

Gymnázium, Praha 6, Arabská

Obor programování



Maturitní práce

Lucián Kučera

Recepty

duben 2021

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská¹⁴ oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 30. duben 2021

Anotace

Aplikace slouží pro vytváření receptů s úmyslem, aby všechno bylo uspořádané a velice přehledné. S vytvářením a upravováním receptu pomáhá technika „táhni a pusť“. Uživatel si může určit, jestli jeho recept bude viditelný pro ostatní uživatele, anebo bude soukromý.

Abstract

Main idea of the application is to create recipes and have everything in great order and looking simply. Technique known as drag and drop helps with creating and editing recipes. User can select if his recipe can be seen by others or his recipe is only for his own use.

Zadání projektu

Aplikace bude sloužit pro vytváření receptů a sdílení receptů. K receptu půjde připojit komentář nebo recept půjde označit za oblíbený. Recept se bude skládat z náčiní, ingrediencí, kroků, fotky a popisu. Krok se skládá z ingrediencí, náčiní, popisu a potřebného času. Uživatel si bude moci v aplikaci podle různých kritérií vyhledat recept, například registrovaný uživatel si může zobrazit všechny oblíbené recepty. Aplikace bude mít uživatelsky přívětivé ovládání a přístup v aplikaci se bude řešit uživatelským systémem.

Obsah

Anotace.....	3
Abstract.....	3
Zadání projektu	3
1. Úvod.....	1
2. Technologie	1
2.1 Technologie použité pro vývoj	1
2.1.1 Visual studio code.....	1
2.1.2 Psql	1
2.1.3 Chrome developer tools.....	1
2.1.4 Heroku	2
2.2 Hlavní technologie.....	2
2.2.1 PostgrgeSQL	2
2.2.2 Express.js.....	2
2.2.3 React.js	2
2.2.4 Node.js.....	2
2.3 Technologie serverové části.....	3
2.3.1 Bcrypt	3
2.3.2 Cookie-parser.....	3
2.3.3 Cors.....	3
2.3.4 Jsonwebtoken	3
2.3.5 Pg	3
2.4 Technologie klientské části.....	3
2.4.1 React-hook-form.....	3
2.4.2 Axios.....	3
2.4.3 React-bootstrap	4
2.4.4 Firebase	4
2.4.5 React-dnd	4
2.4.6 React-dnd-multi-backend.....	4
2.4.7 React dnd html5-backend.....	4
2.4.8 React-dnd-touch-backend.....	4
2.4.9 Immutability helper	4
2.4.10 Uuid	5

2.4.11	Struktura souborů.....	5
2.5	Klient	5
2.5.1	Axios.....	5
2.5.2	Components.....	5
2.5.3	Config	5
2.5.4	Context	5
2.5.5	Pages.....	6
2.5.6	Queries.....	6
2.5.7	responsiveCss.....	6
2.5.8	Utils	6
2.6	Server.....	6
2.6.1	Apis.....	6
2.6.2	Configuration	6
2.6.3	Midelware.....	7
2.6.4	query_functions	7
2.6.5	Utils	7
3.	Struktura Backendu	7
3.1	Návrh databáze.....	7
3.1.1	Users	8
3.1.2	Refreshtokens.....	8
3.1.3	Ingredients.....	8
3.1.4	utensils.....	9
3.1.5	Recipies	9
3.1.6	recipie_ingredients.....	9
3.1.7	recipie_utensils	9
3.1.8	recipie_steps.....	9
3.1.9	step_utensils	9
3.1.10	step_ingredients	10
3.1.11	recipie_like	10
3.1.12	comments	10
3.1.13	comments_like	10
3.2	Autentizace.....	10
3.3	Struktura API	11
3.4	Uložiště obrázků	11
4.	Struktura Frontendu.....	11

4.1	Stránka Přihlášení	11
4.2	Stránka Registrace	11
4.3	Hlavní stránka	12
4.4	Formulář receptu.....	12
4.5	Stránka vašeho receptu	13
4.6	Stránka Ingredience a Nástroje	14
4.7	Stránka sdílených receptů.....	15
4.8	Stránka sdíleného receptu.....	16
5.	Funkce	17
5.1	Tvorba receptu	17
5.2	Sdílení receptu	19
6.	Návod na spuštění	20
6.1	Aplikace na Heroku	20
6.2	Konfigurace Firebase.....	20
6.3	Lokální spuštění.....	21
6.4	Instalace na Heroku	22
7.	Závěr	26
8.	Seznam Obrázků.....	27
9.	Bibliografie	27

1. Úvod

Tuto aplikaci jsem vytvořil, protože jsem na trhu neviděl moc webových aplikací, které by poskytovaly uživateli kontrolu nad vytváření receptů, tak jak bych si představoval. Tato aplikace byla navržena hlavně s myšlenkou kontroly a organizace a druhou hlavní myšlenkou bylo rozšiřování, do aplikace je jednoduché přidat nové systémy.

Pro tvorbu aplikace jsem si zvolil tzv. PERN stack PostgreSQL, Express, React a Nodejs, protože tento seznam technologie jsem používal už na svých předchozích projektech například sociální síť a kalendářové aplikaci. Z mých zkušeností se s těmito technologiemi pracovalo velice jednoduše a pohodlně. Jako vývojářské prostředí jsem zvolil Visual Studio Code.

Webová aplikace skládá ze tří hlavních stránek: vaše ingredience a nástroje, sdílené recepty a vaše recepty. Také jsou zde stránky pro recept, formulář receptů, přihlášení, registraci, sdílený recept a váš recept. Všechny stránky mají svoji úlohu a cesty jsou buď pro přihlášené uživatele nebo nepřihlášené uživatele. Další přístup se už řeší na serveru pomocí „middlewareů“, pole funkcí odehrávající se před dotazem.

Používám framework Express, což jsou funkce middleware, které se spouštějí během životního cyklu požadavku na server Express. Každý middleware má přístup k požadavku HTTP a odpovědi pro každou cestu (nebo cestu), ke které je připojen. Samotný Express je ve skutečnosti složen výhradně z funkcí middleware.

Formát receptu je navržen velice flexibilní, takže každý uživatel bude zaručeně spokojený.

2. Technologie

2.1 Technologie použité pro vývoj

2.1.1 Visual studio code

Samotnou aplikaci jsem vytvářel v textovém editoru Visual Studio Code. Visual Studio Code je vytvořeno firmou Microsoft. Visual Studio Code jsem si vybral, protože má skvělý našeptávač, skvělou podporu pro Javascript a pomocí rozšíření může mít podporu na všechny možné jazyky, knihovny a frameworky.

2.1.2 Psql

Pro práci s databází a vyplňování testových dat jsem si zvolil Psql, což je příkazová řádka, která umožňuje práci s PostgreSQL databázemi. Také mi umožnila vložit tabulky na Heroku. Jednoduše se ovládá a má příjemné uživatelské rozhraní.

2.1.3 Chrome DevTools

Chrome DevTools je rozšíření pro webový prohlížeč Chrome, umožňuje vývojářům kontrolovat funkčnost aplikace ve webovém prostředí. Například: kontrola serverových požadavků naší klientské aplikace, jestli se všechno zobrazuje ve správném pořadí anebo optimalizace stránky.

2.1.4 Heroku

Další důležitou technologií využitou pro vývoj aplikace je Heroku, které mi dovolilo hostovat klientskou a serverovou část mého projektu v produkčním prostředí. Heroku považuji za skvělou zkušenost, kterou rozhodně ještě využiji později v životě.

2.2 Hlavní technologie

Hlavní použitá technologie je takzvaný PERNstack. Je to seznam technologií pro vývoj webových aplikací, skládající se z PostgreSQL, React.js, Express.js a Node.js.

2.2.1 PostgreSQL

PostgreSQL (PostgreSQL Global Development Group, 2021) je mocný, open source relační databázový systém, který si získal silnou reputaci kvůli svojí spolehlivosti a výkonu. SQL databáze slouží pro ukládání dat v tabulkách. Tabulky se skládají z řádků, které obsahují předdefinovaný model objektu.

2.2.2 Express.js

Express.js (Holowaychuk, 2021) je JavaScriptový framework pro vytváření serveru webových aplikací. Pro tvorbu serveru jsem si vybral framework Express, který nabízí funkce middleware, jenž se spouští během životního cyklu požadavku na serveru. Každý middleware má přístup k požadavku HTTP a odpovědi pro každou cestu, ke které je připojen. Samotný Express.js je ve skutečnosti složen výhradně z funkcí middlewaru. Express.js mi umožnil velice snadno vytvořit vlastní robustní API, které jsem mohl využít z klientské aplikace.

2.2.3 React.js

React.js (Walke, 2021) je JavaScriptová knihovna, často se plete s frameworkem, pro vytváření front-endu webových aplikací, která funguje na všech moderních webových prohlížečích.

React.js je založený na tzv. komponentách. React.js komponenta je soubor, který implementuje React.js a skládá se ze dvou povinných složek. První složka je funkce, která může obsahovat JavaScript. Druhá složka je funkce vracející html.

React.js umožňuje tvorbu vlastních komponent, komponenta může být tvořena buď funkcí nebo třídou. Ve svém projektu jsem použil funkční komponenty, protože je to považováno komunitou za moderní a také to zjednodušuje práci se stavem jednotlivých komponent.

Dále React.js nabízí jednoduchou práci s životním cyklem aplikace a stavem aplikace pomocí tzv. háků (hooks), háky jsou funkce, které mají různá využití. Ve funkčních komponentách se využívá hák *useEffect()*, který je *void* a nic nevrací. Jako parametr bere funkci a pole proměnných. Pole proměnných určuje, kdy se má komponenta znovu renderovat. Funkce se skládá ze dvou částí. První část se odehraje při každém znovu renderování komponenty. Druhá část určuje, co se má udělat při odpojení (unmountu) komponenty.

2.2.4 Node.js

Node.js (Dahl, 2021) je JavaScriptové prostředí pro tvorbu webových aplikací. Node.js také zahrnuje npm (Node package manager), npm se stará o všechny stažené externí Node balíčky.

2.3 Technologie serverové části

2.3.1 Bcrypt

Bcrypt (<https://github.com/kelektiv/node.bcrypt.js#readme>, 2021) je funkce pro hashování hesel navržená Nielsem Provosem a Davidem Mazièresem, založená na šifře Blowfish a představená na USENIX v roce 1999. Bcrypt využívá sůl, která pomáhá bránit se útokům duhové tabulky. Uživatel si může nastavit počet iterací hashování hesla, s počtem hashovacích iterací se exponenciálně prodlužuje čas trvání, ale čím více iterací, tím je heslo bezpečněji chráněno.

2.3.2 Cookie-parser

Cookie-parser (<https://github.com/expressjs/cookie-parser>, 2021) je Node balíček, který umožňuje čtení obsahu cookies, tuto funkci využívám při kontrole, jestli je uživatel nadále přihlášen.

2.3.3 Cors

Cors (<https://github.com/expressjs/cors>, 2021) je Node balíček, který umožňuje bezpečnou komunikaci mezi klientem a serverem. Zaručuje, že server zná adresu klientských aplikací, a tak nedochází k chybám spojeným s cizím původem požadavku

2.3.4 JSON Web Token

Jsonwebtoken (<https://github.com/auth0/node-jwebtoken>, 2021), je Node balíček. JSON Web Token (JWT) je internetový standard pro vytváření dat s volitelným podpisem a volitelným šifrováním. JWT obsahuje data ve formě JSON (JavaScript Object Notation). V mé aplikaci JWT využívám na uložení uživatelského id a potvrzení o tom, že je přihlášený. Konkrétně v mé webové aplikaci používám šifru HS256 pro moji vlastní autentifikaci. HS256 funguje na principu jednoho tajemství, kterým se data zašifrují a nazpět odšifrují. Firebase JWT využívá šifru RS256. Tento algoritmus funguje na principu privátního klíče a veřejného klíče. Já mám jenom privátní klíč a tím zašifruji data pro Firebase. Firebase si poté data rozšifruje vlastním klíčem a postará se o validaci dat.

2.3.5 Pg

Pg (node-postgres) (<https://github.com/brianc/node-postgres>, 2021) neboli obecně známý Node balíček pod názvem Node-Postgres. Tento balíček umožňuje připojit PostgreSQL databázi k Node.js prostředí a spravovat připojení. Například na Heroku se vyžaduje ssl připojení databáze v Node.js prostředí.

2.4 Technologie klientské části

2.4.1 React-hook-form

React-hook-form (<https://github.com/react-hook-form/react-hook-form>, 2021) tento Node balíček slouží pro práci s formulářem. Hlavní funkcí je zpřehlednění a validace formuláře a práce se stavem komponenty. Práci se stavem komponenty vykonává zcela za vás.

2.4.2 Axios

Axios (<https://github.com/axios/axios>, 2021) je jeden z nejdůležitějších Node balíčků použitých v mé aplikaci. Axios dává klientovi možnost komunikovat se serverem a následovně zpracovat odpověď serveru. Další velice důležitou funkcí Axiosu jsou

interceptory požadavků. Interceptor je funkce, která se odehraje před každým požadavkem dotyčné instance Axiosu. Interceptory využívám pro umístění přístupového JWT tokenu do autorizačního headeru. Poslední funkcí Axiosu, která je pro mě velice důležitá, jsou Cancel tokeny. Cancel tokeny zruší požadavek, pokud by došlo k odpojení komponenty, která konkrétní požadavek vysílá, tak se předejde tzv. úniku paměti. K úniku paměti může dojít, když se komponenta odpojí při změně stavu v komponentě.

2.4.3 React-bootstrap

React-bootstrap (<https://github.com/react-bootstrap/react-bootstrap>, react-bootstrap, 2021) je balíček tvořící UI mojí aplikace. Velice se mi líbily komponenty *Container*, *Row* a *Col* (kontejner, řádek a sloupec), tyto konkrétní komponenty ulehčují práci s rozložením v aplikaci. Díky tomuto balíčku je UX (user experience, uživatelský zážitek) velice přívětivá.

2.4.4 Firebase

Firebase (Google, 2021) je balíček, který zprostředkovává komunikaci klienta s Firebase aplikací. Balíček umožňuje zvolit si ze služeb Firebase a použít jen ty, které se vám hodí. Já konkrétně používám Firebase storage a Firebase auth. Firebase storage pro ukládání obrázků a Firebase auth pro zabezpečení přístupu k obrázkům, aby například jiný uživatel nevymazal jiný obrázek, než mu patří, také zamezuje útočníkům odchytnout url databáze a vymazat celý její obsah.

2.4.5 React-dnd

React-dnd (<https://github.com/react-dnd/react-dnd>, react-dnd, 2021) je Node balíček na implementaci techniky tah a pusť. Přemýšlel jsem osobně nad react-beautiful-dnd, bylo to sice snazší na použití, ale chybělo množství funkcí a balíček nebyl tak flexibilní. Hlavní výhodou React-dnd je flexibilita, ale podle mého názoru postrádá několik základních funkcí. Například při dragování se stránka automaticky nescrolluje a dokumentace mi nepřišla nepřehlednější, ale dostatečná.

2.4.6 React-dnd-multi-backend

React-dnd-multi-backend (<https://github.com/LouisBrunner/dnd-multi-backend/tree/master/packages/react-dnd-multi-backend>, 2021) spojuje React-dnd-html5-backend s React-dnd-touch-backend. Umožňuje použití aplikace na dotykové obrazovce nebo ovládání myši. Myš je podle mě preferovaný způsob na ovládání, protože takto jsem tuto aplikaci navrhl.

2.4.7 React-dnd-html5-backend

React-dnd-html5-backend (<https://github.com/react-dnd/react-dnd>, react-dnd, 2021) backend pro tah a pusť založený na html5 drag and drop API. Backend se propojí s React-dnd.

2.4.8 React-dnd-touch-backend

React-dnd-touch-backend (<https://github.com/react-dnd/react-dnd>, react-dnd, 2021) je backend pro tah a pusť na dotykových obrazovkách. Backend se propojí s React-dnd.

2.4.9 Immutability helper

Immutability-helper (<https://github.com/kolodny/immutability-helper>, 2021) je nástroj vytvořený pro práci s neměnnými daty, tedy konstantami. Hlavní využití této knihovny je práce s *React.js* hákem *useState()*, který ukládá ve stavu komponenty určitou proměnnou.

Tato knihovna se zaměřuje na složité datové struktury jako vnořované JSON objekty. Lze zde definovat vlastní funkce a používá vývojáři navržený syntax, který připomíná GraphQL.

2.4.10 Uuid

Uuid (<https://github.com/uuidjs/uuid>, 2021) je jeden z nejstahovanějších Node balíčků. Uuid znamená univerzální unikátní identifikátor. V mé aplikaci to využívám pro identifikaci obrázků ve Firebase storage. Zajišťuji tak integritu dat.

2.4.11 Struktura souborů

2.5 Klient

Nachází se ve složce *client/src*.

2.5.1 Axios

V této složce se nacházejí instance Axiosu, které můj projekt používá.

První je *cookieAxios.js*, tento soubor obsahuje instanci Axiosu, které posílá při každém požadavku na server soubory cookies spolu s požadavkem.

Druhá a poslední instance Axiosu je *refreshTokenAxios.js*, tato instance Axiosu je použita na všech požadavcích, jenž požadují autorizovaný přístup v aplikaci. Tato Instance využívá Axios Interceptor na umístění přístupového tokenu do autorizačního záhlaví. Jako první instance posílá při každém požadavku soubory cookies na server.

2.5.2 Components

V této složce se nacházejí React.js komponenty, které nejsou stránkami. Jsou rozděleny po složkách, složky mají název stránky, pokud nejsou obecně využívány na různých stránkách, například *reusableComponents.js* obsahuje univerzální komponenty. Dále se ve složkách dělí na další složky podle toho, k čemu se vztahují. Komponenty se většinou vztahují buď ke krokům receptu, vlastnímu receptu, ingrediencím anebo náčiní.

2.5.3 Config

Obsahuje konfiguraci Firebase na front-endu. Skládá se ze dvou souborů. První soubor *firebase.js* určuje, které služby Firebase používám a vrací instanci Firebase.

Druhý soubor *firebaseConfig* obsahuje konfiguraci Firbease ve formě JSON objektu. Jako výchozí vrací tento JSON objekt. Tato struktura projektu umožňuje snadno a rychle se připojit k jakémukoliv Firebase projektu bez velké námahy, stačí jen změnit JSON objekt.

2.5.4 Context

V této složce se nacházejí komponenty *React.js*, které implementují *React.js* Context API pro spravování globálního stavu aplikace. *Redux* je nejpopulárnější knihovna pro spravování stavu, ale Facebook nedávno vytvořil svoji. Context API funguje na principu háku *useContext()*, který bere jako parametr kontext, tak můžete dostat data z jiné komponenty do další. Data lze přečíst ve všech komponentech, jež jsou zabalené zprostředkovatelem dotyčného kontextu.

Konkrétní příklad *auth.js* má data přihlášeného uživatele, a zprostředkovatel je obalený okolo celé aplikace, aby data byla všude přístupná.

2.5.5 Pages

Zde jsou soubory obsahující *React.js* komponenty, které reprezentují jednotlivé stránky klienta. Soubory jsou nazvané podle jmen stránek. Stránky jsou: Ingredience a náčiní, přihlášení, registrace, hlavní stránka, formulář na recepty, stránka receptu, stránka sdíleného receptu a stránka sdílených receptů.

2.5.6 Queries

V této složce se nacházejí složky obsahující Axios požadavky klienta na server. Složky se nazývají podle serverového rozcestníku. Jednotlivé soubory jsou napsané v jazyku JavaScript a skládají se z asynchronní funkce, která vrací Axios požadavek. Každý soubor obsahuje jenom jednu funkci pro přehlednost projektu.

2.5.7 responsiveCss

ResponsiveCss je jediná složka mého projektu, která neobsahuje JavaScriptový soubor, místo toho obsahuje pouze jen css (Cascade Style Sheets) soubory. Soubory jsou nazvané po jednotlivých stránkách mé aplikace a css je v nich responzivní.

2.5.8 Utils

Utils přeloženo z angličtiny jsou nástroje. V této složce se nacházejí pomocné soubory, které tmelí klientskou aplikaci do jednoho celku.

Nachází se zde nejdůležitější soubor *accessToken.js*, tento soubor obsahuje *get()* a *set()* metodu, access token určuje, jestli uživatel má přístup k API pro přihlášené uživatele na serveru.

Také jsou tu dva soubory *authRoute.js* a *notAuthRoute.js*, které omezují přístup v klientské aplikaci na přihlášené a nepřihlášené uživatele. Nepřihlášený uživatel má přístup pouze ke stránce přihlášení a registrace.

2.6 Server

Nachází se ve složce server.

2.6.1 Apis

Ve složce *apis* jsou API méj webové aplikace. Jsou rozděleny do složek podle serverového rozcestníku. API jsem rozdělil na *comments.js* správa komentáře, *commentsQueries.js* požadavek klienta o poslání komentářů receptu, *ingredients.js* správa ingrediencí, *recipeIngredients.js* správa ingrediencí receptu, *recipeSteps.js* správa kroků receptu, *recipeUtensils.js* správa náčiní receptu, *recipeQueries.js* obsahuje metody, které vrací jednotlivé součásti receptu, *recipes.js* správa receptu, *sharedRecipeQueries.js* vrací sdílené recepty podle různých filtrů, *shareRecipes.js* správa sdílených receptů, *stepIngredients.js* správa ingrediencí kroku, *stepUtensils.js* správa náčiní kroku, *token.js* obnova access tokenu, *updateRecipe.js* úprava receptu, *users.js* správa uživatele a *utensils.js* správa vlastních nástrojů.

2.6.2 Configuration

Ve složce *configuration* se nachází jediný JavaScriptový soubor *db.js*. *Db.js* získá konfiguraci databáze ze souboru *.env* a vrací PostgreSQL pool. Výchozí nastavení je pro lokální použití, ale pomocí kapitoly konfigurace na Heroku lze nastavit na produkční nastavení.

2.6.3 Middleware

Složka *middleware* obsahuje moje vlastní *Express.js* middleware funkce. Všechny funkce až na jednu využívám pro omezení přístupu v aplikaci. *Validation.js* využívám pro validaci tvorby a přihlášení uživatele. *Authorization.js* kontroluje, zdali je uživatel přihlášený, pokud ano, tak uloží do request objektu uživatelské id. Další validací, zdali je uživatel vlastníkem daného objektu: *recipeOwner.js* kontroluje, zdali uživatel vlastní recept, *ingredientsOwner.js* kontroluje, zdali uživatel vlastní ingredienci, *utensilOwner.js* kontroluje, zdali uživatel vlastní náčiní a *commentOwner.js* kontroluje, zdali uživatel vlastní komentář. Poslední middleware *sharedRecipe.js* kontroluje, jestli je recept sdílený.

2.6.4 query_functions

Ve složce *query_functions* se nacházejí metody pro uložení receptu, abych dosáhl větší přehlednosti projektu, jinak by soubor s metodou na uložení receptu měl přes 1000 řádek.

2.6.5 Utils

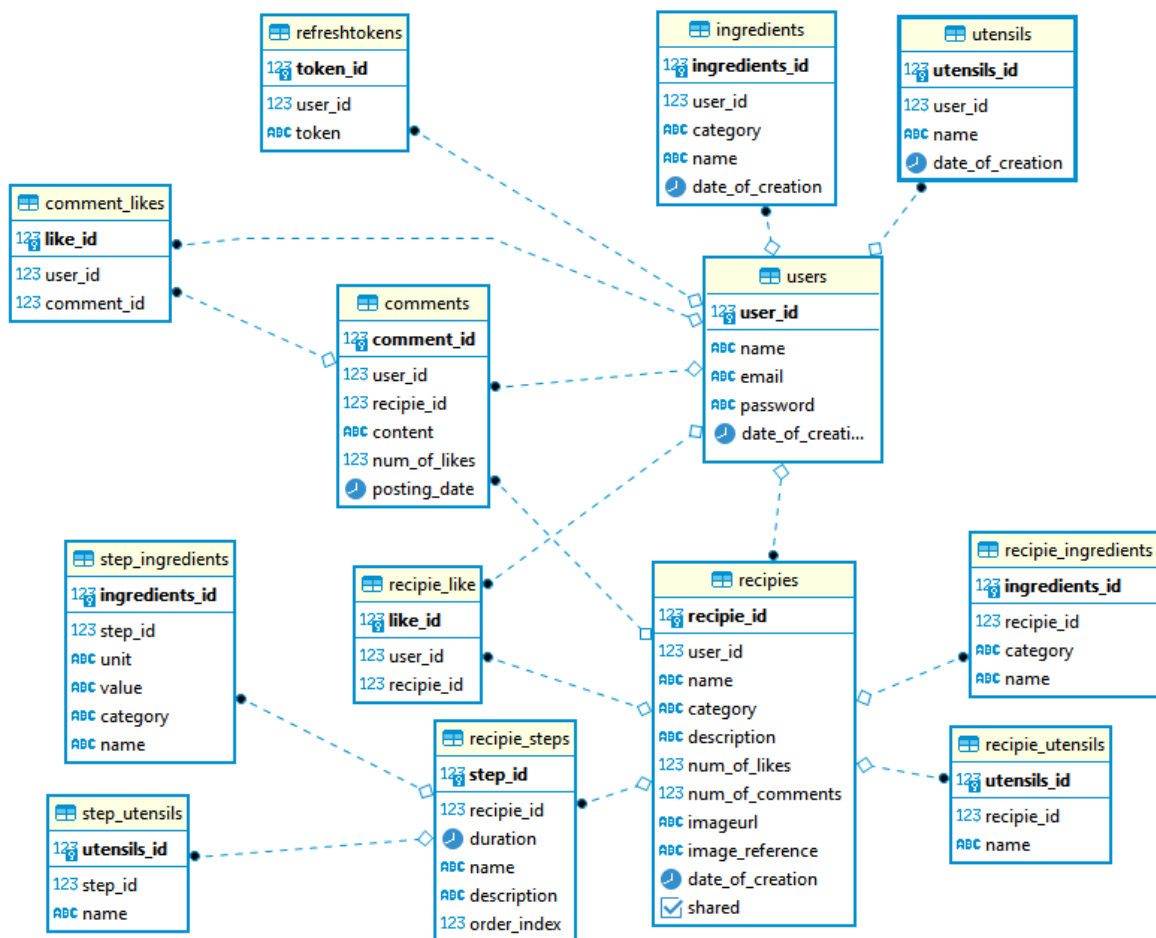
Složka *utils* obsahuje dva soubory. Jeden je JSON soubor, který umožňuje tvorbu Firebase JWT tokenů. Tento konfigurační soubor lze zaměnit za jiný soubor stejného formátu v případě změny Firebase projektu. Druhý soubor *jwtGenerator.js* má tři metody, které vracejí tři různé JWT tokeny Firebase JWT, refresh JWT a access JWT.

3. Struktura Backendu

3.1 Návrh databáze

Jako hlavní databázi jsem si vybral PostgreSQL, která je velmi jednoduchá na instalaci a použití lokálně a na Heroku. Dalším důvodem byla potřeba méj aplikace mít mnoho vztahů mezi daty a SQL databáze jsou na to ideální, posledním důvodem je to, že je opensource.

Moje databázové schéma:



Obrázek 1 ER model mojí databáze

Tabulky:

3.1.1 Users

Tabulka *users* se skládá z *user_id*, což je bigserial primary key neboli id. *Name* je posloupnost znaků určující název uživatele. *Email* je posloupnost znaků určující email, email musí být jedinečný v tabulce. *Heslo* je posloupností znaků, ukládá se v databázi zahashované. Při vytvoření uživatele se uloží datum *date_of_creation*, kdy byl záznam uživatele vytvořen.

3.1.2 Refreshtokens

Tabulka *refreshtokens* má tři parametry. První hodnota je jedinečný identifikátor v tabulce *token_id*. Druhá hodnota je *user_id*, které se váže na uživatele, jenž daný token vlastní. Poslední hodnotou je JWT *token* uložený jako posloupnost znaků. Refresh token zaručuje, že token, který uživatel používá je platný. Pokud se smaže uživatel vlastní token, tak dojde ke kaskádovému smazání tokenu.

3.1.3 Ingredients

Tabulka *ingredients* má pět parametrů. *Ingredients_id* je jedinečný identifikátor v tabulce. Druhá hodnota *user_id* je uživatelův identifikátor. Třetí hodnota *category* je kategorie ingredience, je uložena ve formě posloupnosti znaků. Čtvrtá hodnota *name* je název ingredience, jenž je uložena jako posloupnost znaků. Poslední parametr *date_of_creation* je

datum vytvoření ingredience. Tato tabulka podporuje kaskádové mazání, pokud je objekt s klíčem uživatele smazán, tak je ingredience také smazaná.

3.1.4 utensils

Tabulka *utensils*, tedy nástrojů, má čtyři parametry. *Utensils_id* je jedinečný identifikátor nástroje v tabulce. Další hodnota je *user_id*, které odkazuje na vlastníka. Třetí hodnota *name* je název nástroje, uložený ve formě posloupnosti znaků. Poslední parametr *date_of_creation* je datum vytvoření náčiní. Tato tabulka podporuje kaskádové mazání.

3.1.5 Recipes

Tabulka *recipes*, tj. receptů, má jedenáct parametrů. První parametr *recipe_id* je identifikátor receptu. *User_id* odkazuje na tvůrce receptu. Další parametr *name* je název, název je uložený ve formě posloupnosti znaků. Další parametr *category* je kategorie, je tak jako jméno, uložená v podobě posloupnosti znaků. Popis *description* je uložený ve formě textu, text má neomezený počet znaků. *Num_of_likes* zaznamenává počet oblíbení receptu a začíná na čísle nula. *Num_of_comments* zaznamenává počet komentářů a začíná na čísle nula. *ImageUrl* ukládá url obrázku, který je uložený ve Firebase storage, url je text. *Image_reference* uloží cestu k obrázku ve Firebase uložisti, reference je uložena ve formě textu. Předposlední parametr *date_of_creation* je čas a datum vytvoření receptu, využívá se při seřazení. Poslední parametr receptu *shared* je boolean jestli je recept sdílený. Tabulka receptů podporuje kaskádové mazání.

3.1.6 recipe_ingredients

Tabulka *recipe_ingredients*, neboli ingrediencí receptů, má čtyři parametry. *Ingredients_id* je jedinečný identifikátor v tabulce. *Recipe_id* odkazuje na recept konkrétní ingredience. Třetí parametr *category* je kategorie receptu, která je uložena ve formě posloupnosti znaků. Název *name* je stejného typu jako kategorie. Tabulka podporuje kaskádové mazání.

3.1.7 recipe_utensils

Tabulka *recipe_utensils*, tedy nástrojů, má tři parametry. *Utensils_id* je jedinečný identifikátor nástroje v tabulce. Další hodnota je *recipe_id*, které odkazuje na recept. Třetí hodnota *name* je název nástroje uložený ve formě posloupnosti znaků. Tato tabulka podporuje kaskádové mazání.

3.1.8 recipe_steps

Tabulka *recipe_steps*, neboli kroků receptu, má šest parametrů. První parametr *step_id* je jedinečným identifikátorem tabulky. *Recipe_id* odkazuje na recept, kterému krok patří. *Duration* je odhadovaná doba trvání kroku, typ parametru je čas. *Name* je název kroku, jenž je uložený jako posloupnost znaků. Předposlední parametr *description* je popis uložený jako text v databázi. Poslední parametr je *order_index*, který určuje pořadí kroku.

3.1.9 step_utensils

Tabulka *step_utensils*, tj. nástrojů kroku, má tři parametry. *Utensils_id* je jedinečný identifikátor nástroje v tabulce. Další hodnota je *step_id*, které odkazuje na krok. Třetí hodnota *name* je název nástroje uložený ve formě posloupnosti znaků. Tato tabulka podporuje kaskádové mazání.

3.1.10 step_ingredients

Tabulka *step_ingredients*, neboli ingrediencí kroku, má šest parametrů. *Ingredients_id* je jedinečný identifikátor v tabulce. *Step_id* odkazuje na krok konkrétní ingredience. Třetí parametr *category* je kategorie receptu, která je uložena ve formě posloupnosti znaků. Název *name* je stejného typu jako kategorie. *Unit* je pole určené pro jednotku míry dané ingredience, která je uložena jako posloupnost znaků. *Value* je hodnota, která navazuje na jednotku a ukazuje hodnotu v dané jednotce, je uložena jako text. Tabulka podporuje kaskádové mazání při smazání kroku.

3.1.11 recipe_like

Tabulka *recipe_like*, tedy oblíbení receptu, má tři parametry. *Like_id* je jedinečný identifikátor řádku v tabulce. *User_id* odkazuje na uživatele, kterému komentář patří. *Recipe_id* odkazuje na recept, jež byl ohodnocen jako oblíbený. Kaskádové mazání oblíbení receptu se aktivuje při smazání uživatele, kterému toto oblíbení patří, nebo při smazání receptu, na který toto oblíbení odkazuje.

3.1.12 comments

Tabulka *comments*, tj. komentářů, má šest parametrů. První parametr je *comment_id*, což je jedinečný identifikátor řádku v tabulce komentářů. *User_id* odkazuje na uživatele, jenž napsal tento komentář. Třetí parametr je *recipe_id* odkazující na id receptu v tabulce receptů. Čtvrtý parametr je *content*, česky obsah komentáře, typ obsahu je posloupnost znaků. *Num_of_likes* je počet oblíbení komentáře. *Posting_date* je poslední parametr komentáře a určuje datum a čas vytvoření komentáře receptu. Kaskádové mazání je nastavené u sloupců *recipe_id* a *user_id*.

3.1.13 comments_like

Tabulka *comments_like*, neboli oblíbení receptu, má tři parametry. *Like_id* je jedinečný identifikátor řádku v tabulce. *User_id* odkazuje na uživatele, kterému komentář patří. *Comment_id* odkazuje na komentář, jenž byl ohodnocen jako oblíbený. Kaskádové mazání oblíbení se aktivuje při smazání uživatele, kterému oblíbení patří, nebo při smazání komentáře, na který oblíbení odkazuje.

3.2 Autentizace

Vytvořil jsem si vlastní autentizaci podle návodu, jak zacházet s JWT žetony.

Dva důležité termíny:

- *Refresh token* – žeton s dlouhou trvanlivostí, uložený jako http only cookies. Slouží pro žádost o *access token*.
 - *Access token* – žeton s krátkou trvanlivostí uložený v paměti aplikace a je obnoven, pokud uživatel má *refresh token*.
1. Poté, co se uživatel přihlásí, nebo zaregistruje, dostane *refresh token* jako cookies, pro větší bezpečnost je uložený v databázi, a *access token* v odpovědi od serveru.
 2. *Access token* se uloží do paměti a bude přidán do headeru každého requestu, který vyžaduje autorizovaný přístup.
 3. Pokud *access token* vyprší a uživatel má *refresh token*, tak uživatel dostane nový *access token*.

Tento způsob umožňuje takzvaný „*silent login*“, jestli uživatel má *refresh token*, tak jej stránka automaticky přihlásí.

3.3 Struktura API

Pro vytváření vlastních API používám serverový framework *Express.js*. Každá API je určená pro jiná data, jedna pro jiné data v databázi. Například recepty nebo uživatelé. Tyto API jsou v zásobníku a pokaždé, když uživatel pošle požadavek serveru, tak najde vhodnou cestu na rozcestníku, cesta má další pod rozcestník konkrétní API, zde se najde metoda, která odpoví klientovi na požadavek.

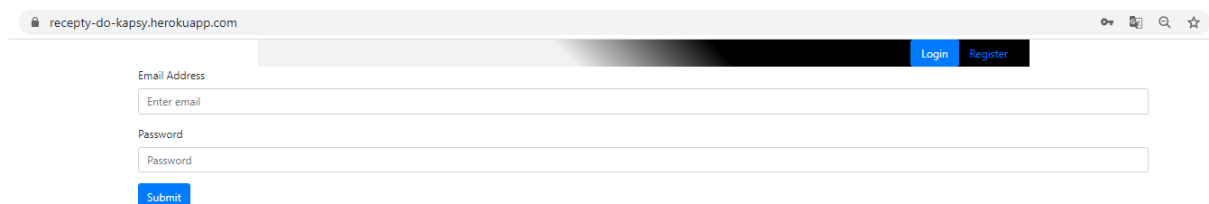
3.4 Uložiště obrázků

Pro ukládání obrázků používám Firebase cloud storage, je velice jednoduché jej připojit k aplikaci a použít. Toto řešení má výhodu v tom, že se výrazně zrychlí výkonnost databáze, protože není zatěžována obrázky. Také jsem změnil pravidla pro uživatele tak, že jenom konkrétní uživatel se může dostat do své složky. Přístup jsem omezil Firebase autentifikací, která je součástí služeb nabízených Firebase. Firebase auth je napojena na moji vlastní autentifikaci JWT tokenem.

4. Struktura Frontendu

4.1 Stránka Přihlášení

Stránka přihlášení je přihlašovací formulář, který po vyplnění zjistí, zda jsou jím zadaná data správná a pokud jsou správná, tak uživatele přesměruje na hlavní stránku. Email a heslo jsou povinná pole. Funkcí této stránky je jenom přihlášení uživatele do aplikace.



Obrázek 2 Stránka přihlášení

4.2 Stránka Registrace

Na stránce pro registraci si uživatel vybere e-mail, který je jedinečný v databázi, uživatelské jméno, které se může opakovat, heslo a potvrdí heslo. Pokud data projdou validací, tak se založí jeho účet. Funkcí této stránky je jenom tvorba nového uživatele.

Obrázek 3 Stránka registrace

4.3 Hlavní stránka

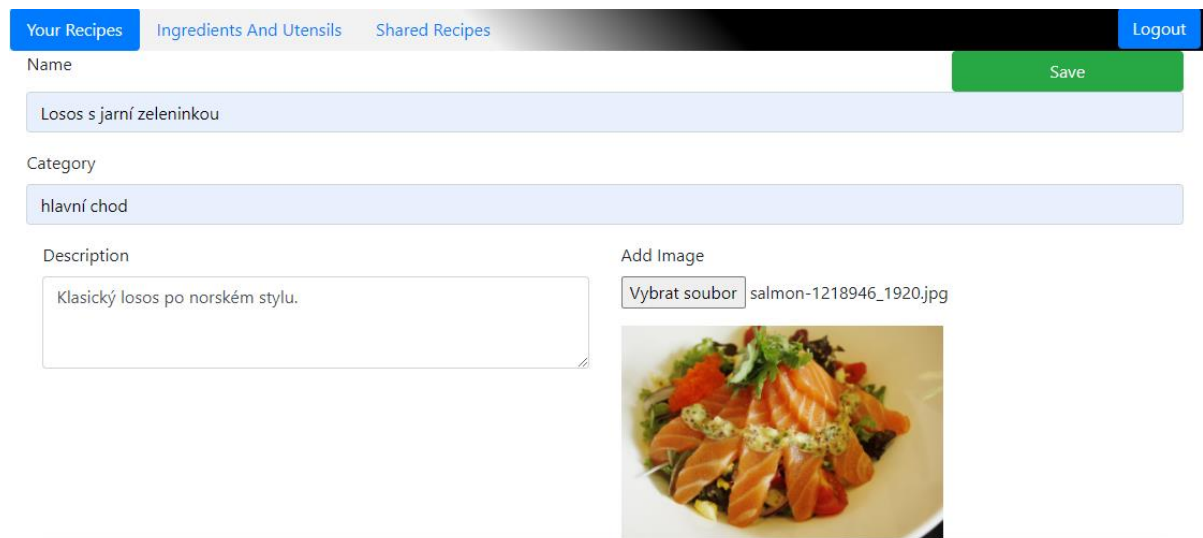
Na hlavní stránce se zobrazuje seznam karty uživatelových receptů. Karta se skládá z názvu receptu, kategorie receptu, fotky receptu a tlačítka, které vám zobrazí rozbalovací nabídku s možností navštívit stránku vlastního receptu, nebo pokud je recept sdílený, tak možnost navštívit stránku sdíleného receptu. Nad seznamem karet vlastních receptů se nachází tlačítko, jenž vás přenesse na stránku formulář receptu, která umožňuje tvorbu nových receptů.

Obrázek 4 Hlavní stránka

4.4 Formulář receptu

Formulář receptu se skládá ze dvou částí. V první části formuláře, pokud uživatel chce, tak si může přetáhnout ingredience nebo nástroje do receptu technikou táhni a pusť. Druhá část se dělí na základní data receptu, ingredience a nástroje receptu, formulář kroku a následující kroky: První krok: v základních datech receptu si uživatel zvolí název receptu, obrázek, kategorii a popis. Druhý krok: v ingrediencích a nástrojích receptu uživatel vidí ingredience a nástroje receptu. Třetí krok: formulář kroku slouží k vytváření kroků receptu. Krok se skládá

z ingrediencí a nástrojů, které si přetáhnete z ingrediencí a nástrojů receptu, jména, času trvání a popisu. Čtvrtý krok: kroky si můžete seřadit, upravit a proházet ingredience mezi nimi.



The screenshot shows a web interface for managing recipes. At the top, there are three tabs: 'Your Recipes' (active), 'Ingredients And Utensils', and 'Shared Recipes'. A 'Logout' button is located on the far right. The form contains several fields: 'Name' with the value 'Losos s jarní zeleninkou', 'Category' with 'hlavní chod', and 'Description' with 'Klasický losos po norském stylu.'. To the right of the description field is an 'Add Image' section featuring a 'Vybrat soubor' button and the filename 'salmon-1218946_1920.jpg'. Below this, a small image of a salmon salad is displayed. A green 'Save' button is positioned at the top right of the form.

Obrázek 5 Formulář základních dat receptu

4.5 Stránka vašeho receptu


Tato stránka slouží k úpravě receptu a přečtení vlastních receptů. Skládá se z: základních dat receptu, zde vidíte název receptu, popis receptu, kategorii receptu, obrázek receptu a tlačítko s možnostmi: úprava, smazání nebo sdílení. Při úpravě dostanete možnost tato data upravit ve formuláři. Ingredience a Nástroje receptu, kde po kliknutí na tlačítko se vám zobrazí nástroje a ingredience receptu. Nástroje můžete přesunout do formuláře kroků, nebo do nástrojů již vytvořeného kroku. Ingredience lze přesunout do ingrediencí formuláře kroku, nebo do ingrediencí kroku. Formulář kroků receptu umožňuje uživateli v jeho již vytvořeném receptu přidat další kroky. Krok se skládá z popisu, názvu, času, ingrediencí a nástrojů. V seznamu kroků lze seřadit kroky dle libosti, upravit, a dokonce i smazat.

[Your Recipes](#)
[Ingredients And Utensils](#)
[Shared Recipes](#)
[Logout](#)

Sumčí polévka

předkrm

Pálivá sumčí polévka podle indiánského stylu



Ingredients and Utensils

⊕Add Step

Zpracování sumce

00:10:00⌚

Vyfiletujeme sumce nožem na kostky okol 3cm na 3cm a omyjeme.

Výroba rajčatového protlaku

00:05:00⌚

Rozkrájíme papriku a čili papriku na prach. Dáme do mixeru na 3minuty.

Vaření

00:30:00⌚

dáme papriku se sumcem do vody mícháme a vaříme 30 minut na maximální teplotě.

Zakončení

00:01:00⌚

Nalijeme polévku do misky a přisypeme majoránku. Dobrou chuť.

Obrázek 6 Stránka vlastního receptu

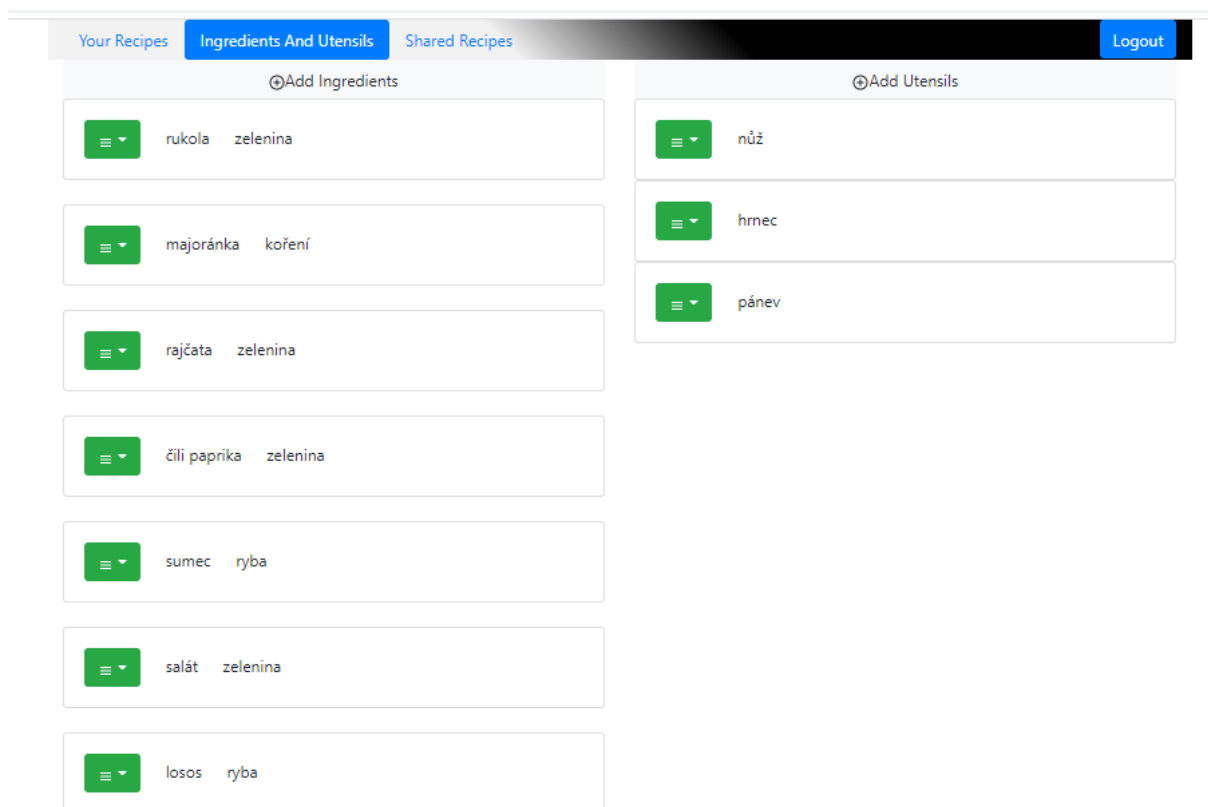
4.6 Stránka Ingredience a Nástroje

Stránka ingrediencí a receptů se skládá ze dvou seznamů. První seznam je seznam ingrediencí skládající se z karet ingredience. Karta ingredience zobrazí název ingredience, kategorii a tlačítko aktivující rozbalovací nabídku, umožňující správu jednotlivých ingrediencí. Ingredience lze smazat nebo upravit. Seznam používá stránkování po deseti ingrediencích.

Nad seznamem ingrediencí se nachází tlačítko, které otevře modální okno s formulářem na tvorbu nových ingrediencí. Povinná pole jsou název a kategorie.

Další důležitou součástí je seznam nástrojů. Seznam nástrojů funguje na podobném principu jako seznam ingrediencí, ale místo názvu a kategorie zobrazuje jenom název.

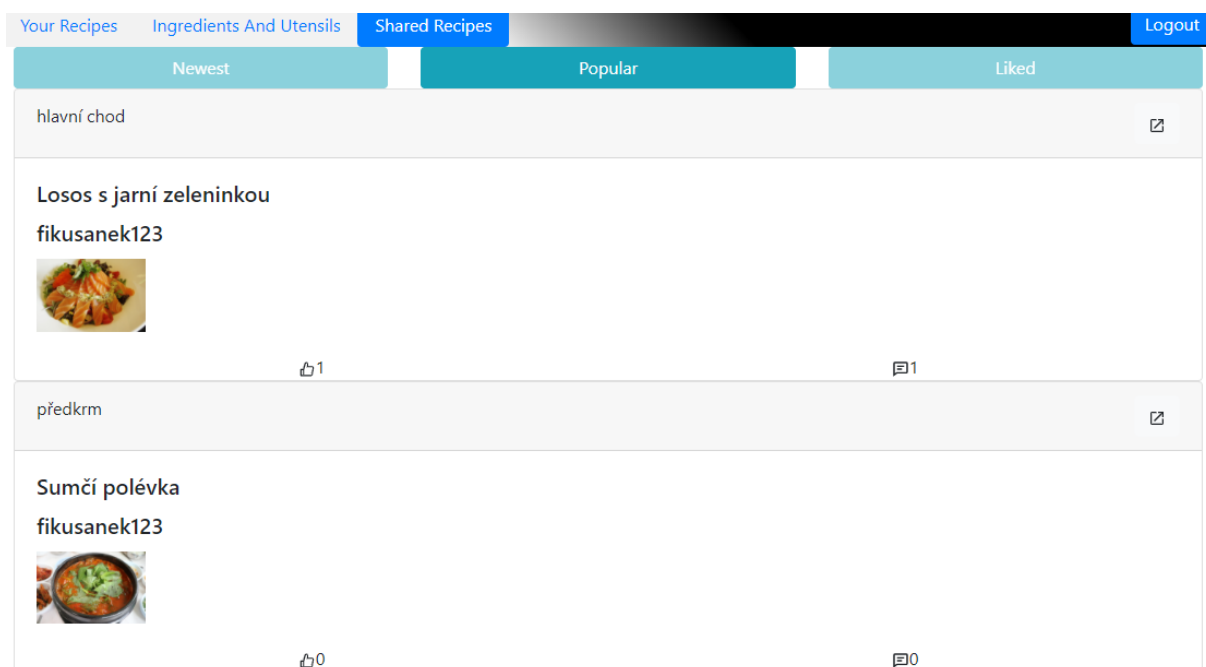
Také nad seznamem nástrojů se nachází tlačítko, které otevře modální okno s formulářem na tvorbu nových nástrojů. Povinné pole je název.



Obrázek 7 Stránka ingredience a nástroje

4.7 Stránka sdílených receptů

Zde uživatel najde recepty ostatních uživatelů. Uživatel si zde vybere kritérium, podle kterého chce vybrat recepty. Dostupná kritéria jsou nejoblíbenější recepty, nejnovější recepty a vaše oblíbené recepty, ale díky skvělému návrhu aplikace je velmi jednoduché další kritéria přidat. V dalším vývoji aplikace rozhodně další přidám a umožním i kombinované hledání. Recepty se zobrazují v podobě karet, karta má tlačítko na přesměrování, počet komentářů, počet oblíbení, popis, název, kategorii a fotku.



Obrázek 8 Stránka sdílených receptů

4.8 Stránka sdíleného receptu

Tato stránka slouží pro přečtení receptu a ohodnocení receptu. Stránka má několik základních částí. První částí jsou základní informace receptu, které se skládají z názvu receptu, kategorie, popisu a obrázku. Dále pod nimi se nacházejí seznamy karet ingrediencí a nástrojů. Pod seznamy ingrediencí a nástrojů se nachází seznam kroků, u kroku si můžete prohlédnout jeho složky.


Na konci stránky se nachází sekce komentářů receptu, kterou lze otevřít tlačítkem. Sekce komentářů se skládá z formuláře na komentáře, který požaduje jen obsah komentáře pro úspěšné vyplnění.

Pod formulářem se nachází sekce, kde vidíte komentáře ostatních uživatelů aplikace. Karta komentáře se skládá z obsahu, počtu oblíbení a pokud jste vlastníkem, tak se nachází na kartě i tlačítko otevírající rozbalovací menu, jenž umožňuje správu komentáře.

Your Recipes
Ingredients And Utensils
Shared Recipes
Logout

Sumčí polévka
fikusanek123
předkrm

2
2

Pálivá sumčí polévka podle indiánského stylu


Ingredients and Utensils

Zpracování sumce

☰

00:10:00 ⌚
Vyfiletujeme sumce nožem na kostky okol 3cm na 3cm a omyjeme.

Výroba rajčatového protlaku

☰

00:05:00 ⌚
Rozkrájíme papriku a čili papriku na prach. Dáme do mixeru na 3minuty.

Vaření

☰

00:30:00 ⌚
dáme papriku se sumcem do vody mícháme a vaříme 30 minut na maximální teplotě.

Zakončení

☰

00:01:00 ⌚

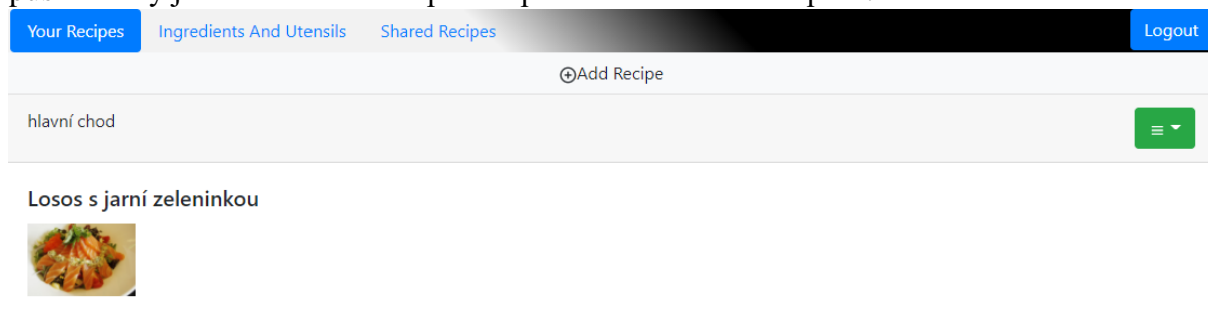
Obrázek 9 Stránka sdíleného receptu

5. Funkce

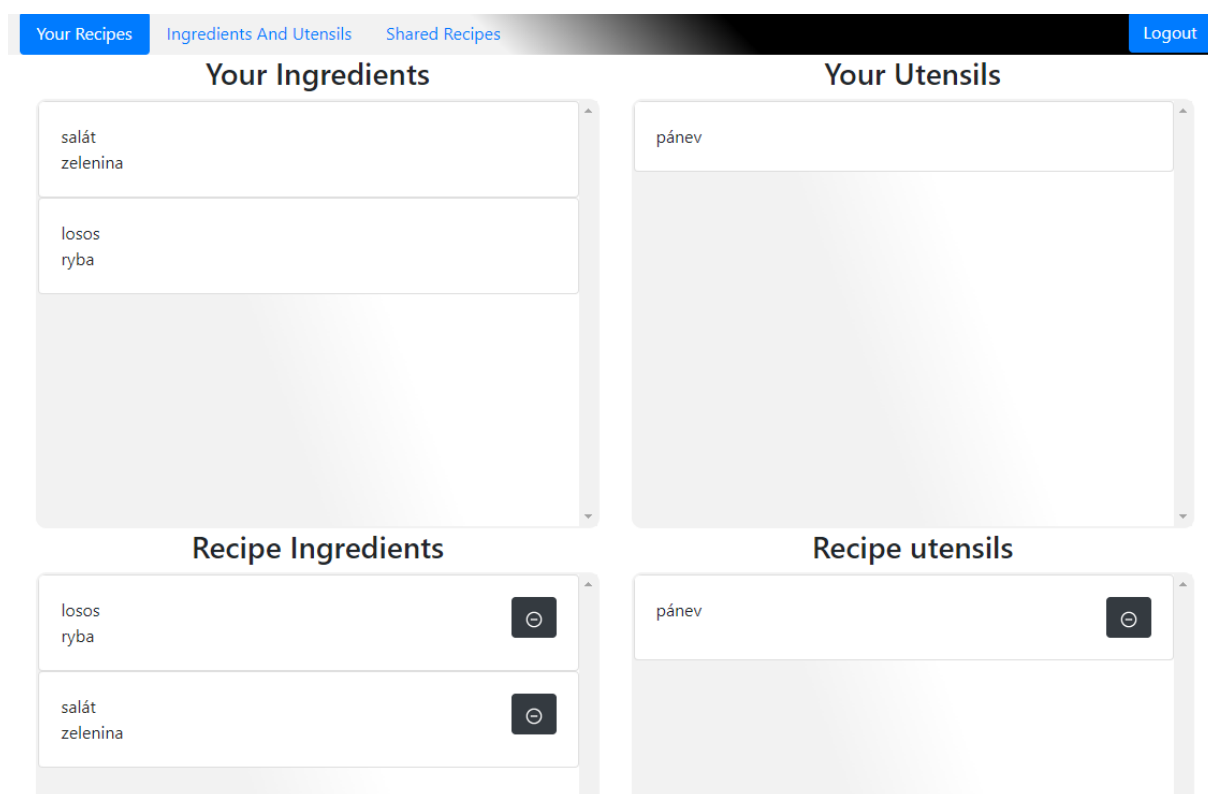
5.1 Tvorba receptu

Nejdříve musí uživatel otevřít hlavní stránku, tam klikne na tlačítko přidat recept, které ho přenesse na stránku formuláře receptu. Následně se uživatel dostane na stránku formuláře, kde si může přetáhnout všechny ingredience a nástroje, které potřebuje. Pro přetáhnutí ingrediencí a nástrojů se používá technika tah a pusť. Poté, co bude spokojený s výběrem, zmáčkne tlačítko další krok. Ve druhém kroku vyplní základní data receptu: název, popis, kategorii a obrázek. Jenom název a kategorie jsou povinné. Dále může uživatel vytvořit kroky receptu. Krok potřebuje název, čas a kategorii, nepovinné je přetáhnout z ingrediencí receptu ingredience a nástroje z nástrojů receptu. Ingredience a nástroje lze přetáhnout technikou tah a

pust'. Kroky je možné seřadit a zpětně upravit technikou tah a pust'.



Obrázek 10 Hlavní stránka



Obrázek 11 Přesouvání ingrediencí a nástrojů

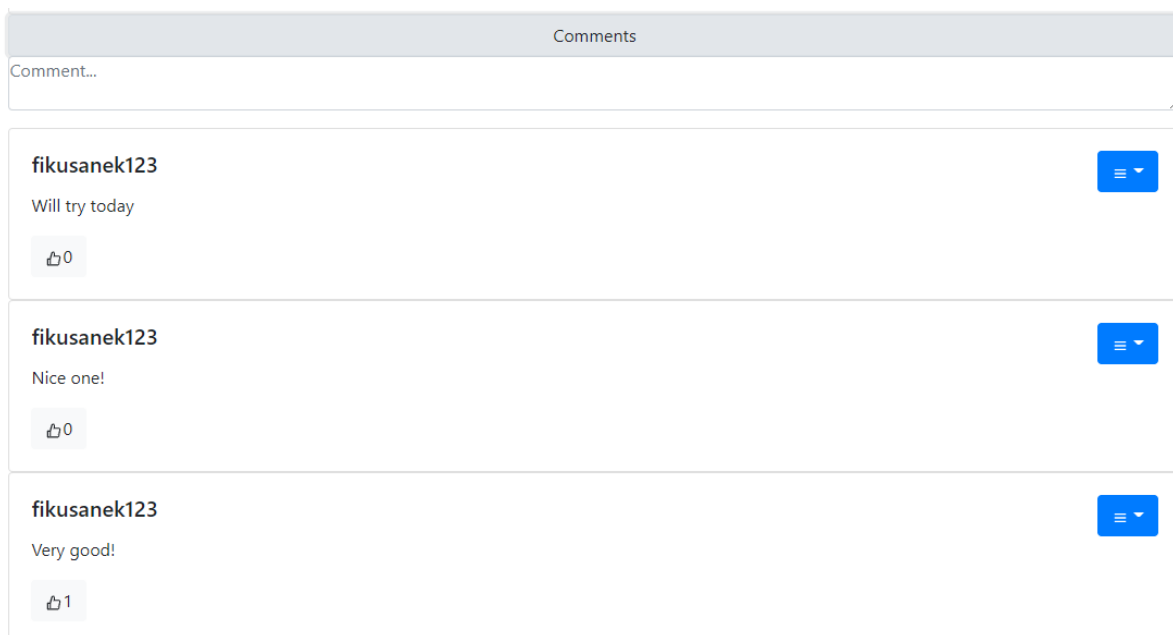
Obrázek 12 Formulář kroku receptu

5.2 Sdílení receptu

Sdílení receptu funguje tak, že si uživatel otevře stránku konkrétního receptu, který chce sdílet. V pravém horním rohu je rozbalovací menu. Uživatel si jej otevře a vybere si možnosti sdílení. Potom se pošle dotaz na server a pokud již recept není sdílený, tak se v tabulce receptů nastaví sloupec *shared* na *true*, *shared* proměnná je typu boolean. Poté si uživatel může zobrazit sdílený recept ve formě stránky sdíleného receptu. Recept teď může být zobrazen ostatními uživateli aplikace a také může být ohodnocen ostatními uživateli.

Na stránce sdílených receptů jsou tři tlačítka. První tlačítko s nápisem *Newest* nechá zobrazit nejnovější recepty. Tlačítko *Popular* nechá zobrazit nejoblíbenější recepty. Tlačítko *Liked* zobrazí vaše oblíbené recepty. Všechny tyto funkce podporují stránkování po deseti.

Hodnocení probíhá také na stránce sdílených receptů a jsou dva způsoby na výběr, jak ohodnotit recept. První způsob je, přidat si jej do oblíbených tlačítkem s ikonou palec nahoru. Druhý způsob je napsat komentář, který lze napsat po otevření sekce s komentáři, přičemž formulář je na vrcholu sekce. Komentář lze taky ohodnotit oblíbením.



Obrázek 13 Komentáře receptu

6. Návod na spuštění

6.1 Aplikace na Heroku

Nejsnadnějším způsobem je spuštění aplikace na Heroku serveru. URL pro spuštění aplikace je uložena v souboru Readme. Není třeba nic instalovat. Aplikace se spustí v prohlížeči např. Chrome.

6.2 Konfigurace Firebase

Pokud chcete mít vlastní Firebase projekt následujte tyto kroky, pokud ne tak pokračuje v návodu na lokální konfiguraci.

1. Vytvořit firebase projekt <https://firebase.google.com/>
2. Povolit autentikaci pro email/password ve vašem firebase projektu
3. Přidat webovou aplikaci do firebase projektu. To vygeneruje konfiguraci s vzhledem:
 - a.

```
var firebaseConfig = {  
  apiKey: "apiKey",  
  authDomain: "authDomain",  
  projectId: "projectId",  
  storageBucket: "storageBucket",  
  messagingSenderId: "messagingSenderId",  
  appId: "appId",  
  measurementId: "measurementId"  
};
```
4. Tuto konfiguraci je třeba přidat do souboru, který je třeba vytvořit v client/src/config/ s názvem firebaseConfig.js. Obsah souboru bude vypadat:

```
a. export default {
  apiKey: "apiKey",
  authDomain: "authDomain",
  projectId: "projectId",
  storageBucket: "storageBucket",
  messagingSenderId: "messagingSenderId",
  appId: "appId",
  measurementId: "measurementId"
};
```

5. Vytvořit ve Firebase projektu storage

6. Změňte pravidla Firebase storage v záložce rules na:

```
rules_version = '2';

service firebase.storage {

  match /b/{bucket}/o {

    match /{userId}/{allPaths=**} {

      allow read, write: if request.auth.uid == userId;

    }

  }

}
```

7. Nechat si vygenerovat konfiguraci serviceAccountu ve Firebase a uložit ho do /server/utills. Je třeba též změnit název cesty serviceAccountu v server/utills/jwtGenerator.js.

- Settings=>project settings=>service accounts=>generate new private key To vám stáhne na počítač klíč ve formátu JSON
- Přesuňte tento soubor do server/utills/
- V souboru server/utills/jwtGenerator.js změňte v require('./{název souboru s klíčem}')

```
i. const serviceAccount = require('./název souboru klíče)
```

6.3 Lokální spuštění

Další možností je lokální instalace.

- Stažení Node.js ze stránky <https://nodejs.org/en/download/>
- Spustit příkazový řádek ve složce, do které projekt budete instalovat
- Stáhnout a nainstalovat **git** ze stránky <https://git-scm.com/downloads>
- Použít příkaz **git clone {url repozitáře}**
- Použít příkaz **npm i** ve složce *client*
- Použít příkaz **npm i** ve složce *server*
- Stáhnout postgresql ze stránky <https://www.postgresql.org/download/>
- Do PostgreSQL shellu psql se můžete dostat dvěma způsoby:
 1. způsob:
 - Přidat cestu k adresáři PostgreSQL *bin* do proměnné prostředí *PATH*
 - Použít kdekoliv příkaz **psql**
 - Po použití příkazu **psql** se přihlásíte do PostgreSQL

- b. 2. způsob:
 - i. Najdete si nainstalovaný SQL shell (psql) na počítači.
 - ii. Tento soubor otevřete a přihlásíte se
- 9. Použít příkaz **CREATE DATABASE {jméno databáze};**
- 10. Použít příkaz **\c {jméno databáze}**
- 11. Zkopírovat obsah souboru **server/database.sql** do příkazového řádku a spustit
- 12. V server/ vytvořit soubor **.env**. ve kterém zadáte uživatelské jméno, které používáte v postgresql, vaše heslo, localhost, port na kterém běží databáze, název vaší databáze a libovolné klíče, které si vymyslíte. Vzor souboru **.env**:

DB_USER=uživatelské jméno vlastníka databáze v postgresQL
DB_PASSWORD=vaše heslo
DB_HOST=localhost
DB_PORT=5432
DB=název databáze
SECRET1=klíč1
SECRET2=klíč2
- 13. Server spustit ve složce *server* **npm run dev**, nebo **node index**
- 14. Klienta spustit ve složce *client* **npm start**

6.4 Instalace na Heroku

Pro instalaci aplikací na Heroku jsou k dispozici výuková videa, ve stručnosti uvádím přehled hlavních kroků:

1. Připravit konfiguraci pro spuštění lokálně (viz. kapitola 10.3)
Při instalaci je třeba zvolit PostgreSQL port 5432
2. Přidat cestu k adresáři PostgreSQL bin do proměnné prostředí *PATH*
3. Změnit soubor server/configuration/db.js takto:

```

const Pool = require('pg').Pool;
require('dotenv').config()

const devConfig = ({
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  database: process.env.DB,
  max: 20,
  idleTimeoutMillis: 30000,
  connectionTimeoutMillis: 2000,
});

const proConfig = ({
  connectionString: process.env.DATABASE_URL,
  ssl: {
    rejectUnauthorized: false
  }
});

const pool = new Pool(process.env.NODE_ENV === 'production' ? proConfig : devConfig);

module.exports = pool;

```

Obrázek 14db.js

4. Použít příkaz v adresáři *server/* **npm i heroku-ssl-redirect**
5. Přesunout obsah složky *server* do hlavní složky a vymazat složku *server*
Za žádnou cenu neaktualizujte importy
6. Nainstalovat Heroku podle instrukcí <https://devcenter.heroku.com/articles/heroku-cli>
7. Použít následující příkazy v hlavní složce:
 heroku login
 heroku create
 heroku addons:create heroku-postgresql:hobby-dev -a "name-app"
 heroku pg:psql -a "name-app"
 (poznámka: je třeba mít PostgreSQL na portu 5432)
8. Zkopírovat obsah souboru **database.sql** do příkazové řádky a spustit
9. Změnit soubor *index.js*, url vaší Heroku aplikace má být umístěno v konfiguraci *corsu*. Jinak váš soubor *index.js* má vypadat nachlup stejně:

```

const express = require('express');
const app = express();
const cors = require('cors');
const server = require('http').createServer(app);
const cookieParser = require('cookie-parser');
const path = require('path');
const sslRedirect = require('heroku-ssl-redirect').default;
//middleware
app.use(cors({
  origin: 'https://recepty-do-kapsy.herokuapp.com/',
  credentials: true
}));
app.use(cookieParser());
app.use(express.json());
//ssl force on heroku
app.use(sslRedirect());
app.use('/users', require('./apis/users'));
app.use('/token', require('./apis/token'));
app.use('/utensils', require('./apis/utensils'));
app.use('/ingredients', require('./apis/ingredients'));
app.use('/recipies', require('./apis/recipies'));
app.use('/recipeUpdate', require('./apis/updateRecipe'));
app.use('/recipieQuery', require('./apis/recipieQueries'));
app.use('/comments', require('./apis/comments'));
app.use('/shared_recipies', require('./apis/shareRecipies'));
app.use('/shared_recipie_query', require('./apis/sharedRecipieQueries'));
app.use('/comments_queries', require('./apis/commentsQueries'));
app.use('/recipe_ingredients', require('./apis/recipeIngredinetns'));
app.use('/recipe_utensils', require('./apis/recipeUtensils'));
app.use('/recipe_steps', require('./apis/recipeSteps'));
app.use('/step_ingredients', require('./apis/stepIngredients'));
app.use('/step_utensils', require('./apis/stepUtensils'));

if (process.env.NODE_ENV === "production") {
  app.use(express.static(path.join(__dirname, "client/build")));
}

app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, "client/build/index.html"));
});

```

10. `const PORT = process.env.PORT || 5000;`

11. Obrázek 15 `index.js`

12. Změnit proxy v souboru `package.json` ve složce `client` na url vaší aplikace

```

"eslintConfig": {
  "extends": "react-app"
},
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
},
"proxy": "https://vast-sierra-08986.herokuapp.com/"
}

```

Obrázek 16 `package.json client`

13. Vymazat složku `node_modules` a soubor `package-lock.json` ve složce `client`
14. Spustit příkaz **npm i** ve složce `client`
15. Spustit příkaz **npm run build** ve složce `client`
16. V souboru `package.json` v hlavní složce vymazat skript „dev“: „nodemon index“ a přidat dva skripty `"start": "node index"`, `"heroku-postbuild": "cd client && npm i && npm run build"`.

```

{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "engines": {
    "node": "v12.18.2",
    "npm": "6.14.5"
  },
  "scripts": {
    "start": "node index",
    "heroku-postbuild": "cd client && npm i && npm run build"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.0",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "firebase-admin": "^9.3.0",
    "heroku-ssl-redirect": "^0.1.1",
    "jsonwebtoken": "^8.5.1",
    "pg": "^8.5.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.4"
  }
}

```

Obrázek 17 package.json

17. Na webovém rozhraní Heroku vaší aplikace na záložce *Settings* v sekci *Config vars* přidat dvě nové proměnné SECRET1, SECRET2, které budou obsahovat libovolné řetězce znaků bez mezer a bez znaku =
18. Spustit tyto příkazy v hlavní složce:


```

git init
heroku git:remote -a "app-name"
git add .
git commit -m "heroku"
git push heroku master

```

7. Závěr

Závěrem si myslím, že jsem zadání splnil na 100 % a projekt byl rozhodně úspěšný v tom, co se snažil uskutečnit. Podařilo se mi vytvořit prostředí pro uživatele, kde mají velkou kreativní svobodu na vytváření hezky strukturovaných receptů. Codebase umožňuje jednoduché

přidávání dalších vlastností. Také by se dal vytvořit podobný projekt pro workflow management. Koncept této aplikace lze rozhodně využít ve všech různých odvětvích.

8. Seznam Obrázků

Obrázek 1 ER model mojí databáze.....	8
Obrázek 2 Stránka přihlášení.....	11
Obrázek 3 Stránka registrace.....	12
Obrázek 4 Hlavní stránka	12
Obrázek 5 Formulář základních dat receptu.....	13
Obrázek 6 Stránka vlastního receptu	14
Obrázek 7 Stránka ingredience a nástroje	15
Obrázek 8 Stránka sdílených receptů.....	16
Obrázek 9 Stránka sdíleného receptu.....	17
Obrázek 10 Hlavní stránka	18
Obrázek 11 Přesouvání ingrediencí a nástrojů.....	18
Obrázek 12 Formulář kroku receptu	19
Obrázek 13 Komentáře receptu.....	20
Obrázek 14 db.js.....	23
Obrázek 15 package.json client.....	25
Obrázek 16 index.js.....	24
Obrázek 17 package.json.....	26

9. Bibliografie

- Dahl, R. (7. 3 2021). *nodejs.org*. Načteno z nodejs.org: <https://nodejs.org/en/download/>
- Google. (7. 3 2021). *Firebase*. Načteno z Firebase: <https://www.npmjs.com/package/firebase>
- Holowaychuk, T. (7. 3 2021). *Expressjs.com*. Načteno z Expressjs.com:
<https://www.npmjs.com/package/express>
- <https://github.com/auth0/node-jsonwebtoken>. (7. 3 2021). *node-jsonwebtoken*. Načteno z
node-jsonwebtoken: <https://www.npmjs.com/package/jsonwebtoken>
- <https://github.com/axios/axios>. (7. 3 2021). *axios*. Načteno z axios:
<https://www.npmjs.com/package/axios>
- <https://github.com/brianc/node-postgres>. (7. 3 2021). *node-postgres*. Načteno z node-postgres:
<https://www.npmjs.com/package/pg>
- <https://github.com/expressjs/cookie-parser>. (7. 3 2021). *cookie-parser*. Načteno z cookie-
parser: <https://www.npmjs.com/package/cookie-parser>
- <https://github.com/expressjs/cors>. (21. 3 2021). *expressjs/cors*. Načteno z cors:
<https://www.npmjs.com/package/cors>

<https://github.com/kelektiv/node.bcrypt.js#readme>. (7. 3 2021). *node.bcrypt.js*. Načteno z node.bcrypt.js: <https://www.npmjs.com/package/bcrypt>

<https://github.com/kolodny/immutability-helper>. (7. 3 2021). *immutability-helper*. Načteno z immutability-helper: <https://www.npmjs.com/package/immutability-helper>

<https://github.com/LouisBrunner/dnd-multi-backend/tree/master/packages/react-dnd-multi-backend>. (7. 3 2021). *dnd-multi-backend*. Načteno z dnd-multi-backend: <https://www.npmjs.com/package/react-dnd-multi-backend>

<https://github.com/motdotla/dotenv>. (7. 3 2021). *dotenv*. Načteno z dotenv: <https://www.npmjs.com/package/dotenv>

<https://github.com/paulomcnally/node-heroku-ssl-redirect>. (16. 3 2021). *heroku-ssl-redirect*. Načteno z npm: <https://www.npmjs.com/package/heroku-ssl-redirect>

<https://github.com/paulomcnally/node-heroku-ssl-redirect>. (27. 3 2021). *node-heroku-ssl-redirect*. Načteno z node-heroku-ssl-redirect: <https://www.npmjs.com/package/heroku-ssl-redirect>

<https://github.com/react-bootstrap/react-bootstrap>. (7. 3 2021). *react-bootstrap*. Načteno z react-bootstrap: <https://www.npmjs.com/package/react-bootstrap>

<https://github.com/react-bootstrap/react-bootstrap>. (7. 3 2021). *react-bootstrap*. Načteno z react-bootstrap: <https://github.com/react-bootstrap/react-bootstrap>

<https://github.com/react-dnd/react-dnd>. (7. 3 2021). *react-dnd*. Načteno z react-dnd: <https://www.npmjs.com/package/react-dnd>

<https://github.com/react-dnd/react-dnd>. (7. 3 2021). *react-dnd*. Načteno z react-dnd: <https://www.npmjs.com/package/react-dnd-html5-backend>

<https://github.com/react-dnd/react-dnd>. (7. 3 2021). *react-dnd*. Načteno z react-dnd: <https://www.npmjs.com/package/react-dnd-touch-backend>

<https://github.com/react-hook-form/react-hook-form>. (7. 3 2021). *react-hook-form*. Načteno z react-hook-form: <https://www.npmjs.com/package/react-hook-form>

<https://github.com/uuidjs/uuid>. (7. 3 2021). *uuidjs*. Načteno z uuidjs: <https://www.npmjs.com/package/uuid>

PostgreSQL Global Development Group. (7. 3 2021). *postgresql.org*. Načteno z postgresql.org: <https://www.npmjs.com/package/firebase>

Walke, J. (7. 3 2021). *Reactjs.org*. Načteno z Reactjs.org: <https://reactjs.org/docs/getting-started.html>