



## Zadání bakalářské práce

<b>Název:</b>	Webová aplikace pro správu a sdílení receptů
<b>Student:</b>	Vojtěch Moravec
<b>Vedoucí:</b>	Ing. Oldřich Malec
<b>Studijní program:</b>	Informatika
<b>Obor / specializace:</b>	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
<b>Katedra:</b>	Katedra softwarového inženýrství
<b>Platnost zadání:</b>	do konce letního semestru 2022/2023

### Pokyny pro vypracování

Vytvořte prototyp webové aplikace pro správu a efektivní zobrazení receptů a surovin, plánování jídelníčku a navrhnete vhodný model sdílení výše zmíněného.

Postupujte v těchto krocích:

- Analyzujte potřeby potenciálních uživatelů, zaměřte se na potřeby frontendové části aplikace.
- Analyzujte existující konkurenční řešení.
- Vytvořte návrh designu aplikace – zaměřte se na různé potřeby uživatele při využívání webu na mobilu a na počítači - při plánování, sdílení či vaření. Optimalizujte zobrazení pro každý z těchto úkonů.
- Prověřte možnosti automatického nákupu potřebných surovin u služeb třetích stran.
- Zvolte vhodné technologie, ve kterých řešení budete implementovat.
- Na základě analýzy, návrhů a designu implementujte funkční prototyp.
- Prototyp podrobte uživatelskému testování a zhodnoťte výsledek testování.



Bakalářská práce

# WEBOVÁ APLIKACE PRO SPRÁVU A SDÍLENÍ RECEPTŮ

Vojtěch Moravec

Fakulta informačních technologií  
Katedra softwarového inženýrství  
Vedoucí: Ing. Oldřich Malec  
13. dubna 2022

České vysoké učení technické v Praze  
Fakulta informačních technologií

© 2022 Vojtěch Moravec. Odkaz na tuto práci.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

Odkaz na tuto práci: Moravec Vojtěch. *Webová aplikace pro správu a sdílení receptů*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

## Obsah

Poděkování	vi
Prohlášení	vii
Abstrakt	viii
Shrnutí	ix
Seznam zkratek	x
Slovník	xi
<b>1 Úvod</b>	<b>1</b>
<b>2 Cíl</b>	<b>3</b>
<b>3 Analýza</b>	<b>5</b>
3.1 Kvalitativní průzkum	5
3.2 Funkční požadavky	5
3.2.1 F1: Správa receptů	5
3.2.2 F2: Sdílení receptů	5
3.2.3 F3: Objednávky přes služby typu rohlík.cz	5
3.2.4 F4: Plánovač	5
3.2.5 F5: Správa spíže	6
3.2.6 F6: Možnost ankety kolem jídelníčku	6
3.3 Nefunkční požadavky	6
3.3.1 U1: Dostupné jako webová aplikace	6
3.3.2 P1: Systém pro jednotky uživatelů	6
3.3.3 S1: Serverless s možností připojení na speciální API v budoucnu	6
3.4 Existující řešení	7
3.4.1 Vareni.cz	7
3.4.2 Toprecepty.cz	7
3.4.3 Recepty.cz	8
3.4.4 Závěr	9
3.5 Nákup surovin	9
3.5.1 Komunikace s Rohlíkem a Košíkem	9
<b>4 Technologie</b>	<b>11</b>
4.1 Výběr	11
4.2 Vue.js	11
4.3 SPA	11
4.4 Vuetify	11
4.5 Vuex	12
4.6 Vue i18n	12

4.7	Vue Router . . . . .	12
4.8	Firebase . . . . .	12
4.8.1	Firestore . . . . .	12
4.9	Fulltextové vyhledávání . . . . .	12
4.10	PWA . . . . .	13
<b>5</b>	<b>Návrh . . . . .</b>	<b>15</b>
5.1	Vzhled aplikace . . . . .	15
5.2	Název a logo . . . . .	16
5.3	Databáze ve Firebase . . . . .	17
5.3.1	Doménový model . . . . .	17
5.3.2	Schéma v dokumentové databázi . . . . .	18
5.4	Uložiště ve Firebase . . . . .	18
5.5	Data v aplikaci . . . . .	18
5.6	Funkce aplikace . . . . .	19
<b>6</b>	<b>Implementace . . . . .</b>	<b>21</b>
6.1	Založení projektu . . . . .	21
6.1.1	Firebase . . . . .	21
6.1.2	Automatický deploy . . . . .	21
6.2	Struktura projektu . . . . .	22
6.3	Router . . . . .	22
6.4	Překlady . . . . .	23
6.5	Vuex . . . . .	23
6.6	PWA . . . . .	24
6.7	Stylování aplikace . . . . .	24
6.8	Firebase . . . . .	24
6.8.1	Firebase Auth . . . . .	24
6.8.2	Firestore cache . . . . .	25
6.8.3	Firestore rules . . . . .	26
6.8.4	Firestore Functions . . . . .	26
<b>7</b>	<b>Testování . . . . .</b>	<b>29</b>
7.1	Uživatelské testování . . . . .	29
<b>8</b>	<b>Možnosti aplikace v budoucnosti . . . . .</b>	<b>31</b>
8.1	Interakce mezi uživateli . . . . .	31
8.2	Časovač . . . . .	31
8.3	Verze FIT . . . . .	31
8.4	Pohodlí pro uživatele . . . . .	31
8.5	Propojení se službami Košík a Rohlík . . . . .	32
8.6	Skupiny . . . . .	32
<b>9</b>	<b>Závěr . . . . .</b>	<b>33</b>
<b>A</b>	<b>Nějaká příloha . . . . .</b>	<b>35</b>
<b>B</b>	<b>Příloha 2 . . . . .</b>	<b>37</b>

## Seznam obrázků

3.1	Model požadavků . . . . .	6
3.2	Hlavní stránka vareni.cz . . . . .	7
3.3	Hlavní stránka toprecepty.cz . . . . .	8
3.4	Hlavní stránka recepty.cz . . . . .	8
5.1	Hlavní obrazovka . . . . .	16
5.2	Vývoj loga . . . . .	17
5.3	Doménový model . . . . .	17
5.4	Struktura dat ve Firestore . . . . .	18
6.1	Adresářová struktura . . . . .	22
A.1	Návrh velkého loga . . . . .	35
A.2	Návrh velkého loga . . . . .	35

## Seznam tabulek

## Seznam výpisů kódu

1	Příklad ochrany stránky proti nepřihlášeným uživatelům . . . . .	23
2	Použití Auth Guardu na stránce profilu uživatele . . . . .	23
3	Překlad pro recepty . . . . .	23
4	Použití překladu . . . . .	24
5	Správná zpráva pro aktualizaci . . . . .	24
6	Zapnutí perzistentního módu . . . . .	25
7	Získání instance Firestore . . . . .	25
8	Metoda pro stažení dat . . . . .	26
9	Pravidlo pro přístup k veřejnému receptu . . . . .	26

*Chtěl bych poděkovat především sit amet, consectetur adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.*



## 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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 13. dubna 2022

.....

## Abstrakt

V této práci řeším, jak navrhnout a vytvořit prototyp webové aplikace, která má uživateli poskytnout jednotné rozhraní pro vaření podle receptů, tedy správu receptů a surovin, nákupy nebo například sdílení mezi uživateli. ? (Jak? Metody atd.). Jako výsledek vzešel prototyp připravený na reálné použití, který splňuje všechny původní požadavky a nabízí i další funkce. Aplikace je veřejně přístupná a pomůže každému, kdo hledá řešení pro ukládání receptů a dalších možností, které na nich staví. Na závěr jsem doplnil možná rozšíření do budoucna, která by aplikaci učinila více komplexní a nabídla uživateli kompletní balíček bez potřeby použití dalších aplikací.

**Klíčová slova** frontend, Vue, Vuetify, recepty na vaření, webová aplikace, serverless, Firebase

## Abstract

Fill in abstract of this thesis in English language. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

**Keywords** frontend, Vue, Vuetify, recipes, web app, serverless, Firebase

## Summary section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem.

## Summary section

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa.

## Summary section

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae

enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

## Summary section

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

## Summary section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Lorem lorem lorem.

## Seznam zkratek

API	Application Programming Interface
SQL	Structured Query Language
NoSQL	Not Only SQL
JSON	JavaScript Object Notation
FE	Frontend
NPM	Node Package Manager
CLI	Command Line Interface
YML	Yaml Ain't Markup Language
JS	JavaScript
PWA	Progressive Web App
CSS	Cascading Style Sheets

## Slovník

Frontend	Část aplikace, kterou vidí uživatel a reaguje s ní
Mesh gradient	Doplnit překlad
Drag and drop	Doplnit překlad
Firestore	Doplnit překlad
Firebase	Doplnit překlad
Cloud Storage	Doplnit překlad
Deploy	Doplnit překlad
Build	Doplnit překlad
Navigation Guard	Doplnit překlad
Prop	Doplnit překlad
Pop-up	Doplnit překlad
Sign-in provider	Doplnit překlad
Local storage	Doplnit překlad
Serverless	Doplnit překlad



## Kapitola 1

# Úvod

Vaření se týká spousty lidí. Mladší generace k tomu používá recepty, které jsou dostupné na internetu. Recepty, které najdou a které se jim osvědčí, jsou často na jiných stránkách a správa těchto receptů se stává nereálná. Na druhou stranu na nejznámějších portálech není možné si recepty ukládat soukromně a tak spoustu starších lidí nevyužije možnosti digitalizace jejich receptů. Většina z nich si totiž uchovává své recepty napsané na listech papíru, což ale v dnešní době již není praktické řešení. Už jen kvůli sdílení receptu s někým, kdo si ho vyžádá, což je celkem běžná záležitost.

Výsledek této práce bude moci použít široká veřejnost – každý kdo vaří podle receptů. Aplikace jim pomůže s jednoduchým zadáním receptů do sbírky, kde mohou mít všechny recepty na jednom místě. Dále budou moci uživatelé využít plánovače, kam lze zanést již přidané recepty či počet porcí. Pokud nejsou zvyklí plánovat a spíše řeší věci na poslední chvíli, bude pro ně připraven rychlý návrh receptu. V neposlední řadě bude jednoduché své recepty sdílet s ostatními uživateli.

V práci popisuji proces vývoje webové aplikace od analýzy přes návrh až po implementaci. Zaměřím se hlavně na frontend v javascriptovém frameworku Vue.js, ale provedu čtenáře i backendovou částí, která je tvořena pomocí nástrojem Firebase od společnosti Google. Začnu sběrem informací od potencionálních uživatelů, pomocí kterých sestavím funkční a nefunkční požadavky. Poté zanalyzuji konkurenční řešení a popíšu jejich plusy či nevýhody. Poté představím technologii, které jsem si vybral pro vývoj. Dále navrhnu uživatelské rozhraní, grafické prvky či datovou strukturu aplikace. Na základě předchozích kapitol implementuji aplikaci a zmíním problémy, se kterými jsem se při vývoji setkal. Sestavím test pro uživatele a zhodnotím jeho výsledky. Na závěr zmíním funkce které jsem nestihl implementovat či rozšíření, která by se v budoucnosti hodila.

Téma mě zaujalo, protože si myslím, že nabízí spoustu míst, pro učení se nových věcí. Také jsem nenašel žádnou alternativu, která by nabízela všechny funkce, které moje řešení poskytuje.







## Kapitola 2

# Cíl

Cílem této bakalářské práce je vytvořit prototyp webové aplikace pro správu receptů. Dále je nutné prozkoumat možnosti zjednodušení nákupu surovin či plánování vaření jídel.

Nejdříve zanalyzuji potřeby potenciálních uživatelů, kde se pokusím zjistit, které všechny funkce by ocenili. Poté se zaměřím na existující řešení, která porovnám s mým řešením a popíšu vylepšení.

Dále vytvořím návrh designu aplikace – wireframy, které zachycují zároveň, jak aplikace vypadá. Důraz kladu na rozdíly mezi použitím na mobilu a počítači.

Na základě analýzy a návrhu vyberu technologie, které použiju pro implementaci funkčního prototypu.

Na konec podrobím aplikaci uživatelskému testování a zhodnotím jeho výsledky.



## Kapitola 3

# Analýza

### 3.1 Kvalitativní průzkum

Nejprve bylo potřeba zjistit, co by potenciální uživatelé aplikace ocenili, tedy získat od nich požadavky. Zvolili jsme kvalitativní průzkum, který se narozdíl od kvantitativního zaměřuje na malou skupinu respondentů. Pro získání odpovědí jsem si připravil sadu otevřených otázek, kterých jsem se držel při rozhovoru. Hovory byly uskutečněny online a každá trvala mezi dvaceti minutami a jednou hodinou.

Všechny rozhovory byly vedeny v rozmezí dvou týdnů a poté jsem z nich sestavil funkční a nefunkční požadavky.

### 3.2 Funkční požadavky

#### 3.2.1 F1: Správa receptů

Pro uživatele je důležité udržovat své recepty aktuální. Je tedy nutné implementovat rozhraní, které umožní pracovat s přidávanými recepty.

#### 3.2.2 F2: Sdílení receptů

Narozdíl od ostatních služeb, které jsou popsány dále v textu, by tato měla poskytovat sdílení mezi uživateli. Na to se pojí i viditelnost receptů, tedy všechny nebudou veřejné, ale bude možné je přidat jako soukromé či neveřejné.

#### 3.2.3 F3: Objednávky přes služby typu rohlík.cz

Převážně mladší generace dnes hojně využívá služeb na rozvoz nákupu. Je tedy potřeba přidat zjednodušený přesun potřebných surovin do nákupních košíků v těchto službách, a tak zefektivnit čas nákupu.

#### 3.2.4 F4: Plánovač

Pro mnoho lidí je důležité naplánovat si kdy mají čas si jídlo uvařit a naopak kdy by ho potřebovali mít již hotové. Pro bude v aplikaci plánovač, kde bude uživatel moci sledovat, kdy ho čeká co uvařit.

### 3.2.5 F5: Správa spíže

Tento požadavek souvisí s těmi předchozími, tedy hlavně ulehčí uživateli nákup surovin a plánování vaření. Pokud uživatel nebude chtít kontrolovat jaké suroviny má doma, funkci nebude muset využít.

### 3.2.6 F6: Možnost ankety kolem jídelníčku

Spíše než v běžném životě by se anketa dala využít například při výletu s kamarády či soustředění s týmem nebo kapelou. Uživatelé by měli možnost hlasovat, v jaký den a jaké jídlo by chtěli.

## 3.3 Nefunkční požadavky

### 3.3.1 U1: Dostupné jako webová aplikace

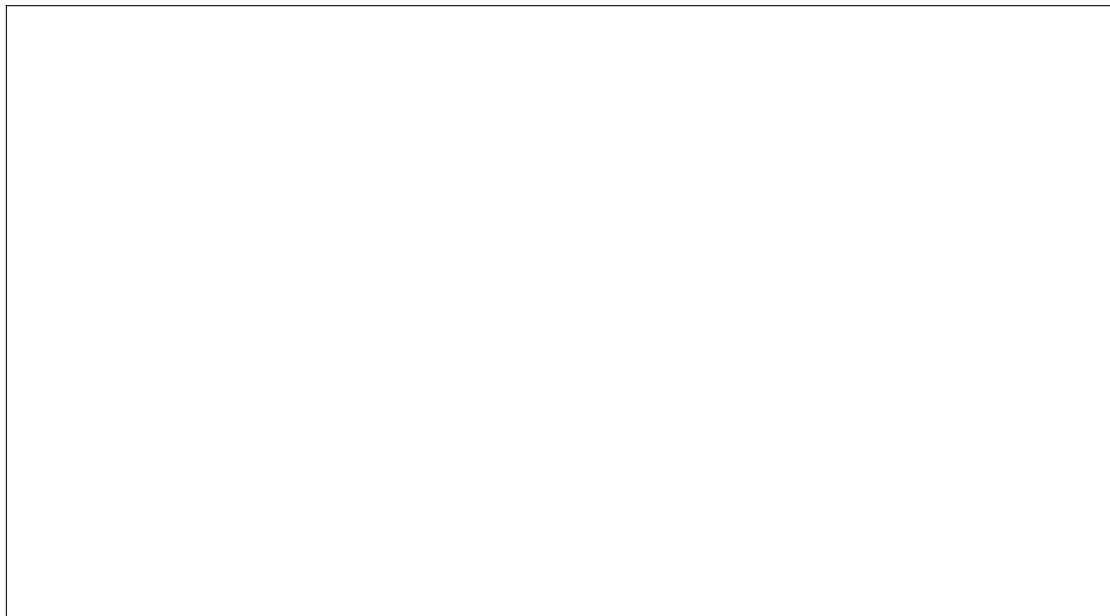
Vzhledem k potřebě mít aplikaci dostupnou jak pro mobily, tak pro počítač, je webová aplikace nejflexibilnější řešení.

### 3.3.2 P1: Systém pro jednotky uživatelů

Aplikaci nebude využívat mnoho uživatelů, ale je nutné myslet na budoucí rozšíření.

### 3.3.3 S1: Serverless s možností připojení na speciální API v budoucnu

Prozatím se pro backendovou část využije *serverless* řešení. Pokud by tato alternativa v budoucnu neposkytovala dostatečné funkce nebo se nevyplatila finančně, je možné přejít na jiný backend.



■ Obrázek 3.1 Model požadavků

## 3.4 Existující řešení

Porovnal jsem nejpoužívanější české portály s recepty. Některé z nich nabízejí další obsah jako například blog či magazín.

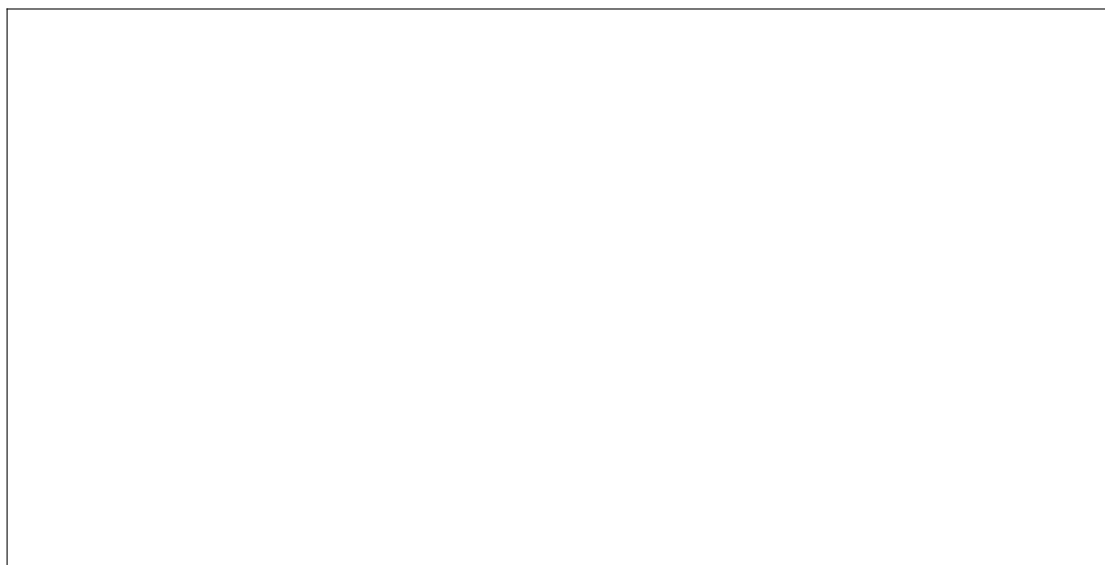
### 3.4.1 Vareni.cz

Na vareni.cz [1] se nachází několik reklam, které jsou velké a rušivé. Není zde možné si přidat soukromý recept a sdílet ho pouze s vybranými uživateli. Celkový koncept přidání receptu je pouze veřejný, není tedy možné si zde vytvořit sbírku oblíbených receptů z různých portálů. Vyhledávání je možné podle názvu, ingrediencí či různých parametrů jako je druh jídla nebo národní kuchyně. Recepty mají hodnocení od jedné do pěti hvězdiček.

Na stránce konkrétního receptu je shrnutí nejdůležitějších vlastností, tedy název, krátký popis, hodnocení a čas vaření. Také v hlavičce kolují různé komentáře. Výhodou jsou návrhy podobných receptů. Naopak velmi špatné je rozložení seznamu ingrediencí a postupu přípravy. Tato část je velmi nepřehledná a je nerozeznatelné kde recept končí a začínají jiné recepty.

Jsou zde kuchařky, ale než že by byly tématické nebo měli společného autora, přišlo mi, že se jedná se o náhodnou kolekci receptů, kde je jich často více než tisíc.

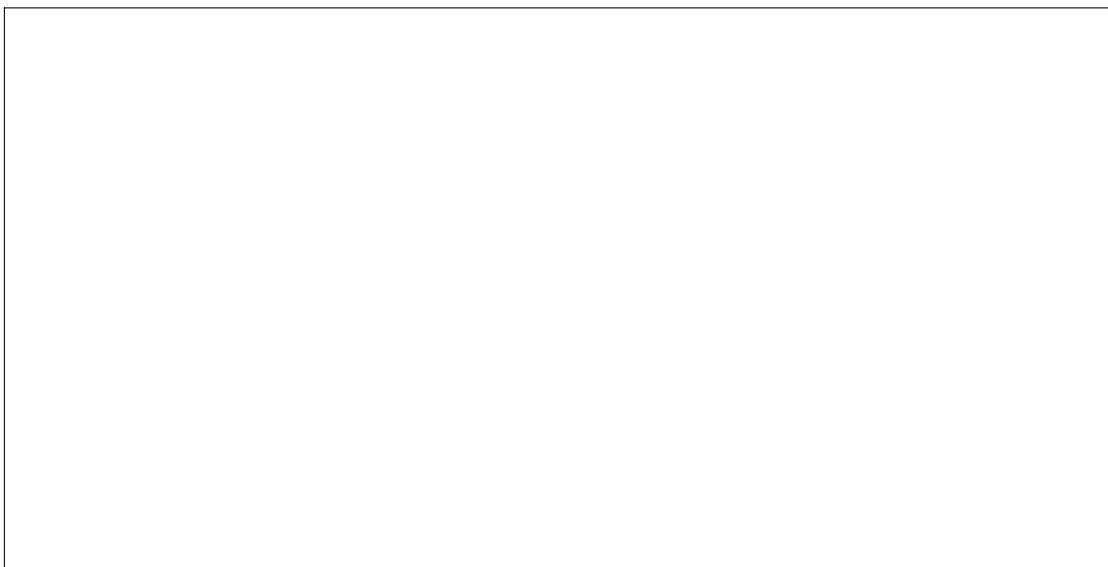
Celkem hezký nápad jsou fotorecepty. Uživatele provedou celým vařením pomocí kroků, přičemž u každého je fotodokumentace jak daný krok provést.



■ Obrázek 3.2 Hlavní stránka vareni.cz

### 3.4.2 Toprecepty.cz

Na tomto portálu [2] byla bohužel nefunkční registrace, takže jsem nemohl nahlédnout na funkce poskytované přihlášeným uživatelům. Opět zde byla přítomna velká reklama, která zabírala většinu stránky. Jinak byl web navrhnut přehledně, ale narazil jsem na několik nefunkčních prvků na mobilním zobrazení. Zhodnotit přidání receptů a jak funguje jejich sdílení zhodnotit nemohu, kvůli výše zmíněným problémům. Našel jsem funkci podobnou doporučování receptů, avšak se pravděpodobně jedná o náhodné doporučení, které nemá nic společného s tím co má uživatel rád. Dále je na webu dostupný online magazín, kde jsou různé články týkající se gastronomie.

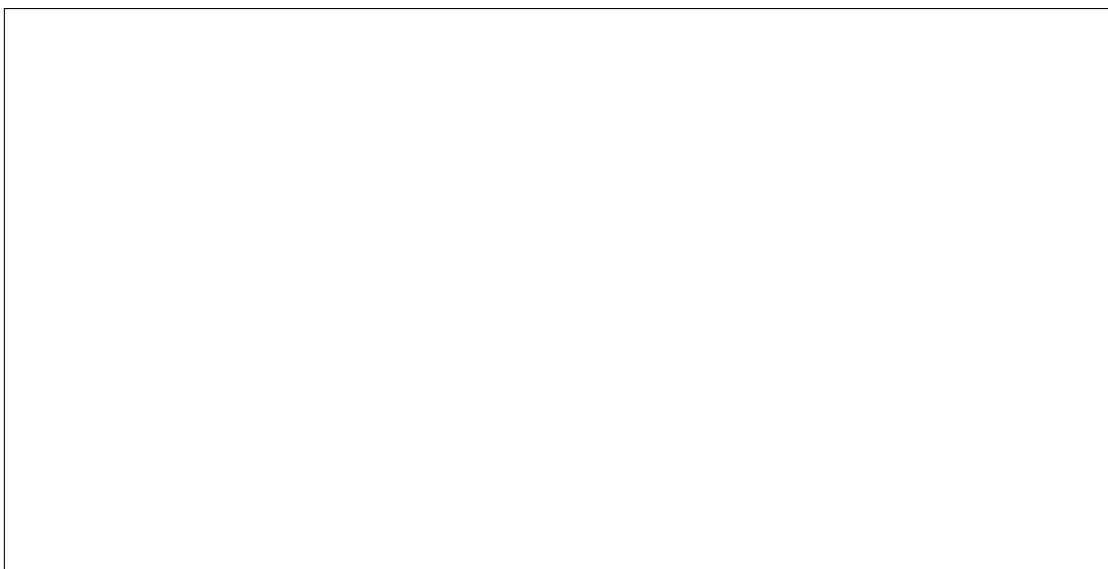


■ **Obrázek 3.3** Hlavní stránka toprecepty.cz

### 3.4.3 Recepty.cz

I třetí zástupce [3] existujících řešení používá reklamu přes celou stránku okolo jejího obsahu. Podobně jako toprecepty.cz je zde magazín obsahující příspěvky na spoustu témat o vaření. U přidání receptu nebylo napsané, co se s receptem stane, zda bude veřejný nebo se zobrazí pouze mě. Po kliknutí na tlačítko „Uložit recept“, se zobrazila stránka s nadpisem „Recept čeká na schválení“. Nebylo tedy opět možné soukromé použití.

Zobrazení receptu je podobné tomu na vareni.cz, ale přijde mi více přehledné. U kroků se zobrazují rady, které ale nemají s daným krokem nic společného, spíše odkazují na náhodné články či recepty.



■ **Obrázek 3.4** Hlavní stránka recepty.cz

### 3.4.4 Závěr

Existující řešení na vyhledávání receptů nabízejí pouze veřejné recepty a mají spoustu reklam. Moje řešení bude poskytovat možnost soukromé sbírky receptů a jejich sdílení s vybranými uživateli či pomocí odkazu.

## 3.5 Nákup surovin

Vedoucí práce již dříve používal aplikaci Zdravý stůl [4]. Tam bylo možné si jídlo objednat přes rohlik.cz (vložit do košíku). Tudíž jsme chtěli tuto funkcionalitu zachovat a přidat možnosti jako například vytvoření objednávky u konkurence - kosik.cz či zobrazení interaktivního nákupního listu.

### 3.5.1 Komunikace s Rohlíkem a Košíkem

Nejdříve jsme se rozhodli kontaktovat Rohlík. Po pár vyměněných e-mailech jsme obdrželi celou dokumentaci k jejich API, které nám otevřelo spoustu možností i do budoucna. Například bychom mohli sledovat, jaké suroviny jsou právě ve slevě a podle toho doporučovat jídla.

Od Košíku jsme dostali pozvání na schůzku, kde jsme si mohli prohlédnout i jejich kanceláře. Na schůzce jsme hned na začátku zjistili, že API pro partnery narozdíl od Rohlíku ještě dostupné není (Rohlík na API pro partnery také teprve pracuje, ale zatím jsme dostali přístup k API pro jejich aplikaci), ale už na něm pracují. Plánované období vydání je první kvartál roku 2022. Pro jeho využití je však potřeba OAuth server, který zatím není dostupný. Jako alternativu jsme dohromady vymysleli link na přidání surovin přímo do košíku, ze kterého nakonec sešlo, protože jsme poté objevili funkci „Nákupní lístek“, kterou bychom mohli využít. Odeslali bychom seznam surovin a uživatel by si je poté mohl vybrat přímo z nabídky na Košíku. Nakonec jsme zjistili, že by se Košíku hodilo rozrůst sbírku receptů a bylo by možné pro uživatele naší aplikace nabídnout jejich recepty Košíku, který by je následně odkoupil.





## Kapitola 4

# Technologie

### 4.1 Výběr

Vzhledem k tomu, že aplikace byla původně zamýšlena jako osobní projekt, který by si následně spravoval sám vedoucí a já jsem měl zkušenosti pouze s frameworkem Vue.js, hlavní technologie, okolo které se projekt postaví, byla předem daná. Poté jsem postupně vybíral další součásti, které bychom mohli využít. Velkou výhodou bylo, že právě vedoucí práce má s většinou z těchto knihoven či pluginů zkušenosti, a tak když se mi něco nedařilo, mohl jsem se na něj obrátit.

### 4.2 Vue.js

Vue je progresivní JavaScriptový framework, který narozdíl od konkurenčních řešení (React, Angular) nezaštituje žádná velká korporace, ale je vyvíjen komunitou.[5] Zvolili jsme verzi 3, protože je to lepší řešení do budoucna, než později aktualizovat celou aplikaci z verze 2. V pozdější fázi vývoje se ale ukázalo, že pro verzi 3 nebyla plně dokončena hlavní knihovna, kterou jsem chtěl využít. Musel jsem tak ponížít verze všech závislostí a přejít tak na verzi 2.

### 4.3 SPA

SPA neboli *single page application* je stránka, která využívá takové architektury, kde použitá technologie nejen kontroluje vzhled stránky, ale i data a manipulaci s nimi a navigaci tím způsobem, že není nutné provádět obnovení stránky.[6]

### 4.4 Vuetify

Vuetify je knihovna implementující různé komponenty, které je možné použít při tvorbě uživatelského rozhraní. Kromě toho usnadňuje práci s rozložením na stránce, přizpůsobením barevného téma, ikonami atd. Další výhodou jsou týdenní aktualizace momentální verze, které přidávají nové funkce a opravují nalezené chyby.[7] Bohužel tato knihovna nebyla v době vývoje plně dokončena a byla pouze v alpha verzi, tudíž spoustu věcí nefungovalo jak by mělo.

## 4.5 Vuex

Knihovna Vuex se využívá pro uložení stavu aplikace. Je potřeba využít u dat, která jsou využívána na více místech a jejich předávání skrz komponenty by bylo jinak složité.

## 4.6 Vue i18n

I18n je rozšíření pro překlady, díky kterému je možné texty v aplikaci napsat v několika jazycích. Nejprve jsem tvořil aplikaci dvojjazyčně v angličtině a češtině, ale nakonec jsem se rozhodl, že prozatím dává aplikace smysl pouze v češtině. Nicméně překlady jsem ponechal a v budoucnu je možné je využít.

## 4.7 Vue Router

V aplikaci je potřeba mít navigaci. Ve Vue se používá Vue Router, který umožní pohyb po různých stránkách.

## 4.8 Firebase

Firebase jsem použil na backendu. To mi umožnilo mít vše na jednom místě. Uložiště, databázi, autorizaci uživatelů, hosting atd.

### 4.8.1 Firestore

Cloud Firestore je NoSQL dokumentová databáze. Narozdíl od SQL databází, které se soustředí na snížení duplikace dat, se dokumentová databáze zaměřuje na časté aktualizace a změny. Největší rozdíl mezi těmito typy je způsob uložení dat. SQL reprezentuje data pomocí tabulek s řádky a sloupci, dokumentová databáze má JSON dokumenty a jejich kolekce. To vede k flexibilnějšímu datovému modelu, rychlejšímu dotazům a lehčímu vývoji pro vývojáře.

Firestore nabízí vlastní řešení bezpečnosti přes Firebase Rules.

## 4.9 Fulltextové vyhledávání

Při práci s Firebase jsem zjistil, že při dotazování se na záznamy není možné filtrovat podle názvu (abych byl přesný, možné to je, ale není to vhodné). Po zkoumání dokumentace, jsem našel stránku s doporučením pro fulltextové hledání. V nabídce byla tři řešení.

- Elastic
- Algolia
- Typesense

Problém jsem řešil s vedoucím a přišli jsme na několik možných řešení sami. Stáhnout si data o všech receptech ve formátu *id:name* a poté filtrovat výsledky hledání na FE. Dále bychom mohli použít Firebase Cloud Functions, kde bychom využili hashování. Nakonec jsme se ale rozhodli, že využijeme jedno z nabízených řešení přímo Googlem.

Vybrali jsme Algolii, kvůli dobré podpoře Vue a Firebase, nejmenší složitosti implementace a bezplatnému základnímu plánu.

Při vývoji jsem ale zjistil, že základní režim (tedy 10 000 čtení za měsíc) nejspíše stačit nebude. Nakonec jsem proto přešel na frontendové vyhledávání. Všechna data jsem stáhnul při načtení aplikace a poté s nimi dál pracoval.

## 4.10 PWA



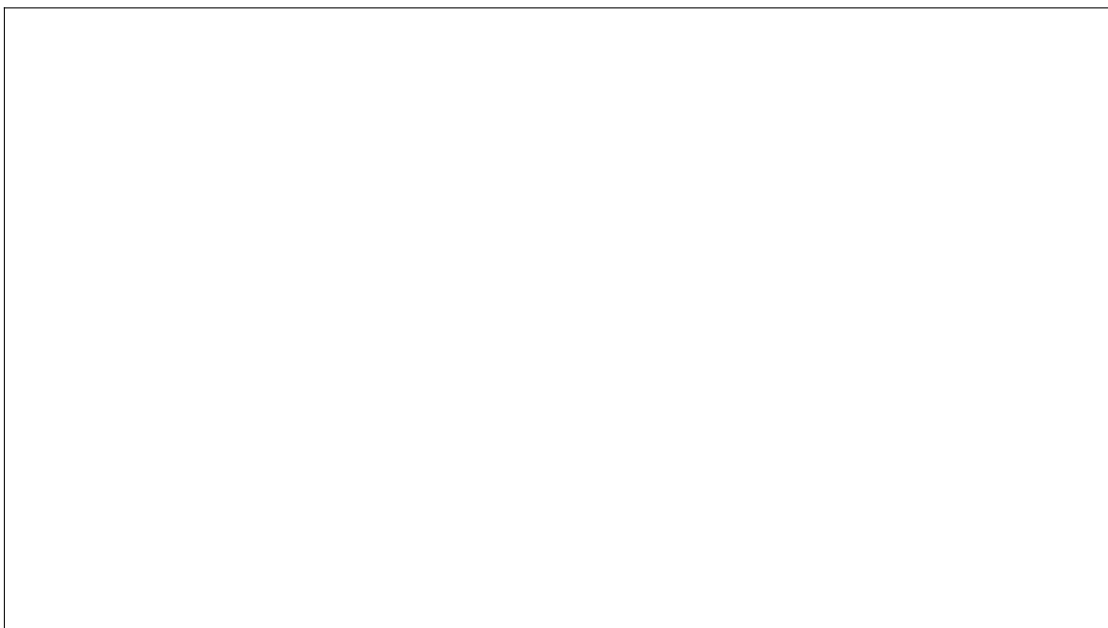
# Návrh

## 5.1 Vzhled aplikace

Při navrhování aplikace se většinou začíná s takzvanými wireframy. Wireframe je rozložení prvků aplikace, které ještě nemají žádný vzhled. Jsou to například tlačítka, sekce pro různý obsah atd. Já jsem zvolil lehce odlišný přístup, tedy vytvoření wireframu již s designem. K této tvorbě jsem použil nástroj Adobe XD [8].

U vzhledu jsem se inspiroval moderními operačními systémy jako např. Windows 11 [9], One UI [10] (nastavba Android od společnosti Samsung) atd. Chtěl jsem dosáhnout jednoduchého, přehledného designu, který bude pro uživatele příjemný na používání. Postupně jsem se pracoval přes několik verzí a ve výsledku jsem zvolil průhledný styl podkladu komponent s *mesh gradientem* na pozadí.

Již od počátku jsem bral v potaz rozdíly mezi zobrazením na mobilu a počítači, díky čemuž jsem mohl design uzpůsobit pro více typů obrazovek. Začal jsem vytvářením úvodní obrazovky, na které se zobrazí rychlé odkazy na nejdůležitější části aplikace, vyhledávání a logo. Tato stránka by měla být přehledná a čistá, proto jsem zvolil ikonky, které vystihují dané odkazy. Díky tomu se při zobrazení na mobilu se tak mohou skrýt pomocné texty.



■ **Obrázek 5.1** Hlavní obrazovka

Dále jsem pokračoval se zobrazením receptů. Na verzi pro počítač jsem na levé straně umístil filtrování a zbylou plochu jsem ponechal pro samotné recepty. Na menších obrazovkách se filtrování přesune nad seznam receptů a již nebude viditelné při posunutí směrem dolů. Na „karty“ s recepty jsem vybral nejdůležitější informace. Tedy název, štítky a obrázek receptu. Nejprve jsem přidal do spodní části karty i ozdobný motiv, později při implementaci jsme se ale s vedoucím dohodnuli, že pouze zabírá místo a bude lepší jej odebrat.

Zobrazení samotného receptu je na velkých a malých obrazovkách velmi odlišné. Zatímco na počítači se uživateli zobrazí všechny informace v několika oddělených obdélnících, na mobilu se každá z těchto částí rozdělí na samostatnou záložku. Také se na spodku obrazovky zobrazí lišta, pomocí které je možné záložky přepínat.

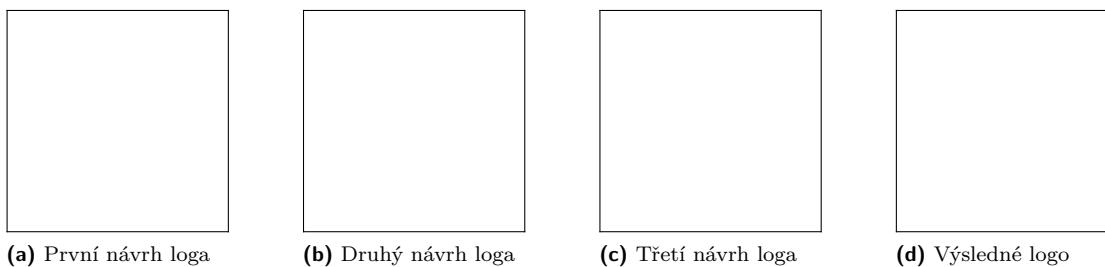
Seznam ingrediencí je stejný jako u receptů. Pro zobrazení ingredience platí totéž.

Plánovač jsem navrhnul jako kalendář, do kterého se dají přesunout recepty pomocí *drag and drop*. Pro rychlý návrh receptu jsem zvolil schéma podobné dnešním seznamkám. Tedy přijmutí popřípadě odmítnutí navrženého receptu je po stranách zobrazené karty a vlevo ze nachází zásobník již zvolených receptů.

## 5.2 Název a logo

Pro aplikaci bylo potřeba vybrat název a logo. Vzhledem k tomu, že středem všeho jsou recepty, hrál jsem si s anglickým slovem *recipe*, až jsem se dostal na Recipeo. Tento název se líbil mně i vedoucímu, tak jsme zaregistrovali doménu [www.recipeo.cz](http://www.recipeo.cz). Poté jsem založil nový soubor v Photoshopu a pustil se do vytváření loga. Chtěl jsem vytvořit jedno s celým názvem, které bychom použili na úvodní stránce a jedno menší, které by se dalo využít jako ikona aplikace, favicon atd.

Zvolil jsem několik fontů z Adobe Fonts a vytvořil návrhy. Menší logo jsem se rozhodl udělat jako první písmeno názvu na pozadí, které se použije v aplikaci. Návrhů bylo celkem šest a na Slacku, který jsme využívali pro komunikaci, jsme hlasovali, které je nejlepší.

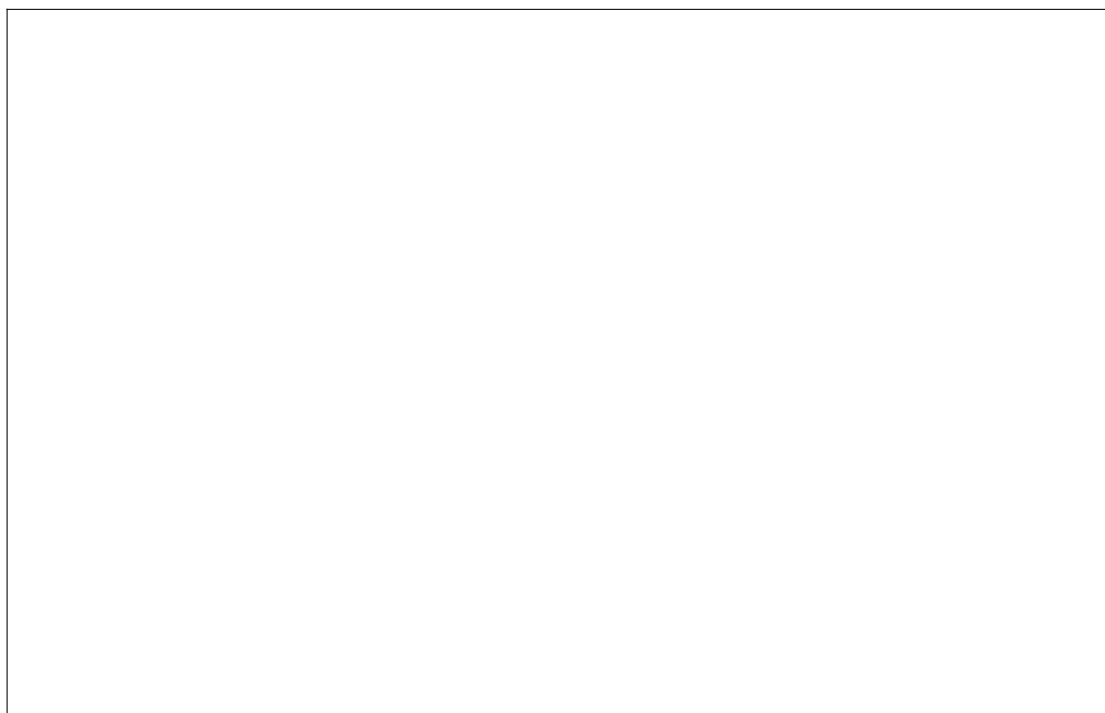


■ **Obrázek 5.2** Vývoj loga

## 5.3 Databáze ve Firebase

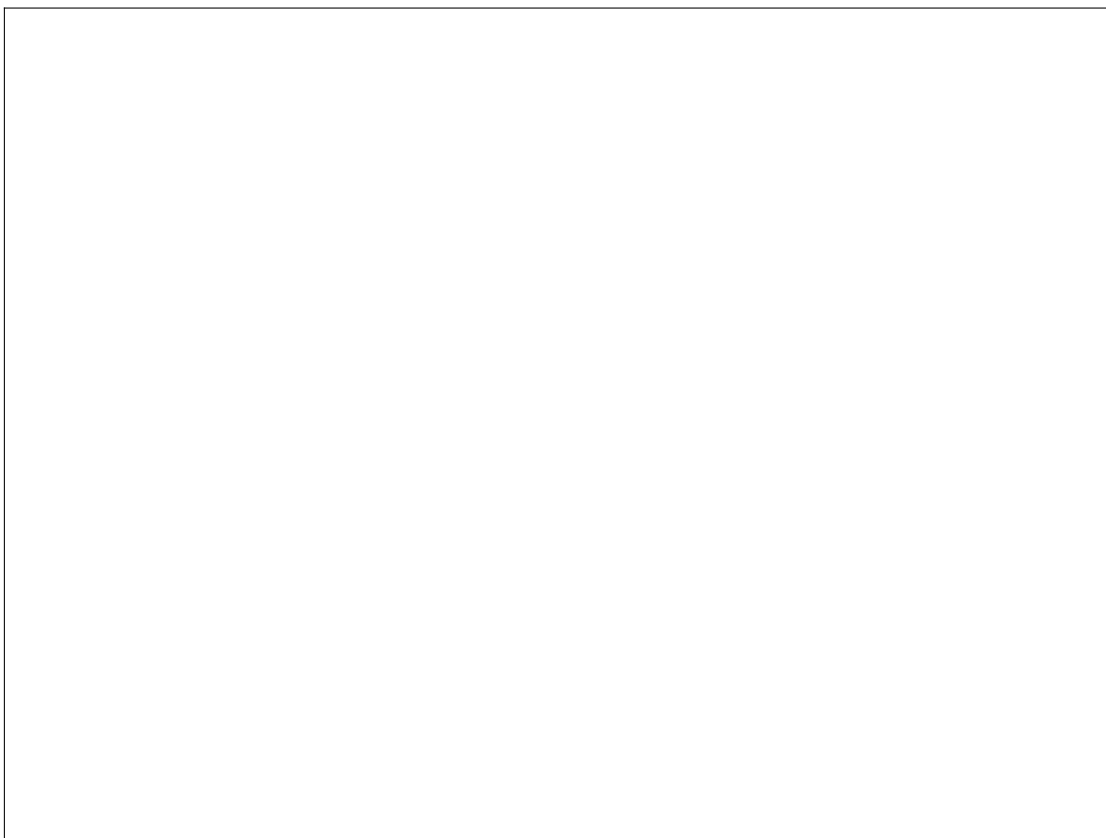
Jak jsem již zmínil, *Firestore* je databáze ve službě *Firebase*. Vytvořil jsem doménový model, který jsem ale poté musel přetransformoval pro dokumentovou databázi.

### 5.3.1 Doménový model



■ **Obrázek 5.3** Doménový model

### 5.3.2 Schéma v dokumentové databázi



■ **Obrázek 5.4** Struktura dat ve Firestore

## 5.4 Uložiště ve Firebase

Firebase poskytuje i vlastní uložisko dat pod názvem *Cloud Storage*. Zde je potřeba ukládat externí obrázky, které uživatelé nahrají k receptu či ingredienci. Strukturu složek jsem zde zvolil stejně jako v databázi.

## 5.5 Data v aplikaci

Kromě uložení dat na serveru je nutné s nimi pracovat také na frontendu. Zde jsme zvolili *Vuex* pro správu dat. Původně můj návrh počítal s tím, že se data budou stahovat postupně, až když je bude uživatel potřebovat. Fungovalo by to tak, že by se například na stránce pro recepty zobrazilo 30 receptů a až když by uživatel prošel přes všechny na konec obrazovky, stáhnuly by se další. S tímto řešením ale nastal problém, protože jsme spoléhali na to, že na fulltextové vyhledávání použijeme službu *Algolia*. Na konec kvůli složitosti řešení a kvůli tomu služba byla nákladná po překročení limitů, jsme tuto možnost zavrhnuli.

Zvolil jsem tedy vyhledávání na frontendu, k čemuž je ale potřeba mít všechna data již při startu aplikace. Začal jsem tedy zkoumat, zda je možné využít cachování a tím zefektivnit čtení z databáze. *Firestore* poskytuje cache již v základu a bylo tedy potřeba ji pouze aktivovat.



## 5.6 Funkce aplikace



## Kapitola 6

# Implementace

Tato kapitola popisuje tvoření prototypu aplikace od založení projektu po vydání první verze. Zmíním zajímavé problémy, které během implementace nastaly a jak jsem je řešil. Vše staví na předchozích kapitolách, tedy analýze, návrhu a použití technologií, které jsem představil dříve.

### 6.1 Založení projektu

Před touto prací jsem neměl žádné zkušenosti se zakládáním větších projektů. Domluvil jsem se tedy s vedoucím a společně jsme založili projekt pro *Vue* aplikaci. Měli jsme již připravený *Github* repozitář, ve kterém jsme projekt verzovali.

Pro založení jsme využili *vue ui*, což je grafické rozhraní pro správu *Vue* projektů. Zde jsme zvolili konfiguraci a přidali pluginy. Poté bylo potřeba některé z nich inicializovat.

#### 6.1.1 Firebase

Nejdříve jsme založili projekt ve *Firebase*. Stačí se přihlásit, otevřít konzoli a kliknout na tlačítko pro vytvoření projektu. Poté se zadá název, popřípadě se projde konfigurací *Google Analytics*. Rovnou jsme přešli na *Blaze plan*, díky kterému se otevřelo více možností. Je ale potřeba si kontrolovat, že aplikace nepřesáhne žádné limity [**FirebaseLimits**], jinak se strhne částka ze zadané platební karty.

Po založení se do projektu přidá aplikace. Díky tomu *Firebase* vygeneruje kód, který se musí do aplikace přidat. Pomocí `npm install firebase` se do projektu přidá potřebný balíček. Poté je potřeba přidat vygenerovaný kód. Dále jsem nainstaloval přes `npm install -g firebase-tools` *Firebase CLI*. Toto rozhraní umožní ovládání všech služeb, které *Firebase* nabízí přímo z konzole. Nakonec je nutné se přihlásit (`firebase login`) a inicializovat projekt (`firebase init`). Při inicializaci je řada možností, vybrat si které služby jsou potřeba, a které nikoliv. Je také možné nastavit automatický *deploy* při nahrání kódu na *Github*.

#### 6.1.2 Automatický deploy

Je velmi pohodlné, když se při změně kódu automaticky nahraje nová verze i na produkci. Proto jsem tuto funkci nastavil hned při zakládání projektu a po celou dobu vývoje, jsem měl do pár minut po nahrání dostupnou verzi pro testování odkudkoliv.

Při `firebase init` se vytvoří *yml* soubory v složce *.github*. Díky těmto souborům se pak při *pull requestu* nebo *merge* spustí proces, který sestaví aplikaci a výsledný *build* nahraje na *Firebase Hosting*.

<code>.github</code>	adresář s konfiguračními soubory Githubu
<code>dist</code>	adresář pro sestavenou produkční verzi
<code>functions</code>	adresář s implementací Firebase Functions
<code>node_modules</code>	adresář s moduli staženými pomocí Yarn/npm
<code>public</code>	adresář s obrázky aplikace a základním souborem
<code>src</code>	zdrojové kódy
<code>assets</code>	obrázky využité v aplikaci
<code>components</code>	Vue komponenty
<code>enum</code>	
<code>lang</code>	překlady
<code>plugins</code>	obaly použitých knihoven
<code>router</code>	adresář pluginu Vue router
<code>service</code>	
<code>store</code>	adresář pluginu Vuex
<code>style</code>	CSS soubory
<code>service</code>	
<code>store</code>	adresář pluginu Vuex
<code>style</code>	CSS soubory
<code>views</code>	Vue komponenty využívané ve Vue router pro zobrazení stránek
<code>firebase.json</code>	konfigurační soubor Firebase
<code>firebaseConfig.js</code>	tajný konfigurační soubor Firebase
<code>package.json</code>	soubor obsahující seznam závislostí a konfiguraci aplikace

■ Obrázek 6.1 Adresářová struktura

## 6.2 Struktura projektu

Vytvoření složek a základní rozdělení jsem ponechal na *vue ui*. Konfigurační soubory se nachází v *rootu* projektu, implementace je poté rozdělena podle zaměření ve složce **src**. Dále se při přidání *Firebase Functions* objevila složka **functions**, která obsahuje kód, který se nahrává na server a dá se poté využít jako jednoduché API. Složka **public** obsahuje základní soubory, jako je **index.html**, do kterého se při přístupu na stránku vloží JavaScriptový kód nebo různé ikony, jako je logo. Pro výsledný *build*, který se nahraje na Firebase hosting se používá složka **dist**.

## 6.3 Router

Jako první věc jsem si připravil základní cesty aplikace. Cesty se přidávají jako pole objektů, kde objekt obsahuje políčka *path*, *name*, *component* a *meta*. Všechny tyto parametry nejsou povinné a u některých cest jsou i další, které popíšu později. *Path* je string, který následuje za adresou stránky např. *www.recipeo.cz/example*. *Name* je název dané cesty, který se používá interně v kódu. Tato vlastnost se hodí, když programátor chce změnit adresu stránky. Díky využití názvu cesty ji nemusí všude v aplikaci přepisovat. *Component* specifikuje komponentu, která se na adrese zobrazí a *meta* jsem zde využil pro změnu titulku stránky.

Pro některé stránky bylo nutné ověřit, zda je uživatel přihlášený. K tomu lze využít *navigation guards*. Tato ochrana se spustí vždy před přístupem na stránku a vyhodnotí, zda je požadavek validní. Pokud není, uživatel je přesměrován na jinou stránku.

Také jsem využil globální úpravy *routeru*, pro aktualizaci názvu stránky. Funguje na stejném principu jako výše zmíněná ochrana, ovšem provede se na všech stránkách.

```

async function authGuard(to, from, next) {
  if (!await Auth.getCurrentUser()) next({ name: "Login" })
  else {
    next()
  }
}

```

■ **Výpis kódu 1** Příklad ochrany stránky proti nepřihlášeným uživatelům

```

{
  path: "/account",
  name: "Account",
  component: Account,
  meta: { title: "account" },
  beforeEnter: authGuard
}

```

■ **Výpis kódu 2** Použití Auth Guardu na stránce profilu uživatele

## 6.4 Překlady

Ačkoliv jsme se nakonec s vedoucím rozhodli aplikaci ponechat pouze v češtině, již od začátku jsem ji psal dvojjazyčně a to sice česky a anglicky. Použil jsem plugin *vue-i18n*. Pro překlady se využívají *.js* soubory, ve kterých se pomocí objektů strukturují cesty, na které se poté odkazuje ve Vue komponentách.

```

recipes: {
  recipes: "Recepty",
  visibility: {
    public: "Veřejný",
    private: "Soukromý",
    unlisted: "Neveřejný"
  },
  noRecipes: "Žádné recepty nenalezeny"
}

```

■ **Výpis kódu 3** Překlad pro recepty

Překlad se vyvolá pomocí funkce *\$t*, které se jako parametr předá cesta překladu.

## 6.5 Vuex

Pomocí Vuex jsem spravoval data přímo v paměti. Vždy po spuštění aplikace se sem nahrála data receptů, ingrediencí, štítků nebo údaje o uživateli. Pomocí *actions* se volal mnou vytvořený obal pro *Firestore*, odkud se stáhla data buď ze serveru nebo s lokální cache. Poté se přes *mutations* přidala nová data přímo do Vuexu. Zvolil jsem data, kterých bylo více jednoho typu, uchovávat v objektech místo obyčejných polí. Získal jsem tím konstantní časovou složitost při přístupu k prvku přes identifikátor. Také jsem díky tomu nemusel kontrolovat, zda recept s daným *id*

```
<span>{{ $t(recipes.noRecipes) }}</span>
```

#### ■ Výpis kódu 4 Použití překladu

existuje, když se stahovala aktualizace. Recept se pouze přepsal novými daty.

## 6.6 PWA

Co se týče *PWA*, změnil jsem v konfiguračním souboru název a tématickou barvu. Poté jsem přidal komponentu pro aktualizaci aplikace. Uživateli se zobrazí, když je na server nahrána nová verze. Uživatel poté klikne na tlačítko *Obnovit* a aplikace se spustí s novou verzí. Tento dialog se mi nejdříve nedařilo správně zobrazovat, protože jsem posílal špatný typ zprávy pro *service worker*. Místo objektu s klíčem *type* jsem posílal pouze string se zprávou.

```
this.registration.waiting.postMessage({ type: 'SKIP_WAITING' })
```

#### ■ Výpis kódu 5 Správná zpráva pro aktualizaci

## 6.7 Stylování aplikace

Vzhled aplikace se při vývoji přizpůsobil knihovně Vuetify, ale celkový koncept zůstal zachován. Některé části aplikace se pročistily a staly se více přehledné. Začínal jsem pouze se světlým režimem, ale v průběhu jsem přidal i tmavý mód. Bylo potřeba zvolit tmavší pozadí, tak aby bylo podobné tomu původnímu. Vytvořil jsem tedy v aplikaci *Illustrator* nový projekt a pustil se do tvoření. Barvy jsem zvolil podle světlého pozadí, ale ztmavil jsem je.

Pro různé režimy jsem odlišil barvy prvků na stránce. Ve světlém jsem se rozhodl pro výrazně modrou `#1d7dee` a k tomu pastelově zelenou `#79ffa1`, naopak v tmavém jsem použil tlumené odstíny těchto barev a to sice `#4372b4` a `#4ca262`. Pro změnu barev jsem využil Vuetify a jeho nastavení *theme*. Komponenty z Vuetify mají většinou *prop color*, díky kterému je možné změnit barvu. Pro prvky mimo Vuetify je možné využít CSS proměnné, která se automaticky vytvoří.

## 6.8 Firebase

### 6.8.1 Firebase Auth

*Firebase* dokáže řešit i přihlášení uživatele. Nejprve jsem chtěl mít dva způsoby přihlášení, přes email a heslo a přes Google. Nakonec po domluvě s vedoucím jsem zvolil prozatím ponechat pouze Google přihlášení, abychom nemuseli řešit ukládání dat uživatele jinam než právě do *Auth*. Zapnutí různých možností autentizace je možné ve webové konzoli. V případě Google stačilo přidat kontaktní email a služba se spustila.

Na frontendu jsem tedy vytvořil formulář pro přihlášení a registraci. Pro budoucí využití při případném přechodu na jinou formu autentizace se tyto formuláře hodí. Ale v první verzi bude zpřístupněn pouze Google *sign-in provider* a to sice přes *pop-up*, který se uživateli zobrazí při kliknutí na tlačítko přihlášení. Metoda která tuto funkčnost poskytuje se dá nainportovat přímo z *Firebase*. Musel jsem ale upravit konfigurační soubor, ve které jsem změnil hodnotu *authDomain* na *recipeo.cz*. Díky tomu se *pop-up* otevře na naší doméně a ne na původní vygenerované.

## 6.8.2 Firestore cache

### 6.8.2.1 Algolia

K implementaci cache jsem přistoupil, když bylo jasné, že Algolia je nepoužitelná. Musel jsem tedy vymyslet jiný způsob fulltextového vyhledávání a jako nejlepší způsob se nabízelo stáhnout všechna potřebná data a filtr aplikovat přímo na zařízení uživatele. To je ale spojeno s nákladnými operacemi nad databází při každém spuštění aplikace, a proto bylo nutné implementovat cache.

Základní spuštění je velmi jednoduché, stačí zavolat importovanou metodu z Firebase.

```
enableIndexedDbPersistence(db, {forceOwnership: true} )
.catch((err) => {
  if (err.code === "failed-precondition") {
    console.log("Multiple tabs open, persistence can only be enabled
      in one tab at a a time.")
  } else if (err.code === "unimplemented") {
    console.log("The current browser does not support all of the
      features required to enable persistence")
  }
})
```

#### ■ Výpis kódu 6 Zapnutí perzistentního módu

Jako parametr je potřeba předat instanci Firestore, kterou lze získat s parametrem pro neo-mezenou velikost, tím pádem se nebudou mazat záznamy kvůli úspoře místa.

```
const db = initializeFirestore(FirebaseApp, {
  cacheSizeBytes: CACHE_SIZE_UNLIMITED
})
```

#### ■ Výpis kódu 7 Získání instance Firestore

Poté jsem vytvořil funkci, která se starala o to, odkud se mají data stahovat. Při každém stažení jsem si v *local storage* aktualizoval časovou značku a díky tomu jsem mohl stahovat pouze data, se kterými mezitím bylo manipulováno. Zde se objevilo několik problémů. Jeden z nich bylo to, že jakmile se přihlásil jeden uživatel, poté se ihned odhlásil a přihlásil se jiný, recepty se nestáhly správně. Bylo proto potřeba aktualizovat časovou značku při každé manipulaci s autorizací.

```

async getData () {
  if(!lastUpdated || !expirationDate) {
    // Download everything from server and add timestamps to localStorage.
    addTimestamps()
    Store.dispatch("getData", { source: FirestoreSource.SERVER, updateOnly: false })
    return
  }

  if(Date.now() > expirationDate) {
    // Download everything from server and add timestamps to localStorage.
    addTimestamps()
    Store.dispatch("getData", { source: FirestoreSource.SERVER, updateOnly: false })
  }
  else {
    // Get data from cache, call query to update new data and set last updated timestamp.
    localStorage.lastUpdated = Date.now()
    await Store.dispatch("getData", { source: FirestoreSource.CACHE, updateOnly: false })
    await Store.dispatch("getData", { source: FirestoreSource.SERVER, updateOnly: true })
  }
}

```

■ **Výpis kódu 8** Metoda pro stažení dat

### 6.8.3 Firestore rules

Bezpečnost je důležitá součást jakéhokoliv softwaru. Firebase proto nabízí takzvané Firestore Rules pro zabezpečení dat na serveru. Díky těmto pravidlům je možné filtrovat požadavky a v případě nutnosti zamítnout přístup k datům. Zvolil jsem pravidla na základě identifikace pomocí *uid* neboli uživatelského identifikátoru. Pravidla fungují tak, že se pomocí cesty ve Firestore vyhledá pravidlo a pokud je splněna podmínka, data se odešlou. Defaultně jsou všechny zdroje privátní, tedy nikdo v nim nemá přístup.

```

match /{path=**}/recipes/{doc} {
  allow read: if resource.data.visibility == "public";
}

```

■ **Výpis kódu 9** Pravidlo pro přístup k veřejnému receptu

### 6.8.4 Firestore Functions

Cloudové funkce jsem využil pro místa v aplikaci, kde nebylo možné pomocí Firestore Rules správně ošetřit bezpečnost dat. Poprvé jsem napsal funkce, díky kterým se stáhnul určitý počet receptů, ke kterým měl uživatel přístup. Využil jsem stránkování, což znamená, že když uživatel přišel na stránku s recepty, stáhlo se pouze omezené množství a až když bylo potřeba, aplikace požádala server o další. Díky tomu jsem nemusel vytvářet žádná pravidla, stačilo nechat celou databázi zamknutou, protože ve Functions se přistupuje k datům přes admin účet, který má práva na veškeré operace. Nakonec toho ale nebyl vhodný přístup, a tak jsem funkce odstranil.

Kde se tento přístup hodil bylo u pozvánek do skupiny. Pozvanému uživateli se musí zobrazit informace o skupině, ten ale ještě ve skupině není, a tak nemá k těmto datům přístup.



Proto se zavolá z aplikace funkce *getGroup*, která odešle zpátky záznam dané skupiny. Poté se uživateli zobrazí dialog s pozvánkou a tlačítko *Připojit*. Po stisku tohoto tlačítka se opět zavolá jedna z funkcí, tentokrát *joinGroup*. Ta zapíše identifikátor uživatele do pole *userIds*, které je v dokumentu dané skupiny. K tomu se dá využít funkce *arrayUnion*, kterou poskytuje Firebase k jednoduchým operacím na poli. Díky tomu se nemůže stát, že by nastala duplikace jednoho uživatele.

Firebase nabízí různé typy emulátorů, ovšem jediný který jsem využil byl právě *Functions emulator*. Díky tomu jsem mohl lokálně testovat, jak vypadají data, která odesílá server a nemusel pokaždé čekat, než se funkce nahrají na server, což při větším počtu trvá až několik minut. Také jsem tento lokální server využil při problémech s právy. Několikrát se mi stalo, že funkce spadla či mi odepřela přístup. Pomocí emulátoru jsem tak zjistil, zda je špatně napsaná funkce nebo není správně nakonfigurovaná. Poté jsem našel v dokumentaci našel, že je potřeba u některých funkcí přidat roli pro autorizování volání funkce v Google Cloud platformě.





## Kapitola 7

# Testování

### 7.1 Uživatelské testování

Po dokončení bude aplikace podrobena uživatelskému testování.



# Možnosti aplikace v budoucnosti

Během vývoje nastaly různé změny a některé funkce byly moc složité na implementaci nebo nebyly nutné v první verzi. V této kapitole nastíním, jakým směrem by se aplikace mohla vydat.

## 8.1 Interakce mezi uživateli

V prototypu je možné pouze vytvořit skupinu, pozvat do ní ostatní uživatele a přidat do ní recepty. Původně jsem zamýšlel i textovou komunikaci, ale nakonec jsme se s vedoucím shodli, že zatím není potřeba. Ovšem do budoucna by se mohla takováto funkce hodit, v kombinaci s komentáři a hodnocením receptů. Poté by uživatelé mohli vybírat mezi veřejnými recepty podle jejich kvality a sdílet své pocity či vychytávky.

Na to by se dalo navázat vytvořením profilů uživatelů. Na tomto profilu by uživatelé mohli přidávat příspěvky jako je tomu například na Instagramu. Na této síti jsou populární krátká videa s recepty a tím pádem je zde potenciál pro případný příliv uživatelů.

## 8.2 Časovač

U návrhu zobrazení receptu na mobilním zařízení jsem přidal časovač. V receptu by mohla být tlačítka u daných kroků, která by spustila časovač na limit, který byl stanoven při vytvoření. Seznam těchto měření by byl dostupný jako plovoucí tlačítko v celé aplikaci a uživatel by tak k vaření potřeboval pouze mobilní zařízení.

## 8.3 Verze FIT

Kromě normálních receptů, by se dala aplikace přepnout do fitness módu, kde by zobrazovala pouze zdravé recepty. Nabídla by také sledování kalorií, plány pro zlepšení stravy či jiné návrhy na úpravu jídelníčku.

## 8.4 Pohodlí pro uživatele

Jak jsem zmínil ve výsledcích kvalitativního průzkumu, uživatelé mají často problémy se zašpiněním displeje na jejich zařízení při vaření. Zabránit nepotřebného kontaktu se zařízením by mohlo pomoci například vynucení, aby obrazovka nezhasnula. Bylo by potřeba prozkoumat *Screen Wake Lock API* nebo nějaký plugin třetí strany.

## 8.5 Propojení se službami Košík a Rohlík

Tato práce měla být původně rozdělena na dvě, na frontend a na backend. Nakonec se nepodařilo sehnat studenta na backendovou část, a tak jsem musel zpracovat fullstack aplikaci. Nezbyl tedy čas na implementaci doporučování, jednoduché přidání do košíku a dalších funkcí napojené na služby Rohlíku a Košíku.

V budoucnu je ale jistě priorita tyto funkce zpřístupnit. Doporučení receptů by šlo založit na aktuálních slevách, daném ročním období či uživatelské nákupní historii. S nákupy se pojí automatické přidání do spíže a sledování počtu surovin. Toto rozšíření spočívá v přidání privátních ingrediencí, ke kterým si uživatel nebo skupina může přiřadit vlastní odkazy na zmíněné služby a počet kusů.

## 8.6 Skupiny

Při pozvání se nekontroluje žádný časový úsek či jiné ověření zda je pozvánka platná. Přidáním tokenu by se dala omezit doba platnosti a tím zabránit nechtěnému šíření. Také stojí za zvážení operace nad členy skupiny. Tedy odebrání členů či smazání skupiny, které je zatím nemožné.



## Kapitola 9

# Závěr

Cílem práce bylo navržení a vytvoření webové aplikace, která usnadní manipulaci s recepty, surovinami či jejich nákupem. Povedlo se naplnit všechny původní požadavky a díky průzkumu přidat další užitečné funkce.

Ze začátku jsem se zabýval průzkumem konkurenčních řešení, kvalitativním průzkumem mezi potencionálními uživateli. Poté jsem pokračoval návrhem papírových modelů a designem, což jsem spojil v jednu věc. Následovala volba technologií a tvorba celého konceptu jak vše bude fungovat. Mezitím jsem se domlouval s poskytovateli online nákupů surovin, zda by bylo možné využít jejich služeb. Také jsem průběžně implementoval a přidával nové změny, které vzešly z problémů, na které jsem v průběhu práce narazil. Nakonec byl konečný prototyp podroben uživatelskému testování.

Do budoucna se nabízí spousta možností okolo poskytnutých služeb od Rohlíku a Košíku. S tím také souvisí tvorba vlastního API, které by aplikace pro tyto operace určitě potřebovala. Tím by se také osvobodila od různých limitů, které využívané technologie s sebou přináší (potažmo verze které jsou zdarma, po překročení limitů začíná být výhodnější platit pouze hosting pro vlastní řešení).





..... Příloha A

# Nějaká příloha



**(a)** První návrh loga



**(b)** Druhý návrh loga



**(c)** Třetí návrh loga

■ **Obrázek A.1** Návrh velkého loga



**(a)** Třetí návrh loga



**(b)** Třetí návrh loga



**(c)** Třetí návrh loga

■ **Obrázek A.2** Návrh velkého loga





Příloha B

## Příloha 2



# Bibliografie

1. *Portál vareni.cz* [online]. Vareni.cz [cit. 2022-03-13]. Dostupné z: <https://www.vareni.cz>.
2. *Toprecepty.cz* [online]. Toprecepty.cz [cit. 2022-03-13]. Dostupné z: <https://www.toprecepty.cz>.
3. *Recepty.cz* [online]. Recepty.cz [cit. 2022-03-13]. Dostupné z: <https://www.recepty.cz>.
4. *Zdravý stůl* [online]. Zdravý stůl [cit. 2021-12-04]. Dostupné z: <https://www.zdravystul.cz/>.
5. *VueJS* [online]. Vue.js [cit. 2021-11-15]. Dostupné z: <https://v3.vuejs.org/>.
6. *Single-Page Application (SPA)* [online]. Vue.js [cit. 2022-03-13]. Dostupné z: <https://vuejs.org/guide/extras/ways-of-using-vue.html#single-page-application-spa>.
7. *Why you should be using Vuetify* [online]. Vuetify, 2021 [cit. 2021-11-15]. Dostupné z: <https://vuetifyjs.com/en/introduction/why-vuetify>.
8. *Adobe XD* [online]. Adobe [cit. 2022-02-01]. Dostupné z: <https://www.adobe.com/cz/products/xd.html>.
9. *Windows 11* [online]. Microsoft [cit. 2022-02-01]. Dostupné z: <https://www.microsoft.com/cs-cz/windows/windows-11>.
10. *One UI 4* [online]. Samsung [cit. 2022-02-01]. Dostupné z: <https://news.samsung.com/global/one-ui-4-update-delivers-an-elevated-mobile-experience-centered-around-you>.