MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Juraj Trappl

Software nutriční podpory zdraví

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber
Studijní program: Informatika
Studijní obor: Programování a softwarové systémy

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... podpis

Týmto by som chcel poďakovať vedúcemu mojej práce, pánovi Mgr. Filipovi Kliberovi a za užitočné rady pri vývoji bakalárskej práce.

Název práce: Software nutriční podpory zdraví

Autor: Juraj Trappl

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber, Katedra distribuovaných a spolehlivých systémů

Abstrakt: Péče o zdraví by měla být to nejdůležitější v životě. Zdravotní problémy se projeví až později, častokrát až když jsou ireverzibilní. Naše práce se snaží o prevenci vytvořením nástroje pro jeho podporu. Cílem práce je implementace webové aplikace poskytující nutriční plán sestávající se z jídelního plánu, pitného režimu a pohybové aktivity. Prezentujeme poznatky o zásadách správné výživy a požadavcích moderních webových aplikací. Samostatná část je věnována návrhu algoritmu tvorby jídel splňujících optimální poměr živin. Uvádíme možnost exportu obsahu na kalendářové platformy. Součástí práce je responzívny design a ochrana proti skriptovacím útokům. Vzhledem k naší práci se domníváme, že aplikace a dodržování její plánů má význam v podpoře zdraví.

Klíčová slova: webová aplikace, zdravé stravování, nutriční plán

Title: Software for nutritional support of healthy lifestyle

Author: Juraj Trappl

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Filip Kliber, Department of Distributed and Dependable Systems

Abstract: Health care should be the most important thing in life. Health problems do not manifest themselves until later, often when they are irreversible. Our thesis seeks to prevent it by creating a tool to support it. The aim of this thesis is to implement a web application providing a nutritional plan consisting of a meal plan, drinking regime and physical activity. We present knowledge of the principles of proper nutrition and modern requirements of web applications. A separate part is devoted to the design of an algorithm for creating foods that meet the optimal ratio of macronutrients. We present the possibility of exporting content to calendar platforms. Part of this thesis is responsive design and protection against scripting attacks. Given our thesis, we believe that application and adherence to its plans is important in promoting health.

Keywords: web application, healthy eating, nutritional plan

Názov práce: Software nutričnej podpory zdravia

Autor: Juraj Trappl

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedúci bakalárskej práce: Mgr. Filip Kliber, Katedra distribuovaných a spolehlivých systémů

Abstrakt: Starostlivosť o zdravie by mala byť to najdôležitejšie v živote. Zdravotné problémy sa prejavia až neskôr, častokrát až keď sú ireverzibilné. Naša práca sa snaží o prevenciu vytvorením nástroja na jeho podporu. Cieľom práce je implementácia webovej aplikácie poskytujúcej nutričný plán pozostávajúci z jedálneho plánu, pitného režimu a pohybovej aktivity. Prezentujeme poznatky zásad správnej výživy a moderných požiadaviek webových aplikácií. Samostatná časť je venovaná návrhu algoritmu tvorby jedál spĺňajúcich optimálny pomer živín. Uvádzame možnosť exportovania obsahu na kalendárové platformy. Súčasťou práce je responzívny design a ochrana proti skriptovacím útokom. Vzhľadom na našu prácu sa domnievame, že aplikácia a dodržiavanie jej plánov má význam v podpore zdravia.

Kľúčové slová: webová aplikácia, zdravé stravovanie, nutričný plán

Obsah

Úvod

Ľudské telo môžeme symbolicky prirovnať k továrni, v ktorej prebiehajú komplexné procesy. Z fyzikálneho hľadiska je človek nástroj premieňajúci jeden typ energie na druhý. Zdrojom vstupnej energie je potrava, ktorú príjme. Od kvality potravy sa odvíja celý chod organizmu. Ovplyvňuje dôležité aspekty ako napríklad imunitný systém, ktorý nás chráni pred chorobami, fungovanie mozgu, ktorý je centrom nervového systému, starnutie, váhu a mnoho ďalších. Tieto pojmy môžeme zhrnúť do jedného - zdravie. Teda to, čo konzumujeme vplýva priamo na naše zdravie a to je v centre našich životov. Ak trpíme zdravotnými problémami alebo máme nedostatok energie, tak je ťažké posúvať sa ďalej v ľubovoľnom smere, či už je to práca, vzťahy alebo sebarozvoj.

Určenie kvality potravín a množstva prijímanej potravy je náročný proces. Existujú celé vedy, ktoré sa týmito problémami zaoberajú. Je teda pochopiteľné, že bežný jedinec nevie správne zvoliť, ktorú potravinu a v akom množstve má prijímať. Preto je vysoký výskyt civilizačných ochorení súvisiacich s nesprávnou výživou, ktorý je charakterizovaný nadbytočným príjmom energie, bielkovín, nasýtených tukov a jednoduchých sacharidov.

Bežné prístupy k riešeniu problému sú na základe národnostných populačných štatistík alebo individuálneho nutričného poradenstva, ktoré realizuje odborník. Populačné štatistiky sú výsledkom mapovania obyvateľstva a stanovujú priemernú celkovú energetickú potrebu. Odborník uskutoční vstupný rozhovor, meranie, vyhodnotí údaje a zostaví jedálny plán.

V našej práci pristupujeme ku problému tak, že pre každého používateľa vytvoríme *na mieru*, podľa individuálnych parametrov, jedálny plán a pitný režim. V snahe poskytnúť aplikáciu čo najširšej skupine populácie, ponúkame niekoľko plánov - optimálny, redukčný a športový. Práca zároveň pomáha osvojiť si základy zdravého životného štýlu.

Z povahy práce rozlišujeme medzi cieľmi nutričnými a informatickými. Nutričné ciele práce sú výber výlučne zdravých potravín v súlade so zásadami správnej výživy, parametrov pre vymedzenie cieľovej skupiny, návrh funkcionalít plánov a metrík pre ich zostavenie. Informatické ciele práce sú návrh algoritmu tvorby jedál a implementácia webovej aplikácie.

V prvej kapitole predstavíme stručný teoretický základ ľudskej výživy, ktorý je potrebný pre pochopenie nutričnej časti práce. V druhej a tretej kapitole budeme analyzovať nutričné ciele a navrhneme algoritmus pre tvorbu jedál. Štvrtá kapitola sa venuje informatickým cieľom. Ďalej uvidíme programátorskú a používateľskú dokumentáciu. Na záver si zhrnieme výsledky práce.

1. Ľudská výživa

Je súborom biochemických a fyziologických procesov, ktorými organizmus prijíma a využíva látky z vonkajšieho prostredia potrebné pre všetky životné funkcie.

Človek realizuje výživu, ktorej je sám konzumentom. Cieľom vedy o výžive je preto príjem takých potravín a živín, ktoré podporujú duševné a fyzické zdravie človeka, jeho pracovnú aktivitu a dlhovekosť.

Z medicínskeho hľadiska je výživa nielen zaobstarávanie rastlinných a živočíšnych produktov, ich kuchynská úprava a konzumácia, ale najmä digescia a absorpcia živín, ich látková premena, utilizácia a exkrécia nepotrebných látok. Ide o materiálny základ pre existenciu človeka, pre jeho vývin, rast, pre obnovu tkanív a orgánov [1].

1.1 Základné pojmy vo výžive

Potrava je súbor požívatín, ktoré slúžia na výživu človeka. **Požívatiny** sú všetky látky, ktoré človek prijíma ústami a sú prostriedkom jeho výživy. Delia sa na **potraviny**, **pochutiny** a **vodu.**

Potraviny sú také požívatiny, ktoré majú energetickú a/alebo biologickú hodnotu (mäso, mlieko, ovocie); sú to zložky ľudskej potravy živočíšneho a rastlinného pôvodu.

Pochutiny sú požívatiny bez výživovej hodnoty (soľ, koreniny, čaj, …), ktoré svojou chuťou a vôňou stimulujú trávenie.

Voda je požívatina, ktorá je základnou súčasťou všetkých potravín a ktorej výživová hodnota spočíva v tom, že je bezpodmienečne potrebná pre látkovú premenu človeka.

Podľa stavu sa požívatiny rozdeľujú na pokrmy (tuhé a kašovité) a nápoje (tekuté). **Jedlo** je vhodná zostava pokrmov, ktoré sa podávajú v určitom čase (raňajky, desiata, …). **Strava** je zostava denných jedál, ktorá sa posudzuje z hľadiska energetickej a biologickej hodnoty. **Živiny** sú zložky potravín, ktoré vytvárajú ich energetickú a biologickú hodnotu a rozdeľujú sa na základné (bielkoviny, tuky a sacharidy) a ochranné (vitamíny, minerálne látky a vodu) [1].

1.2 Živiny

Rastlinná a živočíšna potrava obsahuje organické živiny, ktoré buď dodávajú najmä energiu (sacharidy a tuky), alebo sú predovšetkým stavebnými látkami (bielkoviny).

Bielkoviny (proteíny) sú najdôležitejšou živinou, pretože sú stavebnou zložkou orgánov ľudského tela. Okrem toho sa významnou mierou zúčastňujú aj na zabezpečovaní funkcie tkanív a orgánov. Základom proteínov sú aminokyseliny. Biologická osobitosť bielkovín v tele človeka spočíva v tom, že ľudský organizmus si musí syntetizovať svoje vlastné proteíny z aminokyselín, z ktorých sa tvoria aj iné nebielkovinové dusíkaté látky, ako sú nukleové kyseliny, kreatín a podobne. Pre správnu proteosyntézu je potrebné, aby bolo v tele k dispozícii dostatočné množstvo všetkých 20 aminokyselín a to tak esenciálnych ako aj neesenciálnych. Ľudský organizmus si nevie vytvoriť esenciálne aminokyseliny, a preto ich musí prijímať zo živočíšnych a rastlinných bielkovín v potrave. Energetická hodnota 1 gramu (g) bielkovín je 17 kilojoulov (kJ) [1].

Sacharidy sú najdôležitejším a rýchlym zdrojom energie pre človeka. Ich dostatočný príjem šetrí rezervy telesných proteínov a tukov. Majú aj iné funkcie; podieľajú sa na formovaní štruktúry a činnosti orgánov a ovplyvňujú látkovú premenu. Môžeme ich rozdeliť na energeticky využiteľné napr. glukóza a energeticky nevyužiteľné (gumy a slizy), ktoré sa označujú všeobecne ako vláknina. Energetická hodnota 1 g sacharidov je 17 kJ [1].

Tuky sú organické látky, ktoré orgány tela používajú predovšetkým na zabezpečenie energie pre svoju činnosť. Zúčastňujú sa na tvorbe štruktúr orgánov a zabezpečovaní ich funkcie. Sú nosičmi iných látok, ktoré sú v nich rozpustné, ovplyvňujú imunitu, inflamáciu, karcinogenézu a iné procesy. Môžeme ich rozdeliť na esenciálne a neesenciálne. Základnou zložkou živočíšneho a rastlinného pôvodu sú glycerol a mastné kyseliny. Mastné kyseliny sa rozdeľujú na saturované, monoénové a polyénové. Saturované a monoénové si ľudský organizmus dokáže syntetizovať, ale polyénové musí prijímať v potrave, a preto sa nazývajú esenciálne. Energetická hodnota 1 g tukov je 38 kJ [1].

1.3 Výživa dospelých a zásady správnej výživy

V súčasnosti veda o výžive disponuje obrovskou poznatkovou bázou týkajúcou sa intermediárneho metabolizmu makro- a mikronutrientov a ich vplyvu na vznik metabolických a orgánových ochorení. Na základe týchto poznatkov sa vypracovali medicínske kritéria správnej, teda fyziologickej výživy, ako aj normatívy, ktoré upresňujú optimálny príjem živín a energie u zdravého človeka [1].

Nesprávna výživa sa podieľa na vysokej morbidite a mortalite osôb predovšetkým v 5. a 6. decéniu života. Zvýšenie morbidity a mortality sa zistilo u jedincov s narušenou výživou (malnutríciou), na čo poukazuje v súčasnosti najjednoduchší, ale najčastejšie sledovaný ukazovateľ zmenenej výživy - telový hmotnostný index (BMI). Podstatné pre našu aplikáciu sú intervaly: < 19 - chudosť, 19,1 - 26,0 - normálna hodnota a 26,1 - 30,0 nadhmotnosť [1].

Pri individuálnom hodnotení stavu výživy človeka je potrebné aj individuálne formulovať *total daily energy expenditure* (TDEE) a živín, pretože požiadavky sú u jednotlivých osôb rozdielne v závislosti od pohlavia, veku, fyzickej aktivity, hmotnosti tela a zdravotného stavu. Na základe takto zistených údajov sa stanoví potreba potravín, a to ich množstvo i štruktúra.

Zo stanovenej TDEE sa určí denný príjem bielkovín, tukov a sacharidov. Optimálne rozloženie energie na jednotlivé živiny sa určí v % takto: tuky < 35 % TDEE, sacharidy 45-60 % TDEE a zvyšok tvoria bielkoviny [2].

Pravidelný príjem **bielkovín** je potrebný, pretože ide o dôležité stavebné látky štruktúrnych a funkčných proteínov tela. Pre organizmus sú vhodné najmä živočíšne bielkoviny, predovšetkým chudé bieleho mäso z rýb, hrabavej hydiny a králika, červené mäso, vaječný bielok a nízkotučné mliečne výrobky. Živočíšne proteíny sú výhodnejšie najmä preto, že sú v nich lepšie dostupné esenciálne aminokyseliny a ich zloženie je blízke ľudským bielkovinám. Vhodné sú aj rastlinné bielkoviny, predovšetkým zo strukovín, zemiakov a celozrnných obilnín [1].

Tuky sú v potrave najproblematickejšími živinami. *Nasýtené tuky živočíšneho pôvodu* - bravčová masť, loj, maslo - sú predovšetkým dodávateľmi energie, ale sú aj nositeľmi cholesterolu, takže podporujú rozvoj aterosklerózy a iných metabolických a orgánových ochorení. Ich množstvo v prijímanej potrave by nemalo byť vyššie než ⅓ z celkovej dávky tukov. *Nenasýtené tuky s monoénovými*

mastnými kyselinami, napr. olivový olej, sú zdravotne výhodné a mali by sa na príjme podieľať rovnakým dielom. Zdravotne veľmi výhodný je *nenasýtený tuk morských rýb* - rybí olej. Dôležité je využívanie tukov pri príprave jedál tak, aby sa tepelne a oxidáciou znehodnotili čo najmenej. Odporúčania ku konzumácii tukov sú takéto: nekonzumovať bravčový tuk a maslo, jesť mäso a mäsové výrobky bez viditeľného tuku, nízkotučné mlieko a mliečne výrobky s nižším obsahom tuku ako 1,5 % a syry s obsahom tuku do 30 %. Z rastlinných tukov je vhodný najmä olivový olej, potom slnečnicový a repkový. Veľmi pozitívny je príjem rybieho tuku [1].

Množstvo **sacharidov** má z TDEE zabezpečiť 45-60 %, pričom ide o objemovo najväčší a najľahšie dostupný energetický zdroj pre organizmus. Jednoduché sacharidy, najmä sacharóza, by sa mali konzumovať do hodnoty 10 energetických %. Sacharidy by sa mali prijímať v potrave vo forme polysacharidov ako *komplexné sacharidy*, ktoré sú nositeľmi aj neškrobových polysacharidov. Nevhodná je konzumácia sladkostí a výrobkov z bielej múky. Zdrojmi vhodných sacharidov sú najmä komplexné sacharidy ako sú celozrnné výrobky z obilnín, ryža, kukurica, zemiaky, ovocie, a med [1].

V správnej výžive má významné miesto **vláknina** v potrave. Jej príjem je stále nedostatočný a možno ho zabezpečiť len pravidelnou a zvýšenou konzumáciou celozrnných obilnín, ovocia a zeleniny tzn. 400-500 g surovej alebo varenej zeleniny a ovocie rozdelené do jednotlivých jedál v priebehu dňa [1].

Príjem **kuchynskej soli** je potrebné obmedziť na maximálne 5 g/deň a **alkoholu** na 20-30 g/deň pre nepriaznivý vplyv na srdcovocievny systém, ako aj riziko poškodenia pečene a nádorových ochorení [1].

Pri príprave jedál je potrebná šetrná **tepelná úprava potravy**, najmä z hľadiska použitia vysokých teplôt, časového intervalu a prítomnosti vzdušného kyslíka. Pri nesprávnej kuchynskej príprave ľahko vznikajú mutagény/karcinogény [1].

Významným a často zanedbávaným faktorom je dodržiavanie pravidelnej **frekvencie príjmu potravy** so správnym rozložením príjmu energie v priebehu dňa. Má sa konzumovať v pravidelných intervaloch v priebehu dňa vo forme 3 hlavných jedál - raňajky, obed, večera s 1-2 menšími porciami (desiata, olovrant), pričom energetický príjem sa rovnomerne rozdelí [1].

Výživa pri fyzickej aktivite, ak nie je extrémne náročná (rekreačný a neprofesionálny šport alebo stredne ťažká manuálna práca) sa riadi zásadami správnej výživy so zreteľom na predpokladané potreby energie a živín.

1.4 Potreba energie

Všetky biologické procesy v organizme vyžadujú energiu. Referenčné hodnoty pre príjem energiu sú udávané v kJ a kilokalóriach (kcal) (1 kcal = 4,184 kJ). Potreba energie predstavuje množstvo energie z potravín, ktoré sú nutné pre vyrovnanú energetickú bilanciu. Vyrovnaná energetická bilancia popisuje fyziologický stav, pri ktorom denný energetický príjem zodpovedá energetickej spotrebe, zaručuje telesnú hmotnosť, zloženie tela a telesnú aktivitu, a ktorá podporuje zdravie [2].

Potreba energie na deň zodpovedá TDEE za 24 hodín. Toto množstvo sa vypočíta faktoriálnou metódou, kde je *basal metabolic rate* (BMR) násobená *physical activity level* (PAL) [2].

1.5 Zisťovanie stavu výživy

Spôsob výživy sa určuje na základe stanovenia kvantitatívnych a kvalitatívnych parametrov prijímanej potravy, pri stave výživy ide zase o určenie parametrov:

- antropometrických (hmotnosť tela, výška, kožné riasy, obvod ramena, pásu a bokov)
- somatických (funkčnosť orgánových systémov)
- biochemických (analýza krvi a moču),

ktoré majú vzťah k výžive a ich porovnanie s fyziologickými hodnotami.

2. Nutričné jadro

V úvode sme popísali nutričné ciele. Z prvej kapitoly sme získali teoretický prehľad o ľudskej výžive, ktoré nám pomôžu v ich analýze. Ciele sú:

1 Výber potravín, ktoré spĺňajú nároky na nutričné zloženie v súlade so zásadami zdravej výživy
2 Výber parametrov a vymedzenie cieľových skupín
3 Nutričné funkcionality aplikačných plánov
4 Aplikačné plány a ich rozdelenie
5 Metóda stanovenia energetickej potreby typického klienta

2.1 Výber potravín

V aplikácii chceme ponúkať klientom nutrične vyvážené jedálne zostavy. Tento cieľ je veľmi podstatný pre výsledok celej práce. V prípade, že by sme boli vybrali nevhodné potraviny, tak by síce aplikácia použiteľná bola, ale neviedla by k splneniu nutričného plánu klienta. Dokonca by klientovi mohli hroziť aj závažné zdravotné problémy.

V našej aplikácii chceme umožniť klientovi, v prípade, že je alergik, vylúčiť z jedálnička tie potraviny, ktoré sa zaraďujú medzi najčastejšie potravinové alergény. Nakoľko v tomto prípade sa značne zúži výber možných potravín, tak sme k ich podpore pristupovali tak, že sme zaradili i niektoré špeciálne potraviny, ako je napríklad bezlepkový chlieb.

Pri výbere potravín sme sa obrátili na teoretický výklad z prvej kapitoly o živinách a popise zásad správnej výživy. Zoznam potravín sa nachádza v prílohe č.1.

2.2 Cieľové skupiny

Prototyp klienta sme vymedzili na základe viacerých skutočností. Osoby v detskom a seniorskom veku, profesionálni športovci majú odlišné celkové energetické a nutričné potreby [1]. Prvým kritériom našej cieľovej skupiny je vek v rozmedzí 18 - 65 rokov - dospelí jedinci.

Ďalej sme obmedzili našu cieľovú skupinu tak, že tuční a chudí ľudia si vyžadujú špecializovanú starostlivosť vzhľadom na vysokú pravdepodobnosť pridružených ochorení. Z hodnôt BMI sme získali druhé kritérium - osoby s BMI v rozsahu 17,1 a 30,0.

Nakoľko je BMI iba pomer váhy a výšky, tak je možné, že používateľ zadá nereálne údaje. Pre elimináciu zadania týchto údajov sme ohraničili výšku na základe referenčných hodnôt výšky dospelých. Najnižšia hodnota je u žien 162,6 centimetrov (cm) a u mužov je najvyššia 179,4 cm [3]. Obe hodnoty sme zaokrúhlili na celé desiatky cm a pre rozšírenie spektra sme posunuli obe hranice o 10 cm. Výsledne rozmedzie je 150-190 cm.

2.3 Nutričné funkcionality aplikačných plánov

Navrhli sme tri nutričné funkcie - pitný režim, pohybový režim a jedálny plán.

2.3.1 Pitný režim

Pre stanovenie správneho množstva pitného režimu sme použili odporúčané denné množstvo tekutín, ktoré sme stanovili takto: 30 ml/kg hmotnosti tela [2]. Vypočítané množstvo vody v litroch sme stanovili na 1 deň.

2.3.2 Pohybový režim

Rýchle chudnutie môže viesť k poškodeniu zdravia, ako je nedostatočný príjem vápnika a železa [1]. Do úvahy sme vzali fakt, že prísne redukčné režimy, by mali prebiehať pod dozorom lekára. Dôležité bolo navrhnúť spôsob *zdravého pomalého* chudnutia.

Používateľ bude prijímať dostatok energie ku svojej TDEE a chudnúť bude z energie vydanej pri pohybe a jeho metabolickom efekte. Spolu s nami navrhnutým jedálnym plánom bude prijímať dostatok bielkovín a tým sme zaručili zdravé, efektívne chudnutie, pretože nebude odbúravaná aktívna svalová hmota, ale nežiadúca tuková.

Pre každý deň aplikácia vyberie druh aktivity a stanoví jej dĺžku. Zoznam aktivít (príloha č.2) sme vyplnili takými športovými aktivitami, o ktorých sme sa domnievali, že sú schopné používatelia zvládnuť.

Mieru chudnutia sme určili počtom percent čistého telesného tuku, ktoré klient postupne schudne pri dodržiavaní plánu. Stanovili sme ju na 1 %, t.j. 1 % z telesnej hmotnosti používateľa vynásobíme hodnotou 32 340 kJ (energia 1 kg ľudského tuku) [4]. Výsledok je celkové množstvo energie, ktoré aplikácia rozpočíta rovnomerne pre každý deň plánu.

2.3.3 Jedálny plán

Zo zásad správnej výživy sme zistili, že máme denne konzumovať 3 hlavné jedlá - raňajky, obed a večeru. K hlavným jedlám môžeme pridať ešte 2 vedľajšie jedlá - desiaty a olovranty - ako medzijedlá. V praxi športovcov je druhá večera ako 6. jedlo dňa. Vedeli sme s akými frekvenciami budeme pracovať a otázkou bolo ako správne zvoliť počet jedál pre špecifického klienta popísaného antropometrickými parametrami.

Smerovali sme k tomu, že pre ľudí, ktorí majú TDEE vyššiu, by pre 3 jedlá denne vychádzali obrovské porcie, ktoré by neboli schopní zjesť. Preto sme frekvenciu jedál rozhodli na základe TDEE. Hodnota 8 400 kJ je priemerná celková denná energetická potreba stanovená v ČR [5]. Od týchto hodnôt sme odvíjali nami zvolené frekvencie. Pre klientov s menšou TDEE ako je priemerná TDEE pripravujeme 3 jedlá denne. Pre klientov s vyššou TDEE ako priemerná TDEE, ale menej než 12 000 kJ (výživná strava) pripravujeme 5 jedál denne. Pre klientov s TDEE viac ako 12 000 kJ a športovým plánom pripravujeme vždy 6 jedál denne. Dôvodom je, že z hľadiska frekvencie príjmu potravy je pri fyzickej námahe výhodnejšie prijímať menšie energetické množstvá v častejších intervaloch [1].

Metriku pre výpočet jedálneho plánu sme zvolili BMR s využitím prediktívnych rovníc podľa Mifflina a St. Jeora [2] (hmotnosť v rovniciach je uvedená v kilogramoch (kg) a výška v cm):

Muži: BMR = (10 x hmotnosť) + (6,25 x výška) - (5 x vek v rokoch) + 5

Ženy: BMR = (10 x hmotnosť) + (6,25 x výška) - (5 x vek v rokoch) - 161

TDEE sme potom stanovili vynásobením BMR hodnotou PAL. Stupne fyzickej náročnosti spolu s koeficientmi sú stanovené takto: sedavá - 1,2; ľahká 1.375; mierna - 1.55 [6].

Minimálny časový rozdiel medzi jednotlivými chodmi závisí od viacerých faktorov: teoretické zásady zdravej výživy, dĺžky trávenia živín, dĺžky spánku, pracovnej aktivity a pohybu.

Časový rozptyl sme stanovili od 5.00 do 22.00. Tento rozsah bol vybraný na základe cirkadiálneho rytmu a našej populácie [7].

2.4 Aplikačné plány

Definovali sme cieľovú skupinu. V rámci tejto skupiny sme rozlíšili medzi typmi klientov z hľadiska BMI a pohybovej aktivity: *klienti s optimálnou hmotnosťou*, *jedinci s nadváhou* a *amatérski športovci*. Každá z týchto podskupín si vyžaduje osobitný prístup, ktorý sme zachytili formou špecializovaných plánov.

Dĺžku plánu sme stanovili na 28 dní, pretože sa domnievame, že je to vhodná dĺžka pre vyhodnotenie efektu a potrebu aktualizovať parameter hmotnosti, aby nedošlo ku skresleniu podávaných plánov.

2.4.1 Optimálny plán

Plán je určený pre ľudí z cieľovej skupiny, ktorí majú normálnu hodnotu BMI a chcú sa naučiť zdravo stravovať. Aplikácia poskytuje pitný režim a jedálny plán. Nutričné hodnoty jedálneho plánu sú nastavené tak, aby udržali stálu telesnú hmotnosť používateľa.

2.4.2 Redukčný plán

Plán je určený pre ľudí z cieľovej skupiny s nadváhou, ktorí chcú zdravo redukovať telesnú hmotnosť. Pre používateľov tohto plánu aplikácia poskytne pitný režim, jedálny plán a dennú pohybovú aktivitu.

2.4.3 Športový plán

Plán je určený pre ľudí z cieľovej skupiny, ktorí pravidelne športujú a chcú si udržať svoj telesný výkon. Aplikácia poskytuje jedálny plán so zvýšenou frekvenciou jedál, pokrývajúci zvýšené energetické potreby a pitný režim.

3. Tvorba jedál

V tejto kapitole budeme analyzovať prístupy k tvorbe jedál, riešiť podmienky uskutočnenia a navrhneme algoritmus optimality živín.

3.1 Štruktúra prípravy jedálnych zostáv

Rozlišujeme viac možností ako zostaviť jedálny lístok. Môžeme uviesť napríklad názov pokrmu (napr. rizoto) alebo názov pokrmu i s presnou receptúrou. My sme v aplikácii vybrali úplne iný spôsob, pretože dovoľuje väčšiu variabilitu technologickej prípravy pokrmov i použitie ľubovoľných požívatín. Náš jedálny lístok obsahuje zoznam potravín s presne určeným množstvom (v gramoch). V aplikácii popisujeme odporúčania vhodných technologických úprav pokrmov a dochutení.

S výberom tohto spôsobu vzniká problém samotnej tvorby jedál. Problém je vymyslieť postup, ktorý nám z množiny všetkých nami vybraných zdravých potravín, vygeneruje zoznam potravín za nasledujúcich podmienok:

1 Zoznam nesmie obsahovať potraviny s extrémnym objemom
2 Zoznam nesmie obsahovať navzájom nevhodné kombinácie potravín
3 Musí spĺňať optimálny pomer živín

3.2 Podmienka objemu zodpovedajúceho realite

Existujú typické veľkostí potravín pre jednu porciu (receptúry), napríklad množstvo hľuzovitých príloh (napr. zemiaky) v uvarenom stave je vhodné obmedziť na maximálne 350 g. Týmto spôsobom sme dolné obmedzenia všetkých potravín nastavili na 3 g z praktických dôvodov (napr. váženie 2 g medu).

3.3 Podmienka kombinácie zodpovedajúcej realite

Potraviny sme do výberu zoskupili podľa hlavného makronutrientu. Výsledné skupiny potravín sú:

- raňajkové vaječné proteíny
- raňajkové mäsové proteíny
- raňajkové mliečne proteíny
- obilninové prílohy
- obilniny

- čisté sacharidy
- čisté tuky
- čisté proteíny
- ovocie
- obedové/večerné proteíny
- zelenina pre hlavné jedlá
- zelenina pre vedľajšie jedlá
- orechy
- oleje
- cestovinové prílohy
- hľuzovité prílohy.

Následne sme definovali množinu vzorov typických skladieb pre jednotlivé chody jedál. Prvky množiny vzoru sme nazvali *komponenty*. Každý komponent reprezentuje skupinu potravín, z ktorej sa bude vyberať potravina pre daný komponent.

Pre splnenie dostatočného množstva vlákniny sme ku každému jedlu pridali 100 g zeleniny (vychádzame z podkapitoly 1.3).

3.4 Návrh algoritmu optimality živín

Úlohou algoritmu je pre vybrané potraviny určiť hmotnosti, aby sme splnili určené množstvá živín a tým dodržali optimálny pomer živín.

3.4.1 Riešenie pomocou lineárnej algebry

Ak si predstavíme hmotnosti potravín jedla ako neznáme a potrebné množstvá živín ako lineárne podmienky, tak problém nadobúda algebraickú povahu. Úlohou je riešiť sústavu rovníc s N neznámymi a 3 lineárnymi podmienkami (sacharidy, tuky, bielkoviny). To je známa úloha z lineárnej algebry, a pretože vektor pravých strán nie je nulový, jedná sa o nehomogénnu sústavu, ktorá *nemusí mať riešenie*. Aplikovaním postupov riešenia lineárnej algebry sme neprišli k riešeniu.

3.4.2 Zoskupenia potravín podľa hlavného makronutrientu

Pre jedlo, ktoré generujeme, vyberáme jeden z jeho vzorov. Vzor obsahuje komponenty, pre ktoré zvolíme potraviny. Smer, ktorý sme si vybrali vychádza z organizácie potravín do skupín. Jedným z kritérií je, že potraviny v každej skupine

majú spoločný hlavný makronutrient. Idea prístupu bude teda nasledovná. Pre každý makronutrient si zoskupíme komponenty s rovnakým hlavným makronutrientom. Potraviny navážime tak, aby sme splnili potrebné množstvá makronutrientov. Zdanlivo sme problém vyriešili, ale ďalšie sa objavili.

Prvým problémom je, že zoskupovanie makronutrientov do skupín môže tvoriť ľubovoľne veľké skupiny. Ak máme v agregácii jednu potravinu, tak ju navážime aby splnila 100 % potrebného množstva. Pri väčšom počte je potrebné stanoviť pomer hmotností jednotlivých potravín, pretože nám môže vzniknúť rozpor s odzrkadľovaním skutočnosti. Napríklad sacharidovú zložku raňajok môžu tvoriť obilniny a ovocie. Typicky je ovocie vnímané ako príloha a majoritnú časť tvorí obilnina. Tento problém vyriešime vhodným určením pomerov v implementácii.

Druhým väčším problémom je, že typicky potraviny neobsahujú iba jeden makronutrient. Napríklad bielkovinové potraviny často obsahujú tuky. V skutočnosti tento fakt môže spôsobiť neexistenciu riešenia pri našej stratégii organizácie potravín vzhľadom k hlavnému makronutrientu. Poďme si to demonštrovať na príklade. Vezmime si nami používaný optimálny pomer živín a potravinu s pomerom tukov a bielkovín blízkemu 1:1 (príkladom takejto potraviny je vajce). Vďaka tomu, že energia z tukov je dvojnásobne väčšia než z bielkovín, tak pri tomto pomere, je pravda, že bude potrebná menšia hmotnosť tukov než bielkovín. Podľa zvolenej stratégie navážime takú hmotnosť potraviny, aby splnila potrebné množstvo bielkovín. Tým sme, avšak presiahli potrebné množstvo tukov. Mohli by sme teraz rozoberať situácie, ktoré budú súvisieť s tukovou zložkou, teda kedy ju budeme riešiť, či pred alebo po bielkovinách. Ak by to bolo predtým, tak tuky presiahneme oveľa viac než čakáme, nakoľko ich presiahneme iba samotnou bielkovinovou potravinou. Ak by to bolo potom, tak by napríklad mohlo nastať, že tukovú potravinu ani nevygenerujeme, pretože vidíme, že už sú splnené. Toto je ešte horší problém, pretože z teoretického úvodu vieme, že potraviny predstavujúce tuky obsahujú viac než len tuky, čo je potrebné pre zdravý chod organizmu. Výsledky týchto úvah závisia, samozrejme, od ďalších rozhodnutí a domnievame sa, že v podstate na nich ani nezáleží pri voľbe tejto stratégie. S optimálnymi rozsahmi nepohneme, pretože sú dané vedou o výžive. S potravinami tiež nepohneme, pretože tie sú dané svojou nutričnou skladbou. Mohli by sme takéto potraviny identifikovať a vymazať z nášho výberu. Domnievame sa, že týmto prístupom by sme odstraňovali zdravé potraviny a

znižovali pestrosť a kvalitu jedál, ba dokonca by sme mohli odstrániť také množstvo potravín, že nebudeme schopní tvoriť jedálne plány vôbec. Týmito úvahami prichádzame do problémov neinformatického charakteru.

3.4.3 Zavedenie jednozložkových potravín

Zatiaľ uvažujeme o potravinách obsahujúcich viac makronutrientov naraz. Vidíme, že iba s týmito potravinami nenájdeme vždy riešenie. Hodilo by sa nám začleniť pre každý makronutrient také nové potraviny, ktoré obsahujú iba danú živinu. Nazvime tieto potraviny *jednozložkové*. Totižto to s čím sme doteraz nevedeli pohnúť je to, že ak prvotne navážime množstvá potravín splňujúce povolenú hranicu makronutrientov (zdola), tak nemusíme ladením hmotností potravín dosiahnúť množstvá živín v stanovenom rozsahu (v prípade excesov). S novými jednozložkovými potravinami už ale k požadovenému riešeniu dospejeme. Ak nastane exces množstva živiny, tak znížime potraviny generujúce túto živinu. Zníženie bude rozumné urobiť rovnakým pomerom ako sú definované, že tvoria daný makronutrient. Týmto spôsobom sa môžeme dostať do situácie, kde niektoré živiny môžu byť splnené a niektoré majú nedostatky. V tomto momente vstupujú do procesu náhodne vybrané jednozložkové potraviny. Navážime ich tak, aby doplnili deficit.

Výhoda tejto heuristiky je, že takto dosiahneme presný optimálny pomer, ktorý chceme a nemusíme uvažovať o rozsahu. To je jeden z hlavných nutričných cieľov práce, pretože týmto spôsobom dokážeme pripraviť individuálne jedálne plány pre ľubovoľného klienta z našej cieľovej skupiny.

Nevýhodou sa môže na prvý pohľad zdať praktická stránka. Čo ak sa takto ku každému jedlu bude musieť pridávať vždy niečo? Prakticky táto situácia nastáva u klientov s veľmi vysokým energetickým príjmom. Musíme si, ale uvedomiť, že toto správanie, nakoľko sa javí ako zvláštne, reflektuje prax.

3.5 Analýza algoritmu optimality živín

V predošlej podkapitole sme sa dozvedeli možný postup riešenia problému. Tento postup sme zosumarizovali a v každom bode prístupu sme rozoberali prípady:

(1) Výber vzoru typickej skladby pre typ jedla na vstupe.

(2) Pre každý komponent vzoru nájsť potravinu.

(3) Pre každý makronutrient vytvoriť skupiny potravín, ktoré majú spoločný hlavný makronutrient.
(4) Pre každý makronutrient stanoviť pomery hmotností ako sa budú podieľať potraviny v skupinách na splnení množstva daného hlavného makronutrientu.
(5) Navážiť hmotnosti potravín tak, aby sme splnili potrebné množstvá živín (zdola). Presahy zhora sú povolené.
(6) Pre každý makronutrient skontrolovať množstvo živín v takto vygenerovanom jedle.
    6.a Ak sme presiahli potrebné množstvo, tak vyberieme skupinu potravín generujúcu daný makronutrient a s pomocou pomeru znížime hmotnosti potravín, aby sme presne splnili potrebné množstvo daného makronutrientu.
    6.b V opačnom prípade pokračujeme ďalej.
(7) Skontrolovať deficity živín.
    7.a V najhoršom prípade, je deficit pre každý makronutrient. Potom sme pre každý deficitný makronutrient vyberali jednozložkovú potravinu, navážili ju tak, aby presne spolu s ostatnými pre daný makronutrient splnila potrebné množstvo.
(8) Skontrolovať minimálne množstvá potravín.
    8.a V prípade, že je hmotnosť menej ako je minimálne povolené množstvo, tak potravinu resp. komponent odstránime.

Rozoberieme si jednotlivé body algoritmu, vo vlastnom odseku, z hľadiska zložitostí a dátových štruktúr.

Každé jedlo obsahuje zoznam vzorov. Vzor je zoznam komponentov. Pre rôzne jedlá a vzory veľkosti zoznamov nie sú fixné. Pre obe použijeme dynamické pole. K výberu vzoru typickej skladby pristupujeme náhodne. Časová zložitosť výberu je *O(1)*, pamäťová zložitosť je *O(V * K)*, kde *V* je počet vzorov, *K* je počet komponentov.

Výber potraviny prebieha pseudo-náhodne. Určitý počet predtým zvolených si zapamätáme a ďalšie z rovnakej skupiny volíme bez nich. Po dosiahnutí zvoleného počtu vyprázdnime zoznam zapamätaných. Každý komponent bude obsahovať rozhranie popisujúce potraviny z databáze. Pri výbere sa do neho uloží konkrétna

potravina. To činí *O(K)* pamäte. Celý výber trvá lineárne vzhľadom k veľkosti skupiny (*VP*). Časová zložitosť je *O(K * VP)*. Pri dotazovaní pracujeme s poľom, ktoré nám bude slúžiť pre zapamätanie aktuálne vybraných potravín. Súčet veľkostí štruktúr je rovná *VP*. Celková pamäťová zložitosť je *O(K + VP)*.

Pre každý makronutrient prejdeme zoznam komponentov a vyberieme z nich také, ktorých potravina obsahuje najviac živiny daného makronutrientu. Časová zložitosť je *O(M * K)*, kde *M* je počet makronutrientov. Založíme si slovník, kde kľúčom bude typ makronutrientu a hodnotou zoznam zvolených komponentov. Pamäťová zložitosť je *O(M * K)*.

Každý komponent musí vedieť v akom pomere je s ostatnými komponentmi, ktoré generujú ten istý makronutrient. Škáluje sa to s počtom komponentov, ktoré sa môžu objaviť v jednej skupine. Pre zapamätanie použijeme slovník, kde kľúčom bude typ komponentu a hodnotou bude pomer. Pamäťová zložitosť je *O(K * K)*. Pre každý makronutrient musíme nastaviť pomer každému komponentu. Časová zložitosť je *O(M * K)*.

Váženie potravín prebehne pre každý komponent. Výsledky si budeme musieť niekde pamätať. Pamäťová a časová zložitosť je *O(K)*.

Pri kontrole nové informácie nevznikajú, takže nepotrebujeme ďalšiu pamäť. Pre každý makronutrient prejdeme každý komponent a v najhoršom prípade budeme musieť znížiť hmotnosti každej potraviny. Časovo nás to výjde na *O(M * K)*.

Opäť skontrolujeme každý makronutrient či nastal nejaký deficit. V najhoršom prípade majú všetky makronutrienty deficit a budeme musieť vybrať a navážiť pre každý z nich. Nech *JP* je počet nových jednozložkových potravín. Nová pamäť zahrnieme tak, že *K* navýšime o *JP*, teda všetky doterajšie zložitosti s *K* budú *K + JP*. Pamäťová zložitosť je *O(K + JP)*, teda pre každý makronutrient si pamätáme novú potravinu. Časovo nás prechod bude stáť *O(M * JP)*.

V poslednom kroku kontrolujeme všetky komponenty a pozeráme sa či hmotnosť potravín nie je pod minimálnou povolenou hmotnosťou. Prechod nás bude stáť *O(K + JP)* času.

Pri zložitostiach môžeme zanedbať *M*, pretože je to vždy konštantné číslo 3. Celková časová zložitosť algoritmu pre jedno jedlo je *O(K(VP + 1) + JP)*. Celková pamäťová zložitosť algoritmu pre jedno jedlo je *O(K*K + VP + JP)*.

4. Aplikačné jadro

Nasledujúca osnova predstavuje problémy, nad ktorými sa musíme zamyslieť a vyriešiť pred samotnou implementáciou webovej aplikácie:

1 Vybrať vhodný typ softvéru
2 Výber webových aplikačných frameworkov
3 Dáta aplikácie
4 Výber databázy
5 Vybrať vhodnú architektúru webovej aplikácie
6 Moderné požiadavky webových aplikácii

4.1 Výber vhodného typu softvéru

Klasifikácia aplikačného softvéru má mnohé rozdielne podoby. Zúženie výberu môžeme docieliť tým, že sa zamyslíme nad platformou, kde bude bežať aplikácia. Môžu byť tri rôzne typy platformových aplikácií:

- *desktopové*
- *mobilné*
- *webové*

Domnievame sa, že každý tvorca softvéru by bol rád, ak by jeho dielo používalo čo najviac ľudí. Z tejto domnienky vychádzajú víťazne webové aplikácie, pretože na ich využívanie nám stačí webový prehliadač, ktorý je dostupný v dnešnej dobe na takmer každom mobilnom telefóne alebo počítači.

Vývoj mobilnej alebo desktopovej aplikácie môže byť viazaný na konkrétnu platformu alebo na konkrétny operačný systém. Webové aplikácie opäť závisia iba od existencie webového prehliadača na ľubovoľnej platforme.

4.2 Výber webových aplikačných frameworkov

Voľba vývoja pomocou webových aplikačných frameworkov (WAF) je ovplyvnená potrebou minimalizácie stráveného času pri implementácii samotnej webovej aplikácie. Väčšinu času strávime pri analýze nutričnej teórie, návrhu algoritmu, výberu dát a podobne. Písanie webových aplikácií sa často zjednodušuje použitím WAF. Tieto frameworky uľahčujú *rýchly vývoj aplikácie (Rapid Application Development)* tým, že umožňujú vývojovému tímu sústrediť sa na tie

časti aplikácie, ktoré sú jedinečné pre ich ciele, bez toho, aby museli riešiť bežné vývojové problémy. WAF môže byť dvojaký, pre serverovú a klientsku časť.

Ako **Server-side WAF** použijeme ASP.NET Core 5 rozširujúci .NET developerskú platformu nástrojmi a knižnicami špecificky pre vývoj webových aplikácií. ASP.NET umožňuje vybudovať full stack webové aplikácie pomocou HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScriptu a jazyku C#. Medzi hlavné súčasti ASP.NET Core 5 frameworku patria:

- *multiplatformový vývoj*
- *Razor* - markup syntax na vkladanie serverového kódu do webových stránok, poskytuje jednoduchú, čistú a ľahkú cestu ako vytvoriť dynamický obsah za pomoci HTML a C#
- *Entity Framework* - populárna knižnica pre dátový prístup, ktorá dovoľuje interagovať so silne typovanými objektmi (veľká časť aplikácie sú dáta), podporované Structured Query Language (SQL) aj non-SQL (Structured Query Language)
- *Bezpečnosť* - podporuje priemyselný štandard autentizačných protokolov, vstavané súčasti pomáhajú chrániť aplikáciu proti cross-site scripting (XSS) a cross-site request forgery (CSRF)
- *Model-View-Controller* (MVC) návrhový vzor pomáhajúci oddeliť používateľské rozhranie (view), data (model) a aplikačnú logiku (controller)

Z pohľadu klientskej časti môžeme veľa času stráviť pri návrhu responzívneho rozloženia webového obsahu. Ako **Client-side WAF** použijeme *Bootstrap* - najpopulárnejší framework pre vytváranie responzívnych rozložení vo webových stránkach. Bootstrap prichádza už s vopred naprogramovanými komponentami pre vytváranie formulárov, tlačidiel, navigácií a podobne. Tieto komponenty, plne pokryjú naše nároky. Navyše je bezproblémovo integrovateľný s ASP.NET Core.

4.3 Dáta aplikácie

V rámci dát aplikácie potrebujeme mať uložené údaje o potravinách, druhoch pohybových aktivít, používateľoch a používateľských plánoch.

O potravine si potrebujeme pamätať názov, zoznam alergénov a hodnoty referenčných množstiev živín na 100 g (prílohy sú uvedené v uvarenom stave) [8, 9, 10]. Potravinovú databázu rozdeľujeme podľa zvolených skupín potravín. Zoznam potravín sa nachádza v prílohe č.1 - Potraviny.

O druhoch pohybových aktivít si potrebujeme pamatať názov a hodnotu kJ na 1 kg telesnej hmotnosti za 1 odcvičenú minútu. Zoznam aktivít sa nachádza v prílohe č.2. [9]

O používateľovi si potrebujeme pamätať profilové údaje, antropometrické parametre, aktuálny jedálny a pohybový plán a aktuálny pitný režim. Po skončení plánu poskytneme používateľovi vyhodnotenie stručným popisom výsledkov. Tieto výsledky uložíme do histórie plánov používateľa.

4.4 Výber vhodnej databázy

Existuje mnoho typov databáz a nás budú zvlášť zaujímať dokumentové databázy. V kóde aplikácie sú dáta často reprezentované ako objekt alebo dokument podobný formátu JavaScript Object Notation (JSON), pretože sú efektívnym a intuitívnym dátovým modelom pre vývojárov. Databázy dokumentov uľahčujú vývojárom ukladanie a dopytovanie údajov v databáze pomocou rovnakého formátu modelu dokumentu, aký používajú v kóde svojej aplikácie. Flexibilná, pološtruktúrovaná a hierarchická povaha dokumentov a databáz dokumentov im umožňuje vyvíjať sa podľa potrieb aplikácií. Model dokumentu funguje dobre s katalógmi, profilmi používateľov a systémami správy obsahu, kde je každý dokument jedinečný a časom sa vyvíja. Amazon DocumentDB (s kompatibilitou MongoDB) a MongoDB sú populárne databázy dokumentov, ktoré poskytujú výkonné a intuitívne API pre flexibilný a iteračný vývoj [11].

Z dvoch spomínaných databáz dokumentov si vyberieme MongoDB. Ukladá všetky údaje do zbierok dokumentov binarného JSON a má preddefinovanú schému, ktorá zase mimoriadne zjednodušuje mapovanie medzi doménovými objektmi a databázou. Veci ako vložené / vnorené objekty a polia vo vnútri doménových objektov sú transparentne uložené v databáze. Je ľahké vykonať významné zmeny aplikácií v reálnom čase bez obáv z prerušenia služby - čo znamená, že vývoj je rýchlejší, integrácia kódu spoľahlivejšia a je potrebných menej času správcu databázy [12].

Pre uloženie dát použijeme MongoDB Atlas, čo je cloudová databáza spravovaná autormi MongoDB. Atlas zvláda všetku zložitosť nasadenia, správy a liečenia nasadení na poskytovateľovi cloudových služieb podľa nášho výberu. Pre potreby práce si vytvoríme Atlas cluster, ktorý bude hostovaný najbližšie pri Prahe.

4.5 Výber architektúry webovej aplikácie

Väčšina tradičných aplikácií .NET sa nasadzuje ako jedna jednotka zodpovedajúca spustiteľnému programu alebo ako jediná webová aplikácia bežiaca v jednej doméne Internet Information Services. Tento prístup je najjednoduchším modelom nasadenia a slúži veľmi dobre mnohým interným a menším verejným aplikáciám. Aj napriek tejto jedinej jednotke nasadenia však väčšina netriviálnych obchodných aplikácií profituje z logického rozdelenia do niekoľkých vrstiev [13].

Najbežnejšia organizácia aplikačnej logiky do vrstiev je: *user interface* (UI), *business logic* (BLL) a *data access layer* (DAL). Pomocou tejto architektúry používatelia zadávajú požiadavky prostredníctvom vrstvy používateľského rozhrania, ktoré interaguje iba s BLL. BLL zase môže volať DAL pre žiadosti o prístup k dátam. Vrstva používateľského rozhrania by nemala priamo odosielať žiadne požiadavky na DAL, ani by nemala interagovať s perzistenciou priamo inými prostriedkami. Rovnako by BLL mala interagovať s perzistenciou iba prechodom cez DAL. Týmto spôsobom má každá vrstva svoju dobre známu zodpovednosť [13].

Jedna z nevýhod tohto tradičného vrstevnatého prístupu je, že kompilačné závislosti idú zhora nadol. To je, že UI vrstva záleží na BLL, ktorá záleží na DAL. To znamená, že BLL, ktorá zvyčajne obsahuje najpodstatnejšiu časť logiky v aplikácii je závislá na implementačných detailoch dátovej vrstvy (často aj na existencii databázy). Testovanie biznis logiky v tejto architektúre je častokrát náročné a vyžaduje si prístup ku databáze. *Dependency inversion* (DI) princíp adresuje tento problém, ktorý uvidíme aplikovaný v nami vybranej architektúre [13].

Aplikácie, ktoré sa riadia princípom DI, známe aj pod názvom Domain-Driven Design princípy, majú tendenciu prichádzať s podobnou architektúrou. Dnes sa pre ňu používajú názvy ako ***Onion architecture*** alebo ***Clean architecture***. Táto architektúra dáva biznis logiku a aplikačný model do centra aplikácie. Namiesto toho, aby sme mali biznis logiku závislú na dátovom prístupe alebo iných infraštrukturálnych záujmoch, je závislosť invertovaná: infraštruktúra a

implementačné detaily záležia na aplikačnom jadre. Táto funkcionalita je dosiahnutá definovaním abstrakcií, alebo rozhraní, v aplikačnom jadre, ktoré sú potom implementované typmi v infraštruktúrnej vrstve [13].

Aplikačné jadro nemá závislosti na ostatných vrstvách. Aplikačné entity a rozhrania sú v samotnom jadre. Zvonku, ale stále v jadre, máme doménové servisy, ktorých typická implementácia je definovaná vo vnútornom kruhu. Mimo jadra, obe vrstvy, UI a infraštruktúra, závisia od jadra, ale nie jedna od druhej (nie nutne) [13].

Vďaka tejto architektúre, UI vrstva pracuje s rozhraniami definovanými v jadre za kompilácie a ideálne nevie o implementačných typoch vrstvy infraštruktúry. Počas behu aplikácie, avšak, sú tieto implementačné typy potrebné pre spustenie aplikácie, takže musia byť prítomné a naviazané na rozhrania v jadre vďaka DI [13].

4.6 Moderné požiadavky webových aplikácií

Responzívny webový dizajn je stratégia poskytovania vhodných rozložení webových stránok pre zariadenia na základe veľkosti okna prehliadača. Kľúčom responzívneho webového dizajnu je jeden HTML dokument pre všetky zariadenia, avšak aplikovanie rôznych štýlov na základe veľkosti obrazovky s cieľom poskytnúť najoptimálnejšie rozloženie pre dané zariadenie [14]. Responzívny dizajn webovej aplikácie nám zaručí Bootstrap (4.2).

Viacjazyčná webová stránka umožňuje webovej stránke osloviť širšie publikum. Internacionalizácia zahŕňa **globalizáciu** a **lokalizáciu**. **Globalizácia** je proces navrhovania aplikácií, ktoré podporujú rôzne kultúry. Globalizácia pridáva podporu pre vstup, zobrazenie a výstup definovanej sady jazykových skriptov, ktoré sa týkajú konkrétnych geografických oblastí. **Lokalizácia** je proces prispôsobovania globalizovanej aplikácie, ktorú sme spracovali pre lokalizovateľnosť, konkrétnej kultúre / miestnemu prostrediu. ASP.NET Core poskytuje služby a middleware na lokalizáciu do rôznych jazykov a kultúr. Lokalizácia aplikácie zahŕňa:

- umožnenie lokalizovateľnosti obsahu aplikácie
- poskytnutie lokalizačných zdrojov pre jazyky a kultúry, ktoré podporujeme
- implementácia stratégie výberu jazyka / kultúry pre každú požiadavku [15].

Bezpečnosť sa netýka iba údajov; pokrýva to veľa aspektov. Nejde len o obmedzenie prístupu na webovú stránku alebo do jej konkrétnych častí; ide o zabránenie nahrávania škodlivého obsahu, ukladanie konfigurácie (a ďalších) dát, čo umožňuje prístup k skriptom konkrétneho pôvodu, a čo je najdôležitejšie, vytvoriť bezpečný kanál pre komunikáciu medzi klientmi a serverom [16].

Logovanie nám hovorí čo systém robí, čo sa chystá urobiť, chyby na ktoré narazil apod. Logovanie je neoddeliteľnou súčasťou .NET Core, ktorá na jeho podporu poskytuje niekoľko abstrakcií; netreba dodávať, že je úplne pripojiteľný a rozšíriteľný. Triedy infraštruktúry, rozhrania, abstraktné základné triedy, výčty atď. sú obsiahnuté v NuGet balíku *Microsoft.Extensions.Logging.Abstractions* a built-in implementácie sú obsiahnuté v balíku *Microsoft.Extensions.Logging* [16].

5. Programátorská dokumentácia

Aplikácia pozostáva z viacerých častí, ktoré spolupracujú. Tieto časti rozdeľujeme podľa zvolenej architektúry do niekoľkých samostatných projektov. Najprv si predstavíme organizáciu kódu pre lepšiu orientáciu a následne si ukážeme dôležité časti kódu.

5.1 Organizácia kódu z pohľadu zvolenej architektúry

Jadro aplikácie drží biznis model zahŕňajúci entity, servisy a rozhrania. Tieto rozhrania obsahujú abstrakcie pre operácie, ktoré budú vykonávané infraštruktúrou, napríklad dátový prístup. Niektoré servisy alebo rozhrania budú musieť pracovať s nie-entitnými typmi, ktoré nemajú závislosti na UI alebo infraštruktúre. Budeme ich definovať ako Data Transfer Objects.

Biznis model našej práce je zložený z viacerých častí. Okrem logík z pohľadu samotnej webovej aplikácie (UI, Infraštruktúra) potrebujeme popísať logiku tvorby a uskladnenia nutričného, cvičebného a pitného režimu. Keďže máme monolitickú aplikáciu, tak nemôžu byť implementované ako samostatné aplikácie. Namiesto toho sa na nich budeme pozerať ako súčasť jadra aplikácie. Chceme pokračovať v princípe separácie záujmov a pre každú logiku vytvoríme vlastný projekt, kde jedinou výnimkou je pitný režim, ten bude implementovaný spolu s výživou.

Logika výživy a pitného režimu sa nachádza v projekte *Core.Nutrition*. Logika cvičebných plánov sa nachádza v projekte *Core.Exercise*. Abstrakcie pre operácie používané vrstvou Infraštruktúry alebo UI (vrátane výživy, pitného režimu a cvičebného plánu) sú v projekte *Core.* Problém, ktorý vznikol je v závislostiach medzi týmito projektami. Chceme aby *Core.** referencovalo *Core* a *Core* optimálne nemalo žiadne referencie. *Core* bude teda spolupracovať s okolitým svetom vďaka inverzii závislosti. Pridáme projekt *Core.Common*, ktorý nebude referencovať nič. Do tohto projektu pridáme spoločné veci pre *Core* a *Core.**. Projekt *Core* bude referencovať len *Core.Common*. Pre každú entitu z *Core.**, ktorú potrebuje *Core* do projektu *Core.Common* pridáme rozhranie, ktoré bude zastupovať túto funkcionalitu. *Core* bude vyžadovať interface (môže, pretože je z *Core.Common*) a volanie bude riešiť *Core.**, ktorá predá potrebnú implementáciu.

Infraštruktúrny projekt typicky zahŕňa implementácie dátového prístupu. Navyše projekt by mal obsahovať implementácie servisov, ktoré musia interagovať s

infraštruktúrnymi záujmami. Tieto servisy by mali implementovať rozhrania definované v aplikačnom jadre, takže projekt infraštruktúry by ho mal referencovať [13].

Projekt *Infrastructure* našej aplikácie predstavuje infraštruktúrny projekt Clean architektúry. Z pohľadu externých záležitostí použijeme NuGet balíček - MongoDB.Driver. Je to oficiálny MongoDB C#/.NET Driver, poskytujúci asynchrónnu interakciu s MongoDB. Okrem vyššie spomenutého obsahu projektu, si budeme musieť napísať vlastné Mongo serializéry a deserializéry abstrakcií entít (rozhrania), ktoré sa ukladajú do databáze.

UI v ASP.NET Core MVC aplikácii je vstupný bod celej aplikácie. Tento projekt by mal referencovať jadro aplikácie a jeho typy by mali interagovať s infraštruktúrou striktne cez rozhrania definované v jadre aplikácie. Žiadne priame inštancie alebo statické volania typov z vrstvy infraštruktúry by nemali byť povolené v UI vrstve. Medzi typy UI vrstvy patria Controllers, Views, Filters, *ViewModels* a Startup. Trieda Startup je zodpovedná za konfiguráciu aplikácie a na zapojenie typov implementácie do rozhraní, čo umožňuje správne fungovanie vkladania závislostí za behu programu. Aby bolo možné prepojiť DI v *ConfigureServices* v súbore Startup.cs projektu používateľského rozhrania, bude možno potrebné, aby projekt odkazoval na projekt Infraštruktúra. Túto závislosť je možné eliminovať najjednoduchšie pomocou vlastného DI kontajnera. Najjednoduchší prístup je umožniť UI projektu odkazovať na projekt infraštruktúry [13].

V našej aplikácii sa vrstva UI nachádza v projekte Web. Okrem typov, projekt bude obsahovať aj statické súbory .css a .js, .resx súbory obsahujúce preklad, obrázkové súbory doplňujúce dizajn stránky.

5.2 Štruktúra dedičnosti nad abstraktnou triedou MealComponent

Pri tvorbe jedál sme si určili, že pre každý chod jedla stanovíme množinu vzorov, z ktorých sa jeden vyberie a podľa neho sa bude výsledné jedlo generovať. Každý zo vzorov pozostáva z komponentov, napríklad raňajky majú za vzor komponenty ako vaječné bielkoviny, pečivo, ovocie a zeleninu. Jedinou vlastnosťou komponentu je najviac zastúpený makronutrient, napríklad pečivo predstavuje najmä sacharidy. Abstraktná trieda **MealComponent** reprezentuje takýto komponent, obsahuje jeden atribút a to majoritný makronutrient.

Komponenty sme rozdelili do dvoch skupín - jednozložkové a viaczložkové. Obe skupiny prispievajú svojou úlohou pri tvorbe jedál a následnej úprave do finálneho tvaru.

Jednozložkové sú tie, ktoré generujú jeden makronutrient a používajú sa pri korekcii optimálneho pomeru živín. Tieto komponenty neobmedzujeme na váhe a zároveň ani neriešime ich pomer oproti ostatným komponentom podieľajúcim sa na generovaní patričného makronutrientu. Z tohto dôvodu odvodzujeme samostatný typ pre každý z nich od abstraktnej triedy **MealComponent**. Jednozložkové komponenty odpovedajú makronutrientom, reprezentujú ich triedy **CleanCarbohydrate**, **CleanFat** a **CleanProtein**.

Naopak, viaczložkové komponenty majú obmedzenie maximálnej váhy, aby sme splnili nutričné odporúčania a čo najviac sa priblížili realite. Zároveň obsahujú atribút popisujúci akým pomerom sa budú podieľať konkrétne potraviny v prípade, ak bude viac potravín generovať ten istý makronutrient. Tieto zmeny reprezentuje trieda **WeightLimitedMealComponent**, dediaca od triedy **MealComponent**. Triedy, ktorých zoznam môžeme nájsť v 2.4 (okrem spomínaných jednozložkových), reprezentujú konkrétne dvojzložkové komponenty. Každá z nich dedí od triedy **WeightLimitedMealComponent**.

5.3 Štruktúra dedičnosti nad abstraktnou triedou MealOccasion

Aplikácia ponúka 6 chodov jedál: raňajky, desiata, obed, olovrant, večera a druhá večera. Každý z chodov sa líši vzormi komponentov, z ktorých môže byť zložený, pomerom voči TDEE a druhom zeleniny. Tieto atribúty sú zachytené v abstraktnej triede **MealOccasion**. Od tejto triedy sú odvodené samostatné triedy pre každý z chodov: **Breakfast**, **MidMorningSnack**, **Lunch**, **AfternoonSnack**, **Dinner** a **Supper**.

Ďalej sme pre každý chod rozlíšili medzi chodmi pre rôzne frekvencie podávania jedál denne. Od každého samostatného typu chodu jedla sú odvodené tri ďalšie. Pre raňajky je hierarchie nasledovná: **ThreeMealsBreakfast**, **FiveMealsBreakfast** a **SixMealsBreakfast**. Analogicky pre každý chod.

5.4 Ingrediencie a ich výber

V jadre aplikácie sme definovali entitu **IIngredient**, ktorá popisuje názov, zoznam alergénov a živiny obsiahnuté na 100g hmotnosti. S touto abstrakciou

pracuje trieda **IngredientPicker**, ktorej jedinou úlohou je vybrať pseudonáhodnú ingredienciu pre zadaný komponent.

5.5 Entity nápojov, jedál, cvičení a ich uloženie

V jadre aplikácie sme definovali entity pre každý typ položky plánov: **IScheduledDrink**, **IScheduledMeal** a **IScheduledExercise**.

Pitný režim je najjednoduchší, **IScheduledDrink** popisuje množstvo vody v litroch, ktoré je potrebné vypiť počas jedného dňa. **IScheduledExercise** popisuje typ cvičenia a jeho dĺžku, ktorú je potrebné odcvičiť pre používateľov redukčného plánu v rámci jedného dňa. Posledná entita, **IScheduledMeal** popisuje obsah jedného chodu a teda typ chodu, zoznam názvov ingrediencií spolu s ich hmotnosťami (g) a množstvá makronutrientov v celom jedle.

Jednotlivé položky plánov sú viazané na dátum a čas. Dátová štruktúra slovníku nám plne stačila pre tento účel, kľúčom je dátum a čas, hodnota je položka plánu. Trieda **Plan<TScheduled>** s generickým parametrom **TScheduled** dedí od slovníka s predtým popísaným kľúčom a hodnotou. Generický parameter sme obmedzili marker rozhraním **IPlanItem**, ktoré implementujú všetky tri položky plánov. Táto trieda teda slúži ako dátová štruktúra pre uloženie položiek plánov s konštantným prístupom ku položke pre zadaný dátum a čas. Definuje metódy, ktoré sa častokrát používajú v implementácii. Tak isto sú definované aj extension metódy, ktoré sa nepoužívajú na vrstve, kde je trieda definovaná, z tohto dôvodu extension a nie inštančné.

5.6 Triedy generujúce jedálny plán

Počas doby výpočtu jedného jedla si potrebujeme pamätať medzivýsledky, napríklad ktoré ingrediencie máme vybrané, koľko aktuálne vážia, koľko je v nich dokopy makronutrientov apod. O tieto dáta spolu s metódami, ktoré sú potrebné pre prácu, napríklad prepočítanie makronutrientov ak sa zmení váha ingrediencie sú implementované v štruktúre **MealsComponentSummary**. Druhou pomocnou štruktúrou pre oddelenie logiky výpočtu je **MacroNutrientsCounter** obsahujúca logiku prepočítavania makronutrientov. Dokáže si pamätať aktuálne množstvá, povolené hraničné obmedzenia a množstvá makronutrientov, ktoré je potrebné splniť.

Hlavný algoritmus sa nachádza v triede **MealComposer**, konkrétne v metóde *Compose*. Táto metóda berie ako jediný parameter typ chodu (**MealOccasion**), ktorý má trieda **MealComposer** pripraviť a vypočítať podľa parametrov používateľa. **MealsComponentSummary** vyberie vzor z množiny vzorov, ktorá mu je poskytnutá v parametri typu chodu. **IngredientPicker** vyberie ingrediencie pre každý komponent vo vzore. K hlavným komponentom pridá trieda **VegetableComponentComposer** zeleninovú prílohu. Následne **MealComposer** vypočíta hmotnosti ingrediencií, aby "sedel" optimálny pomer. Vstupujeme do poslednej fázy, trieda **MealComposerFinalizer** skontroluje či je splnený optimálny pomer. V prípade, že nesedí, zarovná množstvá hmotností na 100 % potrebného množstva a zavolá **CleanComponentComposer**, ktorý doplní to čo je potrebné.

5.7 Trieda StartUp

Táto trieda sa stará o konfiguráciu servisov a pipeline požiadaviek aplikácie. Obsahuje dve metódy *ConfigureServices* a *Configure*.

Metóda *ConfigureServices*, ktorá vykonáva to čo jej názov predstavuje, konfiguruje servisy. Servis je znovupoužiteľný komponent, ktorý poskytuje funkcionalitu aplikácii. Metóda obsahuje konfigurácie nastavení databáze a emailovej komunikácie a všetky servisy od biznis modelu (vytváranie plánov, zobrazovanie plánov, formátovanie kalendára, apod.) cez bezpečnosť (XSS, CSRF) až po MVC.

Metóda *Configure* sa používa na špecifikovanie ako bude aplikácia reagovať na Hypertext Transfer Protocol (HTTP) požiadavky. Pipeline požiadaviek sa konfiguruje pridávaním middleware komponentov do **IApplicationBuilder** inštancie. Hosting vytvára túto inštanciu a priamo ju predáva tejto metóde. Obsahuje middleware pre Hypertext Transfer Protocol Secure presmerovanie, statické súbory, autorizačný a autentifikačný middleware a end-pointové routovanie.

5.8 Areas

Oblasti (*areas*) sa používajú na organizovanie súvisiacej funkcionality. Poskytujú spôsob ako rozdeliť webovú aplikácie do menších funkčných celkov, kde každý z nich má vlastné controllery, views a modely. Webová aplikácia používajúca oblasti musí obsahovať nasledujúce:

- štruktúru priečinkov oblastí

- kontrolery sú popísané atribútom **Area**, ktorý asociuje controller s oblasťou
- pridanie route parametra *area* (obsahuje metóda *Configure* triedy **StartUp**)

Webovú aplikáciu sme rozdelili do piatich oblastí:

- *Account* - akcie s účtom používateľa
- *App* - funkcionalita prihláseného používateľa
- *Errors* - oblasť stoku pre vyskytnuté chyby
- *Main* - funkcionalita neprihláseného používateľa
- *Managment* - správa webovej aplikácie

5.9 Validácia modelov

Namespace **System.ComponentModel.DataAnnotations** obsahuje triedy, atribúty a metódy, ktoré validujú .NET aplikácie. Properties modelov webovej aplikácie sme dekorovali atribútmi práve z tohto namespace. Umožnilo nám to odľahčiť verejné metódy kontrolerov od implementácie validácie.

Medzi často používané atribúty patria:

- obmedzenia dĺžky reťazcov (**StringLengthAttribute**)
- overenie správne zadaného formátu (napríklad **EmailAddressAttribute**)
- **CompareAttribute**, ktorý dokáže porovnať či sa rovnajú dve zadané property (reťazec hesla a reťazec potvrdenie hesla)

Každý z použitých atribútov má nastaviteľnú property *ErrorMessage*, ktorú sme použili pre pomenovanie lokalizovateľnej resource prekladajúcej chybovú správu do jazyka aktuálne nastavenej kultúry (viac v 5.11).

5.10 ViewComponents

Vo views webovej aplikácie sme použili view components, ktoré slúžia ako znovupoužitelné komponenty pre opakujúce sa časti HTML kódu. Výhodné sú v tom, že závisia iba na dátach, ktoré sú poskytnuté pri ich volaní. Vo webovej aplikácii sa nachádzajú v namespace **Application.Web.Views.Shared.Components**.

View component sa skladá z dvoch častí: trieda (dedí od **ViewComponent**, obsahuje logiku v metóde *Invoke*) a výsledok, ktorý vracia (view). Pre zachovanie silnej typovanosti jazyka C# sme použili ViewModel, ktorý predstavuje dátový model výsledku vráteného vo výsledku view component.

5.11 Triedy CalendarFormatter, CsvFormatter a ICalFormatter

Aplikácia poskytuje pohodlnú možnosť preniesť si údaje z našej aplikácie do kalendáru, ktorý prijíma typy súborov Comma-separated values alebo iCalendar. Abstraktná trieda **CalendarFormatter** je základ pre triedy konvertujúce generované plány do udalostí v kalendári. Odvodené triedy **CsvFormatter** a **ICalFormatter** obsahujú implementácie prevodov do príslušných formátov.

Prevod z veľkej časti predstavuje prácu s reťazcami, a preto triedy pracujú s triedou **System.Text.StringBuilder**. Okrem tohto zlepšenia výkonu sa pri prevode používa paralelizácia, jednotlivé vygenerované plány sa môžu konvertovať nezávisle na sebe.

5.12 Trieda LogicProviderSelector a trieda ForPlanAttribute

Servisy definované a implementované v jadre, používané UI pracujú rozdielne pre rôzne aplikačné plány. Aplikácia ponúka tri rôzne aplikačné plány a teda bolo potrebné implementovať tri rôzne logiky pre jednu akciu.

Výber konkrétnej logiky je zaručený pomocou Reflection. Atribút **ForPlan** je používaný na označenie tried nesúcich implementáciu pre konkrétny aplikačný plán. Konštruktor atribútu má jeden formálny parameter, výčtový typ popisujúci typ aplikačného plánu. Trieda **LogicProviderSelector<TBase>** prehľadá príslušnú assembly (Application.Core), zistí odvodené triedy od typu danom generickým parametrom **TBase** a vráti inštanciu triedy konkrétnej logiky.

5.13 Lokalizácia

Aplikácia podporuje dve kultúry, českú (cs-CZ) a slovenskú (sk-SK).

Prvý lokalizačný krok, umožnenie lokalizovateľnosti obsahu aplikácie, sme implementovali pomocou generického rozhrania **IStringLocalizer<T>** (z Microsoft.Extensions.Localization.Abstractions.dll), ktoré využíva triedy **ResourceManager** a **ResourceReader** (obe z System.Resources.dll) pre poskytovanie kultúrne-špecifických prostriedkov (*resources*) za run time. Ďalej pre prostriedky obsahujúce HTML sme použili generické rozhranie **IHtmlLocalizer<T>** (z ASP.NET Core). Tieto dve rozhrania sme použili pre kontrolery, správy o chybách dátových anotácií a Mongo Identity (autentifikačný poskytovateľ).

Vo views sme použili servis **IViewLocalizer** (z ASP.NET Core), ktorý poskytuje lokalizované prostriedky pre view. Nachádza miesto prostriedkov zo súborovej cesty daného view. Nakoľko sa pomocou tohto servisu nedajú používať zdieľané prostriedky, použili sme pre túto potrebu rozhranie **IHtmlLocalizer<T>**.

Druhý lokalizačný krok, poskytnutie lokalizačných zdrojov pre jazyky a kultúry, ktoré podporujeme, sme implementovali špecifikovaním dvoch kultúrnych hodnôt, *SupportedCultures* a *SupportedUICultures* v **StartUp** triede a súbormi prostriedkov (*resource files*).

Objekt **CultureInfo** pre *SupportedCultures* určuje výsledky funkcií závislých od kultúry, ako napríklad formátovanie dátumu, času, počtu a mien. *SupportedUICultures* určuje, ktoré preložené reťazce (zo súborov .resx) vyhľadá **ResourceManager**. **ResourceManager** jednoducho vyhľadá reťazce špecifické pre kultúru, ktoré určuje CurrentUICulture. Každé vlákno v .NET má objekty CurrentCulture a CurrentUICulture. ASP.NET Core kontroluje tieto hodnoty pri renderovaní funkcií závislých od kultúry [15].

Pre prostriedky sme najprv nastavili ich cestu v metóde *ConfigureServices* triedy **StartUp** hodnotou *ResourcesPath* na “Resources”. Sú pomenované celým menom typu okrem mena assembly. Hierarchia priečinku odpovedá štruktúre priečinkov oblastí. Zdieľané prostriedky sú umiestnené na najvyššej hladine priečinku Resources.

Tretí a posledný lokalizačný krok, implementácia stratégie výberu jazyka / kultúry pre každú požiadavku, sme implementovali programovým nastavením kultúry. Aplikácia umožňuje používateľovi vybrať si kultúru pomocou tlačidla umiestneného v záhlaví stránky. Využili sme, že aplikácie v ASP.NET Core poskytujú mechanizmus nastavenia kultúry cez cookie. Kód zodpovedný za vykonanie tejto akcie (**CultureController.SetCulture** metóda v oblasti Managment) priloží do HTTP response cookie s nastavením kultúry, cookie expiruje po 1 roku.

5.14 Trieda ErrorFilter

Handluje ľubovoľnú neošetrenú výnimku, ktorá nastane počas exekúcie akcie controlleru alebo iného filtra. Metóda *OnException* loguje správu výnimky a presmeruje výsledok akcie do všeobecného stoku webovej aplikácie, kde oznamuje používateľovi, že nastala neočakávaná chyba. Filter je registrovaný globálne.

5.15 Logovanie

Vo webovej aplikácii sme zaregistrovali logovacie servisy zavolaním metódy *AddMvc* v metóde *ConfigureServices* triedy **StartUp**. Pre logovanie potrebujeme inštanciu rozhrania **ILogger** alebo **ILogger<T>**. Logovanie sme implementovali v public metódach controllerov a triede **ErrorFilter** injektovaním inštancie vyššie spomínaných rozhraní za pomoci DI. Využili sme viacero logovacích levelov (Debug, Information, Warning, Error, Critical).

Pre redukovanie opakovania sa kódu, ktorý loguje vstup do akcie sme pripravili atribút **LogAttribute**. Táto trieda vypíše log levelu Information obsahujúci HTTP verb, oblasť, controller a názov akcie. Týmto spôsobom sa vieme lepšie orientovať.

Logovanie sme implementovali veľmi jednoducho, cieľom bolo popísať všetky cesty exekúcie kódu, exceptions a chyby a pripraviť aplikáciu pre logovanie v prípade rozšírenia.

5.16 Bezpečnosť

Jedným z aspektov bezpečnosti webovej aplikácie je autentifikácia a autorizácia, pretože poskytujeme prostriedky, ktoré si vyžadujú autorizáciu pre prístup ku nim. Také prostriedky (oblasť *App*) sme označili pomocou ASP.NET Core atribútu **AuthorizeAttribute**.

Ako autentifikačného poskytovateľa sme vybrali Mongo Identity. Identity je odporúčaná priamo Microsoftom a Mongo vyplýva z výberu databáze. Pre použitie Identity sme najprv zaregistrovali servisy a prepísali default error describera, aby sme mohli lokalizovať chybové správy. Nastavili sme lockout pri neúspešných zadaniach hesla, potvrdenie emailovej adresy zadanej pri registrácií pred prvým prihlásením, prepísali cestu kde framework presmeruje používateľa v prípade, že nie je autorizovaný. Obmedzenia hesla sme prebrali pôvodné z Mongo Identity.

Definovali sme entitu, v jadre aplikácie, ako má vyzerať používateľ - **IApplicationUser**. Všetky citlivé údaje (properties v tomto prípade) sme označili **PersonalDataAttribute**. Je to požiadavka, ktorý automaticky sprístupní údaje pre uloženie a vymazanie (General Data Protection Regulation - GDPR). Používateľovi sme umožnili vo webovej aplikácii nahliadnuť na svoje dáta alebo si ich odoslať na

emailovú adresu zadanú pre registráciu. Cieľom tohto bolo poskytnúť základ pre plné rozšírenie pre GDPR podporu.

Ďalším aspektom bezpečnosti bolo implementovať podporu pre ochranu proti **CSRF** útokom pomocou ktorých je používateľ podvedený ku vykonaniu nejakej akcie na jednej zo stránok, na ktorých je prihlásený. Používame antiforgery balík **Microsoft.AspNetCore.Antiforgery**. Tento framework generuje skrytý field s tokenom pre každý formulár, posiela cookie s rovnakým tokenom a overí ich identitu. Najprv sme zaregistrovali servis pomocou metódy *AddAntiforgery*. Pri každom formulári, validovanom na server-side sme použili metódu *BeginForm*, ktorá defaultne vypisuje token, keď produkuje <form> tag. Automatickú kontrolu na server-side sme zaručili globálnym filtrom **AutoValidateAntiForgeryTokenAttribute**. Pre Asynchronous JavaScript and XML (AJAX) požiadavky pridávame do hlavičky token, ktorý ASP.NET Core framework zachytí.

5.17 Kalendár udalostí

Najväčšou javascriptovou súčasťou webovej aplikácie je kalendár udalostí. Kalendár ponúka tri možnosti zobrazenia - mesačný, týždenný, zoznam udalostí. Abstraktná trieda **CalendarView** je základom pre každé z týchto zobrazení; **MonthView**, **WeekView** a **ListView**. Metóda *create* vytvára prvky, ktoré sa zobrazia a *render* nastavuje funkcionalitu a zobrazuje nastavené prvky. Kalendár obsahuje klikateľné ikony položiek pitného režimu, cvičebného plánu a jedálneho plánu. Zobrazovanie položiek, posúvanie dopredu/dozadu v rámci jedného pohľadu a prepínanie medzi pohľadmi sme implementovali pomocou AJAXu. Pre generovanie HTML obsahu sme použili knižnicu jQuery.

Vo webovej aplikácii sme pracovali s týmito dvomi technológiami častejšie. Výhodné pre nás bolo, že AJAX poskytuje bohatšiu používateľskú skúsenosť a znižuje množstvo prenosu, jQuery zjednodušuje prácu s HTML kódom a AJAXom.

5.18 Emailová komunikácia

Aplikácia posiela uvítaciu správu po registrácii, je potrebné potvrdiť emailovú adresu po registrácii pred prvým prihlásením alebo jej zmene a posielajú sa na ňu citlivé údaje. Pre posielanie emailov v ASP.NET Core sme použili NuGet balíčky *MailKit* a *MimeKit*, populárne emailové frameworky v .NET. Simple Mail

Transfer Protocol (SMTP) nastavenia sú v konfiguračnom súbore *appsettings.json*. Pre potreby aplikácie sme použili Ethereal - fake SMTP servis. Správy sa nikdy neodošlú na zadanú emailovú adresu, naopak, vrátia sa do nastavenej emailovej schránky. Toto riešenie nám postačí pre potreby aplikácie.

6. Používateľská dokumentácia

V tejto kapitole adresujeme slová používateľovi. Dozvie sa ako aplikáciu nainštalovať a používať.

6.1 Manuál pre inštaláciu

Aplikácia je nasaditeľná na operačných systémoch s Docker clientom podporujúcim linuxové kontajnery (linux/amd64). Navyše, pre nasadenie, je potrebné mať .NET Command Line Interface (v .NET 5 Software Development Kit). Vďaka nemu si vygenerujeme self-signed certifikát kvôli HTTPS. Postup pre nasadenie na operačnom systéme Windows a s podporou virtualizácie sa nachádza v prílohe č.3. Používateľovi stačí internetový prehliadač.

6.2 Návod na použitie

V tejto podkapitole sa používateľ zistí ako a aké akcie môže v aplikácii vykonávať. Domovská stránka je počiatočné miesto, kde môže používateľ začať. Obsahuje stručný štvorkrokový priebeh aplikácie od vytvorenia účtu až po ukončenie.

6.2.1 Pre koho je aplikácia vhodná?

Aplikácia je určená pre osoby s *nadváhou* (BMI <= 30), *amatérskych športovcov* a tých, ktorí sú spokojní so svojou váhou, ale nie so svojou nutričnou životosprávou. Ďalšími podmienkami sú vek (18 - 65 rokov) a výška (150 - 190 cm). Jedálny plán je prispôsobený potrebám alergikov (lepok, kôrovce, arašidy, zeler, horčica, sezam, oxid siričitý a siričitany a mäkkýše).

6.2.2 Prehľad aplikačných plánov

Ponuka plánov sa zobrazí po kliknutí na *Plány* v navigačnom paneli na domovskej stránke. Každý plán obsahuje podmienky pre výber, popis a náučné informácie o nosných pilieroch plánu.

Optimálny plán je vhodný pre ľudí, ktorí by sa radi naučili ako sa zdravo stravovať. Predpokladom pre používanie je BMI v rozsahu normálnej telesnej hmotnosti. Používateľom bude pripravený jedálny plán (3 alebo 5 jedál denne) a denný pitný režim.

Športový plán je vhodný pre amatérskych športovcov. Má za cieľ udržanie športového výkonu dodávaním dostatočného množstva živín. Aplikácia poskytuje jedálny plán s frekvenciou jedál 6 x denne (pridaná druhá večera) a denný pitný režim.

Redukčný plán je pre osoby, ktoré by radi schudli *zdravo* a bez použitia diét. Používateľom bude pripravený jedálny plán, denný pitný režim a cvičenie primeranej fyzickej aktivity na každý deň. Po úspešnom a dôkladne dodržiavanom pláne by mala nastať redukcia čistej tukovej hmoty vo výške 1 % z telesnej hmotnosti.

Dĺžka každého plánu je stanovená na 28 dní, čo činí presne 4 týždne.

6.2.3 Výber aplikačného plánu a pomoc s výberom

Pre zistenie vhodného plánu je pripravená BMI kalkulačka. Po kliknutí na *BMI kalkulačka* v navigačnom paneli na domovskej stránke sa zobrazí formulár. Formulár požaduje po používateľovi údaje o telesnej výške, váhe a odpoveď áno/nie či športuje pravidelne. Aplikácia oznámi používateľovi hodnotu BMI a zoznam vhodných aplikačných plánov pre zadané údaje. V prípade, že žiaden plán nie je pre používateľa vhodný, aplikácia mu to oznámi znakom '-'. Formulár oznamuje a limituje používateľa kritériami spomenutými v podkapitole 6.2.1.

Výber plánu je možný dvomi spôsobmi. Prvý spôsob je cez tlačidlo *Vybrať*, na stránke prehľadov plánov. Po kliknutí si aplikácia zapamätá výber plánu na 1 hodinu. Odporúčame používateľom, ktorí vedia vopred výber svojho plánu. Druhý spôsob je najprv úspešné splnenie registrácie a následný výber plánu po prihlásení.

Ak parametre nesedia pre vybraný plán, výber sa stornuje. Používateľ bude presmerovaný na stránku, kde si môže vybrať iný plán.

6.2.4 Registrácia

Prístup k registrácii môže byť dvojaký. Prvou možnosťou je kliknúť na tlačidlo *Vybrať*, na stránke prehľadov plánu. Po kliknutí na tlačidlo je používateľ presmerovaný na registračnú stránku. Druhou možnosťou je kliknúť na ikonu osoby v navigačnom paneli domovskej stránky. Potom, opäť v navigačnom paneli, kliknúť na *Registrovať sa*. Možnosti výberu nie sú rovnaké (viz 6.2.2).

Údaje požadované pre registráciu sú: krstné meno, priezvisko, používateľské meno, emailová adresa, heslo a jeho potvrdenie. Každý údaj je povinný. Obmedzenia pre jednotlivé údaje sú nasledujúce:

- minimálny počet znakov krstného mena a priezviska je 2 a maximálny 20
- minimálny počet znakov hesla je 6 a maximálny 20
- minimálny počet znakov používateľského mena je 1 a maximálny 20
- heslo musí obsahovať číslicu, nie alfanumerický znak, lowercase a uppercase znak
- minimálny počet unikátnych znakov hesla musí byť 1
- emailová adresa musí byť vo validnom formáte

Po registrácii aplikácia odošle na emailovú adresu uvítaciu správu a odkaz pre potvrdenie emailovej adresy.

6.2.5 Prihlásenie, obnovenie hesla

Používateľ sa môže prihlásiť do účtu kliknutím na ikonu osoby v navigačnom paneli domovskej stránky.

Prihlasovacie údaje sú používateľské meno a heslo zadané pri registrácii. Prvé prihlásenie a každé prvé prihlásenie po zmene emailovej adresy si vyžaduje jej potvrdenie.

Po troch neúspešných pokusoch sa prihlásenie uzamkne na 5 minút. Po uplynutí doby je možné znovu sa prihlásiť.

V prípade, že používateľ zabudol heslo je potrebné kliknúť na odkaz pod heslom v prihlasovacom formulári s textom *Zabudli ste heslo?*. Používateľ bude presmerovaný na stránku, kde napíše emailovú adresu zadanú pri registrácii. Do emailovej schránky bude odoslaný odkaz pre obnovenie hesla. Odkaz presmeruje používateľa na formulár, kde si môže nastaviť nové heslo.

6.2.6 Prvé prihlásenie

Čerstvo registrovaný používateľ sa po prihlásení do aplikácie môže stretnúť s jednou z možností: výber plánu alebo zadanie profilových údajov. Údaje požadované pre vypočítanie plánu sú: vek, pohlavie, výška (cm), váha (kg), úroveň fyzickej aktivity a alergény. Aplikácia používateľa upozorní na zadané údaje mimo

povoleného rozsahu (6.2.1). Plán začína nasledujúci kalendárny deň. V prípade nevhodných údajov pre zvolený plán, bude používateľ presmerovaný na stránku, kde si bude môcť znovu vybrať iný plán.

6.2.7 Domovská stránka účtu

Obsah domovskej stránky vyplýva z využívania plánu. Ak používateľ:

- nevyužíva služby plánu, je obsahom domovskej stránky výber plánu
- v ten istý deň začal plán, je obsahom domovskej stránky informácia, že plán začína nasledujúci kalendárny deň
- využíva služby plánu a je iný deň než prvý, je obsahom stránky prehľad daného dňa.

Prehľad obsahuje dátum, pitný režim, cvičenie [redukčný plán] a chody jedál. V horizontálnom výbere sa vie navigovať pomocou tlačidiel umiestnených po boku. Po kliknutí na tlačidlo > pri niektorom z chodov sa zobrazia dve tabuľky. Prvá obsahuje zoznam ingrediencií s ich hmotnosťami (g). Druhá tabuľka obsahuje zoznam makronutrientov s ich hmotnosťami (g), ktoré sú obsiahnuté v danom jedle.

6.2.6 Profil

Po kliknutí na *Profil* v navigačnom paneli účtu sa zobrazí používateľovi prehľad jeho osobných údajov a údaje o aktuálnom pláne. Medzi zobrazené profilové údaje patrí krstné meno a priezvisko, výška (cm), váha (kg), pohlavie a vek. Medzi zobrazené údaje o aktuálnom pláne patria typ plánu a BMI používateľa zo dňa výberu plánu.

6.2.7 Kalendár udalostí, ikony a formátovanie

Po kliknutí na *Kalendár* v navigačnom paneli účtu sa zobrazí používateľovi navigačný panel kalendára, kalendár samotný a formulár pre export kalendára (v poradí pod sebou). Obsahy navigačného panelu a samotného kalendára sa líšia podľa typu zobrazenia a podľa veľkosti obrazovky.

Navigačný panel obsahuje informatívny popis a šesť tlačidiel rozdelených do dvoch skupín. Informatívny popis slúži pre orientáciu v aktuálnom zobrazení. V mesačnom náhľade obsahuje názov práve zobrazeného mesiaca a rok. V týždennom náhľade je to deň kedy začína práve zobrazený týždeň a deň kedy končí. V zoznamovom náhľade je to rovnaké ako pre týždeň, iba rozsah nie je práve jeden

týždeň, ale deväť dní. Prvá skupina tlačidiel obsahuje tlačidlá pre posun dozadu, návrat na dnešný deň a posun dopredu (v poradí). Druhá skupina tlačidiel mení pohľad, mesačný, týždenný a zoznam (v poradí).

Kalendár udalostí slúži pre prehľad aktuálneho plánu. Obsah kalendára je rozdelený do *jednotiek* predstavujúcich jednotlivé dni v mesačnom zobrazení, jednotlivé hodiny v týždennom zobrazení a jednotlivé dni v zozname. Každá jednotka môže byť buď prázdna, ak používateľ nemá v daný deň nič naplánované, alebo môže obsahovať ikony predstavujúce naplánovaný pitný režim, naplánované jedlo alebo cvičenie. Každá ikona je klikateľná a po kliknutí na ňu sa používateľovi zobrazí obsah plánovanej položky.

Ikona pitného režimu informuje používateľa o tom koľko litrov vody má vypiť vo zvolený deň. Ikona cvičenia informuje používateľa o type a dĺžke cvičenia, ktoré by mal v priebehu zvoleného dňa odcvičiť (neprerušovane). Ikona jedla informuje používateľa o naplánovanom jedle. V mesačnom zobrazení vypíše všetky naplánované jedlá v daný deň. V ostatných dvoch zobrazeniach informuje o zvolenom jednom jedle. Obsah výpisu je typ chodu, čas a zoznam ingrediencií s hmotnosťami (g).

Týždenné zobrazenie je, na rozdiel od mesačného a zoznamu, organizované podľa hodín. Prvý riadok (nemá pridelenú hodinu) obsahuje ikony pre celodenné naplánované položky (pitný režim a cvičenie). Rozsah hodín je od 5.00 - 21.00. Naplánované položky neberú ohľad na minúty, teda raňajky naplánované na 6.30 sa zobrazia v riadku s hodinovým popisom 6.00.

Formulár pre export kalendára udalostí je umiestnený pod samotným kalendárom. Obsahuje jednu položku a to formát exportovaného kalendára. Používateľ má na výber dve možnosti: csv a icalendar. Po vybratí a kliknutí na tlačidlo *Vybrať* sa iniciuje stiahnutie súboru. Stiahnutý súbor je pripravený na import do third-party aplikácie.

6.2.8 Nákupný zoznam

Používateľ má možnosť vypísať zoznam ingrediencií pre jedlá v stanovenom rozsahu. Po kliknutí na *Nákupný zoznam* v navigačnom paneli účtu sa používateľovi zobrazí formulár so štyrmi položkami. Tieto položky ohraničujú výber dní a jedál, z

ktorých sa bude generovať nákupný zoznam. Aplikácia používateľa upozorní pri chybne zadaných údajoch.

6.2.9 História plánov používateľa

Stránka, ktorá sa zobrazí používateľovi po kliknutí na *História* v navigačnom paneli účtu, obsahuje prehľad všetkých doposiaľ úspešne ukončených plánov.

Prehľad o ukončenom pláne obsahuje dátum začiatku a dátum ukončenia, typ plánu, zaujímavé štatistické údaje a hodnoty BMI na začiatku a na konci plánu. Štatistické údaje sú počet zjedených jedál, celkové obsiahnuté kilojouly vo všetkých zjedených jedlách a počet odcvičených minút počas celej dĺžky plánu.

6.2.10 Nastavenia účtu

Prehľad možností nastavenia a akcií s účtom používateľa nájde používateľ po kliknutí na *Nastavenia* v navigačnom paneli účtu.

Zmeniť profilové údaje umožňuje používateľovi zmeniť krstné meno, priezvisko, používateľské meno a emailovú adresu. Po zmene údajov okrem emailovej adresy ostane používateľ prihlásený. Pri zmene emailovej adresy je používateľ odhlásený a je potrebné potvrdiť novú emailovú adresu odkazom zaslaním do jej schránky.

Zmeniť heslo si vyžaduje zadať aktuálne heslo, nové heslo a jeho potvrdenie. Po zmene používateľ ostane prihlásený.

Zmeniť hodiny jedál je obmedzené časovým rozmedzím od 5.00 po 21.59. Je nutné dodržať 90 minútový rozostup. Po zmene sú údaje aktualizované pre všetky položky.

Zmeniť plán je možné iba pre tie plány, ktoré zodpovedajú údajom zadaným po výbere plánu. Táto funkcionalita primárne slúži pre chybný výber plánu, než pre náhodné zmeny plány počas nejakého plánu.

Poslať osobné údaje zašle používateľovi citlivé údaje zapamätané aplikáciou do emailovej schránky. Formát údajov je JSON.

Zmazanie účtu vymaže *všetky* údaje o používateľovi s okamžitou platnosťou.

6.2.11 Tipy

Tipy nájde používateľ kliknutím na *Tipy* v navigačnom paneli účtu. Poskytuje informácie o odporúčaniach tepelnej úpravy a dochutení pokrmov.

6.2.12 Ukončenie plánu a jeho vyhodnotenie, pokračovanie

Po uplynutí doby trvania plánu sa na domovskej stránke používateľa zobrazí formulár. Požadované údaje sú váha (kg) a výška (cm). Používateľ bude po zadaní údajov a po kliknutí na tlačidlo *Vyhodnotiť* presmerovaný na štatistickú stránku. Štatistika obsahuje rovnaké údaje ako boli spomenuté v 6.2.9. Po kliknutí na tlačidlo *Ukončiť plán* bude používateľ presmerovaný na stránku výberu plánu a výsledok bude uložený do histórie.

Používateľ sa môže rozhodnúť pokračovať a znovu si vybrať plán.

6.2.13 Zmena jazyka a kultúry

Aplikácia je lokalizovaná pre český a slovenský jazyk. Zmenu jazyka a kultúry je možné vykonať kliknutím na tlačidlo s vlajkou, ktoré sa nachádza v každom navigačnom paneli. Táto zmena vyžaduje pridanie cookie s dĺžkou trvania 1 rok.

6.2.14 Emailová schránka

Návod prístupu do emailovej schránky sa nachádza v prílohe č.4.

Závěr

V práci sme sa venovali implementácii nutričnej webovej aplikácie, ktorá podporuje zdravé stravovanie, zdravé chudnutie, pitný režim a pohybovú aktivitu. Vybrali sme cieľovú skupinu, pre ktorú sme nachystali trojicu plánov. Navrhli sme algoritmus tvorby jedál, ktorý vytvorí receptúru z výlučne zdravých potravín a splňuje optimálny pomer živín daný zásadami správnej výživy. Webová aplikácia splňuje moderné požiadavky ako responzivita, viacjazyčnosť, bezpečnosť a nápomocné používateľské rozhranie.

Práca splnila ciele, avšak kvalita jej výsledku je do značnej miery ovplyvnená charakterovými vlastnosťami používateľa a výživovou politikou štátu.

Predpokladáme, že môžeme nahradiť tradičný spôsob riešenia s výnimkou patologických stavov (dohľad lekára). Výsledok práce je, že aplikácia je jednoduchá, relevantná, spoľahlivá, šetrí čas a celý proces výrazne zjednodušuje.

Ideálnym rozšírením do budúcnosti by bol širší sortiment potravín (funkčné potraviny), zohľadnenie mikronutrientov, internacionalizácia aplikácie pre viacero kultúr. Ďalšie vhodné rozšírenie pre používateľa by bolo pridať viac edukatívneho obsahu, receptúry s fotografiami a sociálnu sieť pre zdieľanie výsledkov čím by sa podporila interná motivácia.

Seznam použité literatury

[1] BEŇO, Igor. *Náuka o výžive: Fyziologická a liečebná výživa*. 2. vydanie. Martin : Vydavateľstvo Osveta, spol s. r. o., 2003. ISBN 80-8063-126-3.

[2] Metodické doporučení pro zajištění stravy a nutriční péče. Praha : Věstník Ministerstva zdravotnictví ČR, ročník 2020, částka 10, zo dňa 30. 9. 2020. Dostupné na: https://www.mzcr.cz/wp-content/uploads/2020/09/V%C4%9Bstn%C3%ADk-MZ-10-20.pdf

[3] Německá společnost pro výživu (DGE), Rakouská společnost pro výživu (ÖGE), Švýcarská společnost pro výzkum výživy (SGE), Švýcarská společnost pro výživu (SVE). *Referenční hodnoty pro příjem živin*. 2. vydanie. Praha : Výživaservis s.r.o., 2018. ISBN 978-80-906659-3-4.

[4] Kolik energie obsahuje kilogram tuku a co je třeba udělat pro jeho spálení? | Aktin; https://aktin.cz/kolik-energie-obsahuje-kilogram-tuku-a-co-je-treba-udelat-pro-jeho-spaleni [2021.07.17]

[5] Nařízení evropského parlamentu a rady (EU) č. 1169/2011 ze dne 25. října 2011 o poskytování informací o potravinách spotřebitelům, o změně nařízení Evropského parlamentu a Rady (ES) č. 1924/2006 a (ES) č. 1925/2006 a o zrušení směrnice Komise 87/250/EHS, směrnice Rady 90/496/EHS, směrnice Komise 1999/10/ES, směrnice Evropského parlamentu a Rady 2000/13/ES, směrnic Komise 2002/67/ES a 2008/5/ES a nařízení Komise (ES) č. 608/2004. Lucemburk: Úřední věstník Evropské unie, L 304/18, zo dňa 22.11.2011. Dostupné na: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:304:0018:0063:CS:PDF

[6] Physical activity level (PAL) guidelines | Actabit; https://www.actabit.com/physical-activity-levels-pal/ [2021.07.17]

[7] WALKER, Matthew. *Proč spíme: Odhalte sílu spánku a snění*. 1. vydanie. Brno : Jan Melvil Publishing, 2018. ISBN 978-80-7555-050-7.

[8] Pšeno - proso varené | Nutričné hodnoty | Food is good; http://www.foodisgood.eu/sk/potraviny/obilne-vyrobky/ostatne/pseno-proso-varene.html [2021.04.10]

[9] Kalorické tabulky. Dostupné na: https://www.kaloricketabulky.cz/

[10] Kalorické tabuľky. Dostupné na: https://www.kaloricketabulky.sk/

[11] What is NoSQL? | Nonrelational Databases, Flexible Schema Data Models | AWS; https://aws.amazon.com/nosql/ [2021.06.22]

[12] Why Choose MongoDB and When? | Linkedin; https://www.linkedin.com/pulse/why-choose-mongodb-when-niraj-kumar [2021.06.22]

[13] Common web application architectures | Microsoft; https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures [2021.06.19]

[14] NIEDERST ROBBINS, Jennifer. *Learning Web Design: A Beginner's Guide to HTML, CSS, Javascript, and Web Graphics*. 5. vydanie. Sebastopol : O'Reilly Media, Inc., 2018. ISBN 978-1-491-96020-2.

[15] Globalization and localization in ASP.NET Core | Microsoft; https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization?view=aspnetcore-5.0 [2021.07.09]

[16] PERES, Ricardo. *Modern Web Development with ASP.NET Core 3*. 2. vydanie. Birmingham : Packt Publishing Ltd., 2020. ISBN 978-1-78961-976-8.

Seznam použitých zkratek

AJAX - Asynchronous JavaScript and XML

BLL - Business Logic Layer

BMI - Body Mass Index

BMR - Basal Metabolic Rate

cm - centimeter

CSRF - Cross-site request forgery

CSS - Cascading Style Sheets

DAL - Data Access Layer

DI - Dependency Inversion

g - gram

GDPR - General Data Protection Regulation

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

kcal - kilokalória

kg - kilogram

kJ - kilojoule

JSON - JavaScript Object Notation

MVC - Model-View-Controller

PAL - Physical Activity Level

SMTP - Simple Mail Transfer Protocol

SQL - Structured Query Language

TDEE - Total Daily Energy Expenditure

UI - User Interface

WAF - Web Application Framework

XSS - Cross-site scripting

Přílohy

1. Ingredients.zip - obsahuje JSON súbory popisujúce prvky jedál
2. Exercises.zip - obsahuje JSON súbor so zoznamom pohybových aktivít
3. Deployment.md - návod ako deploynúť webový server cez Docker s HTTPS
4. Ethereal.md - návod ako sa prihlásiť do emailovej schránky