**Gymnázium, Praha 6, Arabská**

Obor programování



Maturitní práce

Lucián Kučera

Recepty

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 28. března 2021

# Anotace

Aplikace slouží pro vytváření receptů s úmyslem, aby všechno bylo uspořádané a velice přehledné. S vytvářením a upravování 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 moct v aplikaci podle různých kritérii 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](#_Toc66818446)

[Abstract 3](#_Toc66818447)

[Zadání projektu 3](#_Toc66818448)

[1. Úvod 1](#_Toc66818449)

[2. Technologie 1](#_Toc66818450)

[2.1 Technologie použité pro vývoj 1](#_Toc66818451)

[2.2 Hlavní technologie 1](#_Toc66818452)

[2.3 Technologie serverové části 2](#_Toc66818453)

[2.4 Technologie klientské části 2](#_Toc66818454)

[3. Struktura souborů 3](#_Toc66818455)

[3.1 Klient 3](#_Toc66818456)

[3.2 Server 3](#_Toc66818457)

[4. Struktura Backendu 3](#_Toc66818458)

[4.1 Návrh databáze 3](#_Toc66818459)

[4.2 Autentizace 5](#_Toc66818460)

[4.3 Struktura API 5](#_Toc66818461)

[4.4 Uložiště obrázků 5](#_Toc66818462)

[5. Struktura Frontendu 5](#_Toc66818463)

[5.1 Stránka Přihlášení 5](#_Toc66818464)

[5.2 Stránka Registrace 6](#_Toc66818465)

[5.3 Hlavní stránka 6](#_Toc66818466)

[5.4 Formulář receptu 7](#_Toc66818467)

[5.5 Stránka vašeho receptu 8](#_Toc66818468)

[5.6 Stránka Ingredience a Nástroje 9](#_Toc66818469)

[5.7 Stránka sdílených receptů 10](#_Toc66818470)

[5.8 Stránka sdíleného receptu 11](#_Toc66818471)

[6. Funkce 12](#_Toc66818472)

[6.1 Tvorba receptu 12](#_Toc66818473)

[6.2 Přidání ingredience uživatelem 14](#_Toc66818474)

[6.3 Přidání nástroje uživatelem 14](#_Toc66818475)

[6.4 Sdílení receptu 14](#_Toc66818476)

[6.5 Vyhledávání sdílených receptů 14](#_Toc66818477)

[6.6 Vytváření komentářů 14](#_Toc66818478)

[7. GUI 15](#_Toc66818479)

[8. Testování 15](#_Toc66818480)

[8.1 Testovaní backendu 15](#_Toc66818481)

[8.2 Testování databáze 15](#_Toc66818482)

[8.3 Testování frontendu 15](#_Toc66818483)

[9. Heroku 15](#_Toc66818484)

[10. Návod na spuštění 16](#_Toc66818485)

[10.1 Aplikace na Heroku 16](#_Toc66818486)

[10.2 Lokální spuštění 16](#_Toc66818487)

[10.3 Instalace na Heroku 16](#_Toc66818488)

[11. Závěr 18](#_Toc66818489)

[12. Seznam Obrázků 18](#_Toc66818490)

[13. Bibliografie 18](#_Toc66818491)

1. Úvod

Tuto aplikaci jsem vytvořil, protože jsem na trhu neviděl moc webových aplikací, které by poskytovaly uživateli kontrolu nad vytváření receptů do vysoké míry. Tato aplikace byla navržena hlavně s myšlenkou kontroly a organizace a druhou hlavní myšlenkou bylo rozšiřovaní, 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 „middlewarů“, 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 middlewaru.

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

1. Technologie
   1. Technologie použité pro vývoj

* Heroku
  + Technologie pro hosting
* Chrome developer tools
  + Testovaní aplikace
* Visual studio code
  + Vývoj kódu aplikace
* Psql
  + Přístup k PostgreSQL databázi a psaní příkazů pro databázi
  + Příkazová řádka
  1. Hlavní technologie

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

* PostrgeSQL (PostgreSQL Global Development Group, 2021)
  + SQL databáze slouží pro ukládaní dat
* Express.js (Holowaychuk, 2021)
  + Javascriptový framework pro vytváření serveru, nadstavba Node.js
* React.js (Walke, 2021)
  + javascriptová knihovna pro efektivní vytváření frontendu webových aplikací, které fungují na všech moderních browserech.
* Node.js (Dahl, 2021)
  + Javascriptové serverové prostředí
  1. Technologie serverové části
* Bcrypt (https://github.com/kelektiv/node.bcrypt.js#readme, 2021)
  + hashování hesel
* Cookie-parser (https://github.com/expressjs/cookie-parser, 2021)
  + čtení cookies na serveru
* Cors (https://github.com/expressjs/cors, 2021)
  + povoluje čtení požadavků mezi doménami
* Jsonwebtoken (https://github.com/auth0/node-jsonwebtoken, 2021)
  + vytváření žetonů uživatelům, pro získání přístupu v aplikaci
* Pg(node-postgres) (https://github.com/brianc/node-postgres, 2021)
  + Připojí PostgreSQL databázi
  1. Technologie klientské části
* React-hook-form (https://github.com/react-hook-form/react-hook-form, 2021)
  + Jednoduché zpracovaní dat ve formulářích a testovaní dat ve formulářích
* Axios (https://github.com/axios/axios, 2021)
  + Posílaní požadavků serveru a čtení odpovědi serveru
* React-bootstrap (https://github.com/react-bootstrap/react-bootstrap, react-bootstrap, 2021)
  + Knihovna předem vytvořených komponentů pro jednoduchou stylizaci aplikace
* Firebase (Google, 2021)
  + Ukládaní obrázků na cloudovém uložišti
* react-dnd (https://github.com/react-dnd/react-dnd, react-dnd, 2021)
  + táhni a pusť logika
* react-dnd-multi-backend (https://github.com/LouisBrunner/dnd-multi-backend/tree/master/packages/react-dnd-multi-backend, 2021)
  + spojí dotykový backend s html5 backendem
* react-dnd-html5-backend (https://github.com/react-dnd/react-dnd, react-dnd, 2021)
  + umožnuje táhni a pusť s myší
* react-dnd-touch-backend (https://github.com/react-dnd/react-dnd, react-dnd, 2021)
  + umožňuje táhni a pusť na dotykové obrazovce
* immutability-helper (https://github.com/kolodny/immutability-helper, 2021)
  + pomáhá upravit neměnná data
* uuid (https://github.com/uuidjs/uuid, 2021)
  + generuje uuid, uuid je jedinečný klíč

1. Struktura souborů
   1. Klient

Nachází se ve složce client/src.

* axios
  + instance axiosu
* components
  + react komponenty
* config
  + Konfigurace Firebase
* context
  + zde se ukládají instance React Context API
* pages
  + stránky aplikace
* queries
  + dotazy na server
* responsiveCss
  + css
* utils
  + pomocné metody
  1. Server

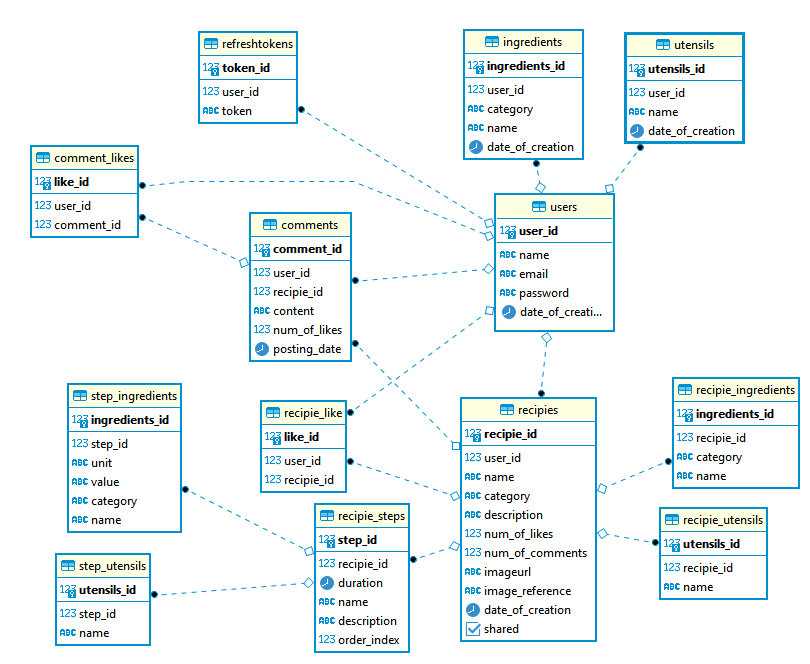
Nachází se ve složce server.

* apis
  + REST API
* Configuration
  + Konfigurace databáze
* Midelware
  + Express middleware
* query\_functions
  + pomocné metody pro řešení komplikovaných dotazů od klienta
* utils
  + pomocné metody

1. Struktura Backendu
   1. Návrh databáze

Jako hlavní databázi jsem si vybral PostgreSQL, je velice jednoduchá nainstalovat a používat lokálně a na Heroku. Dalším důvodem byla potřeba mojí aplikace mít hodně vztahů mezi daty a SQL databáze jsou na to ideální, poslední důvod je opensource.

Moje databázové schéma:



Obrázek 1 ER model mojí databáze

Tabulky:

* users
  + uživatelé
* refreshtokens
  + obnovovací žetony
* ingredients
  + uživatelovy ingredience
* utensils
  + uživatelovy nástroje
* recipies
  + recepty
* recipie\_ingredients
  + ingredience receptu
* recipie\_utensils
  + nástroje receptu
* recipie\_steps
  + kroky receptu
* step\_utensils
  + nástroje kroku
* step\_ingredients
  + ingredience kroku
* recipie\_like
  + označení receptu za oblíbené
* comments
  + komentáře receptu
* comments\_like
  + označeni komentáře za oblíbený
  1. 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.

* 1. Struktura API

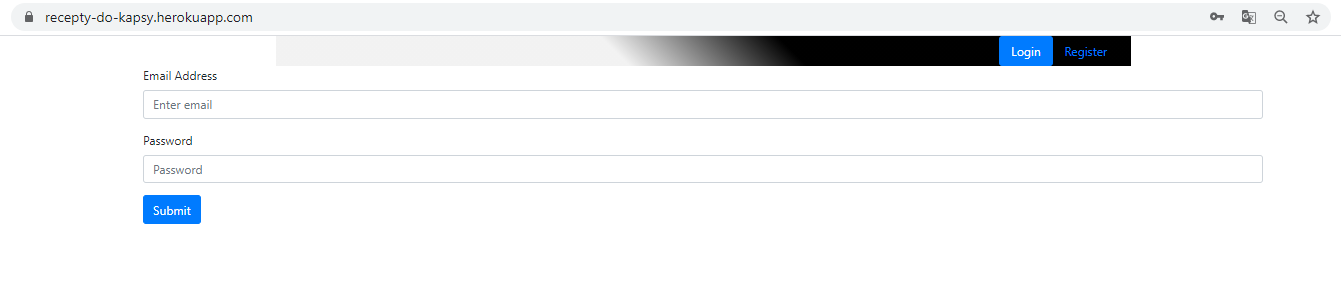
Pro vytváření vlastních API používám serverový framework Express.js. Každá API je určená pro jiná data, jedna pro jiné data v databázi. Například recepty nebo uživatelé. Tyto API jsou v zásobníku a pokaždé, když uživatel pošle požadavek serveru, tak najde vhodnou odpověď.

* 1. Uložiště obrázků

Pro ukládaní 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 přidal zabezpečení pro uživatele, že jenom konkrétní uživatel se může dostat do své složky.

1. Struktura Frontendu
   1. Stránka Přihlášení

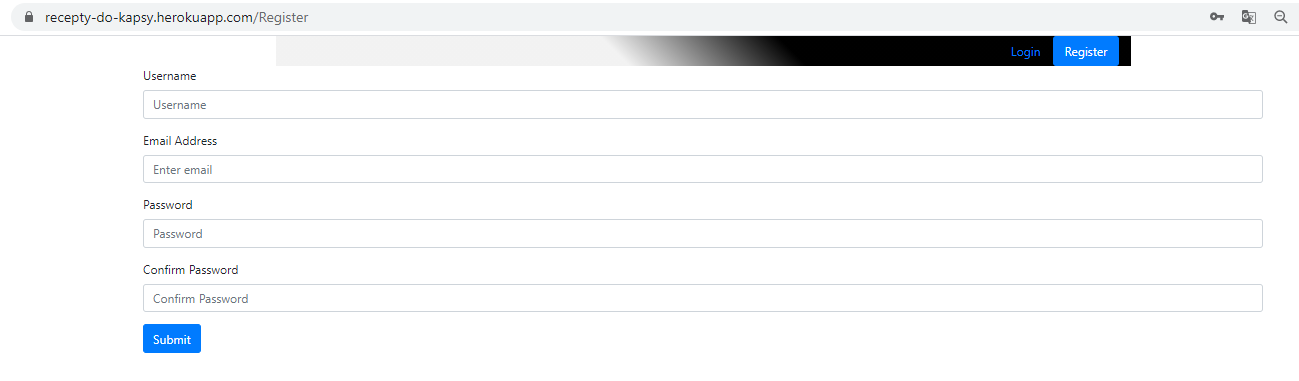
Stránka přihlášení je přihlašovací formulář, který po vyplnění zjistí, zda jsou jím zadaná data správná a pokud jsou správná, tak uživatele přesměruje na hlavní stránku.



Obrázek 2Stránka přihlášení

* 1. Stránka Registrace

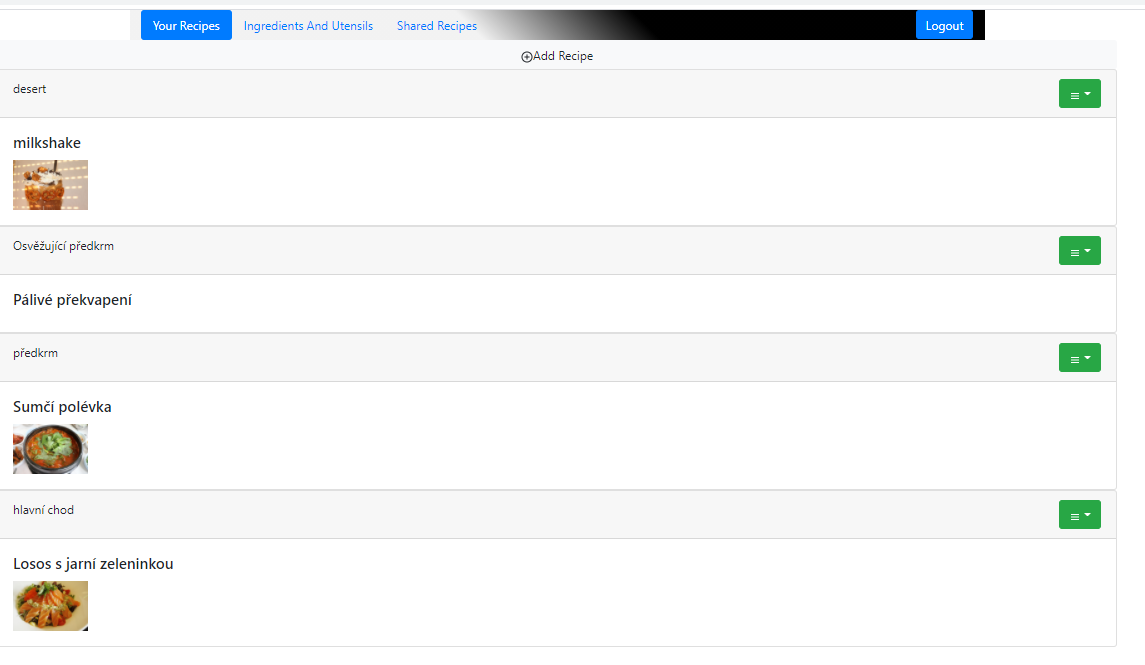
Na stránce pro registraci si uživatel vybere e-mail, který je jedinečný v databázi, uživatelské jméno, to se smí opakovat, heslo a potvrdí heslo. Pokud data projdou validací, tak se založí jeho účet.



Obrázek 3Stránka registrace

* 1. Hlavní stránka

Na hlavní stránce se zobrazují karty uživatelových receptů. A odkaz na formulář pro vytvoření receptu. Z karty receptu se lze přesměrovat na sdílenou stránku receptu, pokud recept je sdílený, anebo na personální stránku receptu, kde recept můžete upravit smazat nebo sdílet.

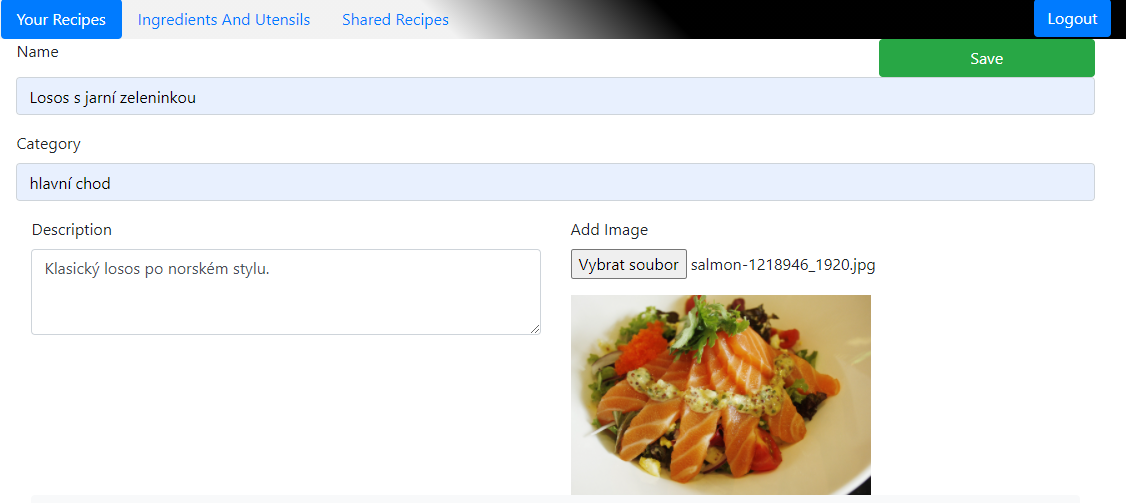


Obrázek 4Hlavní stránka

* 1. Formulář receptu

Formulář receptu se skládá ze dvou částí:

1. V první části formuláře, pokud uživatel chce, tak si může přetáhnout ingredience nebo nástroje do receptu technikou táhni a pusť.
2. Druhá část se dělí na základní data receptu, ingredience a nástroje receptu, formulář kroku, kroky:
   1. V základních datech receptu si uživatel zvolí název receptu, obrázek, kategorii a popis.
   2. V ingrediencích a nástrojích receptu uživatel vidí ingredience a nástroje receptu.
   3. 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.
   4. Kroky si můžete seřadit, upravit a proházet ingredience mezi nimi.

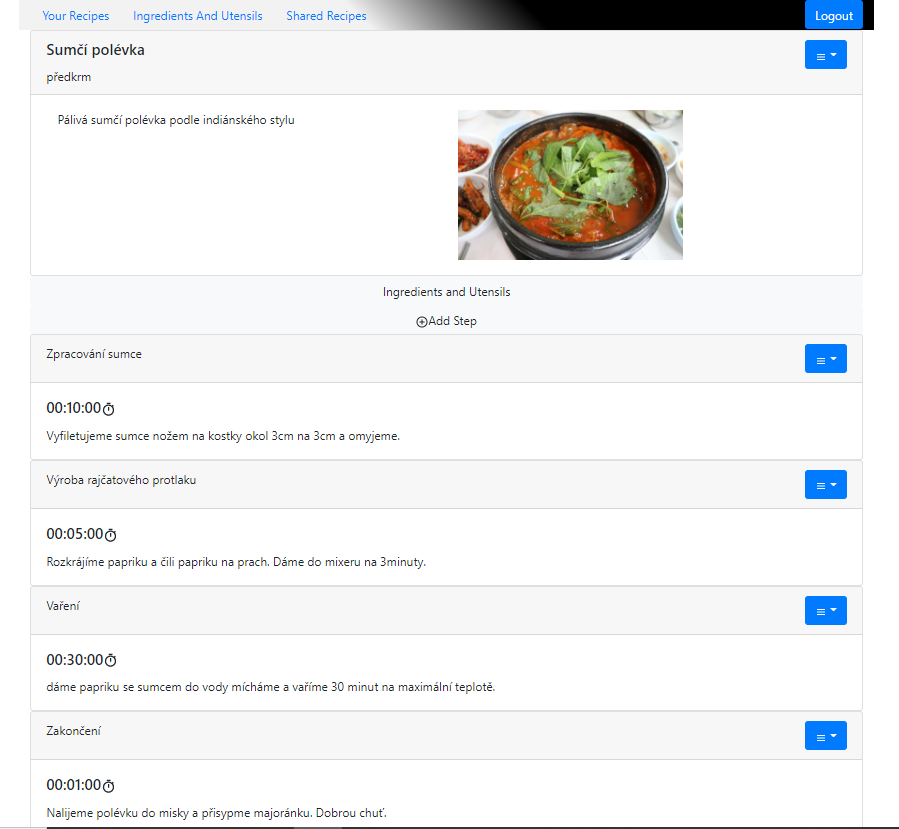


Obrázek 5 Formulář základních dat receptu

* 1. Stránka vašeho receptu

Tato stránka slouží k úpravě receptu a přečtení vlastních receptů. Skládá se z:

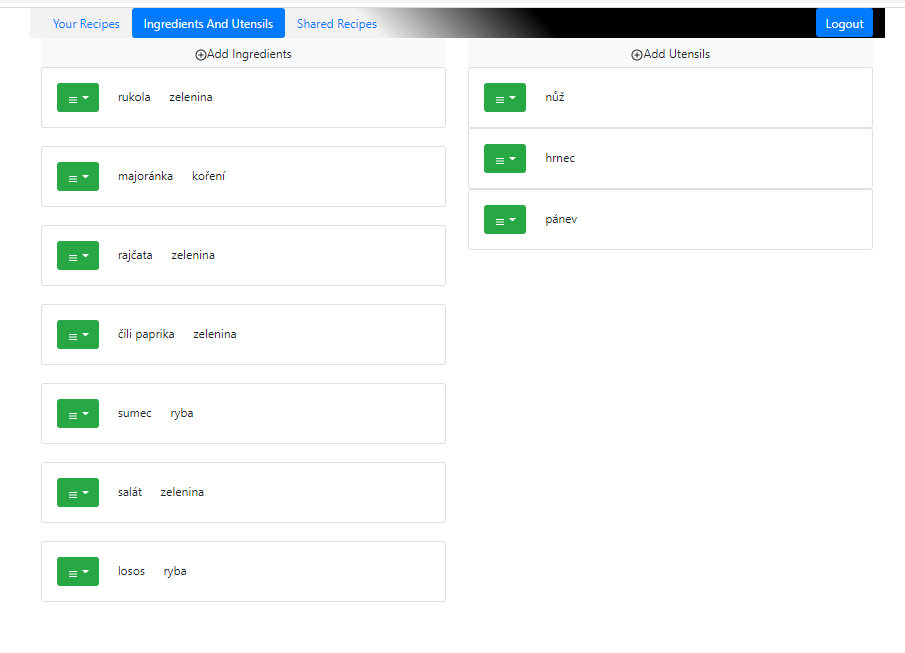
1. Základních dat receptu, zde vidíte název receptu, popis receptu, kategorii receptu, obrázek receptu a tlačítko s možnostmi: úprava, smazaní nebo sdílení. Při úpravě dostanete možnost tato data upravit ve formuláři.
2. Ingredience a Nástroje receptu, po kliknutí na tlačítko se vám zobrazí nástroje 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.
3. Formulář kroků receptu umožňuje uživateli v jeho již vytvořeném receptu přidat další kroky. Krok se skládá z popisu, názvu, času, ingrediencí a nástrojů.
4. V seznamu kroků lze seřadit kroky dle libosti, upravit, a dokonce i smazat.



Obrázek 6 Stránka vlastního receptu

* 1. Stránka Ingredience a Nástroje

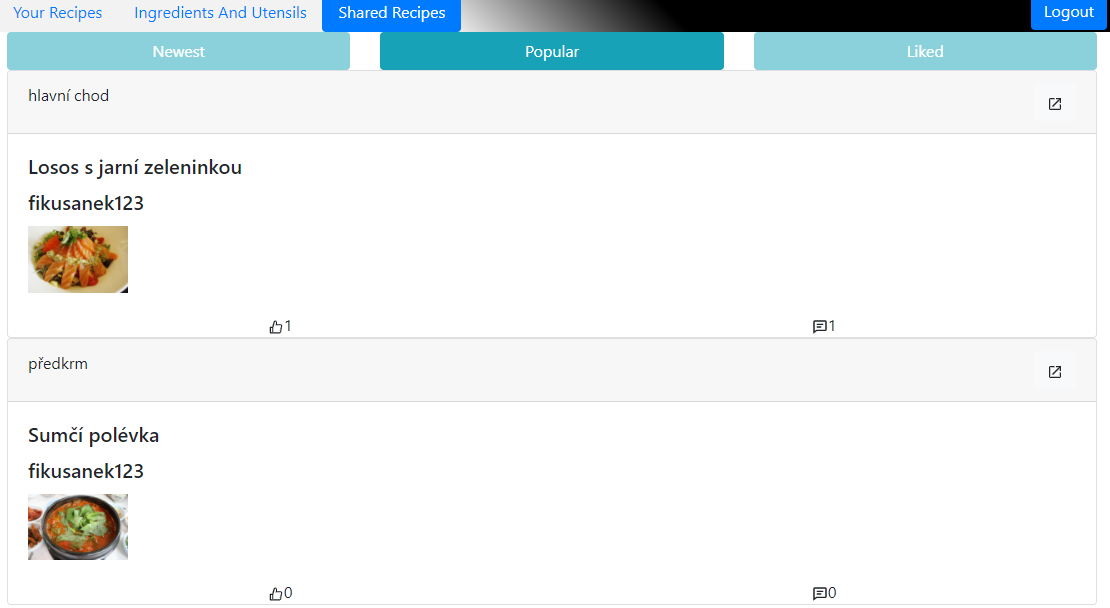
Na této stránce si uživatel může nadefinovat svoje ingredience a nástroje, které muže využít pro tvorbu vlastního receptu. Ingredience si uživatel může upravit nebo je smazat, to samé může uživatel s nástroji.



Obrázek 7Stránka ingredience a nástroje

* 1. 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 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 přesměrovaní, počet komentářů, počet oblíbení, popis, název, kategorii a fotku.

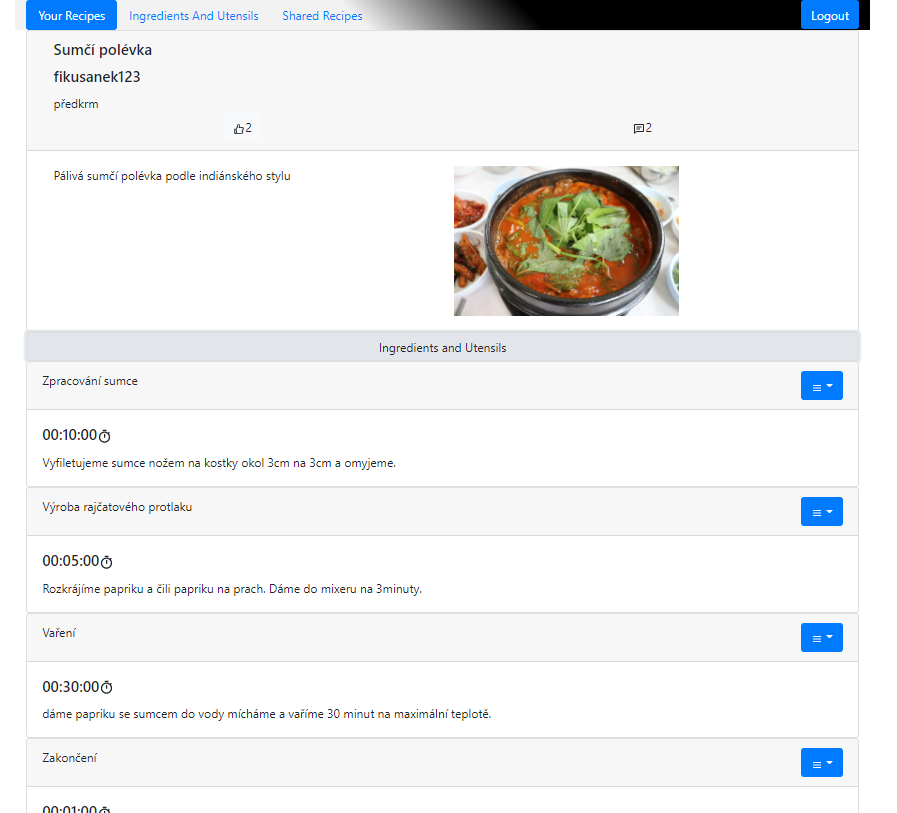


Obrázek 8 Stránka sdílených receptů

* 1. Stránka sdíleného receptu

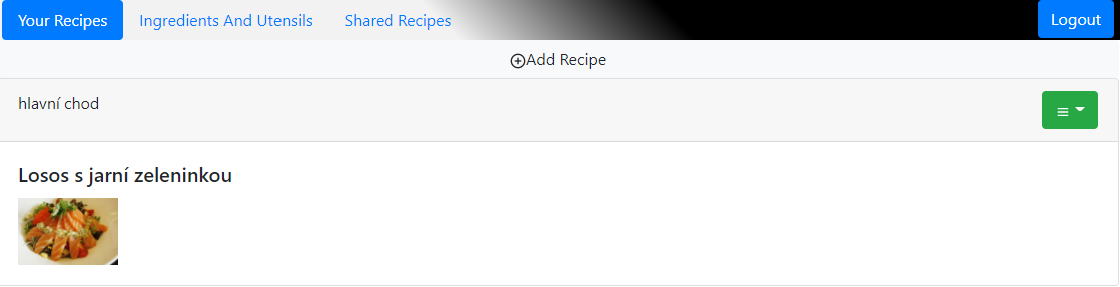
Tato stránka slouží pro přečtení receptu a ohodnocení receptu. Složení Stránky:

1. Základní informace receptu jméno, kategorie, popis obrázek.
2. Ingredience a nástroje receptu
3. Kroky, krok se skládá z jména, času trvání, kategorie, ingrediencí a nástrojů
4. Formulář pro tvorbu nových komentářů a komentáře. Komentář lze upravit, označit za oblíbený a smazat.

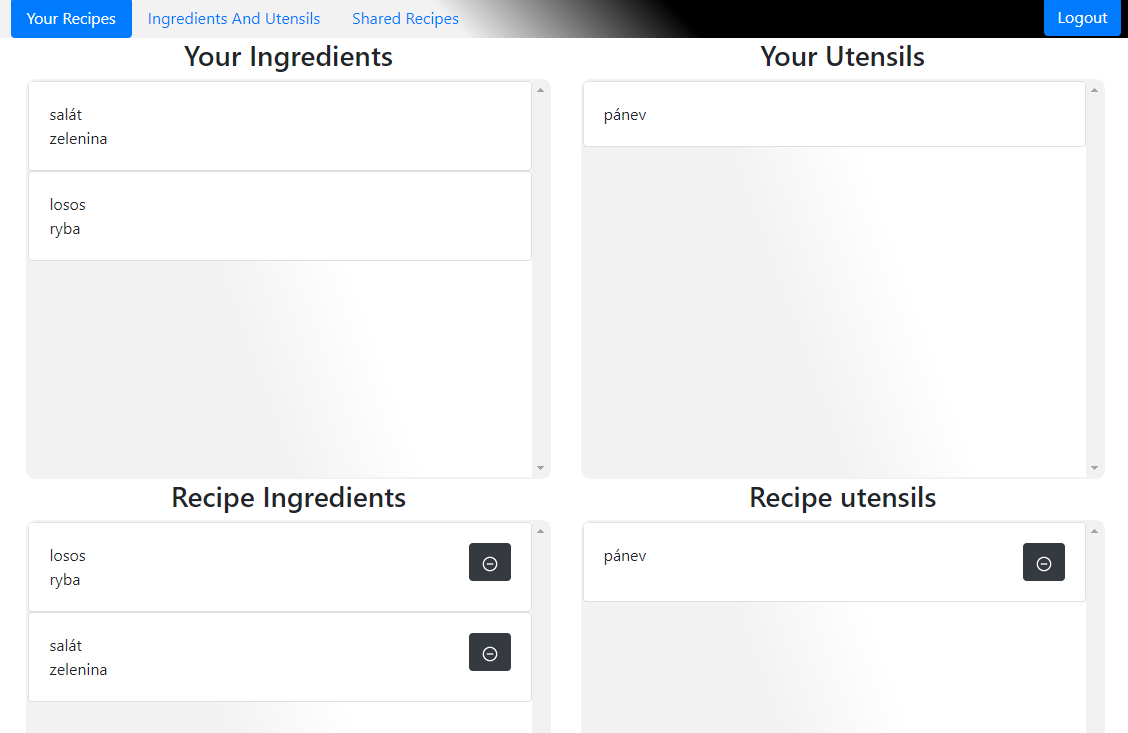


Obrázek 9Stránka sdíleného receptu

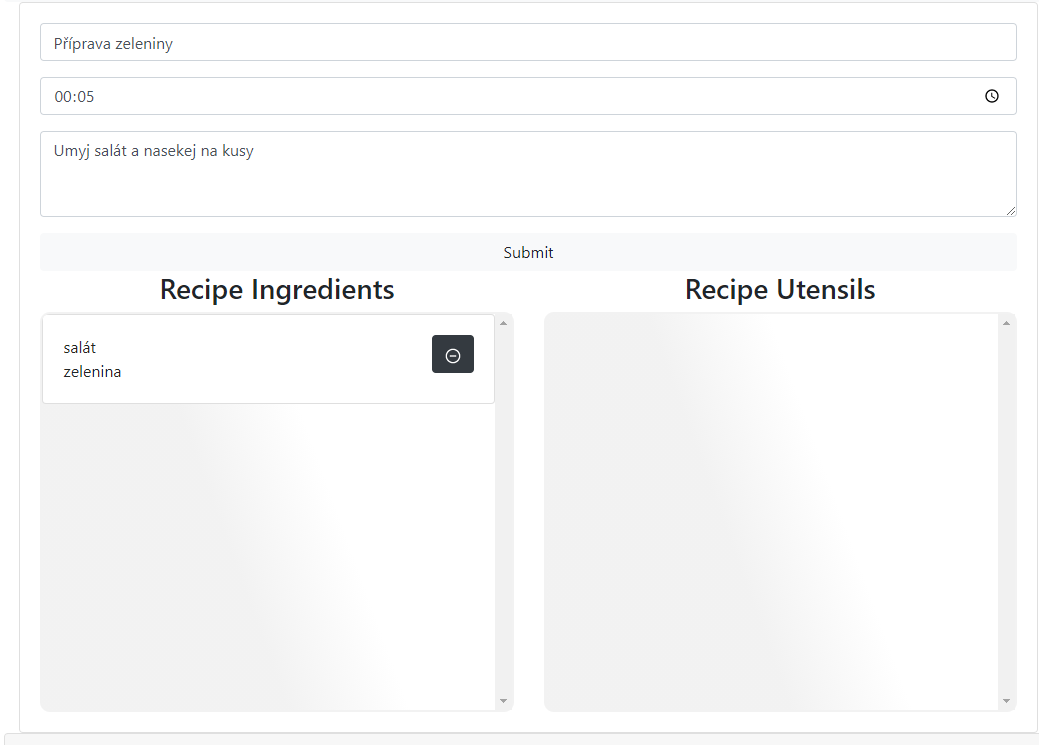
1. Funkce
   1. Tvorba receptu

Nejdříve uživatel musí zajít na hlavní stránku tam klikne na tlačítko přidat recept, to ho přenese na stránku formuláře receptu. Následně se uživatel objeví na stránce formuláře, si může přetáhnout všechny ingredience a nástroje, které potřebuje. 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. Kroky je možné seřadit a zpětně upravit. 

Obrázek 10 Hlavní stránka



Obrázek 11 Přesouvání ingrediencí a nástrojů



Obrázek 12 Formulář kroku receptu

* 1. Přidání ingredience uživatelem

Na stránce ingredience a nástroje, když uživatel zmáčkne tlačítko přidat ingredienci, tak se otevře formulář. Ve formuláři jsou povinná pole název a kategorie.

* 1. Přidání nástroje uživatelem

Na stránce ingredience a nástroje, když uživatel zmáčkne tlačítko přidat nástroj, tak se otevře formulář. Ve formuláři musí uživatel vyplnit název nástroje.

* 1. Sdílení receptu

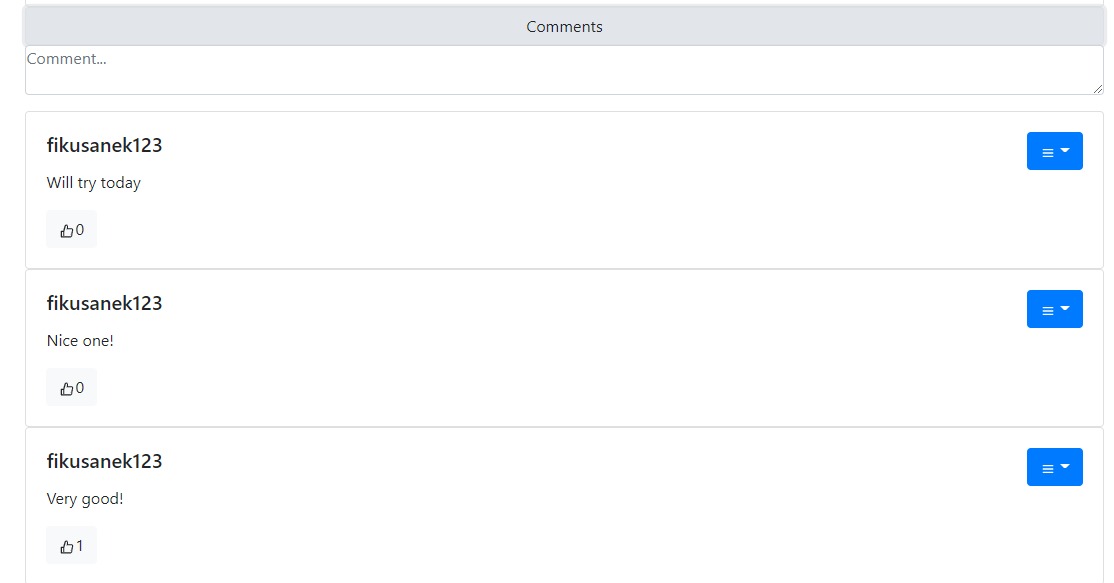
Na stránce osobního receptu je potřeba zmáčknout hamburgerové tlačítko možností a vybrat možnost sdílet.

* 1. Vyhledávání sdílených receptů

Na stránce sdílených receptů je na výběr buď nejnovější, nejoblíbenější nebo oblíbené. Pro vybrání metody vyhledávání je třeba vybrat tlačítko se správným názvem.

* 1. Vytváření komentářů

Komentáře jdou vytvořit na sdílených receptech. Jediné pole nutné vyplnit je obsah komentáře.



Obrázek 13 Komentáře receptu

1. GUI

Grafické rozhraní aplikace bylo vytvořeno pomocí komponentů z knihovny react-bootstrap a css. Knihovna react-bootstrap mi umožnila jednoduše vyzdobit aplikaci a vytvořit rozložení v aplikaci.

1. Testování
   1. Testovaní backendu

Pro testovaní API požadavků jsem použil aplikaci postman, kde jsem zadal testovací data adresu a typ požadavku. Data jsem zadával v JSON datovém formátu.

* 1. Testování databáze

Databázové příkazy jsem si testoval ve psql PostgreSQL příkazové řádce.

* 1. Testování frontendu

Pro vyzkoušení funkčností webové aplikace jsem použil Chrome rozšíření React development tools.

1. Heroku

Jako hostitelskou službu jsem zvolil Heroku, protože má skvělou podporu PostgreSQL. Také podporuje Node.js prostředí, ve kterém jsem aplikaci vytvářel. Přes různé překážky jako vytváření vlastních soukromých hodnot a nastavovaní package.json se rozestavení zdařilo.

1. Návod na spuštění
   1. Aplikace na Heroku

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

* 1. Lokální spuštění

Další možností je lokální instalace.

1. Nejdůležitější je mít stažený Node.js.
2. Použít příkaz v složce client
   1. npm i
3. Použít příkaz v složce server
   1. npm i
4. Vytvořit firebase projekt <https://firebase.google.com/>
5. Povolit authentikaci pro email/password ve vašem firebase projektu
6. Přidat webovou aplikaci do firebase projektu. To vygeneruje konfiguraci s vzhledem:
   1. var firebaseConfig = {  
          apiKey: "apiKey",  
          authDomain: "authDomain",  
          projectId: "projectId",  
          storageBucket: "storageBucket",  
          messagingSenderId: "messagingSenderId",  
          appId: "appId",  
          measurementId: "measurementId"  
        };
7. Tuto konfiguraci je třeba přidat do souboru, který je třeba vytvořit v client/src/config/ s názvem firebaseConfig.js. Obsah souboru bude vypadat:
   1. export default {  
          apiKey: "apiKey",  
          authDomain: "authDomain",  
          projectId: "projectId",  
          storageBucket: "storageBucket",  
          messagingSenderId: "messagingSenderId",  
          appId: "appId",  
          measurementId: "measurementId"  
        };
8. Vytvořit ve Firebase projektu storage
9. 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;

}

}

}

1. Vytvořit lokálně postgresql databázi
2. Je třeba stáhnout postgresql na počítač port, nastavit na 5432 lokálně a v server/ vytvořit soubor .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

1. Také vytvořit v databázi tabulky v souboru server/database.sql.
2. Nechat si vygenerovat konfiguraci serviceAccountu ve Firebase a uložit ho do /server/utils. Je třeba též změnit název cesty serviceAccountu v server/utils/jwtGenerator.js.
   1. Settings=>project settings=>aservice sccounts=>generate new private key To vám stáhne na počítač klíč ve formátu JSON
   2. Přesuňte tento soubor do server/utils/
   3. V souboru server/utils/jwtGenerator.js změňte v require(‘./{název souboru s klíčem}‘)
      1. const serviceAccount = require('./název souboru klíče)
3. Server spustíte ve složce server npm run dev
4. Klienta spustíte ve složce client npm start
   1. Instalace na Heroku

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

1. Nejdřív splnit konfiguraci pro spuštění lokálně
2. Je třeba vymazat localhost:5000 ve všech volání axiosu v klientovi. Zavolání serveru má vypadat /cesta, cesta je cesta v serverovém rozcestníku
3. Najít si heroku konfiguraci pro postgresql a přidat do server/configuration/db.js místo aktuální konfigurace.

const proConfig = ({

connectionString: process.env.DATABASE\_URL,

ssl: {

rejectUnauthorized: false

}

});

1. Místo konstanty pool v server/configuration/db.js

const pool = new Pool (proConfig);

1. Také v server/index.js je třeba změnit adresu pro CookieParser na heroku url
2. Je třeba vymazat npm\_modules a package-lock.json v složce clientovi, potom spustit příkaz npm i v složce client a nakonec je třeba v client složce zadat příkaz npm run build
3. Na vrchol index.js souboru v hlavní složce přidat:
   1. const path = require(‘path‘);
4. V server/index.js vytvořit express midelwear, který zobrazí client/build/inex.html.

if (process.env.NODE\_ENV === "production") {

app.use(express.static(path.join(\_\_dirname, "client/build")));

}

1. Přidat catch all methodu za vsema middelwearama

app.get('\*', (req, res) => {

res.sendFile(path.join(\_\_dirname, "client/build/index.html"));

});

1. Je třeba přesunout obsah složky server do hlavní složky a vymazat složku server.
2. Přidat do heroku postgresql a vytvořit v databázi tabulky v souboru database.sql.

Mít nainstalované heroku příkazy

heroku login

heroku create

heroku addons:create heroku-postgresql:hobby-dev

heroku pg:psql -a "name-app"

je třeba mít Postgres shell na portu 5432

přidat všechny tabulky v souboru databse.sql

1. V package.json v hlavní složce je třeba vymazat skript „dev“:„nodemon index“ a je třeba přidat dva skripty "start": "node index", "heroku-postbuild": "cd client && npm i && npm run build"
2. Přidat config vars na heroku SECRET1, SECRET2
3. Spustit tyto příkazy v hlavní složce
   1. git add .
   2. git commit -m “heroku“
   3. git push heroku master
4. 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 přidávaní dalších vlastností. Také by se dal vytvořit podobný projekt pro workflow managment. Koncept této aplikace lze rozhodně využít ve všech různých odvětvích.

1. Seznam Obrázků

[Obrázek 1 ER model mojí databáze 4](#_Toc66301564)

[Obrázek 2Stránka přihlášení 6](#_Toc66301565)

[Obrázek 3Stránka registrace 6](#_Toc66301566)

[Obrázek 4Hlavní stránka 7](#_Toc66301567)

[Obrázek 5 Formulář základních dat receptu 8](#_Toc66301568)

[Obrázek 6 Stránka vlastního receptu 9](#_Toc66301569)

[Obrázek 7Stránka ingredience a nástroje 10](#_Toc66301570)

[Obrázek 8 Stránka sdílených receptů 11](#_Toc66301571)

[Obrázek 9Stránka sdíleného receptu 12](#_Toc66301572)

[Obrázek 10 Hlavní stránka 13](#_Toc66301573)

[Obrázek 11 Přesouvání ingrediencí a nástrojů 13](#_Toc66301574)

[Obrázek 12 Formulář kroku receptu 14](#_Toc66301575)

[Obrázek 13 Komentáře receptu 15](#_Toc66301576)

1. 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/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). *posgresql.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