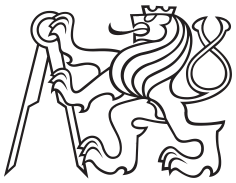
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Progresivní webová aplikace pro správu kalorického příjmu a výdeje

Adam Forgáč

Vedoucí: RNDr. Ondřej Žára
Obor: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Forgáč** Jméno: **Adam** Osobní číslo: **483822**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Progresivní webová aplikace pro správu kalorického příjmu a výdeje

Název bakalářské práce anglicky:

Progressive Web Application for Caloric Intake and Expenditure Management

Pokyny pro vypracování:

Nastudujte problematiku evidence kalorického příjmu jedince v průběhu času. Uvažte existující možnosti správy kalorického příjmu pomocí mobilních či webových aplikací.
Následně navrhněte a naimplementujte progresivní webovou aplikaci (PWA), která bude tento úkol řešit. Klíčové požadované vlastnosti:
- zadávání potravin a jejich kalorických hodnot
- zadávání aktivit a jejich kalorických hodnot
- možnost sdílení těchto údajů
- perzistence dat napříč více zařízeními (možno použít řešení BaaS třetí strany)
- možnost zadávání dat pomocí skenování čárového, případně QR kódu

Uvažte, jaká webová API z rodiny PWA jsou pro tento účel vhodná či klíčová. Popište, je-li nabízená množina technologií dostačující (v porovnání s nativní mobilní či hybridní aplikací). Výsledný produkt otestujte a vystavte veřejně na webu.

Seznam doporučené literatury:

https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
https://firebase.google.com/
https://www.ean-search.org/
https://serratus.github.io/quaggaJS/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ondřej Žára, Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

RNDr. Ondřej Žára	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Děkuji vedoucímu mé práce RNDr. Ondřeji Žárovi za jeho profesionální přístup, ochotu a cenné rady, které mi dal při průběžných konzultacích. Dále děkuji všem uživatelům, kteří se dobrovolně zúčastnili uživatelského testování aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 11. května 2021

Abstrakt

Cílem této práce bylo analyzovat problematiku evidence kalorického příjmu jedince v průběhu času a uvážit existující možnosti správy kalorického příjmu pomocí mobilních či webových aplikací. Dále bylo cílem navrhnout, implementovat a otestovat progresivní webovou aplikaci, která by tento úkol řešila. Aplikace se odlišuje od existujících řešení použitím platformy BaaS třetí strany, dostupností aplikace v offline režimu a umožněním zadávání dat pomocí skenování QR kódu. K automatizovanému testování aplikace je použita moderní knihovna Cypress.

Klíčová slova: kalorický příjem, progresivní webová aplikace, BaaS, QR kód, automatizované testování, Cypress

Vedoucí: RNDr. Ondřej Žára

Abstract

The aim of this thesis was to analyze the issue of recording the caloric intake of an individual over time and consider the existing options for managing caloric intake using mobile or web applications. Furthermore, the goal was to design, implement and test a progressive web application that would solve this task. The application differs from existing solutions by using a third-party BaaS platform, making the application available offline, and enabling data entry by scanning a QR code. A modern Cypress library is used for automated testing of the application.

Keywords: caloric intake, progressive web application, BaaS, QR code, automated testing, Cypress

Title translation: Progressive Web Application for Caloric Intake and Expenditure Management

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Správné fungování lidského organismu vyžaduje vyvážený příjem kalorií a živin. Nekontrolovaný poměr denního příjmu a výdeje kalorií může mít pro člověka nežádoucí účinky a v krajních případech může vést až k závažným zdravotním problémům. Záměrem této bakalářské práce je vytvořit progresivní webovou aplikaci pro správu kalorického příjmu a výdeje. Díky takové aplikaci bude mít jedinec lepší přehled o tom, v jaké míře přispívá konzumovanými potravinami do svého kalorického příjmu.

K výběru tohoto tématu mě přivedla skutečnost, že již několik let používám podobné aplikace pro správu kalorického příjmu a nejsem s nimi plně spokojen. Během používání těchto aplikací jsem zpozoroval mnoho jejich nedostatků a možností pro vylepšení. Vytvořená webová aplikace se bude od existujících řešení odlišovat použitím moderních webových rozhraní a také možností snadného sdílení informací o potravinách pomocí QR kódů.

V teoretické části práce se zaměřuji na analýzu problematiky evidence kalorického příjmu a definuji pojmy, s nimiž v textu později pracuji. Provádím rešerši existujících aplikací řešících danou problematiku a na jejím základě formuluji požadavky na vytvářenou aplikaci. Dále v této části představuji možnosti využití čárových kódů a ospravedlňuji výběr webových technologií pro tvorbu aplikace.

Praktická část je soustředěna na návrh aplikace, včetně vytvoření návrhů obrazovek. V této části také popisuji jednotlivé kroky implementace a podrobnosti o průběhu a výstupech testování aplikace. V závěru práce zmiňuji, do jaké míry se podařilo naplnit cíle stanovené v zadání a jaký bude další vývoj webové aplikace.

Část I

Teoretická část

Kapitola 2

Popis problematiky

V této kapitole popisuji problematiku sledování kalorického příjmu a výdeje. Definuji základní pojmy, se kterými budu v textu pracovat. Dále uvádím postupy a metody, které budou v následujících kapitolách nezbytné pro analýzu a návrh aplikace.

2.1 Seznámení s pojmy

V této sekci jsou stručně představeny důležité pojmy, které souvisí s tématem této bakalářské práce.

- *Nutriční hodnota* je informace, která udává přesný počet a složení živin v potravině. Mezi nejdůležitější živiny ke sledování patří bílkoviny, tuky a sacharidy. Tato trojice živin je běžně označována jako makroživiny[1].

- *Kalorie* je jednotkou energie. Z hlediska příjmu potravy jsou zdrojem kalorií všechny makroživiny (bílkoviny, tuky a sacharidy). Různé typy makroživin mají standardní počet kalorií. Pro bílkoviny a sacharidy jsou to 4 kalorie na jeden gram. Pro tuky to je 9 kalorií na jeden gram[2]. V praxi se běžně používá její násobek kilokalorie (značka *kcal*).

- *Webová aplikace* je počítačový program, který využívá webový prohlížeč. Obdobně jako desktopové a mobilní aplikace, webové aplikace disponují uživatelským rozhraním pro práci s daty[3].

- *Progresivní webové aplikace* (PWAs) jsou webové aplikace, které využívají rozhraní a funkce webového prohlížeče, aby přinesli nativní uživatelský zážitek nezávisle na platformě. Nejedná se o formální standard, ale spíše o návrhový vzor zahrnující použité atributy aplikace a webové technologie. Aby bylo možné označit webovou aplikaci jako PWA, měla by splňovat[4]:

  - komunikaci přes HTTPS (Hypertext Transfer Protocol Secure)
  - použití Service Workers
  - soubor MANIFEST

2.2 Sledování kalorického příjmu a výdeje

Preferovaný poměr mezi denním kalorickým příjmem a výdejem závisí na individuálních cílech jedince. Pro udržení stabilní tělesné hmotnosti je nutné mít vyvážený poměr denního příjmu kalorií a výdeje energie. Pokud je cílem zvýšení tělesné hmotnosti, pak je nutné navýšit denní příjem kalorií nebo snížit energetický výdej. Naopak pro hubnutí je stěžejní snížit denní příjem kalorií nebo případně navýšit energetický výdej.

2.2.1 Bazální metabolismus

Pro určení doporučeného denního příjmu kalorií je zásadní výpočet bazálního metabolismu BMR (z anglického Basal Metabolic Rate).

- *Bazální metabolismus* (značíme BMR) je počet kalorií, které lidské tělo potřebuje pro plnění základních životních funkcí, jako jsou dýchání, tělní oběh, zpracování živin a tvorba buněk. K výpočtu BMR se používá Harris-Benedictova rovnice [5].
  - Pro ženy:

$$BMR = 655 + (9.6 \times m) + (1.8 \times h) - (4.7 \times y)$$

  - Pro muže:

$$BMR = 66 + (13.7 \times m) + (5 \times h) - (6.8 \times y)$$

  kde:
  m = hmotnost jedince v kilogramech
  h = výška jedince v centimetrech
  y = věk jedince v letech

Vztah pro výpočet bazálního metabolismu bude v práci použit pro určení doporučeného denního příjmu kalorií. Vstupní hodnoty do Harris-Benedictovy rovnice budou zohledněny při návrhu datové infrastruktury.

2.2.2 Metabolický ekvivalent úkolu

Mezi důležité aspekty při sledování kalorického výdeje u prováděných aktivit patří index MET (metabolic equivalent of task, česky metabolický ekvivalent úkolu) pro objektivní posouzení výše energetické náročnosti.

- *Metabolický ekvivalent úkolu* (MET) lze označit jako poměr mezi rychlostí lidského metabolismu při fyzické námaze oproti rychlosti metabolismu v klidu. Jeden MET odpovídá zhruba 3.5 mililitrům kyslíku na kilogram tělesné hmotnosti za minutu. Lehké fyzické aktivity jako jsou pomalá chůze nebo práce u počítače odpovídají nanejvýš hodnotě 3 MET. Mezi středně náročné aktivity (3 až 6 MET) patří například rychlá chůze nebo

jóga. Těžkými fyzickými aktivitami s náročností přes 6 MET mohou být například běhání, plavání nebo basketbal. Pro odvození hodnoty indexu MET bývá nejčastěji používán následující vztah[6]:

$$1\ MET = 1\ \frac{kcal}{kg \times h} = 4.184\ \frac{kJ}{kg \times h} = 1.162\ \frac{W}{kg}$$

kde:
$kcal$ = kilokalorie
kg = kilogram
h = hodina
kJ = kilojoule
W = watt

Metabolický ekvivalent úkolu bude v práci použit pro určení náročnosti fyzických aktivit. Pro každého jedince bude index MET u dané aktivity přepočítán na výdej energie.

2.2.3 Doporučený poměr živin

Při sledování kalorického výdeje je rovněž důležité správné rozdělení makroživin ve stravě. Univerzální návod na určení tohoto poměru neexistuje. Obecně lze doporučit pouze přibližné procentuální zastoupení bílkovin, tuků a sacharidů v denním kalorickém příjmu. U průměrného dospělého člověka je zastoupení následující:

- 45 až 65 procent kalorií ze sacharidů
- 20 až 35 procent kalorií z tuků
- 10 až 35 procent kalorií z bílkovin

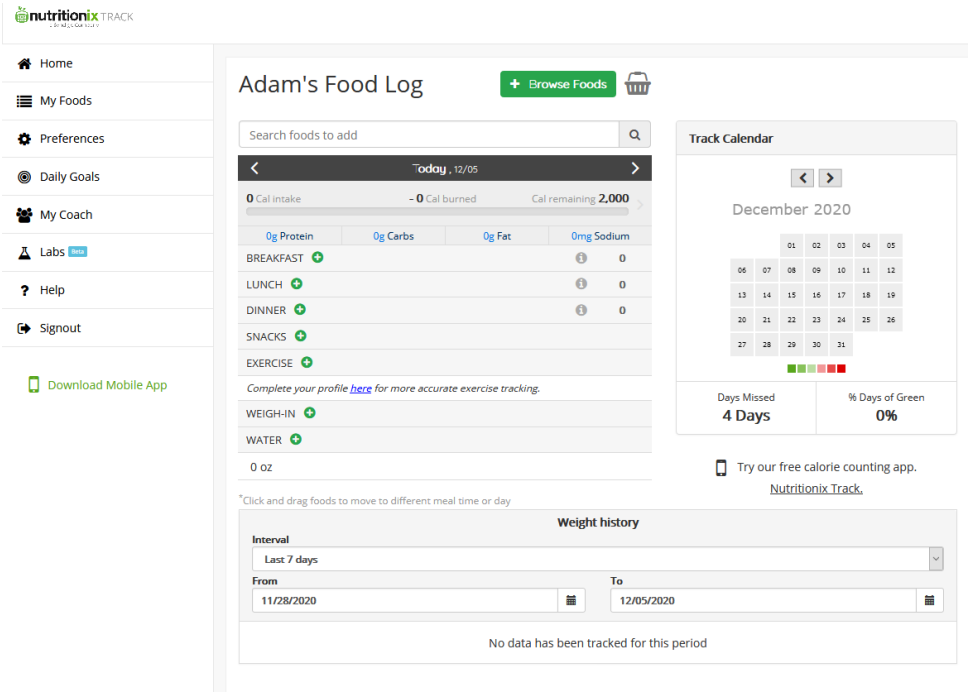
Přesné hodnoty se mohou lišit podle toho, jak často je člověk fyzicky aktivní[7].

Kapitola 3

Existující webové aplikace

V této kapitole se věnuji analýze existujících řešení. Konkrétně se jedná o přehled populárních webových aplikací, které slouží ke sledování kalorického příjmu a výdeje. Popisuji, jakým způsobem problematiku řeší a uvádím hlavní výhody a nevýhody daných řešení. K analýze technologií využívaných v uvedených aplikacích používám webovou službu BuildWith[8].

3.1 Nutritionix



Obrázek 3.1: Screenshot z aplikace Nutritionix, zdroj: [9]

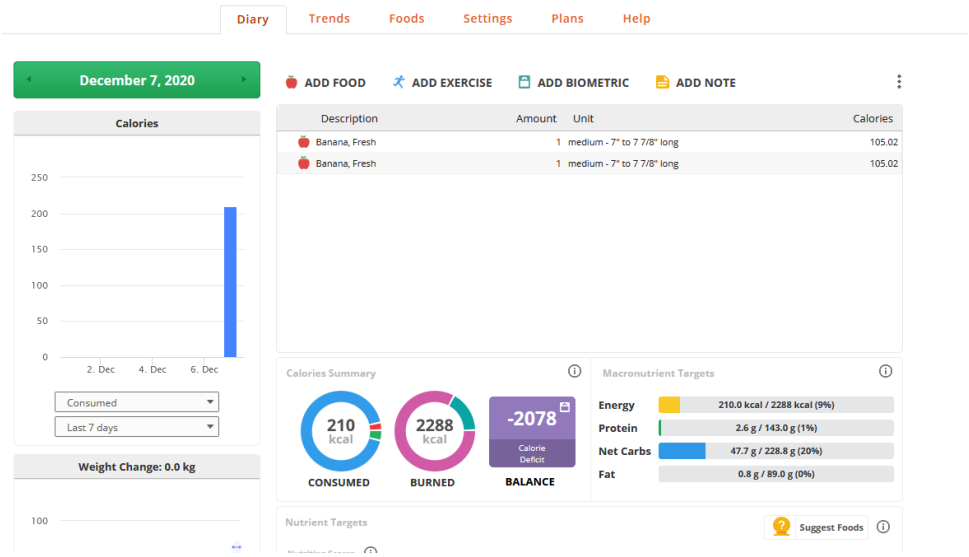
Nutritionix je volně dostupná webová aplikace, která disponuje největší databází potravin na světě. Aplikaci si je možné stáhnout v mobilních verzích pro operační systémy Android a iOS. Umožňuje vytvoření uživatelského

profilu pro sledování kalorického příjmu a výdeje. Pro majitele restaurací je v aplikaci možná tvorba vlastního účtu, přes který mohou vkládat do databáze aplikace pokrmy nabízené ve svých restauracích. Serverová část aplikace je z převážné části napsána v Node.js s použitím frameworku Express.js. Frontend aplikace je napsaný v JavaScriptu s použitím frameworku AngularJS.

Hlavními výhodami této aplikace je velmi rozsáhlá databáze jídel a vysoká přesnost uvedených nutričních hodnot. Odborníci na výživu považují aplikaci Nutritionix za vůbec nejdůvěryhodnější aplikaci pro sledování kalorického příjmu a výdeje na světě.

Nevýhodou je, že aplikace neumožňuje sdílení vytvořených pokrmů mezi dalšími uživateli. Vytvořená jídla je možné si zapsat pouze individuálně. Dalším problémem při používání aplikace je také fakt, že ačkoli je umožněno vyhledávání za pomoci naskenovaných EAN kódů potravin, tak pro české produkty nejsou v databázi žádné relevantní záznamy. Aplikace rovněž nedisponuje žádným offline režimem, a tudíž ji při ztrátě internetového připojení nelze používat.

3.2 Cronometer



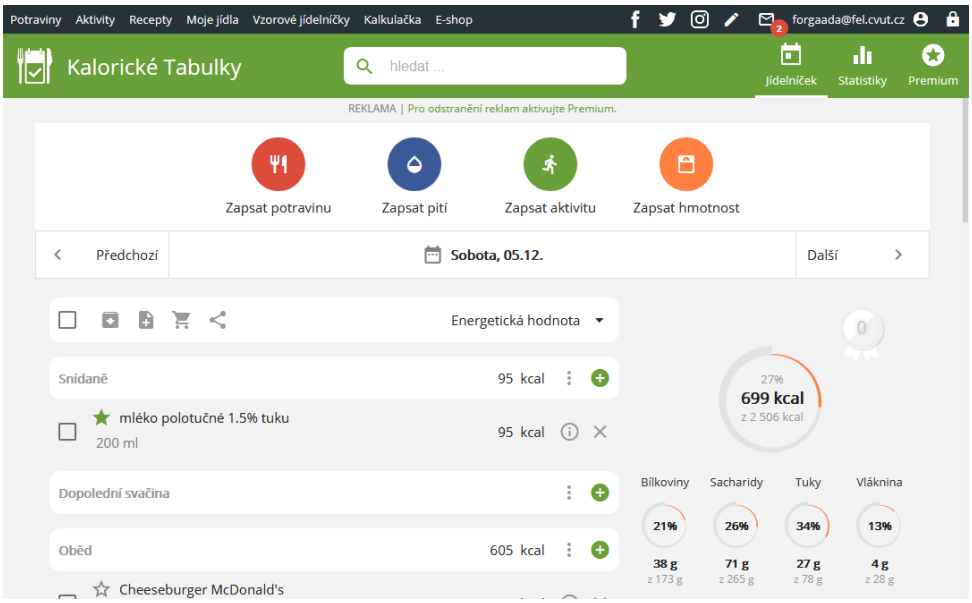
Obrázek 3.2: Screenshot z aplikace Cronometer, zdroj: [10]

Webová aplikace Cronometer patří k uznávaným nutričním aplikacím. Údaje o potravinách uložené v databázi jsou ověřovány odborníky. Na domovské stránce aplikace je možné vidět denní přehled a statistiky příjmu vykreslené do koláčových grafů. Frontend aplikace je napsaný v JavaScriptu s využitím knihovny jQuery. Aplikace běží na webovém serveru Apache.

Výhodou aplikace jsou chytré návrhy na úpravu jídelníčku na základě nedostatečného příjmu vybraných živin.

Hlavní nevýhodou aplikace jsou vysoké ceny zpoplatněných balíčků, které umožňují využití pokročilejších funkcí. Aplikace nemá k dispozici žádný offline režim.

3.3 Kalorické Tabulky



Obrázek 3.3: Screenshot z aplikace Kalorické Tabulky, zdroj: [11]

Na českém trhu jsou Kalorické Tabulky nejpoužívanější webovou aplikací pro sledování nutričních hodnot. Aplikace je dostupná v mobilní verzi pro systémy Android a iOS. Umožňuje vytváření a zápis potravin nebo aktivit. Na domovské stránce se nachází přehled zapsaných potravin vykreslený do koláčových grafů. Backend aplikace je napsán převážně v jazyce PHP. Frontend je napsaný v JavaScriptu s použitím knihoven jQuery, SwipeJS a frameworku AngularJS.

Hlavní výhodou aplikace je její česká lokalizace a také vlastní databáze jídel obsahující tradiční české pokrmy. Na rozdíl od předešlých aplikací je zde možnost sdílení vlastního jídelníčku mezi dalšími uživateli.

Nicméně i zde chybí možnost snadného sdílení přehledu o dané potravině. Lze přeposlat odkaz na danou potravinu, avšak bez možnosti další úpravy před zapsáním do denního příjmu.

3.4 Shrnutí

Z výčtu existujících webových aplikací je zřejmé, že již byla vytvořena komplexní řešení pokrývající problematiku zápisu nutričních hodnot. Některé z těchto aplikací využívají v mobilních verzích také vyhledávání za pomoci

EAN kódu na obalu potravin. Aplikace pracují s obdobnými strukturami pro ukládání záznamů o potravinách. Hlavní roli hrají především makronutrienty.

Žádná z uvedených aplikací neřeší možnost sdílení vytvořených pokrmů pro jiného uživatele. V konkrétních případech se při načítání dat ze serveru musí obnovit aktuální stránka (např. při zobrazení denního příjmu). Aplikace také nemají offline režim, který by zajistil alespoň částečnou použitelnost aplikace v případě ztráty internetového připojení. Tento problém by se dal vyřešit použitím platformy BaaS (Backend as a Service) a volbou vhodného cachovacího modelu.

Kapitola 4

Analýza využití čárových kódů pro účely webové aplikace

Čárové kódy umožňují za pomoci sekvencí čar a mezer s určenou šířkou vizuálně reprezentovat data, která mohou být posléze čtena skenery nebo čtečkami čárových kódů. V analýze existujících řešení v kapitole 3 je uvedeno, že některé webové aplikace umožňují vyhledávání potravin pomocí skenování EAN kódu. V této kapitole představím dva základní typy čárových kódů – 1D (lineární) čárový kód a 2D (dvoudimenzionální) kód. Stručně popíši vlastnosti obou z nich a provedu výběr vhodnějšího řešení pro použití v progresivní webové aplikaci.

4.1 1D čárový kód

1D čárové kódy lze běžně najít na etiketách spotřebního zboží. Tyto kódy jsou vhodné zejména pro komerční účely a kódování produktů. Čtečka tohoto typu čárového kódu čte pouze v jedné ose. Kvůli omezení na jednu osu se však užití těchto kódů stává nepraktické při zakódování řetězců s délkou přes 25 znaků. Mezi populární skupiny 1D čárových kódů se řadí UPC, Code-128 nebo EAN (European Article Number). EAN čárové kódy musejí být pro komerční využití registrovány.

Obrázek 4.1: Ukázka běžného EAN kódu, zdroj: [12]

4.2 2D čárový kód

Zatímco 1D čárový kód kóduje informace pouze v jedné ose, 2D čárový kód kóduje ve dvou osách. Díky tomu v sobě může uchovávat mnohonásobně více informací[13]. Nejpoužívanějším 2D čárovým kódem je QR kód (Quick

Response code). Jeho obliba vzrostla s příchodem chytrých telefonů a v dnešní době se běžně používá například pro sdílení odkazů na webové stránky.

Obrázek 4.2: Ukázka běžného QR kódu, zdroj: [12]

4.3 Shrnutí

Je zřejmé, že použití EAN kódů dává smysl pouze pro vyhledávání existujícího spotřebního zboží. Toto řešení neumožňuje volně vytvářet nové kódy pro vlastní potraviny. Na druhou stranu generování QR kódů je snadné a nabízí širokou škálu využití. Uživatelé mohou například vytvořit vlastní pokrmy a poté sdílet odkaz na jejich stránku za pomoci QR kódu. Jedná se o ideální způsob sdílení informací o pokrmu.

Pro účely této bakalářské práce budu tedy používat zadávání dat pomocí skenování QR kódu. Možnost zadávání dat formou EAN kódu zohledním při návrhu datové infrastruktury, ale nebudu aplikaci rozšiřovat o čtečku tohoto typu kódu.

Kapitola 5

Analýza požadavků

Některé požadavky na aplikaci již byly definovány požadovanými klíčovými vlastnostmi v zadání projektu. Mezi tyto požadavky patří umožnění zápisu potravin a aktivit do denního příjmu, perzistence dat napříč více zařízeními a možnost zadávání dat pomocí skenování QR kódu.

V této kapitole se věnuji podrobnější analýze požadavků na základě zadání projektu a analýzy existujících řešení (viz kapitola 3). Výsledný soubor požadavků je rozdělen na funkční a nefunkční požadavky. Veškeré požadavky obsahují jednoznačný identifikátor, název a slovní popis. Na základě funkčních požadavků poté uvádím možné případy užití z pohledu uživatele. Soubor požadavků i případy užití budou hodnotnými podklady při vývoji aplikace.

5.1 Funkční požadavky

Funkční požadavky říkají, co bude systém uživateli umožňovat. Jedná se o podklad při vývoji aplikace. Funkční požadavky rovněž mohou specifikovat nároky na autentizaci a autorizaci při přístupu k informacím[14]. V tabulce níže je uveden soubor funkčních požadavků na aplikaci.

ID	Název	Popis
FR01	Vytvoření nového účtu	Systém umožní uživateli vytvořit nový účet.
FR02	Přihlášení uživatele	Systém umožní uživateli přihlásit se k jeho účtu.
FR03	Odhlášení uživatele	Systém umožní uživateli se odhlásit.
FR04	Editování účtu	Systém umožní uživateli upravit informace o jeho účtu.
FR05	Vytvoření nového jídla	Systém umožní uživateli vytvořit nové jídlo.
FR06	Zobrazení detailu jídla	Systém umožní uživateli vyhledat a zobrazit detail vytvořeného jídla.
FR07	Editování jídla	Systém umožní uživateli editovat jím vytvořené jídlo.

FR08	Smazání jídla	Systém umožní uživateli smazat jím vytvořené jídlo.
FR09	Přidání jídla do denního příjmu	Systém umožní uživateli přidat vyhledané jídlo do denního příjmu v daném množství.
FR10	Editování jídla v denním příjmu	Systém umožní uživateli editovat jeho záznam jídla v denním příjmu.
FR11	Smazání jídla v denním příjmu	Systém umožní uživateli smazat jeho záznam jídla v denním příjmu.
FR12	Vytvoření nové aktivity	Systém umožní uživateli vytvořit novou aktivitu.
FR13	Zobrazení detailu aktivity	Systém umožní uživateli vyhledat a zobrazit detail vytvořené aktivity.
FR14	Smazání aktivity	Systém umožní uživateli smazat jím vytvořenou aktivitu.
FR15	Přidání aktivity do denního výdeje	Systém umožní uživateli přidat provedenou aktivitu do denního výdeje.
FR16	Editování aktivity v denním výdeji	Systém umožní uživateli editovat provedenou aktivitu v denním výdeji.
FR17	Smazání aktivity v denním výdeji	Systém umožní uživateli smazat provedenou aktivitu v denním výdeji.
FR18	Zobrazení denních příjmů	Systém umožní uživateli zobrazit grafy denních příjmů pro aktuální a uplynulé dny.
FR19	Hledání jídel pomocí QR kódu	Systém umožní uživateli hledání jídel pomocí skenování QR kódu.

Tabulka 5.1: Funkční požadavky

5.2 Nefunkční požadavky

Nefunkční požadavky se nevztahují přímo k funkcím systému. Kladou nároky na vlastnosti systému, jako jsou například dostupnost, použitelnost nebo zabezpečení. Mohou rovněž specifikovat standardy kvality kladené na proces vývoje[15]. V tabulce níže je uveden soubor nefunkčních požadavků na aplikaci.

ID	Název	Popis
NFR01	Použitelnost	Systém bude zobrazovat popisy jednotlivých sekcí, které uživateli usnadní jejich používání.
NFR02	Uživatelská přívětivost	Systém uživateli umožní snadnou navigaci mezi sekcemi za pomoci navigačního panelu. V sekci s denním přehledem budou umístěna tlačítka pro rychlý zápis jídel a aktivit.
NFR03	Offline režim	Systém bude možné používat v omezeném rozsahu i po ztrátě internetového připojení.
NFR04	Kompatibilita	Systém bude kompatibilní s prohlížeči Google Chrome a Mozilla Firefox.

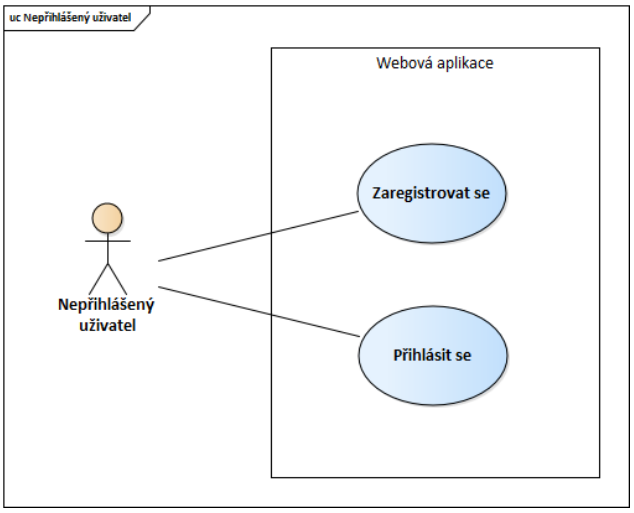
NFR05	Responzivní design	Systém se bude přizpůsobovat podle typu aktuálně používaného zařízení.

Tabulka 5.2: Nefunkční požadavky

5.3 Případy užití

Use Case Diagram (česky diagram případů užití) slouží k popisu chování systému očima uživatele. Diagram vychází z funkčních požadavků a definuje, co by měl systém umět. Jednotlivé diagramy se skládají z případů užití, aktérů a vztahů mezi nimi. V následující sekci uvádím diagramy případů užití vytvořené na základě funkčních požadavků z tabulky 5.1. Pro každý případ užití uvádím identifikátor, název, aktéry, kteří se případu užití účastní a krátký popis.

5.3.1 Nepřihlášený uživatel

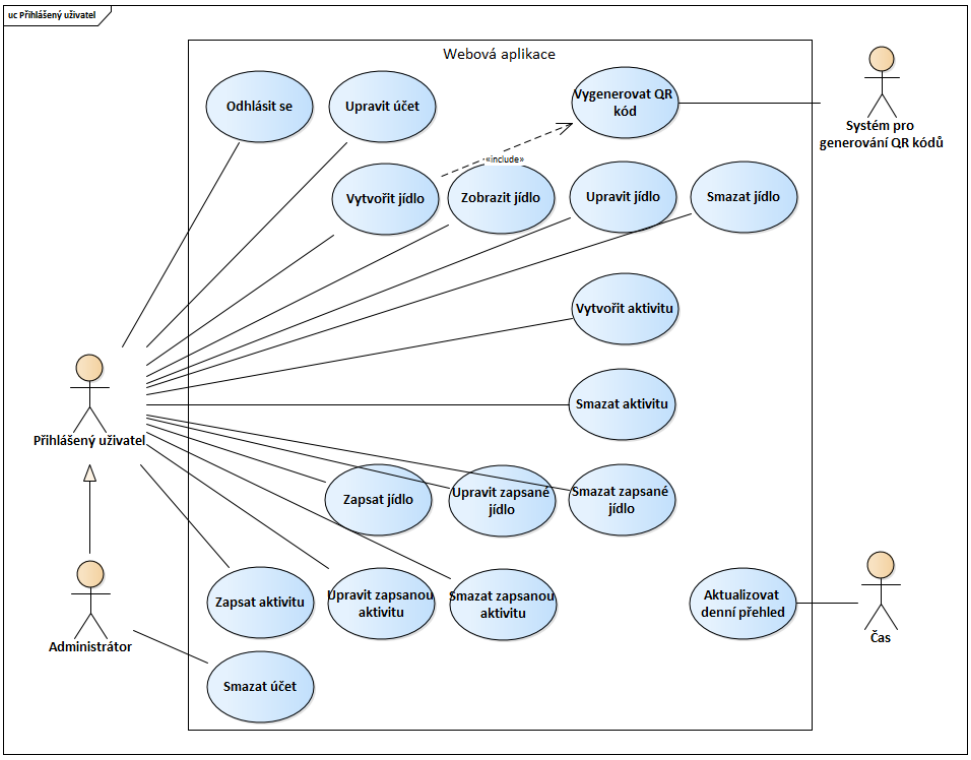


Obrázek 5.1: Use Case Diagram - Nepřihlášený uživatel

ID	Název	Aktér	Popis
UC1	Zaregistrovat se	Nepřihlášený uživatel	Uživatel vyplní a odešle registrační formulář obsahující jméno, email a heslo. Po úspěšné registraci je automaticky přihlášen a přesměrován do aplikace.
UC2	Přihlásit se	Nepřihlášený uživatel	Uživatel vyplní přihlašovací formulář a odešle jej. Po úspěšném přihlášení je přesměrován do aplikace.

Tabulka 5.3: Případy užití - Nepřihlášený uživatel

5.3.2 Přihlášený uživatel



Obrázek 5.2: Use Case Diagram - Přihlášený uživatel

ID	Název	Aktér	Popis
UC3	Odhlásit se	Přihlášený uživatel	Uživatel se za pomoci tlačítka „Odhlásit se“ odhlásí z účtu.
UC4	Upravit účet	Přihlášený uživatel	Uživatel vyplní tabulku s osobními údaji (jméno, rok narození, výška, hmotnost atd.) a klikne na tlačítko „Uložit“.
UC5	Vytvořit jídlo	Přihlášený uživatel	Uživatel vyplní tabulku s údaji o pokrmu (název, kalorie, bílkoviny, tuky, sacharidy atd.) a klikne na tlačítko „Uložit“.
UC6	Zobrazit jídlo	Přihlášený uživatel	Po přechodu na obrazovku s jídlem se uživateli zobrazí informace o jídle, QR kód jídla a tlačítka s možností „Zapsat“, „Upravit“ nebo „Smazat“ dané jídlo.
UC7	Upravit jídlo	Přihlášený uživatel	Oprávněný uživatel (autor pokrmu) upraví hodnoty v detailu pokrmu a klikne na tlačítko „Upravit“.
UC8	Smazat jídlo	Přihlášený uživatel	Oprávněný uživatel (autor pokrmu) klikne v detailu pokrmu na tlačítko „Smazat“.

UC9	Vytvořit aktivitu	Přihlášený uživatel	Uživatel vyplní tabulku pro uložení nové aktivity a klikne na „Uložit“.
UC10	Smazat aktivitu	Přihlášený uživatel	Oprávněný uživatel (autor aktivity) přejde na detail aktivity a klikne na „Smazat“.
UC11	Zapsat jídlo	Přihlášený uživatel	Uživatel si zobrazí detail pokrmu, zadá množství a klikne na „Zapsat“.
UC12	Upravit zapsané jídlo	Přihlášený uživatel	Uživatel v denním přehledu u detailu zapsaného jídla klikne na „Upravit“, zadá nové množství a klikne na „OK“.
UC13	Smazat zapsané jídlo	Přihlášený uživatel	Uživatel v denním přehledu u detailu zapsaného jídla klikne na „Smazat“.
UC14	Zapsat aktivitu	Přihlášený uživatel	Uživatel si zobrazí detail aktivity, klikne na „Zapsat“, zadá dobu trvání prováděné aktivity a klikne na „OK“.
UC15	Upravit zapsanou aktivitu	Přihlášený uživatel	Uživatel v denním přehledu u detailu zapsané aktivity klikne na „Upravit“, zadá novou dobu trvání a klikne na „OK“.
UC16	Smazat zapsanou aktivitu	Přihlášený uživatel	Uživatel v denním přehledu u detailu zapsané aktivity klikne na „Smazat“.
UC17	Aktualizovat denní přehled	Čas	Systém automaticky aktualizuje denní přehled uživatelů v 00:00. Zobrazí se pouze jídla a aktivity zapsané pro daný den.
UC18	Smazat účet	Administrátor	Administrátor odstraní účet vybraného uživatele ze systému.
UC19	Vygenerovat QR kód	Systém pro generování QR kódů	Při vytvoření nového jídla (viz UC5) systém pro generování QR kódů automaticky vytvoří k novému jídlu QR kód, který odkazuje na stránku s daným pokrmem.

Tabulka 5.4: Případy užití - Přihlášený uživatel

5.4 Shrnutí

V této kapitole jsem definoval požadavky na webovou aplikaci a na jejich základě uvedl možné případy užití. Soubor požadavků a diagramy případů užití budou potřebnými stavebními kameny v následujících kapitolách o návrhu a implementaci webové aplikace.

Kapitola 6

Výběr webových API z rodiny PWA

Předmětem této bakalářské práce je mimo jiné také uvážení a výběr webových API (Application Programming Interface) používaných při vývoji progresivních webových aplikací. V této kapitole uvedu tato webová rozhraní a rozhodnu o tom, která z nich budou použita při implementaci webové aplikace.

6.1 Service Worker API

Service Worker je nejdůležitější částí PWA. Umožňuje dosáhnout lepšího uživatelského zážitku při načítání stránky. Lze jej chápat jako vrstvu mezi sítí a prohlížečem, která se chová jako proxy server. Service Worker je samostatný skript, který sice nemá přístup k DOMu (Document Object Modelu), ale zato může používat jiná webová rozhraní za účelem perzistence dat [16]. Vytvoření Service Workeru je nedílnou součástí PWA, a tudíž bude muset být implementován.

6.2 Cache API

Cache API umožňuje ukládat a získávat síťové requesty a k nim korespondující odpovědi. Bylo vytvořeno za účelem poskytnutí rychlých odpovědí bez ohledu na připojení k internetu. Toto API může být rovněž použito jako úložiště dat[17]. Při použití Cache API je rovněž dobré zvolit vhodnou cachovací strategii. Zde uvádím příklady možných strategií[18]:

- Cache only – Jedná se o řešení vhodné zejména pro statické webové stránky. Veškeré odpovědi musejí být předem připraveny.
- Network only – Jak již plyne z názvu, toto řešení nijak neukládá odpovědi do cache, a tudíž v zásadě neumožňuje žádný offline režim.
- Network falling back to cache – Při použití této strategie Service Worker nejdříve přistupuje k internetu. Pokud je uživatel offline, tak Service Worker přistoupí k cache.

- Generic Fallback – U tohoto řešení Service Worker nejdříve přistupuje k cache a poté až k internetu. Pokud se mu nepodaří získat odpověď na daný request ani z jednoho zdroje, tak vystaví chybovou stránku.

6.3 Web Storage

Web Storage API umožňuje ukládání dat na klientu v podobě dvojic klíč – hodnota. Představuje mnohem intuitivnější řešení než například cookies. Web Storage API pracuje s dvěma mechanismy[19]:

- sessionStorage – Ukládá data zvlášť pro každou stránku, dokud nedojde k zavření okna prohlížeče.
- localStorage – V zásadě dělá to samé co sessionStorage, ale uložená data se nesmažou při zavření okna prohlížeče. Přístup k datům je možný za pomoci JavaScriptu.

6.4 IndexedDB

Rozhraní IndexedDB slouží jako úložiště na straně klienta. Umožňuje ukládání velkého množství dat včetně souborů. Na rozdíl od Web Storage API umožňuje IndexedDB ukládat strukturovaná data. Použití tohoto API je v porovnání s localStorage výrazně náročnější[20].

6.5 Push API

Mezi další používaná rozhraní při tvorbě PWA patří Push API. Tato technologie umožňuje přijímat a odesílat notifikace na server, a to i během toho, kdy webová aplikace běží na pozadí. Jedná se o užitečnou technologii při vývoji messagingových aplikací[21].

6.6 Shrnutí a výběr technologií

V této kapitole jsem stručně představil několik základních API pro vývoj PWA. Vzhledem k požadavkům definovaným v kapitole 5 bude vhodné vybrat taková rozhraní, která umožní snadno implementovat offline režim aplikace. Technologie Web Storage a IndexedDB jsou užitečné pro ukládání dat na straně klienta, avšak neřeší aktualizaci zdrojů a případné vystavení chybové stránky.

Pro tento účel jsem zvolil použití Cache v kombinaci s vhodnou cachovací strategií. Konkrétně se jedná o strategii Generic Fallback, která umožní pracovat s aplikací i bez připojení k internetu. Na rozdíl od ostatních strategií v případě nenalezení požadovaného zdroje zobrazí chybovou stránku. Další API z rodiny PWA pro účely tvorby webové aplikace pro správu kalorického příjmu nebudou zapotřebí.

Část II

Praktická část

Kapitola 7
Návrh řešení

V následující kapitole se zaměřuji na výběr technologií a návrh webové aplikace. Zvolené technologie budou použity později při implementaci. Na základě vytvořených návrhů obrazovek bude určeno rozložení komponent aplikace.

7.1 Architektura webové aplikace

Z důvodu úspory času a požadavku na možnost perzistence dat napříč více zařízeními jsem si jako architekturu webové aplikace zvolil tzv. Serverless Architecture. Tento způsob řešení je ideální při tvorbě PWA a umožňuje se při implementaci zaměřit pouze na tvorbu klientské aplikace a neřešit záležitosti týkající se provozu serverů. Toto řešení také výrazně usnadní proces nasazení aplikace a je vhodné i pro vývoj mobilních aplikací.

7.2 Backend

Serverless architektura pracuje s dvěma platformami pro správu serverové části:

- FaaS (Function as a Service) – Tato platforma dovoluje vývojáři nahrávat na cloud samostatné funkcionality a spouštět je jednotlivě. Je oblíbená při vývoji aplikací formou mikroslužeb (microservices).

- BaaS (Backend as a Service) – Tento model umožňuje plně oddělit správu serverové části (správa databáze, cloudové úložiště, hosting, autentizace, atd.). Vývojář se může plně zaměřit na tvorbu klientské aplikace.

Vzhledem k tomu, že implementace mikroslužeb je poměrně náročná a já osobně s ní nemám žádné zkušenosti, tak je pro mě platforma BaaS jistou volbou. V této sekci se věnuji srovnání používaných BaaS platforem. Uvádím jejich stručný popis a výčet zajímavých funkcionalit. V závěru sekce je uvedeno shrnutí a výběr platformy, kterou použiji při tvorbě webové aplikace.

7.2.1 Firebase

Google Firebase je populární BaaS platforma pro vývoj mobilních a webových aplikací. Jedná se o velmi bezpečnou a výkonnou platformu, která nabízí dobrou škálovatelnost při růstu aplikace. Jedinečnou vlastností Firebase je její realtime databáze Cloud Firestore (dříve Firebase Realtime Database). Firebase je dostupná v bezplatné verzi. Umožňuje také autentizaci, hosting a ukládání na webovém úložišti[22].

7.2.2 AWS Amplify

Jedná se o BaaS platformu od společnosti Amazon, která poskytuje služby pro snadný vývoj a nasazení webových i mobilních aplikací. Nedisponuje bezplatnou verzí, ale poskytuje bohaté API, možnost autentizace, sběr analytických dat a má nativní podporu pro AWS Cloud[23].

7.2.3 Apache Usergrid

Platforma Apache Usergrid je určena především pro vývoj mobilních aplikací. Nabízí škálovatelné datové úložiště a NoSQL databázi. Apache Usergrid používá formát JSON a poskytuje funkcionality, jako jsou správa uživatelů, zasílání notifikací nebo geolokace. Platforma je bezplatně dostupná a je open source již od roku 2011[24].

7.2.4 Back4App

Platforma Back4App se skládá z několika open source technologií, jako jsou NodeJS, Parse Server nebo populární NoSQL databáze MongoDB. Back4App je bezplatně dostupná a poskytuje REST a GraphQL API, možnost autentizace a zálohování dat[25].

7.2.5 Shrnutí a výběr technologie

Při výběru BaaS platformy pro účely této bakalářské práce jsem stanovil následující klíčové požadavky:

- dostupnost bezplatného režimu
- možnost hostingu webové aplikace
- možnost autentizace uživatelů

Z uvedených technologií byli nejvhodnějšími kandidáty populární platformy Google Firebase a Back4App.

Rozhodl jsem se pro použití platformy Firebase. Rozhodnutí jsem učinil na základě pozitivnějších ohlasů ze strany uživatelů a skutečnosti, že společnost Google poskytuje rozsáhlé návody pro práci s touto technologií. Firebase rovněž umožňuje navěšování databázových posluchačů na datové kolekce, díky kterým lze snadněji implementovat offline režim aplikace.

7.3 Frontend

Při implementaci progresivních webových aplikací se běžně používají technologie HTML, CSS a JavaScript. Pro tvorbu klientské aplikace se tedy nabízí použití čistého JavaScriptu (Vanilla JS) anebo některého z populárních JavaScriptových frameworků, jako jsou například Angular, Vue.js, Ember.js či React. V této sekci uvádím stručný popis uvedených frameworků.

7.3.1 Angular

Angular je JavaScriptový framework založený na komponentách. Umožňuje vytvářet rychlejší a efektivnější multiplatformní aplikace. Při práci s tímto frameworkem se setkáte s pojmy, jako jsou například TypeScript, Transpiler, Web Component nebo Shadow DOM. Naučit se pracovat s tímto frameworkem muže být náročné i pro zkušeného vývojáře[26].

Jedná se o framework vhodný k tvorbě rozsáhlých Single Page Applications (SPAs). Pro účely tvorby menších webových aplikací se nejeví jako vhodné řešení.

7.3.2 Vue.js

Stejně jako Angular je i Vue.js založený na komponentách. Je oproti němu však snazší na naučení a umožňuje vytváření aplikací použitím Vue CLI (Command Line Interface) nebo přímým vložením skriptu do HTML souboru. Vue.js díky své dobré škálovatelnosti představuje vhodné řešení při tvorbě rozsáhlejších webových aplikací[27].

Přestože se jedná o populární open source řešení, Vue.js bylo vyvíjeno poměrně malou skupinou lidí. To má za následek omezenou podporu tohoto frameworku.

7.3.3 React

React je JavaScriptová knihovna, která je spravována společností Facebook. Umožňuje vytvářet interaktivní uživatelská rozhraní a opakovaně použitelné komponenty. React je možné použít pro vývoj klientských, serverových a také VR aplikací. V porovnání s frameworky Angular a Vue.js má React výrazně kratší učící křivku[28].

Nevýhodou používání Reactu je špatná dokumentace způsobená velmi rychlým vývojem a častými aktualizacemi této technologie. Pro některé vývojáře může být nepohodlné muset se neustále přeučovat práci s knihovními funkcemi.

7.3.4 Ember.js

Ember.js se od roku 2015 řadí k populárním open source JavaScriptovým frameworkům. Umožňuje tvorbu dynamických klientských aplikací, které obvykle obsahují prvky, jako jsou například dashboardy, fóra nebo chaty[29].

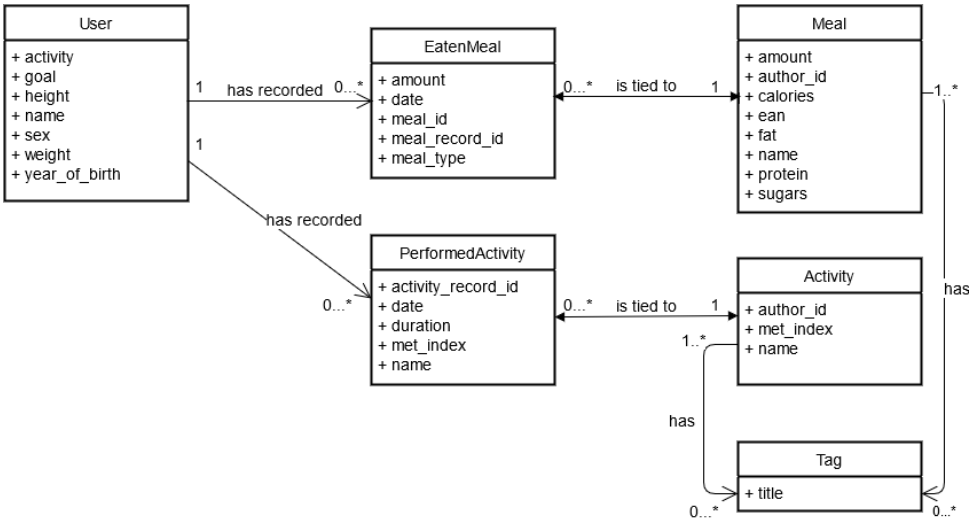
Aplikace LinkedId, Netflix, Nordstrom a mnohé další používají Ember.js. Je to dobré řešení pro vývoj komplikovaných UI. Nejedná se však o ideální řešení při vývoji statických webových aplikací.

7.3.5 Shrnutí a výběr technologie

Uvedené frameworky mají při vývoji webových aplikací oproti Vanilla JS patřičné výhody. Obecně se dá říci, že s použitím vybraných knihovních funkcí, které frameworky nabízejí, lze docílit kratšího kódu než s použitím čistého JavaScriptu. Pro zkušenějšího vývojáře tedy představuje práce s frameworkem výraznou časovou úsporu.

Na druhou stranu, pokud vývojář nemá zkušenosti s používáním frameworků či knihoven, může být nejlepší volbou právě čistý JavaScript. Pro nezkušeného programátora mohou být některé knihovní funkce příliš komplexní a náročné na porozumění. Jejich používání je tedy v daném případě kontraproduktivní. Přínosem použití Vanilla JS při vývoji aplikace je, že se vývojář musí naučit základní koncepty jazyka a vybuduje si tak dobrý základ pro používání jiných knihoven do budoucna. Vzhledem k tomu, že nemám osobní zkušenosti s žádným z frameworků, jsem se rozhodl v rámci vývoje aplikace použít čistý JavaScript.

7.4 Diagram tříd



Obrázek 7.1: Diagram tříd

Diagram tříd slouží k definování entit, které se v aplikaci vyskytují. Je navržen na základě zadání projektu a všech dat potřebných pro implementaci progresivní webové aplikace (viz kapitola 5). V této sekci popisuji jednotlivé datové kolekce tak, jak jsou reprezentovány v databázi Firestore. Kolekce jsou

navrženy na základě diagramu tříd. Některé entity jsou implementovány jako objekty nebo podkolekce jiných kolekcí.

7.4.1 Uživatel

Údaje o uživateli jsou reprezentovány kolekcí users. Data obsahují klíčové informace o uživateli potřebné pro výpočet BMR (viz podsekce 2.2.1). Záznamy v kolekci obsahují tato data:

- id (String) - jednoznačný identifikátor generovaný databází
- name (String) - uživatelem zvolená přezdívka
- activity (number) - index průměrné fyzické aktivity uživatele (od 1,2 do 2,2)
- goal (number) - koeficient představující cíl uživatele (od 0,8 do 1,2)
- height (number) - výška v centimetrech
- sex (String) - pohlaví
- weight (number) - hmotnost v kilogramech
- year-of-birth (number) - rok narození
- eaten-meals (Object []) - objekt obsahující data:
  - amount (number) - množství zapsané potraviny v gramech
  - date (String) - datum zápisu
  - meal-id (String) - identifikátor dokumentu z kolekce meals
  - meal-record-id (String) - identifikátor záznamu
  - meal-type (String) - druh pokru (snídaně, svačina, oběd, odpolední svačina nebo večeře)
- performed-activities (Object []) - objekt obsahující data:
  - activity-record-id (String) - identifikátor záznamu
  - date (String) - datum zápisu
  - duration (number) - trvání aktivity v minutách
  - met-index (number) - index fyzické náročnosti aktivity (viz podsekce 2.2.2)
  - name (String) - název aktivity

7.4.2 Potravina

Údaje o potravinách jsou reprezentovány kolekcí meals. Dokumenty v této kolekci obsahují:

- id (String) - jednoznačný identifikátor generovaný databází
- name (String) - název pokrmu
- amount (String) - typ a udávané množství v řetězci (může se jednat o varianty: 1 gram, 100 gramů, 1 mililitr a nebo 100 mililitrů)
- author-id (String) - identifikátor autora pokrmu, který může sloužit pro udělení oprávnění k editaci
- calories (number) - počet kalorií v uvedeném množství (amount)
- ean (number) - reprezentace čárového kódu EAN, který může sloužit jako identifikátor k vyhledávání potravin v databázi
- fat (number) - počet tuků v uvedeném množství
- protein (number) - počet bílkovin v uvedeném množství
- sugars (number) - počet sacharidů v uvedeném množství
- tags (String []) - pole klíčových slov přidělených k potravině, která mohou sloužit jako parametry při vyhledávání v databázi

7.4.3 Aktivita

Fyzické aktivity zvyšující energetický výdej jsou reprezentovány kolekcí activities. Každá aktivita obsahuje atributy:

- id (String) - jednoznačný identifikátor generovaný databází
- author-id (String) - id uživatele, který aktivitu přidal do databáze
- met-index (number) - index MET slouží k určení fyzické náročnosti dané aktivity (viz podsekce 2.2.2)
- name (String) - název aktivity
- tags (String []) - pole klíčových slov přidělených k aktivitě, která mohou sloužit jako parametry při vyhledávání

7.5 Návrhy obrazovek

Vytvoření návrhů obrazovek je důležitým krokem pro prvotní nastínění vizuální podoby webové aplikace. V rámci návrhů jsem se zaměřil na tvorbu low-fidelity prototypu mobilní verze webové aplikace za pomoci jednoduchého internetového nástroje Moqups[30]. V prototypu demonstruji rozložení komponent na obrazovce a neřeším detaily týkající se designu. Ukázky návrhů obrazovek se nacházejí v příloze B.

Kapitola 8
Implementace

Při vývoji webové aplikace jsem začal implementací jednoduché SPA (Single Page Application). Databázové kolekce jsem vytvořil podle diagramu 7.1 přímo v konzoli Firebase. Předlohou pro vývoj aplikace mi rovněž byly ukázky návrhů obrazovek (příloha B). Až v samotném závěru vývojové fáze jsem implementoval požadovaná API z rodiny PWA (viz kapitola 6).

V této kapitole se věnuji popisu dílčích částí vývoje webové aplikace. Zmiňuji technologie použité při vývoji a představuji vybrané části kódu aplikace. V těchto ukázkách kódu se snažím nezacházet do přílišných detailů, ale pouze demonstruji klíčové funkcionality webové aplikace.

8.1 Vývojové prostředí

Pro pohodlný vývoj webové aplikace jsem zvolil populární JavaScriptové IDE WebStorm[31] od společnosti JetBrains. Jedná se o pokročilé vývojové prostředí, které umožňuje využít našeptávače, kontrolu kódu a také vlastní terminál. Ačkoli se jedná o placený produkt, Fakulta elektrotechnická ČVUT nabízí svým studentům bezplatnou licenci.

Pro práci s databází Firestore a nastavování přístupových práv do aplikace jsem použil konzoli Firebase, která je přístupná z webového prohlížeče. Přes tuto konzoli se rovněž spravuje hosting a autentizace.

8.2 Založení projektu

Ještě před samotným založením projektu jsem do prázdného kořenového adresáře nainstaloval správce balíčků npm (Node Package Manager). Npm je správce balíčků pro JavaScript a umožňuje snadnou instalaci a odinstalaci programů. Npm lze nainstalovat zavoláním příkazu `npm install`.

Při zakládání projektu jsem postupoval podle návodu od společnosti Google [32]. Pro začátek jsem vytvořil prázdný soubor jako podklad pro webovou aplikaci a řídil se následujícími kroky:

- Založit nový projekt v konzoli Firebase

- Registrovat aplikaci u Firebase (zahrnuje vyplnění názvu aplikace a také možnost nastavení Firebase Hosting)
- Přidat Firebase SDK a inicializovat Firebase
- Nasazení aplikace na Firebase Hosting (zavolání příkazu `firebase deploy` v kořenovém adresáři projektu)

Přidání Firebase SDK vyžaduje vytvoření konfigurace, která specifikuje doménu aplikace. Jedná se o nejdůležitější část implementace jakéhokoliv Firebase projektu. Na ukázce 8.1 uvádím typickou strukturu této konfigurace.

```
1  const firebaseConfig = {
2    apiKey: "xxx",
3    authDomain: "xxx-app.app.com",
4    databaseURL: "https://xxx-app.firebaseio.com",
5    projectId: "xxx-app-123",
6    storageBucket: "myapp-project-123.appspot.com",
7    messagingSenderId: "12345678",
8    appId: "1:12345678:web:abcdefg12345678",
9    measurementId: "A-12345678"
10 };
```

Obrázek 8.1: Příklad konfigurace Firebase

8.3 Adresářová struktura projektu

Na obrázku 8.2 je znázorněna adresářová struktura projektu vyjma konfiguračních souborů. Z obrázku lze vidět, že veškeré skripty (kromě Service Workeru) jsou umístěny v jediném adresáři. Tyto skripty jsou rozděleny do ES6 modulů a jsou zkompilovány do jediného souboru – bundle.js. K tomuto úkolu jsem použil balíček rollup.js.

```
cypress ..... automatizované testy v Cypressu
public ..... zdrojové soubory aplikace
    images ..... obrázkové soubory
    scripts ..... javascriptové soubory
        controllers ..... skripty pro práci s daty
        models ..... deklarace tříd
        qr-code-scanner ..... skripty pro práci s QR kódy
        utils ..... podpůrné skripty a deklarace proměnných
        views ..... skripty pro generování obsahu
    styles ..... CSS soubory
    404.html ..... chybová stránka
    bundle.js ..... výchozí skript obsahující veškeré moduly
    index.html ..... výchozí HTML soubor
    manifest.json ..... web app manifest
    sw.js ..... Service Worker skript
```

Obrázek 8.2: Adresářová struktura projektu

8.4 Autentizace

Firebase nabízí širokou škálu možností autentizace uživatele včetně přihlášení přes telefonní číslo nebo sociální sítě. V rámci tvorby webové aplikace jsem implementoval pouze standardní možnost autentizace s využitím emailu a hesla. Pro toto řešení vystavuje Firebase vhodné metody:

- createUserWithEmailAndPassword - vytvoření nového uživatele
- signInWithEmailAndPassword - přihlášení uživatele
- signOut - odhlášení uživatele

8.5 Cloud Firestore

Na základě diagramu tříd jsem v Cloud Firestore vytvořil databázi, která obsahuje kolekce příslušné k definovaným entitám. Dále jsem nastavil Cloud Firestore Security Rules pro řízení oprávnění přístupu k datům tak, aby cizí uživatelé nemohli číst ani upravovat data jiných uživatelů. Na obrázku 8.3 je znázorněn způsob vkládání dat do Firestore.

```
1 db.collection("activities").add({
2             name: this._name,
3             met_index: parseInt(this._met_index),
4             tags: this._tags,
5             author_id: this._author_id
6         })
```

Obrázek 8.3: Vkládání dat do Firestore

8.6 Skenování a tvorba QR kódů

V kapitole 4 jsem nastínil, že do aplikace bude umožněno zadávat data pomocí QR kódů. K tomuto úkolu jsem zvolil již existující řešení, která jsou veřejně vystavena na GitHubu[33]. Prvním z vybraných řešení je samostatně instalovatelný balíček QR Scanner.

QR Scanner byl vytvořen neziskovou organizací Nimiq a je licencovaný MIT licencí, která umožňuje software volně použít pro soukromé i komerční účely. Obsah balíčku je volně ke stažení z GitHub repozitáře[34]. QR Scanner využívá asynchronního API pro možnost pohodlného skenování kódu za běhu aplikace. K zjištění dostupnosti kamery na používaném zařízení je určena asynchronní funkce `QrScanner.hasCamera`. Pro spuštění skeneru a vykreslení obrazu do příslušného `<video>` elementu slouží funkce `QrScanner.start`. Úspěšné načtení kódu nebo zavolání příkazu `QrScanner.stop` ukončí skenování.

Na ukázce 8.4 lze vidět část kódu, která řeší skenování. Naskenovaný kód je posléze parsovaný na podřetězec (substring). Odstranění prvních 46 znaků

z řetězce má za úkol oddělit část obsahující URL adresu webové aplikace a ponechat pouze parametr s identifikátorem pokrmu.

```
1  QrScanner.WORKER_PATH = '../qr-scanner-worker.min.js';
2  qrScanner = new QrScanner(element, (result) => {
3      console.log('Scanner QR code: ' + result);
4      let parsedResult = result.substring(46);
5      if (parsedResult.length > 0) {
6        qrScanner.stop();
7        // zpracovani parametru
8      } else {
9        stopScanning();
10       alert("Jidlo s uvedenym QR kodem nelze nalezt");
11     }
12 });
13 qrScanner.start();
```

Obrázek 8.4: Skenování QR kódu

Druhým z použitých řešení je knihovna QRCode.js. Tato knihovna má rovněž licenci MIT a jejím autorem je Shim Sangmin. Zdrojové kódy i s ukázkovými příklady jsou volně dostupné na autorových GitHub Pages[35]. QRCode.js je multiplatformní generátor QR kódů pro JavaScript. Vygenerované kódy jsou v aplikaci umístěny ke konkrétním pokrmům. Samotný kód v sobě nese URL webové aplikace a jako parametr uvádí ID daného jídla. Při naskenování tohoto QR kódu je uživatel přesměrován na stránku s hledaným jídlem.

8.7 Service Worker

Service Worker je samostatný skript, jehož životní cyklus je nezávislý na webové stránce. Pro instalaci skriptu je nutné jej nejdříve registrovat pomocí příkazu `navigator.serviceWorker.register` uvnitř skriptu webové aplikace. Po registraci Service Workeru spustí prohlížeč na pozadí instalační proces. Během instalačního procesu se obvykle ukládají statické soubory do cache, aby je bylo možné později použít. V případě této webové aplikace se jedná převážně o zdrojové soubory a několik obrázků. Po instalaci následuje proces aktivace, který umožňuje spravovat a případně odstranit staré záznamy v cache. Posledním krokem je specifikovat skriptu, co má dělat s daty uloženými v cache. K tomuto úkolu slouží událost `fetch`.

8.8 Závěr kapitoly

Tato kapitola měla za úkol přiblížit čtenáři proces vývoje progresivní webové aplikace. Screenshoty z vytvořené aplikace je možné najít v příloze C. Aplikace je veřejně vystavena na adrese `https://calory-calculator-13.web.app/`. Její zdrojové kódy jsou k nalezení ve veřejném GitLab repozitáři s adresou `https://gitlab.fel.cvut.cz/forgaada/bp_2021-calorie-calculator`.

Kapitola 9
Testování

Nezbytnou součástí procesu vývoje softwaru je jeho testování. Tato kapitola je zaměřená na popis dvou přístupů, které jsem zvolil k testování vytvořené webové aplikace. Prvním přístupem je kvalitativní uživatelské testování, které slouží k ověření použitelnosti a kontrole základních funkcionalit aplikace. Druhým přístupem je automatizované testování s použitím knihovny Cypress.

9.1 Uživatelské testování

V této sekci popisuji průběh a výstupy kvalitativního uživatelského testování webové aplikace. Za účelem testování aplikace jsem oslovil malou skupinu jedinců, z nichž někteří již měli s testováním softwaru zkušenosti. Uživatelům byla aplikace stručně představena a poté jim byly předloženy vyhotovené testovací scénáře. Průchody testovacími scénáři pokrývají základní funkcionalitu aplikace. Každý scénář se skládá z popisu testovacích dat, testovacích kroků a očekávaného výstupu.

9.1.1 Scénář 1 - Registrace nového uživatele

Testovací data: Neregistrovaný, nepřihlášený uživatel.
Testovací kroky:

1. Uživatel přejde do sekce Registrace uživatele.
2. Uživatel vyplní registrační formulář validními daty a odešle jej.

Očekávaný výstup: Po úspěšném odeslání formuláře je uživatel automaticky přihlášen do aplikace a přesměrován do sekce denního přehledu.

9.1.2 Scénář 2 - Přihlášení do aplikace

Testovací data: Registrovaný, nepřihlášený uživatel.
Testovací kroky:

1. Uživatel přejde do sekce Přihlášení uživatele.
2. Uživatel vyplní přihlašovací formulář a odešle jej.

Očekávaný výstup: Uživatel bude přihlášen do aplikace a přesměrován do sekce denního přehledu.

9.1.3 Scénář 3 - Tvorba nového pokrmu

Testovací data: Přihlášený uživatel.
Testovací kroky:

1. Uživatel přejde do sekce Tvorba nového pokrmu.
2. Uživatel vyplní údaje o pokrmu.
3. Uživatel k pokrmu přidělí alespoň jeden tag a uloží jej.

Očekávaný výstup: Uživatel bude přesměrován na stránku s vytvořenou potravinou. V přehledu potraviny se rovněž zobrazí přidělené tagy a vygenerovaný QR kód.

9.1.4 Scénář 4 - Vyhledávání a zápis potravin

Testovací data: Přihlášený uživatel, uživatelem vytvořený pokrm.
Testovací kroky:

1. Uživatel přejde do sekce Hledání a zápis potravin.
2. Uživatel zvolí z nabídky možnost hledání podle názvu.
3. Uživatel zadá název pokrmu a vyhledá jej.
4. Po zobrazení výsledků hledání uživatel klikne na tlačítko „Zapsat“ u detailu hledané potraviny.
5. Po načtení přehledu potraviny uživatel zadá konzumované množství, zvolí druh jídla a klikne na tlačítko „Zapsat“.

Očekávaný výstup: Uživatel bude přesměrován na denní přehled. Zapsaná potravina je zobrazena a započtena do denního příjmu.

9.1.5 Scénář 5 - Tvorba nové aktivity

Testovací data: Přihlášený uživatel.
Testovací kroky:

1. Uživatel přejde do sekce Aktivity.
2. Uživatel vyplní údaje o nové aktivitě.
3. Uživatel k aktivitě přidělí alespoň jeden tag a uloží ji.

Očekávaný výstup: Vytvořená aktivita se uloží do databáze.

9.1.6 Scénář 6 - Vyhledávání a zápis aktivit

Testovací data: Přihlášený uživatel, uživatelem vytvořená aktivita.
Testovací kroky:

1. Uživatel přejde do sekce Aktivity.
2. Uživatel z výběru zvolí vyhledávání aktivit podle názvu.
3. Uživatel zadá název vytvořené aktivity a vyhledá ji.
4. Po zobrazení výsledků hledání uživatel klikne na tlačítko „Zapsat“ u detailu hledané aktivity, zadá dobu trvání a klikne na tlačítko „OK“.

Očekávaný výstup: Uživateli se zobrazí denní přehled. Zapsaná aktivita je zobrazena a započtena v denním příjmu.

9.1.7 Scénář 7 - Vyhledávání za pomoci QR kódu

Testovací data: Přihlášený uživatel, QR kód vytvořené potraviny.
Testovací kroky:

1. Uživatel přejde do sekce Hledání a zápis potravin.
2. Uživatel klikne na tlačítko „Start“ a potvrdí povolení k přístupu ke kameře.
3. Uživatel namíří kameru na kód a vyčká na přesměrování.

Očekávaný výstup: Uživatel bude přesměrován na detail hledané potraviny.

9.1.8 Výstupy uživatelského testování

Testování se zúčastnilo celkem deset osob. V tabulce 9.1 jsou uvedeny výstupy průchodů testovacími scénáři. Každý průchod náleží k danému testovacímu scénáři (zkráceně TS) a uživateli, který je označen identifikátorem (UID). Úspěšné průchody jsou vyznačeny symbolem ✓ a neúspěšné symbolem ✗.

UID	TS 1	TS 2	TS 3	TS 4	TS 5	TS 6	TS 7
1	✓	✓	✓	✓	✓	✓	✓
2	✗	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓
4	✗	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✗
9	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓

Tabulka 9.1: Výstupy uživatelského testování

Uživatelé měli možnost podat zpětnou vazbu jak k jednotlivým scénářům, tak i k aplikaci jako celku. V naprosté většině případů proběhlo testování úspěšně. Ve dvou případech došlo k chybě při pokusu o registraci. Důvodem bylo zadání špatných hodnot do registračního formuláře. Problém jsem vyřešil úpravou validačních funkcí. V jednom případě nastala chyba při skenování QR kódu. Chyba byla pravděpodobně způsobena nízkou kvalitou kamery použité při skenování. Dále byly uživateli vzneseny následující požadavky:

- Přejmenování vybraných sekcí pro přímočařejší zápis a tvorbu potravin.
- Přidání zaškrtávacích polí pro možnost zobrazování hesel.
- Přidání odkazu na registrační formulář do sekce Přihlášení uživatele.

Uvedené požadavky jsem po ukončení testování do webové aplikace zapracoval.

9.2 Automatizované testování

Druhým zvoleným způsobem testování aplikace je automatizované end-to-end testování pomocí populární knihovny Cypress. Hlavní výhodou této knihovny je její velmi snadné použití a strmá křivka učení. V následující sekci Cypress stručně představím a uvedu, jak jsem ho v práci použil.

9.2.1 Cypress

Cypress.js je moderní knihovna pro frontendové testování webových aplikací. Na rozdíl od většiny nástrojů pro tvorbu automatizovaných testů umožňuje Cypress psaní testů i přípravu dat (tzv. mocking) bez použití Selenia. Jeho hlavní výhodou je, že kombinuje několik nástrojů do jednoho a výrazně tak snižuje nároky na znalosti vývojáře. V porovnání se Seleniem je psaní testů za pomoci Cypress.js rychlejší, jednodušší a spolehlivější[36].

Testy jsou kompilované do JavaScriptu a lze s nimi testovat jakoukoli aplikaci běžící v prohlížeči. Cypress na rozdíl od Selenia nevyžaduje spuštění mimo prohlížeč a používání vzdálených příkazů. Je spuštěn přímo nad instancí běžící aplikace a má tak nativní přístup k elementům stránky.

9.2.2 Tvorba testů

Automatizované testy jsou psané v TypeScriptu. TypeScript lze chápat jako nadstavbu jazyka JavaScript. Jeho syntaxe je velmi podobná. Umožňuje navíc využití statického typování, tříd a rozhraní. Testy pokrývají základní funkcionality aplikace a jsou rozděleny do čtyř souborů (viz obrázek 9.1).

```
fixtures ..... data pro testování
integration ..... zdrojové kódy testů
  activities.spec.ts ..... test zápisu aktivit
  login.spec.ts ..... test přihlášení do aplikace
  meals.spec.ts ..... test tvorby a zápisu jídel
  settings.spec.ts ..... test změny uživatelských údajů
plugins ..... dodatečné pluginy pro testování
screenshots ..... screenshoty automatizovaných testů
support ..... pomocné funkce pro testování
videos ..... záznamy testování
```

Obrázek 9.1: Adresářová struktura Cypress testů

Nejprve bylo zapotřebí knihovnu nainstalovat s použitím npm. K prvotnímu spuštění a vytvoření adresáře jsem zavolal příkaz `npx cypress open`. Testy přistupují k elementům na stránce za pomoci funkcí `cy.get` a `cy.contains`. Chování prohlížeče je stejné, jako kdyby hodnoty zadával sám uživatel. Vstupními hodnotami funkcí jsou id atributy elementů nebo obsažený text.

Na ukázce kódu 9.2 je demonstrována struktura běžného Cypress testu. Konkrétně se jedná o test zápisu aktivity do denního výdeje. Testová sada je specifikována příkazem `context` a jednotlivé testy v ní příkazem `it`. Cypress nejdříve přihlásí uživatele do aplikace, vyplní údaje o aktivitě a poté zkontroluje, že došlo k přesměrování a vytvoří screenshot.

```
1  context('Test recording activities', () => {
2
3    beforeEach(() => { cy.login(); });
4
5    it('Should successfully add activity record to daily
         outcome', () => {
6      const correctFinalUrl = 'https://calory-calculator
         -13.web.app/#state';
7      cy.contains('Zapsat aktivitu').click();
8      cy.get('[id="search-activity-input"]').type('
         Powerlifting', { force: true });
9      cy.get('[id="search-activity-button"]').click({
         force: true });
10     cy.get('[id="Powerlifting-add-button"]').click({
         force: true });
11     cy.get('[id="Powerlifting-input"]').type('60', {
         force: true });
12     cy.get('[id="Powerlifting-confirm-button"]').click
         ({ force: true });
13     // assert that page was loaded properly
14     cy.url().should('eq', correctFinalUrl);
15     cy.screenshot({ capture: 'runner' });
16     cy.logout();
17   });
18 });
```

Obrázek 9.2: Test zápisu aktivity

9.2.3 GitLab CI

Posledním krokem bylo nastavení automatického testování pomocí GitLab CI. Toto řešení umožňuje spouštět testy při každém odeslání do repozitáře. K využití této funkcionality bylo zapotřebí přidat do kořenového adresáře projektu soubor `.gitlab-ci.yml`. Tento skript rozděluje proces spuštění testů do dvou fází. První fáze se nazývá `build` a má za úkol stáhnout a nainstalovat závislosti potřebné k testování. Druhou fází je `test`. Během této fáze jsou spuštěny všechny testy. Testování probíhá v prohlížeči Electron.

9.3 Shrnutí

Uživatelské testování posloužilo svému účelu a ověřilo použitelnost aplikace obyčejnými uživateli. Každému uživateli jsem nastínil princip používání vytvořené aplikace a zaslal tabulku se scénáři. Během uživatelského testování nebyly detekovány žádné kritické chyby týkající se základních funkcionalit aplikace.

Automatizované testování bylo v porovnání s uživatelským testováním daleko větší výzvou. Pro tvorbu testů bylo nutné, aby každý dynamicky generovaný element měl přidělený unikátní atribut. Byl tedy nutný menší zásah do kódu aplikace. S technologií Cypress jsem se učil za pochodu, ale přesto pro mě byla práce s touto knihovnou hodnotnou zkušeností.

Kapitola 10

Závěr

Cílem této bakalářské práce bylo analyzovat existující možnosti správy kalorického příjmu a výdeje a navrhnout, implementovat a otestovat progresivní webovou aplikaci, která bude evidenci příjmu a výdeje kalorií řešit. Na základě provedené analýzy jsem definoval ověřené postupy pro výpočet doporučeného denního příjmu kalorií a makronutrientů jedince. Následně jsem provedl rešerši existujících aplikací zabývajících se evidencí kalorického příjmu. Výstupem rešerše bylo posouzení, zda tyto aplikace řeší optimálně danou problematiku, případně v čem by mohla být vyvíjená webová aplikace lepší oproti existujícím řešením.

Bylo zjištěno, že aplikace nedisponují žádnými offline režimy a také, že pro sdílení informací o potravinách používají pouze limitující EAN čárové kódy. Na základě těchto informací jsem učinil analýzu požadavků na systém a definoval možné případy užití, které posloužily jako podklady při vývoji aplikace. Poté jsem provedl srovnání a výběr vhodných technologií pro vývoj. Konkrétně se jednalo o technologie Google Firebase, Vanilla JS a vybraná webová API z rodiny PWA, s jejichž pomocí jsem aplikaci implementoval.

Výsledná aplikace splňuje veškeré body zadání, prošla uživatelským testováním, byla otestována s použitím knihovny Cypress a je veřejně vystavena na adrese `https://calory-calculator-13.web.app/`. Aplikace se od existujících řešení liší použitím moderních webových API, možností snadného sdílení informací pomocí QR kódů a dostupností offline režimu při ztrátě připojení k internetu.

Každý registrovaný uživatel může aplikaci využívat pro evidenci denního kalorického příjmu a výdeje. Vzhledem k tomu, že aplikace disponuje čtečkou a generátorem QR kódů, je další možnou oblastí využití například pohostinství. Majitelé podniků mohou do databáze snadno přidávat jejich vlastní pokrmy a vygenerované QR kódy poté umístit do své nabídky jídel. Zákazníci si eventuálně naskenují tyto kódy a zapíší si nutriční hodnoty jídel do svých denních příjmů.

Tvorba progresivní webové aplikace pro mě byla velkou zkušeností. Nikdy předtím jsem nevyvíjel aplikaci takového rozsahu. Naučil jsem se pracovat s novými technologiemi a porozuměl jsem principům vývoje progresivních webových aplikací.

10.1 Budoucnost aplikace

Přestože je aplikace v aktuálním stavu plně dostupná pro veřejnost, je zde spousta prostoru pro její vylepšení. V databázi se momentálně nachází pouze několik testovacích pokrmů. Pro lepší použitelnost aplikace mám v plánu vytvořit skript, který do databáze přidá záznamy běžných potravin.

Dalším možným vylepšením je přidání skeneru EAN čárových kódů, jehož využití by bylo možné kombinovat s implementovaným skenerem QR kódů. S touto variantou jsem počítal i při návrhu, a proto jsem k atributům potravin přidal volitelný EAN kód. Aplikaci mám v plánu aktivně používat a podílet se na jejím vylepšování.

Přílohy

Příloha A

Literatura a zdroje

1. *What Is the Meaning of Nutritional Value?* [online]. New York: Live Strong, 2019 [cit. 2021-05-10]. Dostupné z: https://www.livestrong.com/article/63090-meaning-nutritional-value/.
2. *What Are Calories?* [online]. New York: Live Science, 2015 [cit. 2021-05-10]. Dostupné z: https://www.livescience.com/52802-what-is-a-calorie.html/.
3. *Web application* [online]. USA: Computer Hope, 2020 [cit. 2021-05-10]. Dostupné z: https://www.computerhope.com/jargon/w/web-application.htm.
4. *Progressive web apps (PWAs)* [online]. USA: MDN Web Docs, 2020 [cit. 2021-05-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.
5. *What Is Basal Metabolic Rate?* [online]. San Francisco, California: Healthline Media, 2018 [cit. 2021-05-10]. Dostupné z: https://www.healthline.com/health/what-is-basal-metabolic-rate.
6. *Metabolic equivalent of task* [online]. San Francisco, California: Wikimedia Foundation, 2001 [cit. 2021-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Metabolic_equivalent_of_task.
7. *The Best Macronutrient Ratio for Weight Loss* [online]. San Francisco, California: Healthline Media, 2018 [cit. 2021-05-10]. Dostupné z: https://www.healthline.com/nutrition/best-macronutrient-ratio.
8. *BuiltWith* [online]. Sydney: BuiltWith, 2020 [cit. 2021-05-10]. Dostupné z: https://builtwith.com/.
9. *Nutritionix* [online]. USA: Nutritionix, 2020 [cit. 2021-05-10]. Dostupné z: https://www.nutritionix.com/.
10. *Cronometer* [online]. Canada: Cronometer, 2020 [cit. 2021-05-10]. Dostupné z: https://cronometer.com/.
11. *Kalorické Tabulky* [online]. Česká Republika: Kalorické Tabulky, 2020 [cit. 2021-05-10]. Dostupné z: https://www.kaloricketabulky.cz/.

12. *QR Codes vs Barcodes: The Ultimate Battle* [online]. Germany: Egoditor, 2020 [cit. 2021-05-10]. Dostupné z: https://www.qr-code-generator.com/blog/qr-codes-vs-barcodes/.

13. *What's a QR Code And How Is It Different From A Barcode?* [online]. India: Science ABC, 2021 [cit. 2021-05-10]. Dostupné z: https://www.scienceabc.com/innovation/whats-qr-code-how-its-different-from-barcode.html.

14. *What Are Functional Requirements? Types and Examples* [online]. Estonia: WinATalent, 2021 [cit. 2021-05-10]. Dostupné z: https://winatalent.com/blog/2020/05/what-are-functional-requirements-types-and-examples/.

15. *Functional and Nonfunctional Requirements of Software* [online]. Ukraine: GBKSOFT, 2021 [cit. 2021-05-10]. Dostupné z: https://gbksoft.com/blog/functional-and-nonfunctional-requirements-the-detailed-guide/.

16. LOVE, Chris. *Progressive Web Application Development by Example: Develop Fast, Reliable, and Engaging User Experiences for the Web* [online]. UK: Packt Publishing, Limited, 2018 [cit. 2021-05-10]. ISBN 978-17-8728-234-6. Dostupné z: https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=5477666.

17. LEPAGE, Pete. *The Cache API: A quick guide* [online]. England: Google, 2020 [cit. 2021-05-10]. Dostupné z: https://web.dev/cache-api-quick-guide/.

18. ARCHIBALD, Jake. *The Offline Cookbook* [online]. England: Google, 2020 [cit. 2021-05-10]. Dostupné z: https://web.dev/offline-cookbook/.

19. *Web Storage API* [online]. USA: MDN Web Docs, 2020 [cit. 2021-05-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API.

20. *IndexedDB API* [online]. USA: MDN Web Docs, 2020 [cit. 2021-05-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

21. *Push API* [online]. USA: MDN Web Docs, 2020 [cit. 2021-05-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Push_API.

22. *Firebase* [online]. USA: Firebase, 2021 [cit. 2021-05-10]. Dostupné z: https://firebase.google.com/.

23. *AWS Amplify* [online]. USA: Amazon Web Services, 2021 [cit. 2021-05-10]. Dostupné z: https://aws.amazon.com/amplify/.

24. *Apache Usergrid* [online]. USA: The Apache Software Foundation, 2020 [cit. 2021-05-10]. Dostupné z: https://usergrid.apache.org/.

25. *mBaaS Comparison: The best 2021 providers* [online]. USA: Back4App, 2021 [cit. 2021-05-10]. Dostupné z: `https://blog.back4app.com/mbaas-comparison/`.

26. ARORA, Chandermani; HENNESSY, Kevin. *Angular 6 by Example: Get up and Running with Angular by Building Modern Real-World Web Apps, 3rd Edition*. Třetí vydání. UK: Packt Publishing, Limited, 2018. ISBN 978-17-8883-517-6.

27. HALLIDAY, Paul. *Vue. js 2 Design Patterns and Best Practices: Build Enterprise-Ready, Modular Vue. js Applications with Vuex and Nuxt*. První vydání. UK: Packt Publishing, Limited, 2018. ISBN 978-17-8883-979-2.

28. ROLDÁN, Carlos Santana. *React Cookbook: Create Dynamic Web Apps with React Using Redux, Webpack, Node. js, and GraphQL*. První vydání. UK: Packt Publishing, Limited, 2018. ISBN 978-17-8398-072-7.

29. KELONYE, Mitchel. *Mastering Ember.js*. První vydání. UK: Packt Publishing, Limited, 2014. ISBN 978-17-8398-198-4.

30. *Moqups* [online]. Romania: S.C Evercoder Software S.R.L., 2021 [cit. 2021-05-10]. Dostupné z: `https://moqups.com/`.

31. *WebStorm* [online]. Česká Republika: JetBrains s.r.o., 2021 [cit. 2021-05-10]. Dostupné z: `https://www.jetbrains.com/webstorm/`.

32. *Add Firebase to your JavaScript project* [online]. USA: Firebase, 2020 [cit. 2021-05-10]. Dostupné z: `https://firebase.google.com/docs/web/setup`.

33. *GitHub* [online]. GitHub, 2021 [cit. 2021-05-10]. Dostupné z: `https://github.com`.

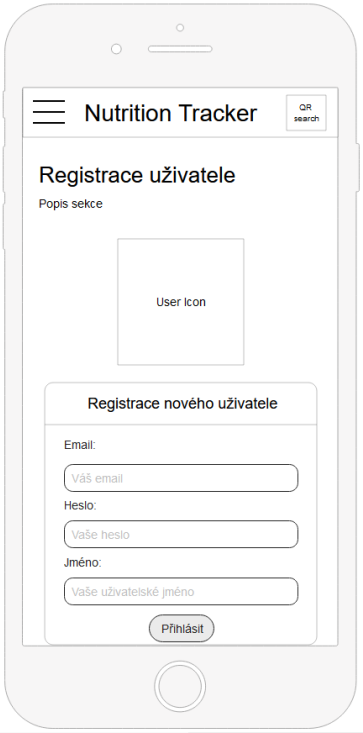
34. *QR Scanner* [online]. GitHub, 2020 [cit. 2021-05-10]. Dostupné z: `https://github.com/nimiq/qr-scanner`.

35. *QRCode.js* [online]. GitHub Pages, 2015 [cit. 2021-05-10]. Dostupné z: `https://davidshimjs.github.io/qrcodejs/`.

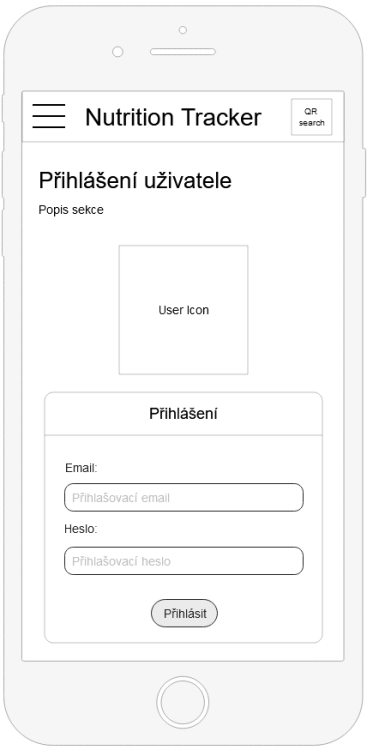
36. *Why Cypress?* [online]. Atlanta, Georgia: Cypress.io, 2021 [cit. 2021-05-10]. Dostupné z: `https://docs.cypress.io/guides/overview/why-cypress`.

Příloha B

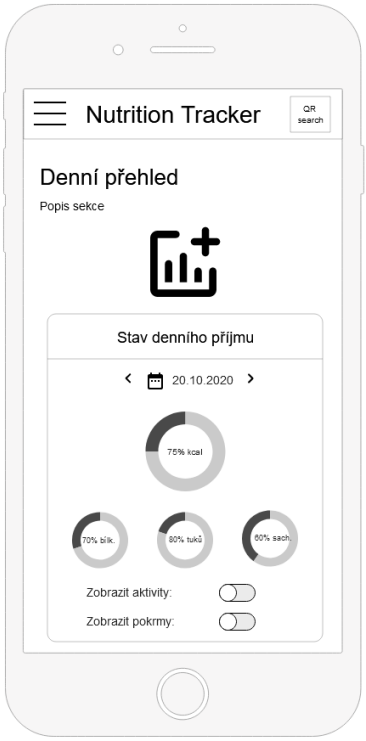
Ukázky návrhů obrazovek



Obrázek B.1: Návrhy obrazovek – Registrace



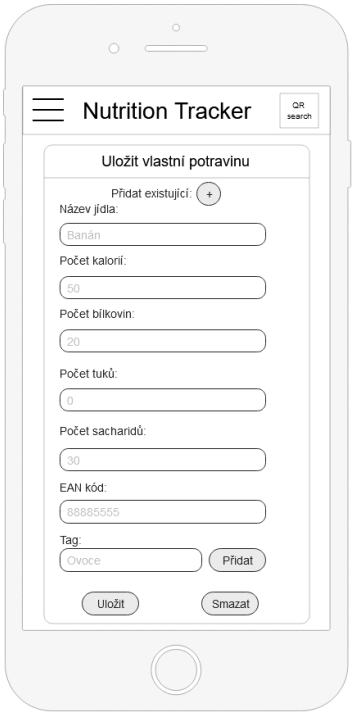
Obrázek B.2: Návrhy obrazovek – Přihlášení



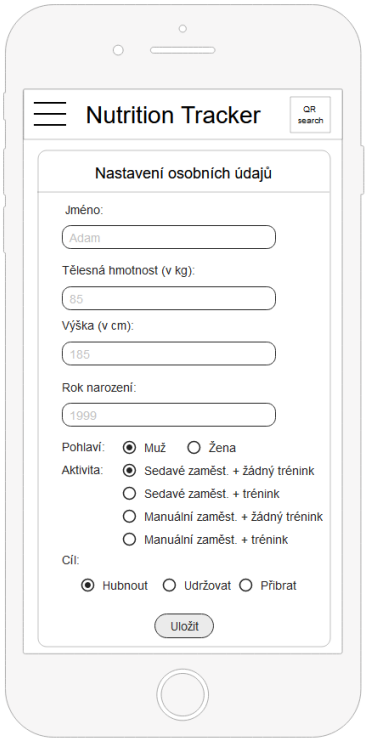
Obrázek B.3: Návrhy obrazovek – Denní přehled

Obrázek B.4: Návrhy obrazovek – Zapsat jídlo

Obrázek B.5: Návrhy obrazovek – Zapsat aktivitu



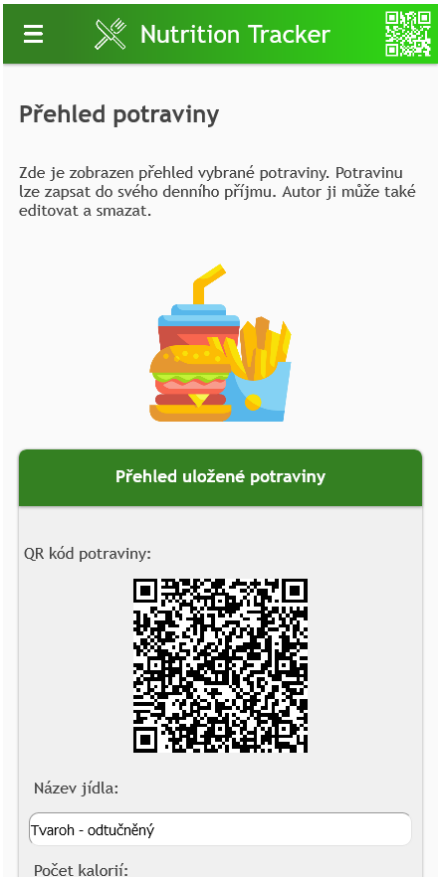
Obrázek B.6: Návrhy obrazovek – Tvorba nového pokrmu



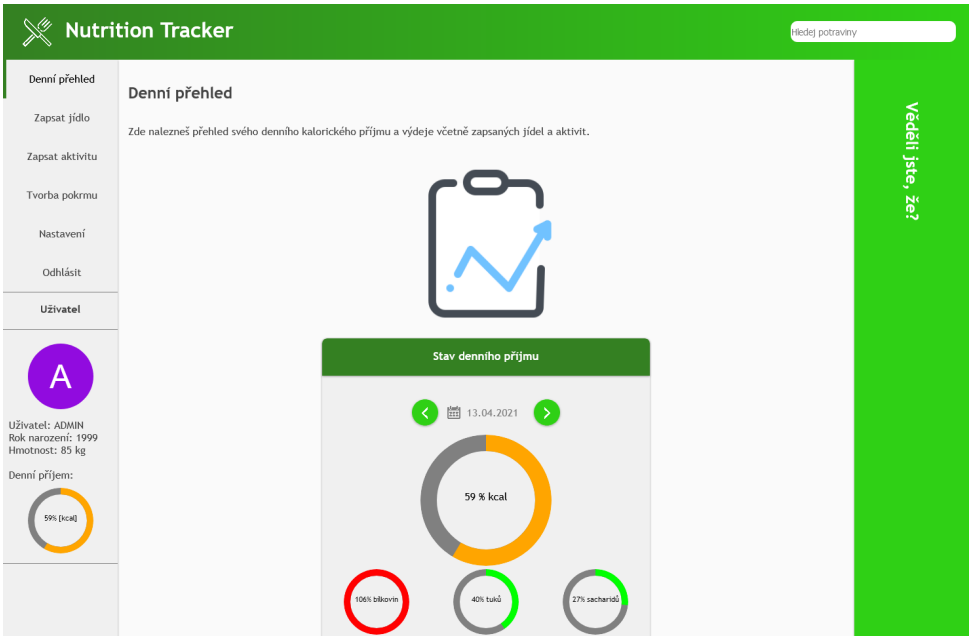
Obrázek B.7: Návrhy obrazovek – Nastavení

Příloha C

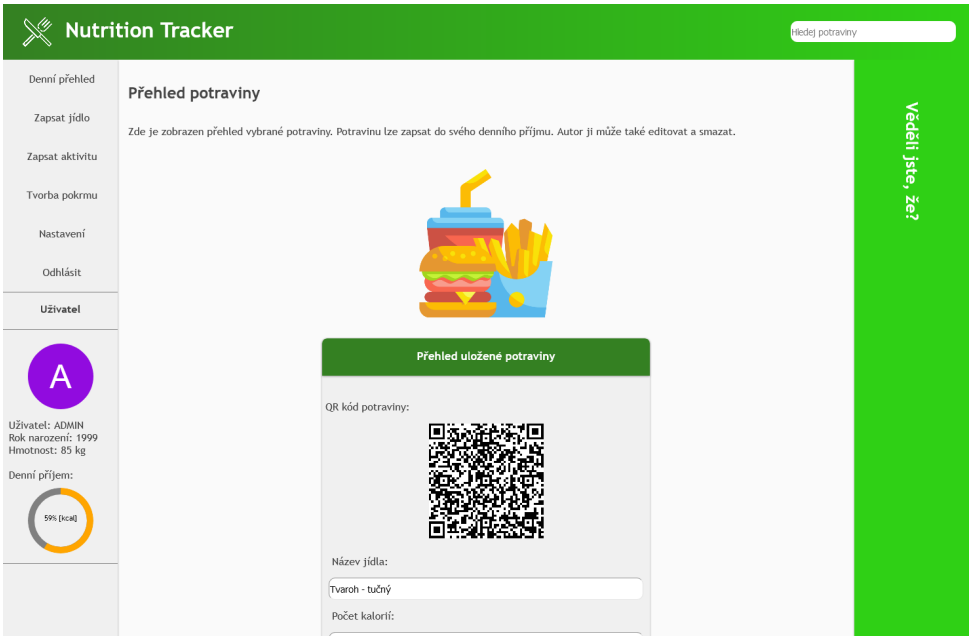
Screenshoty z webové aplikace



Obrázek C.1: Přehled potraviny v mobilní verzi aplikace



Obrázek C.2: Denní přehled ve verzi aplikace pro PC



Obrázek C.3: Přehled potraviny ve verzi aplikace pro PC

Příloha D

Seznam použitých zkratek

API	Application Programming Interface
BaaS	Backend as a Service
BMR	Basal Metabolic Rate
CI	Continuous Integration
CSS	Cascading Style Sheets
DOM	Document Object Model
EAN	European Article Number
FaaS	Function as a Service
ČVUT	České vysoké učení technické
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MET	Metabolic equivalent of task
NoSQL	Not Only SQL
npm	Node package manager
PWA	Progressive Web Application
SDK	Software Development Kit
SPA	Single-page application
UI	User Interface
URL	Uniform Resource Locator
QR	Quick Response

Příloha E

Obsah přiloženého CD

- `images` ..... obrázky uvedené v práci
  - `diagrams` ..... diagramy použité při návrhu aplikace
  - `screenshots` ..... snímky obrazovky webové aplikace
  - `wireframes` ..... návrhy obrazovek
- `testing-scenarios.xlsx` ..... dotazník s testovacími scénáři
- `NutritionTracker.zip` ..... zdrojové kódy aplikace
- `thesis-source.zip` ..... zdrojové kódy textu práce ve formátu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF