

Gymnázium, Praha 6, Arabská

Obor programování



Maturitní práce

Lucián Kučera

Recepty

březen 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, Arabská<sup>14</sup> 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 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
2.2 Hlavní technologie.....	1
2.3 Technologie serverové části.....	2
2.4 Technologie klientské části.....	2
3. Struktura souborů .....	3
3.1 Klient .....	3
3.2 Server.....	3
4. Struktura Backendu .....	3
4.1 Návrh databáze.....	3
4.2 Autentizace.....	5
4.3 Struktura API .....	5
4.4 Uložiště obrázků .....	5
5. Struktura Frontendu.....	5
5.1 Stránka Přihlášení .....	5
5.2 Stránka Registrace .....	6
5.3 Hlavní stránka .....	6
5.4 Formulář receptu.....	7
5.5 Stránka vašeho receptu .....	8
5.6 Stránka Ingredience a Nástroje .....	9
5.7 Stránka sdílených receptů.....	10
5.8 Stránka sdíleného receptu.....	11
6. Funkce.....	12
6.1 Tvorba receptu .....	12
6.2 Přidání ingredience uživatelem.....	14
6.3 Přidání nástroje uživatelem .....	14
6.4 Sdílení receptu .....	14
6.5 Vyhledávání sdílených receptů.....	14

6.6	Vytváření komentářů .....	14
7.	GUI.....	15
8.	Testování.....	15
8.1	Testování backendu .....	15
8.2	Testování databáze.....	15
8.3	Testování frontendu .....	15
9.	Heroku.....	15
10.	Návod na spuštění.....	16
10.1	Aplikace na Heroku .....	16
10.2	Lokální spuštění.....	16
10.3	Instalace na Heroku .....	18
11.	Závěr.....	21
12.	Seznam Obrázků.....	22
13.	Bibliografie .....	22

# 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 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

- Heroku
  - Technologie pro hosting
- Chrome developer tools
  - Testování aplikace
- Visual studio code
  - Vývoj kódu aplikace
- Psql
  - Přístup k PostgreSQL databázi a psaní příkazů pro databázi
  - Příkazová řádka

### 2.2 Hlavní technologie

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

- PostgreSQL (PostgreSQL Global Development Group, 2021)
  - SQL databáze slouží pro ukládání 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 brouzerech.
- Node.js (Dahl, 2021)
  - Javascriptové serverové prostředí

## 2.3 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-jwebtoken>, 2021)
  - vytváření žetonů uživatelům, pro získání přístupu v aplikaci
- Pg(node-postgres) (<https://github.com/brianc/node-postgres>, 2021)
  - Připojí PostgreSQL databázi

## 2.4 Technologie klientské části

- React-hook-form (<https://github.com/react-hook-form/react-hook-form>, 2021)
  - Jednoduché zpracování dat ve formulářích a testování dat ve formulářích
- Axios (<https://github.com/axios/axios>, 2021)
  - Posílání požadavků serveru a čtení odpovědi serveru
- React-bootstrap (<https://github.com/react-bootstrap/react-bootstrap>, 2021)
  - Knihovna předem vytvořených komponentů pro jednoduchou stylizaci aplikace
- Firebase (Google, 2021)
  - Ukládání obrázků na cloudovém uložišti
- react-dnd (<https://github.com/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>, 2021)
  - umožňuje táhni a pusť s myší
- react-dnd-touch-backend (<https://github.com/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

## 3. Struktura souborů

### 3.1 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

### 3.2 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

## 4. Struktura Backendu

### 4.1 Návrh databáze

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

Moje databázové schéma:





- step\_ingredients
  - ingredience kroku
- recipe\_like
  - označení receptu za oblíbené
- comments
  - komentáře receptu
- comments\_like
  - označení komentáře za oblíbený

## 4.2 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.

## 4.3 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 odpověď.

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

# 5. Struktura Frontendu

## 5.1 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 2 Stránka přihlášení

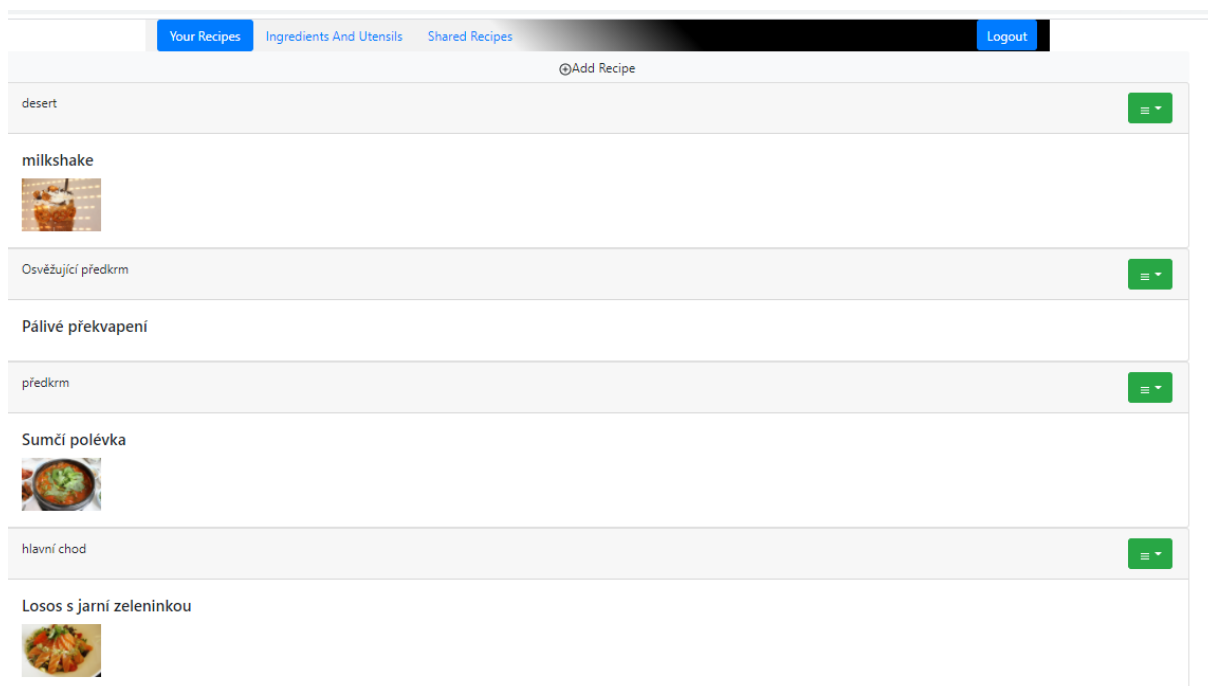
## 5.2 Stránka Registrace

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 3 Stránka registrace

## 5.3 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 4 Hlavní stránka

## 5.4 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:
  - a. V základních datech receptu si uživatel zvolí název receptu, obrázek, kategorii a popis.
  - b. V ingrediencích a nástrojích receptu uživatel vidí ingredience a nástroje receptu.
  - c. 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.
  - d. Kroky si můžete seřadit, upravit a proházet ingredience mezi nimi.

Your Recipes
Ingredients And Utensils
Shared Recipes
Logout

Name
Save

Losos s jarní zeleninkou

Category


hlavní chod

Description

Klasický losos po norském stylu.

Add Image

Vybrat soubor salmon-1218946\_1920.jpg



Obrázek 5 Formulář základních dat receptu

## 5.5 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, smazání 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.

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

Sumčů polėvka
předkrm

Pálivá sumčů polėvka podle indiánského stylu



Ingredients and Utensils

+Add Step

Zpracování sumce

00:10:00

Vyřetujeme 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

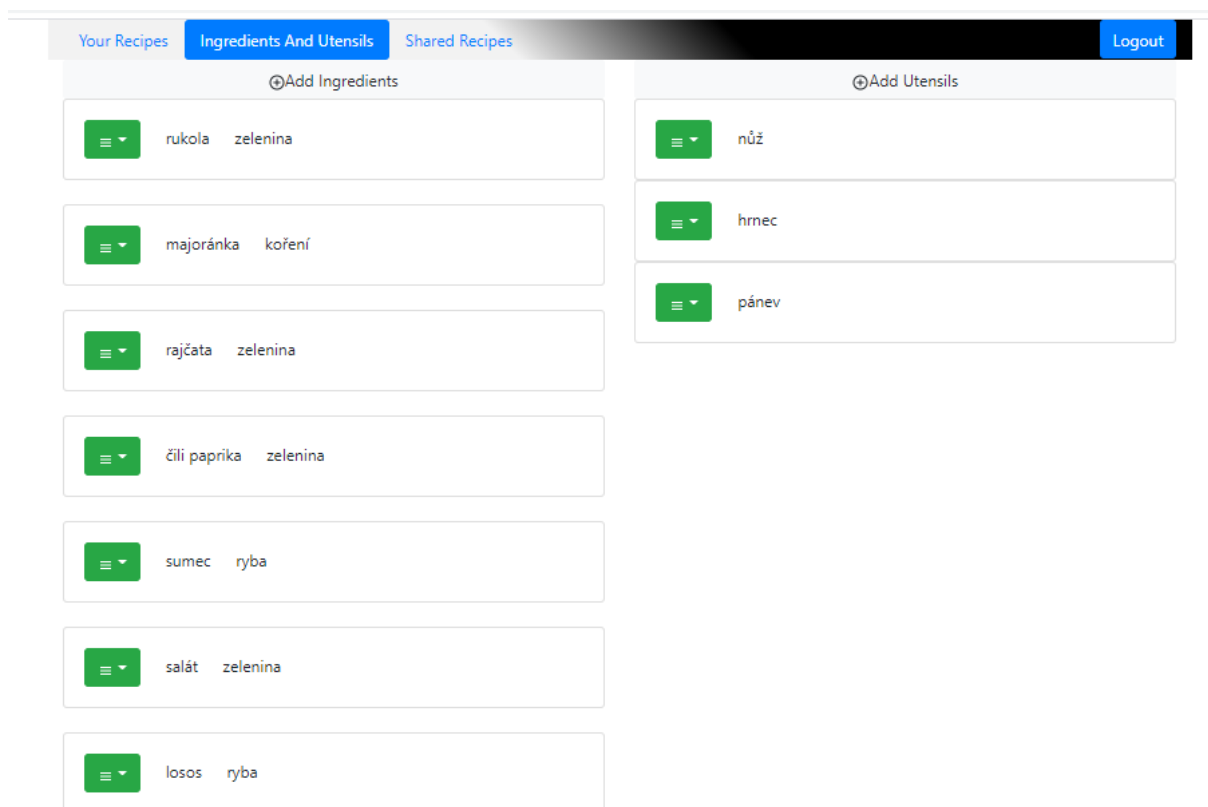
00:01:00

Nalijeme polėvku do misky a přisypme majoránku. Dobrou chuť.

Obrázek 6 Stránka vlastního receptu

## 5.6 Stránka Ingredience a Nástroje

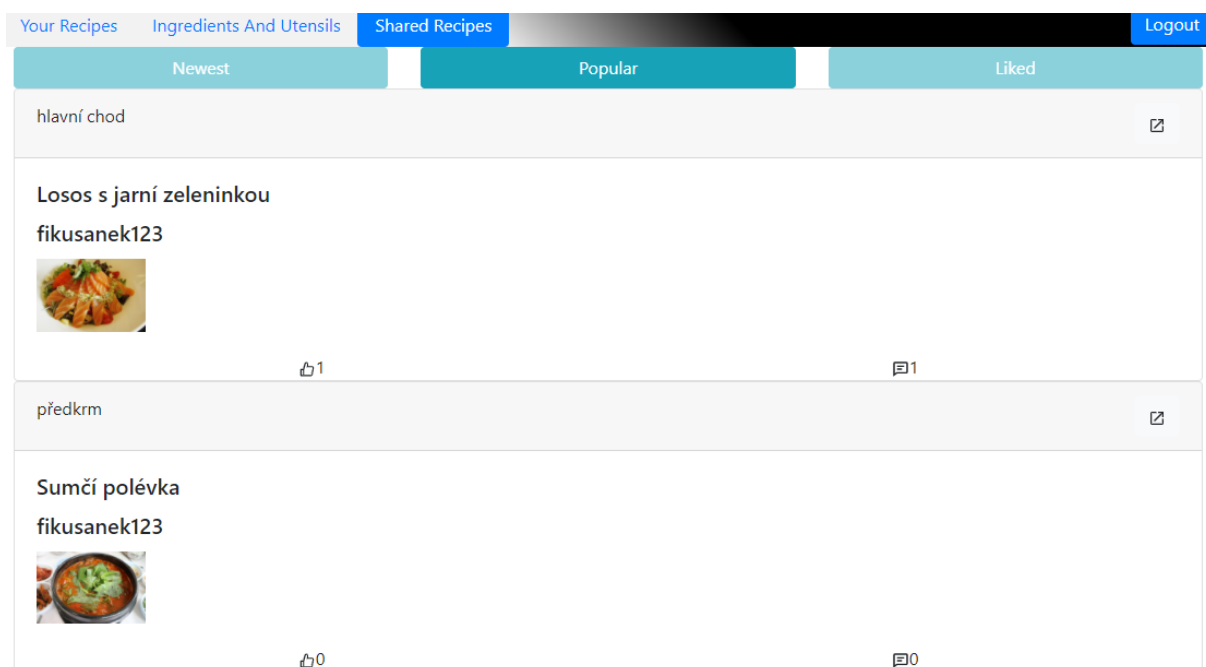
Na této stránce si uživatel může nadefinovat svoje ingredience a nástroje, které může využít pro tvorbu vlastního receptu. Ingredience si uživatel může upravit nebo je smazat, to samé může uživatel s nástroji.



Obrázek 7 Stránka ingredience a nástroje

## 5.7 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 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

## 5.8 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.



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 ⌚

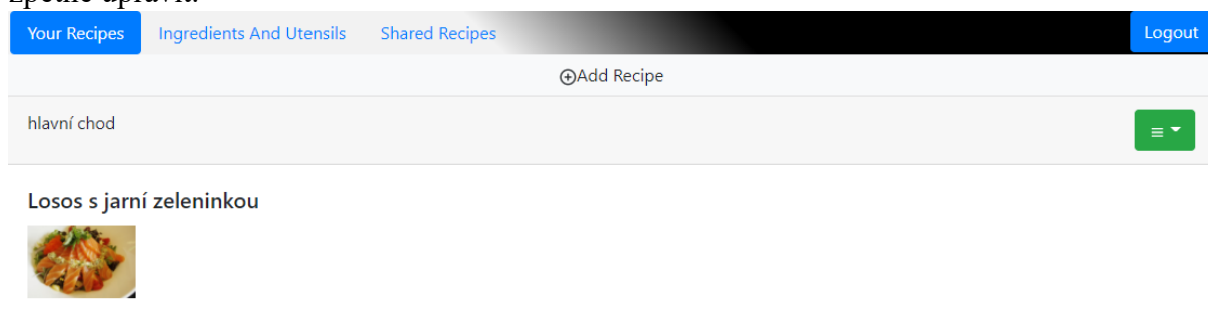
Obrázek 9 Stránka sdíleného receptu

## 6. Funkce

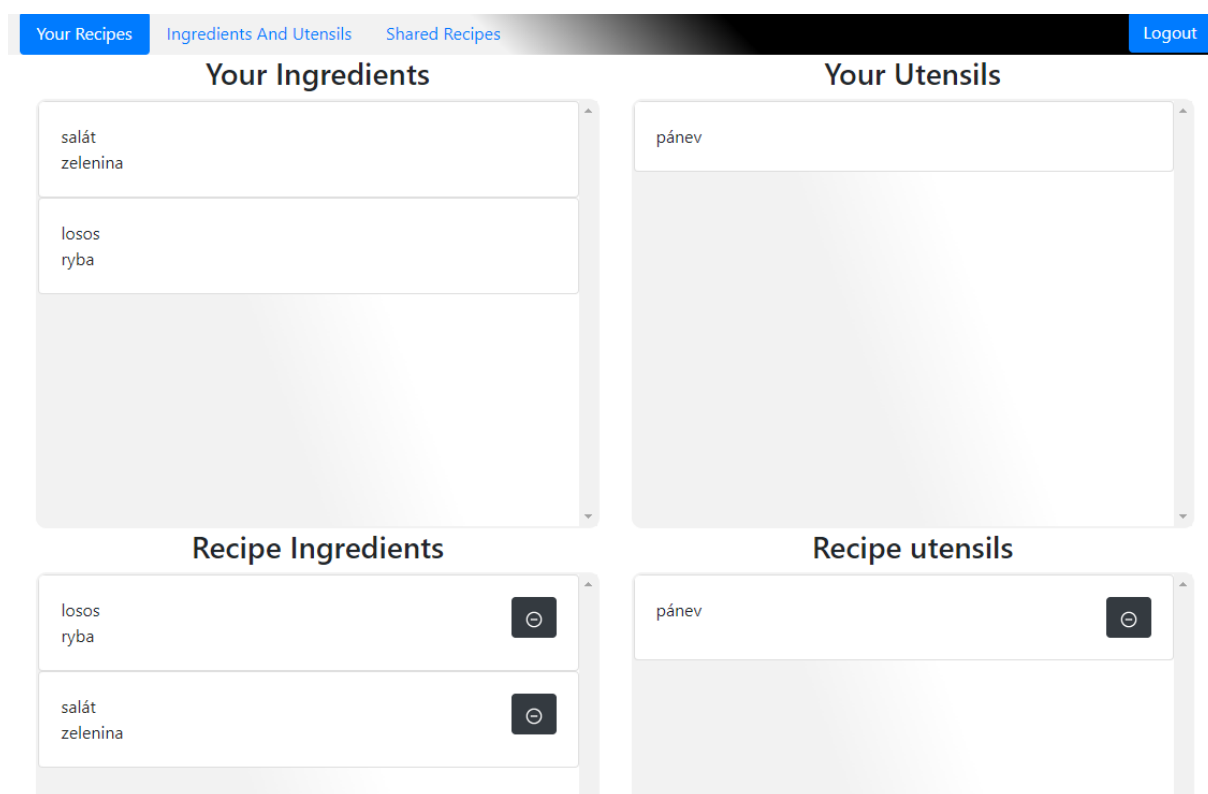
### 6.1 Tvorba receptu

Nejdříve uživatel musí zajít na hlavní stránku tam klikne na tlačítko přidat recept, to ho přenesse 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

## 6.2 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.

## 6.3 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.

## 6.4 Sdílení receptu

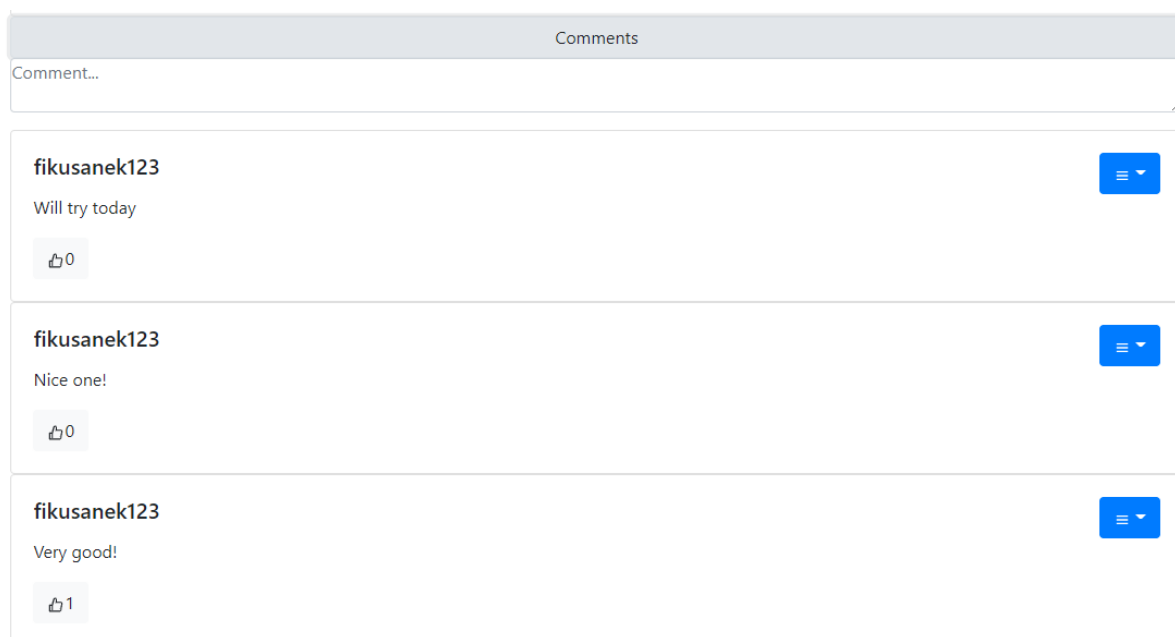
Na stránce osobního receptu je potřeba zmáchnout hamburgerové tlačítko možností a vybrat možnost sdílet.

## 6.5 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.

## 6.6 Vytváření komentářů

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



Obrázek 13 Komentáře receptu

## 7. 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.

## 8. Testování

### 8.1 Testování backendu

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

### 8.2 Testování databáze

Databázové příkazy jsem si testoval ve psql PostgreSQL příkazové řádce.

### 8.3 Testování frontendu

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

## 9. 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 nastavování package.json se rozestavení zdařilo.

# 10. Návod na spuštění

## 10.1 Aplikace na Heroku

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

## 10.2 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/utils. Je třeba též změnit název cesty serviceAccountu v server/utils/jwtGenerator.js.
  - a. Settings=>project settings=>aservice sccounts=>generate new private key To vám stáhne na počítač klíč ve formátu JSON
  - b. Přesuňte tento soubor do server/utils/
  - c. V souboru server/utils/jwtGenerator.js změňte v require('./{název souboru s klíčem}')

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

## 10.3 Lokální spuštění

Další možností je lokální instalace.

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

## 10.4 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/recipeIngredinetes'));
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

```

## 11. 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.

## 12. Seznam Obrázků

Obrázek 1 ER model mojí databáze.....	4
Obrázek 2 Stránka přihlášení.....	6
Obrázek 3 Stránka registrace.....	6
Obrázek 4 Hlavní stránka .....	7
Obrázek 5 Formulář základních dat receptu.....	8
Obrázek 6 Stránka vlastního receptu .....	9
Obrázek 7 Stránka ingredience a nástroje .....	10
Obrázek 8 Stránka sdílených receptů.....	11
Obrázek 9 Stránka sdíleného receptu.....	12
Obrázek 10 Hlavní stránka .....	13
Obrázek 11 Přesouvání ingrediencí a nástrojů.....	13
Obrázek 12 Formulář kroku receptu .....	14
Obrázek 13 Komentáře receptu.....	15
Obrázek 14 db.js.....	18
Obrázek 15 package.json client.....	20
Obrázek 16 index.js.....	19
Obrázek 17 package.json.....	21

## 13. 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>

