.

4.1.3 Klasifikace založená na jménu jídla

Aplikace bude klasifikovat data nejen na základě informací obsažených v názvu jídel ale i na základě ingrediencí získaných z receptů s podobným jménem. K tomuto účelu bude potřeba připravit získávání a vyhledávání těchto informací. Klasifikace pomocí ingrediencí na základě jmen receptů je nezbytná i pro detailnější popis jídel typu „katův šleh“, „boršč“ a jiné názvy, ze kterých není zřejmé, co mohou obsahovat.

4.1.4 Rozdělení na základě typických kuchyní

Mimo statickou klasifikaci popsanou v předešlé části bude aplikace podporovat i dělení jídel podle tradičních kuchyní (např. japonská, americká, česká, ...). K tomuto účelu bude sloužit klasifikace získaná analýzou jména jídla.

4.1.5 Doporučování na základě lokality uživatele

Na základě explicitní zpětné vazby od uživatele bude aplikace rozpoznávat místa, kam uživatel nejčastěji chodí na oběd a bude mu doporučovat jídla v okolí těchto míst. Ideálně bez nutnosti ručního zadávání polohy.

4.1.6 Podpora více způsobů doporučování

Mimo doporučování na základě lokality bude aplikace podporovat i jiné způsoby doporučování, např. bude poskytovat tipy na společný oběd s přáteli. Stejně jako u přidávání více zdrojů dat bude aplikace podporovat podobně modulární způsob pro přidání nového doporučovacího algoritmu.

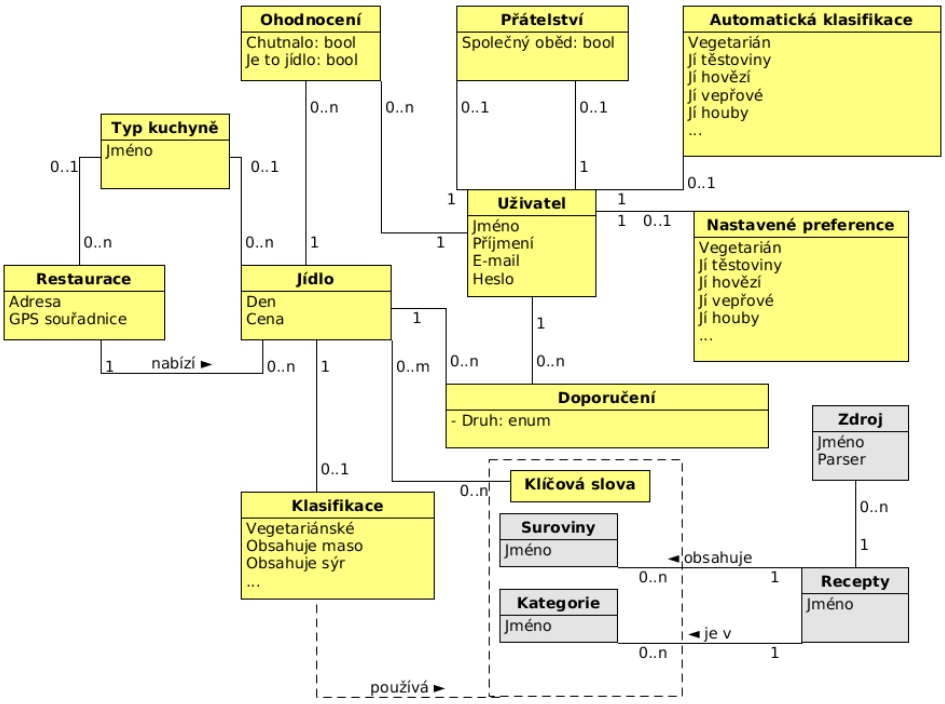
4.1.7 Rozpoznávání „nejídel“

Aby aplikace byla schopná zobrazovat relevantní obsah, je nutné ji naučit rozpoznávat záznamy v denních menu, které neoznačují žádné jídlo. Tímto myslím např. informace o rozvozu, kategorizaci jídel v denní nabídce nebo třeba informaci o tom, kdo v daný den slaví svátek.

Tato část bude vyžadovat zpětnou vazbu od uživatele.

4.2 Volba zdroje dat pro klasifikaci

Jak bylo popsáno v části věnující se požadavkům na backend aplikace, potřebuji identifikovat zdroj receptů, ze kterých mohu čerpat informace o použitých ingrediencích. Zároveň potřebuji najít ideálně takový zdroj, který obsahuje seznam receptů nějak jednoduše dostupný (např. v sitemap). Tento požadavek



Obrázek 4.1: Doménový model

si kladu jednoduše proto, aby bylo možné recepty stáhnout, pokud možno co nejjednodušší cestou, a nebylo třeba procházet např. výsledky vyhledávání.

Na základě těchto požadavků jsem vybral portály `https://recepty.cz` a `https://toprecepty.cz`. Oba poskytující poměrně rozsáhlý seznam receptů. Na detailu receptu je k dispozici kategorizace receptu a zároveň tyto portály nezakazují použití dat pro nekomerční účely (zakazují ovšem jejich publikování [13]).

4.3 Lemmatizace a čeština

Pro snížení dimenzionality klasifikovaných dat bude aplikace potřebovat převádět slova na jednotný tvar.

4.4 Doménový model

V diagramu 4.1 jsem se snažil nastínit vztahy mezi entitami. Tento doménový model jsem následně použil pro návrh vztahů mezi modely aplikace.

4.5 Proces klasifikace

V následujícím textu se pokusím navrhnout proces získávání informací o jídle a jeho klasifikaci.

4.5.1 Získání statických klíčových slov

Prvním krokem bude rozdělení jména jídla na klíčová slova a odstranění spojek. Představa je taková, že z názvu jídla „Vepřová krkovice s bramborovou kaší, zeleninou a salátem“ vzniknou klíčová slova „Vepřová krkovice“, „bramborovou kaší“, „zeleninou“, „salátem“, resp. jejich kanonické formy.

4.5.2 Rozšíření klíčových slov o informace z podobných receptů

Ve druhém kroku bude provedeno vyhledávání v receptech a získání informací o pravděpodobném složení klasifikovaného jídla. Idea je, že v případě klasifikace polévky „boršč“ dojde k vyhledání receptů a zjištění, že „boršč“ obsahuje např. červenou řepu, uzeninu nebo třeba zakysanou smetanu.

4.5.3 Přidání klíčových slov podle částí názvu jídel

V této fázi se spustí podobné vyhledávání jako v předchozí části, jen se ale aplikuje na složená klíčová slova (klíčová slova obsahující více slov). Díky tomu lze zjistit, jaké ingredience obsahuje např. bramborová kaše. V této fázi bude nutné vybírat relevantnější výsledky, aby nedošlo ke zkreslení informací o ingrediencích vlivem např. jiné přílohy.

4.5.4 Zobecnění klíčových slov

Poslední fází tagování jídel je zobecnění získaných klíčových slov. Díky tomuto kroku bude možné například přiřadit klíčové slovo „těstoviny“, pokud v seznamu klíčových slov bude „tortellini“.

4.5.5 Finální klasifikace

Na základě klíčových slov bude sestavena finální klasifikace jídla. Pro tento účel jsem se pokusil sestavit seznam surovin. Klasifikace bude jídla rozdělovat podle následujících kategorií:

- Houby
- Tofu
- Vegetariánské
- Vepřové maso

- Drůbeží maso
- Hovězí maso
- Maso obecně
- Divočina
- Ryby
- Plody moře
- Těstoviny
- Sýry
- Houby
- Indická kuchyně
- Vietnamská kuchyně
- Mexická kuchyně
- Česká kuchyně
- Americká kuchyně
- Japonská kuchyně
- Čínská kuchyně
- Italská kuchyně
- Polévky
- Sladká jídla

4.5.5.1 Statická klasifikace

Jednou částí klasifikace bude prosté ukládání informace označující, zda se v daném jídle vyskytuje daná surovina (např. tofu). K tomuto bude využíván seznam klíčových slov. Klasifikace v tomto případě bude jen binární.

4.5.5.2 Klasifikace na základě podobnosti se skupinou jiných jídel

Tento typ klasifikace bude určen k odhadování, do jaké kuchyně klasifikované jídlo patří. Pro tento účel bude nutné vytvořit seznam receptů patřících do uvedené kategorie.

4.5.6 Zpětná vazba od uživatelů

Aby aplikace dokázala poznat preference uživatele, je nutné od něj tyto informace získat. Pro tento účel bude každé jídlo hodnoceno jednou z následujících možností:

- Chutnalo mi
- Dal/a bych si
- Nechutnalo mi
- Nedal/a bych si
- Toto není jídlo

Rozdělení podle toho, jestli dané jídlo uživatel měl či nikoli, jsem zvolil z toho důvodu, aby bylo možné detekovat oblast, ve které se uživatel nachází, resp. kam nejčastěji chodí na oběd. Poslední možnost zase slouží k tomu, aby uživatelé aplikaci naučili, co jídlo je a co není (kvůli filtrování vstupních dat).

4.6 Závěr

V této části byly popsány požadavky kladené na klasifikaci a doporučování, následně jsem se pokusil navrhnout proces, jakým budou získávány informace o jídlech a jak bude aplikace s těmito informacemi dále pracovat. Podařilo se mi také najít zdroj dat, která budu používat pro detailnější rozpoznávání jídel.

Na základě popsaných požadavků jsem se rozhodl výsledný doporučovací systém implementovat jako „Content based“. Přijde mi, že se tento model lépe hodí pro doporučování denních menu.

Kapitola 5

Výběr technologií

Na základě požadavků a omezení definovaných v předchozích dvou kapitolách zde krátce popíši, jaké technologie jsem vybral pro finální implementaci.

5.1 Serverová část (zpracování dat)

Pro tuto část jsem se rozhodl využít Python. Jednak proto, že jsem chtěl díky této práci získat zkušenosti s tímto jazykem a jednak proto, že se jedná o často využívaný jazyk v oblasti datové analýzy [14] [15] [16].

Mimo samotný jazyk jsem řešil i framework, který by dokázal zjednodušit práci. Rozhodoval jsem se mezi minimalistickým frameworkem Flask a robustním Djangem. Vzhledem k tomu, že jsem nikdy v minulosti v Pythonu neprogramoval, zvolil jsem nakonec Django. A to hlavně proto, že umožňuje tvorbu jak webové aplikace, tak i tzv. „management commands“, což jsou skripty, které lze volat z příkazové řádky na serveru, nicméně zůstávají integrovány do aplikace [17]. Navíc poskytuje velmi robustní ORM [2] [18].

5.2 REST API

Pro vývoj webové části aplikace, resp. jejího backendu (API) jsem vybíral mezi dvěma jazyky – Pythonem a Node.js.

Node.js, jakožto javascript interpretovaný na serveru, jsem původně chtěl vybrat kvůli tomu, že samotná uživatelská část bude psána v javascriptu, tím pádem by webová část celé aplikace byla řešena v jedné technologii.

Nakonec ale vzhledem k rozsahu aplikace, která bude zodpovědná za komunikaci mezi backendem a webovou aplikací, jsem se rozhodl i API napsat v Pythonu, resp. frameworku Django. Nebude tak nutné duplikovat určité části business logiky, bude možné využít existující modely a bude tak možné mít napsaný kompletně celý backend v jedné technologii.

[2] Object-relation mapping

5.3 Uživatelské rozhraní webové aplikace

Frontend webové aplikace bych chtěl vytvořit jako tzv. single page aplikaci. Tedy webovou aplikaci, která funguje v prohlížeči uživatele a mezi jednotlivými interakcemi nevyžaduje znovunačítání stránky. Tím poskytuje lepší komfort při používání.

Pro účely své práce jsem volil mezi frameworky Vue.js a React. Oba dva fungují na podobném principu – komponentově orientované architektuře.

5.3.1 Vue.js

- URL adresa: `https://vuejs.org/`

Vue.js je javascriptová knihovna určená pro vývoj uživatelského rozhraní. Původní autor (Evan You) zveřejnil první verzi frameworku v roce 2014, jedná se o relativně mladý projekt, který ale i tak získal na popularitě [19] [20]. Tato popularita navíc evidentně stále stoupá [21].

Vue.js umožňuje jednoduché začlenění komponent do „ne-Vue.js“ aplikace. Jako šablonovací jazyk využívá standardní HTML, je tak velmi jednoduché začít s vývojem, i když člověk nemá žádné předchozí zkušenosti.

Vue.js je používáno řadou velkých společností, např. Adobe, Netflix, nebo třeba WizzAir [22].

5.3.2 React

- URL adresa: `https://reactjs.org/`

React je knihovna z dílny Facebooku [23], nicméně o její chod se stará i komunita open source vývojářů. React je vydaný pod licencí MIT. Tento framework je určený pro komponentově orientovaný vývoj uživatelského rozhraní, což znamená, že k využití jeho potenciálu je třeba využít i jiné knihovny z jeho ekosystému (např. router, správu stavů atp.).

React používá „Virtual DOM“, díky tomu dokáže efektivně překreslovat jen ty komponenty, které se reálně změnily, nemusí tak překreslovat vždy celý DOM [24]. Jednotlivé komponenty se píší v jazyce zvaném JSX [3]. Jedná se o rozšíření syntaxe javascriptu, ne nepodobné standardnímu HTML.

React je mimo Facebook využíván v celé řadě aplikací, např. Netflix, Khan Academy, New York Times a mnoho jiných [25] [26].

5.3.3 Závěr

Vzhledem k faktu, že jsem s žádným podobným frameworkem nikdy nepracoval, zkusil jsem si vytvořit jednoduchou aplikaci v obou frameworcích a

[3] Javascript XML

rozhodoval jsem se víceméně subjektivně. Nakonec jsem se rozhodl frontend webové aplikace napsat ve frameworku Vue.js, jelikož mi přišel jednodušší na pochopení i na následný vývoj.

V případě, kdy bych plánoval aplikaci zmigrovat např. i do podoby mobilní aplikace, zvolil bych asi spíše React. Toto však v plánu nemám.

5.4 Databáze

Výběr nevhodného typu databáze může mít za následek např. špatný výkon aplikace při zátěži, případně nekonzistenci dat. Z tohoto důvodu jsem věnoval volbě databáze poměrně hodně času. Volil jsem mezi NoSQL databází MongoDB a zástupcem ORDBMS - PostgreSQL.

5.4.1 MongoDB

- URL adresa: `https://www.mongodb.com/`

MongoDB je open source NoSQL, dokumentově orientovaná databáze. Distribuovaná je pod licencemi GNU GPL a Apache. Výhoda tohoto typu databází spočívá v lepší podpoře replikace (oproti tradičním relačním databázím).

MongoDB je využíváno např. firmami SEGA, Coinbase, nebo třeba Adobe [27].

5.4.2 PostgreSQL

- URL adresa: `https://www.postgresql.org/`

PostgreSQL představuje typického zástupce objektově-relačních databází. Databáze je distribuována pod open source licencí MIT. První verze byla vydána již v roce 1996, jedná se tedy o léty prověřenou technologii, která ale zároveň nabízí velice zajímavé funkce (např. podpora snapshotů databáze - MVCC model) (dodat zdroj). Podporuje velké množství datových typů (geometrické tvary, XML včetně podpory Xpath dotazování, JSON, ...) [28].

PostgreSQL je využívána celou řadou velkých společností, např. Spotify, Instagram, nebo třeba Netflix [29].

5.4.3 Závěr

Po velmi dlouhé úvaze a množství přečtených blogů jsem se nakonec rozhodl využít PostgreSQL databázi. Jedním z hlavních faktorů, proč jsem zvolil právě RDBMS[4] oproti NoSQL byl fakt, že s relačními databázemi mám větší

[4]Relation Database Management System

zkušenosti a tím pádem bude pro mne snazší odhalit problémy při implementaci, případně jim předcházet. Druhým důvodem byl fakt, že aplikace nebude zpracovávat tak velké množství dat, aby byl naplno využit potenciál distribuovaných databází.

5.5 Lemmatizace českých výrazů

Vzhledem k faktu, že aplikace počítá s daty v českém jazyce, je třeba vyřešit problém s převodem slov do nějaké jednotné podoby (tokenizace, lemmatizace[5], stemming[6]). V případě angličtiny se tento problém dá řešit pomocí Python knihovny NTLK, která obsahuje poměrně rozsáhlé možnosti v oblasti zpracování jazyka. Pro češtinu bohužel korpus dat chybí a využít tedy nelze.

Po vyloučení NLTK knihovny jsem zkoušel nástroj Majka a analyzátor češtiny pro Elasticsearch.

5.5.1 Majka

- URL adresa: `https://nlp.fi.muni.cz/czech-morphology-analyser/`

Tento malý program slouží k lemmatizaci slov a určení gramatické kategorie vstupního slova.

Majka je novější verzí programu Ajka, opravuje některé problémy svého předchůdce a zároveň přidává novou funkcionalitu [30].

Bohužel jsem ale musel nakonec tento nástroj ze své práce vyloučit, jednak proto, že neumí pracovat s překlepy (které se stávají, neboť denní menu jsou zadávána lidmi), a jednak proto, že nezná ani zdaleka všechna slova používaná v názvech jídel.

5.5.2 Elasticsearch

- URL adresa: `https://www.elastic.co/`

Elasticsearch je primárně engine určený k fulltextovému vyhledávání [31]. Mimo to však nabízí možnosti pro analýzu textu v několika jazycích, mimo jiné i v češtině [32]. Navíc umí spolupracovat i se slovníkem Hunspell, díky kterému je možné analýzu českého textu zpřesnit [33].

Vzhledem k tomu, že lemmatizace je důležitá část při klasifikaci a Elasticsearch tuto funkci zvládá poměrně dobře, rozhodl jsem se ho ve své práci využít.

Využití vyhledávácího enginu Elasticsearch by bylo více než vhodné i k samotnému vyhledávání podobných receptů a jejich ingrediencí a k vyhledávání na webu.

[5] Převedení slova na základní tvar

[6] Nalezení kořene slova

5.6 Serverové technologie

Pro nasazení aplikace na produkční server bude nutné zajistit i další podpůrné služby. Jedná se především o webserver. Samotná volba webserveru však nijak neovlivňuje implementaci a je možné ho kdykoli vyměnit za jiný.

Mimo samotný webserver je vhodné mít na serveru nainstalovaný i Docker a `docker-compose`. Díky tomu bude proces nasazení a aktualizací jednodušší. Nutnost to ale není, aplikace půjde zprovoznit i bez kontejnerizace, jen se bude jednat o náročnější proces.

Kapitola 6

Implementace serverové části

6.1 Databázový model

Finální podoba databáze je určena frameworkem Django, resp. jeho ORM. To si vytváří strukturu automaticky na základě modelů definovaných programátorem. Z toho důvodu bylo zbytečné databázi navrhovat zvlášť, celá struktura se vytvořila automaticky po vytvoření patřičných modelů.

Samotná struktura modelů vychází z doménového modelu, což je další důvod, proč do textu nevkládám grafické znázornění vztahů v databázi. Jednak jména tabulek jsou generovaná samotným Djangem a tím pádem nejsou vždy zcela čitelná, a jednak by tento diagram byl víceméně duplicitní. Pro kompletnost je ale finální model přiložen v přílohách. Zde se nachází ve formátu vhodném pro nástroj pgModeler (`https://pgmodeler.io/`).

6.2 Architektura aplikace

Při návrhu architektury budu vycházet z možností zvoleného frameworku Django. Tento framework obsahuje podporu pro tzv. „aplikace“. Tato aplikace označuje python balíček obsahující sadu vzájemně souvisejících funkcí [34].

Celý backend se skládá z pěti Django aplikací: API, získávání dat, klasifikace, doporučování a restaurace. Tyto aplikace jsou detailně popsány na následujících stránkách.

6.3 API

Tato část se stará o komunikaci aplikace s uživatelskou částí aplikace. S tou komunikuje pomocí REST API, které je rozděleno na dvě části - věřejné a neveřejné.

Pro jednodušší podporu rozšiřování funkcionality API jsem se rozhodl všechny API endpointy verzovat. To znamená, že všechny endpointy jsou prefixovány jako `v1`.

6.3.1 Veřejné API

Funkcionalita veřejné části API je poměrně jednoduchá. Možnosti použití odpovídají případům použití popsaných v části Use case.

6.3.2 API pro přihlášené

Do neveřejné části je vyžadováno přihlášení a poskytuje data týkající se konkrétního uživatele.

Autentizace probíhá pomocí tokenu, který uživatel získá při přihlášení. Veškerou práci spojenou s ověřením uživatele řeší `djangorestframework`.

6.4 Získávání dat

Tato část je ve zdrojových kódech označena jako „crawler“. Řeší stahování denních menu a receptů se surovinami ze zdrojů Zomato.com a MenuPraha.cz. Zároveň se zde řeší filtrace dat a jejich uložení do databáze.

6.4.1 Získávání dat o denních menu

Část aplikace obstarávající stahování denních menu má dvě části. První z nich je samotné získání dat ze zdroje, druhým je pak základní filtrace a transformace dat.

6.4.1.1 Stahování

Konfigurace stahování se ukládají v modelu `RestaurantScraperConfig` (tabulka `crawler_restaurantscraperconfig`). Ta obsahuje informaci o tom, jaká třída je zodpovědná za stahování dat z uvedeného zdroje (scraper), parametry potřebné pro daný scraper (např. URL adresa detailu denní nabídky, případně ID restaurace v Zomato API). Dále obsahuje datum, kdy se má denní nabídka znovu navštívit a pokusit stáhnout, datum posledního úspěšného stažení a interval, za jak dlouhou dobu se má aplikace pokusit toto denní menu opět navštívit. Důvod, proč nastavení vyžaduje informaci o tom, za kolik dní se má k danému zdroji vrátit, je popsán v části Deaktivace restaurací bez denních menu.

Celé stahování je obsluhováno skriptem `download_daily_menu`, který si načte seznam restaurací, jejichž menu má být zpracováno, stáhne data, provede filtraci a uloží do databáze. Zároveň naplánuje další stažení na následující den.

6.4.1.2 Filtrace dat

Jak jsem psal v předchozím textu, data nejsou vždy kvalitní. Proto je třeba udělat s nimi alespoň pár základních operací.

Filtrace denních menu je řešená ve třídě `DailyMenuCleaner` (Python modul `crawler.data_cleaners`). V konstruktoru této třídy je definován seznam filtrů, které se mají aplikovat na stažená data. Samotný filtr je jednoduchá třída, která implementuje metodu `filter`, ve které projde všechna denní menu za den a provede s nimi nějakou transformaci, jako příklad uvedu filtr na odstranění duplicitních názvů jídel:

```
class RemoveDuplicates(BaseFilter):
    def filter(self, menus):
        existing = set()
        filtered = []
        for menu in menus:
            if menu['name'] not in existing:
                filtered.append(menu)
                existing.add(menu['name'])
        return filtered
```

Rozšíření filtrování dat je tedy snadné, stačí napsat patřičný filtr a vložit ho do seznamu filtrů v této třídě.

V aplikaci jsou použity následující filtry:

- Filtr na minimální délku slova
- Extrakce ceny
- Spojení rozděleného názvu jídla
- Filtr „nejídel“
- Odstranění mezer a jiných nechtěných znaků z konce názvu
- Odstranění opakujících se znaků (např. pomlčka pro vytvoření horizontální čáry)
- Odstranění duplicitních názvů jídel

Při filtraci se odstraňuje ze jména jídla cena a ukládá se do samostatného sloupce v databázi. Aktuálně slouží jen pro informativní účely.

Filtry pro odstranění „nejídel“ a pro spojování jmen jsem detailněji popsal v části Problémy a jejich řešení.

Po proběhnutí všech kroků filtrace jsem musel přistoupit ještě k jednomu omezení. Určitá restaurace v Praze totiž na portál MenuPraha.cz zveřejňuje denní menu v poměrně nestandardní podobě - jejich nabídka běžně čítá cca

3000 řádků (což odpovídá necelé polovině všech stahovaných denních nabídek). Z tohoto důvodu jsem omezil počet nabízených jídel na 50.

Po skončení filtrace jsou jídla uložena do databáze.

6.4.1.3 Deaktivace restaurací bez denních menu

Seznam restaurací a jejich informace nejsou do systému zadávány ručně (ačkoli to aplikace umožňuje). Informace o restauracích se získávají pomocí výsledků vyhledávání (v případně Zomato) nebo z URL adres uvedených v sitemap služby MenuPraha.cz. V případě restaurací ze Zomato jsou výsledky poměrně relevantní, vyhledávání se provádí přímo v seznamu restaurací, který je označen jako „denní nabídka".

O něco složitější je situace s restauracemi z MenuPraha.cz. Tento portál obsahuje okolo 3200 restaurací, z nichž je ovšem velké procento neaktivní (nezveřejňují denní nabídku, případně MenuPraha nevyužívají). Proto je velmi neefektivní procházet a stahovat vždy všechny informace.

Kvůli tomu obsahuje konfigurační tabulka sloupec `interval`. Ten označuje, za kolik dní se má denní nabídka restaurace navštívit znovu. Pokud nedojde ke stažení žádných dat, vynásobí se tato hodnota číslem 2 (pokud není víkend, ty se ignorují). To znamená, že další pokus o stažení dat bude proveden za 2 dny. V případě, že ani za dva dny restaurace nemá vypsané denní menu, poslední pokus se naplánuje za další 4 dny. Pokud selže i toto stažení, je restaurace označena jako neaktivní a v následujícím stahování se s ní již nepočítá. Její povolení je možné udělat z administrace ručně.

6.4.1.4 Přidání nového zdroje dat

Nový zdroj dat o denních menu může být nějaký jiný agregátor, webová stránka restaurace nebo jakýkoli jiný zdroj, který obsahuje potřebná data.

První krok pro přidání nového zdroje dat je vytvoření třídy pro získávání dat. Tuto třídu budu označovat jako „scraper". Požadavky na scraper jsou jen dva - musí implementovat metodu `get_menu()` a musí být umístěný v adresáři `crawler/scrapers/daily_menu`. Takový scraper může vypadat třeba následovně:

```
class ExampleScraper:
    def __init__(self, url):
        super().__init__()
        self.url = url
        self.soup = BeautifulSoupWrapper(self.url)

    def get_name(self):
        return "..."

    def get_price(self):
```

```
        return "..."

    def get_menu(self):
        menu = self.soup.select('.menu tr')

        today = date.today().isoformat()
        parsed_menus = {
            today: []
        }

        for daily_menu in menu:
            menu_name = self.get_name(daily_menu)
            price = self.get_price(daily_menu)

            parsed_menus[today].append({
                "name": menu_name,
                "price": price
            })

        return parsed_menus
```

Po vytvoření této třídy je třeba přidat zdroj do fronty ke stahování. K tomu je nutné vytvořit záznam o restauraci (tabulka `restaurants_restaurant`) a konfiguraci stahování (tabulka `crawler_restaurantscraperconfig`). V této tabulce je třeba do sloupce `menu_scraper` vložit jméno třídy (v tomto případě `daily_menu.ExampleScraper`). Parametr předávaný do konstruktoru (metoda `__init__`) je hodnota ze sloupce `scraper_parameters`.

Tím je nový zdroj zaregistrovaný a připravený ke stahování dat.

6.4.2 Získávání informací o receptech

Získaná jídla jsou klasifikována na základě podobnosti jména s nějakým staženým receptem. Z těchto dat se aplikace snaží získat seznam ingrediencí a případně nějakou formu kategorizace.

Pro stahování jmen receptů, seznamu surovin a jejich kategorií využívá aplikace sitemap portálů recepty.cz a toprecepty.cz. Tím je jednoduché získat seznam všech receptů.

V případě, že by se v budoucnosti objevil vhodnější zdroj pro tento typ dat, není problém ho poměrně jednoduše integrovat do aplikace.

Bohužel, stejně jako denní menu, i informace o surovinách dost často obsahují chyby a věci, které tam nepatří. Kvůli tomu bylo potřeba vyřešit filtrování dat i na úrovni jednotlivých surovin u receptů.

6.4.2.1 Filtrace dat

Oproti denní nabídce je filtr o poznání jednodušší. Obsahuje celkem tři kroky:

- Odstranění množství
- Odstranění popisu v závorce
- Odstranění mezer a nechtěných znaků

Filtr pro odstranění popisu v závorce jsem do této části zařadil kvůli tomu, že v seznamu ingrediencí se dost často vyskytovalo nějaké slovní upřesnění, jako např. „dle chuti“. Tyto informace ve valné většině případů nepřidávaly žádnou hodnotu, proto jsem se rozhodl je odstranit.

Ingredience se sice do databáze stahují všechny, nicméně pro finální klasifikaci se využívají jen některé. Konkrétně jsem se rozhodl odstranit 3 % nejčastějších ingrediencí (jedná se o suroviny typu „voda“, „sůl“ atp.) a suroviny obsažené v databázi jen jednou (zde se typicky jedná o překlepy nebo jinak nezajímavá data). Tyto hodnoty jsem odhadl na základě několika pokusů a takto nastavené filtrování mi dávalo relevantnější výsledky.

6.4.2.2 Přidání nového zdroje dat pro klasifikaci

V případě, že do budoucna bude potřeba přidat do aplikace další zdroj s recepty, postup je obdobný jako v případě s novým zdrojem denních menu. Věřím, že čtenář promine, když ho zde nebudu tak detailně popisovat. Místo toho uvedu jen názvy tabulek v databázi a místa v kódu, kde lze zjistit, jak to funguje.

Konfigurace se nachází v tabulce `crawler_recipeingredientcrawler`, sloupec `crawler_class` odkazuje na třídu v `crawler/scrapers/recipes`.

6.5 Klasifikace

Klasifikace slouží k extrakci informací jednak ze jména jídla, a jednak k určení příslušnosti k nějaké typické kuchyni. Před samotnou klasifikací je ale potřeba získat klíčová slova.

6.5.1 Získávání klíčových slov

Získávání klíčových slov obsahuje několik fází. První z nich je analýza na základě názvu receptu. V této fázi se program snaží rozdělit název jídla na části, které spolu souvisí. Rozdělení se pokusím ukázat na následujícím pseudokódu:

```
tags = set()
prepositions = ['s', 'se', 'na', 'k', 'a', 'z']
```

```
words = recipe_name.split(" ")

token = ''
for word in words:
   if word not in prepositions:
      end_of_token = word.endswith(',')
      token += " "+word.strip(",")
   else:
      end_of_token = True

   if end_of_token:
      tags.add(token)
      token = ""
```

Tímto přístupem umí aplikace z názvu „Míchaný salát s roastbeefem, dijonským dresingem a krutony“ získat klíčová slova „Míchaný salát“, „rostbeefem“, „dijonským dresingem“, „krutony“.

Takto získaná slova jsou předána do analyzátoru v Elasticsearch, čímž dojde k jejich převedení do kanonické podoby. Pro tento konkrétní případ to bude vypadat takto: `michany salat`, `dijonsky dresing`, `roastbeeefem`, `krutony`. Jak je vidět, český slovník Hunspell evidentně nezná výchozí tvar slova „rostbeefem“, proto se ve výsledku vyskytuje v nezměněné podobě.

Další fází je přidání informace ze stažených receptů, konkrétně tedy seznam ingrediencí a případně kategorií.

6.5.2 Přidání informací o surovinách

V této fázi se využívají jak informace z receptů (kategorie a suroviny), tak i dříve anotované denní menu. Jako příklad použijeme českou klasiku: „Svíčková na smetaně s houskovým knedlíkem“.

V tomto textu budu jako „jídlo“ označovat jak denní menu, tak stažený recept. Prvním krokem je spuštění fulltextového vyhledávání podle jména jídla. Tento dotaz je zpracován pomocí Elasticsearch a vrátí seznam receptů, které alespoň částečně vyhovují dotazu (vyhledávání počítá s překlepy a částečnou shodou). Každý výsledek obsahuje seznam ingrediencí.

Druhým krokem je výběr relevantních surovin. V této části je počítáno, kolikrát se daná ingredience objevila ve všech nalezených jídlech. Do výsledného seznamu klíčových slov se pak zahrnuje 30 % nejčastějších ingrediencí.

Po provedení těchto kroků získáváme následující seznam klíčových slov:

`maso`, `omacka`, `knedlik`, `houskovy knedlik`, `hovezi`, `hovezi kyta`, `hovezi vyvar` a `smetany`. Tento seznam je samozřejmě závislý na datech, která jsou k dispozici. Je velmi pravděpodobné, že časem se tento seznam změní.

6.5.3 Přidání informací o surovinách na základě složených klíčových slov

Tato část je prakticky stejným krokem jako část předchozí. Jen se zde neanalyzuje celé jídlo ale složená klíčová slova. Pokud použijeme stejný příklad jako výše, pak do této analýzy vstoupí jen `houskovy knedlik`, `hovezi vyvar` a `hovezi kyta`. Samotný proces je pak stejný jako výše, jen selekce surovin je daleko přísnější. V této části se vybírá jen 10 % nalezených ingrediencí.

Důvod existence této fáze je ten, aby aplikace byla schopná (samozřejmě s vhodnými daty) odhadnout, co můžou obsahovat např. zmiňované knedlíky.

6.5.4 Zobecnění klíčových slov

Posledním krokem je zobecnění klíčových slov. Pro tento účel jsem ručně sestavil seznam nadřazených a podřazených slov (druhy těstovin, mas, mořských plodů atp.). Tento seznam pak slouží k přidání informace, že „linguine“ jsou těstoviny a „raclette“ je druh sýra.

Pro jídlo „Linguine s krevetami a tomatovou omáčkou“ pak aplikace umí získat následující klíčová slova: `testovina`, `morsky`, `linguine`, `kreveta` a `omacka tomatovy`.

Vím, že toto není ideální způsob, nicméně se mi nepodařilo vymyslet jinou metodu, která by dosahovala alespoň srovnatelných výsledků. Pokusy s analýzou slov, která se často vyskytují blízko sebe, jsem však vzdal, neboť se mi nepodařilo získat relevantní data. Takto ručně vytvořená data jsou uložena v databázi, kde je možné jejich doplnění případně přizpůsobení dle konkrétních požadavků.

6.5.5 Klasifikace jídla podle obsahu

Prvním krokem je analýza a získání informací, co dané jídlo může obsahovat, případně k jaké typické kuchyni může patřit. Každá analyzovaná informace je zachycena v databázi jako hodnota v intervalu od 0 do 1. Takto vznikne reprezentace jídla, kterou budu v následujících odstavcích označovat jako „klasifikační vektor“.

6.5.6 Klasifikace jídla podle kuchyně

První návrh počítal se stažením již klasifikovaných receptů (např. ze serveru `https://apetitonline.cz`, který poskytuje seznam receptů dělených podle národních kuchyní, např. „mexické“ kuchyně: `https://www.apetitonline.cz/narodni-kuchyne/mexicka`).

Předpokládal jsem, že bude možné stáhnout názvy receptů a jejich suroviny, z těchto dat získat klíčová slova a následně porovnávat s klíčovými slovy klasifikovaného denního menu. Výsledná klasifikace měla být počítána jako procentuální zastoupení klíčových slov z denního menu v patřičné kuchyni.

Tato data se ovšem velmi brzy ukázala jako nepříliš vhodná. Četnost jednotlivých surovin byla víceméně vyrovnaná napříč kategoriemi. Tím pádem klasifikace všech jídel vycházela prakticky stejně, ve většině případů však špatně.

Proto jsem se rozhodl změnit přístup a příslušnost ke konkrétní rozpoznávat na základě existujících denních menu. Denní menu se tedy klasifikují na základě podobnosti ke skupině jiných jídel.

Klasifikace probíhá následovně: z označených jídel jsou extrahovány nejčastěji použitá klíčová slova, která jsou porovnávána s klíčovými slovy klasifikovaného jídla. Výsledná klasifikace je určena poměrem klíčových slov obsažených v referenční skupině proti všem klíčovým slovům daného jídla.

6.5.7 Rozpoznání preferencí uživatele

Mimo manuální označení preferencí uživatele aplikace sama rozpoznává, co si uživatel obvykle dává a co mu chutná. K tomuto slouží hodnocení denních menu. Z těchto dat se spočítají preference uživatele jako aritmetický průměr z hodnocených jídel. Pokud uživatel jídlo označil záporně, hodnoty klasifikačního vektoru se odečítají.

Manuální nastavení preferencí je stejný klasifikační vektor, jen s pozměněným rozsahem - pro tyto účely se používají hodnoty -1 – 1. Hodnota -1 slouží pro označení „To mi nechutná“, 1 pak pro hodnotu „To mi chutná“.

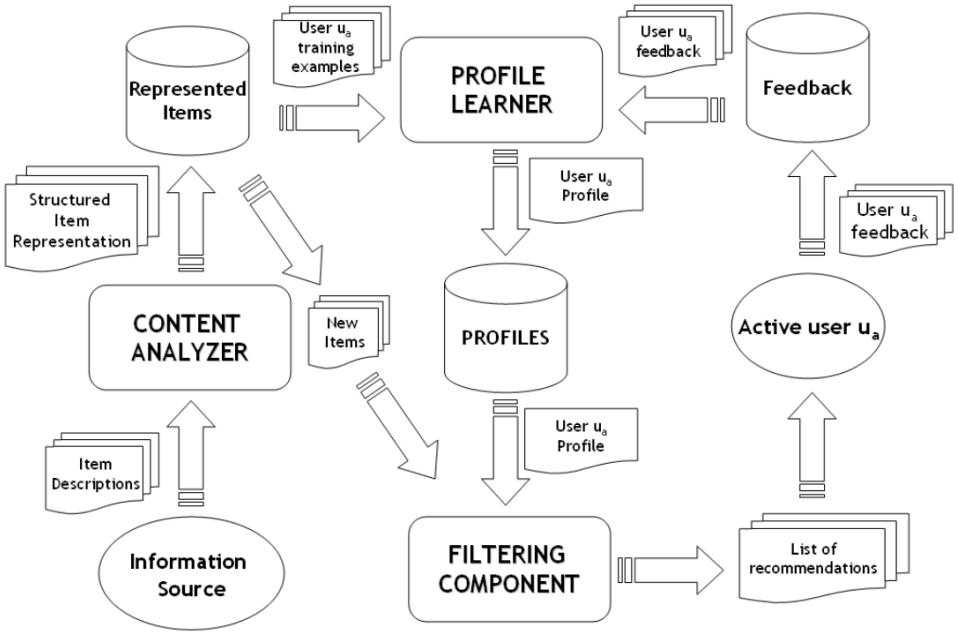
6.6 Tvorba doporučení

V této části jsou řešeny věci týkající se uživatele a generování doporučení. Při návrhu doporučovacího systému jsem se snažil vycházet z high-level architektury popsané na obrázku 6.1, byť v poněkud zjednodušenější podobě. Pokusím se zde popsat jednotlivé kroky, které aplikace dělá za účelem poskytnutí finálního doporučení pro uživatele.

Analýzu vstupních dat a preferencí uživatele zde znovu rozebírat nebudu, neboť tyto fáze byly popsány výše. Zaměřím se na výběr relevantních dat a vytvoření samotných doporučení.

6.6.1 Výběr relevantních dat

Prvním krokem při doporučování je prvotní filtrace relevantních dat. Tato část se dělí do dvou kroků. Prvním z nich je vyfiltrování restaurací na základě polohy, tím druhým je pak další filtrování nabízených jídel. Do doporučování jsou vybírány jen ta jídla, která obsahují více informací (klasifikační vektor jídla nemá příliš mnoho neznámých prvků). Pokud má jídlo vyplněno alespoň 20 % klasifikačních značek, kontroluje se ještě, jestli náhodou neodporuje preferencím uživatele (aby aplikace např. nedoporučovala steak vegetariánovi).



Obrázek 6.1: Vysokoúrovňová architektura content based doporučovacího systému. Převzato z [1]

6.6.2 Získávání osobních doporučení

Prvním krokem je získání klasifikačního vektoru uživatele. U toho se ukládají dva - jeden je vytvářen automaticky, druhý obsahuje informace, které si uživatel ručně nastavil. Tvorba automatického klasifikačního vektoru je jednoduchá - aplikace vybere všechna hodnocení za posledních 30 dní. Výsledný vektor vznikne tak, že kladná hodnocení jsou sečtena a záporná od nich odečtena. Výsledek je následně vydělen počtem hodnocení. Tím pádem jednotlivé prvky v automatickém klasifikačním vektoru odpovídají aritmetickému průměru jednotlivých složek klasifikací jídel.

Během procesu doporučování pracuje aplikace s klasifikačním vektorem uživatele i denního menu. Pro získání osobního doporučení jsou tyto dva vektory sečteny a znormalizovány. Tím vznikne vždy vektor, jehož jednotlivé prvky jsou v intervalu $[-1, 1]$.

Z vyfiltrovaných denních menu je vybráno pět jídel, která jsou nejvíce podobná preferencím uživatele. Samotnou podobnost pak počítám jako absolutní hodnotu ze součtu rozdílů jednotlivých složek vektoru.

Jídla jsou následně seřazena podle podobnosti. Do finálního doporučení se zahrnuje vždy jen jedno jídlo z restaurace. Tím pádem uživatel získá pět tipů na různé restaurace v jeho okolí.

6.6.3 Získávání doporučení společně s přáteli

Doporučování jídla pro skupinu přátel používá doporučování popsané v předchozí části. Pro každého uživatele je vygenerován seznam jeho osobních doporučení. Z těch jsou vybrány restaurace, kam může jít společně co největší počet přátel. Do finálního doporučení se tak ukládají jídla z těchto restaurací.

6.6.4 Zpětná vazba na doporučení

Součástí doporučovacího systému by měla být i funkcionalita pro získávání zpětné vazby od uživatelů, třeba ve formě hodnocení relevance doporučení. Na základě toho by měl doporučovací systém upravovat své budoucí výsledky.

Samotné doporučování aktuálně funguje staticky - data jsou vygenerována na základě podobnosti klasifikace jídla a preferencí uživatele. V uživatelském rozhraní je možné toto doporučení ohodnotit stejně, jako kdyby bylo jídlo uvedeno u nějaké restaurace. Tím pádem poskytuje aplikaci více informací o svých preferencích a ta je zohlední při tvorbě následujících doporučení.

6.7 Restaurace

Poslední část aplikace sdružuje informace a funkce týkající se samotných restaurací a denních menu. V této části aplikace se neřeší nic zajímavého. Rozhodl jsem se to takto oddělit jen čistě pro přehlednost a kvůli rozdělení zodpovědností jednotlivých částí systému.

6.8 Problémy a jejich řešení

V této části bych rád stručně pojednal o problémech, se kterými jsem se během vývoje aplikace setkal. Některé problémy se mi vyřešit podařilo, minimálně za předpokladu že uživatelé nebudou mít snahu aplikaci učit „špatně“, tedy že nebudou zadávat nepravdivá data.

6.8.1 Vstupní formát dat

Jak jsem uvedl v předchozích částech, pro kvalitní analýzu bylo třeba řešit poměrně hodně problémů se vstupními daty. Pokud vstupní data nejsou dostatečně kvalitní, jakákoliv další analýza a práce s těmito daty by byla zbytečná.

Tento problém se mi v práci podařilo vyřešit jen částečně. Bohužel lidská tvořivost je poměrně bohatá a určitě se mi nepodařilo vyřešit všechny problémy, které pro člověka nejsou překážkou, při strojovém zpracování dokáží práci velmi znesnadnit. Aplikace řeší jen následující problémy: extrakci cen z názvu jídla, triviální slučování více řádků menu do jednoho jídla, odstranění záznamů neoznačující jídla, odstranění mnoho opakovaných znaků (typicky

využívané např. pro naznačení vodící linky pomocí mnoha pomlček) a ignorování názvů kratších než 5 znaků (prázdné řádky, různá označení slev a podobných věcí, která sice s menu souvisí, nicméně neoznačují jídlo). Většinu z těchto problémů se mi podařilo (alespoň částečně) vyřešit pomocí regulárních výrazů, o těch zbylých se zmíním v následujícím textu.

6.8.1.1 Slučování víceřádkového menu

Původně jsem plánoval detekovat rozdělení menu na základě přítomnosti ceny u nějakého z řádků. Tento přístup se ovšem ukázal jako velmi nespolehlivý, jednak kvůli faktu, že cena nebyla vždy k dispozici, jednak kvůli tomu, že i když vyplněna byla, vyskytovala se v různých částech rozděleného menu (občas jako první řádek, občas uprostřed, většinou však dosti náhodně).

Z toho důvodu jsem se rozhodl pokusit se detekovat na základě velikosti písmen zadaného menu. Pokud se v názvu denního menu vyskytuje nějaké velké a zároveň malé písmeno, jsou slučovány všechny následující řádky, dokud je prvním znakem malé písmeno. Tzn. následující záznamy budou sloučeny do jednoho:

- Těstoviny linguine s mořskými plody,
- česnekem, feferonkou, petrželkou a
- citrónovým olivovým olejem, zdobené rukolou

Tato metoda sice není stoprocentní, nicméně poskytuje poměrně použitelné výsledky.

6.8.1.2 Rozpoznání „nejídla“

Při analýze jsem chtěl tento problém rozdělit na dvě části – odstranění celého záznamu (protože označuje např. slovní klasifikaci jídel, případně telefonní číslo na rozvoz atp.). Dále pak odstranění jen části názvu abych zachoval označení jídla (např. pro označování jídla jako „Tip šéfkuchaře“).

Pokud uživatel označí záznam jako „nejídlo“, při následující klasifikaci denních menu se získají klíčová slova i takto označených položek, spočítá se jejich absolutní četnost a v případě, že dané klíčové slovo se v „nejídlech“ objevuje alespoň dvakrát, uloží se do databáze. Při stahování a čištění denních menu se pak aplikace dívá do těchto záznamů a v případě, že klíčová slova nového denního menu obsahují minimálně 40 % klíčových slov z „nejídel“, do databáze se neuloží.

Druhý problém - tedy situaci, kdy v názvu jídla je přítomna i část neoznačující jídlo (zmíněný „Tip šéfkuchaře“) jsem se rozhodl nakonec neimplementovat, neboť často docházelo k odstraňování validních částí jmen jídla.

6.9 Použité knihovny

Pro usnadnění práce byly při implementaci backendu použity následující doplňky a knihovny. V tomto seznamu se vyskytují jen ručně instalované doplňky, ne jejich závislosti.

6.9.1 Django

- Verze: 2.1
- URL: `https://docs.djangoproject.com/en/2.1/`
- Licence: BSD

Django je MVC [7] framework navržený pro jednodušší a rychlejší vývoj webů.

6.9.2 pyzomato

- Verze: 0.5
- URL: `https://github.com/fatihsucu/pyzomato`
- Licence: MIT

Tento balíček je velmi jednoduchým wrapperem nad voláním Zomato API. Vlastně neřeší nic jiného, než jen obalení volání REST API do čitelných metod a základní validaci vstupních dat. Knihovna se používá jen u stahování dat ze Zomato API.

6.9.3 requests

- Verze: 2.21
- URL: `https://2.python-requests.org/en/master/`
- Licence: Apache

Knihovna `requests` poskytuje příjemnou správu HTTP požadavků. Aplikace používá knihovnu pro stahování dat a pro dotazování REST API poskytované od Elastic Search.

[7] Model View Controller

6.9.4 beautifulsoup4

- Verze: 2.21
- URL: `https://2.python-requests.org/en/master/`
- Licence: Apache

Knihovna `beautifulsoup4` poskytuje poměrně příjemné rozhraní pro parsování struktury (nejen) HTML dokumentů. Používá se při extrakci dat ze serverů `https://recepty.cz`, `https://toprecepty.cz` a pro stahování denní nabídky z portálu `https://menupraha.cz`.

6.9.5 django-debug-toolbar

- Verze: 1.11
- URL: `https://django-debug-toolbar.readthedocs.io/en/latest/`
- Licence: BSD

Doplněk `django-debug-toolbar` je rozšířením frameworku Django poskytující ladící informace při vývoji, např. jako seznam provedených SQL dotazů, dobu zpracování požadavku a jiné. Tento toolbar se zobrazuje jen ve vývojářském režimu.

6.9.6 djangorestframework

- Verze: 3.9
- URL: `https://www.django-rest-framework.org/`
- Licence: BSD

Django REST Framework je další rozšíření frameworku Django. Tato knihovna jednoduše řeší přidání REST API rozhraní. Knihovna se využívá v samostatné Django aplikaci `api`.

6.9.7 django-cors-headers

- Verze: 2.5
- URL: `https://github.com/ottoyiu/django-cors-headers`
- Licence: MIT

Tato relativně malá knihovna přidává konfigurovatelnou správu CORS [8]. Toto rozšíření je potřeba, neboť API pro aplikaci bude běžet na vlastní subdoméně.

6.9.8 psycopg2-binary

- Verze: 2.8
- URL: `http://initd.org/psycopg/`
- Licence: LGPL (ZPL)

Knihovna `psycopg2-binary` je PosgreSQL driver, obstarává tedy veškerou komunikaci Django aplikace s PostgreSQL databází.

6.9.9 lxml

- Verze: 4.3
- URL: `https://lxml.de/`
- Licence: BSD

LXML je knihovna pro zpracování XML a HTML. Využívá se společně s výše zmiňovanou knihovnou `beautifulsoup4` pro parsování HTML dokumentů.

6.9.10 elasticsearch-dsl

- Verze: 6.4
- URL: `https://github.com/elasticsearch/elasticsearch-dsl-py`
- Licence: Apache

Knihovna `elasticsearch-dsl` poskytuje objektové rozhraní pro komunikaci Python aplikace s vyhledávacím systémem Elasticsearch.

[8] Cross origin resource sharing - bezpečnostní mechanismus pro kontrolu přístupu ke zdrojům pomocí ajaxových požadavků.

Kapitola 7

Implementace webové aplikace

V této části práce bude popsán vývoj webové aplikace. Při vývoji jsem vycházel z wireframů, které jsou součástí analýzy. V této kapitole budou popsány rozdíly, které jsem udělal na základě zpětné vazby, dále představím seznam použitých knihoven a na konci kapitoly budou uvedeny obrázky z finální implementace.

7.1 Rozdíly oproti wireframu

Zpětná vazba po testování wireframů odhalila několik problémů, které jsem se snažil vyřešit při implementaci finální aplikace.

Společně s testery jsme se pokusili upravit textaci aplikace tak, aby byla srozumitelná běžnému uživateli. Tato činnost zahrnovala úpravou textů, složení a pořadí položek v menu aplikace.

Jako druhou úpravu jsem se pokusil upravit samotnou stránku, která byla původně označena jako „Profil“. Tento odkaz jsem přejmenoval na „Sledované“ a zobrazují se na ní jen restaurace, které si uživatel přeje sledovat, případně ty, kam často chodí. Seznam přátel jsem přesunul do samostatné obrazovky. Přehled doporučených jídel jsem také přesunul, konkrétně nad seznam restaurací na titulní straně.

Oproti wireframům ve finální implementaci přibyla možnost nastavit polohu pro tvorbu doporučení. Toto byl jeden z podnětů, které jsem získal během diskuse s testery wireframu. Druhou doplněnou funkcí je možnost označit přítele jako osobu, se kterou chce daný uživatel jít na společný oběd.

Ve finální aplikaci se tedy se seznamem přátel počítá jako s (téměř) neměnným, uživatel má ale jednodušší možnost vybrat, s kým půjde na oběd.

7.2 Administrační rozhraní

Samotná administrace je spravována pomocí administrace vygenerované pomocí Django Adminu. Z toho důvodu se o ní v práci detailně nezmiňuji.

7.3 Použité knihovny

Webová aplikace je stavěna jako SPA [9], k tomu je pro jednodušší vývoj zapotřebí poměrně rozsáhlý ekosystém knihoven. Jak jsem uvedl v kapitole týkající se volby technologií, rozhodl jsem se webovou aplikaci implementovat ve frameworku Vue.js. Z toho vychází většina následujících knihoven, komponent a rozšíření.

V následujícím seznamu uvádím jména balíčků tak, jak jsou k dispozici v balíčkovacím nástroji `npm`. Závislosti potřebné pro vývoj zde neuvádím, obsahují závislosti dané instalací balíčku `vue-cli`.

7.3.1 axios

- Verze: 0.18
- URL: `https://github.com/axios/axios`
- Licence: MIT

Knihovna `axios` poskytuje jednoduché rozhraní pro komunikaci s API pomocí HTTP požadavků. Aplikace ji používá jako výchozí $http objekt.

7.3.2 vue

- Verze: 2.6
- URL: `https://vuejs.org/v2/guide/`
- Licence: MIT

Vue.js, resp. balík `vue` je jádro frameworku. Blíže byl popsánv v rámci volby technologií v sekci Vue.js v rámci volby technologií.

7.3.3 vue-browser-geolocation

- Verze: 1.5
- URL: `https://github.com/scaccogatto/vue-geolocation`
- Licence: MIT

[9] Single Page aplikace

Toto rozšíření obsahuje jednoduché přidání geolokačních služeb do prohlížeče.

7.3.4 vue-router

- Verze: 3.0
- URL: `https://router.vuejs.org/`
- Licence: MIT

`vue-router` je rozšíření Vue.js, které zajišťuje mapování fragmentů URL adres na jednotlivé komponenty, resp. view částí aplikace. Tvoří tak základ pro vývoj SPA aplikací.

7.3.5 vue2-google-maps

- Verze: 0.10
- URL: `https://router.vuejs.org/`
- Licence: MIT

Toto rozšíření přidává komponenty pro práci s Google Maps.

7.3.6 vuetify

- Verze: 1.5
- URL: `https://vuetifyjs.com/en/`
- Licence: MIT

Knihovna `vuetify` obsahuje spoustu komponent pro jednoduché vytvoření uživatelského rozhraní ve stylu Material Design.

7.3.7 vuex

- Verze: 3.0
- URL: `https://vuex.vuejs.org/`
- Licence: MIT

Vuex je knihovna, která zajišťuje centrální správu stavů aplikace. Umožňuje uchovávat a sdílet data mezi komponentami. Podobně jako `vue-router`, tak i `vuex` je součástí Vue.js ekosystému.

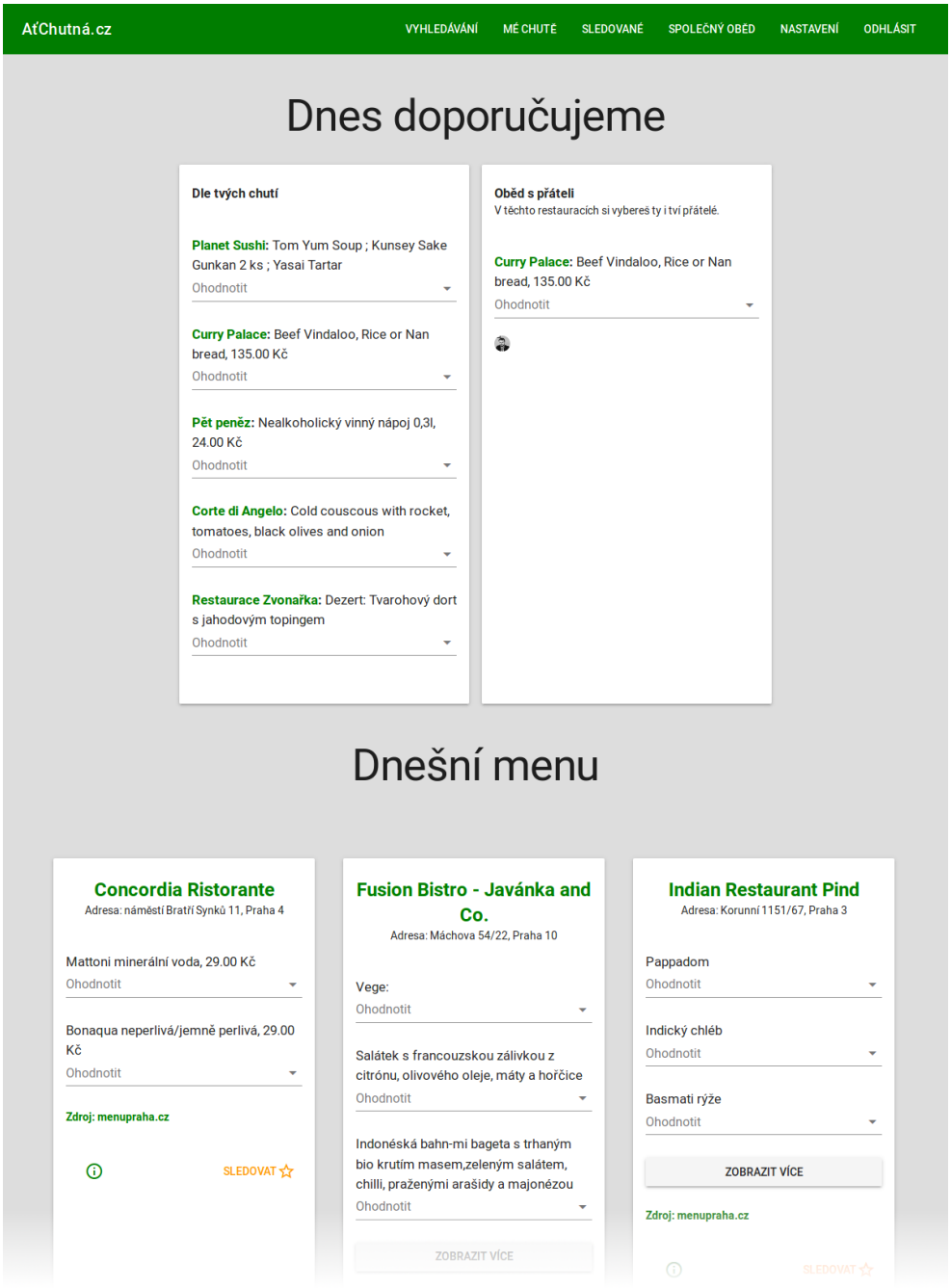
7.3.8 @mdi/font

- Verze: 3.6
- URL: `http://materialdesignicons.com/`
- Licence: MIT

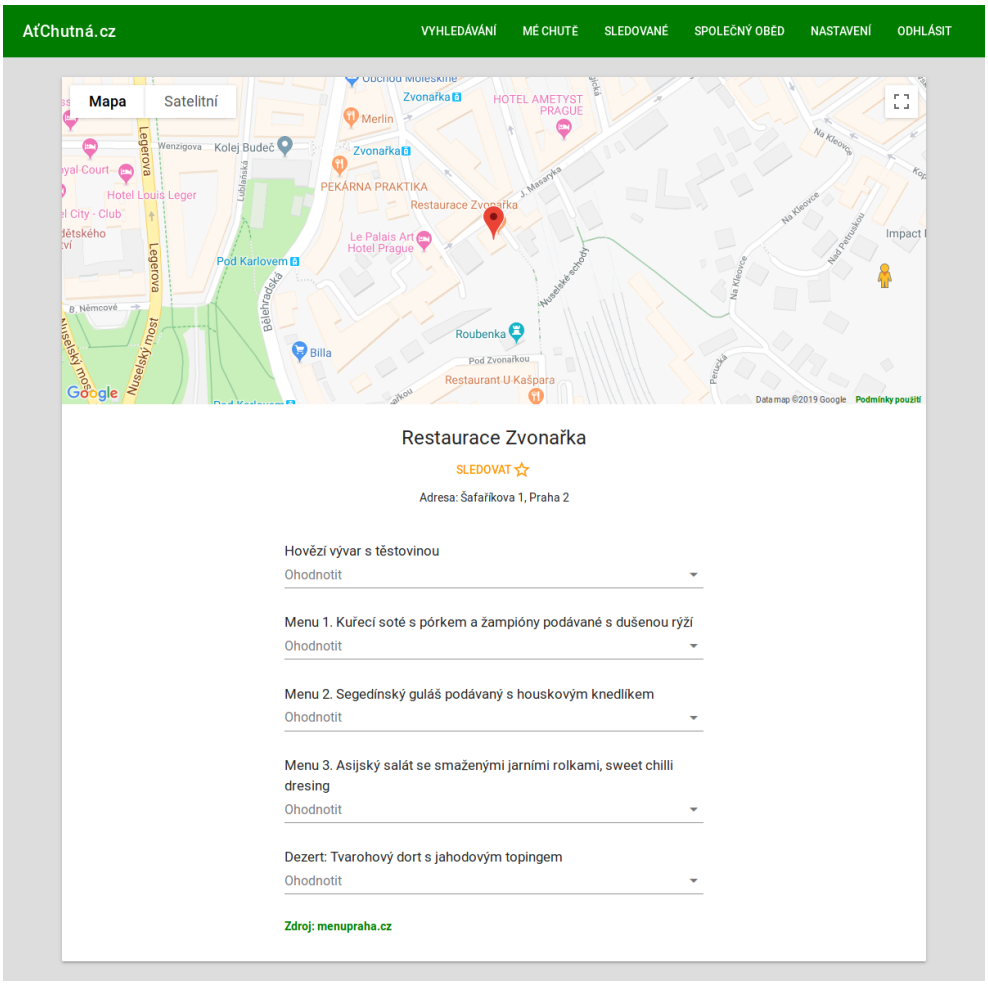
Jedná se o ikonický font, obsahuje ikony ve stylu Material Design a umí se dobře integrovat do knihovny `vuetify`.

7.4 Ukázka finální aplikace

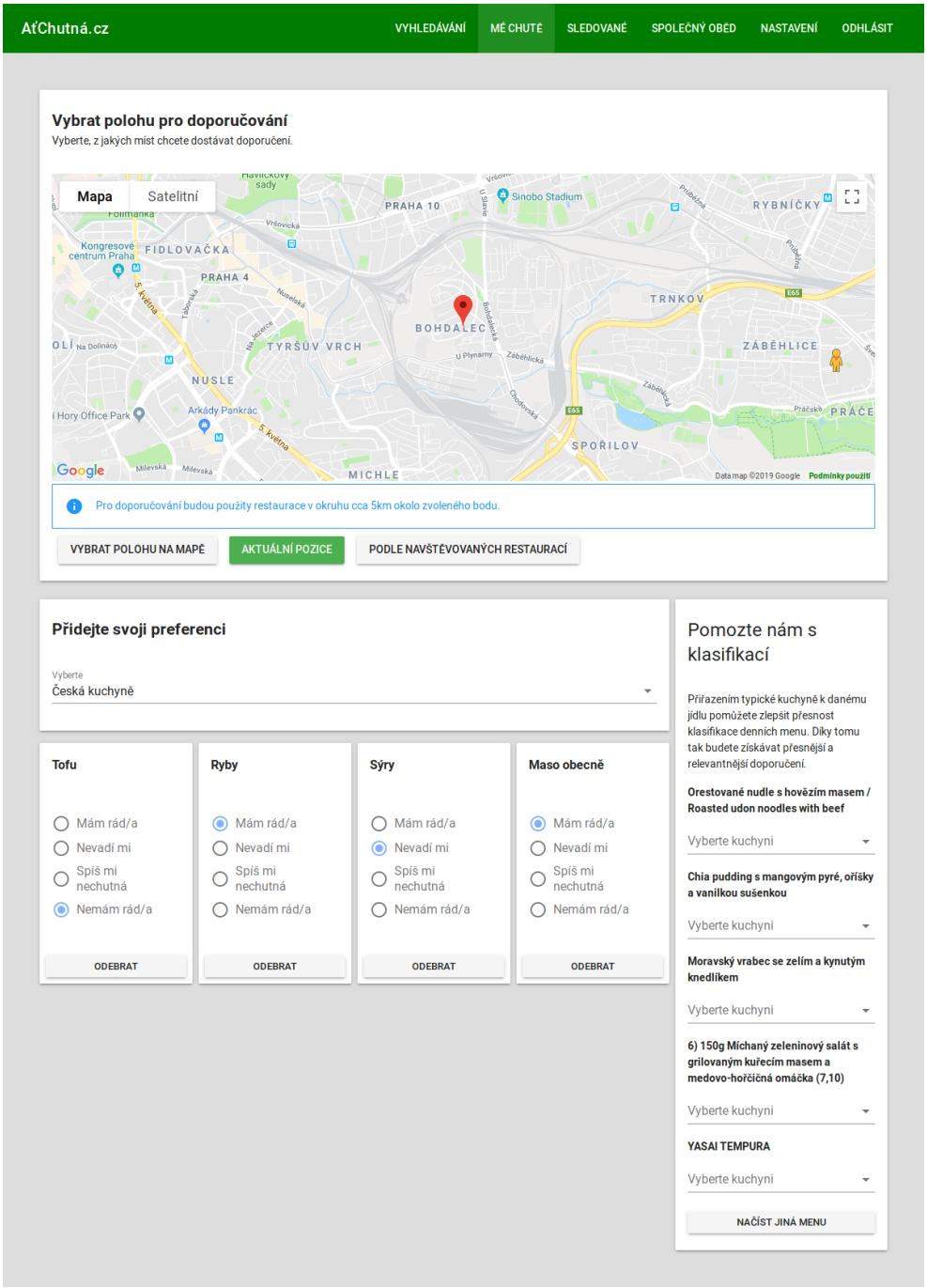
Na závěr kapitoly o implementaci webové aplikace bych rád uvedl pár obrazovek. Samotná aplikace je k dispozici na adrese `https://atchutna.cz`.



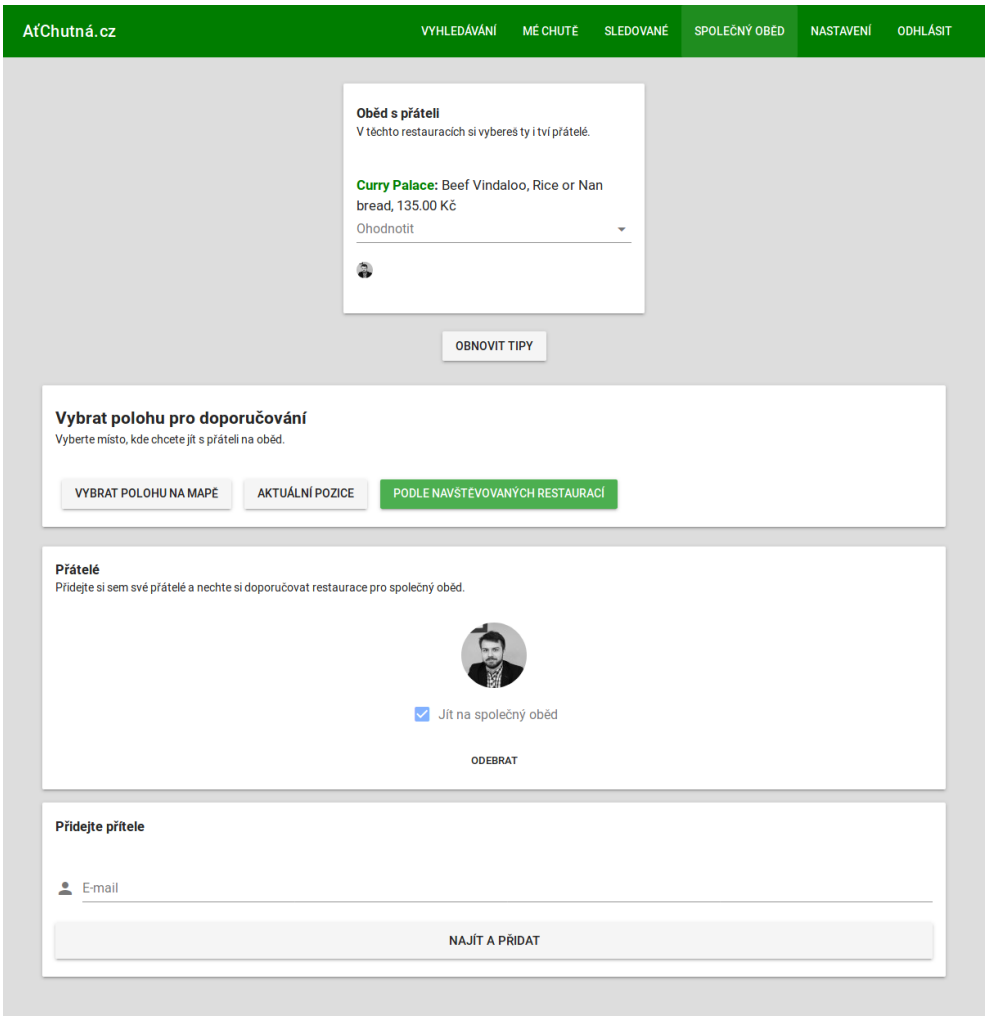
Obrázek 7.1: Přehled denní nabídky



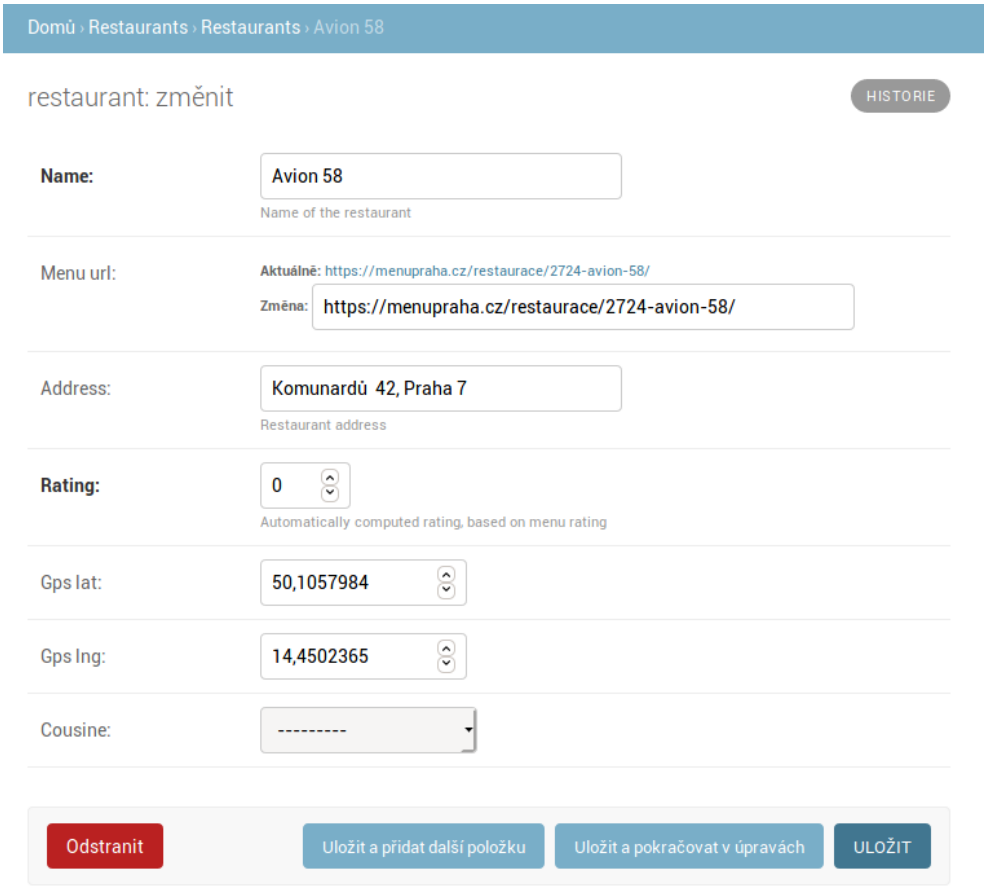
Obrázek 7.2: Detail restaurace



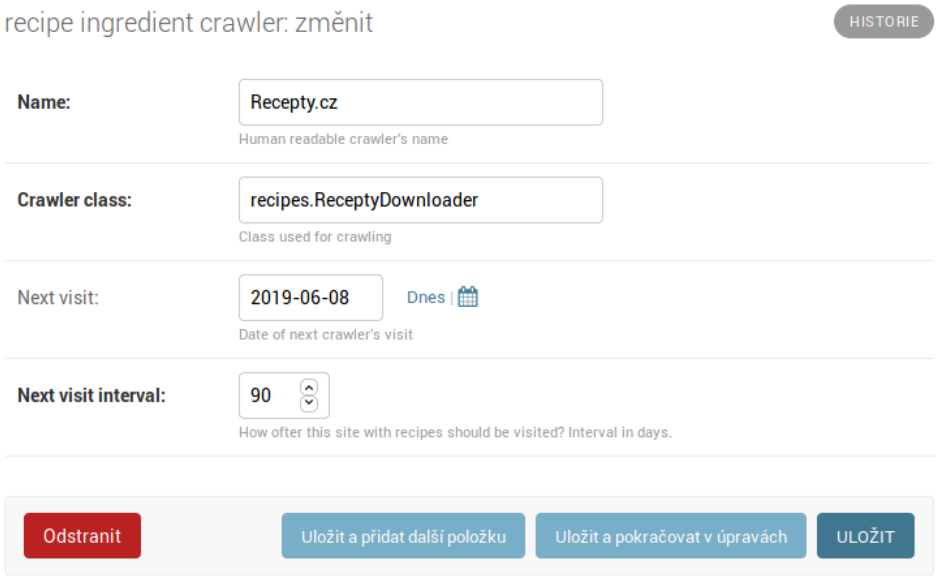
Obrázek 7.3: Nastavení preferencí uživatele



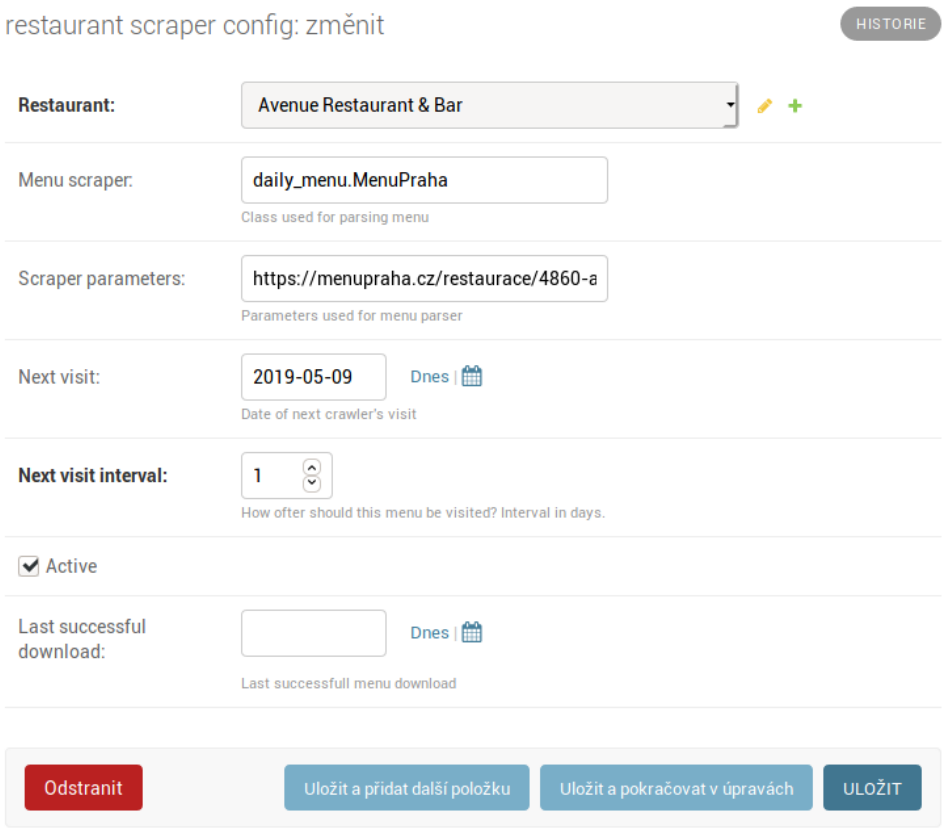
Obrázek 7.4: Seznam přátel a společná doporučení



Obrázek 7.5: Administrační rozhraní - úprava informací o restauraci



Obrázek 7.6: Administrační rozhraní - nastavení stahování receptů



Obrázek 7.7: Administrační rozhraní - nastavení stahování informací o denním menu

Kapitola 8

Testování

Součástí každé implementace by měly být automatizované testy. Pro účely této aplikace jsem pokryl klíčové části backendu unit testy. Pro tento účel využívám Django nadstavbu nad balíčkem `pytest`.

Webová aplikace je pokryta E2E testy pomocí nástroje `cypress`. Vzhledem k jednoduchosti aplikace si myslím, že toto pokrytí testy je dostatečné a v případě potřeby dokáže poměrně rychle objevit problém. Společně s webovým rozhraním je tímto způsobem testováno i API rozhraní aplikace, což je další důvod, proč API nemá samostatné testy.

8.1 Unit testy

Unit testy pokrývají klíčové části aplikace, konkrétně se jedná o proces filtrování vstupních dat, jejich klasifikaci a generování doporučování.

8.1.1 Spuštění testů

Veškeré potřebné nástroje pro testování jsou již v systému k dispozici. Pro spuštění je třeba zadat následující příkaz:

```
$ ./manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....................................
----------------------------------------------
Ran 39 tests in 0.376s

OK
Destroying test database for alias 'default'...
```

Příkaz se spouští v kořenovém adresáři Django aplikace, tedy v adresáři `daily_menu`.

Pro běh testů je třeba, aby byl spuštěný Elasticsearch. Testují se i odpovědi z něj a v případě, že by spojení nebylo dostupné, testy selžou. Vím, že toto řešení není ideální a že zavádí do testů jistou formu nedeterminističnosti, bohužel se mi to ale nepodařilo vyřešit jinak.

8.2 E2E testy

End-to-end testování se zaměřuje na aplikaci jako celek, snaží se co nejvíce přiblížit reálnému chování uživatele. Pro tento účel byl využit testovací nástroj Cypress.io.

8.2.1 Cypress.io

Cypress.io (`https://www.cypress.io/`) je javascriptový nástroj, který umožňuje testovat uživatelské rozhraní webových aplikací. Není vázaný na žádnou konkrétní technologii, lze jej tedy využít pro otestování jak SPA aplikace, tak i aplikací renderovaných na serveru.

8.2.2 Spuštění testů

Nástroj Cypress.io je uveden v seznamu vývojářských závislostí, k jeho instalaci tedy dojde automaticky. Testy jsou připravené v adresáři `daily_menu_web/tests/e2e`. Zde jsou rozděleny do dvou částí - testy veřejného rozhraní a části aplikace vyžadující přihlášeného uživatele.

Před spuštěním je ještě třeba provést drobnou konfiguraci. Je třeba nastavit URL API serveru, na které se má aplikace při testování připojovat. Dále je třeba definovat, pod jakým uživatelem budou prováděny testy. K tomuto účelu slouží soubor `cypress.env.json`. Vzor tohoto souboru je k dispozici pod jménem `cypress.env.example.json`.

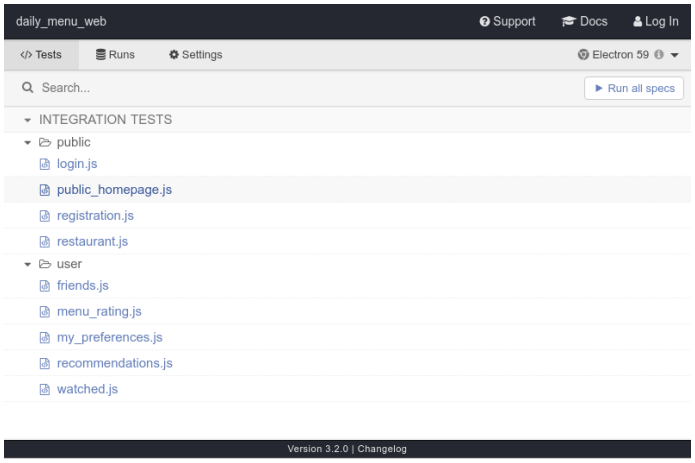
Po úprave konfigurace je možné přejít ke spuštění testů. Pro tento účel je třeba zadat následující příkaz:

```
npm run test:e2e
```

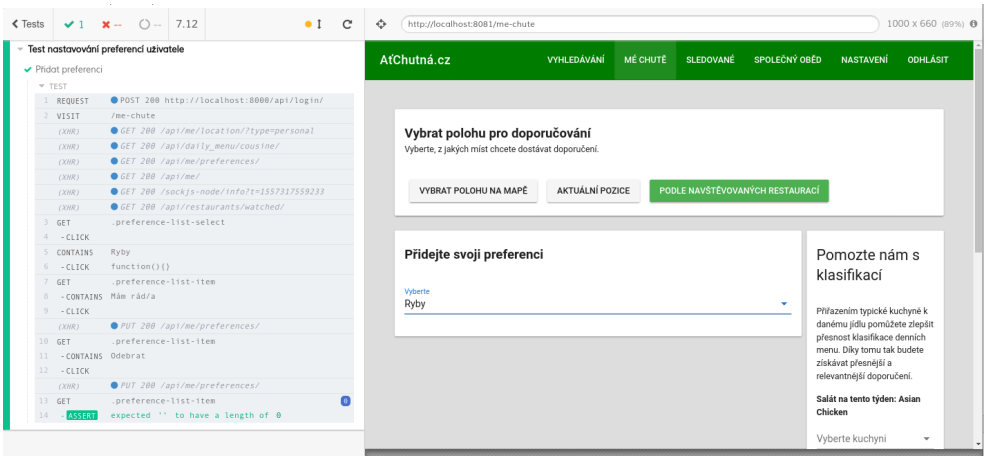
Po chvíli se otevře rozhraní nástroje, ve kterém je k dispozici přehled existujících testů. Ty je možné spouštět buď po jednom, nebo klidně všechny naráz.

Po spuštění dojde k otevření okna prohlížeče a nástroj začne provádět sadu připravených úkonů.

Pomocí tohoto typu testů jsem se snažil pokrýt základní operace, které může uživatel s aplikací dělat. Pro nepřihlášeného uživatele jsou testovány následující části aplikace:



Obrázek 8.1: Cypress.js - hlavní okno



Obrázek 8.2: Cypress.io - detail proběhnutého testu

- Zobrazení seznamu restaurací na titulní straně
- Přihlášení
- Registrace
- Zobrazení detailu restaurace

Pro přihlášeného uživatele pokrývají testy tyto akce:

- Ohodnocení denního menu
- Nastavení preferencí

- Zobrazení seznamu doporučení
- Přidání a odebrání restaurace ze „Sledovaných“
- Správa seznamu přátel

8.3 Výkon API endpointů

V nefunkčních požadavcích bylo definováno, že všechny GET operace nad API musí uživateli poskytnout odpověď do 0,5s. Pro ověření tohoto požadavku jsem se rozhodl využít nástroj `ab`, testování probíhalo se třemi konkurenčními požadavky. Pro každý test byl patřičný endpoint volán celkem 100×.

Pro jednodušší spuštění testů jsem připravil skript `api-performace.sh`. Tento skript lze nalézt v přílohách. Vzhledem k tomu, že většina testovaných endpointů je vázaná na uživatele, je nutné získat autentizační token. K tomu je možné využít následující příkaz:

```
$ curl -X POST \
      http://api.atchutna.cz/v1/login/ \
      --data "username=<email>&password=<heslo>"

# výstup
{"token":"c5a4bf0836a36ba2196cef5e30550bc12f718c12"}
```

Takto získaný token je třeba použít jako parametr pro spuštění zmiňovaného shell skriptu. Spuštění a ukázka výstupu pak může vypadat například takto.

```
$ ./api-performance.sh c5a4bf0836a36ba2196cef5e30550bc12f718c12

/today_menu/restaurants/
Time per request:       77.290 [ms] (mean)
Time per request:       25.763 [ms] (mean, across all concurrent
requests)

/search?q=testoviny
Time per request:       13.996 [ms] (mean)
Time per request:       4.665 [ms] (mean, across all concurrent
requests)

...
```

Testování probíhalo na VPS hostované u společnosti WEDOS, bez vyhrazeného procesoru.

Jak je vidět z výsledků, uživatel získává odpověď do jedné vteřiny ve všech případech, mnohdy s velkou rezervou.

8.4 Závěr

V této kapitole bylo otestování rozhraní aplikace i její backend. Pokrytí testy (*test coverage*) není bohužel tak vysoké, jak by asi být mělo. Tuto část beru jako inspiraci pro budoucí vylepšení aplikace.

Kapitola 9

Spuštění

V této kapitole budou popsány kroky potřebné ke spuštění aplikace na vlastním serveru. Tento postup nepopisuje spuštění aplikace v produkčním prostředí.

Pro zprovoznění aplikace je zapotřebí mít nainstalovaný webserver, Docker, Docker Compose a Git pro získání zdrojových kódů aplikace. Pro zprovoznění webové aplikace je nutné mít nainstalované `npm` pro stažení javascriptových závislostí.

9.1 Instalace doporučovacího systému

Před samotným šputěním aplikace je potřeba nakonfigurovat backend, např. připojení k databázi, Zomato API client ID, SMTP server pro odesílání e-mailů atp. K tomuto účelu je třeba zkopírovat soubor `app.example.py` v adresáři `daily_menu/config` do `daily_menu/config` a upravit patřičné nastavení.

Zomato Client ID je možné získat zde: `https://developers.zomato.com/api`

Po uložení souboru je třeba sestavit Docker image, Pro tento účel je určen nástroj `docker-compose`. Pro sestavení kontejnerů stačí spustit příkaz `docker-compose build`.

Po stažení všech částí a jejich instalaci stačí aplikaci spustit. To se provede zadáním příkazu `docker-compose up`. Po chvíli je aplikace spuštěná a API server je dostupný na adrese `http://localhost:8000`.

9.1.1 Naplnění výchozími daty

V první řadě je nutné zajistit naplnění databáze záznamy, které jsou třeba pro nastavení stahování dat - ať už se jedná o denní nabídku nebo recepty určené ke klasifikaci. K tomuto účelu jsou připraveny příkazy `manage.py init_data`, `menupraha_restaurants` a `zomato_restaurants`. První uvedený inicializuje

potřebná data v databázi, jako např. klasifikované kuchyně, seznam nadřazených a podřazených slov, vytvoření scraperů atp. Druhé dva příkazy slouží ke stažení seznamu restaurací ze Zomato, resp. MenuPraha.cz.

9.1.2 Periodické spouštění úloh

Aby aplikace měla aktuální data, tato data byla správně klasifikována a mohla být doporučována uživatelům, je nutné zajistit periodické spouštění následujících příkazů.

V případě, že je aplikace spuštěna pomocí Dockeru (resp. `docker-compose`), je třeba nezapomenout na to, že tyto příkazy musí běžet uvnitř daného kontejneru.

Pro správnou funkčnost aplikace je třeba zajistit spouštění následujících skriptů. Jejich seznam uvádím v pořadí, v jakém je vhodné aby byly spouštěny.

manage.py download_recipe

Tento příkaz slouží ke stažení receptů určených ke klasifikaci. Jedná se o velmi časově náročnou úlohu. Tento skript získává seznam receptů ze sitemap a stahuje jen ty recepty, které ještě nemá.

Samotné stahování receptů si však samo ukládá informaci o tom, kdy nejdříve se má spustit, proto je možné tento skript spouštět z crontabu např. jednou za týden.

manage.py download_daily_menu

Tato úloha řeší samotné stahování denních menu. Je třeba, aby proběhla minimálně jednou denně.

manage.py cousine_tagger

Příkaz `cousine_tagger` získává klíčová slova z jídel, která byla přiřazena do nějaké kuchyně.

Tento příkaz je třeba spouštět před klasifikací denních menu. Stačí, když proběhne alespoň jednou denně.

manage.py update_search

Tento příkaz řeší tagování surovin a aktualizaci dokumentů v Elasticsearch vyhledávání. Mimo dat určených ke klasifikaci aktualizuje i data určená k vyhledávání na webu. `update_search` musí proběhnout po stažení denní nabídky, tedy také minimálně jednou denně.

manage.py classify_daily_menu

Příkaz `classify_daily_menu` spouští ohodnocování a klasifikaci denních menu. Je potřeba, aby tento příkaz běžel po aktualizaci dokumentů v Elasticsearch.

Pro urychlení klasifikace je možné tento příkaz spustit několikrát paralelně.

manage.py classify_user

Tento příkaz slouží k výpočtu klasifikačního vektoru uživatele. Je třeba, aby tento vektor byl aktuální před spuštěním tvorby doporučení.

manage.py generate_recommendations

Tímto dojde k vygenerování všech typů doporučení pro všechny uživatele. Tvorba všech doporučení pro jednoho uživatele trvá zhruba vteřinu, proto při větším počtu uživatelů není vhodné spouštět tuto úlohu velmi často.

9.2 Instalace webové aplikace

Jako první krok je třeba nainstalovat potřebné závislosti. K tomu slouží příkaz `npm install`, který nainstaluje všechny závislosti uvedené v `package.json`.

Dále je třeba upravit nastavení aplikace - konkrétně se jedná o url API serveru a Google Maps API klíč. Tyto hodnoty se konfigurují v souboru `daily_menu_web/src/settings.js` (je třeba ho vytvořit podle šablony uvedené ve verzovaném souboru `daily_menu_web/src/settings.example.js`).

Po upravení hodnot lze přejít k samotnému sestavení aplikace. Pro tento účel je třeba spustit příkaz `npm run build`. Tímto dojde ke zkompilování aplikace do adresáře `daily_menu_web/dist`.

Pokud je spuštěné a funkční API rozhraní, lze přistoupit ke spuštění webové aplikace. Vzhledem k povaze aplikace není třeba žádná složitá konfigurace. Je nutné zajistit jen načítání obsahu adresáře `daily_menu_web/dist` na doméně `http://atchutna.cz`. Toho můžeme docílit např. přidáním VirtualHostu v následující podobě:

```
<VirtualHost atchutna.cz:80>
    DocumentRoot /var/www/atchutna.cz/dist
</VirtualHost>
```

Po uložení a restartu Apache serveru by aplikace měla být spuštěna a připravena k použití.

9.3 Závěr

V této části byl popsán způsob, jakým je možné zprovoznit aplikaci na svém serveru. Tento způsob nasazení není určen pro produkční provozování. Samotná aplikace, tedy backend a API, je v aktuálním nastavení spouštěna přes příkaz `runserver`, který sám o sobě není určen k provozování produkčních aplikací [35].

Kapitola 10

Možnosti rozšíření, nápady na vylepšení

Myslet si, že aplikace je v této fázi hotová a dokončená by bylo naivní. Rád bych zde uvedl pár nápadů na vylepšení uživatelského prostředí, kvality dat a doporučení, případně rozšíření funkcionality aplikace.

10.1 Zasílání doporučení na e-mail

V závěru uživatelského testování padl dotaz, zda aplikace bude umět rozesílat doporučená denní menu i na e-mail, případně komunikátor typu Slack. Tato funkce je nad rámec této práce.

10.2 Zadávání denních menu z prostředí aplikace

Aplikace aktuálně podporuje získávání dat pouze ze zdrojů třetích stran. Napojení nového zdroje je sice jednoduchá záležitost, nicméně je nutné vytvoření scraperu např. webových stránek.

Možnost zadávat denní nabídku pomocí prostředí přímo v aplikaci jsem do systému neintegroval záměrně. Přišlo mi, že se jedná o funkci nad rámec doporučovacího a klasifikačního systému. Nicméně v případě, že by bylo vyvinuto přehledné uživatelské rozhraní pro správu restaurací, je možné, že by se zvýšila kvalita dat. Uživatelé by totiž neměli potřebu vkládat data v nekvalitní podobě.

10.3 Kvalitnější data pro klasifikaci

I přes veškerou snahu se občas nedaří získat relevantní klasifikaci pro denní nabídku. Jedním z důvodů jsou i data, se kterými je porovnáváno klasifikované jídlo. Data z portálů Recepty.cz a TopRecepty.cz nejsou vždy ideální. Pro

větší přesnost doporučení a klasifikaci preferencí by bylo vhodné najít zdroj kvalitnějších dat.

10.4 Fotografie restaurací

Pro zpřehlednění seznamu restaurací by bylo vhodné jednotlivé restaurace od sebe odlišit i graficky. Např. pomocí jejich loga nebo fotografií. Uživatelé by se tak snadněji zorientovali v seznamu restaurací.

10.5 Vylepšení testů

V aplikaci jsou sice testy pokryty stěžejní části, nicméně samotné testy by určitě šly vytvořit daleko lépe. Jednou z rezerv v testování vidím závislost na externích datech, konkrétně na obsahu databáze Elasticsearch. Pro další vývoj a rozvoj aplikace by bylo žádoucí tuto závislost v testování odstranit.

10.6 Nasazení aplikace na HTTPS

Aktuálně aplikace běží na nezabezpečeném protokolu HTTP. Pro spuštění pro reálné uživatele je třeba zajistit šifrovanou formu komunikace.

Závěr

V práci byly analyzovány aplikace, které poskytují informace o denních menu restaurací. V této fázi jsem s překvapením zjistil, že žádný z těchto portálů nepoužívá žádnou formu doporučování jídel. Jediná služba, Restu.cz, poskytuje návrhy na restaurace.

Na základě získaných poznatků bylo navrženo uživatelské rozhraní aplikace pomocí wireframů. Tento návrh byl otestován a výsledky jsem následně promítnutl do návrhu finální podoby uživatelského rozhraní.

V další části práce navržen způsob anotace a klasifikace jídel. Navržený přístup využívá podobnost názvu jídel, podle které se snaží odhadovat, jaké ingredience může dané jídlo obsahovat. Na základě těchto dat provádí výsledná aplikace klasifikaci jídla do připravených skupin. Z klasifikací jídel jsou odhadovány i chuťové preference jednotlivých uživatelů. Na základě těchto dat jsou pak doporučována nová denní menu.

Výsledná aplikace byla zveřejněna na službě GitHub s open source licencí. Repozitář je veřejně dostupný na adrese `https://github.com/michalkvacek/at-chutna`.

V případě, že má doporučovací systém vhodná data, dává poměrně relevantní výsledky. V opačném případě výsledky tak dobré nejsou, což vnímám jako možnost pro budoucí vylepšení.

Literatura

[1] Ricci, F.; Rokach, L.; Shapira, B.: *Introduction to Recommender Systems Handbook*. Springer, Boston, 2011, ISBN 978-0-387-85819-7. Dostupné z: http://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf

[2] Lee, J.; Sun, M.; Lebanon, G.: A Comparative Study of Collaborative Filtering Algorithms [online]. 2012, [cit. 2019-03-09]. Dostupné z: https://arxiv.org/pdf/1205.3193.pdf

[3] Luo, S.: Introd to Recommender System: Collaborative Filtering [online]. 2018, [cit. 2019-03-09]. Dostupné z: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[4] Pinela, C.: Content-Based Recommender Systems [online]. 2015, [cit. 2019-03-09]. Dostupné z: https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235

[5] Luk, K.: Introduction to TWO approaches of Content-based Recommendation System [online]. 2019, [cit. 2019-03-10]. Dostupné z: https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c

[6] Zomato: O nás [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://www.zomato.com/about

[7] Zomato: How Zomato Order works [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://www.zomato.com/order

[8] Zomato: Zomato API [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://developers.zomato.com/api?lang=cs

[9] Restu: Jak Restu funguje? [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://www.restu.cz/co-je-restu/

[10] Restu: Obchodní podmínky [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://www.restu.cz/obchodni-podminky/

[11] MenuPraha.cz: Podmínky užívání webových stránek MenuPraha.cz [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://menupraha.cz/podminky-uzivani/

[12] Sanz, D. S.; Agrawal, A.: Automated Menu Recommendation System Based on Past Preferences. *International Journal of Advanced Computer Science and Applications*, ročník 5, č. 7, 2014. Dostupné z: http://ijarai.thesai.org/Downloads/Volume5No7/Paper_11-Automated_menu_recommendation_system_based_on_past_preferences.pdf

[13] CENTER, C. N.: Autorská práva k publikovaným materiálům [online]. 2019, [cit. 2019-03-15]. Dostupné z: https://www.cncenter.cz/clanek/2658/autorska-prava-k-publikovanym-materialum

[14] Bachheriya, A.: Top 6 Data Science Programming Languages for 2019 [online]. 2019, [cit. 2019-04-13]. Dostupné z: https://medium.com/datadriveninvestor/top-6-data-science-programming-languages-for-2019-39ba1b6819a8

[15] Babu, A.: Top 8 programming languages every data scientist should master in 2019 [online]. 2019, [cit. 2019-04-13]. Dostupné z: https://bigdata-madesimple.com/top-8-programming-languages-every-data-scientist-should-master-in-2019/

[16] Hayes, B.: Programming Languages Most Used and Recommended by Data Scientists [online]. 2019, [cit. 2019-04-13]. Dostupné z: http://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/

[17] Django: Writing custom django-admin commands [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://docs.djangoproject.com/en/2.1/howto/custom-management-commands/

[18] Django: *Documentation: Making queries [online]*. [cit. 2019-03-09]. Dostupné z: https://docs.djangoproject.com/en/2.1/topics/db/queries/

[19] Cromwell, V.: Evan You [online]. 2016, [cit. 2019-04-25]. Dostupné z: https://bit.ly/2PTIYDj

[20] of JavaScript 2017, T. S.: Front-end Frameworks – Results [online]. 2017, [cit. 2019-04-25]. Dostupné z: https://2017.stateofjs.com/2017/front-end/results/

[21] Miśta, W.: State of Vue.js in 2018 [online]. 2018, [cit. 2019-04-25]. Dostupné z: https://naturaily.com/blog/vue-js-2018

[22] Sajnóg, M.: 13 Top Companies That Have Trusted Vue.js [online]. 2018, [cit. 2019-04-25]. Dostupné z: https://www.netguru.com/blog/13-top-companies-that-have-trusted-vue.js-examples-of-applications

[23] Inc, F.: React: A JavaScript library for building user interfaces [online]. 2019, [cit. 2019-03-09]. Dostupné z: https://reactjs.org/

[24] Inc, F.: Virtual DOM and Internals [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://reactjs.org/

[25] Warcholinski, M.: 10 Famous Apps Using ReactJS Nowadays [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://brainhub.eu/blog/10-famous-apps-using-reactjs-nowadays/

[26] Netflix: 10 Famous Apps Using ReactJS Nowadays [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://medium.com/netflix-techblog/netflix-likes-react-509675426db

[27] MongoDB: Our Customers [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://www.mongodb.com/who-uses-mongodb

[28] PostgreSQL: *Chapter 8. Data Types [online]*. [cit. 2019-03-09]. Dostupné z: https://www.postgresql.org/docs/9.5/datatype.html

[29] stackshare: PostgreSQL [online]. 2019, [cit. 2019-04-25]. Dostupné z: https://stackshare.io/postgresql

[30] RNDr. Pavel Šmerk, P.: ajka =¿ majka [online]. [cit. 2019-04-29]. Dostupné z: https://nlp.fi.muni.cz/czech-morphology-analyser/majka.html

[31] Elasticsearch: *The Heart of the Elastic Stack [online]*. [cit. 2019-04-29]. Dostupné z: https://www.elastic.co/products/elasticsearch

[32] Elasticsearch: *Language Analyzers [online]*. [cit. 2019-04-29]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-lang-analyzer.html

[33] Elasticsearch: *Hunspell Token Filter [online]*. [cit. 2019-04-29]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-hunspell-tokenfilter.html

[34] Django: *Applications [online]*. [cit. 2019-04-01]. Dostupné z: `https://docs.djangoproject.com/en/2.1/ref/applications/`

[35] Django: *Documentation: django-admin and manage.py [online]*. [cit. 2019-03-09]. Dostupné z: `https://docs.djangoproject.com/en/2.1/ref/django-admin`

Příloha **A**

Obsah přiložené SD karty

- `src`
  - `daily_menu` .. zdrojové kódy doporučovacího systému a API rozhraní
  - `daily_menu_web` ..... zdrojové kódy webové aplikace
- `UML` ..... doménový model a use case diagramy ve formátu pro UMLet
- `At Chutna.pdf` ..... wireframe uživatelského rozhraní
- `api-performance.sh` ..... skript na otestování výkonnosti API rozhraní
- `model.dbm` ..... databázový model ve formátu pro nástroj pgModeler
- `thesis.pdf` ..... text práce ve formátu PDF