UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Klára Pantůčková

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Aplikace pro doporučování jídel na základě nutriční bilance uživatele

Bakalářská práce

2025 Klára Pantůčková

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Klára Pantůčková**
Osobní číslo:	**I22126**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace pro doporučování jídel na základě nutriční bilance uživatele**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce bude navrhnout a implementovat aplikaci v jazyce C#, která na základě záznamů o konzumovaných jídlech doporučí uživateli další pokrmy s chybějícími živinami, čímž pomůže naplnit jeho nutriční cíle. Aplikace bude sledovat denní příjem kalorií a makroživin podle uživatelem stanovených hodnot a nabízet personalizované návrhy receptů. Teoretická část by měla zahrnovat rešerši metod a algoritmů pro doporučování a srovnání podobných aplikací. Praktická část se zaměří na implementaci aplikace, její architekturu, použité algoritmy a strukturu databáze. Výsledná aplikace by měla být otestována a zhodnocena z hlediska funkčnosti, přesnosti doporučení a možností dalšího rozvoje.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

PETR, Pavel. Data Mining. Vyd. 3. Pardubice: Univerzita Pardubice, 2010-. ISBN 978-80-7395-325-6.
AGGARWAL, Charu C. Recommender systems: the textbook. Cham: Springer, [2016]. ISBN 978-3-319--29657-9.
GRIFFITHS, Ian. Programming C# 12: Build Cloud, Web, and Desktop Applications. O'Reilly Media, 2024. ISBN 978-1098158361.

Vedoucí bakalářské práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

Prohlašuji:

Práci s názvem Aplikace pro doporučování jídel na základě nutriční bilance uživatele jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 14. 05. 2025

Klára Pantůčková v.r.

PODĚKOVÁNÍ

Ráda bych poděkovala svému vedoucímu bakalářské práce, Ing. Janu Mertovi, Ph.D, za cenné rady, odborné vedení a trpělivost, kterou mi po celou dobu zpracování této práce věnoval.

Dále bych chtěla poděkovat své rodině, která mě po celou dobu studia podporovala, motivovala a vytvářela mi podmínky pro úspěšné dokončení mého studia.

ANOTACE

Cílem této bakalářské práce je návrh a implementace aplikace v jazyce C#, která na základě záznamů o konzumovaných jídlech poskytuje uživateli personalizovaná doporučení dalších pokrmů s ohledem na chybějící živiny. Aplikace sleduje denní příjem kalorií a makroživin ve vztahu k uživatelem stanoveným nutričním cílům a navrhuje recepty, které pomáhají tyto cíle naplnit.

Teoretická část práce se zaměřuje na rešerši existujících metod a algoritmů pro doporučování jídel a srovnání podobných aplikací. Praktická část popisuje proces návrhu a implementace aplikace, včetně architektury systému, použitých algoritmů a struktury databáze. Výsledná aplikace je otestována z hlediska funkčnosti, přesnosti doporučení a jsou navrženy možnosti jejího dalšího rozvoje.

KLÍČOVÁ SLOVA

C#, ASP.NET, doporučovací systémy, optimalizace, simulované žíhání

TITLE

Design and Implementation of a C# Application for Personalized Meal Recommendations Based on Consumed Food Records

ANNOTATION

The aim of this bachelor thesis is to design and implement an application in C# that provides personalized meal recommendations based on records of consumed foods, focusing on missing nutrients. The application tracks daily calorie and macronutrient intake in relation to user-defined nutritional goals and suggests recipes to help meet these goals.

The theoretical part of the thesis focuses on a review of existing methods and algorithms for food recommendation systems and a comparison of similar applications. The practical part describes the design and implementation process, including the system architecture, applied algorithms, and database structure. The final application is tested in terms of functionality, recommendation accuracy, and potential future improvements.

KEYWORDS

C#, ASP.NET, recommender systems, optimization, simulated annealing

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

ÚVOD

Efektivní plánování stravy představuje důležitý nástroj pro udržení nutriční rovnováhy, prevenci chronických onemocnění a podporu fyzické i psychické výkonnosti. Praktické využití tohoto přístupu často vyžaduje nejen znalosti výživových doporučení, ale také schopnost vyhodnocovat aktuální stav příjmu živin a přizpůsobovat jídelníček konkrétním cílům jednotlivce. Vzhledem ke složitosti tohoto procesu nabývají na významu softwarové nástroje, které umožňují automatizované sledování a návrh vhodných jídel.

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace, která poskytuje uživateli doporučení jídel na základě aktuální nutriční bilance. Aplikace je vyvinuta v jazyce C# s využitím architektury ASP.NET Core MVC a umožňuje evidenci konzumovaných pokrmů, výpočet doporučeného příjmu kalorií a makroživin a generování personalizovaných receptů. K výběru nejvhodnější kombinace jídel je využita optimalizační metoda simulovaného žíhání.

Cílem práce je vytvořit funkční systém, který bude schopen reagovat na individuální výživové potřeby uživatele a bude umět nabídnout konkrétní doporučení s ohledem na zadané cíle. Součástí řešení je návrh datového modelu, uživatelského rozhraní a základní testování funkčnosti systému.

Práce je rozdělena do několika tematických částí. Úvodní kapitola se věnuje doporučovacím systémům, jejich základnímu rozdělení a využívaným metodám. Následuje kapitola zaměřená na optimalizační algoritmy. Další část teoretického základu tvoří přehled nutričních principů a doporučení, na jejichž základě jsou formulovány cíle aplikace. Samostatná kapitola je věnována rešerši stávajících řešení, která sloužila jako inspirace při návrhu aplikace.

Na teoretický úvod navazuje popis použitých technologií, návrh systému a implementace hlavních komponent. Postupně představen návrh uživatelského rozhraní, datového modelu, způsob výpočtu doporučených hodnot a samotný algoritmus pro generování jídelníčku. Dále je popsáno uživatelské rozhraní a způsob ovládání aplikace.

Závěr práce obsahuje stručné zhodnocení dosažených výsledků a návrhy dalšího rozšíření systému.

1 DOPORUČOVACÍ SYSTÉMY

Doporučovací systémy (recommender systems) představují softwarové nástroje a techniky, které uživatelům pomáhají objevovat relevantní informace, produkty nebo služby na základě jejich preferencí, chování nebo podobnosti s ostatními uživateli [1]. Subjekt, kterému je doporučování určeno, se označuje jako *uživatel*, a objekt, který je v rámci doporučení nabízen, se nazývá *položka* [2]. Základní princip doporučování je, že existují významné závislosti mezi chování uživatelů a vlastností položek.

1.1 Formulace problému doporučení

Problém doporučení lze formulovat několika způsoby. Jeden z hlavních přístupů je prediktivní verze problému, často označovaná jako problém doplňování matice. Tento přístup spočívá v odhadu toho, jak by konkrétní uživatel ohodnotil určitou položku. Pracuje se přitom s trénovacími daty, která vyjadřují preference uživatelů vůči položkám – obvykle ve formě částečně vyplněné matice o rozměrech *uživatel* × *položka*. Chybějící hodnoty se pak predikují pomocí modelu založeného na těchto datech. [2]

Druhý přístup je označován jako řazení výsledků. Cílem tohoto modelu není předpovídat konkrétní hodnoty hodnocení, ale spíše doporučit uživateli několik nejvhodnějších položek, případně najít nejvhodnější uživatele pro určitou položku. V tomto přístupu nejsou důležité samotné číselné hodnoty predikcí, ale jejich relativní pořadí. [2]

1.2 Cíle doporučování

Cílem doporučovacích systémů je především zvýšení zisku, a to prostřednictvím nabízení relevantních produktů, které si uživatelé s větší pravděpodobností zakoupí. Aby toho bylo dosaženo, zaměřují se algoritmy nejen na samotnou relevanci, tedy schopnost trefit se do uživatelova vkusu, ale také na další aspekty.

Důležitou roli hraje schopnost doporučit položky, se kterými se uživatel dosud nesetkal. Cenná je také schopnost systému nabídnout překvapivě zajímavý obsah, který by si uživatel sám pravděpodobně nevybral. S tím souvisí i diverzita doporučení – vyšší rozmanitost položek v doporučovaném seznamu zvyšuje pravděpodobnost, že si uživatel vybere alespoň jednu z nich a zároveň napomáhá předcházet stereotypnosti výběru. Tyto aspekty nejen zvyšují kvalitu uživatelského zážitku, ale současně podporují dlouhodobý růst a obchodní úspěšnost systému. [2]

1.3 Základní modely doporučovacích systémů

Základní doporučovací systémy lze rozdělit podle typu dat, se kterými pracují, a podle způsobu, jakým tato data využívají. Typicky se rozlišují dva hlavní druhy dat: interakce uživatelů s položkami a informace o vlastnostech uživatelů a položek.

Na základě toho vznikají různé přístupy. Kolaborativní filtrování (collaborative filtering) využívá primárně interakce uživatelů s položkami. Doporučování založené na obsahu (content-based) se opírá hlavně o vlastnosti položek a profily uživatelů.

Systémy založené na znalostech (knowledge-based) se zakládají na konkrétně zadaných požadavcích uživatele a místo historických dat využívají externí znalosti. Hybridní modely (hybrid systems) kombinují více přístupů a jsou tak univerzálnější a odolnější v různých podmínkách. [2]

1.3.1 Modely kolaborativního filtrování

Modely kolaborativního filtrování využívají k vytváření doporučení hodnocení od více uživatelů. Hlavním problémem je, že matice hodnocení bývá velmi řídká, protože většina uživatelů ohodnotí jen malou část dostupných položek. Chybějící hodnocení jsou odhadována na základě podobností mezi uživatelskými preferencemi nebo položkami.

Dalším problémem může být variabilita ve způsobu hodnocení – různí uživatelé používají hodnoticí škálu odlišně. Někteří uživatelé dávají vyšší známky výjimečně, jiní naopak udělují vysoké hodnocení běžně a jen málokdy sahají po nižších hodnotách. To může zkreslit interpretaci preferencí a ovlivnit přesnost doporučení.

V některých případech se při vytváření predikčního modelu využívají optimalizační techniky. [2]

1.3.2 Modely založené na obsahu

Modely doporučování založené na obsahu využívají popisné informace o položkách (např. klíčová slova), historická hodnocení konkrétního uživatele a jeho nákupní chování k vytvoření modelu, který předpovídá jeho zájem o nové položky. Nezávisí na hodnoceních ostatních uživatelů, což je výhodné u nových položek, ale naopak méně vhodné pro nové uživatele. Nevýhodou může být menší rozmanitost doporučení, protože systém doporučuje podobné položky těm, které uživatel dříve ohodnotil. [2]

1.3.3 Modely založené na znalostech

Doporučovací systémy založené na znalostech se uplatňují zejména v situacích, kde není k dispozici dostatek uživatelských hodnocení. K doporučování se využívá podobnost mezi požadavky uživatele a popisem položky. Díky přímému zadávání preferencí mají uživatelé větší kontrolu nad samotným doporučovacím procesem.

Dělí se na modely založené na omezeních, kde uživatel specifikuje konkrétní požadavky, které jsou porovnávány s vlastnostmi položek pomocí pravidel definovaných v systému, a na modely založené na případech, kde uživatel vybere příklad položky, podle kterého systém doporučuje podobné položky.

Zvláštním případem systémů založených na znalostech jsou modely založené na užitku (utility-based). Ty pracují s předem definovanou užitkovou funkcí, která vyhodnocuje, jak moc by daná položka mohla uživatele zaujmout. Přestože různé typy doporučovacích systémů (např. kolaborativní či obsahové) implicitně hodnotí položky podle jejich přínosu pro uživatele, v tomto typu systému je užitek explicitně definován předem. [2]

2 OPTIMALIZACE

Optimalizace je proces, jehož cílem je nalézt nejlepší možné řešení daného problému v určité situaci [3]. Typicky jde o snahu snížit náklady, úsilí nebo jiný negativní faktor, případně naopak maximalizovat přínos, výkon či efektivitu. V praxi lze tyto cíle často vyjádřit pomocí matematické funkce, jejíž hodnota závisí na různých proměnných. Tato funkce se nazývá *účelová* (či *cílová*) funkce (objective function) [4].

2.1 Jednokriteriální a vícekriteriální optimalizace

Optimalizační problémy lze členit podle počtu kritérií, které mají být splněny. Pokud se snažíme nalézt řešení na základě jednoho hodnoticího hlediska, jedná se o jednokriteriální optimalizaci. Naproti tomu vícekriteriální optimalizace zohledňuje současně více různých kritérií, která bývají často v konfliktu.

Vícekriteriální úlohy se v odborné literatuře někdy označují také jako víceúčelové nebo víceatributové. Ačkoli existují metody, které lze rozšířit i na více kritérií, většina klasických optimalizačních algoritmů je navržena primárně pro případy s jedním optimalizačním cílem. [5]

2.2 Spojitá a diskrétní optimalizace

Diskrétní optimalizace se vyznačuje tím, že proměnné v účelové funkci jsou vybírány z konečné, i když často velmi rozsáhlé, množiny možností. Úlohy diskrétní optimalizace bývají často složitější na řešení, protože hodnoty cílové funkce a omezení mohou mezi jednotlivými problémy výrazně kolísat.

Naopak u spojitých optimalizačních problémů je neznámá vybírána z nekonečné množiny hodnot. Řešení těchto úloh je zpravidla jednodušší, protože je možné využít hladkosti funkcí, což umožňuje odvozovat chování funkce v okolí konkrétního bodu. Techniky spojité optimalizace se často využívají i při řešení diskrétních úkolů, například v rámci metody větví a mezí (branch-and-bound). [4]

2.3 Optimalizace s omezeními a bez omezení

Problémy bez omezení nevyžadují, aby výsledné řešení splnilo specifické podmínky. I když proměnné mohou mít některá omezení, ta nejsou v algoritmech zohledňována, protože neovlivňují konečný výsledek. Takové úlohy mohou také vzniknout jako modifikace problémů s omezeními, kde jsou původní podmínky nahrazeny penalizacemi v cílové funkci, jejichž účelem je odradit od porušování těchto podmínek.

Naopak úlohy s omezeními se obvykle objevují v situacích, kde jsou podmínky klíčové, například při zavádění rozpočtových nebo prostorových limitů.

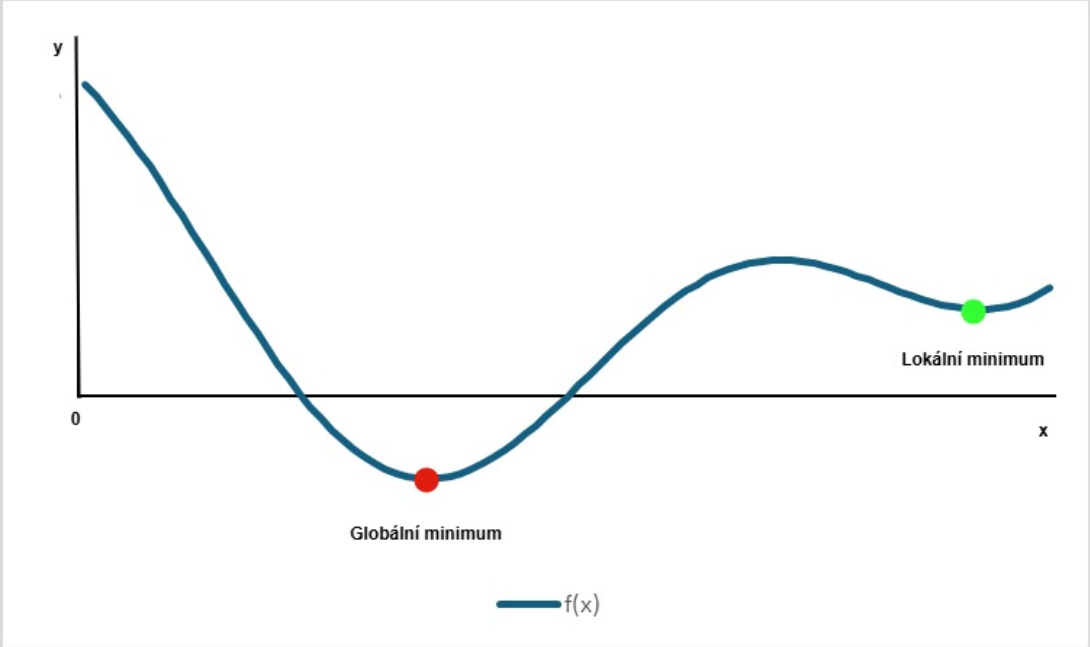
Pokud jsou jak cílová funkce, tak všechna její omezení lineární, hovoříme o lineárním programování. Tyto úlohy jsou pravděpodobně nejčastější a jsou velmi běžné v oblasti managementu, financí a ekonomiky.

Pokud je účelová funkce či alespoň jedna podmínka nelineární, označujeme takový problém jako nelineární programování. Tento typ problémů je častý především v oblasti fyzikálních věd a inženýrství. [4]

2.4 Globální a lokální optimalizace

Mnoho algoritmů pro nelineární optimalizaci hledá pouze lokální minima, se zaměřuje na hledání lokálních řešení, tedy bodů, ve kterých je hodnota cílové funkce nižší než v jejich bezprostředním okolí. Tato řešení však nejsou vždy globální, tedy nejsou nejnižšími hodnotami funkce mezi všemi možnými body.

Zatímco u konvexních a lineárních problémů je každé lokální minimum zároveň globálním, u obecných nelineárních problémů může lokální minimum být odlišné od globálního. [4] Obrázek 1 ukazuje znázornění rozdílu mezi lokálním a globálním minimem.



Obrázek 1: Rozdíl mezi lokálním a globálním minimem (zdroj: vlastní)

2.5 Stochastické a deterministické optimalizace

Zatímco v deterministických optimalizacích je model plně znám, u některých problémů není možné model přesně specifikovat, protože závisí na veličinách, které nejsou v okamžiku formulace známy. Tyto úlohy se označují jako stochastické. Místo použití nejlepšího odhadu pro neznámé veličiny je možno do modelu zahrnout dodatečné informace, které jsou k dispozici, například předpoklady o možných scénářích vývoje a pravděpodobnostech, s jakými se mohou vyskytnout. Algoritmy s těmito informacemi pracují a hledají takové řešení, které bude dobře fungovat v co nejširším spektru možných situací.

Stochastické algoritmy se obecně dají rozdělit do dvou kategorií: heuristické a metaheuristické. Heuristické algoritmy představují jednoduché postupy, které se snaží najít přijatelné řešení složitého problému bez záruky optimálnosti. Často se opírají o znalost konkrétní úlohy a fungují na principu „pokus a omyl“. Jejich hlavní výhodou je rychlost a schopnost nalézt dostatečně kvalitní řešení v rozumném čase. [5]

Metaheuristiky na tyto metody navazují a rozšiřují jejich možnosti. Vycházejí z obecnějších principů, které nejsou vázány na konkrétní typ úlohy, a jejich cílem je zlepšit schopnost algoritmu vyhnout se lokálním extrémům a prozkoumat širší oblast řešení. Typickým rysem metaheuristických metod je vyvážené použití náhodnosti a lokálního prohledávání, což jim umožňuje efektivněji hledat globální optimum.

Mezi přístupy k řešení stochastických problémů patří například optimalizace s omezením pravděpodobnosti, která vyžaduje, aby byly dané podmínky splněny s určitou pravděpodobností. Oproti tomu robustní optimalizace se snaží nalézt takové řešení, které bude přijatelné za všech možných realizací nejistých parametrů. [4]

2.6 Optimalizační metody

Optimalizační algoritmy slouží k hledání nejlepších možných řešení z množiny dostupných variant. Obvykle se rozdělují na deterministické a stochastické, podle toho, zda v průběhu výpočtu pracují s náhodností. Kromě těchto dvou skupin existují i hybridní přístupy, které kombinují výhody obou. V této kapitole budou stručně představeny vybraní zástupci těchto přístupů, které se nejčastěji využívají v praxi. [5]

2.6.1 Deterministické metody

Lineární programování

Lineární programování představuje specifický typ optimalizačních metod, který se zaměřuje na úlohy s lineární cílovou funkcí a lineárními omezujícími podmínkami, jež mohou být vyjádřeny jako rovnosti nebo nerovnosti.

Pro řešení těchto úloh se nejčastěji využívá *simplexová metoda* (simplex algorithm). Tento algoritmus, představený Georgem Dantzigem v roce 1947, systematicky prohledává extrémní body přípustné oblasti, přičemž v každém kroku zkoumá, zda aktuální bod představuje optimální řešení. Pokud není podmínka optimality splněna, algoritmus přechází do sousedního vrcholu s lepší hodnotou cílové funkce. Proces pokračuje, dokud není nalezen optimální bod, nebo není zjištěno, že úloha nemá omezené řešení. [3]

Nelineární programování

Nelineární optimalizační metody se používají k hledání extrémů (minim nebo maxim) v případech, kdy účelová funkce nebo některé z omezení nejsou lineární. Důležitým aspektem je konvexita – pokud funkce není konvexní ani konkávní, je nalezení jejího globálního minima nebo maxima výrazně obtížnější.

Pokud jsou účelová funkce i podmínky vyjádřeny pomocí jednoduchých analytických vztahů, lze při řešení využít klasické analytické metody. V opačném případě, kdy jsou tyto vztahy příliš složité nebo nejsou k dispozici ve formě explicitních funkcí návrhových proměnných, je nutné sáhnout po numerických či aproximačních postupech. [3] [5]

2.6.2 Stochastické metody

Jak bylo uvedeno výše, stochastické algoritmy využívají kombinaci náhodnosti a lokálního prohledávání k nalezení globálního optima. Většina těchto algoritmů byla inspirovaná přírodou.

V rámci této skupiny lze algoritmy dále rozlišit podle toho, zda pracují s jediným řešením, nebo s celou populací řešení. Na základě tohoto kritéria se dělí na *trajektoriální* (bodové, trajectory-based) a *populační* (population-based) metody. Trajektoriální algoritmy, jako například simulované žíhání, sledují trajektorii jednoho řešení v prostoru. Naproti tomu populační metody, jako jsou genetické algoritmy, optimalizace hejnem částic nebo optimalizace mravenčí kolonií, vyhodnocují a upravují větší množinu řešení současně. [5]

Genetické algoritmy

Genetické algoritmy představují evoluční optimalizační metodu inspirovanou Darwinovou teorií přirozeného výběru. Využívají principy jako je selekce, křížení a mutace k postupnému zlepšování řešení dané úlohy. Jsou vhodné zejména pro složité a nelineární optimalizační problémy, u nichž selhávají klasické analytické metody – například tehdy, když není možné účelovou funkci explicitně vyjádřit, nebo je výpočetně příliš náročná.

Hlavní výhodou genetických algoritmů je jejich schopnost řešit složité optimalizační úlohy a možnost paralelizace. Je však nutné vhodně nastavit parametry algoritmu – zejména účelovou (fitness) funkci, velikost populace, míru mutace a křížení či způsob výběru nových jedinců. Nevhodná volba může vést k pomalé konvergenci nebo k nerelevantním výsledkům. [5]

Algoritmus začíná náhodně vygenerovanou množinou řešení, označovanou jako populace. Každý jedinec v populaci je následně ohodnocen pomocí fitness funkce, která určuje kvalitu daného řešení vzhledem k řešenému problému. Na základě těchto hodnot probíhá selekce, během níž jsou vybíráni jedinci (rodiče) pro vytvoření nové generace. Vybrané páry se kříží – pro každý pár se náhodně určí bod křížení (crossover point), ve kterém se genetická informace obou rodičů vzájemně kombinuje.

Výsledkem křížení jsou potomci, kteří přebírají části genetické informace od obou rodičů. Na závěr může každý gen (prvek řetězce) podléhat mutaci, která se provádí s malou pravděpodobností a mění hodnotu genu náhodným způsobem. [6]

Optimalizace hejnem částic

Optimalizace hejnem částic (částicová rojová optimalizace, Particle Swarm Optimization) je algoritmus inspirovaný chováním hejna ptáků nebo ryb. Využívá jednoduchou aktualizaci pozice částic na základě jejich předchozího pohybu, nejlepší nalezené pozice a globálního optima. Každá částice v prostoru reprezentuje potenciální řešení a její pohyb je řízen kombinací deterministické a náhodné složky. Algoritmus je jednoduchý na implementaci a efektivní zejména u složitějších problém s více lokálními extrémy. [5]

Optimalizace mravenčí kolonií

Optimalizace mravenčí kolonií (Ant Colony Optimization) je inspirována chováním mravenců při hledání potravy. Každý umělý „mravenec“ v algoritmu vytváří možné řešení úlohy a pokládá na svou cestu feromon, jehož intenzita se později využívá při výběru dalších cest. Cesty s vyšší koncentrací feromonu jsou pravděpodobněji vybírány ostatními, čímž se posiluje

výhodné řešení. Díky odpařování feromonu se však systém nenechá snadno uvěznit v horších lokálních řešeních. Tím vzniká pozitivní zpětná vazba a algoritmus postupně nachází optimální nebo velmi kvalitní řešení. [5]

Simulované žíhání

Simulované žíhání (Simulated Annealing) [7] je jeden z optimalizačních algoritmů inspirovaných přírodou, konkrétně fyzikálním procesem ochlazování kovu. Při žíhání je materiál nejprve zahříván na vysokou teplotu a poté pomalu ochlazován. Tento postup slouží k odstranění vnitřních vad materiálu. Při vysoké teplotě dojde k přeskupení částic, čímž se narušená krystalová síť materiálu částečně opraví. Při následném chladnutí se částice ustálí v rovnovážných pozicích a zpevní se struktura látky.

V algoritmu je zavedena teplota, která udává pravděpodobnost, s jakou je přijato horší řešení. Na začátku je tato pravděpodobnost vyšší, což umožňuje algoritmu prozkoumávat různá řešení a vyhýbat se uvěznění v lokálních optimech. Postupně se ale pravděpodobnost (teplota) snižuje, takže se algoritmus více zaměřuje na jemné doladění řešení.

Klíčovým prvkem je zvolení správné rychlosti snižování teploty – příliš rychlé ochlazení může vést k horším výsledkům, zatímco příliš pomalé zvyšuje výpočetní náročnost. Správně nastavený průběh teploty umožňuje, aby se algoritmus nakonec ustálil v řešení, které je nejen lokálně, ale i globálně optimální. [8]

3 VÝŽIVA A NUTRIČNÍ BILANCE

Lidské tělo je závislé na rovnováze mezi příjmem energie a její spotřebou, označované jako *metabolická homeostáza*. Tato rovnováha je klíčová pro zajištění základních životních funkcí. Mezi hlavní složky potravy, které ji ovlivňují, patří *makroživiny* (bílkoviny, sacharidy a tuky) a *esenciální minerální látky* (např. sodík, draslík a vápník). Tyto látky hrají zásadní roli například při činnosti nervového systému, svalových kontrakcí a regulaci hormonální aktivity.

V případě nerovnováhy mezi příjmem a výdejem energie dochází k narušení tělesných funkcí. Nedostatek nebo nadbytek živin může vést ke zdravotním problémům, včetně poruch imunitního systému nebo chronických onemocnění, jako je diabetes mellitus 2. typu. Sledování příjmu živin a udržování přiměřené nutriční bilance může pomoci nejen při hubnutí či udržení fyzického zdraví, ale také při prevenci nemocí a podpoře soustředění a produktivity během dne. [9][10]

3.1 Kalorický příjem a bazální metabolismus

K vyjádření energetické hodnoty potravin se obvykle používá jednotka *kalorie*. V praxi se nejčastěji udává v kilokaloriích (kcal), přičemž 1 kcal odpovídá množství energie potřebné k ohřátí jednoho kilogramu vody o jeden stupeň Celsia. Energetická hodnota potravin závisí na obsahu makroživin – sacharidy a bílkoviny poskytují přibližně 4 kcal na gram, zatímco tuky obsahují přibližně 9 kcal na gram. [9]

Celkový energetický výdej organismu se skládá z několika složek: *bazálního metabolismu* (Basal Metabolic Rate, BMR), který zajišťuje základní životní funkce v klidovém stavu; *termického efektu potravy* (Thermic Effect of Food, TEF), tedy energie potřebné ke zpracování přijaté potravy; a výdeje při fyzické aktivitě. Někdy se zahrnuje i *termoregulace*, tedy udržování stálé tělesné teploty.

Bazální metabolismus představuje množství energie, které tělo spotřebuje pro udržení základních funkcí jako dýchání, srdeční činnost, činnost ledvin a krevní oběh. U dospělého člověka tvoří BMR přibližně 50–70 % celkového denního energetického výdeje.

Na výši bazálního metabolismu má zásadní vliv složení těla – zejména poměr beztukové tělesné hmoty (svaly, orgány) k tukové tkáni. Orgány jako mozek, játra, ledviny a srdce tvoří sice jen asi 5–6 % tělesné hmotnosti, ale podílejí se na přibližně 60 % bazální spotřeby energie. Naproti tomu tuková tkáň tvoří často kolem 20 % tělesné hmotnosti, ale přispívá jen asi 5 % k bazálnímu

výdeji. S věkem nebo při změnách tělesné kompozice (např. při hubnutí nebo růstu svalové hmoty) se může bazální metabolismus významně měnit.

V literatuře se lze setkat také s pojmem *klidová metabolická rychlost* (Resting Metabolic Rate, RMR), která popisuje podobnou hodnotu jako BMR. Pokud je RMR přepočtena na 24hodinový výdej, označuje se jako klidový energetický výdej (Resting Energy Expenditure, REE). REE je obvykle o cca 10 % vyšší než BMR, protože měření probíhá za méně přísných podmínek (např. po krátkém odpočinku místo úplného klidu na lačno).

Energetický výdej lze měřit pomocí přímé (direct calorimetry) nebo nepřímé (indirect calorimetry) kalorimetrie, metodou dvojitě značené vody (doubly labeled water) nebo výpočtem na základě odvozených vzorců.

Přímá kalorimetrie sleduje množství tepla uvolňovaného tělem v kontrolovaném prostředí. Nepřímá kalorimetrie měří spotřebu kyslíku a výdej oxidu uhličitého. Metoda dvojitě značené vody určuje celkový energetický výdej pomocí podávání stabilních izotopů vody, jejichž úbytek se sleduje v krvi a moči po dobu přibližně tří týdnů.

Vedle přímých měření lze energetický výdej také odhadnout pomocí vzorců. V průběhu let bylo využíváno mnoho různých způsobů odhadování energetické potřeby. Tyto odhady byly založeny na tělesném povrchu, tělesné hmotnosti nebo výpočtech pomocí regresních rovnic zahrnujících pohlaví, věk, hmotnost a výšku. Ukázalo se, že tyto odhady dobře korelují s výsledky nepřímé kalorimetrie nebo metody dvojitě značené vody.

Nejčastěji používané regresní rovnice v klinické praxi pocházejí od Harrise a Benedicta z roku 1919. Tyto rovnice byly založeny na výsledcích nepřímé kalorimetrie a od té doby jen mírně upraveny. Výpočty se provádějí zvlášť pro muže a ženy – viz Rovnice 1 a Rovnice 2.

Rovnice 1: BMR podle Harris–Benedict (muži)

$$Muž: BMR = 66{,}5 + (13{,}7 \times hmotnost\ [kg]) + (5 \times výška\ [cm]) - (6{,}8 \times věk\ [roky])$$

Rovnice 2: BMR podle Harris–Benedict (ženy)

$$Žena: BMR = 655{,}1 + (9{,}56 \times hmotnost\ [kg]) + (1{,}85 \times výška\ [cm]) - (4{,}7 \times věk\ [roky])$$

V moderní klinické praxi se však často preferuje rovnice Mifflin-St Jeor, která je určena pro výpočet klidového energetického výdeje (REE). Výpočet se taktéž rozlišuje pro muže (Rovnice ***3***) a ženy (Rovnice 4). [11]

Rovnice 3: REE podle Mifflin–ST Jeor (muži)

$$Muž: REE = (10 \times hmotnost\ [kg]) + (6{,}25 \times výška\ [cm]) - (5 \times věk\ [roky]) + 5$$

Rovnice 4: REE podle Mifflin–ST Jeor (ženy)

$$Žena: REE = (10 \times hmotnost\ [kg]) + (6{,}25 \times výška\ [cm]) - (5 \times věk\ [roky]) - 161$$

Podle systematického přehledu odborné literatury je právě rovnice Mifflin-St Jeor považována za nejpřesnější z běžně používaných predikčních rovnic. Dokáže totiž odhadnout klidový metabolismus (RMR) s přesností ±10 % u většího počtu osob (jak obézních, tak neobézních) než kterákoli jiná rovnice. Zároveň vykazuje nejmenší rozptyl chyb, což z ní činí nejvhodnější volbu pro odhad energetických potřeb v klinické praxi. [12]

Pro stanovení celkové energetické potřeby je nutné k hodnotě BMR nebo REE připočítat výdej spojený s fyzickou aktivitou, případně i termický efekt potravy. [11]

3.2 Makroživiny

Živiny jsou látky nezbytné pro základní funkce lidského organismu. Tělo je využívá k produkci energie, růstu, regeneraci tkání, dýchání, pohybu a dalším klíčovým biologickým procesům. Vzhledem k tomu, že lidský organismus si většinu živin nedokáže syntetizovat sám, je nezbytné je přijímat prostřednictvím stravy. Z hlediska funkce a potřeby pro tělo se živiny dělí do šesti základních tříd: sacharidy, tuky, bílkoviny, voda, vitamíny a minerální látky.

Kromě těchto základních živin potraviny často obsahují i látky, které nejsou přímo nezbytné pro fungování organismu. Ty mohou mít jak negativní účinky (například cholesterol, umělá barviva nebo konzervační látky), tak pozitivní vliv na zdraví (například antioxidanty), které hrají důležitou roli v ochraně buněk před poškozením.

Mezi živiny se řadí i skupina látek, které tělo potřebuje ve velkém množství – tzv. *makroživiny*. Do této skupiny patří sacharidy, tuky a bílkoviny, které slouží především jako zdroj energie. Energie je uvolňována z jejich chemických vazeb během metabolických procesů a následně využívána k zajištění základních tělesných funkcí. Voda je rovněž považována za makroživinu, protože ji tělo vyžaduje ve značném množství. Na rozdíl od ostatních makroživin však neobsahuje žádné kalorie a neslouží jako zdroj energie. [9]

3.2.1 Sacharidy

Sacharidy jsou organické molekuly složené z uhlíku, vodíku a kyslíku, které v lidské stravě slouží především jako zdroj energie. Jsou nezbytné i pro správnou funkci nervové soustavy,

srdce a ledvin a slouží jako základní stavební kameny některých složitějších biologických molekul. Z hlediska chemické struktury se sacharidy dělí na dvě hlavní skupiny: jednoduché (monosacharidy a disacharidy) a složené (oligosacharidy a polysacharidy), přičemž jednoduché se rychle vstřebávají a složené uvolňují energii postupně. [9]

Hlavními potravinovými zdroji sacharidů jsou obiloviny, mléčné výrobky, ovoce a škrobová zelenina, jako jsou například brambory. Menší množství sacharidů se nachází také v neškrobové zelenině [9]. Doporučený denní příjem sacharidů dospělého člověka by měl činit 45–65 % celkového energetického příjmu [13].

3.2.2 Tuky

Tuky (lipidy) jsou organické molekuly z uhlíku, vodíku a kyslíku, které se liší od sacharidů tím, že nejsou rozpustné ve vodě. Jejich hlavní funkcí je uchovávání energie, ale rovněž se podílejí na tvorbě buněčných membrán, ochraně orgánů, regulaci tělesné teploty a dalších životně důležitých procesech.

Hlavním zdrojem tuků je například máslo, olej, maso, mléčné výrobky, ořechy a semena [9]. Doporučený denní příjem lipidů je stanoven na 20–35 % celkového denního energetického příjmu [13].

3.2.3 Bílkoviny

Bílkoviny (proteiny) jsou makromolekuly složené z aminokyselin, což jsou organické sloučeniny obsahující uhlík, vodík, kyslík a dusík. V lidském těle se podílejí na tvorbě a udržování kostí, svalové hmoty a kožní tkáně. [9] Slouží také jako enzymy, hormony, transportní látky, stavební složky buněk a tkání, a podílí se na imunitních procesech. [11]

Z hlediska výživy je lze přijímat z různorodých zdrojů, např. z masa, mléčných a rybích produktů nebo ze sóji [9]. Doporučený denní příjem bílkovin se pohybuje mezi 10–35 % celkového energetického příjmu [13].

4 EXISTUJÍCÍ APLIKACE PRO DOPORUČOVÁNÍ JÍDEL

Tato kapitola se zaměřuje na srovnání vybraných existujících aplikací, které slouží k evidenci příjmu potravy a výpočtu nutričních hodnot. Cílem je analyzovat jejich funkcionalitu, přínosy a omezení s ohledem na návrh vlastní aplikace v rámci této práce.

4.1 Eat this much

Eat This Much je dostupná jako webová i mobilní aplikace zaměřená na tvorbu personalizovaných jídelníčků. Doporučuje recepty, které lze filtrovat dle složitosti, ceny a dalších parametrů. Umožňuje detailní nastavení kalorických a nutričních cílů, včetně rozlišení pro jednotlivé dny v týdnu, a generuje jídelní plány s ohledem na zadané preference, rozpočet a dostupné suroviny.

Mezi funkce prémiové verze patří například automatické generování nákupního seznamu, plánování jídel až na týden dopředu, integrace s Apple Health nebo správa virtuální spíže. Aplikace podporuje přidávání vlastních receptů, omezení určitých surovin a nastavení preferencí pro jednotlivá jídla (např. složitost, čas přípravy).

Mezi nevýhody patří absence českého jazykového rozhraní, nefunkčnost offline, chybějící funkce pro více uživatelů a přítomnost převážně amerických potravin a receptů. Některé funkce jsou dostupné pouze v placené verzi, která může být pro některé uživatele cenově méně dostupná. [14]

4.2 Calorie Counter: Cronometer

Cronometer je webová i mobilní aplikace pro sledování výživy, zdraví a životního stylu. Slouží především jako detailní výživový deník – uživatelé si ručně zaznamenávají konzumovaná jídla, doplňky stravy a aktivity, na jejichž základě aplikace vyhodnocuje příjem živin a plnění stanovených cílů. Umožňuje detailní monitoring až 84 živin včetně mikroživin, vitamínů, minerálů a aminokyselin. Uživatelé si mohou nastavit vlastní cíle pro kalorie, makroživiny i mikroživiny.

Cronometer obsahuje databázi více než jednoho milionu ověřených potravin a podporuje skenování čárových kódů. Nabízí také možnost zaznamenávat biometrická data, propojení s chytrými zařízeními (např. Apple Health, Fitbit, Garmin) a tvorbu vlastních receptů. Prémiová verze rozšiřuje funkce o plánování jídel, pokročilé reporty, nástroj pro sledování přerušovaného půstu či import receptů z webu.

Mezi nevýhody patří méně intuitivní rozhraní, nedostupnost v češtině, a fakt, že většina návrhů jídel chybí – aplikace tedy slouží spíše k analýze již zkonzumované stravy než k jejímu plánování. [15]

4.3 YAZIO

Mobilní aplikace YAZIO se vyznačuje přehledným a intuitivním uživatelským rozhraním. Nabízí široké spektrum funkcí, jež zahrnují například sledování kalorického příjmu a příjmu makroživin, možnost plánování jídel s využitím receptů či podporu přerušovaného půstu. Součástí aplikace jsou rovněž motivační prvky, jako jsou série zaznamenaných dnů, upozornění a aktivní komunita uživatelů, které přispívají k udržení pravidelného používání. Aplikace existuje i v českém jazyce.

Pozitivně lze hodnotit i možnost propojení s chytrými zařízeními a cenovou dostupnost – základní funkce jsou k dispozici zdarma a placená verze PRO je cenově výhodnější ve srovnání s konkurenčními produkty.

Mezi hlavní slabiny patří menší a méně přesná databáze potravin, což uživatele často nutí k manuálnímu zadávání nutričních údajů. Aplikace rovněž nenabízí pokročilé možnosti individuálního nastavení nutričních cílů. Některé funkce, například podrobné statistiky či rozšířené režimy přerušovaného půstu, jsou navíc dostupné pouze v placené verzi. [16]

5 POUŽITÉ TECHNOLOGIE

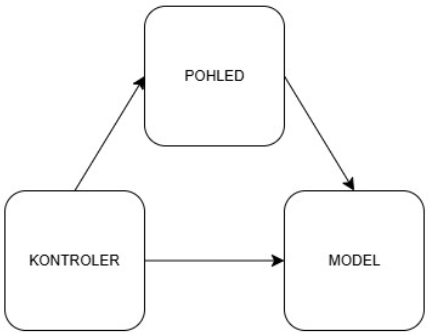
Aplikace byla vyvíjena v jazyce C# s využitím frameworku ASP.NET Core MVC (Model-View-Controller). Pro trvalé ukládání dat byla zvolena databáze LiteDB ve formě lokálního souboru. Namísto externích knihoven pro ORM (objektově relační mapování) byl přístup k datům řešen prostřednictvím LINQ a nativních dotazů. Uživatelé se přihlašují pomocí jednoduchého autentizačního systému s hashováním hesel.

5.1 C# a ASP.NET Core MVC

C# je programovací jazyk vyvinutý společností Microsoft. Je určen pro vývoj mnoha typů aplikací, např. webových, desktopových a mobilních. Jazyk je součástí platformy .NET, která poskytuje výkonné prostředí pro běh aplikací (runtime) a rozsáhlé knihovny tříd. Díky silné typové kontrole, podpoře objektově orientovaného i funkcionálního programování, práci s kolekcemi přes LINQ a vestavěné podpoře asynchronního zpracování nabízí C# vysokou produktivitu při zachování výkonu a bezpečnosti. [17]

ASP.NET Core je platformně nezávislý vysoce výkonný aplikační rámec (framework) s volně dostupným zdrojovým kódem (open-source). Umožňuje vyvíjení webových aplikací, služeb a programovacích rozhraní (API). Nabízí modulárně orientovanou architekturu, testovatelnost a jednoduchou integraci moderních vývojových nástrojů. Součástí ASP.NET Core je také vestavěné vkládání závislostí (dependency injection), flexibilní konfigurace prostředí a možnost nasazení jak na cloudová řešení, tak lokální servery. [18]

ASP.NET Core MVC je vývojová platforma, která využívá architektonický vzor Model-View-Controller (model-pohled-kontroler). Tento vzor rozděluje aplikaci do tří komponent – na datové modely, uživatelské rozhraní (pohledy) a řídicí logiku (kontrolery). Tím je dosaženo oddělení prezentační, řídicí a datové vrstvy aplikace. Toto rozdělení ilustruje Obrázek 2. Požadavky od uživatele zpracovává kontroler, který komunikuje s modelem za účelem provedení potřebné akce nebo získání dat. Následně zvolí odpovídající pohled a předá mu veškerá data potřebná k zobrazení. [19]



Obrázek 2: Architektura MVC (zdroj: vlastní)

5.2 Databázová vrstva

LiteDB (Obrázek 3) je lehká a výkonná NoSQL databáze pro .NET, která ukládá data ve formě dokumentů do jednoho souboru bez nutnosti serverového nasazení. Její struktura je inspirovaná databází MongoDB. [20]

Data jsou ukládána jako dokumenty ve formátu BSON (binární verze JSON). Dokumenty jsou organizovány do kolekcí, které představují logické skupiny záznamů se společnými indexy. Kolekce jsou analogické k tabulkám v relačních databázích. Nevyžadují však pevné schéma, každý dokument může mít odlišnou strukturu. [20]

Nad kolekcemi v databázi byly prováděny dotazy pomocí LINQ (Language Integrated Query, jazykově integrovaný dotaz), což je sada jazykových prvků v C#, které usnadňují práci se soubory dat. Ačkoliv byl původně navržen především pro přístup k relačním databázím, je použitelný i pro dotazy nad objektovými kolekcemi, JSON nebo XML dokumenty. LINQ umožňuje zapisovat dotazy přímo v jazyce C# pomocí přehledné syntaxe připomínající databázové dotazy. Výhodou je také oddělení výběru dat od jejich následného zpracování, což vede k přehlednějšímu kódu. [17]

Obrázek 3: Logo databáze LiteDB (zdroj: [21])

6 NÁVRH APLIKACE

6.1 Analýza požadavků

Návrhu aplikace předcházelo stanovení požadavků, které sloužily jako podklad pro implementaci jednotlivých funkcí. Požadavky byly rozděleny na funkční (Tabulka 1) a nefunkční (Tabulka 2). Funkční požadavky definují konkrétní chování aplikace, nefunkční požadavky se zaměřují na technické a provozní vlastnosti.

Tabulka 1: Přehled funkčních požadavků

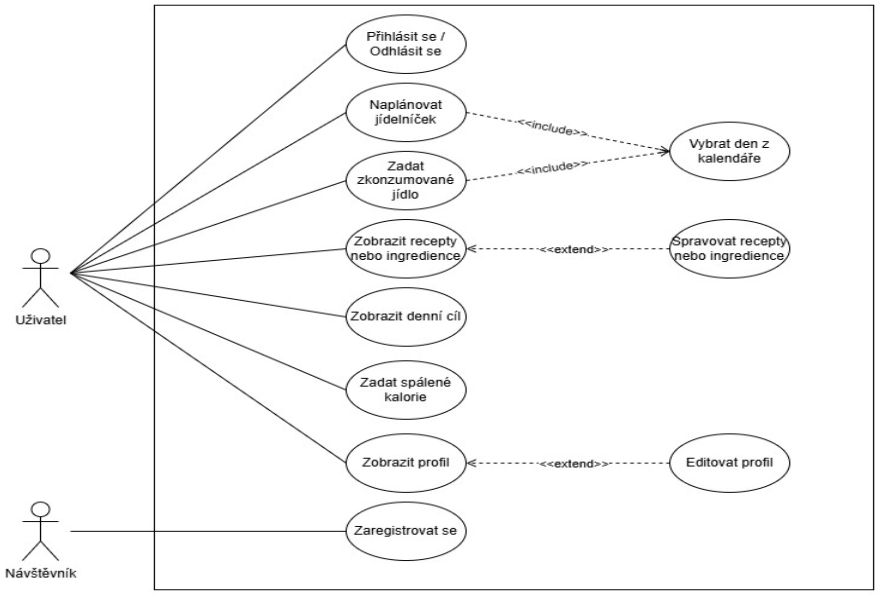
ID	Název požadavku	Popis
F1	Záznam jídla	Uživatel může zadat zkonzumované jídlo včetně ingrediencí a jejich množství.
F2	Výpočet denního cíle	Aplikace na základě výšky, váhy, věku, pohlaví a cíle uživatele určí doporučený denní příjem kalorií a makroživin.
F3	Doporučení jídelního plánu	Na základě chybějících živin aplikace navrhne vhodný denní jídelníček.
F4	Správa receptů a ingrediencí	Uživatel může přidávat, upravovat a mazat recepty a ingredience.
F5	Plánování podle kalendáře	Uživatel si může zvolit konkrétní den, pro který chce naplánovat jídelníček.
F6	Zobrazení nutriční bilance	Aplikace zobrazí aktuální stav přijatých makroživin a kalorií.
F7	Vyhledávání receptů a ingrediencí	Recepty i ingredience lze vyhledávat podle názvu.
F8	Záznam zkonzumovaných jídel	Uživatel má možnost si zapsat, co během dne snědl.
F9	Záznam spálených kalorií	Aplikace umožňuje zaznamenávání spálených kalorií. Denní příjem kalorií a makroživin je následně přepočítán.
F10	Uživatelské účty	Aplikace umožňuje registraci, přihlášení a odhlášení uživatele.

Tabulka 2: Přehled nefunkčních požadavků

ID	Název požadavku	Popis
N1	Uživatelská přívětivost	Aplikace je snadno ovladatelná i pro běžného uživatele.
N2	Architektura a technologie	Aplikace je postavena na architektuře ASP.NET Core MVC a implementována v jazyce C#.
N3	Jazykové prostředí	Uživatelské rozhraní aplikace je v českém jazyce.
N4	Uživatelský přístup	Uživatelé se přihlašují pomocí uživatelského jména a hesla. Hesla jsou uložena v hashované podobě.

6.2 Diagram případů užití

Pro návrh systému bylo důležité popsat základní scénáře, ve kterých uživatel interaguje s aplikací. K jejich zachycení slouží diagram případů užití (use case diagram). Diagram (Obrázek 4) znázorňuje jednotlivé funkce aplikace z pohledu uživatele. Ukazuje také, jaké možnosti má přihlášený a nepřihlášený uživatel. Některé funkce jsou propojeny pomocí vztahů include a extend, které vyjadřují závislosti a rozšíření jednotlivých scénářů. Vztah include označuje povinné začlenění jednoho případu užití do jiného, extend značí rozšíření scénáře, které se provádí pouze za určitých podmínek.



Obrázek 4: Diagram případů užití (zdroj: vlastní)

6.3 Datový model a hlavní entity

Vnitřní logika aplikace je založena na několika třídách, které reprezentují hlavní entity systému. Patří mezi ně *User* (uživatel), *MealPlan* (denní jídelníček), *Recipe* (recept) a *Ingredient* (ingredience).

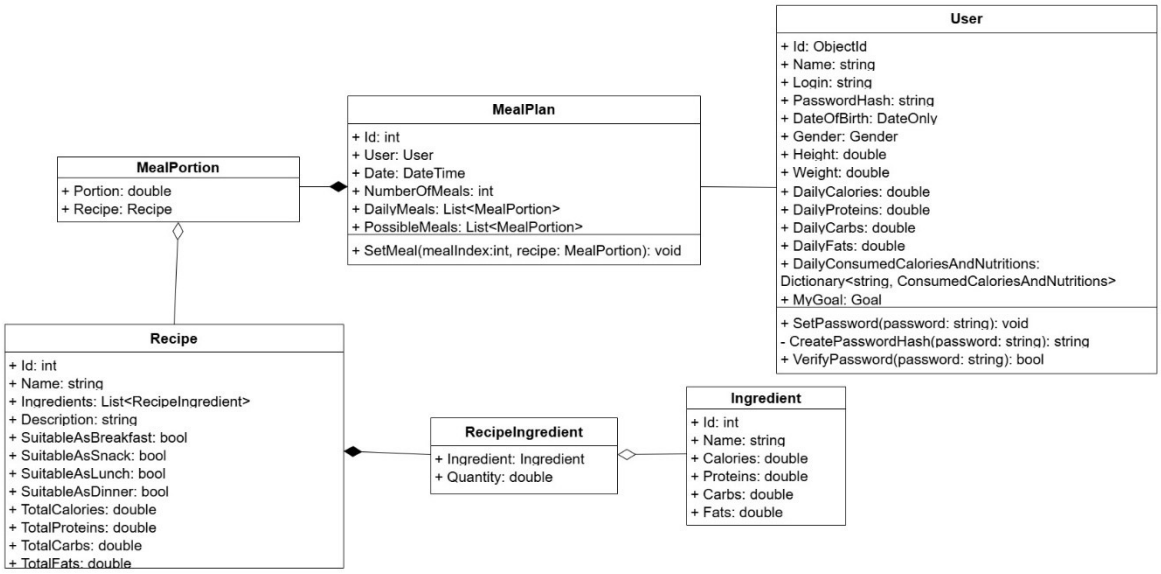
Třída *User* zastupuje uživatele systému. Uchovává jeho osobní údaje, cílové nutriční hodnoty a informace o zkonzumovaných živinách. Je propojena s entitou *MealPlan*, která obsahuje jeho denní jídelníčky.

MealPlan představuje stravovací plán pro jeden den. Zahrnuje datum, počet jídel, seznam plánovaných pokrmů (*PossibleMeals*) a seznam skutečně zkonzumovaných jídel (*DailyMeals*). Každý jídelníček je jednoznačně přiřazen konkrétnímu uživateli.

Třída *MealPortion* reprezentuje jeden naplánovaný pokrm. Vztahuje se ke konkrétnímu receptu a udává velikost porce jako násobek výchozího množství.

Recipe představuje recept složený z více ingrediencí. Sestává se z popisu, informací o vhodnosti receptu pro různé části dne a vlastnosti jako celkový počet kalorií a živin. Na propojení receptu a ingrediencí slouží třída *RecipeIngredient*, která uchovává informaci o množství konkrétní suroviny v receptu. Samotné suroviny jsou reprezentovány třídou *Ingredient*. U každé z nich jsou uvedeny základní nutriční hodnoty vztažené na 100 gramů.

Vztahy mezi výše uvedenými třídami zachycuje Obrázek 5.

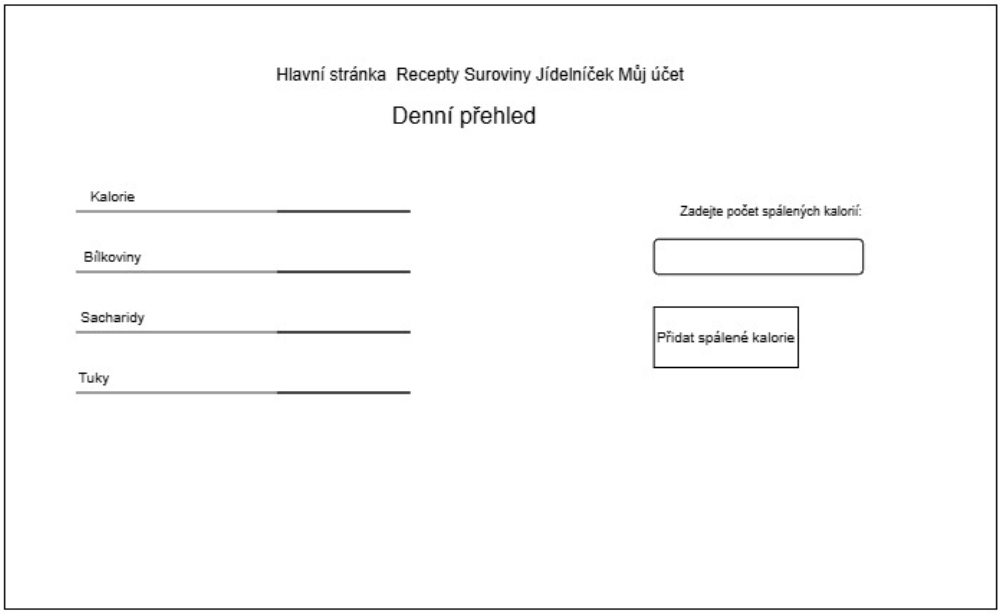


Obrázek 5: Diagram tříd (zdroj: vlastní)

6.4 Návrh rozhraní

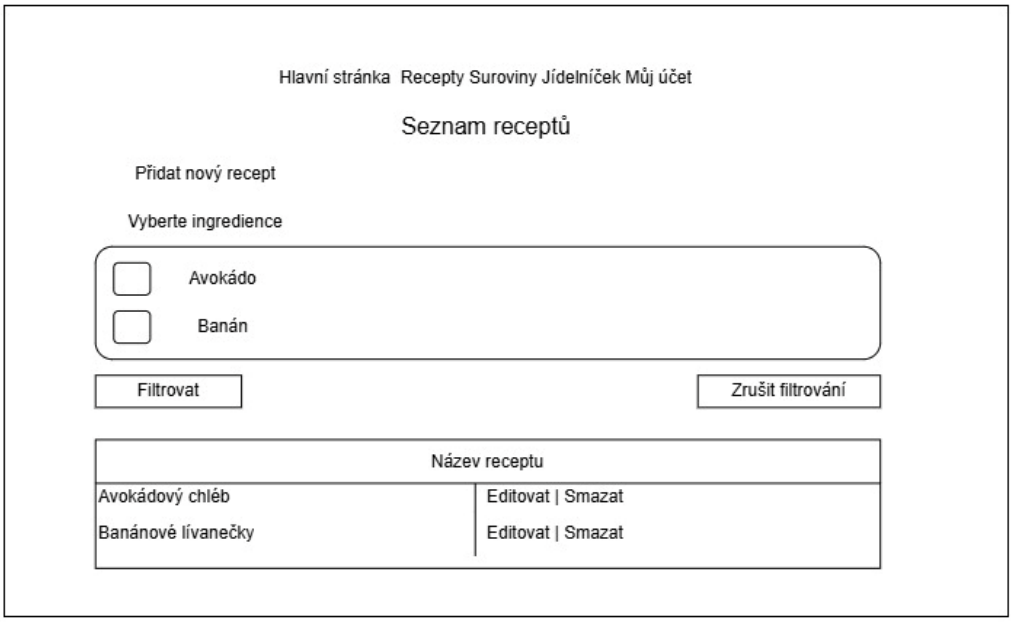
Rozvržení jednotlivých částí uživatelského rozhraní bylo navrženo formou wireframů (drátěných modelů), které sloužily jako podklad pro implementaci. Na následujících obrázcích jsou zachyceny čtyři základní typy stránek, tvořící klíčovou část aplikace.

Obrázek 6 zachycuje wireframe hlavní stránky aplikace. Je zde uveden aktuální stav příjmu kalorií, bílkovin, sacharidů a tuků pomocí ukazatelů průběhu (progress barů). Uživatel zde může zadat spálené kalorie a tím upravit své denní cíle.



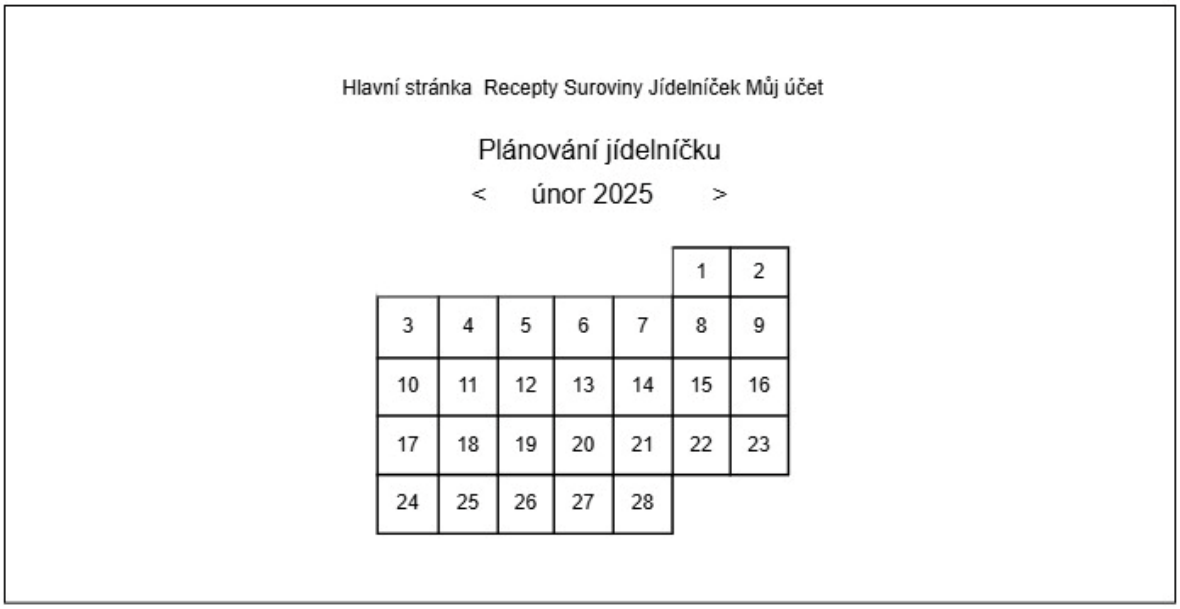
Obrázek 6: Wireframe hlavní stránky aplikace (zdroj: vlastní)

Obrázek 7 je zobrazuje seznam receptů. V horní části okna se nachází odkaz pro přidání nového receptu. Recepty lze filtrovat podle zvolených ingrediencí (uživateli se pak zobrazí pouze ty recepty, na které má všechny suroviny) a vyhledávat je podle názvu. U každého receptu jsou k dispozici akce pro úpravu či smazání.



Obrázek 7: Wireframe seznamu receptů (zdroj: vlastní)

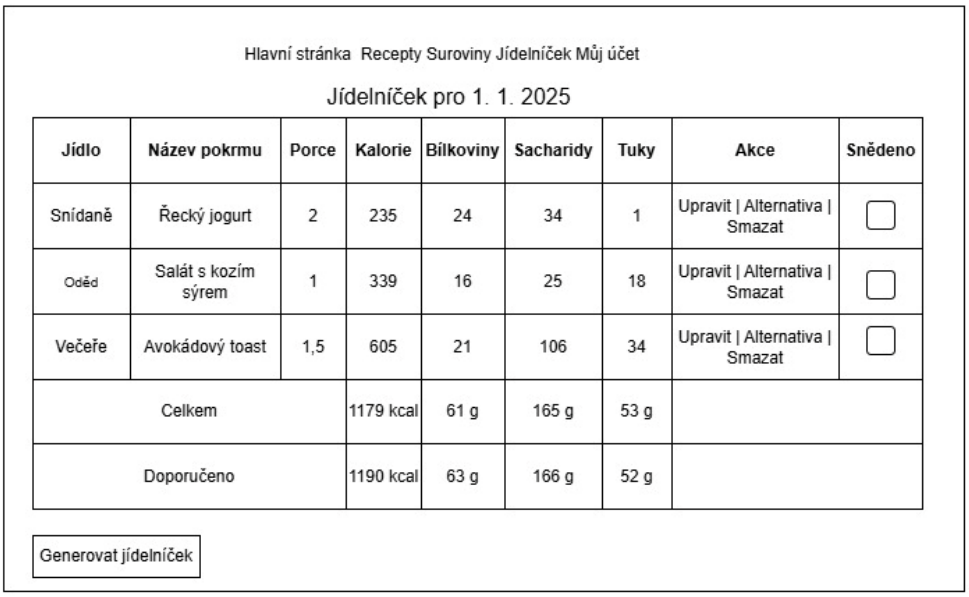
Obrázek 8 ukazuje zobrazení kalendáře pro plánování jídelníčku. Uživatel si zde může zvolit konkrétní den, pro který chce jídelníček vytvořit či zobrazit. Pomocí šipek lze přecházet mezi jednotlivými měsíci.



Obrázek 8: Wireframe plánování jídelníčku (zdroj: vlastní)

Obrázek 9 představuje jídelníček pro zvolený den. Stránka obsahuje seznam naplánovaných jídel včetně množství, kalorických hodnot a jednotlivých makroživin. Každé jídlo je možné

upravit, nahradit nebo odstranit. Také lze označit, zda bylo skutečně zkonzumováno (což vede k aktualizaci přijatých živin). Na stránce je také možnost automatického generování jídelníčku.



Hlavní stránka Recepty Suroviny Jídelníček Můj účet

Jídelníček pro 1. 1. 2025

Jídlo	Název pokrmu	Porce	Kalorie	Bílkoviny	Sacharidy	Tuky	Akce	Snědeno
Snídaně	Řecký jogurt	2	235	24	34	1	Upravit | Alternativa | Smazat	☐
Oběd	Salát s kozím sýrem	1	339	16	25	18	Upravit | Alternativa | Smazat	☐
Večeře	Avokádový toast	1,5	605	21	106	34	Upravit | Alternativa | Smazat	☐
Celkem			1179 kcal	61 g	165 g	53 g		
Doporučeno			1190 kcal	63 g	166 g	52 g		

Generovat jídelníček

Obrázek 9: Wireframe jídelníčku pro daný den (zdroj: vlastní)

7 IMPLEMENTACE

Tato kapitola popisuje praktickou realizaci aplikace, která byla navržena v předchozích částech práce. Postupně jsou představeny klíčové části implementace včetně přenosu požadavků do kódu, výpočtu nutričních hodnot, hodnocení jídelníčků a návrhu doporučení pomocí optimalizačních algoritmů. Funkčnost aplikace byla ověřována ručními testy během vývoje. Testování se soustředilo na hlavní scénáře, jako je autentizace, práce s recepty, výpočty nutričních hodnot a plánování denního příjmu. Aplikace se ve všech testovaných situacích chovala konzistentně a stabilně.

7.1 Přenos požadavků do aplikace

Základem vytvoření řešení bylo implementování funkčních a nefunkčních požadavků uvedených v kapitole 6.1 Analýza požadavků. Každý požadavek byl zohledněn jako konkrétní funkcionalita, realizovaná prostřednictvím odpovídajících kontrolerů a pohledů v rámci architektury ASP.NET Core MVC. V této podkapitole jsou uvedeny vybrané příklady přenosu funkčních požadavků. Některé složitější scénáře, jako je výpočet nutričních cílů nebo algoritmické doporučení receptů, jsou podrobně rozvedeny v následujících částech kapitoly.

Záznam jídla (F1) byl implementován pomocí kontroleru FoodLogController.cs a pohledu LogRecipe.cshtml, který využívá částečný pohled (partial view) _RecipeList.cshtml. Požadavky na správu receptů a surovin a jejich vyhledávání (F4 a F7) byly realizovány skrze IngredientController, RecipeController a příslušných pohledů – např. Edit.cshtml, Delete.cshtml, Create.cshtml.

Nutriční bilance (F6) a záznam aktivity (F9) jsou zobrazeny na hlavní stránce aplikace pomocí MainPageController.cs a pohledu Index.cshtml. Registrace a autentizace (požadavek F10 Uživatelské účty) jsou zajištěny v AuthController.cs.

Při vývoji byly respektovány i nefunkční požadavky, jako je přívětivost rozhraní (N1), použití architektury ASP.NET Core MVC (N2) a české jazykové prostředí (N3). Bezpečné ukládání hesel (N4) je zajištěno pomocí algoritmu SHA-256 (Secure Hash Algorithm), konkrétně metodou SHA256.Create() z knihovny System.Security.Cryptography. Heslo zadané při registraci je převedeno na hashovaný řetězec a uloženo v databázi v uživatelském atributu PasswordHash. Tato operace je realizována metodou SetPassword ve třídě User, která volá interní metodu CreatePasswordHash. Při přihlášení je ověření provedeno porovnáním aktuálně spočítaného hashe s uloženou hodnotou pomocí metody VerifyPassword.

7.2 Výpočet REE a cílových nutričních hodnot

Pro výpočet klidového energetického výdeje (REE) a následné rozdělení doporučeného příjmu energie mezi základní makroživiny byla navržena a implementována samostatná třída CaloriesCalculator.cs.

Kalkulace REE je založena na rovnici Mifflin-St Jeor (Rovnice 3 a Rovnice 4). Metoda využívá jako vstupní parametry výšku, hmotnost, pohlaví a věk uživatele. Následuje ukázka kódu implementace kalkulace REE.

```
public double CalculateREE(double weight, double height, DateTime dateOfBirth,
Gender gender)
        {
            double REE = 0;
            double age = (DateTime.Today - dateOfBirth).TotalDays / 365.25;
            if (weight <= 0 || height <= 0 || age > 150)
            {
                  throw new ArgumentException("Weight and height must be positive
values greater than zero. Age must be lower than 150 years.");
            }
            switch (gender)
            {
                case Gender.Male:
                    REE = (10 * weight) + (6.25 * height) - (5 * age) + 5;
                    break;
                case Gender.Female:
                    REE = (10 * weight) + (6.25 * height) - (5 * age) - 161;
                    break;
            }
            return REE;
        }
```

Rozdělení celkové energetické potřeby mezi jednotlivé makroživiny je realizováno v metodě CalculateMacroNutrients. Při výpočtu jsou zohledněny energetické hodnoty jednotlivých složek – 4 kcal/g u bílkovin a sacharidů a 9 kcal/g u tuků, jak bylo uvedeno v kapitole 3.1 Kalorický příjem a bazální metabolismus. Poměrové rozdělení příjmu (25 % bílkoviny, 25 % tuky, 50 % sacharidy) rovněž vychází z doporučení uvedených v této kapitole. Výsledné hodnoty slouží jako vstupní kritéria pro další moduly systému.

```
public void CalculateMacroNutrients(double calories, out double protein, out double
carb, out double fat)
{
    protein = (calories * ProteinRatio) / ProteinCaloriesPerGram;
    carb = (calories * CarbRatio) / CarbCaloriesPerGram;
    fat = (calories * FatRatio) / FatCaloriesPerGram;
}
```

7.3 Hodnocení jídelníčku

Metoda EvaluateMealPlan ze statické třídy MealPlanEvaluator.cs slouží jako nástroj pro numerické zhodnocení kvality naplánovaného jídelníčku. Jejím cílem je kvantifikovat, jak dobře daný návrh jídelníčku odpovídá zbývajícím nutričním potřebám uživatele. Výsledkem je číselné skóre, přičemž nižší hodnota značí vyšší kvalitu doporučení.

Výpočet skóre je založen na relativní odchylce skutečně navrženého příjmu kalorií, sacharidů, bílkovin a tuků od požadovaných hodnot. Bílkoviny jsou při výpočtu váženy trojnásobně a sacharidy dvojnásobně, protože se při testování ukázalo, že zvýšená váha této složky vede k vhodnějším výsledkům. Aby se zabránilo dělení nulou (například v případě, že uživatel již naplnil denní limit určité makroživiny), je ve jmenovateli použita malá konstanta $\varepsilon = 10^{-6}$.

Tato funkce plní roli užitkové funkce ve smyslu doporučovacích systémů založených na užitku (utility-based), jak bylo popsáno v kapitole 1.3.3 Modely založené na znalostech. Pomáhá porovnat různé návrhy a vybrat takový, který se co nejvíce blíží nutričním cílům uživatele.

```
public static double EvaluatePlan(List<MealPortion> dailyMeals, double
remainingCalories, double remainingCarbs, double remainingProteins, double
remainingFats)
 {
     double consumedProteins = 0;
     double consumedFats = 0;
     double consumedCarbs = 0;
     double consumedCalories = 0;
     foreach (var meal in dailyMeals)
     {
         consumedProteins += meal.Portion * meal.Recipe.TotalProteins;
         consumedFats += meal.Portion * meal.Recipe.TotalFats;
         consumedCarbs += meal.Portion * meal.Recipe.TotalCarbs;
         consumedCalories += meal.Portion * meal.Recipe.TotalCalories;
     }
        double value = ((Math.Abs(remainingCalories - consumedCalories)) /
Math.Max(remainingCalories, Epsilon)) +
          2 * ((Math.Abs(remainingCarbs - consumedCarbs)) / Math.Max(remainingCarbs,
Epsilon)) +
                 3 * ((Math.Abs(remainingProteins - consumedProteins)) /
Math.Max(remainingProteins, Epsilon)) +
              ((Math.Abs(remainingFats - consumedFats)) / Math.Max(remainingFats,
Epsilon));

     return value;
 }
```

7.4 Návrh jídelníčku

Třída RecommendMealPlan.cs je zodpovědná za návrh jídelníčku pro zvolený počet jídel. Při inicializaci nastavuje poměry jednotlivých chodů a připravuje filtry pro výběr vhodných receptů z databáze. Ve třídě je implementována metoda GenerateOptimizedMealPlan. Ta využívá algoritmus simulovaného žíhání pro hledání nejvhodnější kombinace pokrmů, jež odpovídá zbylému kalorickému a nutričnímu cíli uživatele. Tato optimalizační technika je popsána v kapitole 2.6.2 Stochastické metody.

Algoritmus pracuje s následujícími parametry: počáteční teplota 600, rychlost ochlazování 0,97 a 600 iterací. Tyto hodnoty byly stanoveny experimentálně a zůstávají ve všech případech pevně nastavené. Následující úsek zachycuje zjednodušenou strukturu metody GenerateOptimizedMealPlan. Jednotlivé bloky jsou nahrazeny komentáři, které vystihují jejich účel.

```
public List<MealPortion> GenerateOptimizedMealPlan(User user, List<MealPortion>
existingMeals, DateTime date)
 {
     // Výpočet zbývajících kalorií a nutričních hodnot
     // Generování náhodného počátečního jídelníčku

     double temperature = 600.0;
     double coolingRate = 0.97;
     int iterations = 600;

     for (int i = 0; i < iterations; i++)
     {
          // Penalizace duplicit v jídelníčku a výrazných odchylek od očekávaných
velikostí porcí
         // Ohodnocení původního jídelníčku
         // Ohodnocení nového jídelníčku

           if (newScore < currentScore || Math.Exp((currentScore - newScore) /
temperature) > rand.NextDouble())
         {
             currentPlan = newPlan;
         }

         temperature *= coolingRate;
     }
 }
```

Metoda přijímá jako vstup uživatele, existující jídelníček a datum, pro které má být plán sestaven. Na základě těchto vstupů spočítá zbývající denní potřebu kalorií a jednotlivých makroživin. Výstupem je seznam MealPortion, obsahující doporučené recepty a jejich porce.

Nejprve se z celkového denního cíle uživatele odečte již zaznamenaný příjem a vypočítají se jeho zbývající kalorie a nutriční hodnoty. Tyto hodnoty určují cílový stav, ke kterému má optimalizace směřovat. Následně je vygenerován počáteční jídelníček. Tam, kde již pokrm existuje, zůstává zachován, jinak se vloží náhodný recept s náhodnou velikostí porce.

V každé iteraci algoritmus vytvoří kopii aktuálního jídelníčku, ve které nahradí právě jeden pokrm jiným receptem a porcí. Nahrazovány jsou pouze volné pozice. Parametry algoritmu byly zvoleny tak, aby zajistily dobrý poměr mezi kvalitou výsledků a výpočetním časem.

Obě varianty jídelníčku jsou vyhodnoceny pomocí funkce EvaluatePlan ze třídy MealPlanEvaluator. Ke skóre se přičítají penalizace za opakující se recepty a za výrazné odchylky od očekávané velikosti porce pro daný typ jídla.

Nový návrh je přijat, pokud je lepší než předchozí. Algoritmus však s určitou pravděpodobností závislou na teplotě (dle vzorce žíhání) přijímá i horší řešení, což umožňuje překonat lokální minima.

Po každé iteraci se sníží teplota, čímž klesá pravděpodobnost přijetí horších řešení. Na konci hledání tak algoritmus konverguje k nejlepšímu nalezenému jídelníčku.

7.5 Datová vrstva a přístup k informacím

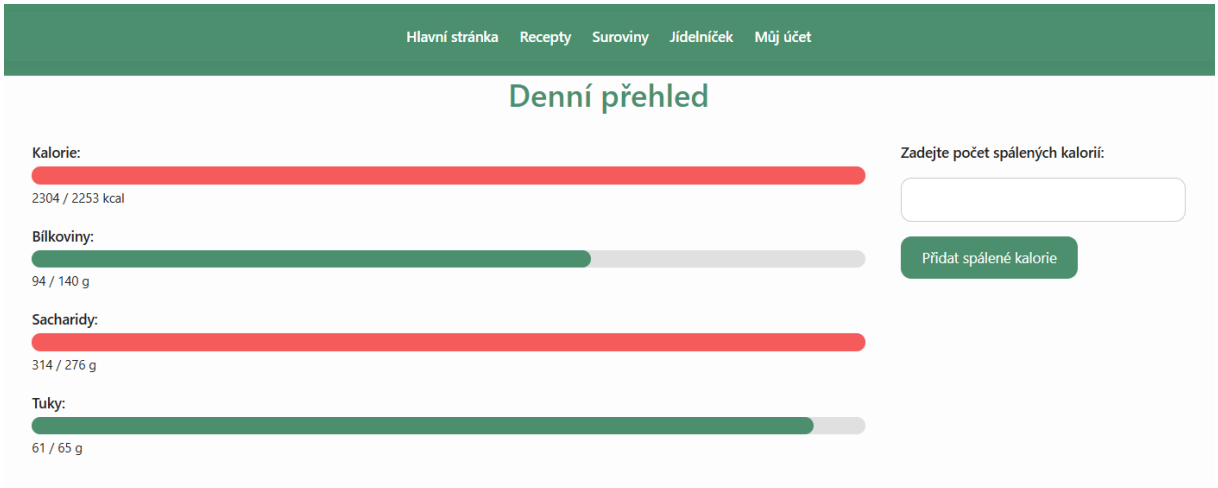
Systém využívá dokumentově orientovanou databázi LiteDB, která umožňuje perzistentní uložení dat do lokálního souboru NutritionalBalance.db ve formátu BSON. Datová vrstva je navržena s důrazem na jednoduchost a přehlednost, přičemž přístup ke kolekcím Users, Recipes, Ingredients, MealPlans je realizován prostřednictvím LINQ dotazů. Datový přístup je zajištěn pomocí třídy DatabaseService.cs.

Třída je zodpovědná za otevření databázového souboru a poskytuje přístup ke kolekcím jednotlivých entit pomocí metody GetCollection<T>. Zároveň obsahuje pomocné metody pro získávání konkrétních záznamů, aktualizaci uživatelského příjmu živin nebo manipulaci s jídelníčkem.

8 UŽIVATELSKÉ ROZHRANÍ A OVLÁDÁNÍ

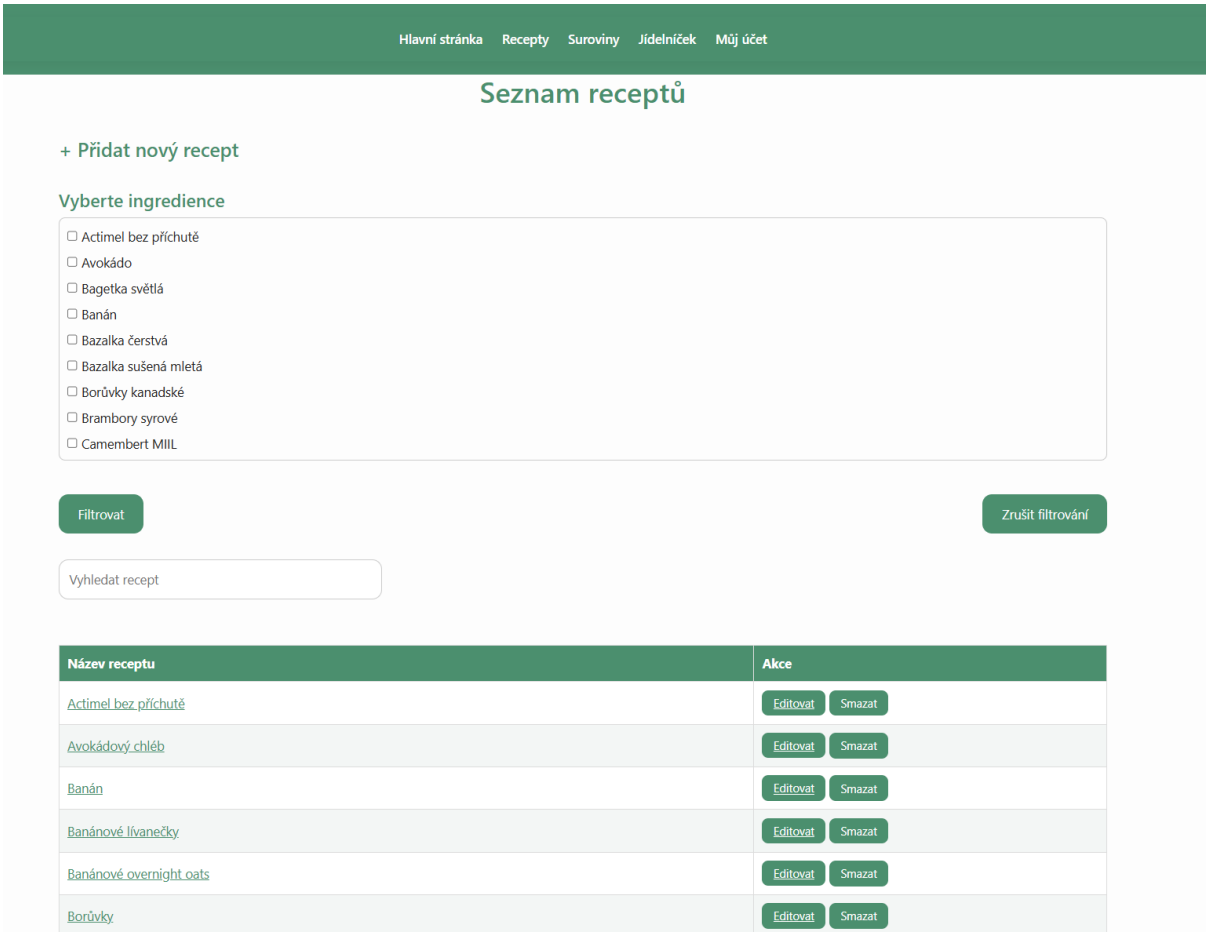
Aplikace je navržena jako webová aplikace v českém jazyce, určená k ovládání prostřednictvím běžných prvků, jako jsou formuláře, tlačítka nebo výběrová pole. Rozhraní je navrženo tak, aby bylo přehledné, intuitivní a vhodné i pro každodenní použití. Jednotlivé funkce jsou rozděleny do tematických sekcí, které jsou dostupné z navigační lišty.

Před použitím aplikace je nutné se přihlásit pomocí uživatelského jména a hesla, případně si vytvořit nový účet. Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku (Obrázek 10). Zde je graficky zobrazen jeho denní příjem energie a makroživin. V případě překročení některé hodnoty se příslušný ukazatel zbarví červeně. Součástí přehledu je také možnost zaznamenat spálené kalorie. Všechny hodnoty se každý den automaticky vynulují.



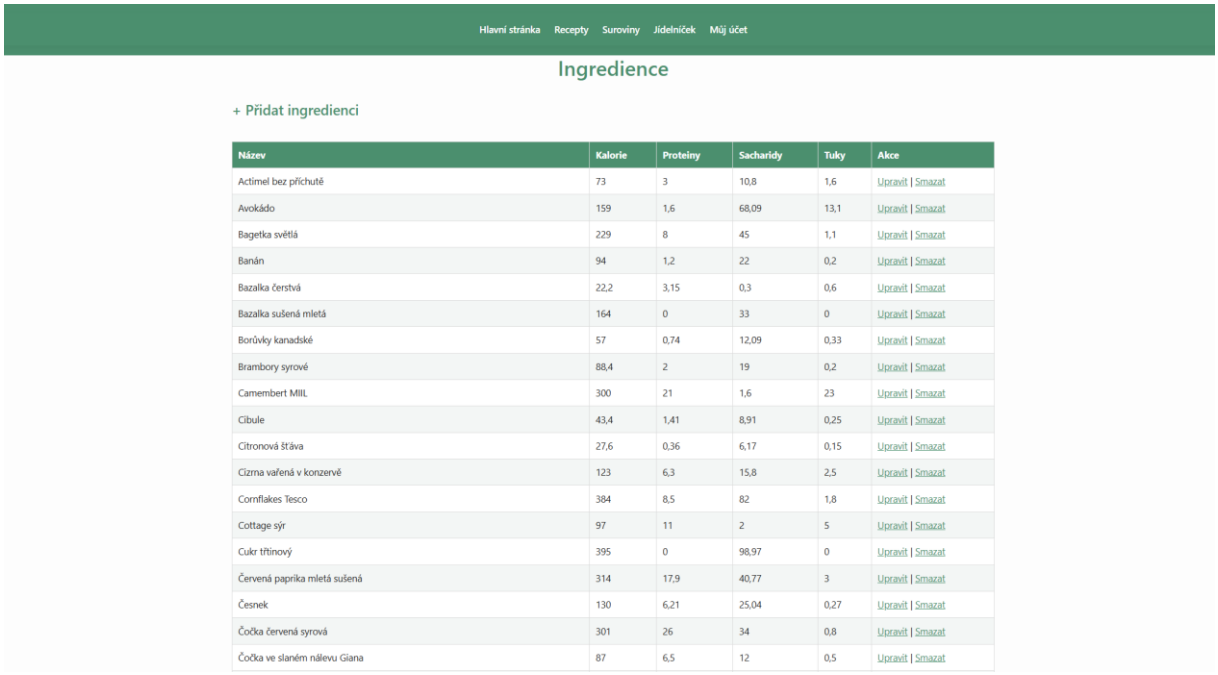
Obrázek 10: Denní přehled (zdroj: vlastní)

Sekce Recepty (Obrázek 11) obsahuje seznam všech dostupných receptů, které může uživatel upravovat nebo přidávat nové. Při tvorbě receptu zadává název, popis (postup), seznam ingrediencí s množstvím a také označení, pro jaké jídlo je recept vhodný (snídaně, svačina, oběd, večeře či kombinace). Recepty lze vyhledávat podle názvu a filtrovat podle dostupných surovin. Filtrování zobrazí pouze ty recepty, na které má uživatel všechny potřebné ingredience.



Obrázek 11: Seznam receptů (zdroj: vlastní)

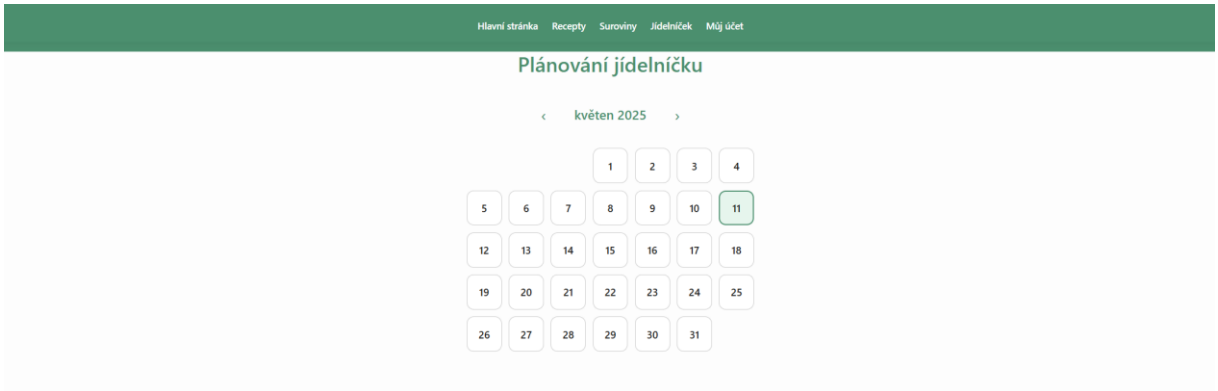
Na kartě Suroviny (Obrázek 12) se nachází přehled všech ingrediencí, které je také možné přidávat, editovat či mazat. Při tvorbě nové ingredience je nutné zadat její název a nutriční hodnoty vztažené na sto gramů.



Název	Kalorie	Proteiny	Sacharidy	Tuky	Akce
Actimel bez příchutě	73	3	10,8	1,6	Upravit | Smazat
Avokádo	159	1,6	68,09	13,1	Upravit | Smazat
Bagetka světlá	229	8	45	1,1	Upravit | Smazat
Banán	94	1,2	22	0,2	Upravit | Smazat
Bazalka čerstvá	22,2	3,15	0,3	0,6	Upravit | Smazat
Bazalka sušená mletá	164	0	33	0	Upravit | Smazat
Borůvky kanadské	57	0,74	12,09	0,33	Upravit | Smazat
Brambory syrové	88,4	2	19	0,2	Upravit | Smazat
Camembert MIIL	300	21	1,6	23	Upravit | Smazat
Cibule	43,4	1,41	8,91	0,25	Upravit | Smazat
Citronová šťáva	27,6	0,36	6,17	0,15	Upravit | Smazat
Cizrna vařená v konzervě	123	6,3	15,8	2,5	Upravit | Smazat
Cornflakes Tesco	384	8,5	82	1,8	Upravit | Smazat
Cottage sýr	97	11	2	5	Upravit | Smazat
Cukr třtinový	395	0	98,97	0	Upravit | Smazat
Červená paprika mletá sušená	314	17,9	40,77	3	Upravit | Smazat
Česnek	130	6,21	25,04	0,27	Upravit | Smazat
Čočka červená syrová	301	26	34	0,8	Upravit | Smazat
Čočka ve slaném nálevu Giana	87	6,5	12	0,5	Upravit | Smazat

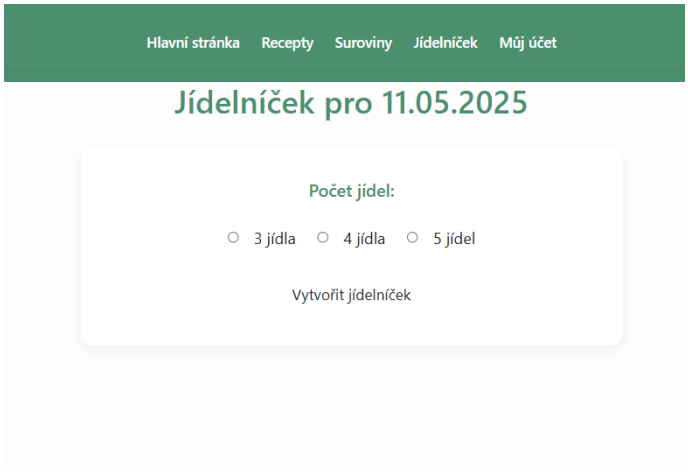
Obrázek 12: Suroviny (zdroj: vlastní)

Plánování jídelníčku probíhá na kartě Jídelníček (Obrázek 13). Zde je zobrazen kalendář, ve kterém si uživatel vybere konkrétní den, pro chce plán vytvořit. Aktuální datum je zvýrazněno zeleně, mezi měsíci lze přepínat pomocí šipek v horní části.



Obrázek 13: Plánování jídelníčku (zdroj: vlastní)

Pokud ještě jídelníček pro zvolený den neexistuje, zobrazí se výzva k zadání počtu plánovaných jídel (Obrázek 14).

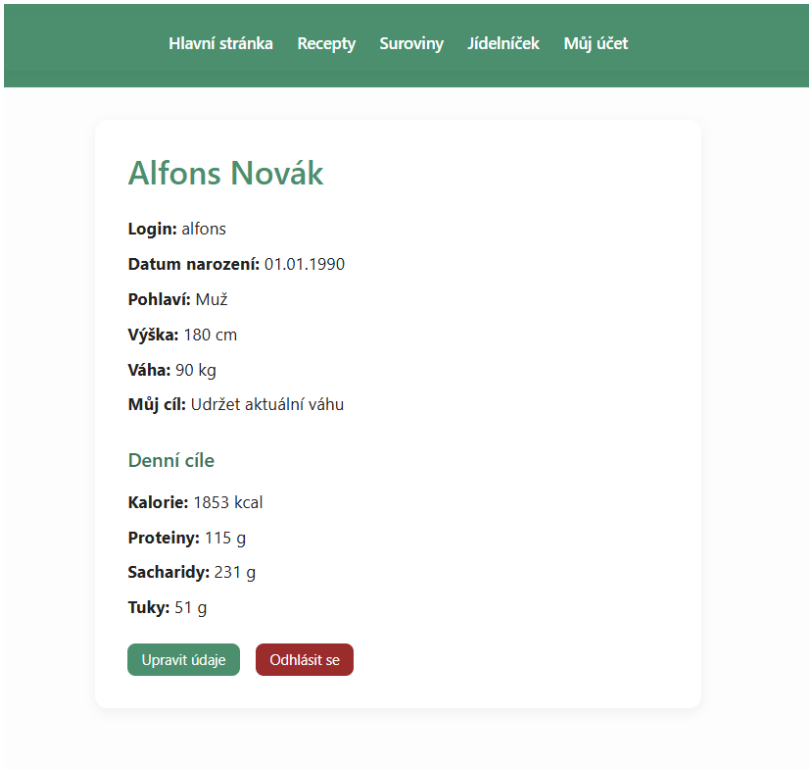


Obrázek 14: Tvorba jídelníčku (zdroj: vlastní)

Následuje zobrazení tabulky s denním plánem (Obrázek 15). Pokrmy lze přidávat ručně, nebo využít tlačítko pro automatické generování. Při opakovaném stisknutí tlačítka se aktualizují pouze ta jídla, která nebyla označena jako snědená. Takto označené pokrmy se promítnou do výživového přehledu, přičemž označit je lze pouze v den, kdy mají být zkonzumovány. Jednotlivé recepty lze automaticky měnit tlačítkem Alternativa, které spustí nový běh algoritmu simulovaného žíhání, tentokrát však omezený pouze na zvolený pokrm. Podrobný postup a složení receptu (přepočítané na počet porcí) se zobrazí po kliknutí na název pokrmu.

Obrázek 15: Jídelníček pro zvolený den (zdroj: vlastní)

V záložce Můj účet jsou zobrazeny osobní údaje uživatele (Obrázek 16). Zde lze upravit zadané informace nebo se odhlásit.



Obrázek 16: Můj účet (zdroj: vlastní)

ZÁVĚR

Bakalářská práce byla zaměřena na návrh a realizaci webové aplikace pro doporučování jídel podle aktuální nutriční bilance uživatele. Cílem bylo vytvořit funkční systém, který bude schopen sledovat denní příjem kalorií a makroživin a na základě toho generovat personalizovaný jídelníček s ohledem na nutriční cíle uživatele.

V první kapitole teoretické části byly popsány principy doporučovacích systémů, jejich základní typy a principy fungování. Další kapitola se věnovala metodám optimalizace, klasifikaci optimalizačních problémů a jednotlivým optimalizačním metodám. V následující kapitole byly probrány základy výživy a nutriční bilance, kalorický příjem, výpočet bazálního metabolismu a klidového energetického výdeje a jednotlivé makroživiny. V poslední kapitole teoretické části byly analyzovány vybrané existující aplikace pro plánování stravy a hodnoceny jejich funkční možnosti.

Praktická část se věnovala popisu použitých technologií, návrhu a implementace. Aplikace byla vyvinuta v jazyce C# s využitím architektury ASP.NET Core MVC. V rámci návrhu byly stanoveny funkční a nefunkční požadavky, vytvořeny diagramy případů užití a hlavních entit a navrženo uživatelské rozhraní.

Součástí implementace je výpočtová logika pro odhad klidového energetického výdeje (REE) podle rovnice Mifflin-St Jeor a stanovení doporučeného denního příjmu makroživin. Pro generování jídelníčku byl vytvořen optimalizační modul založený na algoritmu simulovaného žíhání. Datová vrstva byla realizována pomocí dokumentové databáze LiteDB a přístup k datům je zajištěn prostřednictvím LINQ.

Aplikace představuje funkční řešení základního doporučovacího systému, které je možné dále rozvíjet. Mezi potenciální směry rozšíření patří podpora sledování dalších živin (například vitamínů a vlákniny), zavedení pokročilých uživatelských preferencí (například potravinových intolerancí), možnost plánování stejného jídelníčku pro více osob najednou, možnost exportu seznamu surovin potřebných na jeden den, nastavení vlastních nutričních cílů či nasazení systému do provozu formou veřejné webové služby.

POUŽITÁ LITERATURA

[1] RICCI, Francesco; ROKACH, Lior a SHAPIRA, Bracha (ed.). *Recommender Systems Handbook*. Online. 3rd ed. New York, NY: Springer, c2022. ISBN 978-1-0716-2197-4. Dostupné z: https://doi.org/10.1007/978-1-0716-2197-4. [cit. 2025-04-12].

[2] AGGARWAL, Charu C. *Recommender systems*. Online. Cham: Springer, c2016. ISBN 978-3-319-29659-3. Dostupné z: https://doi.org/10.1007/978-3-319-29659-3. [cit. 2025-04-11].

[3] RAO, Singiresu S. *Engineering optimization: theory and practice*. 4th ed. New York: Wiley, c2009. ISBN 978-0-470-18352-6.

[4] NOCEDAL, Jorge a WRIGHT, Stephen J. *Numerical optimization*. Second edition. Springer series in operations research and financial engineering. New York: Springer, [2006]. ISBN 0-387-30303-0.

[5] YANG, Xin-She. *Engineering Optimization: An Introduction with Metaheuristic Applications*. Hoboken: Wiley, c2011. ISBN 978-0-470-58246-6.

[6] RUSSELL, Stuart J. a NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 2nd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall/Pearson Education, c2003. ISBN 0-13-790395-2.

[7] KŘIVAN, Miloš. *Umělé neuronové sítě*. Oeconomica, 2021. ISBN 978-80-245-2420-7.

[8] GRASMAN, Scott E a GOSAVI, Abhijit. Operations Research. In: *Engineering Management Handbook*. American Society for Engineering Management (ASEM), 2010, s. 65-66. ISBN 0983100500.

[9] ZIMMERMAN, Maureen a SNOW, Beth. *Essentials of Nutrition: A Functional Approach*. Online. V. 1.0. [2012]. ISBN 978-1-4533-5247-2. Dostupné z: https://2012books.lardbucket.org/books/an-introduction-to-nutrition/index.html. [cit. 2025-04-19].

[10] WORLD HEALTH ORGANIZATION [WHO]. *Healthy diet*. Online. World Health Organization. 29 April 2020. Dostupné z: https://www.who.int/news-room/fact-sheets/detail/healthy-diet. [cit. 2025-04-19].

[11] GROPPER, Sareen S.; SMITH, Jack L. a GROFF, James L. *Advanced Nutrition and Human Metabolism*. Online. 5th ed. Belmont: Wadsworth, Cengage Learning, c2009. ISBN 978-0-495-11657-8. Dostupné z: http://repository.poltekkes-kaltim.ac.id/id/eprint/1170. [cit. 2025-04-20].

[12] FRANKENFIELD, David; ROTH-YOUSEY, Lori a COMPHER, Charlene. Comparison of Predictive Equations for Resting Metabolic Rate in Healthy Nonobese and Obese Adults: A Systematic Review. Online. *Journal of the American Dietetic Association*. 2005, vol. 105, no. 5, s. 775-789. ISSN 0002-8223. Dostupné z: https://doi.org/10.1016/j.jada.2005.02.005. [cit. 2025-05-14].

[13] TRUMBO, Paula; SCHLICKER, Sandra; YATES, Allison A. a POOS, Mary. Dietary Reference Intakes for Energy, Carbohydrate, Fiber, Fat, Fatty Acids, Cholesterol, Protein and Amino Acids. Online. *Journal of the American Dietetic Association*. 2002, vol. 102, no. 11, s. 1621-1630. Dostupné z: https://doi.org/10.1016/S0002-8223(02)90346-9. [cit. 2025-05-14].

[14] EAT THIS MUCH, INC. *Eat This Much - Meal Planner*. Online. C2025. Dostupné z: https://www.eatthismuch.com/. [cit. 2025-04-21].

[15] CRONOMETER SOFTWARE INC. *Cronometer*. Online. C2011-2025. Dostupné z: https://cronometer.com/. [cit. 2025-04-21].

[16] YAZIO GMBH. *YAZIO*. Online. C2015. Dostupné z: https://www.yazio.com/. [cit. 2025-04-21].

[17] GRIFFITHS, Ian. *Programming C# 12: Build Cloud, Web, and Desktop Applications*. O'Reilly Media, c2024. ISBN 978-1-098-15836-1.

[18] MICROSOFT. *Overview of ASP.NET Core*. Online. MICROSOFT. Microsoft Learn. 04/21/2025. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0. [cit. 2025-05-04].

[19] SMITH, Steve. *Overview of ASP.NET Core MVC*. Online. MICROSOFT. Microsoft Learn. C2025, 06/17/2024. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-9.0. [cit. 2025-05-04].

[20] *Overview: LiteDB v5 - A .NET NoSQL Document Store in a single data file*. Online. LiteDB: Embedded NoSQL database for .NET. C2014-2022. Dostupné z: https://www.litedb.org/docs/. [cit. 2025-05-04].

[21] *Logo LiteDB*. Online. In: LiteDB. C2014-2022. Dostupné z: https://www.litedb.org/images/logo_litedb.svg. [cit. 2025-05-12].

SEZNAM PŘÍLOH