VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ APLIKACE PRO TVORBU DIETNÍHO PLÁNU

WEB APPLICATION FOR DIETARY PLAN CREATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VOJTĚCH MATĚJÍČEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19919/2016/xmatej51

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Matějíček Vojtěch**
Obor: Informační technologie
Téma: **Webová aplikace pro tvorbu dietního plánu**
Web Application for Dietary Plan Creation
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby webových aplikací. Dále se seznamte s různými dietami a tvorbou dietních plánů.
2. Analyzujte požadavky na webovou aplikaci, která bude umožňovat regisraci uživatelů a po zadání potřebných tělesných údajů a diety vytvoří dietní plán zahrnující doporučený jídelníček a popř. nákupní seznam. Dále je možné zohlednit tělesnou aktivitu uživatele.
3. Bavrhněte aplikaci splňující výše uvedené požadavky.
4. Navrženou aplikaci implementujte a ověřte její funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Peacock, M., Zemánek, J.: Programujeme vlastní sociální síť v PHP 5. 1. vyd. Brno: Computer Press, 2012, 424 s. ISBN 978-80-251-3626-3
- Welling, L., Thomsonová, L: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této bakalářské práce bylo vytvoření webové aplikace, která uživateli generuje denní jídelníčky na základě měření tělesných úbytků. Tyto pokroky uživatel do aplikace vkládá v pravidelných intervalech. Jídla v těchto jídelníčcích odpovídají dietnímu nebo stravovacímu plánu, který si uživatel volí z nabídky tří implementovaných dietních plánů. Součástí bakalářské práce proto bylo nejen nastudování principů tvorby webových aplikací, ale taktéž seznámení se s teorií implementovaných dietních plánů. Aplikace poskytuje možnost zpětného prohlížení generovaných receptů a tvorby nákupních seznamů. Recepty, jídelníčky i nákupní seznamy je možno vytisknout. Uživatel se do aplikace registruje a přihlašuje pomocí e-mailové adresy a hesla.

Abstract

The aim of this bachelor's thesis was creation of a web application which generates daily daily menus based on measuring body losses. User creates records of his losses at regular intervals. Food in these menus correspond to food or dietary plan that user chooses from three implemented plans. Therefore, getting familiar with principles of creating web applications and studying theory of implemented dietary plans were both part of this thesis. Application provides opportunity to review recipes and creates shopping lists. Recipes, menus and shopping lists can be printed. User registers and logs in the application with his e-mail and password.

Klíčová slova

dieta, jídelníček, zdraví, webová aplikace, databáze, PHP, Laravel, Blade, MVC, Sentinel, HTML, JavaScript, Bootstrap

Keywords

diet, menu, health, web application, database, PHP, Laravel, MVC, Blade, Sentinel, HTML, JavaScript, Bootstrap

Citace

MATĚJÍČEK, Vojtěch. *Webová aplikace pro tvorbu dietního plánu.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bartík Vladimír.

Webová aplikace pro tvorbu dietního plánu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vojtěch Matějíček
18. května 2017

Poděkování

Děkuji Ing. Vladimíru Bartíkovi, Ph.D. za jeho cenné rady a vedení této bakalářské práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Způsob, jakým se stravujeme, po celý náš život výrazně ovlivňuje naše zdraví i postavu. V naprosté většině případů je však naše stravování v lepším případě nepřiměřené, v tom horším pak nezdravé. Díky tomu se v České republice projevují velmi často onemocnění způsobená nesprávnou životosprávou, zejména stravováním. Jedním z příkladů je rakovina tlustého střeva, která v četnosti výskytů v České republice obsazuje jedno z prvních míst[1].

Trendem posledních let se ale stává změna stravovacích návyků, kladení vyššího důrazu na kvalitu potravin a dodržování zásad zdravého stravování.

1.1 Motivace

Hlavní funkcí aplikace je dynamická tvorba a správa jídelníčku pro osobu podstupující jeden z dietních plánů, které aplikace nabízí. Jsou to:

- zdravé stravování,
- proteinová dieta,
- dělená strava.

Důvodem, proč tuto aplikaci používat, je nesnadnost dodržování samotných zásad zdravého či dietního stravování bez striktnějšího dohledu či konkrétních pokynů. Neznalost vhodných jídel nebo správné skladby používaných surovin může vést k vytvoření nesprávných, někdy i nezdravých návyků.

Aplikace generuje explicitní jídelníček, který je uživateli předán k úplnému dodržování. Díky tomu se na uživateli projevují tělesné úbytky váhové i rozměrové. Tyto úbytky uživatel pravidelně zaznamenává a na jejich základě probíhá další generování.

Pro zvýšení motivace je uživateli znázorněno bodové ohodnocení jednotlivých pokroků, které může uživatel směnit za různé odměny v podobě oblíbených jídel nebo nápojů, které nejsou v souladu s dietním plánem nebo návyky správného stravování.

Kapitola 2

Dietní plány pro webovou aplikaci

Kapitola obsahuje základní popis diet a stravovacích plánů, na jejichž základě probíhá výběr jídel při generování jídelníčků webovou aplikací. Uživateli je umožněno volit mezi těmito dietními plány, a to v nastavení uživatele, které je detailněji popsáno v kapitole 4.3.5.

Výběr jídel pro generování jídelníčku podle zvoleného dietního plánu zajišťuje model **FoodPicker** (viz Modely (kapitola 5.1), Modely pro jídelníček (kapitola 5.1.3)), který na základě uživatelova dietního plánu spouští konkrétní algoritmus, jenž zvolí nejvhodnější recept pro dané denní jídlo.

Následují kapitoly věnované detailním popisům diet a stravovacích plánů, které byly zvoleny pro řízení tvorby jídelníčků. Konkrétní typy diet byly zvoleny na základě dotazování uživatelů online dotazníkem. Dalšími dietami navrhovanými při dotazování, ale nebyly implementovány, byly např. *Dieta podle A. Mačingové, různé druhy Krabičkové diety nebo Paleodieta.*

2.1 Zdravé stravování

Tento stravovací plán nepopisuje dietu, ale spíše předepisuje soubor návyků stravovacího stylu. Klade si za cíle vštípit uživateli dodržování správných zásad zdravého stravování a vyhýbání se potravinám, které nejsou vhodné pro lidský organismus a mohou být nahrazeny[22]. Příkladem je záměna pečiva z bílé pšeničné mouky pečivem celozrnným nebo získávání sacharidů jinak než z curku, který je tvořen jednoduchými sacharidy poskytujícími „rychlou“ energii. Ingredience, ze kterých se skládají recepty pro tuto dietu, mají nízký **glykemický index (GI)**[19], což je veličina definující efekt na změny množství sacharidů v lidském těle.

Cílem je taktéž vytvoření návyků správné volby složek obsažených v jídle, které jsou doporučeny ke konzumaci v určitou denní dobu (přijímání méně sacharidů ve večerních hodinách, atd.). Výběr vhodných potravin a určování vhodnosti jídla v denní dobu je zajištěno vytvořenými recepty, které se skládají pouze z kvalitních a zdravých ingrediencí. Denní jídlo *(snídaně, oběd, ...)* je určeno vztahem databázové entity receptu s typem jídla. Proměnným faktorem, závislým na uživateli, je tedy pouze množství jídla a jeho energetická hodnota.

Množství energie, kterou uživatel přijímá z jídla doporučeného aplikací, je počítáno pomocí **bazálního metabolismu (BMR)**[23]. Ten určuje množstí energie, kterou lidské tělo spaluje v klidovém stavu. Na základě tohoto množství jsou uživateli vybírána jídla, která toto množství energie naplňují. Díky tomu uživatel hubne bez pocitu hladu. Toto množství energie je spočítáno podle hodnot, které o sobě uživatel uvádí. Podle pohlaví se vzorce, které jsou pro výpočet použity, liší.

Hodnota uživatelova bazálního metabolismu je přepočítána po každém vkládání pokroků v tělesném úbytku. Díky přepočítání je zpřesněno generování stravovacího plánu, proto se uživateli doporučuje vkládat nové údaje v pravidelném intervalu tří dnů. Tato dieta není časově omezena. Dodržování návyků získaných podstupováním tohoto dietního plánu zlepšuje zdraví a kondici uživatele.

2.2 Proteinová dieta

Proteinová dieta, známá též jako nízkosacharidová dieta nebo ketonová dieta, funguje na principu výrazného omezení množství sacharidů obsaženého v přijímané potravě[25]. Toto množství by nemělo přesahovat 40 – 60 gramů *(doporučené množství se může lišit podle hmotnosti uživatele)*. Zavedení tohoto způsobu stravování způsobuje přechod těla do stavu tzv. **ketózy**[28], kdy tělo přestává brát energii pro své funkce z přijímaných sacharidů, kterých nemá dostatečné množství, ale začíná zpracovávat tělesné zásoby tuku. Pro dosažení tohoto metabolického stavu je právě třeba omezit příjem sacharidů na nutné minimum.

Aby bylo zamezeno ztrátě svalové hmoty, což bývá častým nežádoucím účinkem mnoha diet založených na omezení přísunu kalorií, je omezení příjmu sacharidů kompenzováno vyšším množstvím přijímaných bílkovin. Ty slouží jako stavební materiál pro svalovou hmotu a udržují její množství.

Průběh proteinové diety se dělí na tři fáze, které se liší skladbou jídelníčku. Každá fáze trvá omezenou dobu, v aplikaci je tato doba určena na 3 týdny. Těmito fázemi jsou:

- **Zaváděcí fáze** – Úvodní fáze proteinové diety spočívá v nahrazování všech denních jídel potravinami s vysokým obsahem bílkovin (proteinů) a velmi nízkým obsahem sacharidů. Podstatnou část jídelníčku tak tvoří proteinové nápoje, proteinové polévky a proteinové kaše. První 2 – 4 dny této fáze mohou být pro uživatele podstupujícího tuto dietu náročné z důvodů přechodu na naprosto odlišný stravovací režim, avšak po několika dnech si tělo zvyká. Během cca 3 – 5 dní dodržování stravování podle dietního plánu přechází tělo do stavu **ketózy**, která je důležitou součástí fungování této diety. V této etapě nastává nejrazantnější úbytek tuku.

- **Navazování běžné stravy** – V této fázi je k proteinovým jídlům z první fáze přidáváno jedno denní jídlo, které odpovídá běžnému stravování. Váhový úbytek se zmenšuje, ale nezastavuje se, protože přijímané množství sacharidů je stále nízké a ketóza pokračuje.

- **Návrat k běžné stravě** – Podobně jako druhá fáze je tato fáze založena na připojování běžného jídla k jídelníčku odpovídajícímu proteinové dietě. Běžná jídla jsou v této fázi přidávána dvě. Díky tomu si tělo na návrat k běžným jídlům zvyká pozvolna, čímž je zabráněno tzv. JOJO-efektu, tedy nabrání odbouraných kil zpět. Ketóza stále pokračuje, i když v omezené intenzitě. Kvůli tomu již není váhový a tukový úbytek tak razantní, i když stále probíhá.

Po skončení třetí fáze proteinové diety je uživatelův dietní plán změněn aplikací automaticky na Zdravé stravování. Tento plán lze změnit v nastavení, a to buď zpět na libovolnou fázi PFroteinové diety nebo na Dělenou stravu.

2.3 Dělená strava

Podobně jako Zdravé stravování, ani Dělená strava nepopisuje dietní plán, ale spíše způsoby a návyky ve stravování, které se řídí určitými pravidly. Stejně tak doba dodržování těchto návyků proto není omezená.

Jídelníček je tvořen třemi skupinami jídel[24][27]:

- **Bílkovinná *(proteinová)* strava** – Skupina jídel (v případě této aplikace skupina receptů) s obsahem bílkovin převyšující množství obsažených sacharidů. Jídla jsou kyselého charakteru, mají tedy hodnotu pH nižší než 7.
- **Sacharidová strava** – Skupina jídel s převládající sacharidovou složkou. Tato jídla mají vyšší hodnotu pH, jsou tedy zásaditého charakteru.
- **Neutrální strava** – Sacharidy i bílkoviny jsou ve vyváženém poměru.

Pomocí modelu **FoodPicker** jsou jídla pro denní jídelníčky generována střídavě z každé skupiny (každý den jedna skupina pro všechna jídla).

Rozdělení jídel podle převládající složky je dobré zejména pro zjednodušení trávení potravin, nezpůsobování překyselení žaludku a nepříjemného pocitu pálení žáhy a zpracování většího množství živin a energie, která je méně ukládána v podobě tukových zásob.

Kapitola 3

Analýza požadavků pro aplikaci

Zjišťování uživatelských požadavků a představ o funkcionalitě aplikace proběhl formou online dotazování vzorku asi 130 potenciálních uživatelů. Na základě výsledku dotazování byly realizovány některé funkce navržené tvůrcem aplikace. Od některých funkcí bylo upuštěno a další funkce byly přidány na základě návrhu respondetů.

Na základě ohlasů respondentů byly v aplikaci zvoleny pro realizaci následující funkce:

- ✓ **Automatická tvorba jídelníčku** – Respondentům byla nabídnuta možnost sestavit si jídelníček na základě návrhu, který jim aplikace poskytne. Tato možnost byla 72 % uživatelů zamítnuta a nahrazena funkcí tvorby jídelníčku zcela v režii aplikace.
- ✓ **Zaznamenávání úbytků hmotnosti a rozměrů uživatele** – Nadpoloviční většina dotazovaných (64 % uživatelů) byla ochotna sdělovat aplikaci své pokroky v tělesných úbytcích za účelem dynamické tvorby stravovacího plánu.
- ✓ **Bodování dosažených tělesných úbytků** – V zájmu dosažení vyšší motivace uživatelů k dodržování podstupovaného dietního plánu 67 % respondentů hlasovalo pro zavedení bodovacího systému.

Tyto funkce byly naopak většinou dotazovaných zamítnuty, proto bylo upuštěno od jejich realizace:

- × **Komunikace mezi uživateli** – Protože si každý uživatel registruje vlastní uživatelský účet, sdílí aplikace některé vlastnosti s jinými známými sociálními sítěmi. Většina uživatelů (77 % dotazovaných) ale možnost komunikace s jinými uživateli od aplikace nevyžadovala.
- × **Sdílení svých pokroků** – Podobně jako zasílání zpráv mezi uživateli, nebylo 89 % dotázaných ochotno ani sdílet své pokroky s jinými uživateli. Od funkce interakce mezi uživateli bylo proto kompletně ustoupeno.
- × **Vkládání tělesných složek** – Protože většina dotazovaných (67 %) nevlastní diagnostický přístroj pro měření procentuálního zastoupení jednotlivých složek těla *(tuk, svalová hmota, voda, kosti)*, nebylo sledování tohoto složení zařazeno mezi funkce aplikace.

Následují funkce, které byly navrženy některými respondenty a vyhodnoceny jako přínosné pro aplikaci:

+ **Profilový obrázek** – I když byly zavrženy veškeré funkce pro interakci mezi uživateli, zůstal přesto požadavek pro vložení uživatelského obrázku, a tím personalizaci části vzhledu aplikace. Do aplikace je možné vkládat fotku typu *JPEG* o maximální velikosti 512 kB.

+ **Odměny za body** – Aby bodování uživatelských pokroků nebylo pouze zdánlivě abstraktní informací (mající vliv pouze na chování aplikace, které ale uživatel nepozoruje), byl na základě několika návrhů respondentů zřízen odměnový systém formou „e-shopu“, kde uživatel smění získané body za odměny.

Jednotlivé naimplementované funkce aplikace jsou vyobrazeny na modelu případů užití v příloze B.

O tvorbě jednotlivých funkcí a jejich realizaci pomocí programové abstrakce pojednávají následující kapitoly.

Kapitola 4

Návrh webové aplikace

Tato kapitola popisuje proces vytváření webové aplikace, použité postupy a prostředky nutné k jejímu vývoji a testování.

4.1 Struktura aplikace

Tato kapitola pojednává o nástrojích, které byly použity pro tvorbu webové aplikace. Následuje bližší seznámení s použitým vývojovým prostředím a frameworkem, ve kterém byla aplikace napsána.

4.1.1 Vývojové prostředí a jazyk PHP

Jako jazyk, ve kterém bude aplikace napsána, byl zvolen skriptovací jazyk **PHP** v kombinaci se značkovacím jazykem **HTML**, který předepisuje vzhled webové aplikace. Jazyk PHP byl vybrán pro svou snadnost a přehlednost použití a množství funkcí a nástaveb (umožňujících např. snadné napojení relační databáze) a kvalitní a přehlednou dokumentaci. Aplikace je programována v jazyce PHP verze 7.

Jazyk PHP je dynamicky typovaný skriptovací jazyk[21]. Je nejčastěji používaným jazykem pro vykonávání funkcí webových aplikací *(v dubnu 2017 používalo 82,6 % webových stránek PHP[20])* na straně poskytovatelského serveru. Tento jazyk je multiplatformní a lze jej s aplikováním pouze minimálních změn snadno přenášet mezi různými operačními systémy. Jeho velkou výhodou je možnost vytváření tzv. asociativních polí[17], která jsou mnohokrát použita frameworkem Laravel popsaným kapitole 4.1.2.

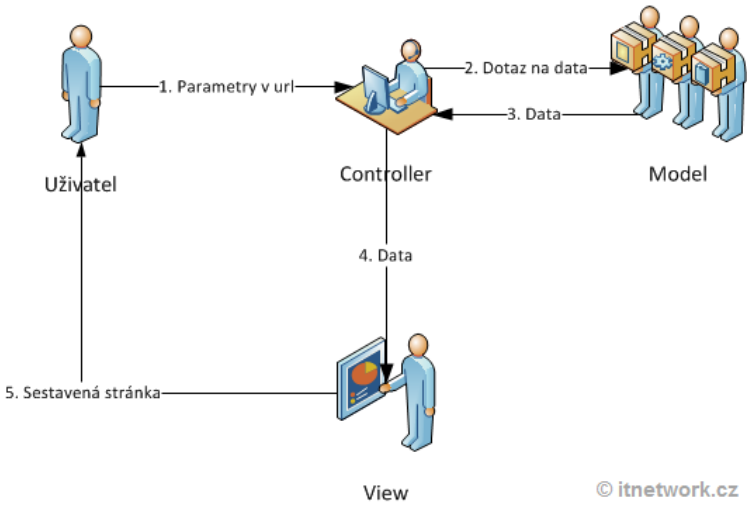
Dalšími jazyky používanými pro programování webových aplikací jsou např. **C#.NET, Java** nebo **Python**.

Pro usnadění vývoje webové aplikace bylo třeba zvolit vhodné vývojové prostředí. Ze široké nabídky dostupných editorů byl zvolen editor **PHP Storm** od firmy **JetBrains**. Tento editor je firmou JetBrains poskytován se studentskou licencí a je optimalizován pro použití jazyka PHP spolu s frameworky založenými na návrhovém vzoru MVC, který byl pro tvorbu aplikace použit.

4.1.2 Framework Laravel

Pro vytváření celé aplikace byl použit rámec (framework) **Laravel** ve verzi 5.3. Tento rámec fungující na architektuře návrhového vzoru MVC *(Model, View, Controller)* poskytuje

kromě vytvoření a napojení relační databáze k aplikaci také oddělení front-endové (vizuální) části od back-endové části (programové logiky). Framework Laravel pracuje s konzolovým programem **Artisan**, který programátorovi poskytuje řadu užitečných funkcí, od běhu lokálního serveru až po práci s databází. O vytvoření projektu v tomto frameworku se stará program **Composer**, který zajišťuje vytvoření všech závislostí jednotlivých souborů v celém projektu.



Obrázek 4.1: Schéma návrhového vzoru MVC [30]

Návrhový vzor MVC, na jehož základě je postaven i framework Laravel, dělí strukturu aplikace na

- **modely** – třídy reprezentující jednotlivé programové entity *(např. User, Recipe)*,
- **pohledy** *(Views)* – soubory obsahující HTML kódy jednotlivých stránek a bloky PHP kódu pro zobrazení dynamického obsahu *(např. denní jídelníček)*,
- **ovladače** *(Controllers)* – třídy sloužící pro propojení pohledů s jednotlivými modely a zajištění interakce mezi uživatelem a aplikací *(např. zpracování dat z registračního formuláře)*.

Tímto rozdělením je zajištěna vyšší úroveň čistoty kódu a umožněna snadnější udržovatelnost aplikace. Ve větších projektech je použitím podobného frameworku umožněna také nezávislá týmová spolupráce. Schéma návrhového vzoru je i se znázorněním interakcí jednotlivých prvků zobrazeno na obrázku 4.1.

O navigaci webovou aplikací se stará *router*[14] definovaný v souboru `web.php`. Tento soubor obsahuje všechny existující odkazy aplikace, na jejichž základě volá předepsanou funkci definovanou v ovladači. Router přesměrovává jak adresy navštívené pomocí odkazu, tak i adresy definované atributem *action* ve formuláři.

V routeru lze rovněž definovat přístupnost daných adres jednotlivým kategoriím uživatelů pomocí třídy **middleware**[13]. Tato třída definuje omezení přístupu a chování při jeho zamítnutí (obvykle přesměrování na přístupnou stránku).

4.2 Práce s databází

Webová aplikace pro tvorbu dietních plánů umožňuje uživateli registraci vlastního uživatelského účtu. Pro tento účel je nutné nejen přehledné uživatelské rozhraní, ale i prostor pro ukládání uživatelských dat. K tomu bylo zvoleno použití relační databáze. Jednou z důležitých funkcí frameworku Laravel je napojení aplikace na tuto databázi. K nastavení připojení slouží konfigurační soubor `.env` obsahující údaje nutné pro připojení k databázi a čtení a zápisu dat.

4.2.1 Definice databázového schématu

K vytvoření jednotlivých tabulek databáze slouží migrace[7]. Migrační soubory obsahují schémata jednotlivých tabulek, datových typů jejich sloupců, základních integritních omezení a výchozích hodnot. Migrace je provedena programem Artisan příkazem

`php artisan migrate`.

Pokud došlo k úpravám schémat v již použitém migračním souboru, je třeba pro znovuprovedení migrace přidat atribut `refresh`, který ale vrátí změny provedené všemi migracemi, a tím resetuje a vyprázdní celou databázi.

K uchování dat a jejich opětovnému uložení do databáze slouží **sázecí třídy** *(seeder)*[9]. Lze je vytvořit ručně s libovolnými (nebo imaginárními) daty nebo pomocí nástroje `iseed`[26], doplněného do programu Artisan, vytvořit otisk databáze do sázecí třídy. Uložení obsahu souboru do databáze je pak provedeno při migraci spuštěné s přepínačem `--seed`.

Výsledná podoba databáze aplikace, jejích obsažených entit a vztahů mezi nimi je znázorněna na Entity-Relationship diagramu v příloze A.

4.2.2 Definování vztahů mezi entitami

Definování vztahů mezi jednotlivými entitami je nutno provést vytvořením integritních omezení již při tvorbě databáze. V migračním souboru definujeme vytvoření položek cizích klíčů odkazující na primární klíče jiných tabulek voláním metody `foreign()`. Tímto jsou definována integritní omezení na databázové úrovni. Diagram vztahů mezi jednotlivými databázovými entitami je součástí přílohy A.

Pro přístup k entitám z jiných vázaných entit je nutno tato integritní omezení definovat také na programové úrovni, tedy pro každý model definovat vztahy s jinými modely[11]. K tomu slouží metody předdefinované v samotném frameworku. Pro definování vztahu *One to One* a *One to Many* slouží metoda `hasOne()`, resp. `hasMany()` pro vlastníka a `belongsTo()` pro vlastněnou entitu.

Definování vztahu *Many to Many* je realizováno metodou `belongsToMany()` pro vlastníka i vlastněnou entitu. K tomu je třeba již nadefinovaná vazební tabulka obsahující cizí klíče odkazující na obě entity, se kterými sdílí vztah *One to Many*.

Pro nejsnadnější vytvoření těchto definic je nutné dodržovat konvence v pojmenování tabulek v databázi podle názvů jednotlivých entit:

- Název tabulky je množným číslem názvu modelu a je psán malými písmeny (Pro model `User` existuje tabulka `users`.).

- Položka primárního klíče tabulky má název `id`,
- Cizí klíč se skládá z názvu entity doplněného o koncovku `_id`. Je psán malými písmeny (V tabulce `users` se nachází cizí klíč `diet_id`.).
- Vazební tabulka sloužící pro realizování vztahu *Many to Many* je pojmenována názvy obou entit v abecedním pořadí oddělenými podtržítkem (např. `ingredient_recipe`).
- Název metody pro definování vztahu mezi entitami je shodný s názvem vztahující se entity. Je psán malými písmeny. U vztahů *One to Many* a *Many to Many* používáme množné číslo (např. `public function ingredients()` v modelu `Recipes`).

Při dodržení těchto konvencí je možno vztahy definovat pouze vložením názvu modelových tříd v adresáři `App\` jako jediného argumentu funkcím `has*()` a `belongsTo*()`. Při jejich nedodržení je třeba jako další argumenty přidat i primární a cizí klíče obsažené v dotčených tabulkách.

4.2.3 Databázové dotazy

Pro tvorbu databázových dotazů nabízí framework Laravel dva nástroje: *QueryBuilder*[8] a *Eloquent*[10], který je aktivní nástavbou QueryBuilderu. Oba tyto nástroje se tak v mnohém podobají, včetně syntaxe. Jejich rozdíl spočívá především ve způsobu přístupu do databáze, a tím rychlosti zpracování dotazu. Zatímco QueryBuilder přistupuje v databázi přímo do tabulky zadané uživatelem, Eloquent pracuje s modely. Díky tomu je provedení dotazu QueryBuilderem rychlejší, jeho použití je však výhodné až při práci s větším objemem dat.

Výhoda Eloquentu je v jeho srozumitelnosti a jednoduchosti použití. Jak již bylo zmíněno, Eloquent pracuje s jednotlivými modely, které jsou napojeny na své korespondující tabulky v databázi. Umožňuje vytvoření, aktualizaci i smazání záznamu v databázi odpovídajícího konkrétní instanci daného modelu.

Dotazy `SELECT` jsou generovány automaticky pomocí QueryBuilderu. Ten poskytuje široké spektrum funkcí nahrazujících různé SQL výrazy, např. `WHERE` (metoda `where()`), `HAVING` (metoda `whereHas()`) nebo `IN` (metoda `whereIn()`). Návratovou hodnotou těchto funkcí je první nalezený odpovídající výskyt v databázové tabulce (volaný metodou `first()`) nebo kolekce entit odpovídajících danému dotazu (metoda `get()`). Touto kolekcí lze procházet cyklicky (např. pomocí cyklu `foreach`).

Obdobným způsobem lze pomocí Eloquentu získat kolekci entit modelu, které jsou s dotazovaným modelem vázány předem nadefinovaným vztahem. Např. kolekci všech uživatelů podstupující danou dietu vrátí volání property `users` v modelu *Diet*, kde mezi modely *Diet* a *User* je definován vztah *One to Many*.

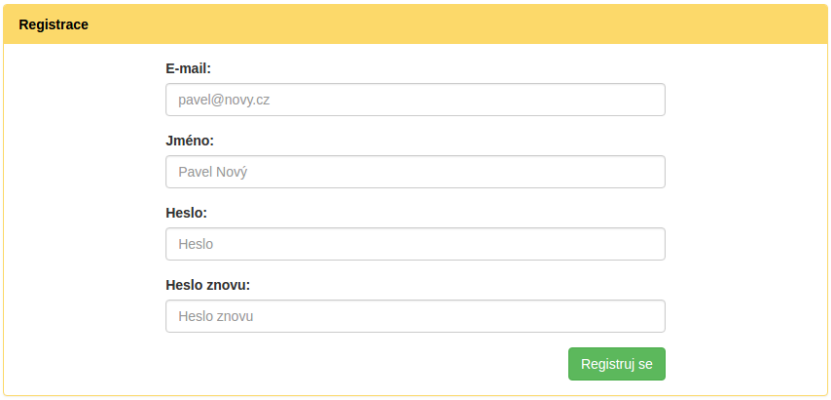
4.3 Autentizace uživatele

Framework Laravel poskytuje balíček funkcí `Auth`[3] pro vytvoření základního systému umožňujícího autentizaci uživatele a přístup k jeho účtu. Počet nástrojů je ale omezený, proto tento balíček nebyl použit.

Místo něj byl zvolen framework **Sentinel**[15], který množství dostupných nástrojů značně rozšiřuje a je optimalizován pro použití s frameworkem Laravel.

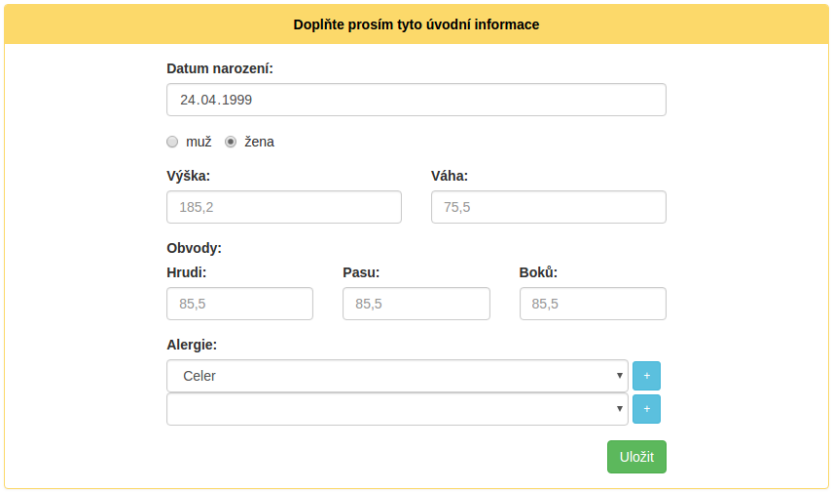
4.3.1 Registrace

Registrace je realizována pomocí jednoduchého formuláře obsahujícího pole pro e-mailovou adresu, jméno a dvě pole pro vložení a potvrzení hesla. Náhled registračního formuláře je vyobrazen na obr. 4.2. Po zadání všech těchto údajů jsou tyto údaje předány ovladači `AuthenticationController`, který se stará o jejich zpracování. Údaje jsou poté zkontrolovány validátorem, který je v Laravelu obsažen. V případě úspěchu je použit balíček Sentinel, který vyhledá v databázi případné existující registrace se stejnou adresou pro případ kolize. Pokud žádné nenalezne, je vytvořen záznam v tabulce `users`. Účet je vytvořen, ale nelze se na něj prozatím přihlásit.



Obrázek 4.2: Registrační formulář

Dále byl vytvořen záznam v tabulce `activations`, kde se uchovávají údaje o aktivaci uživatelského účtu. Tuto aktivaci lze provést pomocí odkazu odeslaného uživateli na e-mail, který zadal. Po aktivaci je uživatel přihlášen a vyzván k vložení základních údajů o svém těle *(výšku, váhu a míry)*, alergiích na potraviny a svém pohlaví do formuláře na obr. 4.3. Tyto údaje jsou použity pro první generování jídelníčku. Výchozí dietou pro nově registrovaného uživatele je *Zdravé stravování*.



Obrázek 4.3: Vkládání úvodních informací o uživateli

4.3.2 Přihlášení uživatele

Přihlášení registrovaného uživatele na jeho účet je zhotoveno jednoduchým formulářem obsahujícím pole pro e-mailovou adresu a heslo. Jeho podoba je zobrazena obrázkem 4.4.

Tyto údaje jsou odeslány ovladači `AuthenticationController`, který zkontroluje, zda se v databázi nachází uživatel s těmito přihlašovacími údaji a jestli je jeho účet aktivován. V případě jakékoli chyby je uživatel přesměrován zpět na přihlašovací formulář doplněný o chybovou hlášku.

Přihlašovací formulář také obsahuje tlačítko pro zapamatování uživatele pomocí souboru Cookies. Uložení souboru Cookies provádí funkce `authenticateAndRemember()` z balíčku Sentinel po úspěšném příhlášení uživatele do aplikace.

Obrázek 4.4: Přihlašovací formulář

Posledním prvkem přihlašovacího formuláře je odkaz *"Zapomněl(a) jsem heslo"* sloužící pro resetování hesla k uživatelskému účtu. Kliknutí na tento odkaz přesměruje uživatele k novému formuláři, kde je po něm požadována e-mailová adresa, pod kterou se do aplikace registroval.

Tato adresa je ke kontrole předána ovladači `AuthenticationController`, který opět zkontroluje její výskyt v databázi. V případě úspěšného i neúspěšného vyhledávání je uživateli vrácena hláška o úspěchu. Tím je uživateli utajeno, že se případná neznámá adresa v databázi nevyskytuje.

V případě úspěchu je na uživatelův e-mail odeslána zpráva s odkazem vedoucím k formuláři na vytvoření a potvrzení nového hesla. O propojení uživatelského účtu a nového hesla se stará entita `Reminder`, obsažená v balíčku Sentinel. Ta obsahuje unikátní kód, který nahrazuje autentizaci uživatele při tvorbě nového hesla. Díky tomu lze obnovení hesla provést se stejným odkazem pouze jednou.

Po vytvoření nového hesla je uživatel přesměrován zpět k přihlašovacímu formuláři, kde je vyzván k přihlášení s novým heslem.

4.3.3 Role uživatelů

Každému uživatelskému účtu je při jeho vytvoření přiřazena role. Ta určuje míru oprávnění uživatele v aplikaci a určení zobrazeného obsahu. V aplikaci pro tvorbu dietního plánu jsou použity pouze dvě role:

- **Admin** – administrátor aplikace, zajišťuje správu receptů, ingrediencí a diet. Generování jídelníčku pro tohoto uživatele neprobíhá.
- **Basic** – základní uživatelská role, pro tohoto uživatele jsou generovány jídelníčky na základě vkládaných pokroků v tělesném úbytku.

Pro každou z těchto rolí existuje skupina směrovacích pravidel v souboru `web.php` za pomoci vlastního middlewaru.

4.3.4 Odhlášení uživatele

Pro odhlášení z uživatelského účtu slouží odkaz *"Odhlásit se"* v horním pravém rohu aplikace. Odkaz nezpůsobuje přesměrování, ale odeslání formuláře pro odhlášení. Touto technikou je zajištěno, že k odhlášení dojde pouze kliknutím na odkaz, a nikoli napsáním `"/logout"` na konec adresy aplikace v poli prohlížeče – v takovém případě by router způsobil výjimku. Po odhlášení je uživatel přesměrován na úvodní stranu aplikace.

4.3.5 Nastavení uživatelského účtu

Stránka pro nastavení uživateslkého účtu je řešena jako skupina formulářů pro jednotlivá nastavení. Obsluha jednotlivých nastavení je implementována příslušnými metodami ovladače `SettingsController`. Uživateli je pomocí nich umožněno upravit následující atributy svého účtu:

- **Profilový obrázek**: K uložení nahraného profilového obrázku je použito veřejné úložiště přístupné ve frameworku Laravel ve složce `public/avatars`. Aplikace uchovává pouze jeden profilový obrázek. Nahrání nového obrázku způsobí přepsání aktuálního. Smazání uživatelského profilového obrázku způsobí použití výchozího obrázku ve tvaru ikony uživatele.
- **Heslo**: Změna hesla je podmíněna zadáním aktuálně platného hesla. Jeho platnost kontroluje balík Sentinel funkcí `validateCredentials`. Pro uložení nového hesla je poté třeba vložit toto heslo dvakrát pro potvrzení. Obě zadání jsou kontrolována validátorem pro svou shodnost a předepsané rozpětí. Po úspěšné validaci je aktualizován šifrovaný otisk hesla v tabulce `users` v databázi.
- **Aktuální dieta**: Změna diety je realizována výběrem jedné z nabízených diet ve formuláři se vstupem typu *select*. Výchozí dietou po registraci uživatele je *Zdravé stravování*. V nastavení je možné tuto dietu změnit. U diet s předepsanou délkou trvání je dieta aktualizována automaticky po vypršení této doby, nebo je možné tuto dietu změnit ručně.
- **Alergie**: Nastavení a změna alergií probíhá podobně jako nastavení diety pomocí formuláře se vstupem typu *select*. Protože lze zadat 0 až 14 alergií (jedná se o vztah *Many to Many*), lze přidat až 13 dodatečných polí select pro vložení další alergie.
- **Kompletní vymazání účtu**: Smazání uživatelského účtu je podmíněno zadáním hesla. Po jeho validaci balíkem Sentinel jsou odstraněny veškeré uživatelské pokroky, týdenní a denní jídelníčky a vazby mezi nimi a historie zobrazených a vyřazených receptů. Poté je odstraněn záznam o uživateli v tabulkách `users` a `activations`. Tímto je uživatelský účet trvale odstraněn.

Kapitola 5

Návrh entit pro webovou aplikaci

V této kapitole bude blíže rozebrána abstrakce webové aplikace, její rozložení na logické celky zastoupené jednotlivými entitami a vytváření spojitostí mezi těmito entitami.

5.1 Modely *(Models)*

Pro zápis a čtení z databáze pomocí *Eloquentu* je třeba pro každou databázovou entitu vytvořit korespondující model. Pro usnadnění práce s modely a databází byla dodržena konvence pojmenování entit definovaná v kapitole 4.2.2. Díky propojení databázových tabulek s modely lze přistupovat k jednotlivým sloupcům tabulky, vytvářet nové záznamy v tabulkách a upravovat či mazat existující záznamy. Vytváření nových modelů umožňuje konzolový program **Artisan** pomocí příkazu

`php artisan make:model` *NazevModelu*.

Modely jsou pro zvýšení přehlednosti rozděleny podle svého významu a příslušnosti na následující tři celky.

5.1.1 Modely pro uživatele

Tato skupina modelů slouží pro vytváření a uchovávání údajů o uživatelském účtu a vývoji uživatelských pokroků za dobu jeho využívání této aplikace. Těmito modely jsou:

- **User** – základní model zastupující entitu samotného uživatele. Jeho korespondující databázová tabulka `users` obsahuje údaje o uživateli, které po celou dobu jeho používání aplikace zůstanou nezměněny *(datum narození, výška, pohlaví, jméno, atd.)*. Kromě těchto atributů obsahuje také údaje zpracovávané frameworkem Sentinel sloužící pro autentizaci uživatele do systému aplikace.

- **Diet** – tento model zastupuje dietní plán, který si uživatel zvolí pro generování denních jídel. Při registraci je jako výchozí dieta přiděleno uživateli *Zdravé stravování*. Tuto dietu je poté možno změnit v uživatelském nastavení (viz 4.3.5). K dietě jsou vázány jednotlivé recepty vztahem *Many to Many*, tedy jeden recept může být použit ve více dietních plánech.
  Korespondující tabulka `diets` obsahuje údaje o délce trvání diety. Doba, po kterou uživatel dietní plán dodržuje, může být omezená nebo neomezená. V případě omezené doby trvání diety je zaznamenán údaj o délce tohoto trvání v týdnech. Pro

neomezenou dobu je namísto této doby uchována nulová hodnota. Po uplynutí doby dodržování dietního plánu (u časově omezených diet) dochází k automatické změně dietního plánu.

- **Progression** – model Progression umožňuje sledování uživatelových pokroků v tělesné hmotnosti a mírách. Zaznamenávají se míry přes hrudník, pas a boky. Tyto pokroky jsou po svém zaznamenání zpracovány a obodovány. Výsledné body každého pokroku jsou vypočítány podle vzorce

$$\sum_{k=1}^{n} r1[partie] * \frac{r2[partie]}{pocet\_zadanych\_mereni} \tag{5.1}$$

  kde platí, že

  - $n = pocet\_merenych\_partii$,
  - $r1[partie] = -(toto\_mereni[partie] - predchozi\_mereni[partie])$,
  - $r2[partie] = -(toto\_mereni[partie] - prvni\_mereni[partie])$.

  K výslednému obodování pokroku se taktéž přičítají bonusové body za provedenou fyzickou aktivitu v době mezi novým a předchozím měřením tělesných úbytků.

- **Physical** – model pro zaznamenání fyzické aktivity uživatele. Zaznamenávají se aktivity *(jízda na kole, posilování, běh, apod.)* měřené v jednotkách podle svého druhu. Fyzická aktivita je ohodnocena body s různými koeficienty podle náročnosti jednotlivých aktivit.
  K připočítání fyzické aktivity nedochází okamžitě po jeho zadání. Místo toho se záznamy uchovávají v databázi spolu s časovým razítkem svého vytvoření. K bodování aktivity dochází při vkládání měření tělesných úbytků, kdy jsou body přičteny k bodům za příslušné měření. Takto započítávány jsou všechny fyzické aktivity zapsané uživatelem do systému v období mezi dvěma posledními měřeními.

- **Reward** – posledním ze skupiny modelů pro uživatele je model Reward. Tento model slouží pro ztvárnění odměn pro uživatele za své nasbírané body. Tyto body může uživatel směnit za odměnu v podobě „prohřešku“ proti svému dietnímu plánu. Tento prohřešek v podobě např. *kávy, čokolády* nebo *alkoholického nápoje* je uživateli připsán po dobu 3 dnů, po jejichž uplynutí je vazba mezi uživatelem a odměnou zrušena. Ke zrušení této vazby může dojít i označením odměny jako „vyčerpané“ uživatelem na hlavní straně aplikace. Odměny lze směnit za body vícekrát, libovolně do vyčerpání zbývajících bodů.

5.1.2 Modely pro recepty

Tato skupina modelů slouží pro vytváření a spravování jednotlivých receptů, které jsou použity pro tvorbu jídelníčků podle dietního plánu. Každý recept je tvořen skupinou následujících modelů:

- **Recipe** – výchozí model pro recept. Tento model obsahuje jako atributy kolekce ingrediencí, typů jídel *(snídaně, oběd, atd.)* a diet, ke kterým přísluší. Dalším atributem je popis postupu pro přípravu receptu. Tento model je napojen na databázovou tabulku `recipes`, kde jsou kromě postupu uchovávány i informace o energetické hodnotě

receptu a jeho obsahu tuků, bílkovin a sacharidů. Tyto údaje jsou přepočítávány po přidání či odebrání vazeb na ingredience nebo úpravě jejich potřebného množství. Poslední položkou v databázi je uživatelské hodnocení receptu, kde se uchovává průměr všech hodnocení receptu provedeného jednotlivými uživateli.
Jednotlivá hodnocení se uchovávají ve vazební tabulce pro vazbu mezi uživatelem a receptem. Tato vazba slouží rovněž jako historie receptů dříve generovaných uživateli pro jejich pozdější zobrazení. Ve vazební tabulce se dále nachází údaj o vyřazení receptu z uživatelova generování – touto cestou může uživatel označit recepty, se kterými nebyl z různých důvodů spokojen, a tím znemožnit jejich znovupoužití při generování jídelníčku. Tato akce je poté snadno vratná v detailu receptu.

- **Ingredient** – jednotlivé ingredience, které jsou potřeba pro přípravu jídla podle receptu. Každá ingredience obsahuje údaje o své energetické hodnotě a poměru tuků, bílkovin a sacharidů v ní obsažených a údaj o výchozí měrné jednotce *(g, ml, ks)*. Tento údaj je vložen při vytváření nového záznamu o ingredienci. Je-li výchozí jednotka *gram* nebo *mililitr*, je uváděna energetická hodnota na 100 g (ml). V případě, že měrná jednotka je *kus*, je třeba doplnit referenční hmotnost (objem) jednoho kusu ingredience, ke kterému bude energetická hodnota uváděna.
  S modelem **Recipe** sdílí ingredience vztah *Many to Many*, tedy jedna ingredience může být použita v různých receptech. Tento vztah je vytvořen pomocí vazební tabulky, v níž se nachází údaj o množství ingredience použité v daném receptu v její výchozí jednotce.

- **Alergen** – tento model zastupuje jeden ze 14 alergenů typicky se vyskytujících v potravinách. Značení alergenů vychází z vyhlášky *113/2005 Sb. O způsobu označování potravin a tabákových výrobků, § 8 odstavec 10*[29].
  Tyto alergeny se vážou k jednotlivým ingrediencím vztahem *Many to Many*, kdy je možno tyto vazby vkládat nebo upravovat administrátorem při vytváření nebo editaci údajů o ingredienci.
  Dále se jednotlivé alergeny vážou k uživateli, kdy vazba na alergen značí uživatelovu alergii na danou ingredienci. Tyto vazby lze vytvořit v úvodním vyplnění uživatelských údajů a měnit v uživatelském nastavení, viz 4.3.5. Pokud je uživateli generován do jídelníčku recept obsahující ingredicenci, na kterou je uživatel alergický, je na to zřetelně upozorněn barevným odlišením ingredience v detailu receptu. Tento detail lze zobrazit kliknutím na jednotlivé položky v jídelníčku.

- **Type** – model zastupující druh denního jídla, pro které je daný recept vhodný *(snídaně, přesnídávka, oběd, svačina, večeře)*. Vhodnost pro daný typ jídla je poté zohledněna při generování receptů pro tvorbu jídelníčku.
  Protože jeden recept může být vhodný pro více denních jídel, je mezi modely Type a Recipe vazba *Many to Many*.

5.1.3 Modely pro jídelníček

Poslední skupinou modelů jsou modely pro plnění hlavní funkce této webové aplikace, tedy vytváření a obnovování jídelníčků komponovaných podle dietního plánu, který uživatel podstupuje. Celá funkce je realizována pomocí následujících čtyř modelů:

- **Daymeal** – model ztvárňující denní jídelníček. Každému uživateli je při registraci vytvořeno a přiděleno sedm instancí těchto entit, stejně jako sedm záznamů v data-

bázové tabulce `daymeals`, která s tímto modelem koresponduje. Na každou instanci modelu Daymeal je použitím vazební tabulky vázáno pět instancí typu Recipe, které znázorňují pět jednotlivých denních jídel, tedy *snídani, přesnídávku, oběd, svačinu* a *večeři*. Tyto recepty jsou přidělovány metodou obsaženou v modelu Foodpicker (viz kapitola 5.1.4) podle typu denního jídla, pro který je daný recept vhodný.
Denní jídelníček je platný do uplynutí konkrétního dne, pro který je jídelníček vytvářen. Po vypršení této platnosti ale není mazán záznam dané instance v databázi, ale pouze zrušeny vazby mezi touto instancí denního jídelníčku a denními jídly, která byla pro tento den generována.

- **Weekmeal** – podobně jako znázorňuje model Daymeal denní jídelníček, znázorňuje model Weekmeal jídelníček pro celý týden. Uživateli je při registraci vytvořena jediná instance tohoto typu a její příslušný záznam v databázové tabulce `weekmeals`. Tento záznam, stejně jako záznamy v tabulce `daymeals`, zůstává pro uživatele stejný po celou dobu používání aplikace.
  Vazba mezi příslušnými entitami modelů Weekmeal a Daymeal je realizována opět pomocí vazební tabulky. Ta kromě cizích klíčů obou entit obsahuje také číselný údaj, jež určuje konkrétní den v týdnu (počítáno od pondělí), který je danou instancí denního jídelníčku znázorněn.

- **FoodPicker** – tento model nereprezentuje žádnou databázovou entitu. Jeho využití spočívá v řízení výběru vhodných receptů pro použití ve stravovacím plánu. Při každém generování je vytvořena jeho dočasná instance, která dále pomocí v něm obsažených metod dělí recepty podle vhodnosti v závislosti na dietním plánu podstupovaném uživatelem, typu generovaného denního jídla *(snídaně, oběd, atd.)* a dále podle kritérií vypočítaných z pokroků zaznamenaných uživatelem. Tyto pokroky jsou převáděny na bodové hodnocení, které je dále použito pro konkretizaci požadavků na dané denní jídlo. Nejvhodnější recept je poté předán pro napojení na denní jídelníček, jehož generování probíhá.

- **Shoplist** – podobně jako model FoodPicker, ani model Shoplist nezastupuje v aplikaci žádnou databázovou entitu. Úkolem tohoto modelu je shromažďovat informace o ingrediencích v jednotlivých receptech a jejich množství. Tyto informace dále slouží k výpočtu množství jednotlivých surovin potřebného k přípravě receptů generovaných v jídelníčku.
  Tyto informace jsou po svém zpracování předávány uživateli v podobě přehledné tabulky, kde u každé suroviny je vyčísleno i potřebné množství uvedené ve výchozí jednotce. Tuto tabulku si uživatel může vytisknout jako jednoduchý a přehledný nákupní seznam.

5.1.4 Generování jídelníčků modelem FoodPicker

Jak již bylo zmíněno v předchozí kapitole, o výběr vhodných jídel při jednotlivých generováních denních jídelníčků se stará model **FoodPicker**. Tento model nezastupuje žádnou databázovou entitu, jeho funkcí je pouze sjednocení algoritmů pro výběr vhodného receptu z databáze receptů, zastoupené modelem **Recipe**.

Generování nového denního jídelníčku nastává bezprostředně po zadání nového měření pokroku v tělesných úbytcích. Pro správné fungování aplikace je toto měření po uživateli vyžadováno v pravidelném třídenním intervalu. Po vložení měření je vypočítána hodnota uživatelova akutálního bazálního metabolismu, podle které jsou nastavena kritéria pro výběr jednotlivých jídel. Protože generování denního jídelníčku probíhá v intervalu tří dnů, generují se současně jídelníčky pro tři další dny. Při dodržování správného intervalu vkládání tělesných měření tak uživatel vidí vždy jídelníček na nejméně tři, nejvíce šest dní dopředu.

Nedodržení vkládání měření nových pokroků v tělesném úbytku neomezí generování jídelníčků. Každý den zpoždění je kompenzován zvýšením množství denních jídelníčků, tedy dní, pro které jsou jídelníčky generovány. Tento počet dní je shora omezen na šest, stejně jako při registraci do aplikace. Generování více dní najednou by dramaticky snížilo efektivitu podstupovaného stravovacího či dietního plánu a taktéž omezilo uživatele v možnosti stravovací plán změnit – změna tohoto plánu je po jejím provedení v nastavení uskutečněna až při následujícím generování jídelníčku.

Model FoodPicker pracuje se záznamy v databázi uživatele o dietním či stravovacím plánu, který podstupuje. Podle tohoto plánu vybírá vhodné recepty, jež třídí podle typu denního jídla, které je v danou chvíli vyžadováno. Kritéria pro výběr jednotlivých jídel jsou odlišná pro jednotlivé dietní plány. Pro každý plán je tedy nutno použít vlastní metodu obsahující dotazovací algoritmus pro výběr nejvhodnějšího receptu z databáze. Příslušnost receptu k dané dietě je dána jejich vztahem, definovaným v kapitole 4.2.2.

Uživateli jsou přednostně vybírány recepty, které mu v minulosti nebyly poskytnuty. Po vyčerpání těchto receptů jsou recepty vybírány podle data jejich výběru. Proto nedochází k častému opakování receptů. Díky parametrům výběru, které jsou upravovány na základě dosažených výsledků v tělesných úbytcích, nejsou recepty vybírány ani cyklicky.

Kromě generování celých denních jídelníčků po zadání uživatelova tělesného měření se model FoodPicker stará také o nahrazení výskytu receptu v jídelníčcích po jeho vyřazení. Vyřazení receptu vyvolá jeho odstranění ze všech denních jídelníčků, které byly uživateli generovány a jejichž platnost ještě nevypršela. Uživateli je místo tohoto receptu vyhledáno jiné jídlo odpovídajícího typu denního jídla, které vyhovuje kriteriím **posledního zadaného měření**. Pro tato nahrazování jsou použita pouze dříve generovaná jídla. Tím je zabráněno, aby si uživatel vygeneroval kompletní nový jídelníček dříve, než mu aplikace umožní vložit nové měření (tedy dříve než za 3 dny).

5.2 Pohledy *(Views)*

Pro předání vizuálního výstupu aplikace uživateli slouží pohledy *(Views)*[16]. Jedná se o soubory zdrojového kódu kombinujícího jazyky HTML a JavaScript, používané pro tvorbu webových stránek s prvky skriptovacího jazyka PHP.

Pro zachování čistoty kódu je ve frameworku Laravel programová logika oddělena od těchto pohledů, takže množství programových bloků v PHP vložených v HTML kódu je minimální. Nelze však tvořit dynamické webové aplikace, které by alespoň minimum těchto bloků neobsahovaly.

Vzhled databáze je založen na open-source CSS frameworku Bootstrap[18]. Tento framework umožňuje jednoduše vytvořit vzhledově přijatelné webové stránky a aplikace s uživatelsky přívětivým rozhraním. Výchozí barevnou kombinací prvků aplikace je kombinace barev oranžová a bílá.

5.2.1 Blade PHP

Pro ještě větší zpřehlednění a kultivaci kódu slouží nástroj **Blade PHP**[12], vycházející z jazyka PHP jako jeho nástavba, který syntaxi nejčastěji používaných konstrukcí tohoto jazyka nahrazuje vlastní, zjednodušenou syntaxí psanou přímo v kódu HTML - není třeba používat uvozovací značky pro vložení programového bloku (`<?php ... ?>`). Mezi tyto konstrukce patří např. podmínky ***if*** – ***else***, cykly ***foreach*** nebo funkce pro výpis řetězce na obrazovku uživatele (***echo***).

Další vlastností této nástavby je rozšíření vytvořené HTML šablony o různý obsah definovaný v jednotlivých pohledech. Tímto způsobem je omezena redundance zdrojového kódu a zabráněno nesourodosti aplikace jako celku. Pohled sloužící jako HTML šablona je předepsaná struktura prvků bez vloženého textového nebo grafického obsahu. V této struktuře jsou definovány sekce, do nichž bude později vkládán obsah definovaný jednotlivými rozšiřujícími pohledy. V těchto pohledech již není vytvářena kompletní podoba HTML dokumentu, ale pouze obsah určený k naplnění jednotlivých sekcí.

Programové konstrukce jazyka Blade pracují s proměnnými předávanými ovladačem, kterým je pohled volán. Těmito proměnnými mohou být jednoduché nebo strukturované datové typy, např. instance třídy modelu. Samotný pohled je v ovladači volán např. tímto voláním:

```
$recipes = Recipes::all();

return view('recipes')->with(['recipes' => $recipes]);
```

Proměnná **$recipes** zde představuje kolekci všech receptů uložených v databázi aplikace. K této kolekci přistupujeme pomocí Eloquentu přes model Recipes. Metoda `with()` přidává pole jednotlivých proměnných. Jednotlivé proměnné jsou v pohledu volány podle klíče, který je jim v metodě `with()` přidělen.

Protože jsou v aplikaci použity dvě role uživatelů - *uživatel, administrátor*, jsou i pohledy rozděleny do podobných skupin podle uživatelů, kterým jsou aplikací vykreslovány. Určování přístupu uživatelů k daným pohledům je definováno pomocí **směrovacích skupin** *(Routegroups)* s přístupem omezeným pomocí **middleware**.

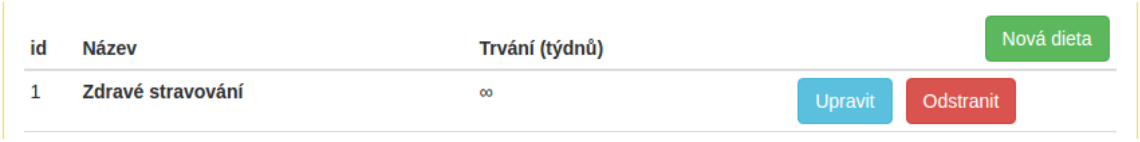
5.2.2 Pohledy pro administraci

Protože úkolem administrátora je správa databázového obsahu aplikace, odvíjí se od toho i jemu vykreslované pohledy. Náplní práce administrátora je vkládat, upravovat a mazat jednotlivé položky v databázi ingrediencí, receptů a diet. Souhrnně se tyto úkony označují pojmem **CRUD** operace *(Create, Read, Update, Delete)*.

Pro vkládání nových záznamů o dietách, receptech a ingrediencích slouží příslušný pohled `create`. Tento pohled obsahuje jednoduchý formulář pro vkládání jednotlivých údajů o nové položce. V HTML předpisu je předepsáno, která pole jsou povinná a která naopak ne, stejně jako např. typ vkládané hodnoty (text, číslo, výčet). Pro vytvoření vztahů mezi jednotlivými entitami (např. přiřazení ingrediencí k receptu) slouží pole typu *select* s nabídkou existujících entit. Pro vytvoření vztahu *Many to Many* je toto pole vybaveno tlačítkem „+“ pro vložení dalšího pole s nabídkou. Přidání nových polí je ovládáno **JavaScript** funkcí a je omezena počtem entit obsažených v databázi.

Odesláním formuláře jsou data předána pomocí **routeru** příslušnému ovladači, který zprostředkovává jejich vložení do databáze.

Po vložení položky do databáze je tato nová položka zobrazena v seznamu všech položek téhož typu. Toto zobrazení je předepsáno jako tabulka (viz obr. 5.1) v příslušném pohledu `list`. Položky v seznamu jsou řazeny podle svých primárních klíčů. V tabulce jsou dále zobrazeny některé další základní informace. Posledními dvěma prvky každého řádku zobrazené tabulky jsou tlačítka pro editaci a vymazání záznamu z databáze.



Obrázek 5.1: Zobrazení seznamu diet

Formulář pro vyplnění dat k editaci jednotlivých záznamů vychází z předlohy pohledů pro vytvoření nového záznamu. Jediný rozdíl je v předvyplnění formuláře údaji již existujícího záznamu v databázi, který se administrátor rozhodl upravit. Po odeslání jsou data stejně jako při vytváření předána příslušnému ovladači, kde tentokrát nedochází ke vkládání nového záznamu, ale k přepsání stávajícího záznamu upravenými daty.

Mazání jednotlivých záznamů je uskutečněno pomocí tlačítka na potvrzení prázdného formuláře (uživateli skrytého) s metotou z `POST` na `DELETE`. Toto řešení bylo zvoleno z bezpečnostních důvodů, aby se (stejně jako při odhlašování) uživateli zamezilo volání metody pro vymazání záznamu pouhým vepsáním URL adresy do pole prohlížeče. Způsob přechodu na definovanou cílovou adresu je dále zkoumán routerem, který zaznamenáním takového pokusu vyvolá výjimku, která zamezí dokončení operace.

5.2.3 Pohledy pro uživatele

Základním pohledem, který se uživateli nabízí po přihlášení, je **uživatelský profil** (stejně tak i *administrátorovi*, ale pouze se základními informacemi - jméno, funkce, profilový obrázek). Na tomto profilu se kromě uživatelských informací (jméno, věk, profilový obrázek) zobrazují informace o podstupovaném dietním plánu, údaje o uživatelském pokroku z předchozího měření doplněné o vypočítanou hodnotu BMI *(Body mass index, indikátor tělesné hmotnosti v poměru s jeho výškou)*[2] a její slovní ohodnocení. Následují informace o příštím

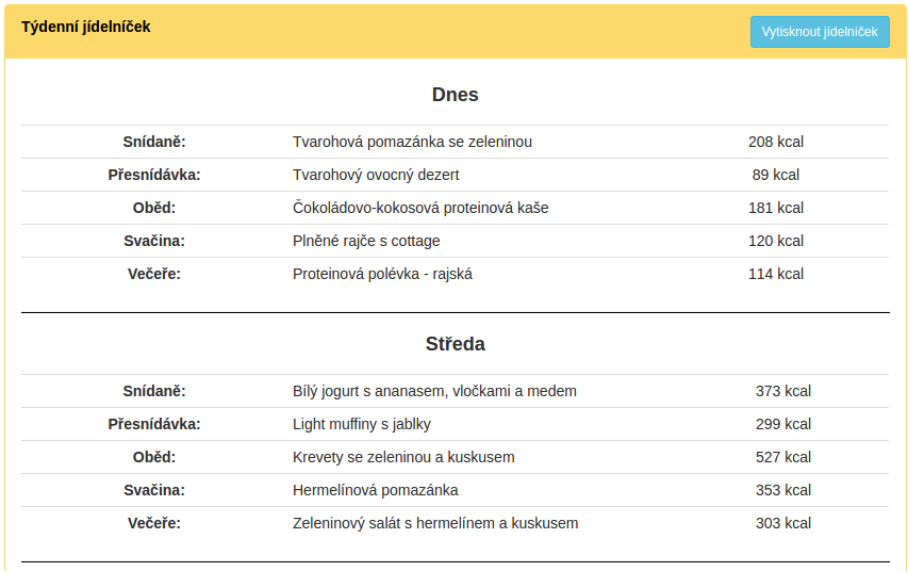
jídle podle jídelníčku s odkazem na recept pro jeho přípravu. Pokud si uživatel zakoupil za své body nějaké odměny (viz odst. Rewards v kap. 5.1.1), jsou tyto odměny také zobrazeny na úvodní obrazovce. Posledním prvkem zobrazeným na této obrazovce je výzva pro vložení nových údajů z měření pokroků.

Hlavní skupinou uživatelských pohledů jsou pohledy pro zobrazení jídelníčků a receptů. Těchto pohledů je celkem pět:

- **Denní jídelníček** – zobrazení všech denních jídel pro tento den včetně předpokládaných časů jejich konzumace s předepsaným rozestupem.

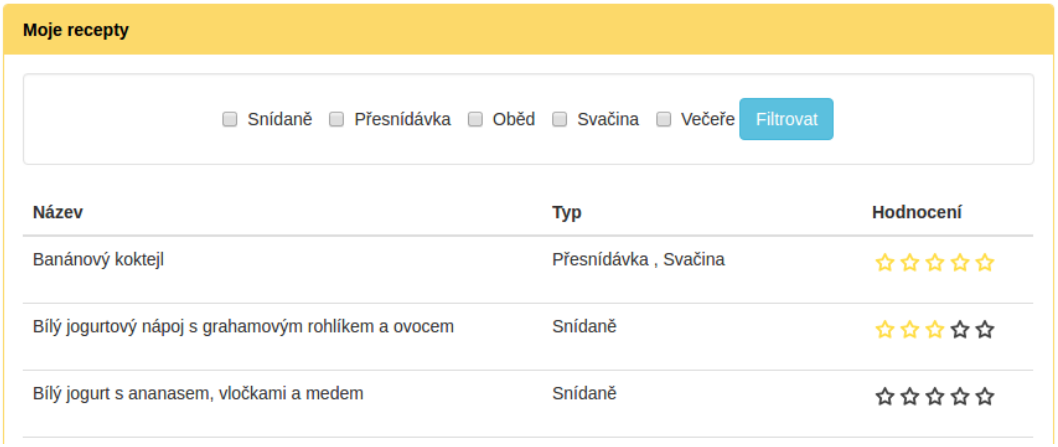
Obrázek 5.2: Denní jídelníček

- **Týdenní jídelníček** – seskupuje generované denní jídelníčky pro celý týden od aktuálního dne. Dny, pro které nebyl jídelníček dosud generován, jsou zobrazeny bez receptu.



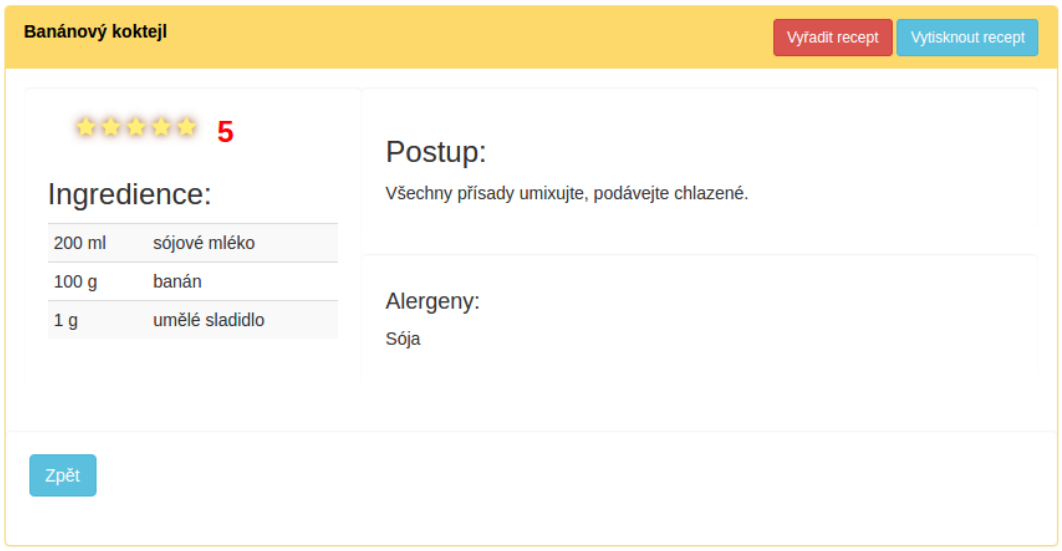
Obrázek 5.3: Týdenní jídelníček (část)

- **Zobrazené recepty** – historie receptů v minulosti generovaných uživateli. Recepty jsou zobrazeny v přehledné tabulce spolu s jejich ohodnocením provedeným uživatelem. Zobrazení receptů lze filtrovat podle příslušného denního jídla.



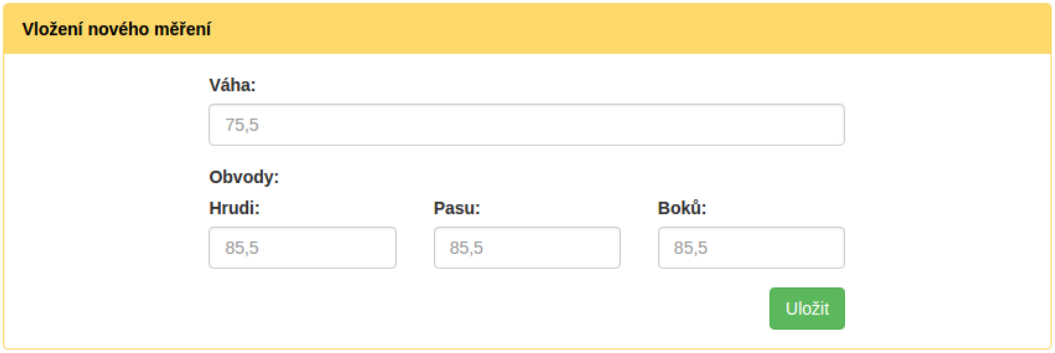
Obrázek 5.4: Tabulka zobrazených receptů (část)

- **Vyřazené recepty** – recepty, které se uživatel rozhodl nepoužívat v receptech. Po rozkliknutí detailu těchto receptů lze příslušným tlačítkem recept zpět navrátit mezi aktivní recepty.
- **Recept** – detail receptu obsahující potřebné ingredience včetně množství, postup přípravy receptu a alergeny. Detail taktéž obsahuje uživatelské hodnocení ve stylu pěti hvězd doplněné o průměr hodnocení provedených všemi uživateli. Na tento detail se uživatel může snadno dostat kliknutím na kterýkoliv recept v jeho generovaném jídelníčku – denním i týdenním – nebo rozkliknutím některého ze zobrazených receptů. Detail obsahuje tlačítko pro vyřazení receptu ze seznamu pro generování a jeho nahrazení jiným jídlem.



Obrázek 5.5: Detailní zobrazení receptu

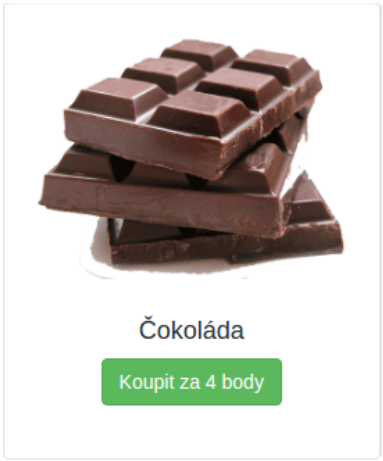
Další skupinou uživatelských pohledů jsou pohledy pro zobrazení a vkládání uživatelských měření **pokroků v tělesném úbytku**. Tyto úbytky jsou zobrazeny v přehledné tabulce, ve které jsou seřazené podle data vložení (nejnovější nahoře). Toto tabulkové zobrazení je pro ještě větší přehlednost a představu uživatele o průběhu a změnách jeho tělesných rozměrů doplněno spojnicovým grafem obsahujícím údaje o všech měřených tělesných partiích. Tento graf je generován nástrojem **Charts**[4] doplněným mezi nástroje frameworku Laravel.



Obrázek 5.6: Vložení tělesného měření

Vložení nového pokroku je možné pomocí jednoduchého formuláře v pohledu `create`, který stejně jako formuláře pro administrační účely aplikace odesílá shromážděná data přes router do příslušného ovladače. Data vkládaná do tohoto formuláře jsou omezena shora i zdola, aby se rozměry mezi jednotlivými měřeními nelišila více než o pět. Vložením nového měření se k počítanému bodování přičítají i bonusy za fyzickou aktivitu. Tato fyzická aktivita je vkládána pohledem `activity` obsahujícím opět jednoduchý formulář pro vkládání údajů o provedené aktivitě *(kilometry, čas)* podle jejího typu.

Pohled `rewardStore` slouží jako „e-shop“ odměn za body, které uživatel doposud nasbíral za své tělesné úbytky a fyzickou aktivitu. Stránka je rozdělena na šest položek odměn, z nichž si může vybrat kteroukoliv, na jejíž cenu (v bodech) stačí jeho nasbírané body. Po jejím „zakoupení“ je uživateli tato odměna na tři dny přičtena a je mu odečten příslušný počet bodů.



Obrázek 5.7: Jedna z položek v nabídce odměn

Dalším uživatelským pohledem je **Nákupní seznam**. Obsah tohoto pohledu je generován modelem **Shoplist** a slouží jako tabulkový výpis všech surovin a jejich množství, jež jsou potřeba na přípravu všech receptů, které jsou uživateli v daném čase generovány.

Pohledy **Denní jídelníček, Týdenní jídelníček, Detail receptu a Nákupní seznam** je uživateli umožněno vytisknout. Tato možnost je realizována pomocí speciálních k tomu určených pohledů vybavených funkcemi v jazyce **JavaScript**, které umožňují tisk stránky. Tyto pohledy jsou tvořeny jako alternativy webových pohledů upravené pro **formát A4**.

5.2.4 Pohledy nezávislé na roli uživatele

Jak již bylo zmíněno v kapitole 5.2.3, jedním z těchto pohledů je první pohled, který je uživateli po jeho přihlášení vykreslen – **uživatelský profil**. Protože je ale pro uživatele značně omezen jeho obsah, nebude dále v této kapitole zmiňován.

Pohled totožný pro obě uživatelské role *(uživatel, administrátor)* je pohled **Nastavení**, definovaný souborem `settings`. Tento pohled sestává z jednotlivých formulářů pro změny údajů o uživateli. Údaje, jejichž změnu je možné provést, jsou vyjmenovány v kapitole 4.3.5. Odeslání formulářů, a tím provedení úpravy údajů, je třeba potvrdit ve vyskakovacím oknu generovaném funkcí v JavaScriptu. Pro vykonání smazání účtu (poslední z formulářů) je třeba zadat heslo.

Nezávislé na roli jsou i pohledy vykreslené bez přihlášení uživatele. Mezi ně patří **úvodní stránka** obsahující stručné informace o funkcích webové aplikace a v ní obsažených dietních plánech. Tento pohled je vykreslen při prvním načtení stránky ve webovém prohlížeči. Dalšími z těchto pohledů jsou pohledy pro autentizaci – **Přihlášení, Registrace, Úvodní informace, „Zapomněl jsem heslo“** a **Vložení nového hesla**. Všechny tyto pohledy obsahují jednoduchý formulář pro vkládání údajů podle své funkce.

Posledními skupinami pohledů jsou pohledy pro definování obsahu e-mailových zpráv a vykreslení chybových hlášení. E-mailové zprávy jsou posílány uživateli při úkonech spojených s autentizaci – při registraci a žádosti o obnovení hesla. Odesílání zpráv řídí metody ovladače `AuthenticationController`. Stránky s chybovým hlášením jsou vykreslovány uživateli v případě, že je zachycena výjimka ohlašující určitý druh problému. Výjimky, které mohou nejčastěji nastat, jsou:

- **Požadovaná stránka nebyla nalezena** – uživatel zadal adresu, jejíž cíl nebyl definován žádným pravidlem routeru.
- **Neplatné odeslání formuláře** – sezení uživatele vypršel časový limit a **CSRF token**[6] zadaný do formuláře pro bezpečnost aplikace nemůže být ověřen.
- **Jiná chyba** – zachycení výjimky, na kterou nebyla definována výchozí reakce.

5.3 Ovladače *(Controllers)*

Jako středobod mezi logikou entit, definovanou pomocí **modelů**, a vizuálním výstupem aplikace, předepsaným v **pohledech**, slouží **ovladače**[5], které volají jednotlivé instance obou těchto skupin, a tím zajišťují jejich interakci. Volání jednotlivých ovladačů a jejich metod je prováděno v **routeru**, který obsahuje předpis těchto volání v závislosti na URL adrese, kterou uživatel zadá, nebo na kterou je odkázán (odkazem nebo odesláním formuláře). Přístup k jednotlivým metodám ovladačů je řízen pomocí **Middleware**, který tento přístup může povolit, nebo zamezit, např. v závislosti na roli uživatele nebo pro přihlášeného či odhlášeného uživatele.

Jednotlivé ovladače se většinou pojí s konkrétním modelem, jehož činnost ovládá. Například řízení **CRUD operací** nad jednotlivými modely je definováno v odpovídajících ovladačích metodami pro každou z těchto operací. Program **Artisan** nabízí funkci `resource`, která při vytváření ovladače automaticky generuje těla funkcí pro odpovídající CRUD operace a taktéž předepíše pravidlo pro router, které tyto operace volá po načtení URL odpovídající metodou.

Stejně jako modely i ovladače jsou děleny do logických celků podle svého uplatnění a vztahujících se modelů.

5.3.1 Ovladače pro uživatele

Tato skupina ovladačů zajišťuje řízení průběhu autentizace uživatele, nastavení jeho uživatelského účtu a doplňkové funkce *(např. tisk receptů a jídelníčků)*. O všechna tato řízení se starají následující ovladače:

- **HomeController** – Ovladač pro řízení vykreslení obsahu webové aplikace po jejím načtení webovým prohlížečem. Vykreslení úvodní obrazovky a základních informací o dietách je řízeno tímto ovladačem.
- **AuthenticationController** – účelem tohoto ovladače je řízení průběhu registrace, přihlášení a odhlášení uživatele a dalších úkonů, které s tímto procesem souvisí. Ovladač pracuje především s modelem `User`, zpřístupněným pomocí balíku **Sentinel**, který mu poskytuje doplňkové funkce usnadňující autentizaci uživatele.

  Tyto funkce jsou volány jednotlivými metodami, které jsou volány pomocí směrování v **routeru**. Díky tomu je možné snadno určit požadovanou činnost aplikace, a to např. vykreslení pohledu s registračním formulářem a poté zpracování dat do něj zadaných uživatelem.

  Jednotlivé funkční prvky tohoto ovladače jsou podrobně popsány v následujících odstavcích.

  - **Registrace uživatele** – Při zpracování požadavku uživatele o registraci je nejprve vykreslen pohled obsahující registrační formulář. Po vyplnění požadovaných údajů jsou tyto údaje předány routerem další metodě jako požadavek typu **Request**. Tato entita obsahuje vždy údaje z odeslaného formuláře jako atributy pojmenované označením vstupu *(atribut „name“ v HTML elementu „input“)*. Data z formuláře jsou zkontrolována validátorem, který ověří správnost jejich tvaru *(tvar e-mailu, délka hesla, shoda hesla s opětovným zadáním pro*

kontrolu) a uložena do tabulky `users` v databázi. Dále je do databázové tabulky `activations` vložen záznam o vytvoření uživatelského účtu a na zadaný e-mail je odeslána zpráva obsahující odkaz pro dokončení této aktivace. Bez tohoto dokončení není možné uživatele přihlásit. Dokončení registrace způsobí vytvoření sedmi instancí denních jídelníčků napojených na jednu instanci týdenního jídelníčku a jejich přidělení uživateli. Posledním krokem je vložení vstupních údajů o uživateli – jeho aktuální míry, hmotnost, datum narození a alergie na potraviny.

- **Přihlášení a odhlášení uživatele** – Do vytvořeného a aktivovaného účtu lze uživatele přihlásit pomocí jeho registrované e-mailové adresy a hesla. Jakmile router zaznamená požadavek o přihlášení uživatele, příslušná metoda v ovladači vykreslí pohled s přihlašovacím formulářem. Zpracování údajů z tohoto formuláře probíhá podobně jako při registraci, tedy nejprve probíhá validace vložených údajů. Po validaci je v databázi uživatelů vyhledán záznam s odpovídající e-mailovou adresou a kontrolována shoda hashovaných otisků hesel (heslo samotné není uchováváno, pouze řetězec obsahující jeho otisk vytvořený funkcí `bcrypt()`). Pokud uživatel povolil v registračním formuláři možnost „Zůstat přihlášen“, je příslušnou funkcí v balíku Sentinel uložen do webového prohlížeče záznam v podobě souboru Cookies a uživatel je automaticky přihlášen při příští návštěvě webové aplikace. Po úspěšném přihlášení do aplikace je uživatel odkázán na hlavní stranu aplikace obsahující vykreslený pohled s uživatelským profilem.

  O odhlášení se stará jednoduchá metoda volající funkci balíku Sentinel, která zruší aktivní sezení uživatele. K této metodě se lze dostat pouze metodou `POST`, tedy odeslaným formulářem. Díky tomu je zajištěno, že není možné odhlásit uživatele pouze zadáním `/logout` na konec URL adresy (tedy požadavkem metody `GET`).
- **Obnovení zapomenutého hesla** – Po načtení tohoto požadavku je uživateli vykreslen pohled obsahující formulář, jehož jediným vstupem je pole pro zadání registrované e-mailové adresy. Tato adresa je vyhledána v databázi uživatelů. I při neúspěšném vyhledání (tj. neexistuje uživatel s touto adresou) je uživateli vrácena hláška o odeslání zprávy obsahující odkaz pro obnovení hesla. Toto opatření je provedeno z důvodů neposkytnutí informace o neexistující registraci. Pro registrovaný e-mail je v databázové tabulce `reminders` vytvořen záznam obsahující náhodně vytvořený kód, pomocí kterého je uživatelskému účtu přiděleno nové heslo bez nutnosti přihlášení uživatele. O vytvoření i zpracování tohoto záznamu se stará balík Sentinel. Přechodem na odkaz odeslaný v e-mailové zprávě se uživatel dostává na stránku obsahující pohled s formulářem pro nové heslo a jeho potvrzení opětovným zadáním. Tato data jsou opět v ovladači kontrolována validátorem. Po úspěšné validaci je aktualizováno heslo v uživatelské tabulce a uživatel je odkázán zpět na přihlašovací formulář, kde již bude moci použít nové heslo.

- **SettingsController** – Tento ovladač řídí chování uživatelského nastavení. Toto nastavení je detailně popsáno v kapitole 4.3.5. Ovladač obsahuje několik metod, každou zvlášť pro jednotlivá nastavení. Tyto metody pracují především s modelem `User`, ve kterém je v tomto nastavení možno provést nejvíce změn.

  Protože změna diety provedená v ovladači metodou `changeDiet()` není patrná okamžitě, ale až při následujícím generování jídel korespondujících s dietním plánem, je

nutno kromě cizího klíče odkazujícího na aktuální dietu uložit do databáze také údaj o této změně. Tento údaj je používán pro správné zaznamenání začátku dodržování jiného dietního plánu u diet, které jsou dodržovány pouze po omezenou dobu. Začátek je tedy brán jako den generování jídelníčku a ne jako den změny diety v nastavení.

Nastavení profilového obrázku je realizováno jako formulář, jehož vstupem je soubor nahraný uživatelem. Po nahrání je tento soubor validován na správný typ *(přípona jpg a jpeg)* a na velikost *(max. 512 kB)*. Uložení obrázku je poté provedeno pod názvem upraveným podle primárního klíče uživatele a obrázek je umístěn do veřejného adresáře. Pod uloženým názvem je obrázek dále načítán v HTML kódu uživatelského profilu. Po vymazání tohoto obrázku je název nastaven zpět na výchozí obrázek.

- **PrintController** – Metody tohoto ovladače jsou zavolány při každém požadavku uživatele pro tisk aktuální stránky. Tisknutelnými stránkami jsou Detail receptu, Denní a Týdenní jídelníčky a Nákupní seznam. Ovladač používá pro tisk každé stránky jinou metodu volající pohled upravený pro tisk na stránku formátu A4, který je vybavený JavaScript funkcí pro jeho tisk. Po vytištění stránky nebo stornování tisku je uživatel odkázán zpět na předchozí stranu.

- **RewardController** – Tento ovladač řídí nákup odměn za body získané vkládáním uživatelských pokroků v tělesných úbytcích. Stránka pro nákup odměn má podobu zjednodušeného „e-shopu“, kde jsou jednotlivé odměny uživateli zobrazeny, a pokud má nasbíráno dostatečné množství bodů, je uživateli umožněno si tuto odměnu „zakoupit“. Tento nákup je platný tři dny, nebo do doby, kdy uživatel potvrdí, že svou odměnu čerpal. Nakoupené odměny jsou uživateli zobrazeny na hlavní stránce jako součást uživatelského profilu.

5.3.2 Ovladače pro diety a recepty

Ovladače pro recepty slouží především jako prostředky pro zpracování vytvořených a upravených údajů v dietách, receptech a ingrediencích. Kromě metody pro zobrazení těchto entit tedy všechny slouží k administraci a běžnému uživateli zůstávají nepřístupné. Metody v těchto ovladačích pracují s instancemi modelů `Diet`, `Ingredient` a `Recipe` a implementují řízení **CRUD operací** nad těmito modely.

- **Vytvoření nového záznamu *(Create)*** – O vytvoření záznamů jednotlivých modelů se starají dvě metody příslušného modelu. Metoda `create()` způsobí vykreslení pohledu obsahujícího formulář pro vkládání všech potřebných údajů.

  Metoda `store()` tyto údaje zpracovává a ukládá je do databáze jako novou entitu naplněnou údaji z formuláře. Přístup do databáze je realizován pomocí **Eloquentu**, který kromě vytvoření samotného záznamu v tabulce korespondující s daným modelem vytváří i vazby mezi novými entitami a vztahujícími se existujícími prvky. Tyto vztahující se prvky jsou pro další administraci přístupny jako atributy daného modelu.

- **Zobrazení záznamu *(Read)*** – Tato metoda je jako jediná přístupná běžnému uživateli. Metoda po svém volání vyhledá záznam v příslušné databázové tabulce a vytvoří entitu, které přiřadí atributy tohoto záznamu. Atributy předává pohledu, který je používá pro své vykreslení. Je implementována pouze pro ovladače modelů `Diet` a `Recipe`. Pro model `Ingredient` nemá využití, protože prvky tohoto modelu nejsou uživateli nabízeny jinde než v detailu receptu.

- **Úprava existujícího záznamu *(Update)*** – Podobně jako při vytváření nového záznamu i o úpravu existujícího záznamu se starají dvě metody ovladače. Metoda `edit()` vytvoří novou instanci modelu upravované entity a přiřadí jí atributy jejího záznamu v databázi. Tuto instanci použije při vykreslení formuláře pro vkládání informací o záznamu, tentokrát ale naplněnou existujícími daty vyčtenými z databáze. Tato data je možno upravovat.

  Po uložení dat je volána funkce `update()`, která znovu vyhledá danou entitu v databázi a její jednotlivé atributy upraví podle změněných informací vložených do formuláře. Tato změněná entita poté přepíše stávající záznam v databázi.

- **Smazání záznamu z databáze *(Delete)*** – Tato metoda je volána tlačítkem vyskytujícím se v pohledech pro zobrazení všech záznamů konkrétního modelu (`list`) a pro editaci existujícího záznamu (`edit`). Její volání způsobí vyhledání záznamu v databázi podle primárního klíče a jeho následné odstranění. Pokud existují vazby mezi mazanou entitou a entitami jiného modelu, jsou tyto vazby před samotným smazáním odstraněny. Po provedení odstranění záznamu je administrátor odkázán zpět na výpis všech záznamů.

Kromě výše popsaných operací splňuje ovladač **RecipeController** funkci řízení vyřazení receptu z generování jídelníčků. Tuto akci vyvolá uživatel tlačítkem v pohledu na detail receptu. Metoda k tomu určená poté přidělí receptu příznak o jeho vyřazení, a pokud je tento recept použit v jídelníčku z předchozího generování, je jeho výskyt vyhledán, zrušen a nahrazen jiným jídlem stejného typu podle nastavení odpovídajícímu dietnímu plánu.

5.3.3 Ovladače pro jídelníčky

Generování jídelníčků probíhá na základě zpracování údajů o uživatelově pokrocích v tělesných úbytcích. O sběr těchto údajů z formuláře pro jejich vložení se stará ovladač `ProgressionsController`. Ten zajišťuje vykreslení formuláře a zpracování vložených dat. Tato data jsou vkládána do databázové tabulky `progressions`, odkud jsou brána pro bodové ohodnocení.

Generování nového jídelníčku je spuštěno při zpracování údajů o novém měření tělesného úbytku. Pokud uživatel dodržuje třídenní interval tělesných měření, zbývají mu tři dny generovaných jídel z minulého generování a přibývají tři další dny – v den měření zná uživatel šest dní dopředu svůj jídelníček. S každým dnem zpoždění přibývá jeden den pro generování denních jídelníčků, čímž klesá efektivita tvorby stravovacích plánů.

K těmto pokrokům a jejich bodovému ohodnocení jsou připočítávána také hodnocení fyzické aktivity zaznamenané uživatelem. Údaje o těchto aktivitách taktéž zpracovává stejný ovladač a ukládá je do speciální databázové tabulky.

Kromě vkládání nových pokroků plní tento ovladač i funkci zobrazení předchozích pokroků v tělesných úbytcích. O toto zobrazení se stará metoda `index()`, která pokroky zobrazuje nejen jako přehlednou tabulku, ale taktéž předává příslušnému pohledu graf, generovaný metodou v modelu `User`, na kterém může uživatel všechny své pokroky pozorovat a sledovat tak tendenci jejich průběhu.

Zobrazení jednotlivých jídelníčků zajišťuje ovladač `MealsController`. Ten obsahuje metody pro zobrazení denních i týdenních pohledů na připravený jídelníček. Dále obsahuje metody pro zobrazení jídel, která byla v minulosti použita pro generování jídelníčku. Díky tomu

může uživatel recepty na tato jídla zobrazit a podle nich je připravit i poté, co jsou odstraněna z jídelníčku po vypršení jeho platnosti. V tomto seznamu jídel může uživatel rovněž filtrovat zobrazená jídla podle typu denního jídla. Tím se zvyšuje jednoduchost zobrazení a rychlost uživatelovy orientace aplikací.

Poslední funkcí ovladače je vložení hodnocení receptu. Toto hodnocení provádí uživatel na základě své spokojenosti s receptem a ovladač jej ukládá do vazební tabulky definující vztah mezi receptem a uživatelem. Uživatel provádí hodnocení v podobě přidělování počtu hvězd (1 až 5). Ve výčtu zobrazených receptů se uživateli zobrazuje graficky počet hvězd, které receptu udělil, v detailním pohledu na recept potom graficky jeho vlastní hodnocení a číselně průměr hodnocení provedených všemi uživateli.

Posledním použitým ovladačem je `ShoplistController`, který zajišťuje vykreslení pohledu s tabulkou všech surovin potřebných pro přípravu jídel v aktuálním jídelníčku. Samotné sčítání surovin probíhá metodou implementovanou v modelu `Shoplist`, ovladač tento soupis rozdělí na tři shodné seznamy a vykreslí tři tabulky obsahující potřebné suroviny. Rozdělení seznamu na tři tabulky bylo zvoleno z důvodů zvýšení přehlednosti stránky a zlepšení orientace mezi jednotlivými surovinami.

Kapitola 6

Zkušební provoz a uživatelské testování

6.1 Testovací data

Pro účely vývoje a testování aplikace byla databáze napojená na aplikaci naplněna daty vhodnými k simulování jejího reálného provozu. Byly vytvořeny dva fiktivní uživatelské účty

- **admin@admin.com** – účet s rolí administrátora
- **basic@basic.com** – účet běžného uživatele – role *basic*.

Administrátorem byly do aplikace přidány požadované diety. Aby bylo možno přidat recepty, bylo nejprve nutno vložit ingredience v těchto receptech obsažené. Po jejich vložení lze vytvořit vztah receptu s ingrediencí, stejně jako vztah receptu s dietou určující jeho použitelnost při absolvování plánu. Pro jednotlivé položky ingrediencí a receptů byly též vytvořeny vztahy s položkami alergenů a typů jídla. Záznamy těchto entit byly dříve přidány *sázecí třídou* (viz kapitola 4.2.1).

Pro zachování vložených dat i při opakovaných vymazáních databáze během jejího vývoje byly vytvořeny sázecí soubory nástrojem `iseed`. Díky těmto souborům byla data po obnovení databáze opětovně nahrána automaticky.

Pomocí uživatelského účtu *basic* bylo prováděno odlaďování nesrovnalostí a chyb při zpracování dat, které způsobovaly problémy při samotném generování jídelníčků. Většina těchto chyb byla způsobena nesprávným využitím zpracovaných dat při generování databázového dotazu pro vyhledání nejvhodnějšího receptu k naplnění jídelníčku.

6.2 Uživatelské testování

Po dobu cca tří týdnů probíhalo testování aplikace za pomoci reálných uživatelů. Cílem tohoto testování bylo odladění programátorem neodhalených chyb a nedostatků za pomoci nového náhledu poskytnutého testovacím subjektem.

6.2.1 Testování prostředí aplikace

Chybami odhalenými testujícími uživateli byly především problémy při registraci a aktivaci uživatelského účtu pomocí aktivačního e-mailu. Aktivační e-mail v některých případech nedorazil a bylo jej třeba znovu odeslat – do ovladače `AuthenticationController` byla proto doplněna metoda pro implementaci opětovného odeslání e-mailu. Možnost opětovného odeslání aktivačního e-mailu je uživateli zobrazena při pokusu o přihlášení neaktivního účtu.

Dalšími úpravami na základě informací získaných od testujících uživatelů byly úpravy uživatelského rozhraní. Díky těmto úpravám bylo rozhraní vylepšeno za účelem zvýšení intuitivnosti, přehlednosti a jednoduchosti.

Mezi tyto změny patří úprava uživatelského profilu, konkrétně pořadí informací, které jsou zde zobrazeny. Hlavní změnou bylo přídání notifikace oznamující požadavek pro vložení měření tělesných pokroků.

Další změnou bylo přidání odkazů pro rychlou navigaci aplikací do bočního panelu obsaženého ve většině pohledů. Tyto odkazy se liší pro jednotlivé pohledy a odkazují uživatele do částí aplikace příbuzných části, v níž se právě nachází.

6.2.2 Testování hlavní funkce aplikace

Dalším testovaným kritériem aplikace byla efektivita při vytváření dynamického stravovacího plánu na míru uživateli. Pro vytvoření ideální představy o účincích aplikace byla doba uživatelského testování příliš krátká. Vyskytly se ale náznaky pozitivního dopadu vytvořených stravovacích plánů na testované uživatele. Toto testování bylo zhodnoceno následovně:

- Uživatelské testování hlavní funkce aplikace trvalo 3 týdny a účastnilo se ho 8 osob.
- Pro prvních 6 dní byl pro všechny generován jídelníček podle dietního plánu **Zdravé stravování**, u některých uživatelů byl jídelníček změněn na:
  - **proteinovou dietu** – 2 osoby (1 muž, 1 žena),
  - **dělenou stravu** – 3 osoby (2 muži, 1 žena),

  3 osoby (2 muž, 1 žena) dietní plán neměnili a dále pro generování používali výchozí dietní plán.
- Oba uživatelé testující **proteinovou dietu** zaznamenali znatelný úbytek hmotnosti i partií, v nichž se usazuje tuk (pas, boky).
- Jeden z mužů testujících **dělenou stravu** zaznamenal znatelný úbytek hmotnosti. U ostatních uživatelů nastaly úbytky spíše zanedbatelné, ale došlo ke zlepšení jejich fyzické kondice a zmírnění únavy po jídle.
- Změny hmotnosti u mužů testujících **zdravé stravování** byly nižší než u uživatelů testujících proteinovou dietu, ale nebyly zcela zanedbatelné. Znatelné byly pokroky v úbytcích obvodů břicha. Stejně jako u předešlých uživatelů došlo ke zlepšení kondice. Žena, která testovala tuto dietu, nepociťovala žádné změny k lepšímu, ale ani k horšímu.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo vytvoření webové aplikace. Tato aplikace napomáhá uživateli ke zlepšení stravovacích návyků a dodržování dietních plánů. Přínosem aplikace je automatické generování dietních či zdravých jídelníčků podle potřeby uživatele, jejichž tvorba by pro neznalého uživatele byla náročná. Dodržování těchto jídelníčků a úprava stravovacích návyků na jejich základě přispívá k zlepšení zdraví a vzhledu uživatele.

Podoba a funkcionalita aplikace jsou výsledkem analýzy požadavků potenciálních uživatelů provedené na základě jejich odpovědí ve veřejném dotazování, které proběhlo před začátkem samotné implementace webové aplikace.

Pro zajištění správné funkce tvorby jídelníčků podle stravovacích plánů bylo třeba dietní a stravovací plány nastudovat a zvolit vhodný algoritmus pro výběr jednotlivých jídel k seskupení do jídelníčků. Tyto jídelníčky generované pro jednotlivé dny tvoří stravovací plán přizpůsobený potřebám uživatele na několik dní dopředu.

Back-end aplikace je napsán ve frameworku Laravel vytvořeném v jazyce PHP, Framework vychází z návrhového vzoru *Model, View, Controller*. Front-end je vytvořen pomocí pohledů v jazyce HTML, JavaScript a Blade PHP a jeho styl vychází z CSS frameworku Bootstrap. Pro účely vývoje aplikace fungovala na lokálním serveru, na kterém se vyskytovala i její databáze. Důležitou prerekvizitou k vytvoření této aplikace bylo získání znalostí o ovládání těchto nástrojů.

Ve finální fázi vývoje aplikace bylo provedeno uživatelské testování, které poukázalo na některé chyby či nedostatky, které díky tomu mohly být odstraněny. Stejně tak napomohlo k zlepšení intuitivnosti uživatelského rozhraní.

V projektu Webová aplikace pro tvorbu dietních plánů bude možno nadále pokračovat zvyšováním množství dietních a stravovacích plánů dostupných uživateli a receptů, ze kterých tyto plány čerpají. Dále je možno zvýšit množství funkcí, které aplikace uživateli poskytne, např. některých funkcí navrhovaných dotazovanými uživateli, jejichž implementace nebyla v rámci bakalářské práce provedena.

Literatura

[1] Česká republika a rakovina v číslech. online, 2011.
URL http://www.linkos.cz/co-musite-vedet/ceska-republika-a-rakovina-v-cislech/

[2] About Adult BMI. online, 2015.
URL https://www.cdc.gov/healthyweight/assessing/bmi/adult_bmi/index.html

[3] Authentication. online, 2016.
URL https://laravel.com/docs/5.3/authentication

[4] Charts documentation. online, 2016.
URL https://erik.cat/projects/Charts/docs/4

[5] Controllers. online, 2016.
URL https://laravel.com/docs/5.3/controllers

[6] CSRF Protection. online, 2016.
URL https://laravel.com/docs/5.3/csrf

[7] Database: Migrations. online, 2016.
URL https://laravel.com/docs/5.3/migrations

[8] Database: Query Builder. online, 2016.
URL https://laravel.com/docs/5.3/queries

[9] Database: Seeding. online, 2016.
URL https://laravel.com/docs/5.3/seeding

[10] Eloquent: Getting Started. online, 2016.
URL https://laravel.com/docs/5.3/eloquent

[11] Eloquent: Relationships. online, 2016.
URL https://laravel.com/docs/5.3/eloquent-relationships

[12] Laravel Templates. online, 2016.
URL https://laravel.com/docs/5.3/blade

[13] Middleware. online, 2016.
URL https://laravel.com/docs/5.3/middleware

[14] Routing. online, 2016.
URL https://laravel.com/docs/5.3/routing

[15] Sentinel by Cartalyst. online, 2016.
URL https://cartalyst.com/manual/sentinel/2.0

[16] Views. online, 2016.
URL https://laravel.com/docs/5.3/views

[17] Arrays. online, 2017.
URL http://php.net/manual/en/language.types.array.php

[18] Bootstrap. online, 2017.
URL http://getbootstrap.com/

[19] Glykemický index. online, 2017.
URL http://www.kaloricke-tabulky.cz/temata/show/glykemicky-index-gi/18/

[20] Historical trends in the usage of server-side programming languages for websites. online, 2017.
URL
https://w3techs.com/technologies/history_overview/programming_language

[21] Gilmore, W. J.: *Velká kniha PHP 5 a MySQL*. Zoner Press, třetí vydání, 2011, ISBN 978-80-7413-163-9.

[22] Havlíček, P.; Lamshová, P.: *Jídlo jako životní styl*. Mladá Fronta, první vydání, 2010, ISBN 978-80-204-2154-8.

[23] Jílek, J.: Bazální metabolismus – co to je? online, 2016.
URL http://www.bazalnimetabolismus.cz/co-to-je

[24] Mandžuková, J.: *Dělená strava*. Vyšehrad, druhé vydání, 2012, ISBN 978-80-7429-280-4.

[25] Novák, P.: *Jak tuky pálit a neukládat*. Petr Novák, druhé vydání, 2016.

[26] Orangehill: Laravel 5 Inverse Seed Generator. online, 2016.
URL https://github.com/orangehill/iseed

[27] Trampota, J.: Dělená strava. online, 2010.
URL http://www.delena-strava.cz/

[28] Vilímovský, M.: Co je ketóza a jak ovlivňuje vaše zdraví? online, 2016.
URL https://cs.medlicker.com/1109-ketoza#co-je-to-ketoza

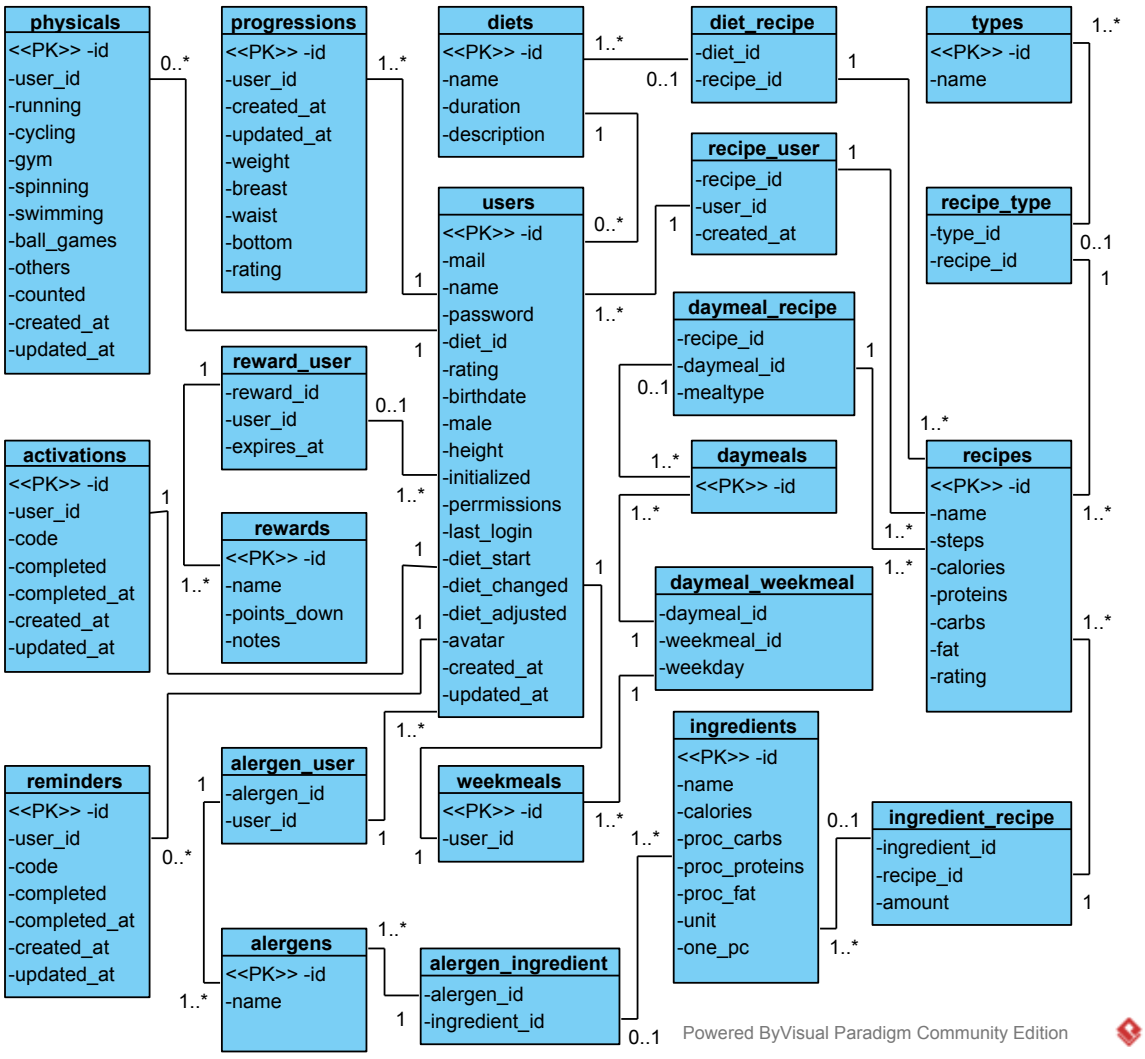
[29] Vondrová, M.: Označování alergenů od 13.12.2014 – je legislativně stanoveno v souladu s potravinovým právem. online, 2014.
URL http://www.akc.cz/clanek-1078/oznacovani-alergenu-od-13-12-2014-%E2%80%93-je-legislativne-stanoveno-v-souladu-s-potravinovym-pravem

[30] Čápka, D.: MVC architektura. 2016.
URL
https://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/

Přílohy

Příloha A

Entity-Relationship Diagram



Příloha B

Model případů užití

Příloha C

Obsah přiloženého DVD

- `/`
  - `README.md` ..... informace pro spuštění aplikace
  - `project/` ..... veškeré zdrojové kódy práce, včetně přejatých
  - `bakalarska_prace/` ..... pouze zdrojové kódy vytvořené v rámci bakalářské práce
    - `app/` ..... modely aplikace
      - `Http/`
        - `controllers/` ..... ovladače aplikace
        - `middleware/` ..... třídy Middleware
    - `database/` ..... soubory pro vytvoření a naplnění databáze
      - `migrations/` ..... soubory pro definování struktury databáze
      - `seeds/` ..... sázecí soubory pro naplnění databáze
    - `public/` ..... složka veřejného úložiště
      - `home/` ..... obrázky zobrazené na úvodní stránce
      - `navbar/` ..... ikony na horní navigační liště
      - `uploads/` ..... obrázky nahrané uživatelem
        - `avatars/` ..... profilové obrázky
      - `rewards/` ..... obrázky z e-shopu odměn
    - `resources/`
      - `views/` ..... jednotlivé pohledy aplikace
    - `routes/` ..... soubor pro směrování
  - `doc/` ..... technická dokumentace bakalářské práce
    - `latex/` ..... šablona technické dokumentace pro LaTeX
    - `bakalarska_prace.pdf` ..... text technické dokumentace v pdf