ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace pro sestavovaní jídelníčku
Student:	Bogdan Bodnar
Vedoucí:	Ing. Miroslav Hrončok
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte webovou aplikaci, která umožní sestavovat jídelníček podle potřebných živin. Aplikace musí mít API i webové UI; obojí poskytující následující funkce:

- Přihlášení přes lokální účet a služby 3. stran
- Určení denní dávky živin, nj. základních (kalorie, cukry, tuky), ale i známých vitamínů a minerálů
- Zaznamenávání konzumovaných potravin
- Zobrazení statistiky konzumování živin za zvolené období
- Sestavení příděl z jídel podle denní dávky
- Smysluplně a rozmanitě sestavený příděl (aplikace např. nesmí stále doporučovat stejné potraviny)

Poskytované API musí splňovat principy REST a HATEOAS. Aplikace musí využívat API poskytované Ministerstvem zemědělství USA. Kód i API musí být psán i dokumentován v anglickém jazyce. Kód musí být vhodně členěn, splňovat konvence zvoleného jazyka a frameworku a obsahovat extensivní sadu testů.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 17. února 2017

České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra počítačových systémů

Bakalářská práce

Webová aplikace pro sestavovaní jídelníčku

Bogdan Bodnar

Vedoucí práce: Ing. Miroslav Hrončok

10. května 2017

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Miroslavu Hrončokovi za odborné vedení a cenné připomínky. Také bych rád poděkoval Ing. Petru Viktorinovi a Ing. Miroslavu Hrončokovi za znalosti a praktické zkušenosti získané při absolování předmětu MI-PYT, které jsem uplatnil během implementace této práce.

Děkuju svým kamarádům a spolužákům Alehu Kuchynskému, Alexandru Shatrovskemu a Uladzislavu Maherovi za rady a pomoc při testování této aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl 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 10. května 2017 ……

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

BODNAR, Bogdan. *Webová aplikace pro sestavovaní jídelníčku.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se zabývá návrhem a implementací webové aplikace, která umožní usnadnit sestavování přídělu potravin pro lidi tak, aby dostávali potřebné denní dávky důležitých živin. Na základě funkčních a nefunkčních požadavků na systém je navržena aplikace. V práci jsou diskutovány možnosti řešení problémů, které byly při návrhů a realizací odhaleny. Výsledkem implementační části práce je webový systém umožňující sestavovat příděl potravin dle živin potřebných uživatelům.

Klíčová slova sestavovaní přídělu, sledování spotřeby živin, webová aplikace, Python, Django

Abstract

The paper deals with the design and implementation of a web application that will enable the compilation of the diet plan for people so that they will receive the necessary daily doses of important nutrients. An application is designed based on system requirements. The thesis discusses the possibilities of solving the problems that were discovered during the design and realization. The result of the implementation part of the thesis is a web system that allows to compose the food ration according to the nutrients necessary for the users.

Keywords computational nutrition, tracking nutrient consumption, diet, python, django

Obsah

Seznam ukázek kódu

Seznam obrázků

Úvod

Bakalářská práce se zabývá implementací webové aplikace umožňující uživatelům snadné sledování konzumovaných živin a automatickým sestavováním jídelníčku na základě osobní potřeby nutričních hodnot.

Pro plnohodnotný růst a rozvoj člověka je potřebný plný komplex určitých živin, obsažených v jídle. Celkem člověk potřebuje pro zachování života a udržení zdraví přes padesát nutričních složek [1]. Jejich rovnováha je velmi důležitá pro organismus člověka. Pro většinu lidí dodržování denní normy živin není podstatné, ale existuje určitá skupina lidí, pro které je to nezbytné.

Cukrovka je závažný problém celého světa a obzvlášť v České republice. V roce 2012 se léčilo v ČR s diabetem více než 841 tisíc osob, což představuje přibližně 8% populace. Procento osob léčených pro diabetes dlouhodobě roste. Oproti předchozímu roku došlo k nárůstu o zhruba 16 tisíc osob [2]. Díky udržování diety je možné zabránit vzniku diabetu, ale také dalším nemocím, například žaludečním a střevním onemocněním, obezitě, atd.

Cílem bakalářské práce je navrhnout a vytvořit aplikaci, která výše uvedeným skupinám lidí usnadní sestavování přídělu potravin tak, aby dostávali potřebné denní dávky důležitých živin.

Tato bakalářské práce je sestavená ze čtyř částí. V první části se zabývám analýzou požadavků a zkoumáním existujících řešení problematiky a v další části sestavuji návrh na aplikaci. Důležitou součástí práce, je také kapitola

implementace, která blíže specifikuje strukturu aplikace a problémy řešené během vývoje. V poslední kapitole popisuji provedené testování aplikace. Po těchto kapitolách je uveden závěr, kde shrnuji výsledky mé práce.

Kapitola 1

Analýza

V této kapitole provádím rozbor zadání a průzkum existujících řešení.

1.1 Požadavky

V této sekci jsou rozepsané požadavky na aplikaci, které jsou uvedené v zadání této práce.

1.1.1 Funkční požadavky

FP1: Určení denní dávky živin, nj. základních (kalorie, cukry, tuky), ale i známých vitamínů a minerálů

Popis: Možnost volby uživatelem určitých živin pro sledování. Nastavení denní minimální a maximální potřebné hodnoty zvolených živin

FP2: Zaznamenávání konzumovaných potravin

Popis: Přidávání, upravování a mazání konzumovaných potravin. Nastavení a úprava hodnot konzumovaných potravin

FP3: Zobrazení statistiky konzumování živin za zvolené období.

Popis: Zobrazení hodnot konzumovaných živin v souladu s denní dávkou předem zvolených živin za určité období. Možnost zobrazení konzumovaných potravin za určitý den

FP4: Sestavení přídělu z jídel podle denní dávky

Popis: Možnost generování jídelníčku podle nastavených živin. Příděl musí být sestavený smysluplně a rozmanitě (aplikace např. nesmí stále doporučovat stejné potraviny). Generovaný jídelníček musí být sestaven z hotových jídel, které jsou uvařené z jednotlivých potravin

FP5: Přihlášení přes lokální účet a služby 3. stran

Popis: Možnost vytváření lokálního účtu, nebo registrace pomocí služeb třetích stran např. sociální sítě.

1.1.2 Nefunkční požadavky

1. Poskytované API musí splňovat principy REST a HATEOAS
2. Aplikace musí využívat API poskytované Ministerstvem zemědělství USA
3. Kód i API musí být psán i dokumentován v anglickém jazyce
4. Kód musí být vhodně členěn, splňovat konvence zvoleného jazyka a frameworku a obsahovat extensivní sadu testů

1.2 Současný stav řešení problematiky

Funkcionalitu aplikace, popsanou v požadavcích na aplikaci, je možné rozdělit na dvě souvislé části. Jednou z ní je sledování živin, umožňující zaznamenávání a zobrazení statistiky. Druhá část se zabývá sestavováním přídělu na základě živin potřebných pro uživatele. Žádná aplikace, která zahrnuje dvě zmíněné funkcionality v současnosti neexistuje a je potřeba prozkoumat je odděleně.

1.2.1 Existující služby pro sledování živin

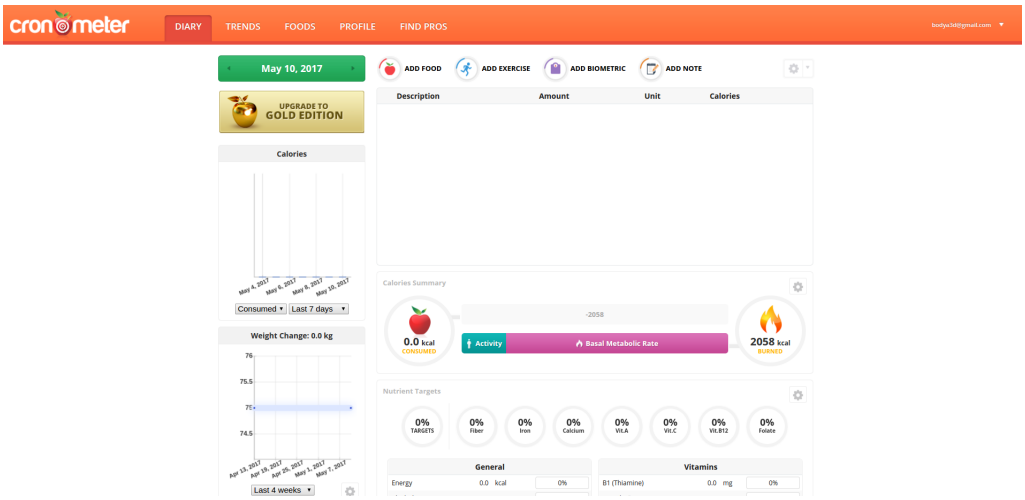
V dnešní době existuje mnoho různých aplikací, které umožňují zaznamenávání konzumovaných potravin, a které následně zobrazí statistiku jejich spotřeby. Většina z nich pracuje jen se základními druhy živin jako jsou energie, cukry, tuky atd. a neumožňují zaznamenávat vitaminy, minerály a ostatní známé živiny uvedené v následujícím seznamu živin [1]:

- Energy
- Water
- Carbs
  - Fiber
  - Starch
  - Sugars
- Lipids
  - Fats
    - Monounsaturated
    - Polyunsaturated
    - Omega-3
    - Omega-6
    - Saturated
    - Trans-Fats
  - Cholestero
- Proteins
  - Cystine
  - Histidine
  - Isoleucine
  - Leucine
  - Phenylalanine
  - Threonine
  - Tryptophan
  - Tyrosine
  - Valine
- Vitamins
  - B1 (Thiamine)
  - B12 (Cobalamin)
  - B2 (Riboflavin)
  - B3 (Niacin)
  - B5 (Pantothenic Acid)
  - B6 (Pyridoxine)
  - Folate
  - Vitamin A
  - Vitamin C
  - Vitamin D
  - Vitamin E
  - Vitamin K
- Minerals
  - Calcium
  - Copper
  - Iron
  - Magnesium
  - Manganese
  - Phosphorus
  - Potassium
  - Selenium
  - Sodium
  - Zinc
- Alcohol
- Caffeine

1.2.1.1 CRON-O-Meter

Jedním z trackerů, které berou v úvahu všechny známé živiny je CRON- O-Meter [3], jenž má veškeré potřebné funkce pro tracker, který ale neumožňuje uživatelům nastavit specifickou denní dávku zvolených živin.

CRON-O-Meter bude sloužit mně částečně jako inspirace.



Obrázek 1.1: *Hlavní stránka CRON-O-Meter*

1.2.2 Existujicí služby vytvoření jídelníčku

V současnosti existuje mnoho aplikací pro sestavování přídělu potravin. Mezi nejpopularnejší patří:

1.2.2.1 EatThisMuch.com

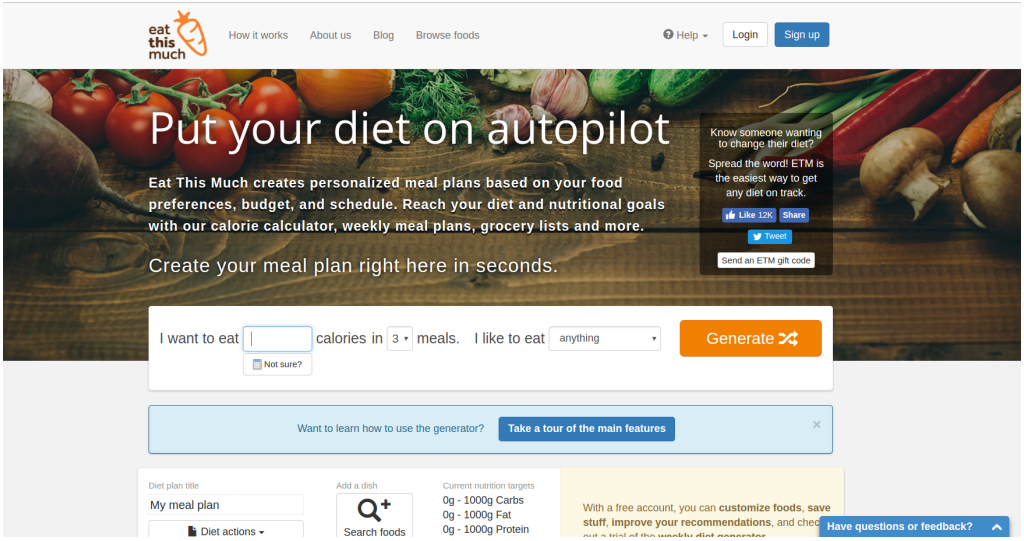
EatThisMuch.com je populární webová služba, která generuje plán stravování podle potřebného počtu energie, množství příjmů a přednosti jídla.

Zajímavou možností této aplikace je zvolení druhů diety například paleo, vegetariánské atd. Aplikace umožňuje uživatelům nastavovat meze sacharidů, tuků, proteinů a má možnost omezit kvantitu sodíku a cholesterolu. Avšak tato aplikace neposkytuje možnost nastavování potřebných mezí makroživin a mikroživin.

1.2.2.2 Free Custom Fitness Meal Planner

CustomMealPlanner.com je webová aplikace, která je určena především pro sportovce. Tato služba generuje plán stravování na základě:

- fyzických parametrů uživatele – váhá, růst, věk, pohlaví
- cíle, které se chystá dosáhnout – narůstání svalů, ztráta váhy atd.



Obrázek 1.2: *Hlavní stránka EatThisMuch.com*

- délce kardio cvičení
- druhu diety

Custom Fitness Meal Planner neumožňuje uživatelům nastavování potřebných mezí živin.

1.2.2.3 Shrnutí

Výše zmíněné služby uvažují v potaz pouze základní živiny (energie, cukry, tuky atd.). Služby generující jídelníček neuvažují žádné živiny, což pro naše cíle nestačí.

Existují také aplikace, které negenerují jídelníček, ale slouží pouze jako seznam jídel a receptů s možností zvolení jídla na určitý den. Tyto aplikace nesplňují pozadavky, takze je v této práci nebudeme rozebírat.

1.2.3 Matematický popis problému

Problém sestavení jídelníčku na základě zvláštní nutriční potřeby se dá popsat jako problém hledání množiny:

$$X = \{f_{m_1}, f_{m_2}, \ldots, f_{m_{k-1}}, f_{m_k}\} \subset \{f_1, f_2, \ldots, f_{n-1}, f_n\} = F$$
$$\{m_1, m_2, \ldots, m_{k-1}, m_k\} \subset \{1, 2, \ldots, n-1, n\}$$

kde X je hledaná množina jídel, F je množina veškerých jídel

Přičemž musí platit:

$$s = \sum_{i=m_1}^{m_k} \nu(f_i) = \sum_{i=m_1}^{m_k} \begin{pmatrix} n_1^i \\ n_2^i \\ \vdots \\ n_{t-1}^i \\ n_t^i \end{pmatrix} \in \begin{pmatrix} < x_1; y_1 > \\ < x_2; y_2 > \\ \vdots \\ < x_{t-1}; y_{t-1} > \\ < x_t; y_t > \end{pmatrix} = N$$

kde s je vektor součtu obsažených v množině X nutričních hodnot,

N je vektor intervalů požadovaných živin

1.2.4 Existující algoritmy pro generovaní jídelníčku

V průběhu rešerše jsou také uvedeny další vědecké práce, které popisují algoritmy generující jídelníček. V této části práce se budu zabývat zkoumáním existujících algoritmů a výběrem nejvhodnějšího z nich.

1.2.4.1 Automated Menu Planning Algorithm for Children: Food Recommendation by Dietary Management System using ID3 for Indian Food Database

Tato vědecká práce se zabývá návrhem algoritmu pro automatické sestavení jídelníčku pro děti. Tento algoritmus je postaven na základě rozhodovacího dřeva [4]. Použité rozhodovací dřevo funguje jako filtr, který rozhoduje, zda konkrétní jídlo může být zvoleno pro dětský jídelníček či nikoli.

Toto dřevo užívá následující rozhodovací faktory:

- dostupnost – ano/ne

- oblíbenost jídla dětmi – ano/ne
- kategorie děti – s podváhou/normální/mající nadváhu
- celkový obsah nutričných hodnot – malý/střední/vysoký

Výše uvedený algoritmus nevyhovuje cílům práce, protože nebere v úvahu zvlaštní potřeby nutričních hodnot. Výsledkem algoritmu není připravený jídelníček, ale jenom seznam jídel vhodných k sestavení menu pro děti.

1.2.4.2 Dietary Menu Planning Using an Evolutionary Method

Mezi existující způsoby automatického generování jídelníčku patří algoritmus navržený Barbarou Koroušić Seljak. Daný genetický algoritmus umožňuje generování týdenního jídelníčku na základě chuti, konzistence, barvy, teploty, tvaru a způsobu přípravy jídel [5].

Zkoumaný algoritmus nefunguje na základě zvláštních nutričních potřeb, kromě toho jsou genetické algoritmy poměrně složité a jsou nad rámec mojí práce.

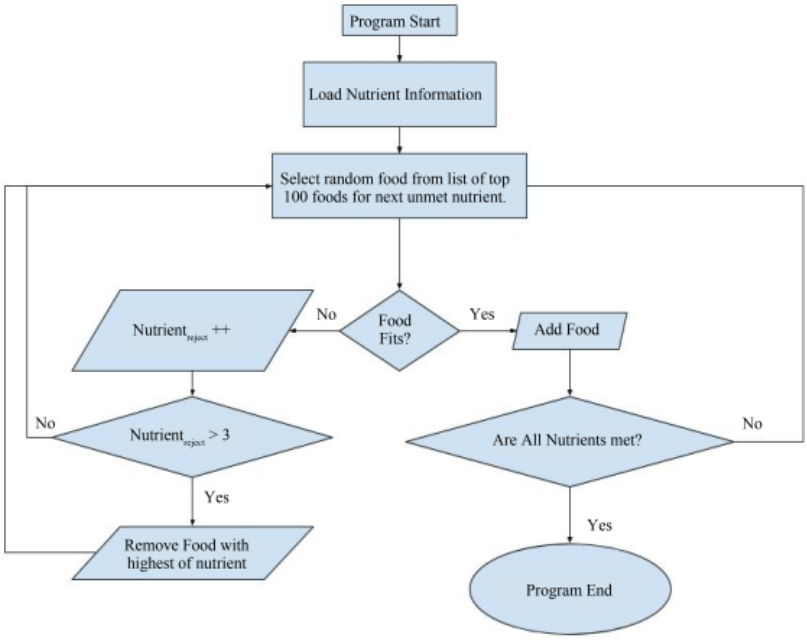
1.2.4.3 An Algorithm to Generate a Diet Plan to Meet Specific Nutritional Requirements

Tento algoritmus slouží pro generování dietního plánu a funguje na základě specifických nutričních potřeb. Daný algoritmus bere v úvahu dolní hranici, ideální hodnotu a horní hranici makroživin a mikroživin z následujícího seznamu:

- Calcium
- Phosphorus
- Magnesium
- Potassium
- Sodium
- Iron
- Manganese
- Copper
- Zinc
- Selenium
- Vitamin A
- Vitamin C
- Vitamin E
- Vitamin K
- Thiamin
- Roboflavin
- Niacin
- Pantothenic Acid
- Folate

Algorimus začíná z prázdného seznamu jídel, v průběhu iterování výše uvedených živin přidává náhodně zvolenou potravinu ze seznamu výrobků z nejvyšším obsahem zkoumané živiny. Pokud při třech iteračních průchozích překračuje horní hranici jedné z živin, potravina s největším obsahem této živiny bude smazána ze seznamu. Algorimus se zastaví, když se potkají veškeré nutriční hodnoty obsažené v generovaném seznamu jídel a potřebné intervaly nutričních hodnot, které si uživatel zvolil, viz. blokové schéma algoritmu 1.3 [6].

Zajímavá možnost tohoto algoritmu je bodování (score) generovaných jídelníčků. V tomto připadě je nutné od uživatele nastavení šesti parametrů pro každou zvolenou nutriční hodnotu, což je pro běžného uživatele příliš náročné.



Obrázek 1.3: *Blokové schéma algoritmu pro generování jídelníčku*

Daná metoda řeší problém, který je popsán v sekci 1.2.3 a bude použit pro vývoj aplikaci, avšak má několik nedokonalostí, které potřebují zlepšení.

Jeden z problémů je sestavení jídelníčků z jednotlivých potravin. Na ukázkovém příkladu generovaný jídelníček obsahuje jednotlivé výrobky, což je naprosto nepoužitelné pro uživatele. Tento problém se dá řešit pomocí předem sestavených jídel z jednotlivých složek, které musí být v uvařeném stavu.

Druhý problém, se kterým se setkáme při použití zkoumaného algoritmu je donucení uživatele k vyplnění normy všech uvedených živin. Problém se snadno řeší pomocí poskytnutí uživateli možnosti si zvolit potřebné živiny pro generování jídelníčku, avšak nastavení intervalu základních živin (energie, tuky, proteiny a sacharidy) je nezbytné pro automatické sestavení použitelného jídelníčku.

1.3 Technologie

1.3.1 API

Ze zadání a nefunkčních požadavků mé práce vyplývá, že serverová část webové aplikace musí mít vhodné API pro komunikaci s klientskou stranou. Z možností architektury byla zvolena RESTful architektura.

1.3.1.1 REST

RESTful - je architektura rozhraní pro komunikaci mezi distribuované webové služby.

Pro komunikaci se využívá URI, HTTP a XML nebo JSON. Pro jednotný přístup ke zdroji a provedení CRUD metod se využívá metody HTTP protokolu viz. tabulka 1.1. RESTful architektura je podmíněna následujícími principy:

Jednotné rozhraní: každý zdroj v REST architektuře je reprezentován URI. Každá zpráva odeslána od klienta k serveru má v sobě veškerou informace pro její zpracování serverem.

Bezstavová intrakce: žádná dočasná informace není ukládána na serveru mezi zpracováním klientských požadavků. Všechny potřebné informace jsou odeslány v URL, parametrech, hlavičce nebo v těle dotazu.

Možnost cachování: klient může cachovat odpovědi ze serveru. Server musí definovat v odeslaných datech možnost a dobu cachování.

Klient-Server: klient je oddělen od serveru a není spojen s datovým úložištěm.

CRUD operace	HTTP metoda
Create	POST
Read	GET
Update	PUT
Delete	DELETE

Tabulka 1.1: Vztah CRUD operací v RESTful architektuře k HTTP metodám

Vrstvený systém: klient nemůže kdykoliv zjistit, zda je připojen ke koncovému serveru nebo k mezi-serveru. Středová vrstva pomáhá prosazovat zásady zabezpečení a zlepšovat škálovatelnost systému tím, že umožňuje vyvažování zátěže.

Kód na vyžádání: server může rozšířit funkci klienta přenosem spustitelného kódu [7].

REST nabývá na významu a stává se spolu s JSON defacto standardem pro API webových služeb. Jeho rozšíření napomohlo i to, že se nijak zásadně neliší od standardního volání a získávání dát pomocí HTTP, pouze je zobecňuje. Dá se říct, že REST je architektura, která umožňuje CRUD operace pomocí standardních HTTP dotazů [8] .

1.3.1.2 HATEAOS

HATEOAS — omezení nebo pravidlo pro REST architekturu. Pricip HATEOAS spočívá v tom, že hypermedia-driven server poskytuje klientům informace k dynamickému procházení REST rozhraní webu tím, že obsahuje v odpovědi hypermedie odkazy [9]. V podstatě to znamená, že odpovědi z webové služby musí obsahovat informace o požadovaném objektu v podobě hypermedia odkazů na vztahující se objekty a na objekt samotný. Ukažeme si to na malém příkladu, předpokládejme, že v naší aplikaci uživatel má profil a seznam konzumovaných potravin. Tedy odpověď na požadavek na informaci o uživatele je na ukázce 1.1.

Taková struktura odpovědi ze serveru umožňuje klientovi nejenom přijímat informace o objektech na konkrétním URI, ale také například odesílat korektní požadavky na vztahující se objekty nebo operace. Přičemž klient si nemusí

```
{
 "firstname":"Dennis",
 "lastname":"Crawford",
 "links":[
    {
       "rel":"self",
       "href":"http://localhost:8080/user/1"
    },
    {
       "rel":"profile",
       "href":"http://localhost:8080/user/1/profile"
    },
    {
       "rel":"consumed-products",
       "href":"http://localhost:8080/user/1/consumed-products"
    }
 ]
}
```

Ukázka kódu 1.1: HATEOAS – příklad odpovědi

pamatovat URI vztažených objektů, které serverová strana může dynamicky měnit v závislosti na stavu aplikace.

1.3.1.3 NDB API

Jedním z nefunkčních požadavků na aplikace ze zadání je použití API nad NDB, které poskytuje Ministerstvo zemědělství USA. NDB je velká databáze, obsahující informace o obsahu 150 složek nutričních hodnot z více než 8,5 tisíc potravinách a značkových výrobků [10]. Jednou z výhod NDB je informace o potravinách v uvařeném stavu, což se hodí pro sestavení seznamu uvařených jídel. NDB API poskytuje následující možnosti získání informace:

Food Reports: obsah nutričních hodnot v jednotlivých potravinách

Lists: seznamy potravin, živin, nebo skupin potravin

Nutrient Reports: obsah konkrétních nutričních hodnot v jednom nebo více potravinových výrobcích

Search: full-text hledání potravin v databázi

Poskytované rozhraní umožní snížit objem uložených dat v budoucí databázi aplikace, dále také zajistí obdržení aktuální informace o obsahu nutričních hodnot.

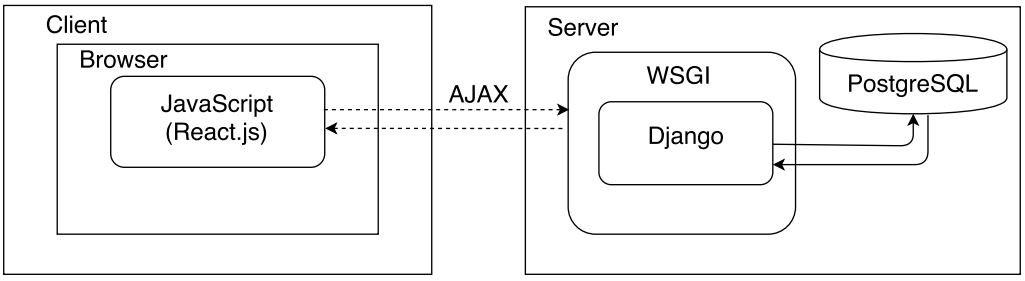
Licence NDB se šíří jako volné dílo (public-domain), což usnadňuje užívání této služby, avšak pro užívání API je potřeba zaregistrovat aplikaci a získat API klíč. Kromě toho poskytované API má omezení na tisíc požadavků za hodinu, což pro naše cíle bohatě stačit.

Kapitola 2

Návrh

Celá aplikace bude sestavená ze serverové a klientské části. Na serverové části bude běžet aplikace napsaná v jazyce Python pomocí frameworku Django. Obsluhovat požadavky z klientské části bude WSGI server. Jako datové úložiště bude sloužit databáze PostgreSQL. Klientská část, poběží na počítače klienta v prohlížeči.

Při prvním požadavku na server klientský prohlížeč obdrží ze serveru JavaScript kód (tzv. bundle), který v podstatě je JS aplikace napsaná pomocí knihovny React.js. Dal komunikace mezi serverem a klientem (pokud je potřebná) bude probíhat v podobě AJAX požadavků a odpovědí viz. schéma 2.1.



Obrázek 2.1: *Schéma architektury aplikace*

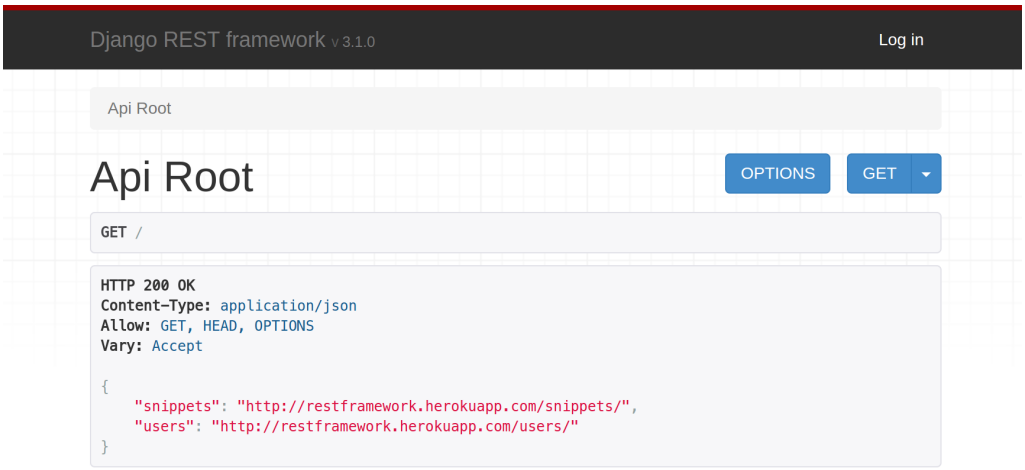
2.1 Backend

2.1.1 Django

Konečnou volbou pro vývoj serverové části je Django framework verze 1.10.5.

Django je vysokoúrovňový webový framework, který umožňuje rychlý vývoj webových aplikací s čistým a pragmatickým návrhem. Django bere na sebe většinou nepříjemnosti vývoje webových služeb a umožňuje se soustředit na vývoje funkcionality aplikace. Django je poskytován zdarma a má otevřený zdrojový kód [11]. Django aplikace poběží na vestavěném serveru WSGI.

Django REST framework Jelikož jedním z nefunkčních požadavků na aplikaci je vytváření REST API, řešením je tedy použití tzv. Django REST frameworku verzi 3.6.2 [12]. DRF zahrnuje v sobě různé nástroje usnadňující vytváření REST architektury, například serializace objektů, která podporuje ORM a tzv. Web browsable API, který umožňuje komunikovat s API pomocí prohlížeče bez použití jiných pomocných nástrojů viz. obrázek 2.2.



Obrázek 2.2: *Ukázka Web browsable API v DRF*

2.1.2 PostgreSQL

Ačkoliv ORM, které poskytuje Django, umožňuje pracovat s několika relačními databázemi (SQLite, MySQL atd.), jsem zvolil pro vývoj a konečné nasazení PostgreSQL verze 9.6.2.

2.2 Frontend

2.2.1 React.js

Klientská JavaScript aplikace bude napsána pomocí knihovny React.js verzi 15.4.2 [13], který používá rozšíření JavaScriptu — JSX [14], který poskytuje tzv. syntaktický cukr (syntactic sugar) a přidává do JavaScriptu možnost vytvářet komponenty pomocí HTML kódu. Na kompilování JSX kódu je nutné použít speciální nástroj, kterým bude sloužit babel. Konečný balíček (bundle) s JS aplikací bude sestaven pomocí nástrojů webpack, který kromě toho umožňuje zapojit nástroje pro snadnější vývoj aplikace například hot-reloader a source-map. Sestaveny pomocí webpack bundle bude odeslán ze serveru na první požadavek od klienta. Další komunikace mezi React.js aplikaci a API na serverové straně bude probíhat prostřednictvím AJAX požadavků.

2.3 Další nástroje použité během vývoje

CVS Pro správu verzí projektu bude sloužit Git [15]. Pro uložení gitového repositáře bude použit GitHub [16].

Správa Python modulu PyPA je doporučený nástroj pro instalace Python modulů [17]. Moduly se budou instalovat do virtuálního prostředí (virtualenv).

Sprava JS balicku JS balíčky budou instalované a spravované pomocí nástroje npm [18].

2.4 Uživatelské role a případy užití

Na základě funkčních požadavků a analýzy existujících služeb byly rozpracované následující skupiny uživatelů:

2.4.1 Nepřihlášený uživatel

Anonymní, nepřihlášení, a neregistrování uživatelé. Můžou se zaregistrovat nebo přihlásit se, jinak nemají žádné práva.

2.4.2 Přihlášený uživatel

Přihlášený uživatel je oprávněn prohlížet si, přidávat, upravovat a mazat ze svého účtu konzumované potraviny. Může nastavovat svou potřebnou denní dávku určitých živin a na jejich základě generovat jídelníček na den.

2.4.3 Administrator

Administrátor má práva přihlášeného uživatele, také má právo přidávat, upravovat a mazat jídla, které slouží pro generování přídělu.

2.4.4 Případy užití

Na základě funkčních požadavků a uživatelských rolí byl sestaven diagram 2.3 případů užití. V diagramu 2.3 je naznačeno, jak výše uvedené skupiny uživatelů mohou používat aplikace.

2.5 Návrh uživatelského rozhraní

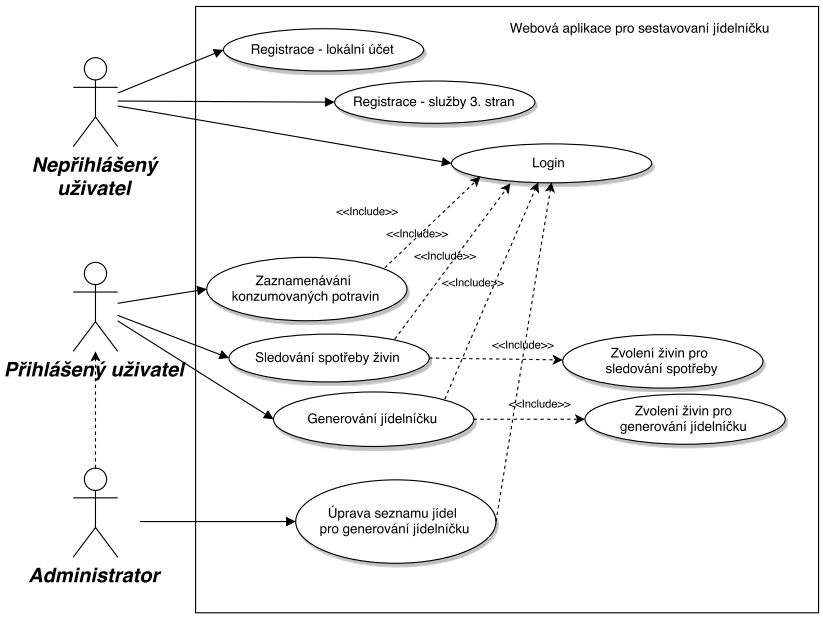
Na základě uživatelských rolí a případů užití jsou navrženy nasledující stránky klientské aplikaci:

Home: úvodní stránka projektu

Login: stránka pro přihlášení

Registration: registrační stránka

- následující stránky jsou dostupné po úspěšném přihlášení:



Obrázek 2.3: *Diagram případů užití*

Profile: nastavení profilu. Na této stránce si uživatel může zvolit živiny pro sledování a pro generování jídelníčku a nastavení jich hodnot

Overview: přehled konzumovaných potravin za den, týden a měsíc vzhledem k nastavené denní spotřebě

Add Prodcut: přidání nového záznamu konzumovaných potravin. Na této stránce uživatel je schopen prohlédnout si informace o obsahu nutričních hodnot v potravinách

Consumed products: stránka se záznamy o konzumovaných potravinách. Uživatel může si prohlédnout seznam konzumovaných potravin za určitý den, který je možné zvolit pomocí kalendáře a smazat je v případě chybného zaznamenávání

Diet plans: stránka s generací jídelníčku. Po nové generaci uživatel obdrží seznam jídel v jídelníčku a celkový počet nutričných hodnot předem zvolených v nastavení profilu

Pro lepší přehled byly navrženy náhledy webového rozhraní (wireframy) stránek viz. přílohy na stránce 61, které pak byly použité pro implementační část práce.

Kapitola 3

Implementace

V této kapitole je rozepsaná implementační část práce.

3.1 Vnitřní struktura aplikace

Pro vývoj aplikace jsem zvolil následující strukturu, která je předepsaná na strance 40. Django framework podporuje rozdělení na znovupoužitelné aplikace (tzv. apps). Mé vyčleněné aplikace jsou detailně popsané v následujících sekcích.

Zdrojový kód React.js aplikace je umístěn ve stejném adresáři pro usnadnění vývoje. Podle potřeby je možné oddělit JS aplikace do izolovaného projektu a nechat je bežet na node.js serveru například pomocí express.js.

3.2 Backend

3.2.1 Django-aplikaci

Jak už bylo zmíněno v kapitole návrhu na stránce 34, práce pro vývoj backendové části jsem použil framework Django. Každá Django aplikace je reprezentována python-balíčkem a mají podobnou adresářovou strukturu viz. diagram 3.2.

V této sekci v každé uvedené django-aplikaci je přiložen třídní diagram. Celkový třídní diagram seskupený podle django-aplikací je v příloze A.1.

- `main_app/` ..... hlavní (konfigurační) app
- `profiles/` ..... app s profily uživatelů
- `tracker/` ..... app pro sledování živin
  - `ndb_api/` ..... python-module pro práce s API NDB
- `diet_planner/` ..... app pro generování jídelníčku
- `webpack/` ..... nástroj pro sestavení JS kódu (bundler)
- `react_frontend/` ..... react.js aplikace
  - `src/` ..... zdrojové JS kódy
  - `static_dist/` ..... sestavené JS kódy
- `docs/` ..... dokumentace

Obrázek 3.1: *Adresářová struktura projektu*

- `fixtures/` ..... adresář s inicializacnimi daty
- `migrations/` ..... databázové migrace
- `tests/` ..... adresář s testy
- `__init__.py` ..... inicializace python modulu
- `admin.py` ..... generování administračního rozhraní
- `models.py` ..... modely
- `permissons.py` ..... přistupová práva
- `serializers.py` ..... serializaci objektu (DRF)
- `urls.py` ..... adresy
- `views.py` ..... pohledy na modely

Obrázek 3.2: *Struktura Django aplikaci*

3.2.1.1 Profiles

Aplikace profiles se zabývá spravováním uživatelských údajů. Pomocí této aplikace uživatel má možnost zvolit si určité živiny pro sledování a pro generování jídelníčku a nastavení jich hodnot.

Z několika možností rozšíření uživatelského modelu [19] jsem zvolil vytváření modelu profilu s relace jedné ku jedné s uživatelským modelem viz. diagram 3.3. Vytváření objektu profilu při vytváření nového uživatele (např. při registraci) zajišťuje vestavěný v Django ”odesílatel signálu” tzv. Signals [20].

Třídní diagram této django-aplikace je na obrázku 3.3 na strance 47.

```
from django.db.models.signals import post_save
from django.dispatch import receiver

# ... Profile model ...

    @receiver(post_save, sender=User)
    def create_user_profile(sender, instance, created, **kwargs):
        if created:
            Profile.objects.create(user=instance)

    @receiver(post_save, sender=User)
    def save_user_profile(sender, instance, **kwargs):
        instance.profile.save()
```

Ukázka kódu 3.1: Příklad použití Django Signals pro vytváření profilu uživatele

3.2.1.2 Tracker

Tato aplikace slouží pro zaznamenávání a další sledování konzumovaných potravin a obsažených živin.

Třídní diagram této django-aplikace je na obrázku 3.4 na stránce 48.

Vzhledem k tomu, že NDB disponuje informace o živinách, počet kterých je omezen na 150 položek. Bylo řešeno uložit je do inicializačních dat (tzv. fixtures).

Jelikož NDB API umožňuje hledat potraviny ve své databázi podle názvu, bylo rozhodnuto, že hledáním se bude zabývat klientská aplikace. To znamená, že pro zaznamenávání konzumované potraviny fungují následující princip: klientská část, po hledání požadovaného výrobku, obdrží identifikační klíč z bázi NDB, který pak odešle na API serveru spolu s nastavenou hodnotou uživatelem. Pak serverová strana samostatně požaduje data o obsažených nutričních hodnotách ve zvoleném produktu z NDB a spočítá podle hodnoty skutečný obsah nutričních hodnot ve zvoleném výrobku. Ilustrace fungování tohoto principu je vidět na zjednodušeném schématu API požadavků 3.5.

3.2.1.3 Diet Planner

Aplikace pro generování denního jídelníčku funguje na základě algoritmu, který je popsán v rešeršní kapitole na stránce 27.

Třídní diagram této django-aplikaci je na obrázku 3.6 na strance 49.

Jak už bylo zmíněno, tento způsob má problém generování použitelného jídelníčku kvůli výběru ze seznamu jednotlivých potravin. Řešením problému je vytváření seznamů plnohodnotných jídel, které jsou složené z několika ingrediencí. Žádný zdroj obsahující informace o složkách obsažených v jídlech (v uvařeném stavu) vzhledem k NDB neexistuje. A proto data pro sestavení testovacího seznamu jídel jsem čerpal z webového portálu `www.taste.com.au` [21]. Pro testování bylo sestaveno celkem 49 jídel. Data o jídlech jsem přidal do inicializačních dat aplikace (fixtures).

Příklady generování jídelníčků:
Zakladní nastavení denní spotřeby:

Name	Lower Bound	Upper Bound
Energy	2000	2500
Protein	70	240
Lipid (Fat)	50	100
Carbohydrate	245	460

Výsledné jídelníčky:

Name	Count
Fish cakes	1
Cheesecake	1
Cherry tomato and chickpea salad	2
Citrus chicken	1
Beef stroganoff	1
Fish with avocado salsa	1
Avgolemono	1
Coconut Eton mess	2

3.2.2 Komunikace s NDB

Komunikace s databází poskytované Ministerstvem zemědělství USA [22] jsem oddělil v samostatném balíčku (viz. struktura projektu na stránce 40). Před použitím tohoto balíčku je potřeba přidat do konfiguračního souboru API klíč, který byl získán při registraci aplikace na portálu NDB.

3.2.3 Administrátorské rozhraní

Jak plyne z uživatelských rolí, které jsou popsány v kapitole rešerše na stránce 36, aplikace musí mít vhodné administrátorské rozhraní, které umožní snadné přidání jídel.

Administrátorské rozhraní aplikace bylo vytvořeno pomocí zabudovaného v Djangu automatického generování rozhraní pro administrátorské záležitosti. Stránka s přidáním a úpravou jídel je ukázána na obrázku 3.7.

3.2.4 Autentizace

Proces ověření identity uživatelů je implementován použitím tokenu. Po úspěšném přihlášení klient dostane klíč, který musí přiložit k hlavičce s každým AJAX požadavkem.

3.2.5 Přihlášení přes služby tretich stran

Registrace a přihlášení uživatelů pomocí služeb třetích stran je zajištěno použitím aplikace s otevřeným zdrojovým kódem django-allauth [23]. Tento balíček usnadňuje integraci s většinou známých služeb užívajících protokol OAuth a OAuth2.

Z různých možností zvolení služeb třetích stran ve své aplikaci jsem zvolil Facebook [24], který patří mezi nejpopulárnější sociální sítě v dnešní době. Pro zajištění autentizace pomocí služby Facebook je nezbytné zaregistrovat aplikaci na portále facebook for developers [25].

3.2.6 Autorizace

Přístupová práva jsem řešil pomocí systému práv, který je vestavěný v Djangu. Rozhodování o povolení přístupu probíhá na základě příslušnosti požadovaného objektu k uživateli.

3.2.7 Dokumentace

Podle zadání práce zdrojové kódy a API byly dokumentované.

3.2.8 Dokumentace kódu

Dokumentace zdrojového kódu byla vytvořena pomocí nástrojů pro generování dokumentace Sphinxdocs [26]. Sestavená dokumentace je ve složce `docs/_build`.

3.2.8.1 API

Dokumentace použití konečných bodů (endpoints) API byla vytvořena pomocí nástrojů [27] a připojena pro prohlédnutí na URL `<server_url>/docs/`.

3.3 Frontend

Jak už bylo zmíněno v návrhové kapitole pro implementace klientské části aplikace jsem použil knihovnu React.js. Ukázka adresářové struktury JS aplikaci je na stránce 50.

3.3.1 Kontejnery

Na základě funkčních požadavků, které jsou popsané v rešeršní kapitole na stránce 22 a vytvořených náhledů (viz. přílohy na stránce 61), byly implementované odpovídající kontejnery. Kontejnery – tzv. chytré komponenty (smart components) mohou narozdíl od hloupých (dumb components) spravovat svůj stav.

3.3.2 Redux

Knihovna React.js nutí předávát data od rodičovských komponentů k potomkovým (parent-to-childs data flow), Což není pohodlné, pokud chceme sdílet stejná data v různých komponentech. Například ve každém z komponentů bychom potřebovali uživatelská data pro vykonání AJAX požadavků.

Řešením tohoto problému bylo užití kontejneru stavu (state container). Kontejner stavu pro sdílení dat mezi komponenty byl vytvářen pomocí frameworku Redux. Redux je framework sestavený s množiny knihoven, které umožňují oddělit a spravovat stav aplikace v samotném kontejneru [28].

Redux realizuje v aplikaci architekturu podobě MVC, ale narozdíl od MVC, Redux pracuje s (akcemi) actions, reducery (reducers), úložišti (stores) a komponenty (components) nebo kontejnery (containers) viz. adresářová struktura aplikaci na stránce 50.

3.3.3 Routing

Pro routovaní kontejnerů byla použita knihovna react-router. Konfigurace routovaní patří do souboru route.js v kořenovém adresáři zdrojových kódů aplikace. Historie návštěvy stránek je spravována pomocí balíčku browserHistory, který patří do knihovny react-router.

3.3.4 AJAX požadavky

Asynchronní JS požadavky prochází pomocí knihovny axios [29]. Tato knihovna má několik výhod, například automatické parsovaní JSON dat a podpora klientské strany proti CSRF útokům, viz. ukázka z příkladem použití této knihovny 3.2.

3.3.5 Styly a Komponenty

Pro zajištění adaptivity a sourodého pohledu komponentů uživatelského rozhraní byl použit framework Bootstrap [30]. Většina komponentů (navigace, tlačítka, tabulky, formuláře atd.) byla použita z knihovny React-Bootstrap. [31].

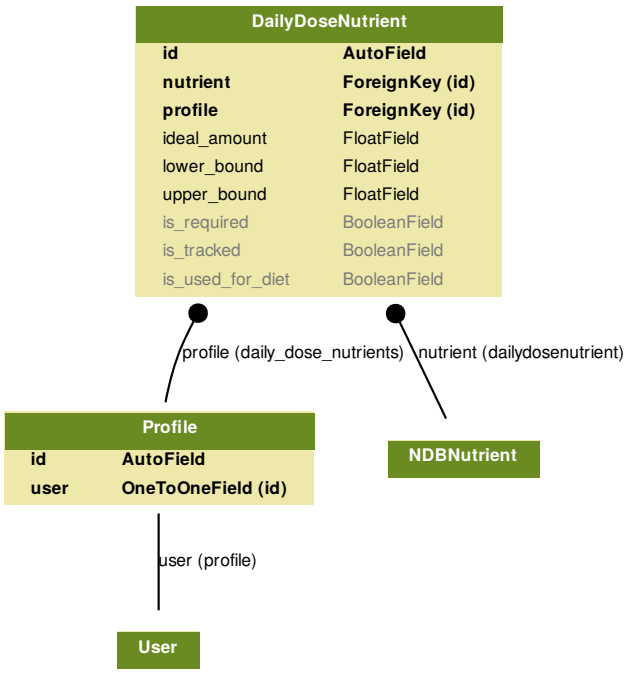
```
axios.post(`${SERVER_URL}/api/user/${state.auth.userId}/products/`, {
    'id_in_ndb': product,
    'value': value
  }, {
    headers: {
      Accept: 'application/json',
      Authorization: `Token ${state.auth.token}`
    }
  })
  .then(resp => {
    dispatch({
      type: ADD_PRODUCT_SUCCESS,
      payload: { statusText: "The product was successfully added." }
    });
  })
  .catch(err => {
    // Zpracování chyby
    });
  });
```

Ukázka kódu 3.2: Příklad použití knihovny axios pro přidání nového záznamu

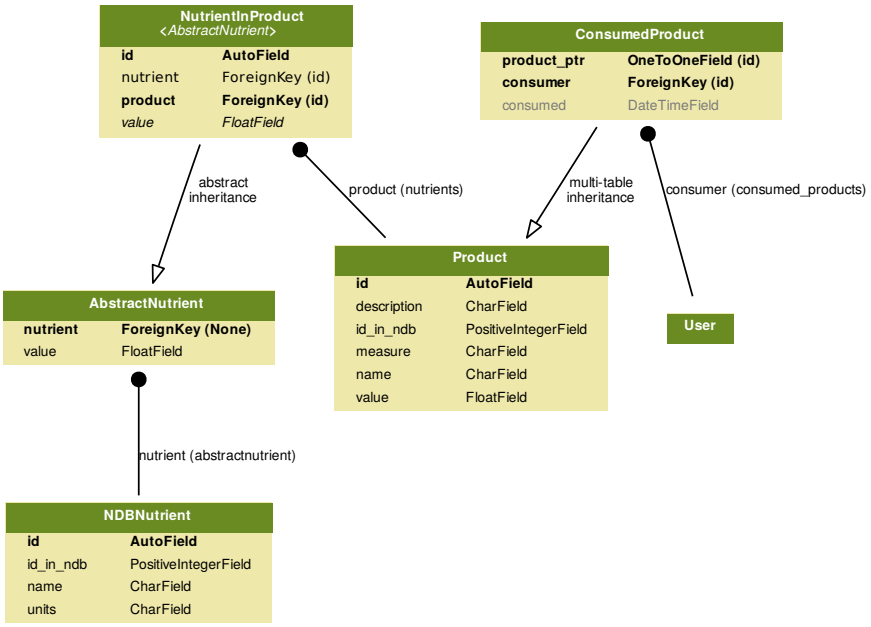
V komponentu AddProduct byla použita knihovna React Bootstrap Typeahead [32]. Tato knihovna poskytuje textové rámečky s možností hledání a nápovědou na základě zadaných uživatelem písmen nebo klíčových slov. Zobecněně řečeno, po každém zadaném znaku se provádí AJAX požadavek do NDB a klient obdrží seznam potravin se shodným názvem. Po tomto seznamu se pak provádí filtrace a uživatel je schopen zvolit určitou potravinu pro následující přidání záznamu.

3.3.6 Shrnutí

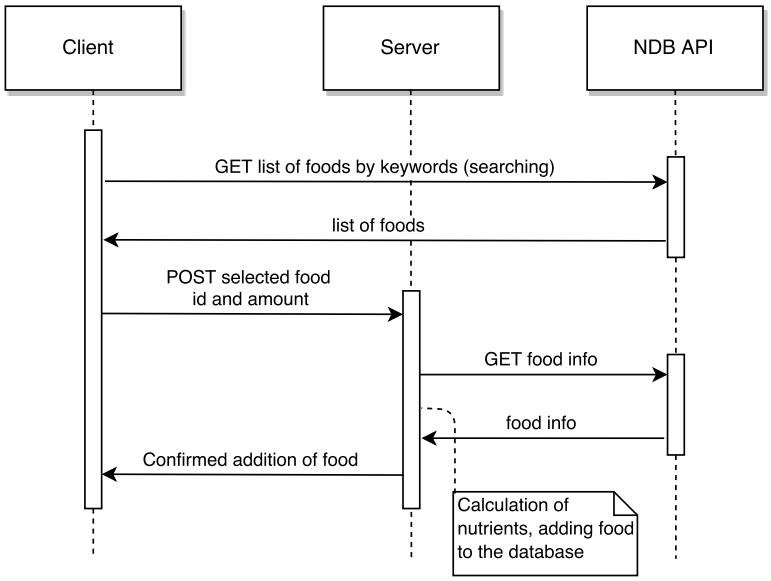
Snímky obrazovky s uváděny v této sekci, komponenty aplikace jsou k dispozici v přílohách na stránce 65.



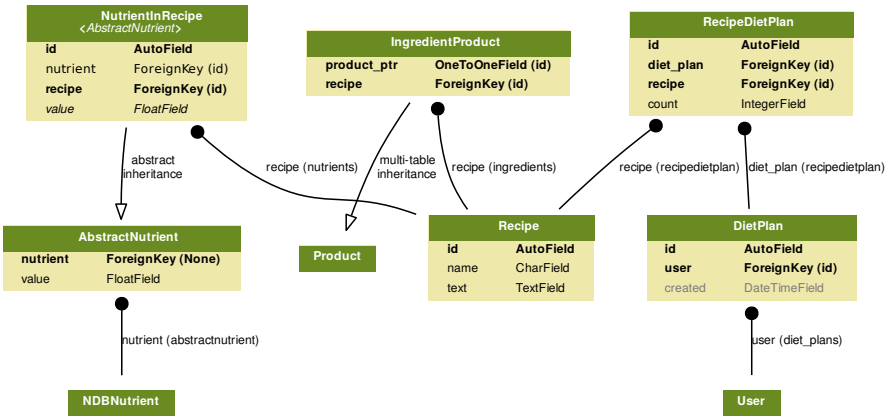
Obrázek 3.3: *Diagram tříd django-aplikaci profiles*



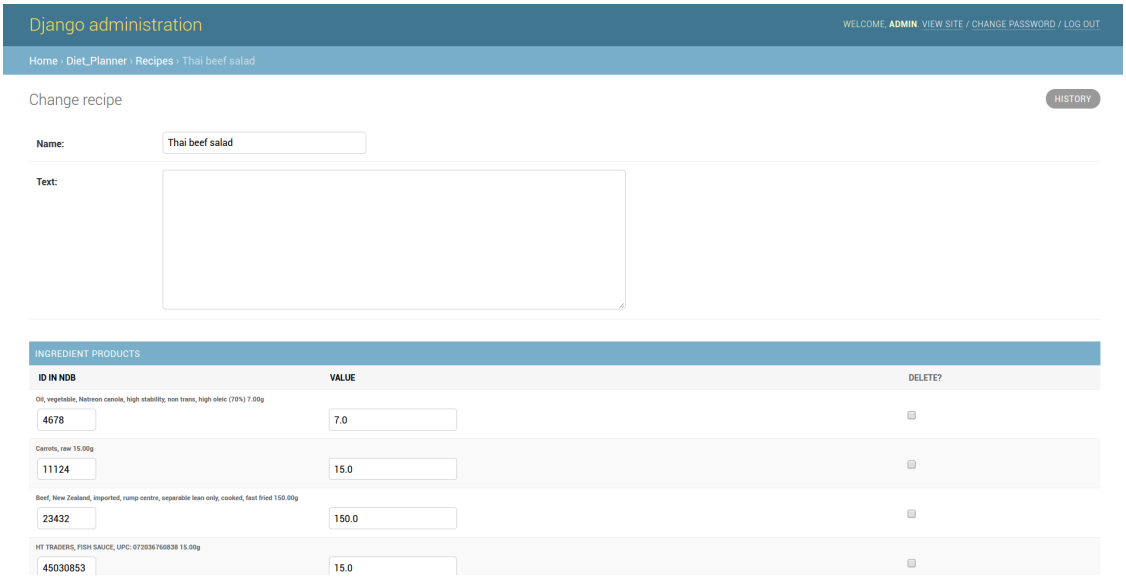
Obrázek 3.4: *Diagram tříd django-aplikaci tracker*



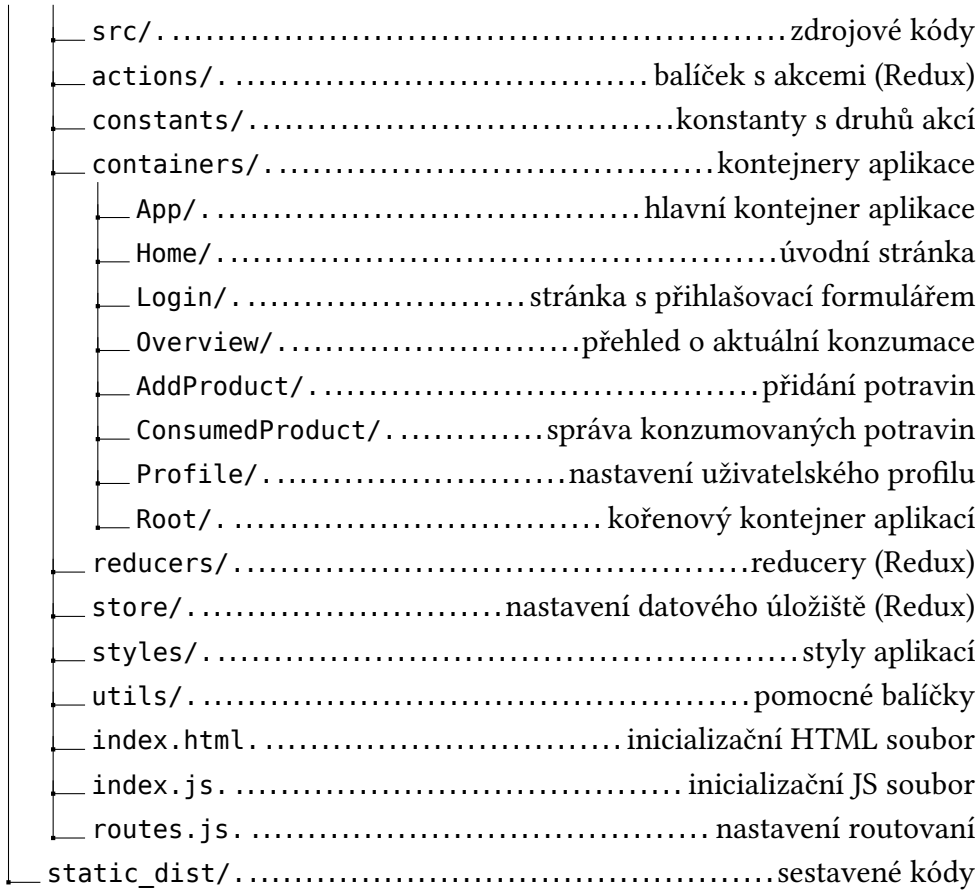
Obrázek 3.5: *Zjednodušené schéma API požadavků probíhající při přidání nového jídla*



Obrázek 3.6: *Diagram tříd django-aplikaci diet-planner*



Obrázek 3.7: *Administrátorské rozhraní pro přidání a úpravu jídel*



Obrázek 3.8: *Adresářová struktura JS aplikace*

Kapitola 4

Testování

4.1 Jednotkové testy

4.1.1 Testovací data

Potřebné data pro testování jsou uloženy do inicializačníchch dat (fixtures). Inicializační data se nahrávají do testovací databáze automaticky před spouštěním testů.

4.1.2 Testovaní modelu

Testování modelů se provádí pomocí zabudovaného v Django rozšiřeni nad balíčkem unittest ze standardní Python knihovny. Toto rozšiření umožňuje provádět každý test uvnitř transakce, což zajišťuje izolace testů.

V testování modelů se provádí kontrola:

- CRUD operaci
- Chování vztažených modelů
- Přepsané metody (override methods)

4.1.3 Testovaní API

Testování API se provádí pomocí třídy APITestCase s knihovny rest_framework. Instance této třídy obsahují klient jako atribut, pomocí tohoto klientu se dá odesílat požadavky na API a pak kontrolovat jich výsledky.

4.1.4 Mock NDB

Pro testování musel jsem udělat mockovani odpovědi z NDB API, aby testy se mohli provádět lokálně. Pro teto účely jsem vytvořil vlastní dekoratér nahrazující funkce get_product_from_ndb pomocí knihovny mock [33].

4.2 Testování použitelnosti

Testování použitelnosti (usability testing) je průzkum, který se provádí za účelem určení pohodlnosti užívání zvoleného artefaktu a případně odhalení problémů, které ztěžují nebo omezují uživatelům používaní zkoumaného artefaktu.

Ve své práci jsem provedl dále popsané testování použitelnosti webového uživatelského rozhraní vytvořené aplikace s cílem odhalit problémy použitelnosti a navrhnout řešení.

4.2.1 Průzkum uživatelů

Vzhledem k tomu, že aplikace zatím není v provozu a nejsou dostupná statistická data o uživatelích dané služby, případně skupiny uživatelů a typické případy užívání jsou odvozené od specifiku a zaměření portálu.

Sportovec

Cíle užívání portálu sportovcem, může být zvýšení svalové hmoty, snížení tukového polštáře nebo běžná podpora zdravotního stavu. Vzhledem k těmto účelům typickým případem užití aplikace je zaznamenávání konzumovaných potravin a další sledování základních nutričních hodnot.

Zvláštně chorobný člověk

Druhý potenciálním uživatelem portálu je skupina lidí, kteří mají zvláštní nemoc, při které potřebují přesně dodržovat denní dávku zvolených živin. Typický případ užití aplikace této skupiny uživatelů je zvolení speciálních živin pro soustředěné sledování.

Pro výše uvedené skupiny generování jídelníčku je dobrou pomůckou pro snadné dodržování potřebných nutričních hodnot a testování této činnosti vzhledem k všem skupinám nezbytné.

4.2.2 Příprava testování

Na základě průzkumu uživatelů a typických úkolů byly připravené odpovídající testovací scénáře, vstupní a výstupní dotazníky viz. přílohy na stránce **??**. Pro testování byla zvolena tzv. metoda hallway-testing [34], základní myšlenka této metody je zvolení náhodných respondentů. Výhodou užití hallway-testing metody je vyhnutí se dlouhému hledání speciálních respondentů z cílových skupin, přičemž náhodnost jejich výběru zajišťuje velké pokrytí, což pomáhá odhalit větší množinu závažných problémů.

4.2.3 Probíhání testování

Testování použitelnosti se probíhalo v laboratoři [35]. Zúčastnili se tři respondenty a byly použité dva scénáře viz. stranka **??**. Během testování byli natočené AV záznamy obrazovky a chování respondentů při splnění testových úkolů.

4.2.4 Výsledky testování

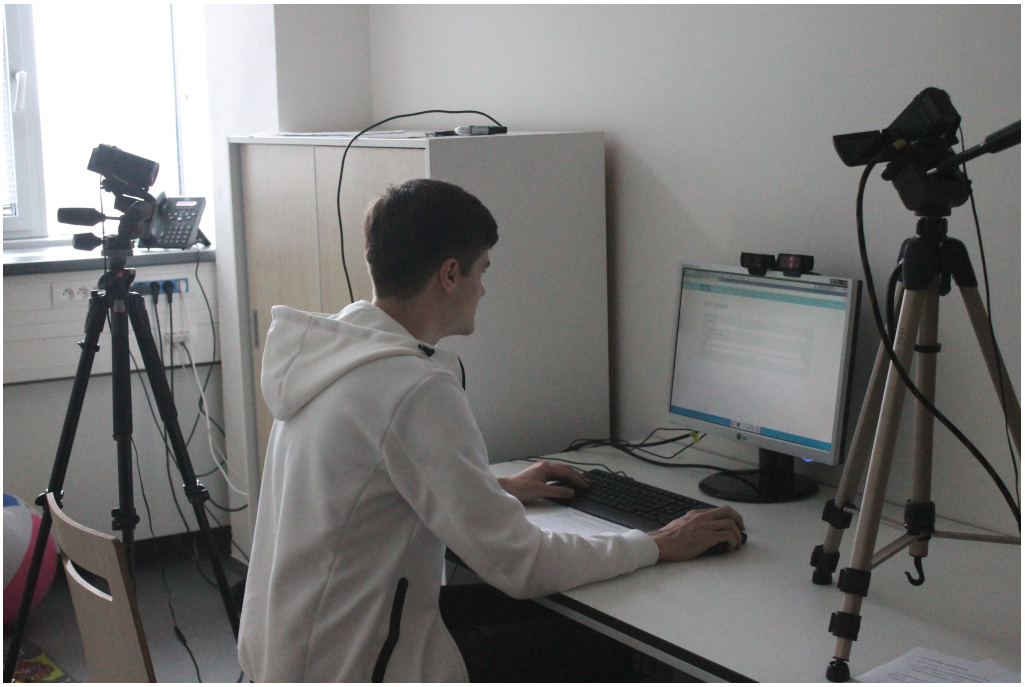
Účelem testování bylo ověřit použitelnost uživatelského rozhrání.

Všichni respondenty se shodli, ze aplikace je celkem použitelná, má dobrý vzhled a ovládací prvky. Však během analýzy AV záznamů byla nalezená i sada problémových míst:

- některé popisky a termíny byly nejasné a matoucí
- není jasné že informace o potravních lze získat pomocí stránky Add Product

- chyby odezva při obnovení nastavení a mazání záznamů
- chyby odezva při mazání záznamů

Některé drobné opravy bílý provedené okamžitě, avšak větší změny nejsou v rozsahu této práce.



Obrázek 4.1: *Testování použitelnosti*

Závěr

Cílem bakalářské práce byl návrh a implementace webové aplikace, umožňující snadné sestavování přídělu potravin uživatelům tak, aby dostávali potřebné denní dávky důležitých živin. Výsledek své bakalářské práce považuju za úspěšný, protože se mi podařilo dosáhnout všech vytyčených cílů.

Po analýze požadavků na aplikaci, existujících řešení a požadovaných technologií jsem provedl návrh a implementace serverové a klientské části aplikace, které nakonec byly otestovany.

Za velký přínos pro sebe považuju získané praktické zkušenosti. Během této práce jsem se naučil vyvíjet plnohodnotné webové aplikace s použitím moderních technologií. Získal jsem praxe vývoje backendové části poskytující REST API pomocí frameworku Django, také jsem se naučil vyvíjet klientské aplikace pomocí knihovny React.js.

Výhled

Tato aplikace má mnoho možností pro další rozvoj:

Rozdělením aplikace na serverovou a klientskou část bylo umožněno snadné vytváření dalších klientů, například pro nejpopulárnější mobilní platformy. Kromě toho knihovna React Native umožňuje vytváření nativních aplikací pro iOS a Android.

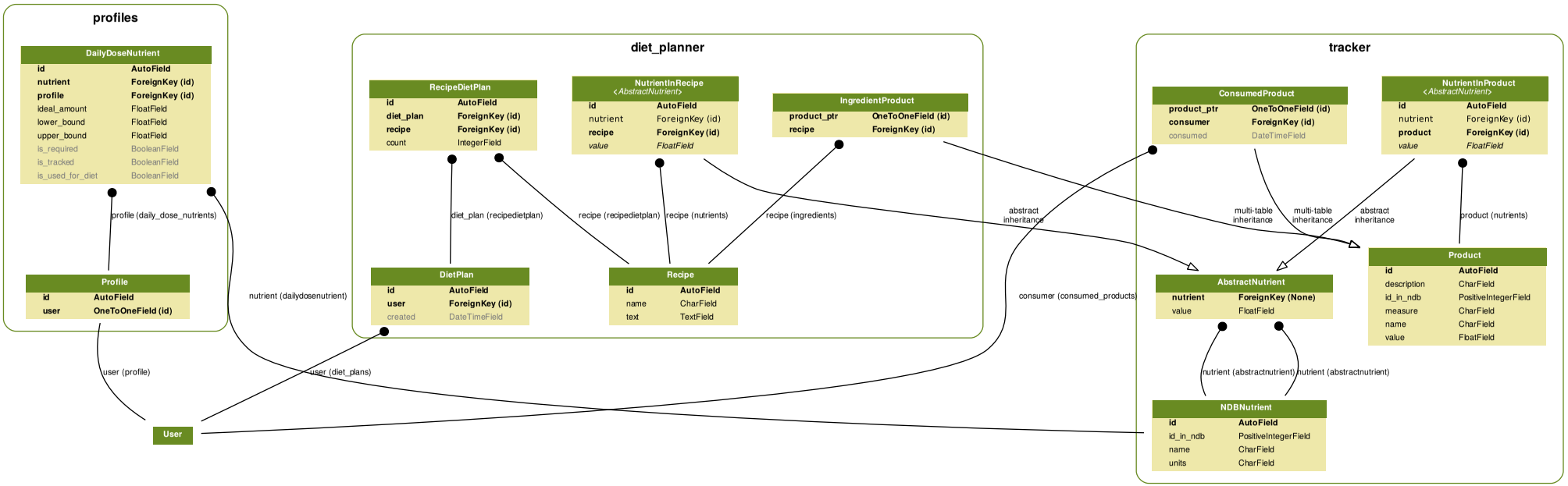
Možnosti rozšíření funkcionality:

- kalkulace denní spotřeby na základě fyzických parametrů uživatele (růst, váha, pohlaví, tělesná aktivita atd.)
- generování jídelníčku s ohledem na diety (veganské, paleo atd.)
- sestavování jídelníčku s určitých potravin, které uživatel má doma
- vytváření nákupního seznamu

Věřím, že moje aplikace pomůže usnadnit a případně prodloužit život pro někteří lidi.

Dodatek A

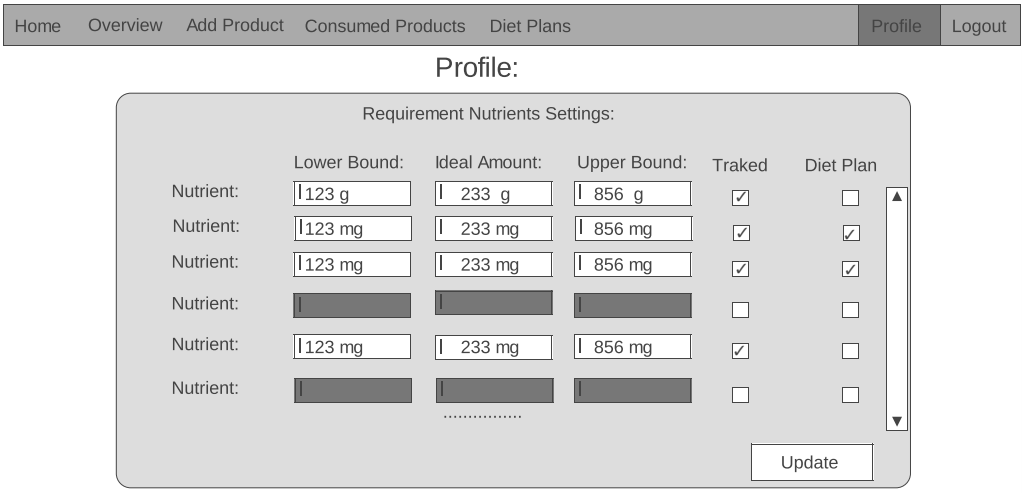
Diagram tříd



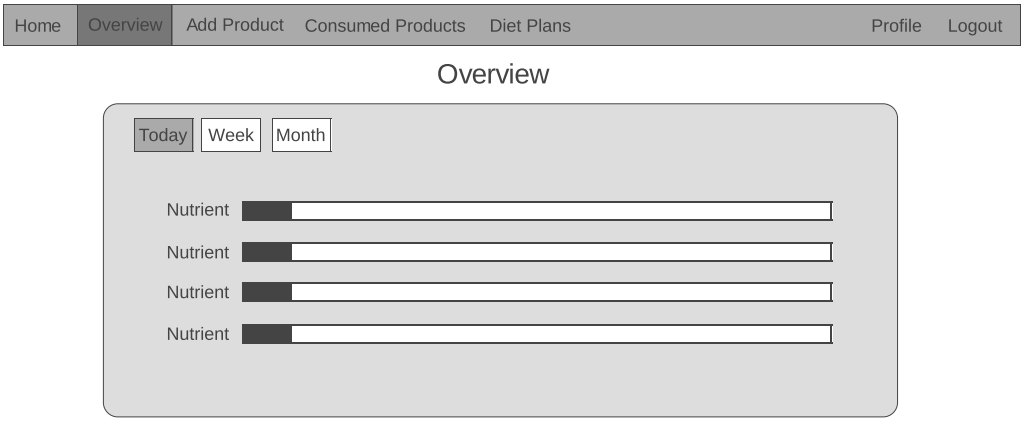
Obrázek A.1: *Diagram tříd seskupený podle django-aplikací*

Dodatek B

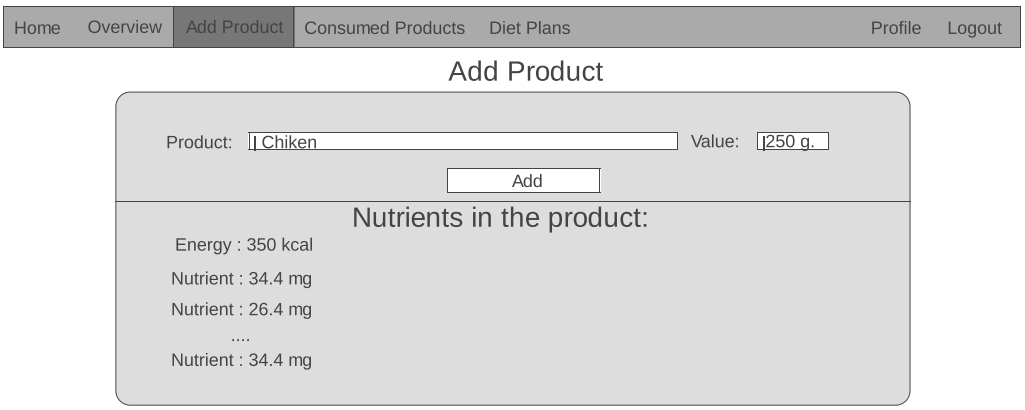
Náhledy webového rozhraní (wireframes)



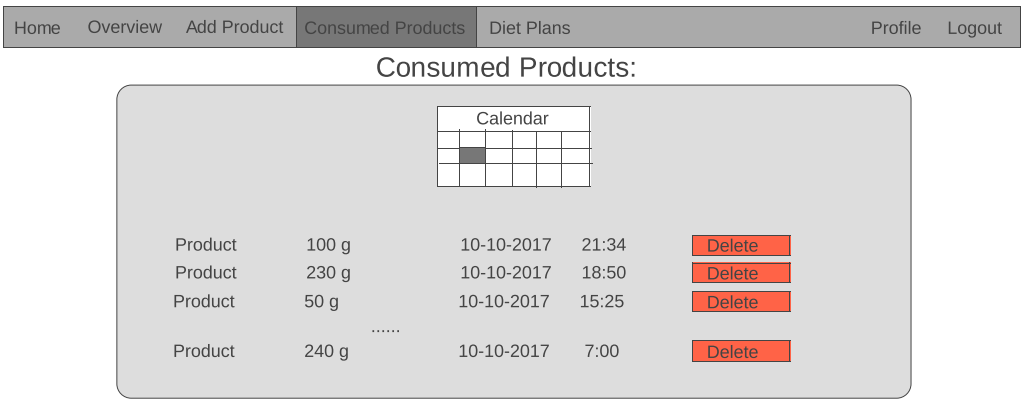
Obrázek B.1: *Wireframe stránky profile*



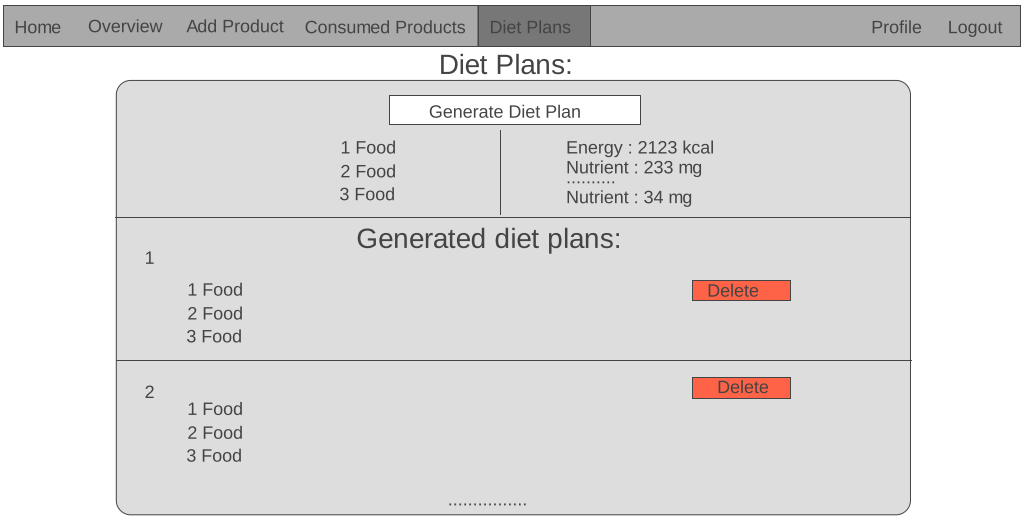
Obrázek B.2: *Wireframe stránky overview*



Obrázek B.3: *Wireframe stránky add-product*



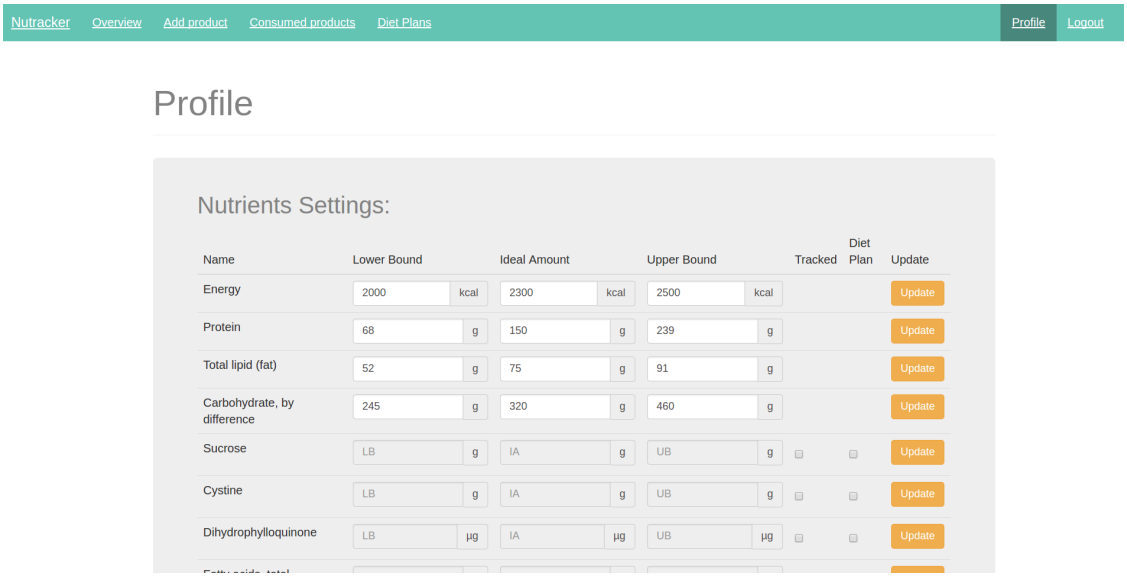
Obrázek B.4: *Wireframe stránky consumed-products*



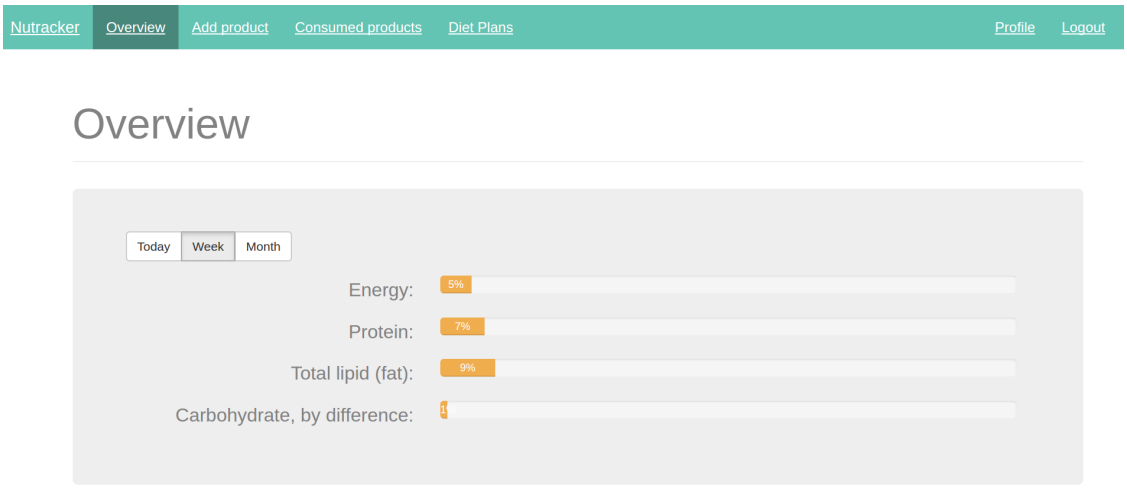
Obrázek B.5: *Wireframe stranky diet-plans*

Dodatek C

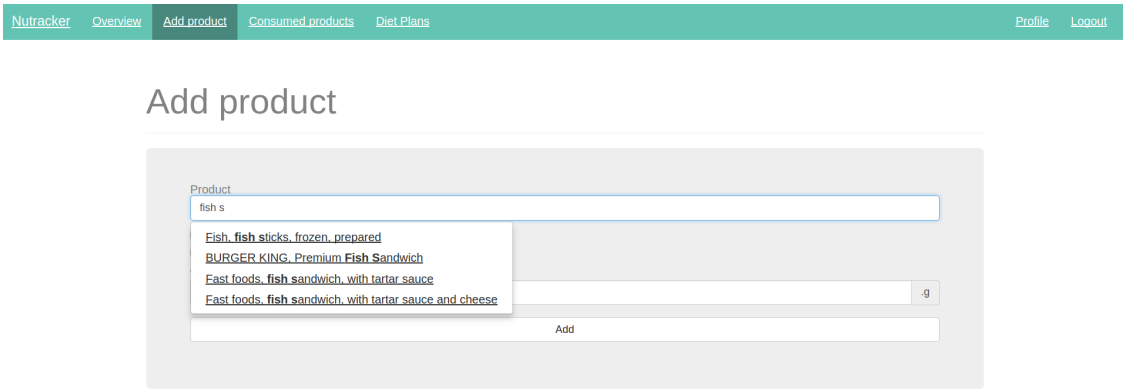
Snímky obrazovky s pohledy aplikace (screenshots)



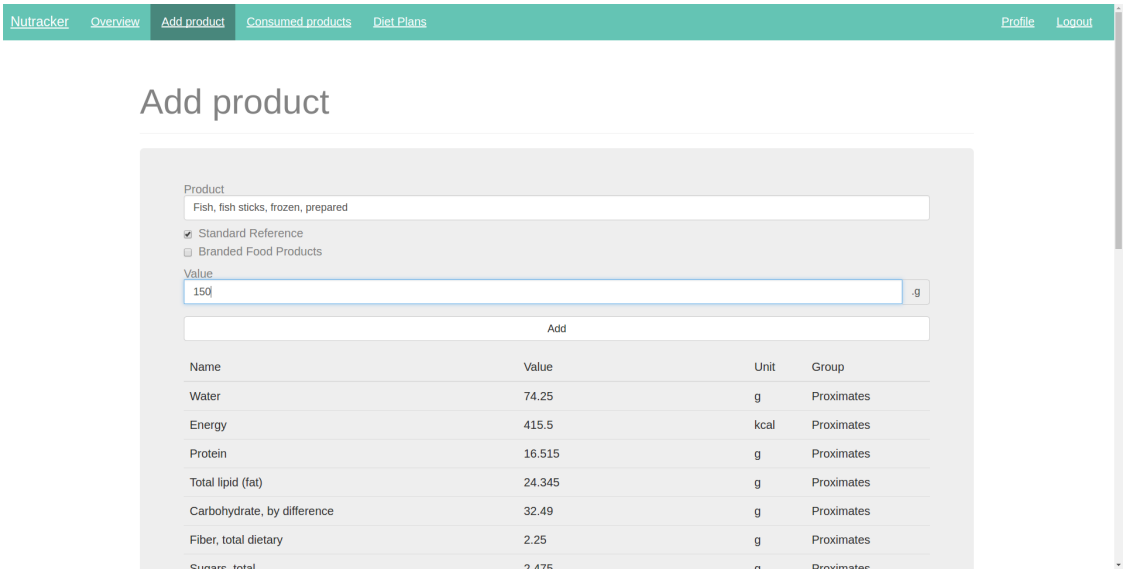
Obrázek C.1: *Snímkek stranky profile*



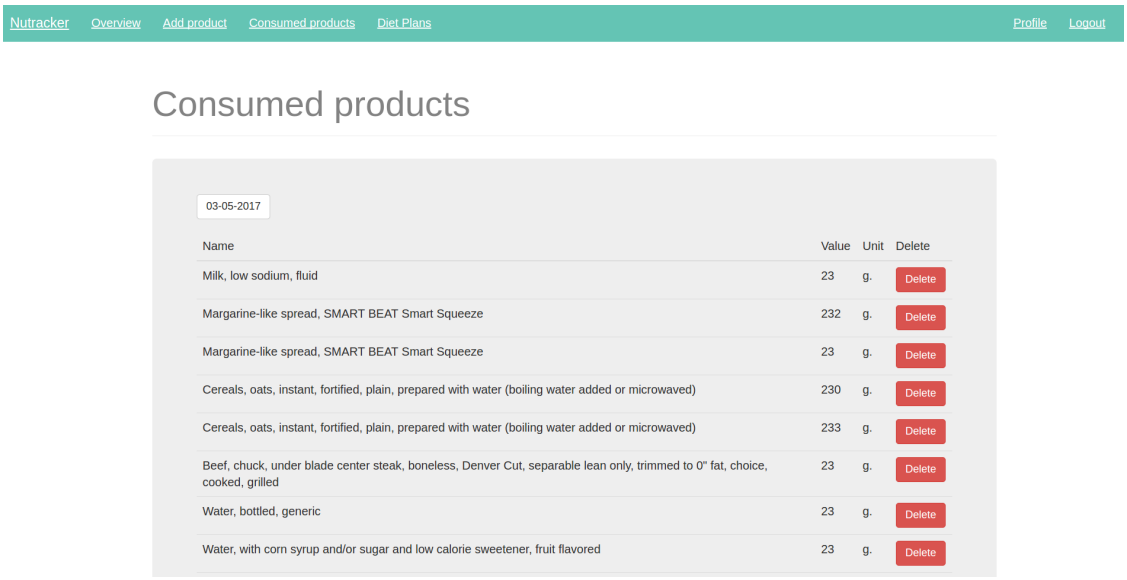
Obrázek C.2: *Snímkek stranky overview*



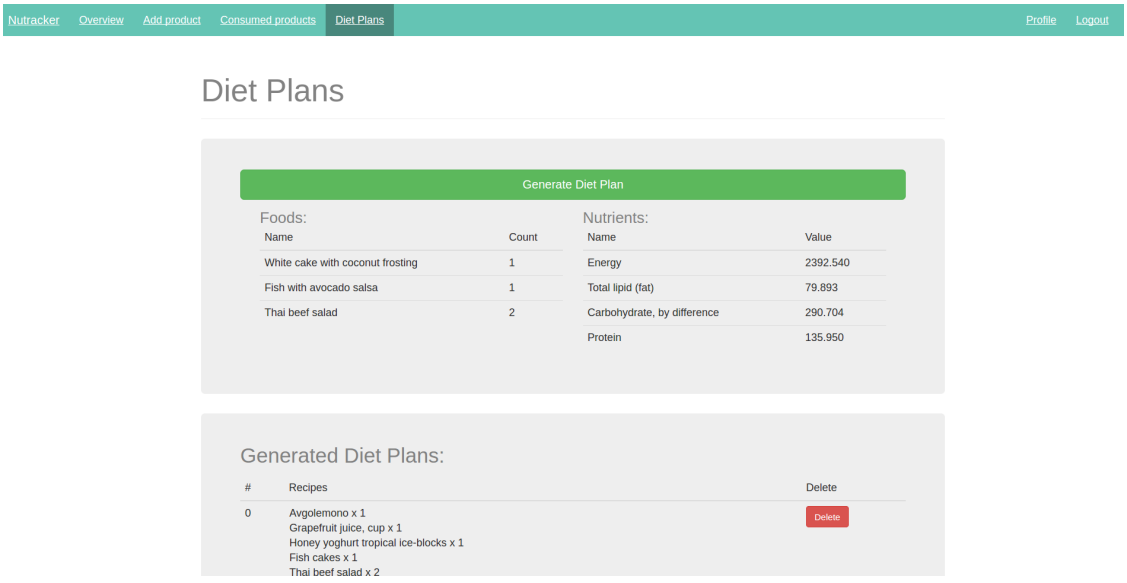
Obrázek C.3: *Snímkek stranky add-product 1*



Obrázek C.4: *Snímkek stranky add-product 2*



Obrázek C.5: *Snímkek stranky consumed-products*



Obrázek C.6: *Snímkek stranky diet-plans*

Dodatek D

Přílohy k testování použitelnosti

D.1 Scénáře

D.1.1 Úvod

Webová aplikace Nutracker slouží pro sestavování jídelníčku na základě předem zvolených živin, ale také usnadňuje zaznamenávání a sledování obsažených v konzumovaných živin potravinách.

Pro následující scénáře předpokládejte, že jste obdrželi od svého dietologa následující denní dávky živin:

Name	Lower Bound	Ideal Amount	Upper Bound
Energy	2000	2300	2500
Protein	70	150	240
Lipid (Fat)	50	75	100
Carbohydrate	245	320	460

D.1.2 Scénář 1

Výchozí stav: Úvodní stránka portálu

Cílový stav: Splnění všech bodů

Úkol 1

Uživatel: Přihlaste se pomocí profilu na Facebook.

Moderator: Uživatel klikne na tlačítko Login. Poté uživatel klikne na tlačítko Přihlásit se pomocí Facebook a uvede své údaje v modálním okénku. Po přihlášení bude přesměrován na stránku Přehled.

Úkol 2

Uživatel: Předpokládejme že jste snědli bramborový salát v hmotnosti 175 gramů a chcete ho přidat do seznamu konzumovaných potravin.

Moderátor: Uživatel klikne na navigační tlačítko Add Product, kde najde záznam o bramborovém salátu, zadá hmotu a zmáčkne tlačítko přidat.

Úkol 3

Uživatel: Předpokládejme že máte jeden chybný záznam o konzumování nápojů "Coca-cola" a chcete ho smazat.

Moderátor: Uživatel klikne na navigační tlačítko Consumed Product, kde najde záznam o nápojů "Coca-cola" zmáčkne tlačítko Delete.

Úkol 4

Uživatel: Teď chcete podívat jak konzumovány salát odpovídá vaší denní dávce.

Moderátor: Uživatel klikne na navigační tlačítko Overview, kde dostane chybovou hlášku o potřebě zadání nastavení nutričních hodnot v profilu. Potom klikne na navigační tlačítko Profil a nastaví nutriční hodnoty podle tabulky.

Úkol 5

Uživatel: Odhlaste se.

Moderátor: Uživatel zmáčkne tlačítko Logout.

D.1.3 Scénář 2

Výchozí stav Úvodní stránka portálu

Cílový stav Splnění všech bodů

Úkol 1

Uživatel: Přihlaste se pomocí profilu na Facebook.

Moderator: Uživatel klikne na tlačítko Login. Poté uživatel klikne na tlačítko Přihlásit se pomocí Facebook a uvede své údaje v modálním okénku. Po přihlášení bude přesměrován na stránku Přehled.

Úkol 2

Uživatel: Najdete informace kolik cukru obshue 100 g. bananu.

Moderátor: Uživatel klikne na navigační tlačítko Add Product, kde najde záznam o bananu salátu, zadá hmotu 100 g. a vyhledá obsah tuku.

Úkol 3

Uživatel: Najdete informace kolik cukru obshue 100 g. bananu.

Moderátor: Uživatel klikne na navigační tlačítko Add Product, kde najde záznam o bananu salátu, zadá hmotu 100 g. a vyhledá obsah tuku.

Úkol 4

Uživatel: Už máte nastavení profilu a chcete si generovat jídelníček.

Moderátor: Uživatel klikne na navigační tlačítko Diet Planš, kde zmáčkne tlačítko Generate Diet Plán a dostane nový jídelníček.

Úkol 5

Uživatel: Odhlaste se.

Moderátor: Uživatel zmáčkne tlačítko Logout.

D.2 Vstupní dotazník

Vyplňte prosím tento dotazník, který pomůže vylepšit výsledky testování:

- pohlaví Muž/Žena
- věk
- víte k čemu je to sledovat denní dávku nutričních hodnot? Ano/Ne
- už jste někdy sledovali nutriční hodnoty? Ano/Ne
- už jste někdy použily aplikace umožňující generovat jídelníček? Ano/Ne

D.3 Vystupní dotazník

Odpovězte nám prosím na několik otázek, jaký máte z testovaní dojem:

- kroky 1. Scénáře přišli vám jasné a bezproblémově?
- pokud ne, co a proč?
- kroky 2. Scénáře přišli vám jasné a bezproblémově?
- pokud ne, co a proč?
- jak jste se orientoval v prostředí?
- poznali jste jak ovládat jednotlivé prvky a kcemu slouží?
- pokud ne, které a proč?
- dobré se vám pracovalo?
- pokud ne, proč?
- napište prosím své pozitivní dojmy:
- napište prosím své negativní dojmy:

Dodatek E

Seznam použitých zkratek

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

API Application Programming Interface

REST Representational state transfer

JSON JavaScript Object Notation

CRUD Create, Read, Update, Delete

HATEAOS Hypermedia as the Engine of Application State

ID3 Iterative Dichotomiser 3 (algorithm)

NDB National Nutrient Database

URI Uniform Resource Identifier

WSGI Web Server Gateway Interface

MVC Model-View-Controller

MTV Model-Template-View

ORM Object-Relational Mapping

SQL Structured Query Language

DRF Django REST framework

CVS Concurrent Version System

CSRF Cross Site Request Forgery

Bibliografie

1. BOHUMIL TUREK Petr Šíma, Irena Michalová. *Jak a proč výživa ovlivňuje zdraví. Zdravotní tvrzení na potravinách*. 2013. ISBN 978-80-905096-8-9. Dostupné také z: http://www.foodnet.cz/soubor.php?id=17428&kontrola=cabb2adcafecd33a5263c7713f3ffbfa. [cit. 2016-12-11].
2. ČR, ÚZIS. *Péče o nemocné cukrovkou 2012*. 2013. ISBN 978-80-7472-082-6.
3. *CRON-O-Meter*. Dostupné také z: https://cronometer.com/.
4. *Automated Menu Planning Algorithm for Children: Food Recommendation by Dietary Management System using ID3 for Indian Food Database*. 2015. Dostupné také z: http://www.sciencedirect.com/science/article/pii/S1877050915005712. [cit.2016-12-11].
5. *Dietary Menu Planning Using an Evolutionary Method*. 2007. Dostupné také z: http://ev.fe.uni-lj.si/5-2007/Korousic.pdf. [cit.2016-12-11].
6. *Computational Nutrition: An Algorithm to Generate a Diet Plan to Meet Specific Nutritional Requirements*. 2016. Dostupné také z: http://file.scirp.org/pdf/ETSN_2016052716163416.pdf. [cit.2016-12-11].
7. *REST principles explained | Servage Magazine*. Dostupné také z: https://www.servage.net/blog/2013/04/08/rest-principles-explained/. [cit.2017-04-20].
8. MALÝ, Martin. *REST: architektura pro webové API - Zdroják*. 2009. Dostupné také z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/. [cit.2016-12-11].

9. *Understanding HATEOAS*. Dostupné také z: https://spring.io/understanding/HATEOAS. [cit.2017-04-20].
10. *USDA National Nutrient Database for Standard Reference, Release 28*. 2016. Dostupné také z: https://www.ars.usda.gov/ARSUserFiles/80400525/Data/SR/SR28/sr28_doc.pdf. [cit.2017-05-01].
11. *The Web framework for perfectionists with deadlines | Django*. Dostupné také z: https://www.djangoproject.com/. [cit.2017-05-01].
12. *Django REST framework*. Dostupné také z: http://www.django-rest-framework.org/.
13. *React - A JavaScript library for building user interfaces*. Dostupné také z: https://facebook.github.io/react/. [cit. 2017-05-13].
14. *JSX In Depth - React*. Dostupné také z: https://facebook.github.io/react/docs/jsx-in-depth.html. [cit. 2017-05-13].
15. *Git*. Dostupné také z: https://git-scm.com/. [cit.2017-05-01].
16. *GitHub*. Dostupné také z: https://github.com/. [cit.2017-05-01].
17. *pip 9.0.1 : Python Package Index*. Dostupné také z: https://pypi.python.org/pypi/pip. [cit.2017-05-01].
18. *npm*. Dostupné také z: https://www.npmjs.com/. [cit.2017-05-01].
19. *How to Extend Django User Model*. 2016. Dostupné také z: https://simpleisbetterthancomplex.com/tutorial/2016/07/22/how-to-extend-django-user-model.html. [cit.2017-05-01].
20. *Signals | Django documentation | Django*. Dostupné také z: https://docs.djangoproject.com/en/1.11/topics/signals/. [cit.2017-05-01].
21. *Recipes, recipes and recipes - Taste*. Dostupné také z: http://www.taste.com.au/. [cit. 2017-05-13].
22. *U.S. Department of Agriculture*. Dostupné také z: http://www.usda.gov/wps/portal/usda/usdahome.
23. OSTATNI, Raymond Penners a. *django-allauth*. Dostupné také z: http://www.intenct.nl/projects/django-allauth/. [cit. 2017-05-13].
24. *Facebook*. Dostupné také z: https://www.facebook.com/. [cit. 2017-05-13].

25. *Facebook for Developers*. Dostupné také z: https://developers.facebook.com/. [cit. 2017-05-13].

26. BRANDL, Georg; SPHINX TEAM, the. *Sphinx documentation contents — Sphinx 1.5.6 documentation*. Dostupné také z: http://www.sphinx-doc.org/en/stable/contents.html. [cit. 2017-05-15].

27. KONSTANTINIDIS, Emmanouil. *DRF Docs*. Dostupné také z: http://drfdocs.com/. [cit. 2017-05-15].

28. *Read Me · Redux*. Dostupné také z: http://redux.js.org/. [cit. 2017-05-14].

29. URALTSEV, Nick. *mzabriskie/axios: Promise based HTTP client for the browser and node.js*. Dostupné také z: https://github.com/mzabriskie/axios. [cit. 2017-05-14].

30. *Bootstrap · The world's most popular mobile-first and responsive front-end framework*. Dostupné také z: http://getbootstrap.com/. [cit. 2017-05-14].

31. *React-Bootstrap*. Dostupné také z: https://react-bootstrap.github.io/. [cit. 2017-05-14].

32. GIOVANOLA, Eric. *React Bootstrap Typeahead Example*. Dostupné také z: http://ericgio.github.io/react-bootstrap-typeahead/. [cit. 2017-05-15].

33. *mock 2.0.0 : Python Package Index*. Dostupné také z: https://pypi.python.org/pypi/mock. [cit. 2017-05-16].

34. *What is Hallway Usability Testing? - Definition from Techopedia*. Dostupné také z: https://www.techopedia.com/definition/30678/hallway-usability-testing. [cit.2017-04-20].

35. *SAGElab | Síťová multimediální laboratoř*. Dostupné také z: https://sagelab.cesnet.cz/. [cit. 2017-05-16].

Dodatek F

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src` ..... zdrojové kódy implementace
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `BP_Bodnar_Bogdan_2017.pdf` ..... text práce ve formátu PDF