Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

E-pijaca

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Beograd, 2023.

Sadržaj

[1. Korisnički zahtev 1](#_Toc134447612)

[2. Opis sistema 2](#_Toc134447613)

[1. Opis slučajeva korišćenja 2](#_Toc134447614)

[1.1. SK1: Slučaj korišćenja – Dodavanje proizvoda 2](#_Toc134447615)

[1.2. SK2: Slučaj korišćenja – Pretraživanje proizvoda 3](#_Toc134447616)

[1.3. DS1: Slučaj korišćenja – Registracija 4](#_Toc134447617)

[2. Opis arhitekture aplikacije 5](#_Toc134447618)

[3. Opis procesa slučajeva korišćenja (dijagrami sekvenci) 6](#_Toc134447619)

[3.3. DS1: Dijagram sekvenci – Dodavanje proizvoda 6](#_Toc134447620)

[3.2 DS2: Dijagram sekvenci – Pretraživanje proizvoda 6](#_Toc134447621)

[3.2 DS3: Dijagram sekvenci – Registracija korisnika 8](#_Toc134447622)

[4. Model podataka 11](#_Toc134447623)

[5. Detaljna specifikacija REST API-ja 12](#_Toc134447624)

[3. Kratak opis tehnologija korišćenih u aplikaciji 14](#_Toc134447625)

[4. Korisničko uputstvo 15](#_Toc134447626)

[5. Prikaz reprezentativnih delova koda 21](#_Toc134447627)

[6. GitHub link ka repozitorijumu 21](#_Toc134447628)

[7. Reference 22](#_Toc134447629)

# Korisnički zahtev

Potrebno je kreirati veb aplikaciju koja simulira rad e-pijace. Korisnici ove aplikacije mogu da pregledaju širok asortiman proizvoda iz različitih kategorija, kao što su hrana, pića, slatkisi i slično. Pored pregleda proizvoda, korisnicima je dostupna i mogućnost pretrage po ključnim rečima, što im omogućava da pronađu željeni proizvod brzo i efikasno. Na stranicama proizvoda, korisnici mogu da vide detaljne informacije, slike proizvoda, ocene korisnika i cene proizvoda.

Korisnici imaju mogućnost kreiranja svog naloga na sajtu, što im omogućava bržu i efikasniju kupovinu. Takođe, za naše korisnike izdvojili smo informacije o receptima za jela sa proizvodima iz prodavnice, što im pruža dodatnu vrednost pri kupovini.

Administratori ove aplikacije imaju mogućnost da pregledaju narudžbine i statistiku prodaje, što im omogućava da bolje razumeju potrebe svojih korisnika i da planiraju dalje marketinške kampanje i prodajne strategije. Takođe, administratori imaju mogućnost da dodaju nove proizvode, ažuriraju postojeće i brišu nepotrebne proizvode, što im omogućava da efikasno upravljaju asortimanom proizvoda. Dodavanje novih proizvoda može se obaviti jednostavno i brzo, a administratori mogu da unesu sve potrebne informacije, uključujući naziv, opis, sliku, cenu i kategoriju. Ažuriranje proizvoda je takođe jednostavno, a administratori mogu da izmene bilo koju informaciju o proizvodu, uključujući cenu, opis, sliku i kategoriju. Brisanje proizvoda se može obaviti jednostavno i brzo, a administratori mogu da uklone proizvod sa sajta ako je proizvod prestao da se prodaje ili se više ne nalazi u ponudi.

# Opis sistema

## Opis slučajeva korišćenja

U ovoj aplikaciji, identifikovano je više slučajeva korišćenja, od kojih ćemo izdvojiti dva:

1. Dodavanje proizvoda
2. Pretraživanje proizvoda
3. Registracija

### 1.1. SK1: Slučaj korišćenja – Dodavanje proizvoda

Naziv SK: Dodavanje novog proizvoda u bazu

Aktori SK: Administrator

Učesnici SK: Administrator i sistem (program)

Preduslov: Sistem je uključen i administrator ima pristup funkcionalnosti za dodavanje novog proizvoda u bazu.

Osnovni scenario SK:

1. Administrator poziva sistem da otvori formu za dodavanje novog proizvoda u bazu. (APSO)
2. Sistem prikazuje formu za unos podataka o proizvodu. (IA)
3. Administrator unosi podatke o proizvodu u formu. (APUSO)
4. Administrator proverava da li je uneo ispravne podatke o proizvodu. (ANSO)
5. Administrator poziva sistem da sačuva podatke o proizvodu u bazu. (APSO)
6. Sistem proverava da li su uneti podaci ispravni i validni. (SO)
7. Sistem dodaje novi proizvod u bazu i prikazuje administratoru poruku: “Sistem je uspešno dodao novi proizvod u bazu”. (IA)

Alternativni scenario:

4.1 Ukoliko administrator nije uneo ispravne podatke o proizvodu, sistem prikazuje mu poruku o grešci i vraća ga na korak 2. (IA)

6.1 Ukoliko sistem utvrdi da su podaci neispravni, prikazuje se administratoru poruka o grešci i vraća ga na korak 2. (IA)

7.1 Ukoliko sistem ne može da sačuva podatke o proizvodu u bazu, prikazuje se administratoru poruka o grešci i vraća ga na korak 2. (IA)

### 1.2. SK2: Slučaj korišćenja – Pretraživanje proizvoda

Naziv SK:  
Pretraživanje proizvoda

Aktori SK:  
Korisnik

Učesnici SK:  
Korisnik i sistem (program)

Preduslov: Sistem je uključen i Korisnik je ulogovan pod svojom šifrom. Sistem prikazuje listu proizvoda.

Osnovni scenario SK

1. Korisnik unosi vrednost po kojoj pretražuje proizvode. (APUSO)
2. Korisnik poziva sistem da pronađe proizvode po zadatoj vrednosti. (APSO)
3. Sistem traži proizvode po zadatoj vrednosti. (SO)
4. Sistem prikazuje korisniku podatke o proizvodima i poruku: “Sistem je našao proizvode po zadatoj vrednosti”. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe proizvode on prikazuje korisniku poruku: “Sistem ne može da nađe proizvode po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)

### DS1: Slučaj korišćenja – Registracija

Naziv SK:  
Registracija korisnika

Aktori SK:  
Korisnik

Učesnici SK:  
Korisnik i sistem (program)

Preduslov: Sistem je uključen i prikazuje formu za registraciju. Učitana je lista svih korisnika.

Osnovni scenario SK

1. Korisnik poziva sistem da kreira nalog. (APSO)
2. Sistem kreira nalog. (SO)
3. Sistem prikazuje korisniku nalog i poruku: “Sistem je kreirao nalog“. (IA)
4. Korisnik unosi podatke u nalog. (APUSO)
5. Korisnik proverava da li je korektno uneo podatke u nalog. (ANSO)
6. Korisnik poziva sistem da zapamti podatke o nalogu. (APSO)
7. Sistem pamti podatke o nalogu. (SO)
8. Sistem prikazuje korisniku zapamćeni nalog i poruku: “Sistem je zapamtio nalog“. (IA)

Alternativna scenarija

3.1 Ukoliko sistem ne može da kreira nalog on prikazuje korisniku poruku: “Sistem ne može da kreira nalog”. Prekida se izvršenje scenarija. (IA)

8.1 Ukoliko sistem ne može da zapamti podatke o nalogu on prikazuje korisniku poruku “Sistem ne može da zapamti nalog”. (IA)

## Opis arhitekture aplikacije

Naša aplikacija je kreirana kroz nekoliko ključnih segmenata i komponenti koje čine njenu arhitekturu. U ovom poglavlju ćemo objasniti dizajn aplikacije.

Prva i najvažnija komponenta je front-end koji je izrađen uz pomoć React biblioteke. Ova komponenta omogućuje korisnicima da pregledaju proizvode, dodaju ih u korpu i obave kupovinu. Podaci koje korisnici unose na front-endu se šalju na back-end komponentu na dalju obradu.

Druga komponenta je back-end, koja je razvijena korišćenjem Laravel framework-a. Ova komponenta omogućuje administraciju proizvoda i narudžbina, kao i manipulaciju podacima u bazi podataka. Back-end komponenta se oslanja na ORM (Object-Relational Mapping) za komunikaciju sa bazom podataka, što omogućuje izvršavanje CRUD (Create, Read, Update, Delete) operacija nad podacima.

Treća i kritična komponenta je baza podataka, u kojoj se skladište svi podaci o proizvodima, narudžbinama, korisnicima i ostalim informacijama neophodnim za nesmetani rad aplikacije. Baza podataka je izrađena korišćenjem MySQL sistema za upravljanje bazama podataka.

Povezivanje full stack aplikacije i baze podataka se vrši korišćenjem PDO (PHP Data Objects) biblioteke koja omogućava povezivanje PHP aplikacije sa bazom podataka. Ovaj način povezivanja omogućava brzo i sigurno izvršavanje upita nad bazom podataka, kao i lako održavanje baze podataka.

## Opis procesa slučajeva korišćenja (dijagrami sekvenci)

### DS1: Dijagram sekvenci – Dodavanje proizvoda

Osnovni scenario SK

1. Administrator poziva sistem da otvori formu za dodavanje novog proizvoda u bazu. (APSO)
2. Sistem prikazuje formu za unos podataka o proizvodu. (IA)
3. Administrator poziva sistem da sačuva podatke o proizvodu u bazu. (APSO)
4. Sistem dodaje novi proizvod u bazu i prikazuje administratoru poruku: “Sistem je uspešno dodao novi proizvod u bazu”. (IA)

Slika na kojoj se nalazi dijagram

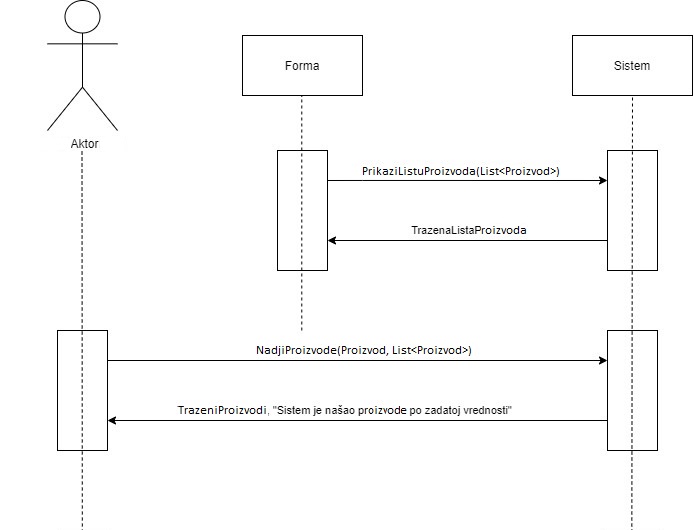
Opis je automatski generisan

Slika 1. Dijagram sekvenci – Dodavanje proizvoda

### 3.2 DS2: Dijagram sekvenci – Pretraživanje proizvoda

Osnovni scenario SK

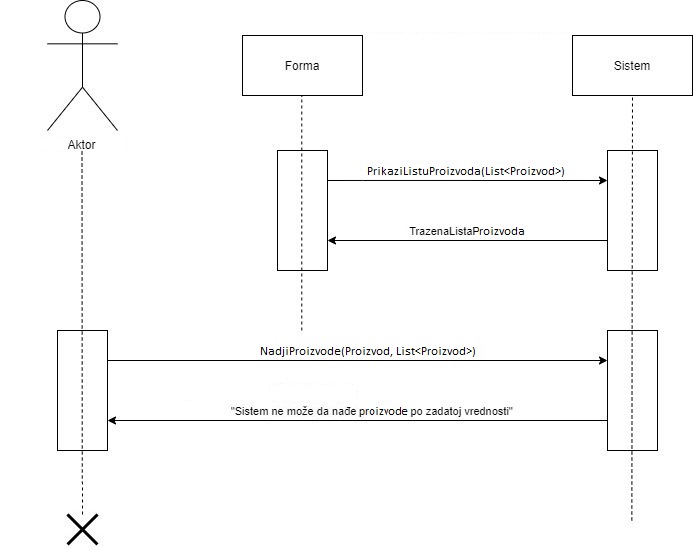
1. Forma poziva sistem da prikaže listu proizvoda. (APSO)
2. Sistem prikazuje formi listu proizvoda. (IA)
3. Korisnik poziva sistem da pronađe proizvode po zadatoj vrednosti. (APSO)
4. Sistem prikazuje korisniku podatke o proizvodima i poruku: “Sistem je našao proizvode po zadatoj vrednosti”. (IA)



Slika 2. Dijagram sekvenci – Pretraživanje proizvoda

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe proizvode on prikazuje korisniku poruku: “Sistem ne može da nađe proizvode po zadatoj vrednosti”. Prekida se izvršenje scenarija. (IA)



Slika 3. Dijagram sekvenci – Pretraživanje proizvoda/alternativni scenario 1

### 3.2 DS3: Dijagram sekvenci – Registracija korisnika

Osnovni scenario SK

1. Korisnik poziva sistem da kreira nalog. (APSO)
2. Sistem prikazuje korisniku nalog i poruku: “Sistem je kreirao nalog“. (IA)
3. Korisnik poziva sistem da zapamti podatke o nalogu. (APSO)
4. Sistem prikazuje korisnikuocu zapamćeni nalog i poruku: “Sistem je zapamtio nalog“. (IA)

Slika na kojoj se nalazi dijagram

Opis je automatski generisan

Slika 4. Dijagram sekvenci – Kreiranje naloga

Alternativna scenarija

* 1. Ukoliko sistem ne može da kreira nalog on prikazuje korisniku poruku: “Sistem ne može da kreira nalog”. Prekida se izvršenje scenarija. (IA)

Slika na kojoj se nalazi dijagram

Opis je automatski generisan

Slika 5. Dijagram sekvenci – Kreiranje naloga/alternativni scenario 1

4.1 Ukoliko sistem ne može da zapamti podatke o nalogu on prikazuje korisnikuocu poruku “Sistem ne može da zapamti nalog”. (IA)

## 4. Model podataka

Slika na kojoj se nalazi dijagram

Opis je automatski generisan

Slika 6. PMOV

## Detaljna specifikacija REST API-ja

Ovo je RESTful API (Application Programming Interface) koji služi za upravljanje proizvodima (products) i kategorijama (categories) proizvoda. API ima pet ruta za kontroler ProductController za upravljanje proizvodima, kao što su dodavanje, brisanje, ažuriranje i pregled.

Endpoints:

GET /products

Opis: Ova ruta vraća sve proizvode koji su dostupni u bazi podataka.

Response: JSON objekt koji sadrži popis svih proizvoda u bazi podataka. Svaki proizvod u popisu ima sledeća polja: id, name, description, price, image\_url i category\_id.

GET /products/{id}

Opis: Ova ruta vraća jedan proizvod s određenim id brojem.

Response: JSON objekat koji sadrži informacije o proizvodu s traženim id brojem. Polja su ista kao i u prethodnoj ruti.

POST /products

Opis: Ova ruta služi za dodavanje novog proizvoda u bazu podataka.

Request: JSON objekat koji sadrži informacije o novom proizvodu. Polja su: name, description, price, image\_url i category\_id.

Response: JSON objekat koji sadrži informacije o novom proizvodu, uključujući i id broj.

PUT /products/{id}

Opis: Ova ruta služi za ažuriranje postojećeg proizvoda u bazi podataka.

Request: JSON objekat koji sadrži polja koja će se ažurirati. Moguće su promene u poljima: name, price, image\_url i category\_id.

Response: JSON objekat koji sadrži informacije o ažuriranom proizvodu.

DELETE /products/{id}

Opis: Ova ruta služi za brisanje proizvoda iz baze podataka.

Response: HTTP status 204 (No Content) ako je proizvod uspješno obrisan.

GET /categories

Opis: Ova ruta vraća sve kategorije koje su dostupne u bazi podataka.

Response: JSON objekat koji sadrži popis svih kategorija u bazi podataka. Svaka kategorija u popisu ima sljedeća polja: id i name.

# Kratak opis tehnologija korišćenih u aplikaciji

JavaScript (JS) je popularni programski jezik koji se koristi za izradu interaktivnih web stranica i aplikacija. JS se izvršava na klijentskoj strani, što znači da se kod izvršava na strani korisnika, a ne na serveru. Zahvaljujući tome, JS može dodati interaktivne elemente na web stranicu, kao što su animacije, dinamičko ažuriranje sadržaja i drugo.

React je popularni JavaScript okvir za izradu korisničkog interfejsa (UI) na webu. React koristi JSX (ekstenziju sintakse za JavaScript), koja omogućava pisanje HTML i CSS kod u JS datotekama. React se koristi za izradu jednostraničnih aplikacija (SPA) i dinamičkih UI elemenata, kao što su drveća, grafovi i interaktivni elementi.

PHP je server-side programski jezik koji se često koristi za izradu web aplikacija. PHP je vrlo popularan i lako dostupan jezik koji ima mnogo ugrađenih funkcija i alata za izradu web aplikacija. PHP se izvršava na serveru, što znači da se kod izvršava na strani servera prije nego što se šalje klijentu. PHP se koristi za obradu formi, stvaranje sesija, slanje e-pošte i drugo.

Laravel je popularni PHP okvir za izradu web aplikacija. Laravel pruža mnoge ugrađene funkcije i alate za brzo i jednostavno razvijanje aplikacija. Laravel se koristi za izradu aplikacija s bazom podataka, autentikaciju korisnika, rutiranje i slanje e-pošte.

CSS (Cascading Style Sheets) je kaskadni stilski jezik koji se koristi za definisanje izgleda i stilova na web stranicama. CSS se koristi za definisanje boja, fontova, pozadina, margina i drugih vizualnih elemenata na web stranicama.

HTML (Hypertext Markup Language) je osnovni jezik za strukturiranje i prikazivanje sadržaja na web stranicama. HTML se koristi za definisanje strukture web stranice, kao što su zaglavlje, tijelo i podnožje, kao i za dodavanje teksta, slika, formi i drugih elemenata.

Bootstrap je popularni front-end okvir za razvoj responsivnih web stranica i aplikacija. Bootstrap se sastoji od predložaka HTML i CSS koda, koji se mogu koristiti za brzo i jednostavno dizajniranje i stiliziranje web stranica. Bootstrap također uključuje JavaScript biblioteku koja pruža dodatne funkcije, kao što su skočni prozori, prikazivanje modala i drugo.

Axios je popularna JavaScript biblioteka za slanje HTTP zahteva sa klijenta na server. Koristi se u mnogim web aplikacijama, uključujući i React aplikacije, za pozivanje API-ja i manipulaciju podacima. Axios ima mnoge korisne funkcije, uključujući automatsko pretvaranje JSON odgovora u JavaScript objekte, podešavanje različitih opcija zahteva, rukovanje greškama, kao i napredne mogućnosti korišćenja Promise i async/await funkcija.

# Korisničko uputstvo

Pri ulasku na aplikacju čeka nas početna strana



Slika 7. Pocetna stranica

Tu vidimo navigacioni meni. Ta komponenta se prikazuje na svim stranicama sajta. Zatim vidimo naslovnu sliku kao I kratak reklamni tekst.

Slika na kojoj se nalazi Vebsajt

Opis je automatski generisan

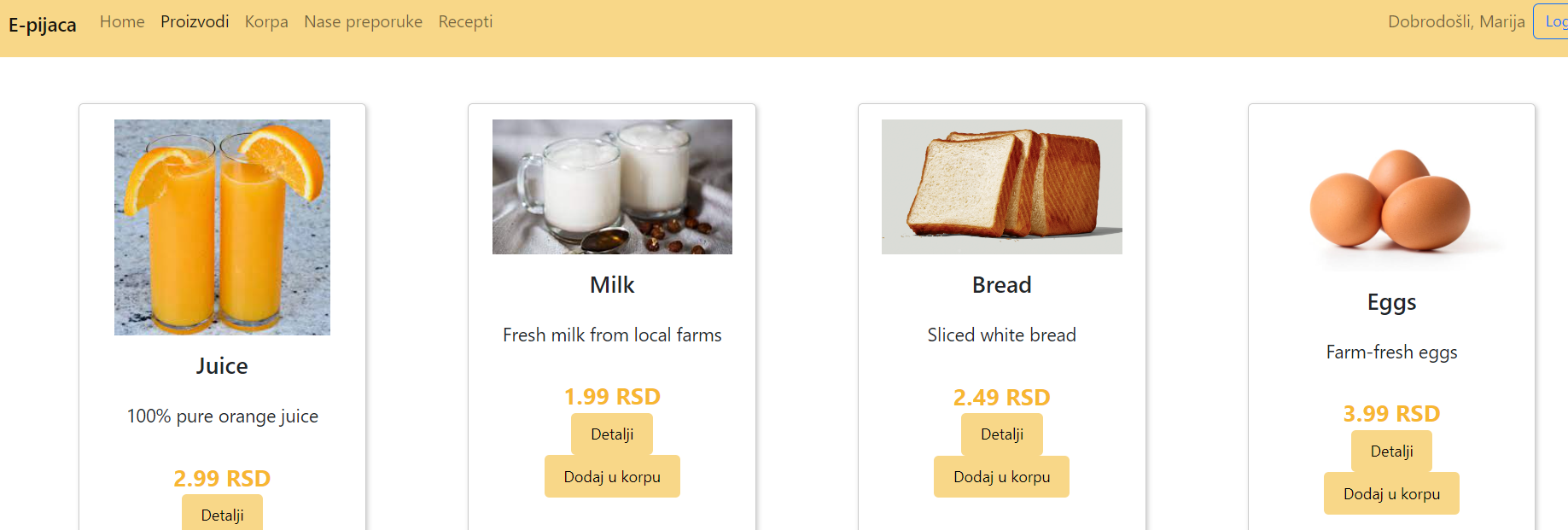
Slika 8. LoginPage

Slika na kojoj se nalazi Vebsajt

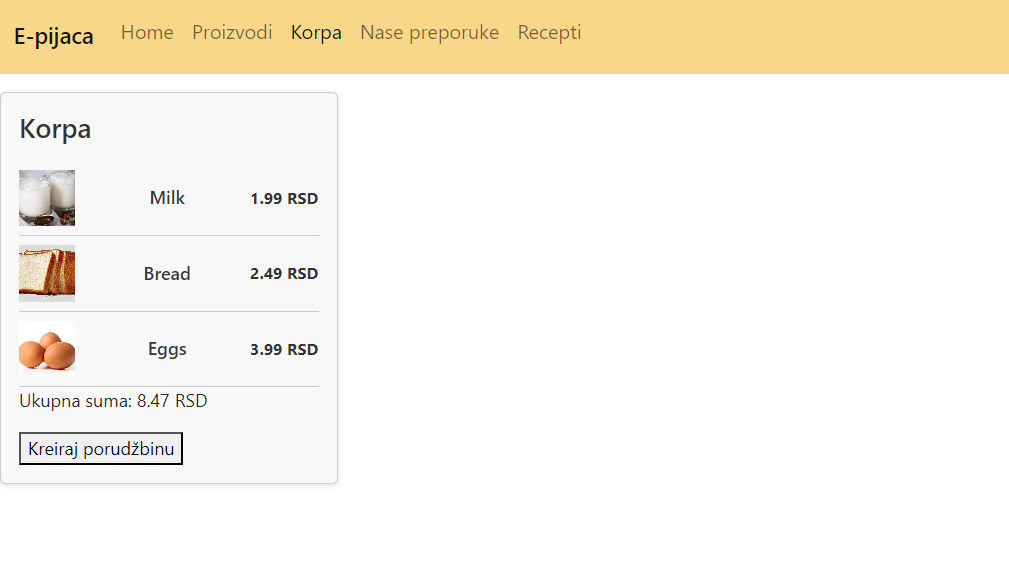
Opis je automatski generisan

Slika 9. RegisterPage

Nakon što se korisnik uloguje na sistem, njemu se prikazuju svi proizvodi koje imamo u ponudi. Ovde korisnik može da filtrira proizvode, da ih dodaje u korpu I da ih prikazuje u različitim valutama.



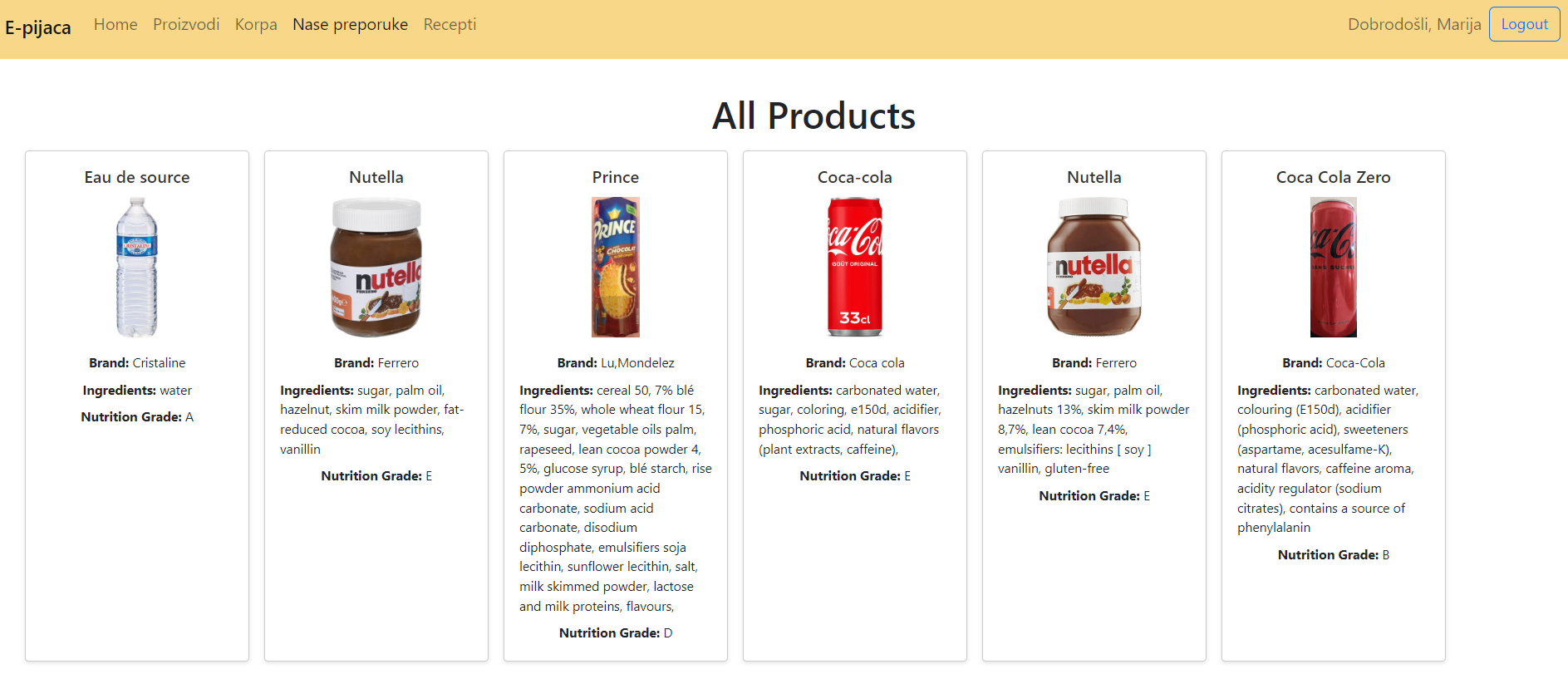
Slika 10. Ponuda



Slika 11. Korpa

Na stranici na kojoj je prikazana korpa se omogućava korisniku da potvrdi kupovinu I time se kreira porudžbina u bazi. Administrator kasnije može da obradi porudžbinu.

Za naše korisnike izdvojili smo proizvode koje preporučujemo, kao I zdrave recepte koje mogu da prave korišćenjem omiljenih proizvoda.



Slika 12 preporuke

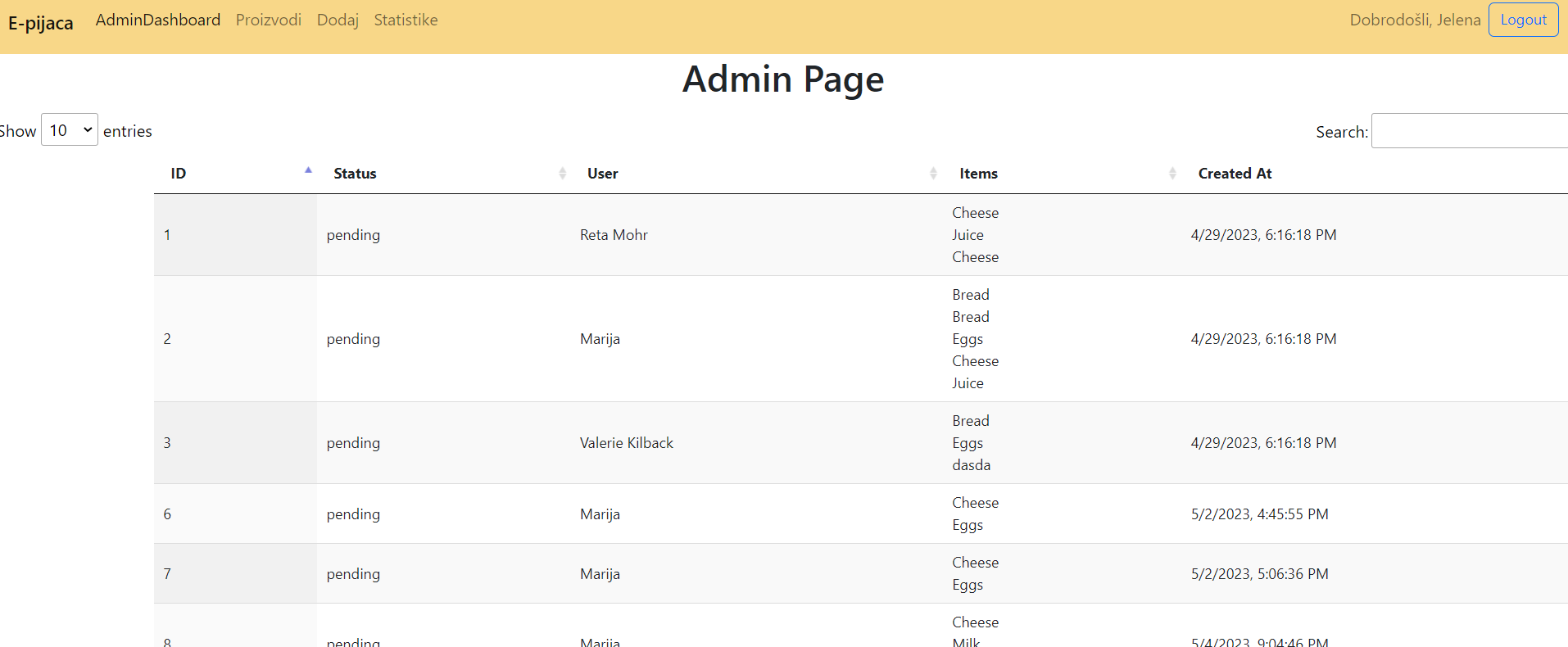
Slika na kojoj se nalazi tekst

Opis je automatski generisan

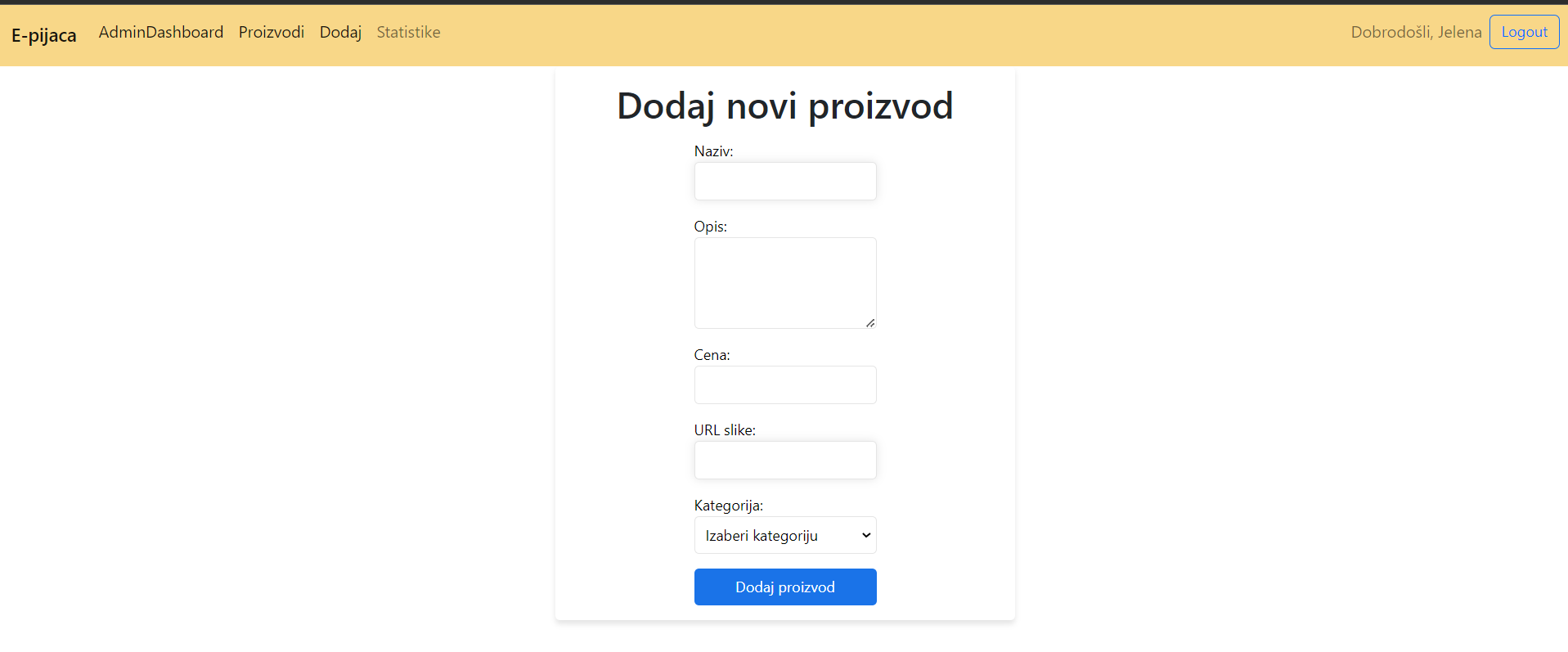
Slika 13 recepti

Radi lakšeg snalaženja, korisnici mogu da filtriraju recepte po nazivu, kategoriji i poreklu.

Kada se uloguje administrator, njemu se prikazuje stranica na kojoj može da vidi sve porudžbine koje su kreirane.



Slika 14. Admin dashboard

Odabirom opcije Dodaj se administrator u otvara forma preko koje može da unese podatke o novom proizvodu. 

Slika 15 dodaj novi proizvod

Klikom na opciju Proizvodi se admnistratoru prikazuje stranica na kojoj su preko DataTable tabele prikazani svi proizvodi koje imamo u ponudi. Ovde korisnik može da vrši izmenu ili brisanje proizvoda. Radi lakšeg snalaženja administrator je omogućeno filtriranje, sortiranje kao I paginacija proizvoda.

Slika na kojoj se nalazi tekst

Opis je automatski generisan

Slika 16 manipulacija nad proizvodom

Klikom na opciju Statistike, korisniku se otvara sledeća stranica

Slika na kojoj se nalazi tekst

Opis je automatski generisan

Slika na kojoj se nalazi grafikon, kružni grafikon

Opis je automatski generisan

Slika 17 statistike

# Prikaz reprezentativnih delova koda

<?php

use App\Http\Controllers\AuthController;

use App\Http\Controllers\OrderController;

use App\Http\Controllers\ProductController;

use App\Http\Controllers\ReviewController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| API Routes

|--------------------------------------------------------------------------

|

| Here is where you can register API routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| is assigned the "api" middleware group. Enjoy building your API!

|

\*/

// Rute za kontroler ProductController

Route::get('/products', [ProductController::class, 'index']);

Route::get('/products/{id}', [ProductController::class, 'show']);

Route::post('/products', [ProductController::class, 'store']);

Route::put('/products/{id}', [ProductController::class, 'update']);

Route::delete('/products/{id}', [ProductController::class, 'destroy']);

// Rute za kontroler OrderController

Route::get('/orders', [OrderController::class, 'index']);

Route::get('/orders/{id}', [OrderController::class, 'show']);

Route::post('/orders', [OrderController::class, 'store']);

Route::put('/orders/{id}', [OrderController::class, 'update']);

Route::delete('/orders/{id}', [OrderController::class, 'destroy']);

// Rute za kontroler ReviewController

Route::get('/reviews', [ReviewController::class, 'index']);

Route::get('/reviews/{id}', [ReviewController::class, 'show']);

Route::post('/reviews', [ReviewController::class, 'store']);

Route::put('/reviews/{id}', [ReviewController::class, 'update']);

Route::delete('/reviews/{id}', [ReviewController::class, 'destroy']);

// Registracija korisnika

Route::post('/register', [AuthController::class, 'register']);

// Prijava korisnika

Route::post('/login', [AuthController::class, 'login']);

Route::group(['middleware' => ['auth:sanctum']], function () {  //obicni ulogovani korisnici

    Route::post('/logout', [AuthController::class, 'logout']);

});

Route::middleware(['auth:sanctum','isAPIAdmin'])->group(function(){ //ako je ulogovan admin

    Route::post('/logout', [AuthController::class, 'logout']);

});

<?php

namespace App\Http\Controllers;

use App\Models\Product;

use Illuminate\Http\Request;

use App\Http\Resources\ProductResource;

use Illuminate\Support\Facades\Validator;

class ProductController extends Controller

{

    public function index()

    {

        $products = Product::all();

        return ProductResource::collection($products);

    }

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'name' => 'required|max:255',

            'description' => 'required',

            'price' => 'required|numeric',

            'image\_url' => 'required|string',

            'category\_id' => 'required|exists:categories,id',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $product = Product::create($request->all());

        return new ProductResource($product);

    }

    public function show($id)

    {

        $product = Product::findOrFail($id);

        return new ProductResource($product);

    }

    public function update(Request $request, $id)

    {

        $product = Product::findOrFail($id);

        $validator = Validator::make($request->all(), [

            'name' => 'max:255',

            'price' => 'numeric',

            'image\_url' => 'url',

            'category\_id' => 'exists:categories,id',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $product->update($request->all());

        return new ProductResource($product);

    }

    public function destroy($id)

    {

        $product = Product::findOrFail($id);

        $product->delete();

        return response()->json(null, 204);

    }

}

<?php

namespace App\Http\Controllers;

use App\Models\Order;

use Illuminate\Http\Request;

use App\Http\Resources\OrderResource;

use Illuminate\Support\Facades\Validator;

class OrderController extends Controller

{

    public function index()

    {

        $orders = Order::all();

        return OrderResource::collection($orders);

    }

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'status' => 'required',

            'user\_id' => 'required|exists:users,id',

            'items' => 'required',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $order = Order::create($request->all());

        return new OrderResource($order);

    }

    public function show($id)

    {

        $order = Order::findOrFail($id);

        return new OrderResource($order);

    }

    public function update(Request $request, $id)

    {

        $order = Order::findOrFail($id);

        $validator = Validator::make($request->all(), [

            'status' => 'max:255',

            'user\_id' => 'exists:users,id',

            'items' => 'json',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $order->update($request->all());

        return new OrderResource($order);

    }

    public function destroy($id)

    {

        $order = Order::findOrFail($id);

        $order->delete();

        return response()->json(null, 204);

    }

}

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class AuthController extends Controller

{

    public function register(Request $request)

    {

        $validator = Validator::make(

            $request->all(),

            [

                'password' =>  'required' ,

                'name' => 'required|string|max:100',

                'email' => 'required|string|max:100|email',

            ]

        );

        if ($validator->fails())

            return response()->json($validator->errors());

        $user = User::create([

            'name' => $request->name,

            'email' => $request->email,

            'password' => Hash::make($request->password)]);

        $token = $user->createToken('auth\_token')->plainTextToken;

        return response()->json(['data' => $user, 'acess\_token' => $token, 'token\_type' => 'Bearer']);

    }

    public function login(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'email' => 'required|email',

            'password' => 'required|string',

        ]);

        if ($validator->fails()) {

            return response()->json(['success'=>false]);

        }

        if (!Auth::attempt($request->only('email', 'password'))) {

            return response()->json([

                'status'=>401,

                'message'=>'Invalid credentials',

            ]);

        }

        $user = User::where('email', $request['email'])->firstOrFail();

        if(!$user){

            return response()->json([

                'status'=>401,

                'message'=>'Invalid credentials',

            ]);

        }else{

            if($user->admin==1){

                $role='admin';

                $token = $user->createToken($user->email.'\_AdminToken',['server:admin'])->plainTextToken;

            }else{

                $role=' ';

                $token = $user->createToken($user->email.'\_Token',[''])->plainTextToken;

            }

        }

       // $token = $user->createToken('LoginToken')->plainTextToken;

        $response = [

            'status'=>200,

            'username'=>$user->name,

            'token' => $token,

            'role'=> $role,

            'id'=>$user->id

        ];

        return response()->json($response);

    }

    public function logout()

    {

        auth()->user()->tokens()->delete();

        return response()->json(['Uspešno ste se izlogovali!']);

    }

}

import { useEffect, useState } from 'react';

import './App.css';

import DetaljiProizvoda from './DetaljiProizvoda';

import Home from './Home';

import Katalog from './Katalog';

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import axios from 'axios';

import Login from './Login';

import Register from './Register';

import Navbar from './Navbar';

import 'bootstrap/dist/css/bootstrap.min.css';

import AdminPage from './AdminPage';

import ProductList from './ProductList';

import AddProduct from './AddProduct';

import EditProduct from './EditProduct';

import NasePreporuke from './NasePreporuke';

import Korpa from './Korpa';

import Recepti from './Recepti';

import AdminPageStats from './AdminPageStats';

function App() {

  const [user, setUser] = useState(null);

  const [korpa, setKorpa] = useState([]);

  const [orders, setOrders] = useState([]);

  const fetchOrders = async () => {

    try {

      const response = await axios.get("http://127.0.0.1:8000/api/orders");

      console.log(response.data)

      setOrders(response.data.data);

    } catch (error) {

      console.error("Error fetching orders:", error);

    }

  };

  useEffect(() => {

    fetchOrders();

  }, []);

  const handleLogin = data => {

    setUser(data);

    console.log("HANDLE LOGIN"+data)

  };

  const handleLogout = () => {

    setUser(null);

  };

  const [proizvodi, setProizvodi] = useState([]);

  useEffect(() => {

    axios

      .get("http://127.0.0.1:8000/api/products")

      .then((response) => {

        setProizvodi(response.data.data);

      })

      .catch((error) => {

        console.log(error);

      });

  }, []);

  const [ocene, setOcene] = useState([]);

  useEffect(() => {

    axios

      .get("http://127.0.0.1:8000/api/reviews")

      .then((response) => {

        setOcene(response.data.data);

      })

      .catch((error) => {

        console.log(error);

      });

  }, []);

    function onAdd(newProduct){

      let products = proizvodi;

      products.push(newProduct)

      setProizvodi(products)

    }

    function onUpdate(newProduct){

      let products = proizvodi;

      for(var i=0;i<proizvodi.length;i++){

        if(proizvodi[i].id==newProduct.id){

          proizvodi[i]=newProduct;

        }

      }

      setProizvodi(products)

    }

  return (

    <BrowserRouter>

        <Navbar user={user} onLogout={handleLogout} />

      <Routes>

        <Route path="/" element={ <Home></Home>}></Route>

        <Route path="/preporuke" element={ <NasePreporuke></NasePreporuke>}></Route>

        <Route path="/korpa" element={ <Korpa korpa={korpa} setKorpa={setKorpa}></Korpa>}></Route>

        <Route path="/recepti" element={ <Recepti></Recepti>}></Route>

        <Route path="/proizvodi" element={<Katalog proizvodi={proizvodi} korpa={korpa} setKorpa={setKorpa}></Katalog>}></Route>

        <Route path="/proizvodi/:id" element={ <DetaljiProizvoda proizvodi={proizvodi}></DetaljiProizvoda>}></Route>

        <Route path="/login" element={ <Login onLogin={handleLogin}></Login>}></Route>

        <Route path="/register" element={ <Register></Register>}></Route>

        <Route path="/admin" element={ <AdminPage proizvodi={proizvodi} orders={orders}></AdminPage>} ></Route>

        <Route path="/admin/products" element={<ProductList products={proizvodi} setProducts={setProizvodi}></ProductList>}></Route>

        <Route path="/admin/products/add" element={<AddProduct onAdd={onAdd} ></AddProduct>}></Route>

        <Route path="/admin/stats" element={ <AdminPageStats  ocene={ocene} proizvodi={proizvodi} narudzbine={orders}></AdminPageStats>} ></Route>

        <Route path="/admin/products/edit/:id" element={ <EditProduct proizvodi={proizvodi} onUpdate={onUpdate}></EditProduct>}></Route>

      </Routes>

  </BrowserRouter>

  );

}

export default App;

import React from 'react';

import { Link, NavLink, useNavigate } from 'react-router-dom';

import axios from 'axios';

const Navbar = ({ user, onLogout }) => {

    let navigate=useNavigate();

    const handleLogout = () => {

        axios.post('http://127.0.0.1:8000/api/logout', null, {

            headers: {

                Authorization: `Bearer ${window.sessionStorage.getItem("auth\_token")}`,

                Accept: 'application/json',

                'Content-Type': 'application/json',

            },

        })

        .then(response => {

            onLogout();

            navigate("/");

        })

        .catch(error => console.error(error));

    };

  return (

    <nav className="navbar navbar-expand-lg">

      <div className="container-fluid">

        <Link to="/" className="navbar-brand">E-pijaca</Link>

        <button className="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

          <span className="navbar-toggler-icon"></span>

        </button>

        <div className="collapse navbar-collapse" id="navbarNav">

          <ul className="navbar-nav me-auto">

            {user && user.role=='admin' && (

                <><li className="nav-item">

                    <NavLink to="/admin" className="nav-link" activeClassName="active">AdminDashboard</NavLink>

                </li>

                <li className="nav-item">

                <   NavLink to="/admin/products" className="nav-link" activeClassName="active">Proizvodi</NavLink>

                </li>

                <li className="nav-item">

                <   NavLink to="/admin/products/add" className="nav-link" activeClassName="active">Dodaj</NavLink>

                </li>

                <li className="nav-item">

                <   NavLink to="/admin/stats" className="nav-link" activeClassName="active">Statistike</NavLink>

                </li>

                </>

            )}

            { user &&  user.role!='admin' &&

                (<><li className="nav-item">

                <NavLink to="/" className="nav-link" activeClassName="active" end>Home</NavLink>

                </li>

                <li className="nav-item">

                <NavLink to="/proizvodi" className="nav-link" activeClassName="active">Proizvodi</NavLink>

                </li>

                <li className="nav-item">

                <NavLink to="/korpa" className="nav-link" activeClassName="active">Korpa</NavLink>

                </li>

                <li className="nav-item">

                <NavLink to="/preporuke" className="nav-link" activeClassName="active" end>Nase preporuke</NavLink>

                </li>

                <li className="nav-item">

                <NavLink to="/recepti" className="nav-link" activeClassName="active">Recepti</NavLink>

                </li>

                </>)

            }

          </ul>

          <ul className="navbar-nav">

            {user ? (

              <>

                <li className="nav-item">

                  <span className="nav-link">Dobrodošli, {user.username}</span>

                </li>

                <li className="nav-item">

                  <button className="btn btn-outline-primary" onClick={handleLogout}>Logout</button>

                </li>

              </>

            ) : (

              <>

                <li className="nav-item">

                  <NavLink to="/login" className="nav-link" activeClassName="active">Login</NavLink>

                </li>

                <li className="nav-item">

                  <NavLink to="/register" className="nav-link" activeClassName="active">Register</NavLink>

                </li>

              </>

            )}

          </ul>

        </div>

      </div>

    </nav>

  );

};

export default Navbar;

import React, { useEffect, useState } from "react";

import { useParams } from "react-router-dom";

const DetaljiProizvoda = ({ proizvodi }) => {

  const [proizvod, setProizvod] = useState(null);

  const { id } = useParams();

  useEffect(() => {

    const pronadjiProizvod = () => {

      const pronadjeniProizvod = proizvodi.find((proizvod) => proizvod.id == id);

      setProizvod(pronadjeniProizvod);

    };

    pronadjiProizvod();

  }, [id, proizvodi]);

  if (!id || proizvod == null) {

    return <div>Loading...</div>;

  }

  function average\_review() {

    var sum = 0;

    var counter = 0;

    for (let i = 0; i < proizvod.reviews.length; i++) {

      sum += proizvod.reviews[i].rating;

      counter++;

    }

    return (sum / counter).toFixed(2);

  }

  function renderStars(rating) {

    const roundedRating = Math.round(rating);

    let stars = [];

    for (let i = 1; i <= 5; i++) {

      stars.push(

        <span key={i} className={`star ${i <= roundedRating ? "filled" : ""}`}>

          ★

        </span>

      );

    }

    return stars;

  }

  return (

    <div className="detalji-proizvoda">

      <img src={proizvod.image\_url} alt={proizvod.name} />

      <h2>{proizvod.name}</h2>

      <p>{proizvod.description}</p>

      <div className="proizvod-info">

        <div className="proizvod-info-item">

          <span className="label">Kategorija:</span>

          <span className="value">{proizvod.category}</span>

        </div>

        <div className="proizvod-info-item">

          <span className="label">Cena:</span>

          <span className="value">{proizvod.price} RSD</span>

        </div>

        <div className="proizvod-info-item">

          <span className="label">Ocene:</span>

          <span className="value">{renderStars(average\_review())}</span>

        </div>

      </div>

      <h3>Recenzije korisnika</h3>

      {proizvod.reviews.map((review) => (

        <div className="review" key={review.id}>

          <h4>{review.title}</h4>

          <p>{review.body}</p>

          <div className="review-info">

            <div className="review-info-item">

              <span className="label">Korisnik:</span>

              <span className="value">{review.user.name}</span>

            </div>

            <div className="review-info-item">

              <span className="label">Ocena:</span>

              <span className="value">{review.rating}/5</span>

            </div>

          </div>

        </div>

      ))}

    </div>

  );

};

export default DetaljiProizvoda;

import React, { useState, useEffect } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

const Katalog = ({proizvodi,setKorpa,korpa}) => {

  const [prikaziDetalje, setPrikaziDetalje] = useState(false);

  const navigate = useNavigate();

  const handleDetaljiClick = (id) => {

    navigate(`/proizvodi/${id}`);

  };

  const toggleProizvodUKorpi = (proizvod) => {

    const proizvodUKorpi = korpa.find((item) => item.id === proizvod.id);

    if (proizvodUKorpi) {

      setKorpa(korpa.filter((item) => item.id !== proizvod.id));

    } else {

      setKorpa([...korpa, proizvod]);

    }

  };

  return (

    <div className="katalog">

      {proizvodi.map((proizvod) => (

        <div className="proizvod" key={proizvod.id}>

          <img src={proizvod.image\_url} alt={proizvod.name} />

          <h3>{proizvod.name}</h3>

          <p>{proizvod.description}</p>

          <span className="cena">{proizvod.price} RSD</span>

          <button className="dugme-katalog" onClick={() => handleDetaljiClick(proizvod.id)}>

              Detalji

            </button>

            <button

              className="dugme-katalog"

              onClick={() => toggleProizvodUKorpi(proizvod)}

            >

              {korpa.find((item) => item.id === proizvod.id)

                ? "Izbaci iz korpe"

                : "Dodaj u korpu"}

            </button>

        </div>

      ))}

    </div>

  );

};

export default Katalog;

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useParams, useNavigate } from 'react-router-dom';

const EditProduct = ({proizvodi,onUpdate}) => {

  const { id } = useParams();

  const navigate = useNavigate();

  const [product, setProduct] = useState(null);

  useEffect(() => {

    const getProduct = async () => {

        for(var i=0;i<proizvodi.length;i++){

            if(proizvodi[i].id==id){

                setProduct(proizvodi[i])

            }

        }

        console.log(product)

    };

    getProduct();

  }, [id]);

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      let result=await axios.put(`http://127.0.0.1:8000/api/products/${id}`, product);

      console.log(result.data.data)

      onUpdate(result.data.data)

      alert('Proizvod uspešno ažuriran!');

      navigate('/admin/products');

    } catch (error) {

      console.error('Error updating product:', error);

    }

  };

  if (!product) {

    return <div>Loading...</div>;

  }

  return (

    <div className="edit-form-container">

      <h1>Uredi proizvod</h1>

      <form onSubmit={handleSubmit} className="edit-form">

        <label>

          Ime:

          <input

            type="text"

            value={product.name}

            onChange={(e) => setProduct({ ...product, name: e.target.value })}

            required

          />

        </label>

        <br />

        <label>

          Opis:

          <textarea

            value={product.description}

            onChange={(e) => setProduct({ ...product, description: e.target.value })}

            required

          />

        </label>

        <br />

        <label>

          Cena:

          <input

            type="number"

            value={product.price}

            onChange={(e) => setProduct({ ...product, price: e.target.value })}

            required

          />

        </label>

        <br />

        <label>

          URL slike:

          <input

            type="text"

            value={product.image\_url}

            onChange={(e) => setProduct({ ...product, image\_url: e.target.value })}

            required

          />

        </label>

        <br />

        <label>

          Kategorija:

          <select

            value={product.category\_id}

            onChange={(e) => setProduct({ ...product, category\_id: e.target.value })}

            required

          >

            <option value="">Izaberi kategoriju</option>

            <option value="1">Hrana</option>

            <option value="2">Piće</option>

            <option value="3">Slatkiši</option>

          </select>

        </label>

        <br />

        <button type="submit">Ažuriraj proizvod</button>

      </form>

    </div>

  );

};

export default EditProduct;

import React, { useState } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

const AddProduct = ({onAdd}) => {

    const [name, setName] = useState('');

    const [description, setDescription] = useState('');

    const [price, setPrice] = useState('');

    const [imageUrl, setImageUrl] = useState('');

    const [category, setCategory] = useState('');

    let navigate = useNavigate();

    const handleSubmit = async (e) => {

        e.preventDefault();

        try {

          const newProduct = {

            name,

            description,

            price,

            image\_url: imageUrl,

            category\_id: category,

          };

        let response= await axios.post('http://127.0.0.1:8000/api/products', newProduct);

        console.log(response.data.data)

          alert('Proizvod uspešno dodat!');

          onAdd(response.data.data)

          navigate('/admin/products')

        } catch (error) {

          console.error('Greška prilikom dodavanja proizvoda:', error);

        }

      };

      return (

        <div className="add-product-container">

          <h1>Dodaj novi proizvod</h1>

          <form onSubmit={handleSubmit}>

            <label>

              Naziv:

              <input type="text" value={name} onChange={(e) => setName(e.target.value)} required />

            </label>

            <label>

              Opis:

              <textarea value={description} onChange={(e) => setDescription(e.target.value)} required />

            </label>

            <label>

              Cena:

              <input type="number" value={price} onChange={(e) => setPrice(e.target.value)} required />

            </label>

            <label>

              URL slike:

              <input type="text" value={imageUrl} onChange={(e) => setImageUrl(e.target.value)} required />

            </label>

            <label>

              Kategorija:

              <select value={category} onChange={(e) => setCategory(e.target.value)} required>

                <option value="">Izaberi kategoriju</option>

                <option value="1">Hrana</option>

                <option value="2">Piće</option>

                <option value="3">Slatkiši</option>

              </select>

            </label>

            <button type="submit">Dodaj proizvod</button>

          </form>

        </div>

      );

};

export default AddProduct;

# GitHub link ka repozitorijumu

<https://github.com/marija141414>

# Reference

* Vlajić S, „Projektovanje softvera“, FON, 2020.
* <https://www.w3.org/standards/webdesign/htmlcss>
* <https://www.mysql.com/>
* Lazarević B, Marjanović Z, Aničić N, Babarogić S, „Baze podataka“, FON.