STU FIIT

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

My Food Court

Projektová dokumentácia - Inžinierske dielo

Akademický rok: 2016/2017

Predmet: Tímový projekt

Členovia tímu: Jozef Balún, Martin Prekala, Miroslav Rác, Matej Schwartz, Štefan Schwartz, Igor Šimko

Vedúci tímu: Ing. Michal Kompan, PhD.

Obsah

1. Inžinierske dielo ZS **7**

1.1. Úvod 7

1.2. Ciele projektu 7

1.2.1. Čo naša aplikácia je a čo nie je? 8

1.3. Návrh architektúry 8

1.3.1. Frontend 10

1.3.2. API 13

1.3.3. Sťahovač dát 16

1.3.4. Dátový model 18

1.3.5. Server 20

1.4. Testovanie 21

1.5. Technická dokumentácia 22

1.6. Príručky 24

2. Inžinierske dielo LS **25**

2.1 Úvod 25

2.2 Celkový pohľad na systém 25

2.2.1 Globálne ciele projektu 25

2.2.2 Opis architektúry 27

2.2.2.1 Frontend 27

2.2.2 API 29

2.2.3 Sťahovač dát 29

2.2.4 Dátový model 30

2.2.5 Server 31

2.2.6 Funkcionalita 32
2.2.7 Referencie 34
2.2.8 Zoznam priložených e-dokumentov 34
2.3 Moduly systému 34
Prihlasovanie a registrácia 35
Analýza 35
Návrh 35
Implementácia 36
Testovanie 37
2. Profil 37
Analýza 37
Návrh 37
Implementácia 38
Testovanie 39
3. Zoraďovanie výsledkov 39
Analýza 39
Návrh 40
Implementácia 40
Testovanie 44
4. Zoznam jedál 44
Analýza 44
Návrh 44
Implementácia 45
Testovanie 48
5. Detail jedla 48
Analýza 48

Návrh 48
Implementácia 49
Testovanie 50
6. Hodnotenie 50
Analýza 50
Návrh 50
Implementácia 51
Testovanie 51
7. Mapa 52
Analýza 52
Návrh 52
Implementácia 53
Testovanie 53
8. Zvolenie polohy 53
Analýza 53
Návrh 53
Implementácia 54
Testovanie 55
9. Navigácia 55
Analýza 55
Návrh 55
Implementácia 55
Testovanie 56
2.4 Zamrazená verzia aplikácie 56
Príloha A - Motivačný dokument **57**
Príloha B - Príručky ZS **63**

Getting started - Angular app 63

ElasticSearch 65

Príloha C - Inštalačná príručka LS 70

Aplikácia 70

API 70

Crawler 70

1. Inžinierske dielo ZS

1.1. Úvod

V tomto dokumente je spísaná technická dokumentácia k predmetu Tímový projekt. V jednotlivých bodoch opisuje prácu na projekte, v ktorom sa snažíme vytvoriť personalizovaný nástroj na odporúčanie jedál tvoriacich denné menu stravovacích zariadení.

Kapitola 1.2 vysvetľuje naše ciele, ktoré chcem v projekte dosiahnuť. Vzhľadom k rozdeleniu predmetu na dva semestre sa v prvom, zimnom, semestri budeme venovovať základnej funkcionalite aplikácie bez pridanej hodnoty, ktorou bude odporúčanie na základe používateľom zadaných parametrov. Následne je stručne opísané, čo naša aplikácia je a čo naopak nie je. V súčasnosti totiž existuje množstvo podobných aplikácií, a aj preto je dobré odlíšiť náš cieľ hneď na začiatku práce.

1.2. Ciele projektu

Výsledkom tohto projektu bude aplikácia, ktorá agreguje informácie o denných ponukách jedál. Používateľovi zobrazuje personalizovanú ponuku, ktorá berie ohľad na osobné preferencie pre suroviny a spôsob prípravy jedál, jeho osobné výživové potreby, či obmedzenia vo forme alergénov.

Personalizované výsledky aplikácia navrhne na základe predošlého výberu a podľa informácií, ktoré získa neinvazívnou formou jednoduchých otázok. Existujúce služby jednoducho zobrazujú dáta o ponuke jedál. Naše riešenie však získané dáta spracuje, pochopí a navrhne personalizovanú ponuku na základe profilu o stravovacom návyku používateľa.

Dáta o ponukách reštauračných zariadení sú pravidelne získavané z dostupných webových lokalít alebo ich reštaurácie poskytujú prostredníctvom webového rozhrania.

Pre používateľa aplikácia predstavuje asistenta v rozhodnutí, ktoré robí každý deň a zároveň reštauračným zariadeniam slúži ako efektívny marketingový nástroj s cielením na správne publikum.

Informácie sú najcennejšou komoditou. Našim dlhodobým zámerom je vybudovať databázu informácií o trende stravovacích návykov ľudí v konkrétnych regiónoch.

1.2.1. Čo naša aplikácia je a čo nie je?

Aplikácia je a poskytuje:

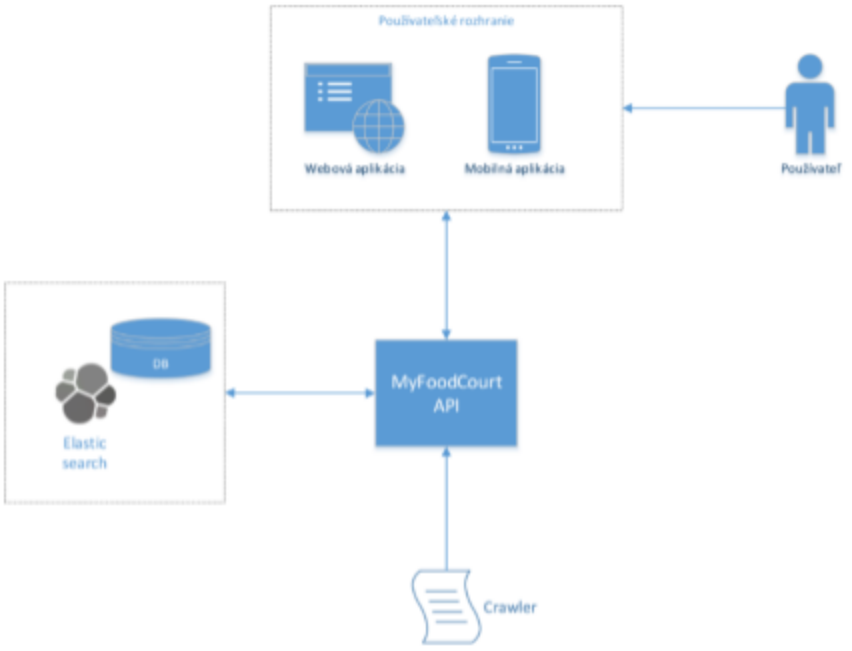
- agregátor informácií o dennej ponuke jedál,
- personalizovaná ponuka jedál v blízkom okolí,
- aplikácia dokáže zhodnotiť stravovací návyk používateľa.

Aplikácia nie je:

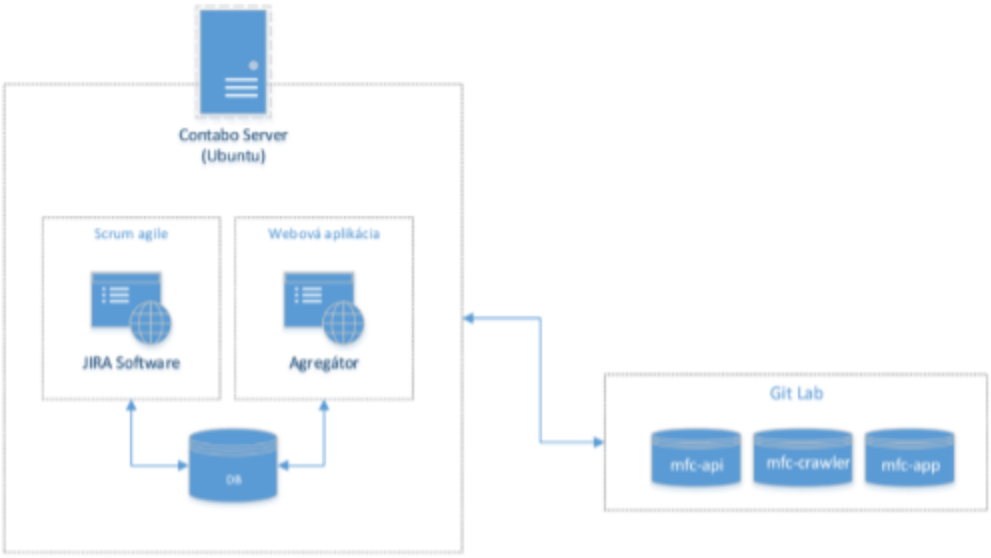
- výživový poradca,
- sociálna sieť,
- katalóg reštauračných zariadení.

1.3. Návrh architektúry

Jadrom celého systému je API, ktorá ukladá dáta do *PostgreSQL* databázy, a tiež využíva *Elasticsearch*. Využíva ju frontend aplikácia písaná v *Angular 2*, ktorá predstavuje používateľské rozhranie a je možné ju otvoriť v prehliadači na počítači alebo mobile. Neskôr bude exportovaná ako natívna mobilná aplikácia. Dáta zbiera sťahovač (angl. crawler), ktorý ich ukladá do databázy vytváraním požiadaviek na API.



Infraštruktúra API

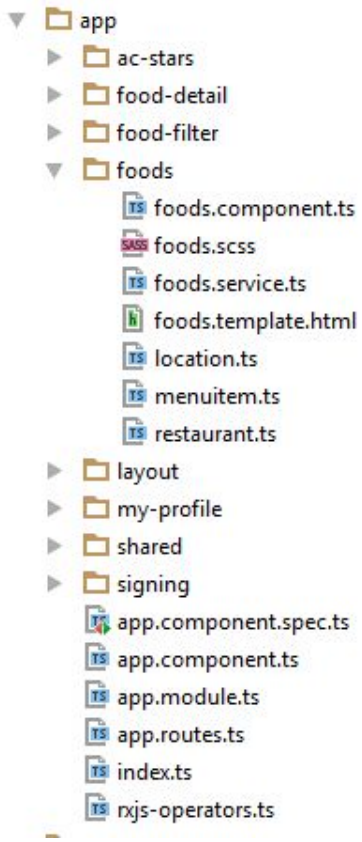


Celková infraštruktúra

1.3.1. Frontend

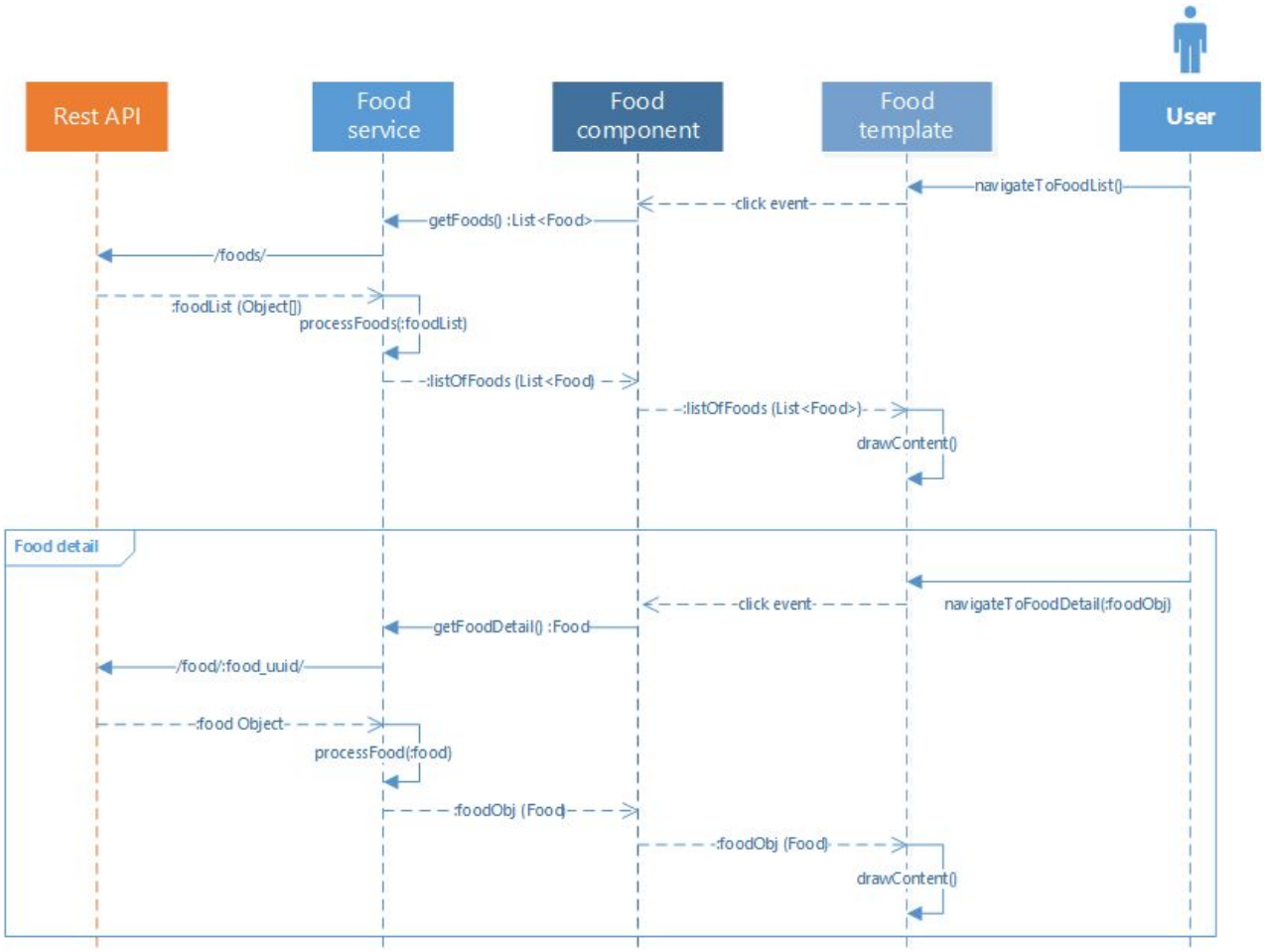
Frontendová časť aplikácie je postavená na modernej technológii *Angular 2* od spoločnosti *Google*, ktorá komunikuje s API za pomoci REST-ových služieb. Tento rámec nám prináša množstvo výhod, na základe ktorých sme sa pre neho rozhodli. Okrem jeho rýchlosti, jednoduchosti a flexibilite má rôzne stratégie, ako tvoriť aplikácie pre rôzne platformy, či už pre webové prehliadače alebo mobilné zariadenia. Kód je členený na jednotlivé komponenty, čím sa stáva prehľadný a ľahko zrozumiteľný. Aplikácia je postavená na *Angular-CLI*, spolu s čím prichádzajú výhody spojené s inštalovaním závislostí, testovaním a budovaním aplikácie (angl. build application) pre produkčnú alebo vývojársku verziu.

Na obrázku *Náhľad na štruktúrované priečinky podľa funkcie* je možné vidieť, ako je zdrojový kód rozdelený na priečinky podľa funkcií (angl. feature-like division), ktoré obsahujú komponent, HTML šablónu, službu, CSS štýly a používané triedy.



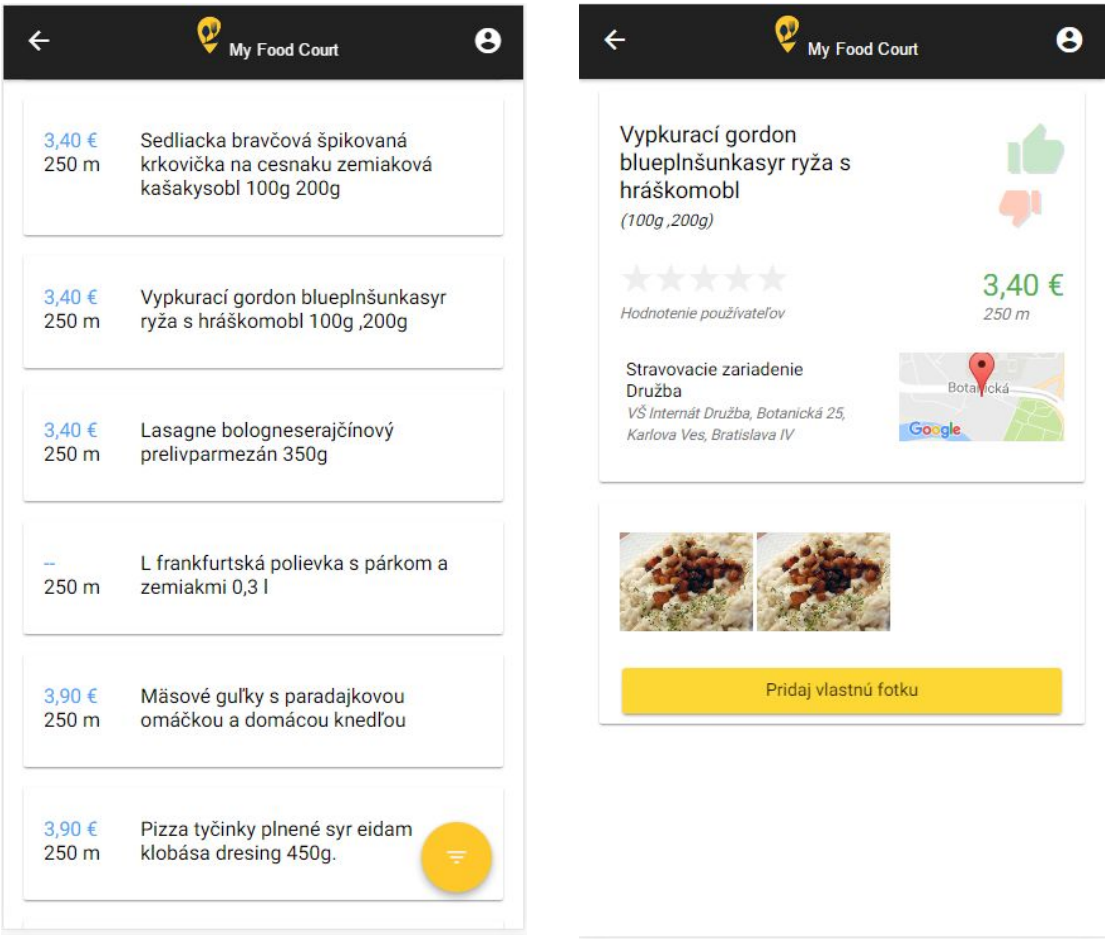
Náhľad na štrukturované priečinky podľa funkcie

Komponent slúži ako prostredník medzi šablónou, ktorá zobrazuje dáta a službu, ktorá komunikuje s API. Na obrázku je znázornený príklad, ako komponent pre zoznam jedál komunikuje so službou a šablónou.



Sekvenčný diagram pre zoznam jedál

Na ukážke nižšie môžme vidieť aktuálny zoznam jedál (naľavo) a detail jedného jedla (napravo). V hornej časti obrazovky je jednoduchý navigačný panel s preklikom naspäť na domovskú stránku a na *Môj profil*. Vnútro obrazovky je už vykreslený daný komponent - foods.component alebo foods-detail.component. Všetky obrazovky používaju karty na oddelenie jednotlivých informácií a celý dizajn je spravený v material dizajne, pričom sa používajú angular-material komponenty a knižnice.



Ukážka komponentov foods a foods-detail

Komponenty sú smerované za pomoci smerovača *Angular-Router*. V ukážke je znázornená hlavná časť súboru *app.routes.ts*, kde sú spísané všetky URL adresy a k nim prísluchajúce komponenty. Všetky komponenty, resp. URL adresy, okrem prihlasovacieho a registračného, nie sú dostupné pre neprihlásených používateľov. Trieda *LoggedInGuard* je zodpovedná o aktivovanie daného komponentu podľa prihlásenia.

```
export const routes = [
    {path: '', component: FoodsComponent, canActivate: [LoggedInGuard]},
    {path: 'food/:uuid/restaurant/:ruuid', component: FoodDetailComponent, canActivate: [LoggedInGuard]},
    {path: 'map/:ruuid', component: MapComponent, canActivate: [LoggedInGuard]},
    {path: 'login', component: LoginComponent},
    {path: 'registration', component: RegistrationComponent},
    {path: 'my-profile', component: MyProfileComponent, canActivate: [LoggedInGuard]},
    {path: '**', redirectTo: ''}
];
```

Definovanie URL adries pre komponenty

Služby komunikujú s API za pomoci HTTP požiadaviek. Každá služba si v konštruktore vytvára prostredníka *HTTPClient*, čo je nami vytvorená trieda pre podporu globálneho spracovania všetkých požiadaviek na API. Táto trieda rozširuje samotnú angularovskú HTTP službu. Dopĺňajú sa v nej potrebné hlavičky, taktiež sa týmto spôsob jednotne zaobchádza s chybami, alebo je vytvorený priestor pre znázorňovanie stavu požiadavky.

1.3.2. API

API v našej aplikácii predstavuje predovšetkým poskytovateľa REST služieb, teda poskytuje rozhranie na komunikáciu s databázou v pevne stanovenej a štruktúrovanej forme. Tieto služby využívajú ostatné časti systému, teda sťahovač dát a frontend aplikácia.

API je programovaná v jazyku PHP vo verzii 7. Využívame populárny rámec *Laravel 5.3*. Na ukladanie dát do perzistentnej pamäte využívame *PostgreSQL* databázu. Niektoré dáta sú indexované vo vyhľadávacom nástroji Elasticsearch.

Poskytované služby umožňujú ostatným aplikáciam manipulovať s entitami v databáze v *JSON* formáte. Takto dovoľuje vytvárať, upravovať a čítať informácie o reštauráciach, jedlách, denných ponukách a o používateľoch. Samotné entity sú popísané v časti Dátový model.

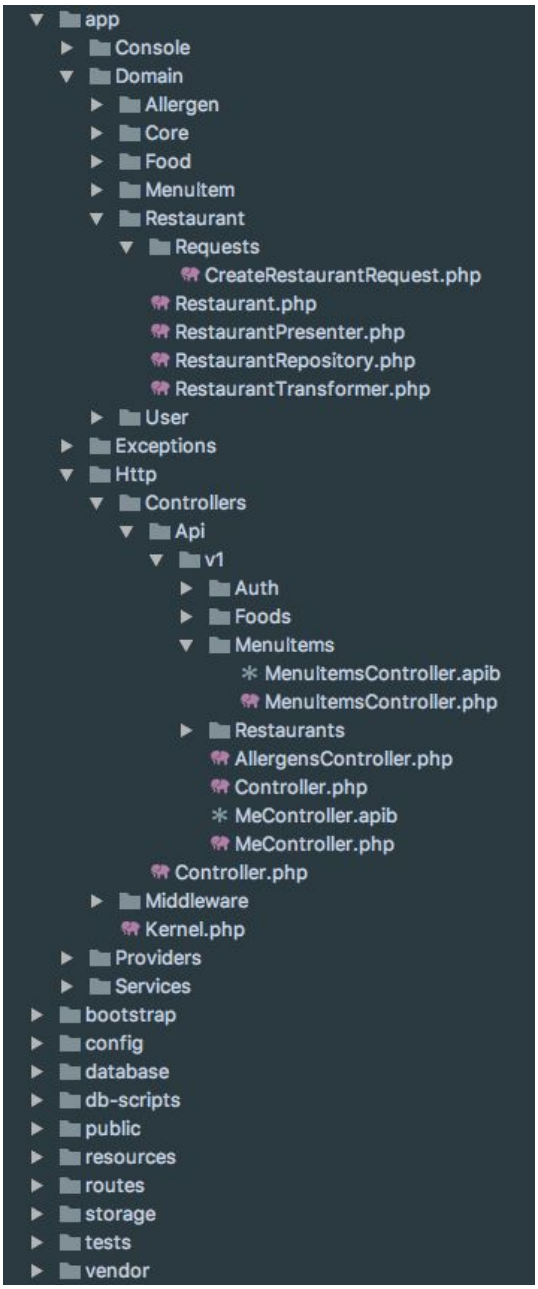
API tiež poskytuje služby na autentifikáciu, analýzu vstupov na posúdenie, či sa jedná o jedlo a tiež poskytuje možnosť prevziať dennú ponuku jedál podľa dátumu a lokality v štruktúrovanej forme tak, aby s ňou mohla frontend aplikácia jednoducho pracovať.

Štruktúra API je rozdelená na logické časti - komponenty. Pri aktualizovaní alebo pridávaní nových bodov pre prístup je veľmi ľahké nájsť časti, ktoré sú za to zodpovedné. Nižšie na obrázku *Štruktúra API* je zobrazené rozdelenie funkcionalít do priečinkov a logických častí.

Časť API zodpovedná za prácu s databázou sa nachádza v zložke *Domain*, kde sú všetky modely a kód pristupujúci alebo formátujúci výstupy a vstupy do databázy. Nachádza sa tu aj validácia dát.

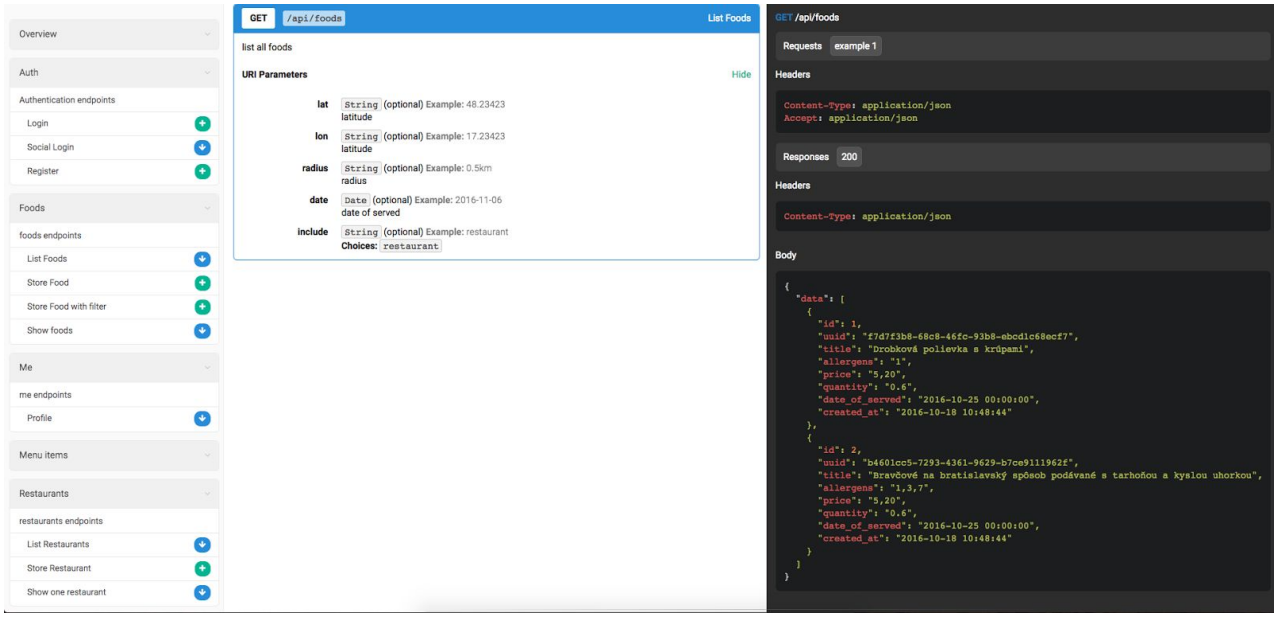
Časť starajúca sa o logiku API je v zložke Controllers, ktorá je taktiež rozdelená do zložiek podľa názvu komponentu. Tu sú zadefinované funkcie jednotlivých prístupových bodov API.

V zložke database sa nachádzajú migrácie (vytváranie databázovej schémy) a triedy, ktoré napĺňajú databázu dátami. Migrácie sú využívané pre verziovanie stavu databázy a pre rýchle a automatizované vytváranie zmien v databáze. Pre nové úpravy v databáze sa vytvorí nová migrácia, ktorá obsahuje kód prislúchajúci zmene databázy, a taktiež v prípade potreby je možné zmeny vrátiť prostredníctvom časti migrácie pre vrátenie zmien do pôvodného stavu.



Štruktúra API

Pre prácu s API je nutné poznať všetky jej prístupové body a informácie o vstupných a výstupných dátach. Na to nám slúži dokumentácia generovaná pomocou nástroja *Aglio*. Vďaka nej je možný rýchly prístup k informáciám, a to aké prístupové body API poskytuje (funkcionalita) a aké dáta sú nutné na ich získanie. Príklad je zobrazený na obrázku Dokumentácia API.



Dokumentácia API

1.3.3. Sťahovač dát

Sťahovač dát (ang. crawler) je aplikácia, ktorá získava dáta o ponuke jedál, ktoré sú dostupné na webových sídlach tretích strán. Dáta sú získavané pravidelne vždy začiatkom týždňa na čo najdlhšie možné obdobie dopredu - spravidla jeden týždeň.

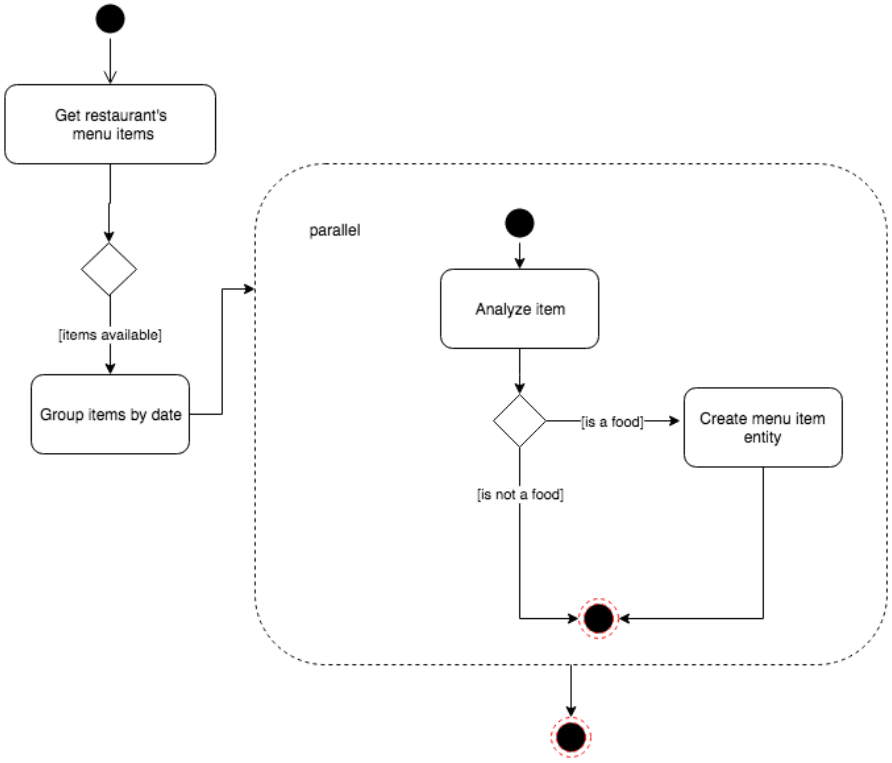
Sťahovač funguje ako samostatná aplikácia písaná v jazyku JavaScript fungujúca na softvérovom systéme *Node.js*. Tento systém sme pre sťahovač vybrali kvôli neblokujúcej udalosťami riadenej architektúre, ktorú využíva. Tá nám umožňuje paralelne spracovávať viacero webových stránok súčasne, a tak spracovanie dát nespomaľuje dlhá odozva webových sídiel.

Dostupné webové lokality a aj naše REST služby majú obmedzenia v počte požiadaviek, ktoré dokážu obslúžiť v danej jednotke času. Preto sme implementovali radu, ktorá je spracovávaná postupne s časovým odskokom medzi požiadavkami tak, aby sme nezahltili cieľové koncové body. Kedže je každá webová stránka spracovávaná asynchrónne a úkony sú riadené udalosťami, dokázali sme navrhnúť efektívny spôsob získavania a ukladania dát, pričom stále rešpektujeme obmedzenia na počet požiadaviek, ktoré môžeme v danom časovom úseku vytvoriť.

Sťahovač môže pracovať v dvoch módoch. Je ho možné spustiť tak, aby objavoval nové entity reštaurácií, ktoré ukladá do databázy prostredníctvom API. Tento variant je však pomalší, pretože vyžaduje vykonať niekoľkonásobne viac HTTP požiadaviek na získavanie aj ukladanie dát. Druhou možnosťou je spustiť sťahovač v režime, v ktorom najprv prostredníctvom REST

služby prevezme zoznam známych reštaurácií. Ďalej pristupuje už iba priamo na webovú stránku obsahujúcu ponuku jedál pre danú reštauráciu.

Reštaurácie nedodržiavajú štruktúru dennej ponuky a často do zoznamu jedál vkladajú aj iné oznamy. Preto sú jednotlivé položky dennej ponuky odosielané do API na posúdenie, či ide o názov jedla alebo nejaký oznam alebo iný text nesúvisiaci s ponukou reštaurácie. Položiek je veľké množstvo, a preto sú spracovávané paralelne.



Paralelné spracovanie dennej ponuky pre jednu reštauráciu

Sťahovač je spúšťaný pomocou programu *CRON* pravidelne začiatkom každého týždňa, pričom je spúšťaný viackrát s časovým odstupom. Reštaurácie nezverejňujú svoje denné ponuky v rovnakom čase, a tak sa môže stať, že v čase prvého sťahovania dát nebudú dostupné všetky údaje. Preto je sťahovač spúšťaný viac krát, aby sme získali všetky dáta a aby boli čo najaktuálnejšie.

Pre zaručenie behu jednej inštancie sťahovača používame uzamykací súbor, ktorý slúži k tomu, aby sťahovač nemohol byť spustený naraz niekoľko krát. Takýto prípad by zbytočne duplikoval požiadavky na uloženie dát do API a sťahovanie údajov zo stránok. Uzamykací súbor pracuje tak, že pri spustení prvej inštancie sťahovača sa vytvorí a obnovuje sa v určitých malých časových intervaloch (napr. 1s). Ak nastavený CRON spustí znovu ďalšiu inštanciu sťahovača, ten

skontroluje, či je vytvorený uzamykací súbor. Ak áno, znovu sa nespustí. Avšak v prípade nekorektného ukončenia prvej inštancie sťahovača by bolo možné, že sa uzamykací súbor nezmaže, a zapríčiní tak nespustenie nových inštancií. V takomto prípade sťahovač kontroluje aj poslednú aktualizáciu uzamykacieho súboru. Keďže je počas behu sťahovača obnovovaná v malom časovom intervale, je možné s určitosťou povedať, že žiadna inštancia sťahovača nie je spustená a je možné opätovne spustiť novú inštanciu.

1.3.4. Dátový model

Databázovy server PostgreSQL sa nachádza na FIIT Cloud serveri. V momentálnom stave nášho produktu pracujeme primárne so šiestimi entitami. Dáta sú aktualizované na týždennej báze, vždy s príchodom nových obedových menu lístkov.

Všetky entity obsahujú atribúty *uuid* (generovaný jedinečný identifikátor), *created_at* (časový údaj vytvorenia záznamu) a *updated_at* (časový údaj zmenenia záznamu).

Foods

Základná entina, resp. tabuľka v databáze. Ukladajú sa v nej údaje o jedlách, ako je ich názov a zoznam alergénov. Táto tabuľka sa mapuje na základe cudzieho kľúča v tabuľke *Menu items.*

Restaurants

Reštaurácia obsahuje základne údaje o adrese, názve a tom, kde presne sa nachádza v podobe GPS koordinátov.

Menu items

Entita vyjadruje základný vzťah medzi jedlom (Foods) a reštauráciou (Restaurants). Nakoľko môže mať viacero reštaurácií v ponuke tie isté jedlá je nutné mať tento vzťah vyjadrený touto tabuľkou. Teda v tejto tabuľke sú okrem cudzích kľúčov na tabuľky *Foods* a *Restaurants* aj cena, kvantita a dátum podávania jedla v reštaurácii.

Social profiles

Sociálne profily používateľov. Momentálne evidujeme Google a Facebook. Pri prihlásení prostredníctvom týchto profilov, sa používateľovi vytvorí aj záznam v tabuľke *Users.* Obsahuje atribúty ako typ sociálnej služby v podobe jedinečného identifikátora, jej popis a názov.

Users

Entita určená pre uchovávanie údajov o používateľoch. Záznam v tejto tabuľke sa vytvára pri prihlásení používateľa prostredníctvom sociálnych sietí, a takisto pri jeho registrácií. Okrem

základných používateľskych informácií si uchovávame aj tzv. *token*, ktorý slúži ako bezpečnostný prvok pri komunikácií s RESTovými službami.

Allergens

Zoznam alergénov. Vzťah s entitou *Foods* neexistuje z dôvodu, že nie je nutné si uchovávať tento vzťah, nakoľko alergény sú uložené ako postupnosť znakov v entite *Foods*. V tejto tabuľke sa nachádzajú len statické dáta k bližšej špecifikácií jednotlivých alergénov.

Allergen menu item

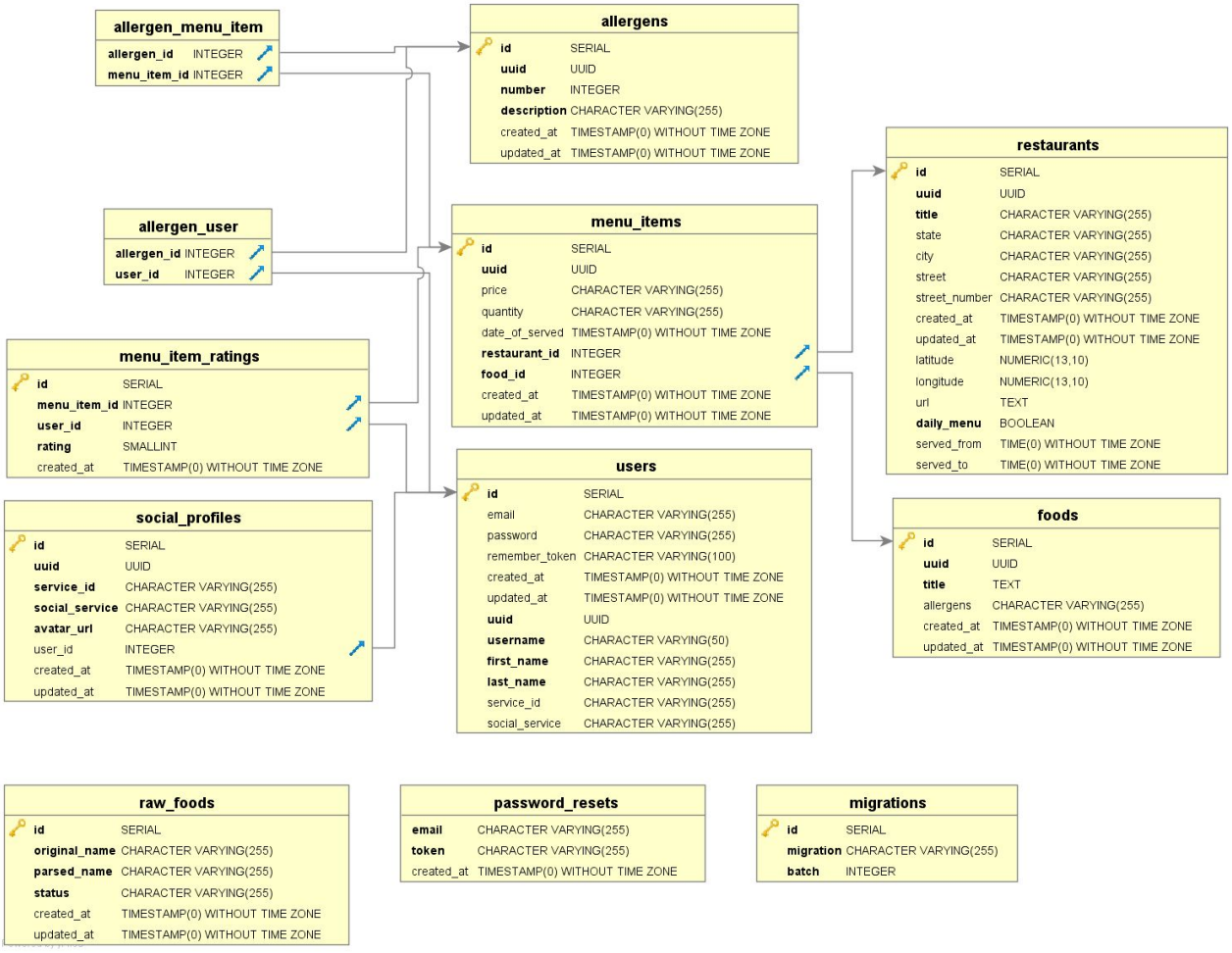
Väzobná tabuľka medzi alegrénmi a samotným jedlom s jeho alegrénmi. Tabuľka obsahuje iba dva cudzie kľúče vytvárajúce väzbu.

Allergen user

Väzobná tabuľka medzi používateľom a alergénmi, ktorá nám umožňuje držať informáciu, ktorý používateľ má ktoré alergény. Tento vzťah je dôležitý pre ďalšie odporúčanie jedál. V tabuľke sa nachádzajú iba dva cudzie kľúče tvoriace väzbu medzi entitami.

Menu item ratings

Väzobná tabuľka zachytávajúca feedback od používateľov na dané jedlo v určitej reštaurácii. Tabuľku tvorí väzba medzi entitami *menu_items* a *users*, obsahuje tiež hodnotu hodnotenia a časovú pečiatku.



Dátový model

1.3.5. Server

Server zabezpečuje beh všetkých aplikácií online. Na serveri je nainštalovaný operačný systém *Ubuntu 16.04*, ktorý poskytuje jednoduchú správu a inštaláciu aplikácií, bezpečnosť a zabezpečuje stály a nerušený chod.

Pre zobrazovanie webových aplikácií používame webový server *Nginx*. Pre konfiguráciu nie je jednoduchší ako známy HTTP server Apache, ale poskytuje viac priestoru pre konfiguráciu a je od začiatku stavaný pre lepší manažment zdrojov. Využívame aj jeho vstavaný proxy server, ktorý potrebujeme pri smerovaní na interné aplikácie. Inštalácia pre operačný systém Ubuntu je veľmi jednoduchá a konfigurácia Nginx je prehľadná a intuitívna. Nginx priamo spracováva všetky požiadavky pre API a samotnú aplikáciu, no zodpovedá aj za aplikácie, ktoré pomáhajú pri manažovaní. Jedná sa o jednoduché databázové rozhranie *Adminer* či Wiki (*BookStack*).

Požiadavky pre Java aplikáciu *Jira* tiež spracováva Nginx. Vďaka nemu môžeme pristupovať na webové aplikácie pomocou krajších URL adries bez potreby zadávania portov.

Ako databázové riešenie pre aplikáciu sme vybrali *PostgreSQL,* pretože je možné nainštalovať rozšírenie pre prácu s mapovými objektami a všetky systémy pre manažment túto databázu podporovali. Wiki však pre svoj chod potrebovala databázu *MySQL,* a preto sever disponuje oboma typmi databáz.

Jednou z veľmi dôležitých aplikácií nainštalovaných na našom serveri je softvér pre kontinuálne nasadzovanie a správu našich aplikácií Bamboo od spoločnosti Atlassian. Je veľkým nápomocníkom pri nahrávaní nových verzí zdrojových kódov a testovaní aplikácii. API, sťahovač aj aplikácia sú vďaka nemu vždy aktuálne a Bamboo zabezpečuje, že sú aj plne fuknčné. Pri zachytení novej zmeny z verziovacieho systému GitHub Bamboo stiahne nové zmeny na server a vykoná potrebné kroky. Pri aplikáciach spísaných v jazyku PHP sa jedná o inštaláciu nových závislostí cez Composer a v prípade našej aplikácie Npm. Stará sa tiež o automatické testovanie funkčnosti pri nasadení nových verzií.

Na serveri sú nainštalované tri aplikácie pre manažment (Jira, Bamboo, BookStack) a naše časti aplikácie. Taktiež máme databázové systémy (PostgreSQL, MySQL) a vyhľadávací engine Elasticsearch.

1.4. Testovanie

Testovanie jednotlivých častí aplikácie prebieha na viacerých úrovniach. Každá časť (API, frontend, sťahovač) je testovaná inak. Keďže jadrom celej aplikácie je získavanie, ukladanie, posielanie údajov, najviac automatizovaných testov obsahuje API.

Pre jednoduchú správu testovania a nasadzovania využívame *CI Bamboo* od firmy Atlassian. Bamboo je veľmi dôležitou súčasťou pri nasadzovaní a testovaní jednotlivých častí aplikácie. Po zmene v repozitári stiahne nové zmeny a nasadí na server. Akonáhle je nová verzia stiahnutá, spúšťa automatizované testy. V našom prípade to sú *PHPUnit* testy v zložke tests v API repozitári. Po otestovaní odošle informáciu do slacku o stave testov. V Bamboo je taktiež možné po neúspešnom testovaní (aspoň 1 neúspešný test) vrátiť stav aplikácie do predchádzajúceho funkčného stavu. Tým zabezpečíme funkčnosť aplikácie v každom okamihu.

Testovanie však prebieha aj u každého člena tímu v lokálnom prostredí. Pre zabezpečenie správnosti aplikácie je nutné, aby všetky testy boli úspešné už v lokálnom prostredí.

Pre frontend a ostatné časti aplikácii nie sú zatiaľ dostupné automatizované testy z dôvodu využitia času pre iné časti aplikácie, ktoré sme spolu s product ownerom zhodnotili ako dôležitejšie.

1.5. Technická dokumentácia

Jedna zo súčastí architektúry našej aplikácie je API, bližšie opísaná v časti 1.3.2. API. Každá dobrá API musí byť popísaná a musí obsahovať dokumentáciu k jednotlivým službám, ktoré vykonáva. V našom prípade používame štandard *API Blueprint*, ktorý je následne pregenerovaný do vizuálnej podoby HTML kódu pomocou nástroja Aglio.

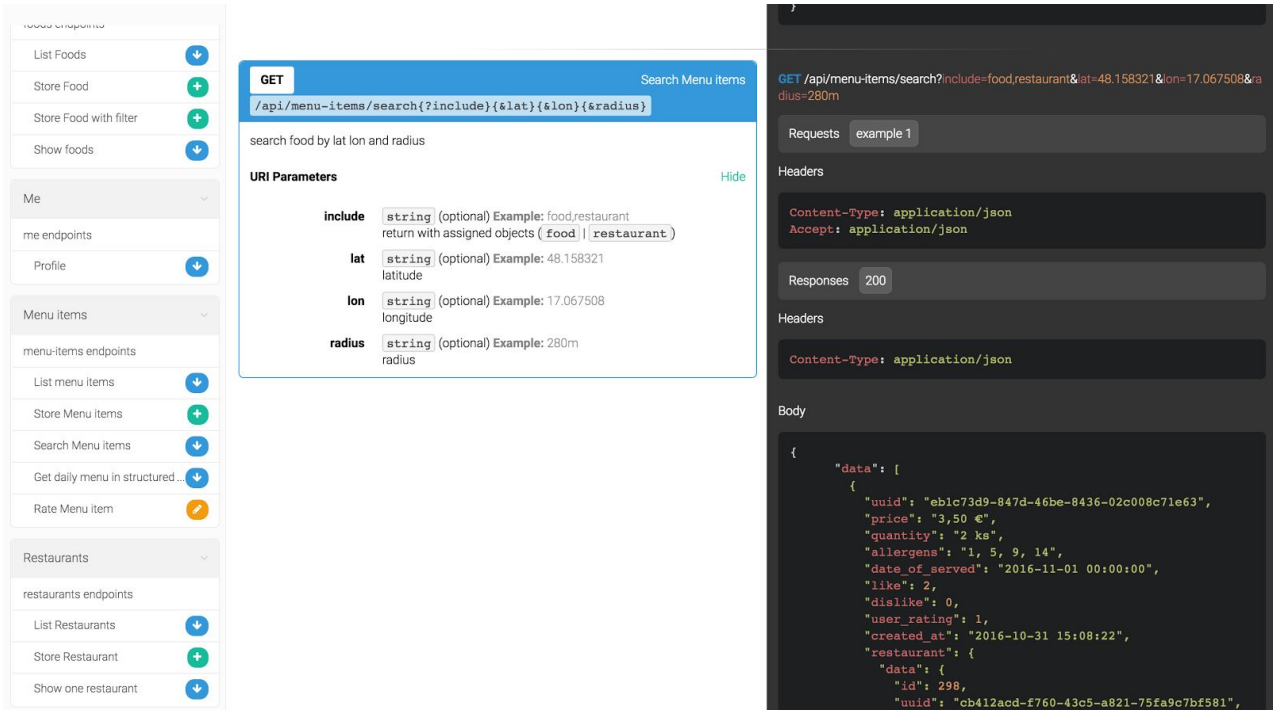
Výhodou takéhoto riešenia je, že dokumentácia sa nachádza priamo v rámci aplikácie a nie je potrebné otvárať inú službu. Popis služieb (controllerov) sa vždy nachádza hneď vedľa controlleru v samostatnom súbore pomenovanom rovnako ako controller, iba s príponou .apib.

Dokumentácia je teda súčasť štruktúry aplikácie a dopĺňa ju programátor v rámci implementácie konkrétnej úlohy. Pred odovzdaním úlohy musí programátor spustiť príkaz `php artisan docs:generate`, ktorý automaticky pre-generuje dokumentáciu. Nová dokumentácia potom ide v rámci samostatného commitu do repozitára.

Na testovanie endpointov a služieb využívame aplikáciu `paw` prípadne `postman`. Obe tieto aplikácie po nainštalovaní rozšírenia poskytujú export volania služby vo formáte API Blueprint. Z týchto exportov napĺňame súbory .apib, čo nám značne uľahčuje vytváranie dokumentácie.

```
## Search Menu items [GET /api/menu-items/search{?include}{&lat}{&lon}{&radius}]
search food by lat lon and radius

+ Parameters
    + include: food,restaurant (optional) - return with assigned objects (`food` | `restaurant`)
    + lat: `48.158321` (optional) - latitude
    + lon: `17.067508` (optional) - longitude
    + radius: 280m (optional) - radius

+ Request (application/json)

    + Headers

            Accept: application/json

+ Response 200 (application/json)

    + Body
            {
              "data": [
                {
                  "uuid": "eb1c73d9-847d-46be-8436-02c008c71e63",
                  "price": "3,50 €",
                  "quantity": "2 ks",
                  "allergens": "1, 5, 9, 14",
```

Príklad súboru .apib popisujúceho službu



Príklad vygenerovanej časti dokumentácie

1.6. Príručky

Kedže nie každý člen tímu ovláda všetky súčasti a artefakty architektúry aplikácie, vytvorili sme si niekoľko návodov resp. príručiek. Tieto dokumenty sú obsiahnuté v spoločnej WIKI. Príručky popisujú ako si nastaviť prostredie, prípadne nejaký nástroj.

Elastic search

Obsahuje informácie. kde a akým spôsobom je nástroj nainštalovaný, ako je možné k nemu pristúpiť a ako ho používať. Príručka sa nachádza v prílohe D 6.2. Elasticsearch.

Angular App

Táto príručka slúži ako jednoduchý úvod do vývoja frontendovej časti aplikácie v Angualr frameworku. Príručka bola dôležitá hlavne na začiatku vývoja a pre členov tímu, ktorí sa začínali zoznamovať s frameworkom. Príručka sa nachádza v prílohe D 6.1. Getting started - Angular app.

2. Inžinierske dielo LS

- "Big picture"
  - *Úvod - o čom je tento dokument, ciele, ohraničenia.*
  - *Globálne ciele projektu na letný semester*
  - obsahuje aj celkový pohľad po zimnom semestri - vrátane priorít riešenia
  - *Celkový pohľad na systém*
  - (na túto časť je nutné v letnom semestri dávať najvyšší dôraz, je to najdôležitejšia časť dokumentácie, ktorá zhŕňa všetky výsledky projektu, zahŕňa podľa typu projektu opis prototypu/produktu ako celku, opisuje architektúru, modely, celkovú funkcionalitu, ohraničenia plus referencie/zoznam priložených e-dokumentov)
- Moduly systému
  - Analýza
  - Návrh
  - Implementácia
  - Testovanie
- Príručky (podľa potreby)
- Technická dokumentácia (aj generovaná)

2.1 Úvod

V tomto dokumente je popísaný výsledok nášho úsilia reprezentovaný funkčnou aplikáciou. V prvej časti sú zadefinované ciele projektu, ktoré sme si stanovili začiatkom semestra. Vo zvyšnej časti dokumentu je popísané, ako sa nám tieto ciele podarilo naplniť.

V dokumente je popísaný celkový pohľad na systém, kde sa zameriavame na architektúru a implementáciu jednotlivých funkcionalít v systéme. Každá vlastnosť aplikácie je opísaná z pohľadu frontendu, ale aj backendu. Tiež je k nim pridaná aj analýza, návrh a spôsob testovania.

2.2 Celkový pohľad na systém

2.2.1 Globálne ciele projektu

Zimný semester sa niesol v duchu riešenia nášho asi najväčšieho problému, čím boli dáta. Získavané dáta z dostupných webových miest boli do značnej miery neštruktúrované, každá reštaurácia uvádzala svoju ponuku rozdielnym spôsobom. Kedže našim cieľom bolo v rámci zimného semestra vytvoriť agregátor ponuky jedál v používateľovom okolí, potrebovali sme dáta dostať do vizuálne priaznivej podoby. Riešenie tohto problému sa dostalo na úroveň, kedy si

vieme dynamicky pridávať pravidlá pre konkrétne reštaurácie, s ohľadom aj na prioritu pravidiel, či už sa jedná o globálne pravidlá pre všetky gastronomické prevádzky alebo konkrétne zariadenia. Toto pokročilé riešenie nám zabezpečilo požadovaný tvar jedál a mohli sme začiatkom roka urobiť prvý verejný release aplikácie.

Na začiatku letného semestra sme si definovali víziu, kam smerujeme a čo všetko chceme do konca semestra stihnúť. Samozrejme, naplánovali sme si aj hlavné body, ktoré musíme zvládnuť na ceste za našou víziou.

Kedže už na začiatku tohto predmetu sme s témou chceli vytvoriť personalizovanú ponuku jedál, jednohlasne sme sa zhodli, že musíme zvládnuť:

1. základnú klasifikáciu jedál,
2. vytvorenie prvého modelu používateľa, aj keby mal byť spočiatku veľmi jednoduchý,
3. vytvorenie rôznych odporúčačov, podľa ktorých sa bude upravovať zoradenie ponuky jedál.

Tieto body sa stali našou hlavnou prioritou aj s ohľadom na to, že sme chceli využiť vynikajúce poznatky nášho vedúceho projektu z oblasti personalizácie a odporúčania.

Na ceste za našou víziou sme identifikovali niekoľko čiastkových cieľov, bez ktorých nebudeme schopní stanovenú víziu aj naplniť.

1. Profil používateľa
    1.1. registráciu,
    1.2. prihlásenie,
    1.3. sledovanie aktivity v aplikácii,
    1.4. možnosť definovať si prípadné intolerancie, resp. alergény v rámci profilu.
2. Distribúcia aplikácie medzi ľudí
    2.1. anasoft,
    2.2. študenti.
3. Získanie spätnej väzby
    3.1. Implicitá
    3.2. Explicitná

Používateľ nám tak odovzdá čo najviac informácií o možnostiach v stravovaní. Aby sme však aj nejakých používateľov získali, budeme potrebovať rozšíriť povedomie o našej aplikácii medzi širšiu verejnosť.

Ďalším dôležitým krokom bude získanie spätnej väzby, či už priamo cez aplikáciu v podobe hodnotenia jedla, alebo všeobecne na aplikáciu ako takú. Takúto všeobecnú spätnú väzbu sme explicitne získavali od používateľov, ktorým sme aplikáciu doručili na testovanie.

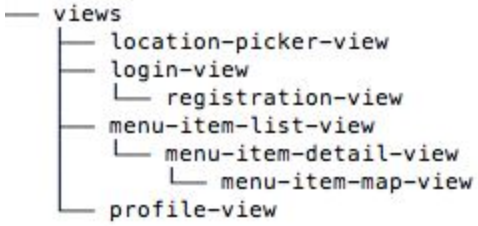
Jasne definovaná vízia a jednotlivé body na ceste za touto víziou nám pomohli k tomu, že sme dokázali veľkú časť našej vízie naplniť. Jediným cieľom, ktorý sa nám nepodarilo úplne naplniť, je vytvorenie modelu používateľa.

2.2.2 Opis architektúry

2.2.2.1 Frontend

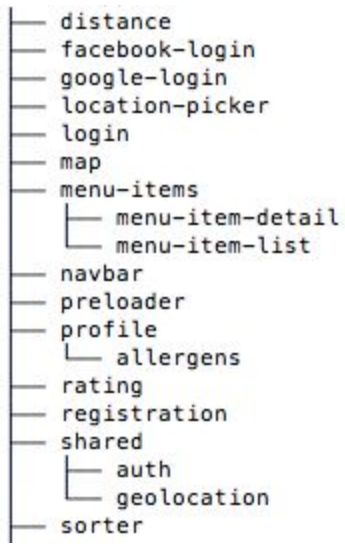
Už začiatkom letného semestra sme sa rozhodli vykonať refaktoring kódu vo frontendovej časti aplikácie. Hlavným dôvodom bola zle navrhnutá pôvodná štruktúra, ktorá bola z dlhodobého hľadiska neudržateľná, a taktiež nás obmedzovala v možnostiach kontrolovania niektorých vlastností aplikácie. Pri pridávaní novej funkcionality sa kód stával neprehľadným a nevhodne previazaným. Išlo najmä o vzťahy medzi jednotlivými komponentami. Motiváciou bolo teda navrhnúť novú štruktúru kódu, ktorá bude jednoduchšia na pochopenie, a tiež vytvorí vhodné prostredie na pravidelné zavádzanie novej funkcionality. V poslednom rade sme tiež zvolili nový spôsob vytvárania používateľského prostredia, ktorý nám ušetril množstvo času pri navrhovaní, ale aj samotnom vyvíjaní grafickej vrstvy aplikácie. Grafické spracovanie aplikácie malo tiež priniesť pokrok, keďže vďaka novému dizajnu aplikácia dáva pocit profesionálnej natívnej aplikácie.

Najinvazívnejšou zmenou prešla štrukúra kódu. Hlavnou ideou zmeny štruktúry bolo oddeliť behaviorálne a štrukturálne komponenty aplikácie. Vytvorili sme štrukturálne komponenty, ktoré predstavujú jednotlivé obrazovky v aplikácii, pričom je dodržaná hierarchia, ktorá naznačuje mapovanie obrazoviek. Zo štruktúry teda intuitívne vyplýva, na akú obrazovku sa používateľ môže ovládaním aplikácie dostať z aktuálne viditeľnej obrazovky.



Mapa obrazoviek, resp. štruktúru priečinkov predstavujúcich obrazovky aplikácie

Tento prístup nám okrem zvýšenia kvality kódu priniesol aj iné výhody v možnostiach používateľského rozhrania. Zmenili sme spôsob, akým sa zobrazujú jednotlivé obrazovky. V súčastnosti sa každá obrazovka vnára do jej nadradenej obrazovky a počas renderovania plynulou animáciou prekryje grafickú vrstvu tejto rodičovskej obrazovky. Pri návrate späť na predošlú, resp. rodičovskú, obrazovku sa opačnou animáciou spätne odkrýva jej vrstva. Nie je teda viac potrebné nanovo prepočítavať predošlý stav obrazovky, ktorý používateľ zanechal (napr. ak sa používateľ posúval v zozname jedál). Taktiež nie je potrebné znovu sťahovať dáta pre danú obrazovku, resp. vykonávať zložitý manažment dočasnej pamäte. Zvýšili sme tým aj úroveň používateľského zážitku, pretože aplikácia v súčasnom stave pôsobí plynulejšie, rýchlejšie a pamätá si zmeny v predchádzajúcich stavoch obrazovky, ktoré používateľ vykonal, a očakáva ich po návrate späť z predošlej obrazovky.



Hierarchia behaviorálnych komponentov v aplikácii

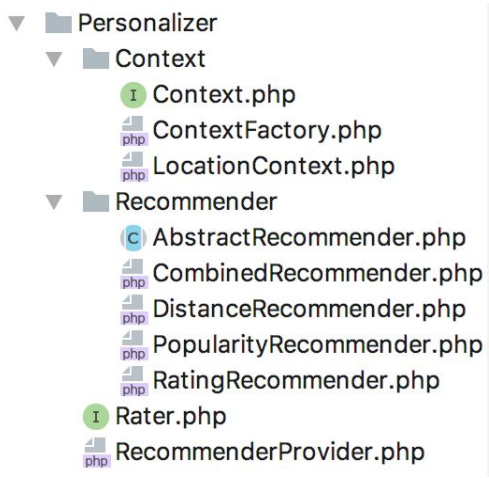
Ostatné komponenty, teda behaviorálne komponenty a k nim prislúchajúce služby a direktívy, boli rozdelené na menšie znovupoužiteľné služby. Aplikovali sme tak architektúru podľa vzoru *Microservice Architecture pattern*. Táto zmena priniesla výhody pre vývojárov v podobe

preh ľadnejšieho kódu s jasne definovanými hranicami funkcionality a preväzovania jednotlivých komponentov. Kód sa takto stal menej chybovým, ľahšie udržiavate ľným a rýchlosť dodávania novej funkčnej funkcionality sa rapídne zvýšila.

Ako už bolo spomenuté, aplikácia zmenila aj svoj vzh ľad. Motiváciou však nebolo dosiahnuť iba používate ľsky príjemnejšie rozhranie. Táto zmena mala aj praktické dôvody, ktoré nám pomohli ušetriť množstvo času. Na skladanie nových obrazoviek sme využili nami zakúpenú fungujúcu galériu konštrukčných grafických prvkov, ktoré sme mohli v našej aplikácii použiť a nemuseli sme ich vyvíjať. Ušetrili sme tak čas, ktorý bolo potrebné investovať do navrhovania používate ľského rozhrania, a tiež aj jeho programovania.

2.2.2 API

Pri navrhovaní nových funkcionalít systému sme dbali na využite návrhových a dizajnových vzorov, aby sme tak vytvorili robustné, no ľahko rozšírite ľné riešenie.



Ukážka štrukturálneho rozloženia modulu odporúčania

Vývoj nových funkcionalít ovplyvnil pridávanie nových prístupových bodov a rozširovanie API dokumentácie pre jednotlivé prístupové body. Každý prístupový bod musí byť vytvorený a opísaný v dokumentácií.

Počas vývoja sme aktualizovať používané knižnice a aj samotný rámec (*angl. framework*) na najnovšie verzie. S touto aktualizáciou používame aj najnovšiu verziu nástroja ElasticSearch 5.3.

2.2.3 Sťahovač dát

Na sťahovači dát sme vykonali optimalizáciu pri parsovaní dát a HTTP volaniach a dokázali sme optimalizovať rýchlosť parsovania na 4 minúty, čo je zlepšenie o približne 60%. Sťahovač dát bol

navrhnutý tak, aby mohol byť jednoducho rozšíriteľný o adaptér na parsovanie dát aj z iného webového zdroja. Tieto webové zdroje sme rozšírili o webové stránky študentských jedální Eat&Meet a VENZA.

2.2.4 Dátový model

V databázovom modeli došlo k niekoľkým zmenám z dôvodu rozvoja funkcionality aplikácie.

Menu item views

V tabuľke sa ukladá aktivitu používateľov a ich kliknutia na jednotlivé položky v zozname jedál. Tabuľka tiež zachytáva väzbu medzi používateľom a zoznamom jedál *user_id*, *menu_item_id*, takisto pozíciu jedla v liste *list_position* a polohu používateľa *latitude*, *longitude*.

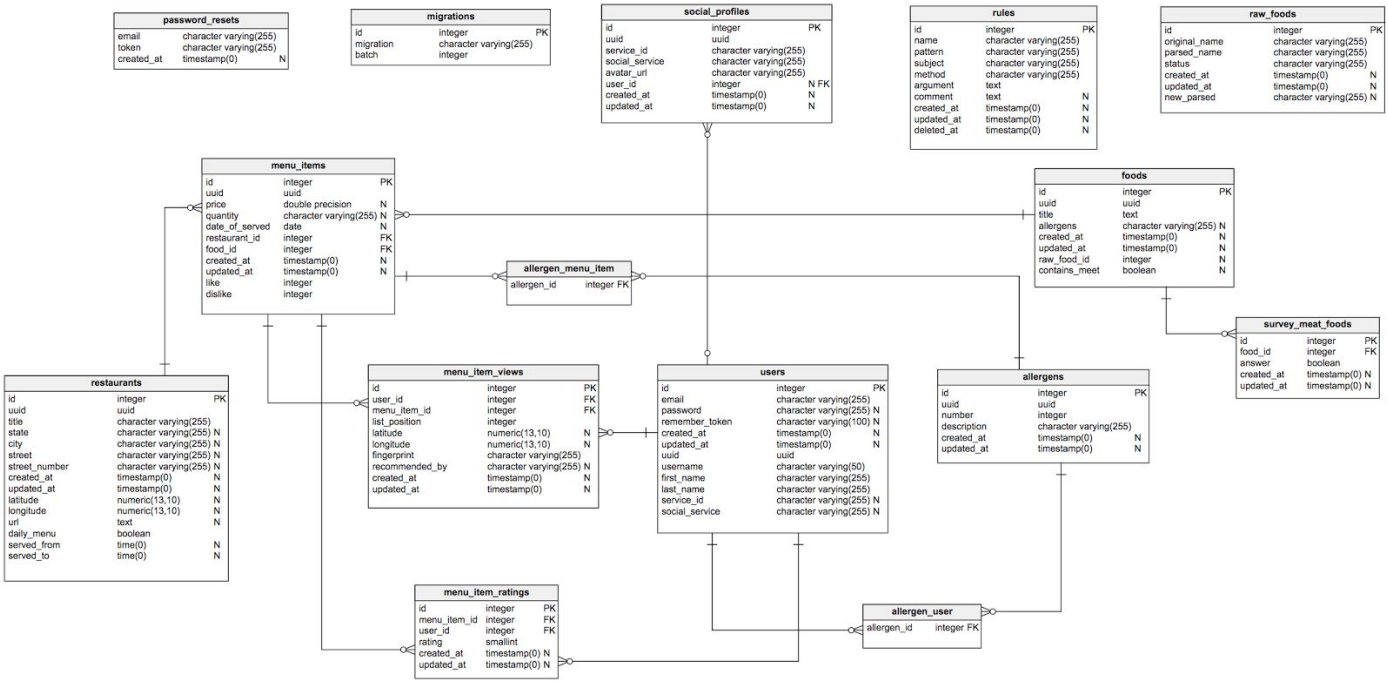
Rules

V samostatnej tabuľke *rules* sa ukladajú regulárne výrazy, ktoré slúžia na predspracovanie názvov jedál. Dôležité atribúty tabuľky sú *pattern*, ktorý obsahuje samotný regulárny výraz a *method*, podľa ktorého sa volá metóda na spracovanie tohto výrazu.

Survey meat foods

Do tabuľky zapisujeme posudky od ľudí z aplikácie *rozhodni.pokope.sk*, kde môžu rozhodnúť o tom, či nejaké jedlo obsahuje mäso alebo nie. V tabuľke sa nachádza cudzí kľúč *food_id* ako referencia na jedlo a atribút *answer*, v ktorom je uložená odpoveď *true* alebo *false*.

V niektorých už existujúcich tabuľkách došlo k úprave dátových typov, prípadne doplnenie atribútov. Najdôležitejšie úpravy nastali v tabuľke *menu_items*, kde boli pre optimalizáciu pridané atribúty *like* a *dislike*. V tabuľke *foods* sme doplnili atribút *contains_meat*, ktorý určuje, či jedlo obsahuje mäso alebo nie.



Dátový model

2.2.5 Server

Serverové riešenie prešlo aktualizáciou spolu s manažmentom celého tímového projektu. Virtuálny server Ubuntu má nainštalované takmer všetky aplikácie, ktoré boli využívané v zimnom semestri. Zmeny v nastali hlavne v službách pre nasadzovanie a deployment.

Aplikácia Bamboo od spoločnosti Atlassian bola koncom zimného semestra vyradená a funkcionalita, ktorú sme využívali, sme nahradili vlastným riešením spolu s testovacím nástrojom Travis. Bamboo nám slúžilo na jednoduchú administráciu projektov, ktoré priamo odkazovali na repozitár vo verziovacom systéme. Pomocou týchto projektov boli vždy najnovšie verzie častí aplikácie nasadené a otestované. Keďže Bamboo nepodporuje jednoduchý spôsob testovania a následného zobrazenia stavu testov v službe GitHub, rozhodli sme sa túto službu nahradiť lepšou. Travis je služba ktorá podporuje priamu integráciu GitHub repozitárov a poskytuje rôzne nastavenia samotného testovania ako aj akcií v prípade stavov testov.

Na serveri sme teda vytvorili priečinky pre nasadzovanie a deployment častí aplikácie a pre Travis sme k nim vytvorili prístup. Travis po úspešnom zbehnutí všetkých testov nasadí aplikáciu ku nám na server. Je to z dôvodu, že Travis je služba, ktorá beží na vlastných serveroch, a teda nie je možné ju nainštalovať ako Bamboo na náš server. V prípade, že otestovanie aplikácie neprebehne v poriadku, Travis označí vetvu v GitHube ako problémovú a nie je možné ju nasadiť

do testovacej aplikácie. Vďaka tejto optimalizácii sa nám podarilo rýchlejšie a hlavne bezchybnejšie vydávať nové verzie aplikácie.

Ďalšou aktualizáciou prešiel aj systém ElasticSearch. Keďže v rámci API framework Laravel prešiel aktualizáciou, bolo potrebné aktualizovať aj verziu ElasticSearch. Bolo potrebné všetky dáta zálohovať a nainštalovať najnovšiu verziu. Po úspešnej inštalácii bolo nutné vytvorenie indexov a vloženie dát. Momentálne teda využívame ElasticSearch vo verzii 5.2 (predtým 2.4).

Kvôli obmedzeniu funkcionality získavania polohy v prehliadači používateľa iba pod bezpečnostným certifikátom, sme boli nútení pre našu aplikáciu vytvoriť subdomény a vygenerovať dostupný LetsEncrypt certifikát pre API a aplikáciu. Vďaka oddeleným doménam sa nám prvky ukladané do prehliadača nemiešali s aplikáciami manažmentu a vďaka certifikátom môžeme získavať polohu používateľa prostredníctvom prehliadača.

2.2.6 Funkcionalita

Celková funkcionalita, ktorá vznikla počas celého projektu sa dá rozdeliť do viacerých bodov. Výsledkom spoločného úsilia nie je čisto iba aplikácia, ktorá je dostupná na webe, ale aj bočné úlohy. Rozdelenie je nasledovné:

- Sťahovač dát z webu
- Parser
- API na backende
- Odporúčacie a klasifikačné moduly zapojené do backendu
- “Admin” rozhranie pre možnosť jednoduchého triedenia jedál
- Hlavná aplikácia dostupná na webe

Vytvorenie modulu pre sťahovanie dát je esenciálna záležitosť projektu. Dáta získané z webových stránok, ktoré zobrazujú denné menu pre reštaurácie sú spracované v ďalšom kroku. Je potrebné vytvoriť viacero sťahovačov, keďže dáta sa nachádzajú na rôznych stránkach a v rôznych formátoch. Pre každú individuálnu stránku je vytvorený jeden sťahovač, ktorý odosiela dáta na server.

Potom, čo sú dáta získané z webu odoslané na server, sú spracované parserom, ktorý dokáže získať informácie ukryté v reťazcoch jedál. V názvoch jedál sa často ukrývajú ďalšie informácie, ako napríklad gramáž, alergény, či cena. Parser ich dokáže rozoznať a vytvoriť tak plnohodnotný objekt s atribútmi, ktorý je následne uložený do databázy. Vytvorenie univerzálneho parsera pre každý reťazec získaný z rôznych reštaurácii sa zdá byť nemožné. Preto bolo vytvorených viacero parserov, ktoré sa spúšťajú nad reťazcom podľa jeho pôvodu z webovej lokality. Každá reštaurácia dopĺňa názvy jedál vo svojom špecifickom formáte, a tak takéto prispôsobenie parsera prinieslo zlepšenie.

API na backende tvoria RESTové služby, na ktoré sa dokážu pripájať aplikácie, či už ide o Angular aplikáciu alebo sťahovač. Služby obsahujú metódy na manipuláciu s objektami.

Do API sú zahrnuté moduly pre odporúčanie a klasifikovanie jedál. Tieto moduly sú naprogramované v jazyku PHP a používajú pritom nástroj ElasticSearch, ktorý vďaka svojej rýchlosti a funkciám pomáha riešiť tieto problémy. Základnými štatistickými metódami vieme posúdiť podobnosť jedál a na základe podobnosti určiť či sa jedná skutočne o plnohodnotné jedlo alebo o jedlo obsahujúce vo svojom názve náznak toho či je alebo nie je mäsité. Rovnako nám tento nástroj pomáha mapovať názvy jedál na pripravené recepty s cieľom získať podrobné informácie o ingredienciách jedla a jeho príprave.

Robustnosť návrhu v module odporúčania nám umožňuje vytvárať rôzne kombinácie odporúčačov, pri ktorých sa vieme pohrať s váhami jednotlivých algoritmov odporúčania a tak dať väčší dôraz na konkrétnu metriku.

Pre možnosť klasifikovania jedál a taktiež jeho vyhodnocovanie je potrebné mať vzorku už zatriedených dát. Aby sme mohli rýchlo a viacerí naraz zatrieďovať jedlá, bola vytvorená aplikácia, ktorá nám to umožňuje. V aplikácii sa vždy zobrazí názov jedla a používateľ ho môže jednoduchým klikom zatriediť do dvoch skupín. Momentálne pracujeme so skupinami mäsité a bezmäsité jedlá. Vďaka týmto zatriedeným jedlám vieme presnejšie vyhodnocovať úspešnosť triedenia ako aj vylepšovať samotný algoritmus.

Najdôležitejším komponentom pre používateľa je frontendová aplikácia. Aplikácia aj vďaka vyššie spomenutým komponentom má niekoľko funkcionalít, vďaka ktorým si používateľ môže v pohodlí vyberať jedlá a listovať medzi nimi. Sú to:

- Prihlasovanie a registrácia osobitne alebo cez účty Facebook či Google
- Profil používateľa s jeho informáciami a možnosťou označenia alergénov
- Zoznam jedál v okolí používateľa, ktorý je vždy zoradený, buď podľa voľby alebo podľa štandardného odporúčača
- Zoraďovanie výsledkov na základe vzdialenosti, obľúbenosti, popularity a typu jedla (mäsité/bezmäsité)
- Detail jedla s jeho rozšírenými informáciami a reštauráciou
- Hodnotenie jedla vo forme *like* alebo *dislike*
- Reštaurácia zobrazená na mape v prípade, že by používateľ nevedel, kde sa reštaurácia nachádza
- Zvolenie polohy používateľom v prípade, že mu nefunguje GPS. Takýmto spôsobom si môže zaznačiť k akému bodu sa mu majú zobraziť jedlá

2.2.7 Referencie

Začiatkom semestra sme s naším projektom a aktívnou účasťou na podujatiach v novo vzniknutom komplexe Binarium dostali ponuku na mentoring od zakladateľov spoločnosti 0100 Campus, ktorým sa zapáčila myšlienka nášho projektu ale aj mladý schopný a ambiciózny tím. Poskytli nám zadarmo priestory, v ktorých sa môžeme spoločne stretávať a riešiť náš projekt. Na dvojtýždennej báze sme mali spoločné stretnutia, kde sme im prezentovali náš progres v projekte a smerovanie na ďalšie obdobie.

Veľkou pomocou pre nás, ako pre obyčajných programátorov z FIIT, bol ich pohľad na biznis stránku takéhoto projektu. Aj my sme sa potrebovali preorientovať viac na biznis stránku, čo bolo pre nás pomerne náročné. Všetci sme sa chytili tejto príležitosti a v domnení, že keď o nás nebudú ľudia vedieť, tak aplikácia môže byť super, ale nikto ju nebude používať, sme sa vydali do reštauračných zariadení a zisťovali, aké sú ich potreby a akým spôsobom by sme sa vedeli dohodnúť na získavaní obedových ponúk priamo od nich. Na druhej strane sme sa snažili náš prototyp dostať medzi ľudí, konkrétne do spoločnosti Anasoft, kde nám opäť aj kolegovia z 0100 Campusu výrazne pomáhali svojimi kontaktami a odporúčaniami.

Počas celého semestra sme mali možnosť zúčastňovať sa bezplatne aj na všetkých udalostiach, ktoré sa uskutočňovali v tejto komunite, čo sme neraz aj využili. Boli to hodnotné prednášky úspešných ľudí, ktorí ukazovali cestu, ako sa dostať na medzinárodnú biznis scénu s projektom alebo to boli prednášky aké sú najlepšie právne formy pre startupy, prípadne akcie ako spojiť prácu popri škole a pod.

2.2.8 Zoznam priložených e-dokumentov

Na priloženom médiu sa nachádzajú zdrojové kódy častí aplikácie:

- mfc-crawler-js (aktuálny sťahovač dát)
- mfc-crawler (prvotná verzia sťahovača dát)
- mfc-api (webové služby)
- mfc-app (frontend aplikácia)
- mfc-web (webová stránka tímového projektu)

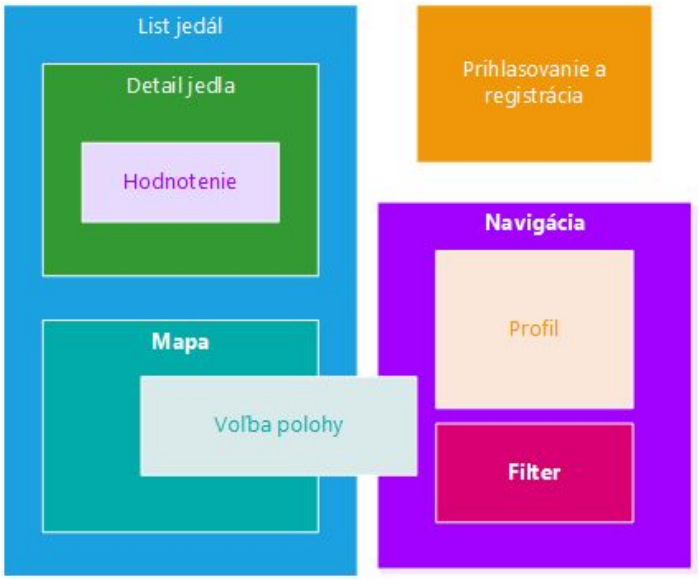
- Mfc-decision (nástroj pre označovanie jedál)

Na priloženom médiu sa tiež nachádza:

- Riadenie tímového projektu
- Technická dokumentácia webových služieb

2.3 Moduly systému

V tejto časti bližšie opisujeme moduly systému a ich vzájomnú komunikáciu. Presný popis všetkých prístupových bodov sa nachádza v dokumentácií https://dev.api.pokope.sk/docs, nasledujúcej časti budeme hľadieť skôr na logiku vykonávajúcu sa na pozadí.



Moduly aplikácie

1. Prihlasovanie a registrácia

Analýza

Na to, aby sme mohli zbierať údaje o činnostiach používateľoch a vytvoriť tak personifikované vyhľadávanie, je nutné mať používateľov uložených v databáze a vedieť určiť, aké akcie vykonali.

Návrh

Prihlasovanie pomocou účtov: MFC (email a heslo), Google a Facebook.

Registrácia do MFC je na samostatnej obrazovke. Nebude sa odosielať žiaden potvrdzovací email. Registráciu pomocou sociálnych sietí cez Google a Facebook účet tvorí samotné prihlasovanie, a preto nie je nutná žiadna ďalšia implementácia registrácie.

Polia[1]: meno*, email*, heslo*, potvrdenie hesla*

V prípade neúspešného prihlásenia pomocou Google a Facebook účtu bude používateľ presmerovaný na MFC s chybovým hlásením. V prípade prihlasovania pomocou MFC bude používateľ upozornený chybovým hlásením.

Po úspešnom prihlásení sa zobrazí zoznam jedál spolu s navigáciou.

Implementácia

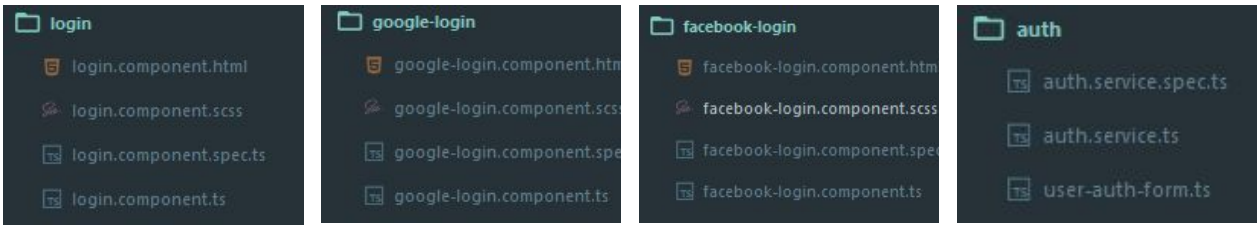
Backend

V prípade Facebook a Google prihlásenia boli použité štandardy OAuth a OAuth 2.0. V každom prípade registrácie, či už cez sociálne siete alebo klasickým spôsobom, sa vytvorí nová entita používateľa. Ak sa jedná o registráciu cez sociálnu sieť, vytvorí sa aj záznam o sociálnom profile, aby sme vedeli pri následnom prihlásení identifikovať, či sa jedná o používateľa registrovaného alebo používateľa využívajúceho sociálnu sieť.

Boli vytvorené endpointy na registráciu, prihlásenie, prihlásenie pomocou sociálnej siete a odhlásenie.

Frontend

Vytvorené nové komponenty *login, google-login, facebook-login* na základe generického návrhu *Angular* komponentov a autentifikačnej služby *auth*. Pre komponenty *google-login* a *facebook-login* sú využívane oficiálne knižnice.



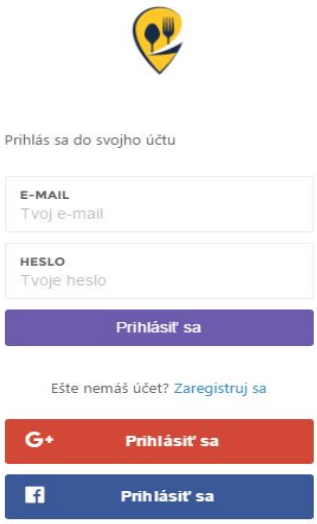
Jednotlivé komponenty prihlásenia

Príklad prihlásenia prostredníctvom Google účtu:

[1] Polia označené * budú povinné

Po inicializácii sa po kliknutí na tlačidlo odošle požiadavka prostredníctvom Google API. Po prijatí odpovede sa prostredníctvom *auth* služby uložia overenia používateľa do lokálneho úložiska používateľa.

Vytvorenie MFC účtu



Prihlásenie

Testovanie

V rámci testovania boli pre BE vytvorené integračné testy pre overenie REST služieb a aj pre samotné uloženie používateľských údajov a následného prihlásenia.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

2. Profil

Analýza

Používateľovi chceme umožniť výber jeho alergénov a odhlásenie z aplikácie. Z tohto dôvodu je v aplikácií vytvorený modul profilu.

Návrh

Prístup do profilu z hlavnej navigačnej lišty v aplikácii. Ikona profilu v ľavej časti navigácie.

Po kliknutí sa zobrazí nová obrazovka, ktorá bude obsahovať:

- generickú navigáciu s tlačidlom *späť* na zoznam jedál,

- meno a email používateľa,
- tlačidlo *odhlásiť*,
- zoznam používateľových alergénov a
- tlačidlo pre uloženie zmien v profile.

Používateľ bude mať možnosť vybrať si pomocou zaškrtávacích políčok svoje alergény. Všetky alergény budú zobrazené v zozname pod sebou a pri každom bude zaškrtávacie políčko.

Po kliknutí na tlačidlo uloženia sa profil uloží bez upozorňovacieho hlásenia.

Implementácia

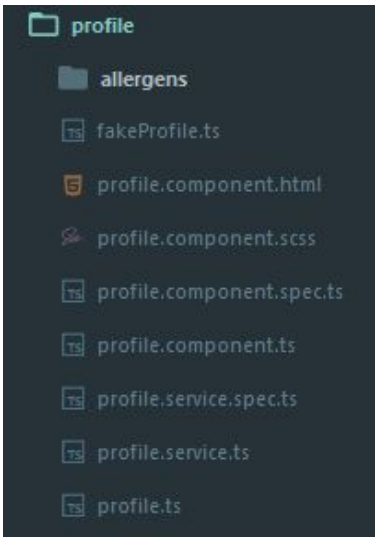
Backend

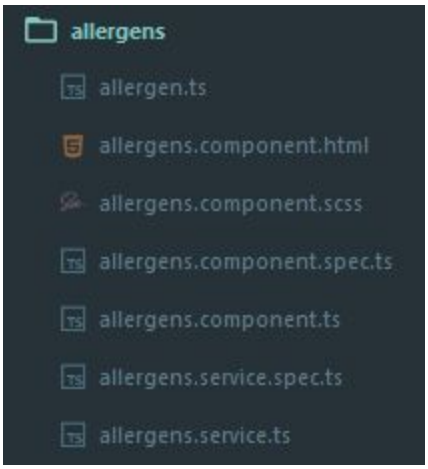
Na uloženie alergénov používateľa sme vytvorili endpoint, ten príjme dáta a následne do väzobnej tabuľky zapíše príslušné väzby. Riadenie týchto činností zabezpečuje príslušný *AllergensController* a jeho metódy.

Profilové informácie o prihlásenom používateľovi sa posielajú cez kontroler *MeController* po vykonaní GET žiadosti na príslušný endpoint.

Frontend

V aplikácii pribudol komponent *profile* a *allergens.*

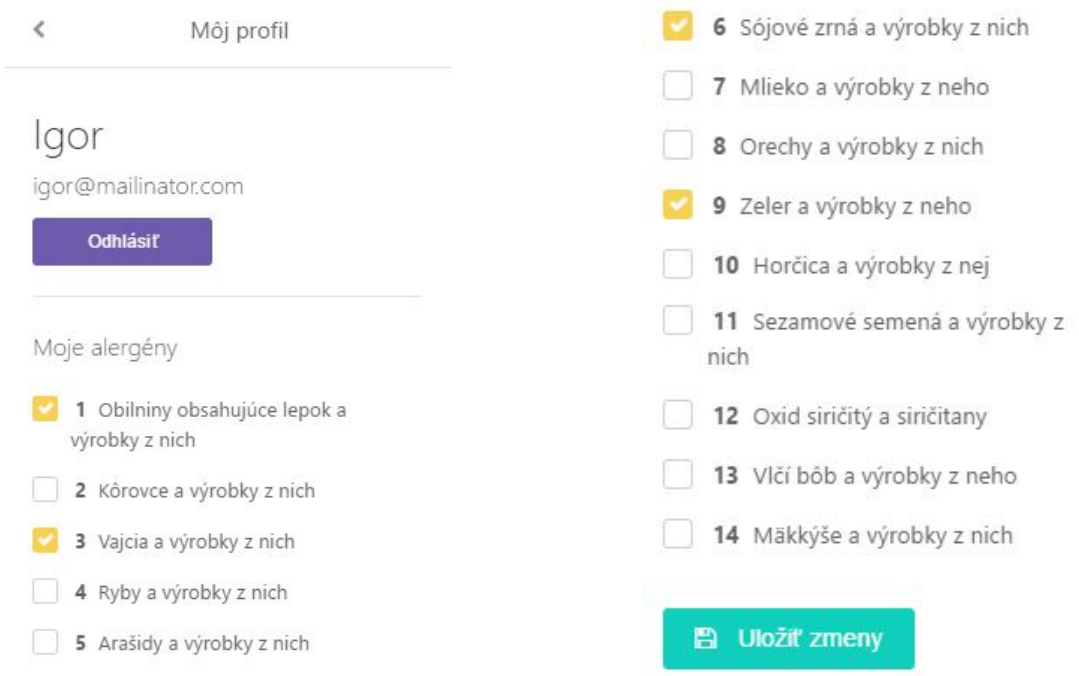




Komponent profile *Komponent allergens*

Po kliknutí na profil sa prostredníctvom REST služby stiahnu údaje o používateľovi a vykreslia sa na obrazovke.

Po kliknutí na tlačidlo späť (<) sa prostredníctvom REST služby stiahne zoznam jedál a zobrazí sa obrazovka pre zoznam jedál.



Profil používateľa spolu so zoznamom alergénov

Testovanie

Pre BE časť boli vytvorené integračné testy pre pokrytie možnosti uloženia a odoslania profilu používateľa prostredníctvom REST služieb.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

3. Zoraďovanie výsledkov

Analýza

Pre personalizované vyhľadávanie obedových menu je nutné vytvoriť možnosť výberu jednotlivých odporúčačov, ktoré ovplyvňujú poradie jedál v zozname.

Návrh

Tlačidlo pre filter v pravom hornom rohu navigácie. Po kliknutí sa rozbalí ponuka, v ktorej budú umiestnené nasledujúce možnosti s príslušnými ikonkami a názvom odporúčača:

- popularita (rozhodujúci faktor je počet klikov na dané jedlo),
- obľúbenosť (zobrazujú sa primárne jedlá, ktoré používateľ označil páčikom),
- vzdialenosť (zoznam je usporiadaný vzhľadom na vzdialenosť od najbližšej ponuky jedla) a
- obsah mäsa, teda rozdelenie ponuky na jedlo mäsité a bezmäsité.

Implementácia

Backend

Pri implementácii odporúčačov na backende sme kládli dôraz na vopred stanovené požiadavky, ktoré nám majú uľahčiť aj vývoj nových odporúčačov v budúcnosti. Naše požiadavky sú nasledovné:

- Jednoduchá implementácia nových odporúčačov
- Odporúčač zoraďuje položky na vstupe, kde najlepšie hodnotené položky sú tie, ktoré najviac vyhovujú preferenciám používateľa
- Odporúčače zohľadňujú kontext, v ktorom sa používateľ nachádza
- Odporúčače je možné kombinovať

Na základe týchto požiadaviek sme navrhli rozhranie pre implementáciu, a tiež aj základnú implementáciu zdieľanej funkcionality formou abstraktného odporúčača. Náš návrh je vyjadrený nasledujúcim diagramom.

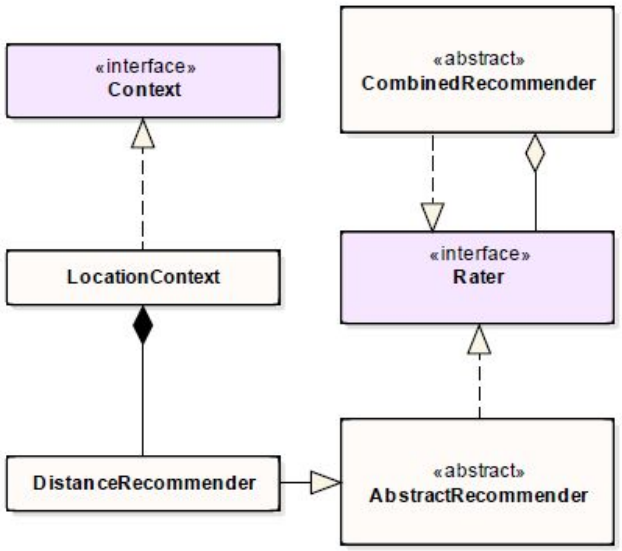


Diagram návrhu rozhrania pre implementáciu zoraďovania výsledkov

Na diagrame sú znázornené rozhrania a abstraktné triedy, ktoré umožňujú splniť všetky stanovené požiadavky. Zároveň je v diagrame aj ukážka použitia, keď sú znázornené dve konkrétne implementácie, ktorými sú *DistanceRecommender* a *LocationContext*. Ide o odporúčač, ktorý navrhuje resp. uprednostňuje jedlá nachádzajúce sa čo najbližšie k aktuálnej polohe používateľa. Poloha používateľa je kontextom, s ktorým odporúčač pracuje. Na diagrame je tiež znázornená konkrétna implementácia kontextu polohy používateľa.

Požiadavka na možnosť kombinácie odporúčačov je veľmi dôležitá. Ak sa používateľ rozhodne, že chce preferovať jedlá podľa vzdialenosti, stále sa stáva, že je v ponuke viacero jedál v rovnakej vzdialenosti. Tieto jedlá je tiež potrebné ohodnotiť a zoradiť podľa preferencie používateľa. Preto je potrebná možnosť odporúčače kombinovať. Každému odporúčaču je možné nastaviť váhu, ktorú má oproti ostatným odporúčačom v nejakej kombinácii.

```php
public static function createDistanceRater(array $data)
    {
        $rater = new CombinedRecommender();

        //popularita je pre pouzivatela vzdy zaujimava
        $rater->attachRecommender(new PopularityRecommender());

        /*
         * realne hodnotenia maju vsak vacsiu vahu,
         * hodnoteniam pouzivatelov dame dvojnasobu vahu
         * oproti popularite
         */
        $rater->attachRecommender(new RatingRecommender(), 2);

        //pouzivatela zaujimaju jedla v blizskosti
        $distance = new DistanceRecommender();
        //vaha vzdialenosti je oproti ostatnym najdolezitejsia
        $rater->attachRecommender($distance, 1000);

        //odporucac potrebuje poznat kontext polohy pouzivatela
        $distance->createContext(ContextFactory::LOCATION, [$data['lat'], $data['lon']]);

        return $rater;
    }
```

Kód vytvárajúci kombinovaný odporúčač s dôrazom na vzdialenosť

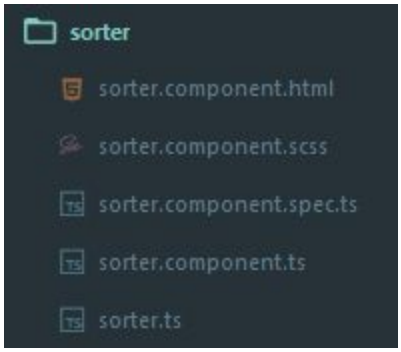
Použitie takto vytvoreného odporúčača je už jednoduché.

```php
$scores = $rater->rate($menuItems->pluck('id'));
$sortFnc = function ($item) use ($scores) {
    return $scores[$item->id];
};

return $menuItems->sortByDesc($sortFnc);
```

Použitie odporúčača v klientskej triede

Frontend

V rámci FE časti bol vytvorený nový komponent *sorter*.



Jednotlivé možnosti odporúčača sú reprezentované triedou obsahujúcou *type*, *icon* a *name*.

```
static popularityRecommender = {
    type: "popularity",
    icon: "fa fa-line-chart",
    name: "Populárne",
};
```

Vysvetlenie jednotlivých pojmov:

- type - určuje parameter, ktorý sa posiela do REST služby
- icon - ikona zobrazená na stránke
- name - názov zobrazený na stránke

Ponuka filtra

Po zvolení konkrétneho odporúčača je odoslaná požiadavka na REST službu BE časti a je vrátený nový zoznam jedál, ktorý je vykreslený v aplikácii.

Pre jednotlivé odporúčače bol navrhnutný model vyjadrený diagramom tried:

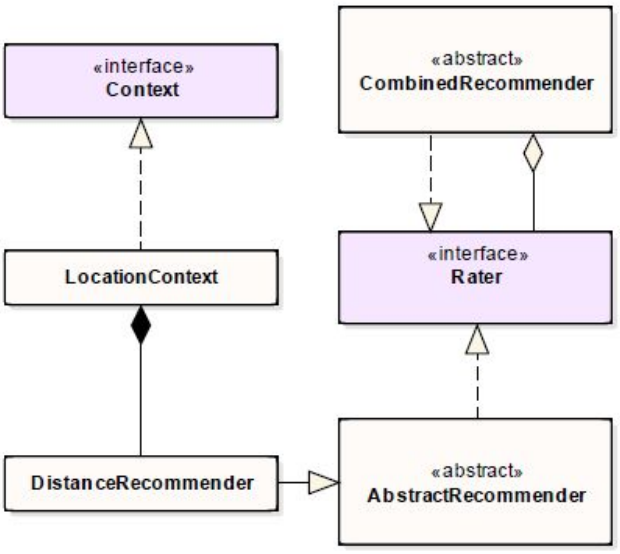


Diagram tried pre model odporúčača

Testovanie

V rámci BE časti boli vytvorené integračné testy pre testovanie REST služby pre vrátenie listu jedál na základe konkrétneho filtra.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

4. Zoznam jedál

Analýza

Zobrazenie jedál, resp. obedových menu je základnou funkciou aplikácie MFC. Používateľovi poskytuje možnosť rozhodnutia sa medzi viacerými možnosťami.

Návrh

Zoznam jedál bude zobrazený na základnej obrazovke MFC.

Jedna položka v zozname predstavuje jedno jedlo z obedového menu. Položka bude obsahovať:

- celý názov jedla,
- farebne vyznačenú vzdialenosť D na základe pravidiel:
    - Zelená farba D < 500
    - Modrá farba 500 =< D < 1000
    - Žltá farba 1000 <= D < 1500
    - Červená farba D > 1500
- výstražné znamenie v prípade konfliktu s alergénom používateľa.

Implementácia

Backend

V rámci backendu je vytvorená služba cez ktorú si frontend preberie dáta. Táto služba je navrhnutá tak, že vracia používateľovi zoznam jedál podľa jeho požiadaviek, t.j. zobrazenie ponúk z reštaurácií, ktoré sa nachádzajú v okolí polohy používateľa, resp. od polohy ním zvolenej. Prijaté súradnice sa odošlú na Elasticsearch a ten vráti zoznam identifikátorov reštaurácií v okruhu zadanej polohy. Následne už iba vyberieme ponuky týchto reštaurácií z relačnej databázy a zoznam jedál pokračuje ďalej, kde prechádza cez ďalšie filtre.

Aby sme zabezpečili, že používateľ nedostane v ponuke obedové menu reštaurácie, ktorá už v dopytovanom čase obedové menu neponúka, filtrujeme ponuky obedového menu podľa času podávania.

V tejto metóde takisto prebieha filtrovanie alebo zoraďovanie ponuky jedla. Podľa používateľom zvolenej metódy sa aplikuje na zoznam jedál príslušný algoritmus odporúčania a zoradí ponuku. Takto precízne pripravená ponuka jedál môže ísť naspäť k používateľovi cez *json response*.

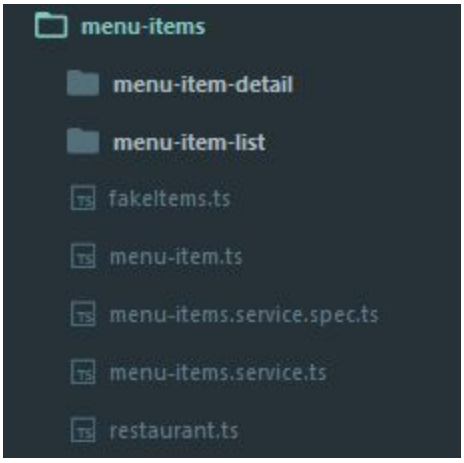
V nasledujúcej ukážke kódu môžeme vidieť dôležitú metódu pripravujúcu zoznam jedál pre aktuálneho používateľa.

```php
public function structured(FilterRequest $request)
{
    try {
        $restaurants = app(RestaurantRepository::class)->search($query);
        $this->menuItemRepo->pushCriteria(new MenuItemsInLocation($restaurants));
    } catch (\Exception $e) {
        $this->response->errorBadRequest($e->getMessage());
    }

    if (isset($params['date'])) {
        $this->menuItemRepo->pushCriteria(new MenuItemsInDate($params['date']));
    }
    // get only meaty or meatless foods in database query
    $raterParam = RaterFactory::RATER_REQUEST_PARAM_NAME;
    if (isset($params[$raterParam]) && in_array($params[$raterParam], ['meaty', 'meatless'])) {
        $isMeat = $params[$raterParam] == 'meaty' ? 1 : 0;
        $this->menuItemRepo->pushCriteria(new Meaty($isMeat));
    }
    // get only items that are served
    $this->menuItemRepo->pushCriteria(new MenuItemsServed(date('H:i:s')));
    $menuItems = $this->sortMenuItems($menuItems, $request);
    $menuItems = $this->addDistancesByContext($menuItems, $params);

    return $this->response->collection($menuItems, new StructuredMenuItemTransformer());
}
```

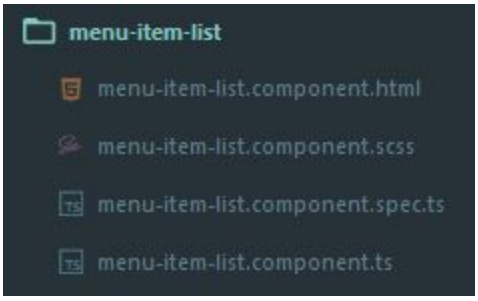
Metóda pripravujúca zoznam jedál pre používateľa

Frontend

V časti FE bol vytvorený komponent *menu-items* a *menu-item-list:*

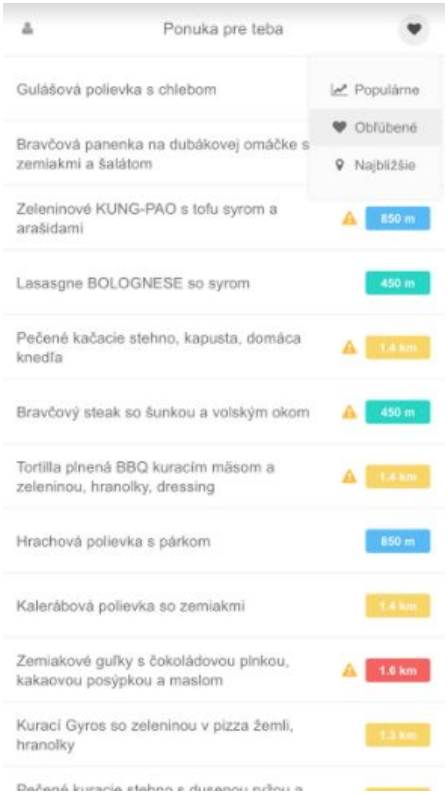


Komponent menu-items



Komponent menu-item-list

Po načítaní základnej obrazovky je odoslaná požiadavka na REST službu BE časti, ktorá vráti zoznam jedál na základe predvoleného filtra (Populárne). Po kliknutí na jedlo sa zobrazí obrazovka pre detail jedla.



Zoznam jedál aj s upozornením na obsah alergénov

Testovanie

V rámci BE časti boli vytvorené integračné testy pre testovanie REST služby pre vrátenie listu jedál.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

5. Detail jedla

Analýza

Ak používateľ chce zistiť bližšie informácie o jedle, nemôže tak urobiť na základnej obrazovke. V prípade, ak by boli všetky informácie o jedle na základnej obrazovke, bol by zoznam veľmi dlhý a nepreh ľadný. Detail jedla má slúžiť na poskytovanie bližších informácií jednak o jedle, ale aj o samotnej reštaurácii.

Návrh

Po kliknutí na jedlo v zozname na základnej obrazovke sa zobrazí obrazovka pre detail jedla.

Detail bude obsahovať:

- Navigáciu s názvom reštaurácie a tlačidlom späť na základnú obrazovku
- Názov jedla
- Cena jedla
    - max. 2 desatiné miesta spolu so znakom eura €
- Hodnotenie jedla
    - like/dislike tlačidlá s počtom hodnotení
- Alergény
    - V prípade konfliktu s alergénmi používateľa zvýrazniť červenou farbou
- Názov reštaurácie a adresy (malé písmo)
- Malý statický obrázok mapy s pinom na adrese reštaurácie

Po kliknutí na mapu sa zobrazí obrazovka pre mapu (bližší opis v module pre mapu).

Opis hodnotenia v samostatnom module.

Implementácia

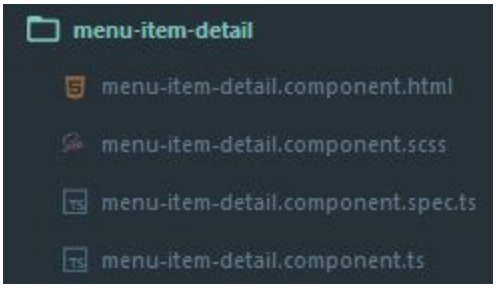
Backend

Na detail jedla slúži konktrétny endpoint, kde sa podľa identifikátora jedla získajú z databázy všetky potrebné údaje pre zobrazenie v aplikácií a naformátujú sa do požadovaného tvaru.

Keď používateľ klikne na niektoré z jedá v zozname jedál urobí sa request na prístupový bod a zapíše sa táto aktivita do príslušnej tabuľky. Akcia sa vykonáva na pozadí a používateľ nie je v nej priamo zainteresovaný.

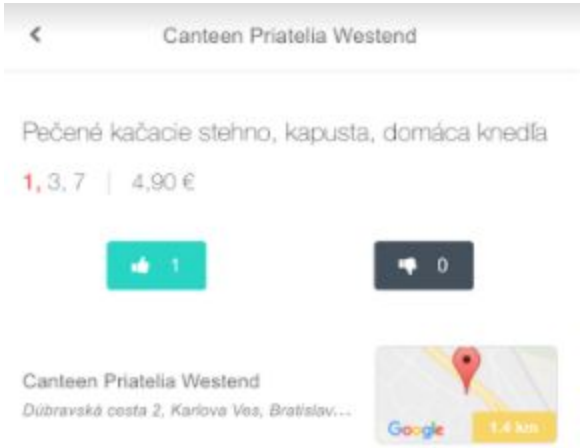
Frontend

Nový komponent *menuitem-detail:*



Po kliknutí na jedlo v základnej ponuke sa odošle požiadavka na REST službu BE časti, ktorá vráti detail zvoleného jedla a naplnia sa informácie na obrazovke pre detail jedla.

Mapa je inicializovaná prostredníctvom Google Maps API, kde je odoslaná požiadavka s údajmi o polohe reštaurácie a API vráti obrázok v požadovaných veľkostiach (130x70).



Detail jedla

Testovanie

V rámci BE časti boli vytvorené integračné testy pre testovanie REST služby pre vrátenie detailu jedla.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

6. Hodnotenie

Analýza

Ak chcem používateľovi odporúčať jedlá, ktoré mu chutia a má rád, musíme získavať spätnú väzbu aj vo forme ohodnotenia jedla.

Návrh

Používateľ bude mať možnosť ohodnotiť (hodnotenie môže byť buď *like* alebo *dislike* alebo žiadne, no nikdy nemôže byť *like* a *dislike* zároveň) jedlo nasledovne:

- Like - tlačidlo s páčikom hore
  - Zvýraznené v prípade zvolenia tejto možnosti.
- Dislike - tlačidlo s páčikom dole
  - Zvýraznené v prípade zvolenia tejto možnosti.
- Ani jedna možnosť - žiadne hodnotenie

- žiadne tlačidlo nebude zvýraznené

Implementácia

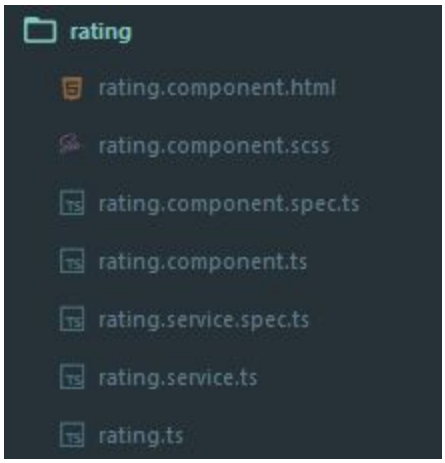
Backend

Záznam o hodnotení jedla používateľom sa ukladá ako do väzobnej tabuľky medzi používateľom a príslušným jedlom hodnotenie môže mať 3 stavy: používateľ jedlo hodnotil kladne = 1, používateľ hodnotil jedlo záporne = -1, používateľ zrušil svoje hodnotenie, v takomto prípade sa vymaže záznam o jeho hodnotení.

Pre optimalizáciu SQL dopytov na databázu si vždy udržiavame aktuálny stav Lajkov a Dislajkov na každom jedle, pri novom hodnotení od používateľa tento počet príslušne upravíme.

Frontend

Nový komponent v FE časti *rating:*



Kde hodnotenie je reprezentované ako trieda s atribútmi:

- like
- dislike
- user_rating - hodnoty -1 pre dislike, 0 pre neutrálne hodnotenie, 1 pre like
- type - typ hodnotenia (jedla)
- uuid - unikátny identifikátor

Testovanie

V rámci BE časti boli vytvorené integračné testy pre testovanie REST služby pre vrátenie detailu jedla.

FE časť bola testovaná manuálne na základe vytvorených scenárov.

7. Mapa

Analýza

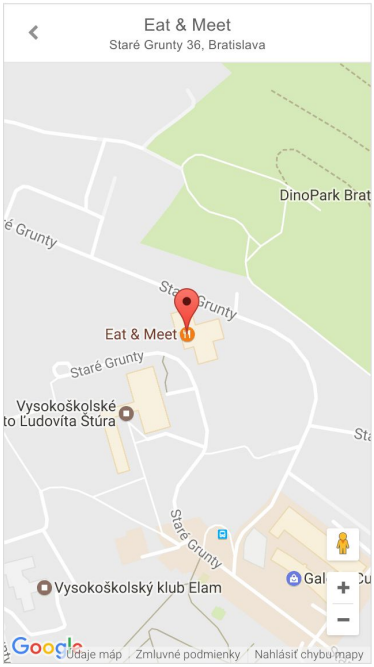
Nie vždy používateľ vie kde sa konkrétna reštaurácia nachádza, môže byť novo vzniknutá alebo sa používateľ nachádza v mieste kde sa nestravuje pravidelne. Z týchto dôvodov sme pridali komponent mapy.

Návrh

Používateľ bude mať náhľad mapy už pri detaile jedla, avšak ak by mu tento detail nestačil bude si vedieť po kliknutí na tento detail s mapou. Na mapy použijeme voľne dostupné google mapy. Používatelia ich dôverne poznajú a vedia s nimi interagovať.

Obrazovka mapy bude obsahovať:

- Názov reštaurácie
- Adresu reštaurácie
- Samotnú mapu
    - Pin na mape v mieste adresy reštaurácie



Implementácia

Backend

Tento komponent nevyužíva priamo backendové služby. Informácie o reštaurácií si prenáša z predošleho requestu.

Frontend

Komponent zahŕňa priame zobrazenie mapy pomocnou Google maps API, cez ich poskytované SDK v podobe knižnice pre našu verziu Angularu.

Informácie zobrazujúce sa na tejto obrazovke sa prenášajú z detailu jedla, kde sa priamo z prístupového bodu získajú informácie o polohe reštaurácie vo formáte *latitude*, *longitude*, a tak isto aj adresa zariadenia, ktorá sa zobrazuje v navigačnej časti obrazovky.

Informácie o polohe reštaurácie sa zadajú do pripraveného google komponentu a na mape sa v mieste vykreslí pin.

Testovanie

FE časť bola testovaná manuálne na základe vytvorených scenárov.

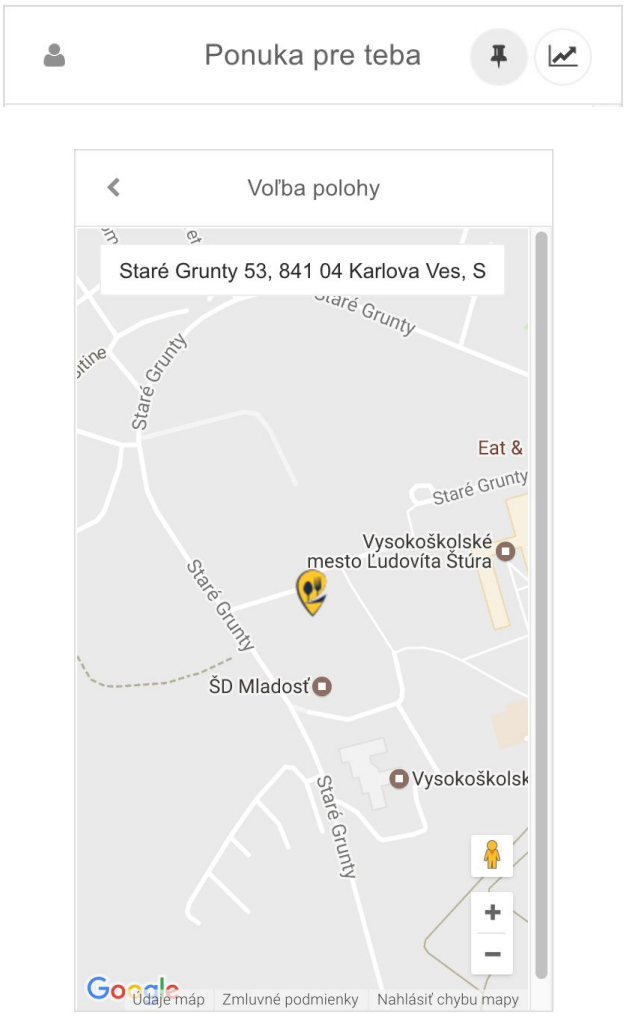
8. Zvolenie polohy

Analýza

Hlavné dôvody pridania tejto funkcionality bolo nie vždy 100% zistenie polohy používateľa na základe GPS prípadne HTML komponentu. A tiež aj z dôvodu modelovej situácie, kedy si používateľ chce pozrieť ponuku jedál v mieste, v ktorom sa momentálne nenachádza, ale vie, že sa tam bude stravovať. Je možné že používateľ bude poznať iba adresu preto chceme pridať aj automatické dopĺňanie adresy pomocou nástrojov google.

Návrh

- Do komponentu navigácia pridáme tlačidlo pinu.
- Po kliknutí na tlačidlo sa otvorí mapa.
- Na mape sa nachádza jeden pin, ktorý určuje aktuálnu polohu používateľa.
- Posúvaním mapy používateľ nastaví svoju alebo požadovanú polohu.
- Kliknutím do vstupného poľa môže používateľ zadať adresu, pin a poloha sa mu nastavia presne na zadanú adresu.



Implementácia

Backend

Komponent mapy nevyužíva priamo služby backendu.

Frontend

V rámci komponentu zvolenia polohy sa používajú služby Google, či už na zobrazenie mapy s pinom, ktoré fungujú rovnako ako v prípade komponentu Mapa. V tejto obrazovke je tiež možné vpísať adresu do vstupného poľa, ktoré je spracovávané Google službou a vyhľadáva adresy ulíc na slovensku a ponúkne nám nájdené zhody. Podľa tejto adresy sa automaticky nastaví pin na mape.

Rovnako tento princíp funguje aj z opačnej strany, kde pohybujeme mapou, pričom pin je staticky v strede, a v input poli sa automaticky menia adresy podľa polohy pinu na mape.

Takto definovaná poloha nahradí automaticky získavanú polohu aplikáciou a v ďalších requestoch bude použitá už nová poloha.

Testovanie

FE časť bola testovaná manuálne na základe vytvorených scenárov.

9. Navigácia

Analýza

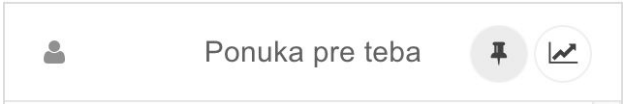
Aby sa používateľ vedel orientovať v aplikácií a dokázal sa posúvať medzi jednotlivými obrazovkami, vytvoríme jednoduché menu, ktoré sa bude vždy podľa potreby aktuálnej obrazovky prispôsobovať.

Návrh

Menu sa bude nachádzať na všetkých obrazovkách, v niektorých obrazovkách bude slúžiť iba na prechod do prechádzajúcej obrazovky.

Pri základnej obrazovke ponuky jedla je navigačné menu bohaté.

1. V ľavo sa nachádzať ikona používateľa ktorá slúži na prechod do profilu prihláseného používateľa.
2. V strednej časti sa nachádza názov aktuálnej obrazovky.
3. Na pravej strane je ikona pinu, ktorá sluzi na manuálne nastavenie polohy používateľa.
4. Ako posledná je ikona grafu, pomocou ktorej si používateľ bude vedieť prepínať ponuku jedál podľa jeho preferencie.



Implementácia

Backend

Komponent mapy nevyužíva priamo složby backendu.

Frontend

Implementačná časť frontendu je rozdelená na dve skupiny komponentov. Prvá rieši priamo funkcionalitu a volania na backend, tá druhá zasa preklikávanie medzi obrazovkami a tok aplikácie. Navigačný komponent sa vkladá do komponentov druhej skupiny. V nej sa vždy nastavia parametre navigácie pre danú obrazovku. Pomocou týchto parametrov vieme definovať aké akcie sa majú zobraziť a aké nie pre danú obrazovku.

Komponent pre zoraďovanie jedál v zozname je z pohľadu používateľa obyčajné výberové tlačidlo (angl. select box). Tento komponent je možné vložiť medzi HTML značky navigačného komponentu a vďaka funkcionalite, ktorú nám Angular 2 prináša, je možné vo vnútri navigácie presne umiestniť toto výberové tlačidlo a používať ho. Rozhranie, ktoré nám to umožňuje, sa volá *ng-content*.

Testovanie

FE časť bola testovaná manuálne na základe vytvorených scenárov.

2.4 Zmrazená verzia aplikácie

Zmrazená verzia aplikácie obsahovala nasledujúce moduly:

- Prihlasovanie a registrácia
- Profil s možnosťou označenia alergénov
- Zoznam jedál
- Zoraďovanie výsledkov podľa vzdialenosti, popularity, obľúbenosti a typu jedla (mäsité/bezmäsité)
- Detail jedla
- Hodnotenie jedla
- Mapa
- Navigácia

Pre detailnejší opis modulov viď kapitolu Moduly systému.

Príloha A - Motivačný dokument

Tím

Tím pre projekt *My Food Court* je zložený z informatikov, ktorých práca je ich hobby a radi pracujú na inovatívnych nápadoch. Hlavnou výhodou tímu sú bohaté skúsenosti v oblastiach, ktoré pokrývajú veľkú časť projektu. Každý z nich už potvrdil svoje poznatky v praxi, a preto pri návrhu projektu vie dopredu odhadnúť náročnosť jednotlivých častí a byť tak prínosom pri návrhu a implementácii.

Jozef Balún

Po niekoľkých rokoch práce ako freelancer sa v roku 2014 pridal ku skupine Pixwell, s ktorou momentálne pracuje. Skupina sa prerástla v softvérovú firmu. Venujú sa vývojom webových a mobilných aplikácií založených na REST komunikácii.

Tiež bol zamestaný v spoločnosti Goldmann systems, a.s., kde pracoval v tíme na vývoji backoffice aplikácie pre bankové subjekty.

Počas pracovnej praxe sa stretával s technológiami ako PHP framework Laravel, frontend framework AngularJS alebo buildovací nástroj Jenkins. Zaujíma ho oblasť spracovania a analýzy dát, kde cíti potenciál pre svoj ďalší rast.

Prečo je pre tím dôležitý?
Má veľký záujem o analýzu a spracovanie dát. Svojou odhodlanosť postaviť sa každému problému a prepracovať detaily pomôže vytvoriť skutočne kvatitný produkt s potenciálom uspieť.

Martin Prekala

V máji 2015 začal pracovať vo firme Centaur a.s., ktorá je členskou firmou Softec Group. Ihneď sa stal súčasťou jedného z najväčších projektoch celého združenia, na ktorom momentálne pracuje vyše 40 ľudí. Projekt je vyvíjaný metodikou Scrum, Martin denne rieši vzniknuté problémy s testermi, analytikmi a architektami.

Na backende pracuje s Javou v kombinácií s rule enginom Drools a frameworkami ako Spring či Hibernate, venuje sa aj vývoju používateľského rozhrania s využitím technológie AngularJS a

RESTových služieb. Denne sa stretáva s buildovacími a platformovými technológiami Gradle, Jenkins, Grunt.

Prečo je pre tím dôležitý?
Má výborne skúsenosti v oblasti vývoja frontendu v AngularJS. Baví ho navrhovať systémy, a potom to preniesť do praxe. Rád rieši chúlostivé a netradičné situácie, ako v živote, tak aj v kóde. Jeho kľúčové slová: abstrakcia, znovupoužitelnosť a prehľadnosť.

Miroslav Rác

Množstvo skúseností nadobudol počas 6 rokov trvajúceho obdobia freelancingu. V súčasnosti pôsobí v spoločnosti Adbee digital s.r.o., kde okrem vývoja webových aplikácií mentoruje junior programátorov.

V budúcnosti chce byť lídrom vývojarského tímu, s ktorým bude realizovať rôzne nápady s biznis potenciálom.

Prečo je pre tím dôležitý?
Svoje projekty programuje s veľkou dávkou kreativity, má zmysel pre detail a teší sa “peknému” a znovupoužiteľnému kódu.

Je pripravený zaviesť postupy, ktoré pomôžu tímu vybudovať si príjemnú atmosféru založenú na kvalitnej komunikácii. Zaujíma sa o psychológiu, fascinuje ho vesmír a evolúcia.

Matej Schwartz

Matej sa niekoľko rokov venuje webovým technológiám a vývoju webových aplikácií. Po rokoch práce ako freelancer pracuje v Adbee digital s.r.o. ako web developer. Už dlhšie pracuje na projekte Call centra v Cloude, ktorého počet používateľov sa stále rozrastá.

Prečo je pre tím dôležitý?
V extrakcii dát z webov má veľké skúsenosti nadobudnuté v QualityUnit s.r.o. ako data miner. Svojou dôkladnosťou prispeje pri navrhovaní a implementácii aplikácie.

Štefan Schwartz

Popri štúdiu sa s tímom podieľal na vývoji webových stránok v neziskovej organizácii a momentálne tento tím vedie. Súčasne už vyše roka pracuje v internetovej banke ZUNO BANK AG ako dátový analytik na oddelení komunikácie so zákazníkmi. Štefan je súčasťou tímu na analýzu získavaných dát, ktorá je vzhľadom na charakter banky veľmi dôležitá. Priblížil sa aj k témam UX a webovej analytiky.

Prečo je pre tím dôležitý?
Je najstarší z tímu a svojimi skúsenosťami s vedením dokáže udržať tím na ceste k úspešnému koncu. Už viac ako rok sa venuje analýze dát a práci v databázach čím pokrýva časť projektu. Tiež je prínosom v oblasti UX.

Igor Šimko

Počas svojho pôsobenia sa venoval viacerým projektom, počas ktorých nadobudol rozličné znalosti najmä vo vývoji viacvrstvových Java aplikácií a Android aplikácií.

Momentálne pôsobí ako vývojár v spoločnosti Unicorn systems. Počas jeho doterajších pracovných skúseností pracoval v tímoch využívajúcich agílnu metodiku vývoja softvéru - scrum. K vývoju pristupuje systematicky a úlohy rieši s analytikmi alebo testermi.

Má skúsenosti s viacero rámcami ako Spring či Hibernate, navrhovaním databázových modelov, či spravovaním aplikačného servera.

Prečo je pre tím dôležitý?
Igorove skúsenosti s metodikou scrum vedia tímu pomôcť pri vedení ako aj vývoji projektu a jeho manažovaní. Nie je mu cudzia téma neurónových sietí, ktoré sú pre tento projekt jednou z nevyhnutných častí. Je spoľahlivý a zvyknutý pracovať pod tlakom a v časovej tiesni.

Opis témy My Food Court

Milan pracuje v centre mesta, a tak mu je pohodlné sa denne stravovať v reštauráciách. Okrem toho, že sa môže počas obedu porozprávať s kolegami, každý deň má na výber jedlá z rozsiahlej ponuky. V blízkom okolí má k dispozícii tri reštauračné zariadenia, ktoré ponúkajú jedlá formou denného menu.

Vždy pred poludním navštívi rôzne webové lokality, na ktorých má dostupné jedálne lístky jednotlivých zariadení. Na základe ponuky sa rozhodne, ktorú reštauráciu navštívi. Po dlhých mesiacoch si všimol, že sa v ponuke striedajú tie isté jedlá. Najradšej má paradajkovú polievku, a tiež veľmi obľubuje ázijskú kuchyňu. Ak má niektorá reštaurácia v ponuke túto polievku alebo Kung Pao, namieri si to práve tam.

Milan to nemá jednoduché. Uvedomuje si, že strava vo veľkej miere ovplyvňuje jeho zdravie. Navyše trpí intoleranciou na laktózu. Preto dáva veľký dôraz na výber jedál, ktoré konzumuje. Je hravý typ, a tak rád vyskúša akúkoľvek asistenciu pri tejto každodennej rutine. Ak sa mu riešenie jeho problému zapáči, veľmí rád sa o skúsenosť podelí so svojimi kamarátmi.

Aký je náš zámer?

Výsledkom nášho úsilia vyvinutého v nasledujúcich dvoch semestroch bude aplikácia, ktorá združuje informácie o každodenných ponukách na jednom mieste. Používateľovi zobrazuje personalizovanú ponuku, ktorá **berie ohľad na osobné preferencie** pre suroviny a spôsob prípravy jedál, jeho osobné výživové potreby, a tiež prípadné obmedzenia vo forme alergénov.

Personalizované výsledky aplikácia navrhne na základe predošlého výberu a podľa informácií, ktoré získa formou jednoduchých otázok. Aplikácia disponuje používateľsky príjemným prostredím a nenásilným správaním.

Dáta o ponukách reštauračných zariadení sú získavané z dostupných webových lokalít. Taktiež je reštauračným zariadeniam ponúknuté rozhranie na vkladanie ponuky jedál na väčšie časové obdobie dopredu. Alternatívne majú možnosť poskytnúť API rozhranie, cez ktoré sú informácie získavané pravidelne a automaticky našou aplikáciou.

Čo naša aplikácia je, a čo nie je?

Aplikácia je a poskytuje:

- agregátor informácií o dennej ponuke jedál,
- personalizovaná ponuka jedál v blízkom okolí,
- aplikácia dokáže zhodnotiť stravovací návyk používateľa.

Aplikácia nie je:

- výživový poradca,
- sociálna sieť,
- katalóg reštauračných zariadení.

Ako to uskutočníme?

Najdôležitejšiu časť našej aplikácie predstavuje získavanie a analýza veľkého objemu dát. Pri návrhu zúročime vedomosti získané na predmetoch *Vyhľadávanie informácií* a *Objavovanie znalostí*.

Na spracovanie dát plánujeme využiť existujúce nástroje na spracovanie slovenského textu, konkrétne **NLP nástroje na lematizáciu a stemovanie**, ktoré vznikli pod vedením Ing. Mariána Šimka, PhD.

Vzhľadom na potrebu rýchleho vyhľadávania údajov v databáze, s našou tému súvisí predmet *Pokročilé databázové technológie.* Predpokladáme, že využijeme nástroj **ElasticSearch**. Potrebujeme vytvoriť asociácie medzi dátami na základe rôznych faktorov, preto nás veľmi podporí predmet *Neurónove siete*.

Aby sme vytvorili príjemné používateľské prostredie, sme pripravení využiť možnosť testovania aplikácie vo výskumnom centre UXI na FIIT.

Kto je naša cieľová skupina?

Používateľmi aplikácie sú ľudia stravujúci sa v jedálňach alebo v reštauráciách formou denného menu. Táto **skupina je pomerne rozsiahla**, pretože obed je neodmysliteľnou súčasťou dňa. Takúto možnosť stravovania využíva množstvo pracujúcich ľudí, predovšetkým vo veľkých mestách.

Druhou cieľovou skupinou sú samotné **stravovacie zariadenia**, pre ktoré aplikácia znamená lacnú a efektívnu formu marketingu. Vďaka našej aplikácii dokážu osloviť potenciálnych zákazníkov, ktorí majú skutočný záujem ich zariadenie navštíviť.

Aký je biznis model?

Aplikácia nie je spoplatnená pre samotných používateľov. Pre reštaurácie však predstavuje marketingový nástroj, a tak platia za impresie formou pay per click.

Existujú na trhu podobné riešenia?

Na súčasnom trhu je dostupných niekoľko aplikácií v rámci domény reštaurácie a jedlo. Sú zamerané na vyhľadávanie reštaurácií a ponúkajú ich na základe lokality, podľa referencií a hodnotenia zákazníkov.

Niektoré z nich sú postavené na báze sociálnych sietí. Medzi najznámejšie patria *Zomato, Yelp, Grubhub, Urbanspoon* a *Eat24*. Časť z nich sleduje nutričné hodnoty jedál, o ktorých informuje používateľa. Tieto aplikácie však nevytvárajú profil stravovacieho návyku a nepersonalizujú výsledky.

V čom je naše riešenie inovatívne?

Existujúce služby jednoducho zobrazujú dáta o ponuke jedál. Naše riešenie získané dáta spracuje, pochopí a navrhne personalizovanú ponuku na základe profilu o stravovacom návyku používateľa.

Prečo si používatelia vytvoria vzťah k aplikácií?

Používateľovi sú zobrazované iba ponuky, o ktoré prejavuje reálny záujem. Nemusí tak každodenne filtrovať desiatky položiek v ponukách rôznych zariadení.

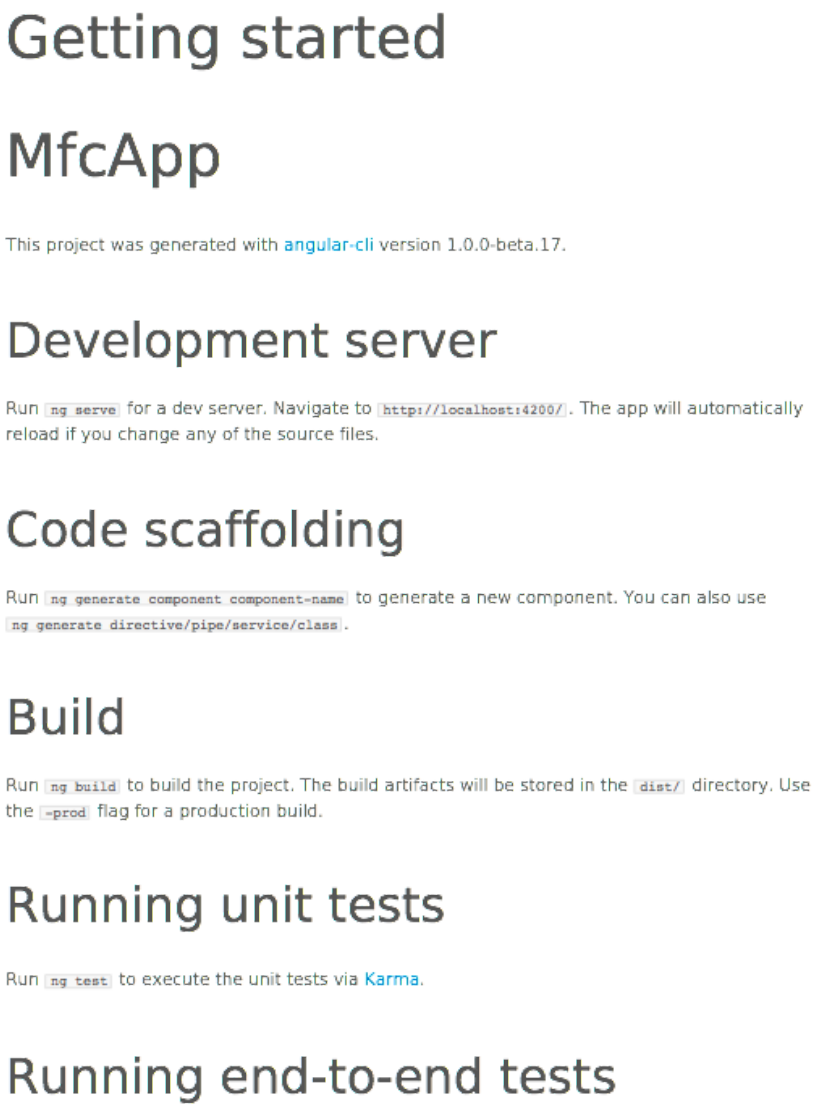
Aplikácia si vytvára k jej používateľovi vzťah tak, že mu zobrazuje rôzne ciele (tzv. *achievementy*), ktoré dosiahol alebo ešte len môže dosiahnuť. Neustále mu je zobrazovaný vývoj jeho postupu k dosiahnutiu daného cieľa - napríklad eliminovať sacharidovú zložku stravy.

Aký je náš dlhodobý cieľ?

Informácie sú najcennejšou komoditou. Našim dlhodobým zámerom je vybudovať databázu informácií o trende stravovacích návykov ľudí v konkrétnych regiónoch.

Príloha B - Príručky ZS

Getting started - Angular app



Getting started

MfcApp

This project was generated with angular-cli version 1.0.0-beta.17.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive/pipe/service/class`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory. Use the `-prod` flag for a production build.

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via Protractor. Before running the tests make sure you are serving the app via `ng serve`.

Deploying to Github Pages

Run `ng github-pages:deploy` to deploy to Github Pages.

Further help

To get more help on the `angular-cli` use `ng --help` or go check out the Angular-CLI README.

Created Sat, Nov 5, 2016 2:29 PM by Martin Prekala
Last Updated Sat, Nov 5, 2016 2:30 PM by Martin Prekala

ElasticSearch

Elasticsearch 2.4.1

K inštalácii

Dáta uložené: /var/lib/elasticsearch

Miesto inštalácie: /usr/share/elasticsearch/

Konfiguračný súbor: /etc/elasticsearch/elasticsearch.yml

Inicializačný skript: /etc/init.d/elasticsearch

Logy: /var/log/elasticsearch/

Názo clustra: my-food-court

Elasticsearch je nainštalovaný ako service, takže vždy po reboote sa naštartuje sám. Práca s ním sa vykonáva pomocou príkazu:

sudo service elasticsearch status/start/stop/restart

Bližšie info k inštalácii https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-on-ubuntu-16-04

Nieje nainštalovaná najnovšia verzia kvôli dodatočnému pluginu, ktorý potrebujeme. K čistej inštalácii je doinštalovaný plugin pre lematizovanie https://github.com/vhyza/elasticsearch-analysis-lemmagen. Zložka /usr/share/elasticsearch/plugins/elasticsearch-analysis-lemmagen.

Taktiež sa inštaluje https://github.com/SlovakNationalGallery/elasticsearch-slovencina.

Používanie

API elasticsearch je dostupné len lokálne na adrese http://127.0.0.1:9200/. Z domény nie je dostupný kvôli bezpečnosti. Bude ho využívať iba MF-API. V prípade, že chcete otestovať jeho funkčnosť alebo si niečo vyskúšať v termináli spustite:

curl -X GET http://127.0.0.1:9200/

Na requesty z vonka použite port 8018

curl -X GET http://team05-16.studenti.fiit.stuba.sk:8018

Čitateľný výstup v konzole

Ak pristupuješ k elasticu cez konzolu, existuje možnosť zobraziť návratový JSON v peknom formáte pomocou programu **jq**. Stačí doň ouput poslať cez pipe.

```
curl -XGET http://localhost:9200/_settings | jq
```

?
GET mfc/restaurants

Všetky reštaurácie vo vzdialenosti X

metóda GET (parametre nižšie) na adresu http://localhost:9200/mfc/restaurants vráti reštaurácie vo vzdialenosti 200km od **pin.location**, teda od konkr. bodu

GET method body

```
{     "filter"
 : {             "geo_distance"
 : {                 "distance" : "200km"
,                 "pin.location"
 : {                       "lat"
 : 40,                       "lon"
 : -70
                }
            }
        }
    }
```

Všetky reštaurácie vo vzdialenosti X + vzdialenosť v km

```
curl -XGET http://localhost:9200/mfc/restaurants/_search -d '
{
 "fields" : ["title"],
 "script_fields" : {
 "distance" : {
    "params" : {
       "lat" : 48.1589701,
       "lon" : 17.0636278
    },
    "script" : "doc[\u0027pin.location\u0027].distanceInKm(lat,lon)"
 }
 }
}'
```

Výstup


```
"hits": [
  {
 "_index": "mfc",
 "_type": "restaurants",
 "_id": "295",
 "_score": 1.0,
 "fields": {
 "title": [
    "Pizzeria Rotunda Restaurant"
 ],
    "distance": [
       9.721408485030103
    ]
 }
 }]
```


?
POST mfc/restaurants

metóda POST (parametre nižšie) na adresu http://localhost:9200/mfc/restaurants pridá záznam do elastiku.

POST method body


```
{    "id"
: 123456,    "title": "Testing restaurant Presov"
,    "pin"
 : {         "location"
 : {             "lat"
 : 41.12,             "lon"
 : -71.34
        }
    }}
```


---

mfc/food_decisions

Index slúži na **rozhodovanie, či zadaný string je jedlo alebo nie je**.

Mapping

Uchovávame informácie:

- input - surový text, teda raw dáta z tretej strany
- output - nami upravená forma, prestavuje "pekný výstup"
- is_food - boolean hodnota o tom, ci záznam je jedlo, alebo nie je

```
PUT /mfc/_mapping/food_decisions
{
    "properties":{
        "input":{
            "type":"string",
            "fields":{
                "raw":{
                    "type":"string",
                    "index":"not_analyzed"
                }
            },
            "analyzer":"slovencina",
            "search_analyzer":"slovencina_synonym"
        },
        "is_food":{
            "type":"boolean"
        },
        "output":{
            "type":"string",
            "fields":{
                "raw":{
                    "type":"string",
                    "index":"not_analyzed"
                }
            },
            "analyzer":"slovencina",
            "search_analyzer":"slovencina_synonym"
        }
    }
}
```

Vytváranie dokumentu

Do path sa pridáva číslo (v ukážke 1), ktoré predstavuje ID jedla v databáze.

Index obsahuje iba dokumenty, o ktorých **je rozhodnuté, či ide o jedlo alebo nie**.

?
PUT /mfc/food_decisions/1

```
{
    "input": "Polievka: Cícerová krémová s krutónmi 250ml  /  Vývar s rezancami  250ml",
    "output": "Polievka cícerová krémová s krutónmi / Vývar s rezancami",
    "is_food": true
}
```

Úprava dokumentu

Do path sa pridáva číslo (v ukážke 1), ktoré predstavuje ID jedla v databáze.

?

POST /mfc/food_decisions/1/_update

```
{
    "doc":{
        "is_food": false
    }
}
```

Zisťovanie, či string predstavuje jedlo

coming soon...

Created Thu, Oct 27, 2016 7:11 AM by Matej Schwartz
Last Updated Mon, Nov 21, 2016 5:27 PM by Miroslav Rác

Príloha C - Inštalačná príručka LS

Aplikácia

Pre vývoj a nasadenie aplikácie je nutné mať nainštalované nástroje:

- Node.js + npm
- Angular-cli

Postup prípravy pre vývoj:

- Stiahnutie repozitára do lokálneho priečinka
- Inštalácia potrebných modulov prostredníctvom príkazu “npm install”
- Spustenie aplikácie v developerskom režime “ng serve”
- Po každej úprave v aplikácii bude zmena zobrazená v aplikácii automaticky

Postup prípravy pre nasadenie:

- Stiahnutie repozitára do priečinka na serveri
- Inštalácia potrebných modulov prostredníctvom príkazu `` `npm install` ``
- Build aplikácie príkazom `` `ng build` ``
  - Tento príkaz môže obsahovať dodatočné parametre
  - Build pre testovanie `` `ng build --dev --base-href /` `` (kompilácia zdrojových súborov a nastavenie hlavnej zložky aplikácie, využívanie testovacích nastavení)
  - Build pre produkciu `` `ng build --prod --aot --base-href /` `` (kompilácia zdrojových súborov, minifikácia, optimalizácia, nastavenie hlavnej zložky)

API

Požadované nástroje:

- Composer
- PHP 7.0
- Elasticserach 5.3
- Aglio
- Gulp

Inštalácia

1. Stiahnutie repozitára

2. `composer install`
3. Nastavenie databázy a ďalších environment nastavení v .env súbore
4. `php artisan migrate` - vytvorenie tabuliek
5. `php artisan serve` - aplikácia beži na http://localhost:8000
6. `./vendor/bin/phpunit` - spustenie testov
7. `php artisan docs:generate` - vygenerovanie dokumentácie

Sťahovač

Pre vývoj a nasadenie crawlera je nutné mať nainštalované nástroje:

- Node.js + npm

Postup vývoja:

- Stiahnutie repozitára do lokálneho prostredia
- Inštalácia závislostí `npm install`
- Skopírovanie konfiguračného súboru z priečinka `/sample/config.sample.js` do `/config.js`
- Nastavenie pripojení a logovania v konfiguračnom súbore
- Spustenie skriptu pomocou `node index.js`