

**PANEVROPSKI UNIVERZITET APEIRON
FAKULTET INFORMACIONIH TEHNOLOGIJA**

Redovne studije

Smjer „Programiranje i softversko inženjerstvo”

**„PROJEKTOVANJE I REALIZACIJA
MOBILNE APLIKACIJE KORIŠĆENJEM
NERELACIONIH BAZA PODATAKA”**

(diplomski rad)

Mentor

Doc. dr Dražen Marinković

Student

Jovo Simić

Index br: 260-19/RITPS

Banja Luka, oktobar 2021.

Sadržaj

Skraćenice i akronimi.....	4
1. Uvod.....	5
2. Stack.....	6
2.1. React Native.....	6
2.2. ExpressJS.....	6
2.3. MongoDB.....	7
3. Razvojni proces.....	8
3.1 Backend.....	8
3.1.1. prvi komit - intial commit.....	8
3.1.2. podešavanje baze - setup database.....	8
3.1.3. podešavanje osnove za rute i šeme - set up foundation for routes and schemas.....	9
3.1.4. podešavanje šeme za produkte i kategorije - set up product and category schema. . .	12
3.1.5. završavanje rute i šema za produkte i kateogrije - products and categories routes and schemas finished.....	13
3.1.6. korisnik model, rute i JWT autentikacija - users model, routes and jwt auth.....	15
3.1.7. dodavanje order i orderitem šema - added order and orderitems schema.....	18
3.1.8. završavanje order ruta - finished orders routes.....	20
3.1.9. dodavanje prenosa fajlova - added file upload.....	22
3.1.10. ispravka, dodavanje parametra id na put request u products.js - forgot to add params id on put request for products.js.....	24
3.1.11. dodavanje mogućnosti za obične korisnike da koriste narudžbe - added ability for regular users to use and post their orders.....	24
3.2. Frontend.....	25
3.2.1. Kreiranje nove expo aplikacije - Created a new Expo app.....	25
3.2.2. dodavanje productcontainer, list, card i header komponenti - added productcontainer, list, card and header.....	25
3.2.3. dodavanje polje za pretragu i funkcionalnost - added search field and functionality	28
3.2.4. dodavanje banner sa istaknutim produktima - added banner.....	31
3.2.5. dodavanje kategorija - added categories.....	33
3.2.6. dodavanje navigacije - added navigation.....	36
3.2.7. dodavanje pojedinačnih produkta, navigacija i komponente - added single product navigation and components.....	38
3.2.8. dodavanje redux i podešavanje akcija i statea za korpu - added redux and set up cart state and actions.....	40
3.2.9. dodavanje funkcionalnosti korpi - added functionality to cart.....	42
3.2.10. završavanje checkout stranice - finished checkout page.....	48
3.2.11. spajanje frontenda sa backendom - connected frontend with backend.....	54
3.2.12. dodavanje logina i registracije za korisnike i dodavanje toasta za produkte i login - added user login, register and added toast for adding products and login.....	56
3.2.13. dodavanje funkcionalnosti za registrovanje i login, dodavanje profila korisnika - added functionality to register and login, added user profile page.....	62
3.2.14. dodavanje admin panela i produkt liste u njemu - added admin panel and product list in it.....	67

3.2.15. dodavanje stilizovanih dugmadi i semafora za dostupnost - added styled buttons and traffic light for availability.....	74
3.2.16. dodavanje mogućnosti za dodavanje, uređivanje i brisanje produkta iz admin panela - added ability to add, edit and delete items from admin panel.....	76
3.2.17. dodavanje mogućnosti za brisanje i dodavanje kategorija iz admin panela - added ability to add and delete categories in admin panel.....	81
3.2.18. dodavanje narudžbi u admin panel i korisničkom profilu - added orders to admin panel and user profile.....	83
4. Izled aplikacije.....	86
4.1. Početna strana.....	86
4.2. Search bar.....	87
4.3. Pojedinačni produkt.....	88
4.4. Korisnički profil.....	89
4.5. Korpa.....	90
4.6. Checkout.....	91
4.7. Admin Panel.....	92
4.8. Narudžbe – Admin strana.....	93
4.9. Kreiranje novog produkta.....	94
4.10. Kategorije u admin panelu.....	95
5. Zaključak.....	96
6. Literatura.....	97

Skraćenice i akronimi

VR – Virtual Reality (Virtuelna stvarnost)

DOM – Document Object Model

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JSON – JavaScript Object Notation

ODM – Object Data Modeling

JWT – JSON Web Token

AWS – Amazon Web Services

1. Uvod

Aplikacija prezentovana u diplomskom radu je FullStack Mobile Shop za čiju realizaciju su korišćene primarno 4 tehnologije, a to su:

- React Native
- ExpressJS
- MongoDB
- NodeJS

Korisnik može da pregleda produkte, registruje nalog, poruči produkte na svoju adresu, pregleda narudžbe i upravlja svojim nalogom. Pored običnog korisnik takođe postoje i admin korisnici koji mogu da upravljaju shopom u smislu dodavanja, uređivanja i brisanja produkta, te upravljanje narudžbama. Aplikacija je cross-platform, što znači da je dostupna i na Android i iOS telefonima. Dizajn je fokusiran na jednostavnost i pristupačnost svim korisnicima.

2. Stack

2.1. React Native

React je JavaScript biblioteka otvorenog koda kreirana od strane Facebook-a. Koristi se za razvojni dio aplikacija za Android, Android TV, iOS, macOS, tvOS, Web i Windows omogućavajući developerima da koriste React biblioteku sa specijalnim komponentama. Takođe se koristi za razvoj Oculus VR aplikacija.

Radni principi React Native-a su potpuno isti kao kod React-a, osim što React Native ne manipuliše DOM preko Virtual DOM-a. React Native radi u pozadinskom procesu direktno na uređaju koji interpretuje JavaScript napisan od strane developera. Sintaksa izgleda slično HTML-u i CSS-u, ali ih ne sadrži. Umjesto njih poruke iz JavaScript threadova se koriste za manipulisanje nativnih prikaza.^[1]

Hello world primjer:

```
1 import { AppRegistry, Text } from 'react-native';
2 import * as React from 'react';
3
4 const HelloWorldApp = () => {
5   return <Text>Hello world!</Text>;
6 }
7
8 export default HelloWorldApp;
9
10 AppRegistry.registerComponent('HelloWorld', () => HelloWorldApp);
```

2.2. ExpressJS

Express je web aplikacija u okviru NodeJS servera. Koristi se za jednostranične, multistranične i hibridne web aplikacije. Poprilično je minimalan u svakom smislu pa je potrebno samo par stvari da bi se pokrenuo server sa Expressom. Prva stvar koju je potrebno uraditi je instalirati express paket koristeći node package manager ili npm. Instalacija u određenom direktorijumu se obavlja putem komande npm install express. Nakon instalacije kreira se app.js fajl u kojem će se instancirati express u varijabli sa sledećim kodom:

```
const express = require("express")
```

Aplikacija se instancira u varijablu te pozova funkciju express:

```
const app = express()
```

Postavljanje rute koja će na svaki get request odgovoriti sa nekim tekstom:

[1] https://en.wikipedia.org/wiki/React_Native

```
app.get('/', (req,res) => {  
    res.send('Hello World!')})
```

Poslednja stvar koja je potrebna je slušanje na portu:

```
app.listen(5000, () => {  
    console.log('Listening on port 5000')  
})
```

Nakon pokretanja app.js fajla korištenjem komande `node app.js` u terminalu moguće je posjetiti <http://localhost:5000> i vidjeti web aplikaciju.^[2]

2.3. MongoDB

MongoDB je jedna od najpopularnijih NoSQL bazi podataka. Napisana je u C++ i otvorenog je koda. U ovoj bazi su podaci sačuvani kao JSON dokumenti sa dinamičkim šemama. MongoDB čini integraciju podataka jednostavniju i brzu. Osmišljena je kao baza za opštu upotrebu i ogroman broj projekata je danas koristi^[3]. Primjer objekta sačuvanog u MongoDB bi izgledao ovako:

```
{  
  "student": {  
    "name": "John",  
    "class": "Intermediate",  
    "address": {  
      "street": "2293 Example Street",  
      "City": "Chicago",  
      "State": "IL"  
    }  
  }  
}
```

[2] <https://en.wikipedia.org/wiki/Express.js>

[3] <https://en.wikipedia.org/wiki/MongoDB>

3. Razvojni proces

3.1 Backend

3.1.1. prvi komit - intial commit

Razvojni proces započinje korištenjem komande “mkdir backend”, “cd backend” i “npm init -y” za inicijalizaciju node projekta. Zatim se instaliraju paketi express i dotenv^[4] sa komandom „npm install express dotenv“. Dotenv paket se koristi za .env fajl u kojem su sačuvani senzitivini podaci kao šifre. Nakon instalacije se instancira express, koristi se express.json() middleware za parsiranje JSON -a u aplikaciji i sluša se na portu 5000.

```
... .. @@ -0,0 +1,10 @@
1 + const express = require("express");
2 + require("dotenv/config");
3 +
4 + const app = express();
5 +
6 + app.use(express.json());
7 +
8 + app.listen(5000, () => {
9 +   console.log("Server running on port 5000");
10 + });
```

3.1.2. podešavanje baze - setup database

Dodaj se paket mongoose^[5] koji je zapravo ODM za MongoDB dizajniran da radi u asinhronom okruženju. U ovom projektu se koristi za svu komunikaciju sa bazom i kreiranje modela. Nakon instalacije se poziva u app.js i koristi se connect funkcija koja prima connection url. U ovom projektu baza je hostovana na Mongo Atlas cloud servisu, pa se iz dashboarda dobije konekcijski link i stavlja se u .env fajl.

[4] <https://www.npmjs.com/package/dotenv>

[5] <https://www.npmjs.com/package/mongoose>


```
... .. @@ -1,10 +1,23 @@
1 1   const express = require("express");
2 + const mongoose = require("mongoose");
3 +
2 4   require("dotenv/config");
3 5
4 6   const app = express();
5 7
6 8   app.use(express.json());
7 9
10 + mongoose
11 +   .connect(process.env.MONGO_URL, {
12 +     useNewUrlParser: true,
13 +     useUnifiedTopology: true,
14 +     writeConcern: {
15 +       j: true,
16 +     },
17 +   })
18 +   .then(() => console.log("Connected to database"))
19 +   .catch((err) => console.log(err));
20 +
8 21  app.listen(5000, () => {
9 22    console.log("Server running on port 5000");
10 23  });
```

3.1.3. podešavanje osnove za rute i šeme - set up foundation for routes and schemas

U ovom koraku se postavljaju osnovi za sve ono što će se nalaziti u ovoj aplikaciji, a to su: produkti, kategorije, korisnici i narudžbe. S obzirom da bi app.js bio ogroman ako sve rute budu u jednom fajlu, to će se rastaviti u folderu routes. Kreira se folder routes komandom „mkdir routes && cd routes“ te se kreiraju fajlovi products, categories, users i orders komandom „touch products.js categories.js users.js orders.js“. U ovim fajlovima se importuje express i instancira funkcija router iz express biblioteke, te se exportuje. Pored ruta potrebno je kreirati i modele za bazu. Kreira se folder models i u njemu se generišu fajlovi category.model.js, order.model.js, product.model.js i user.model.js. U modelima se importuje paket mongoose i poziva se funkciju Schema koja prima objekat koji opisuje model. Za sada se tu samo nalazi „name: String“ jer se postavlja osnova. Nakon kreiranja koriti se export funkcija modela iz mongoose paketa koja prima string (naziv dokumenta u biblioteci) i scheme. Pored svega tog instalira se i paket cors koji pruža pomoć pri vezivanju frontend-a i backend-a.

```

  7 backend/models/category.model.js
  ...
  1 + const mongoose = require("mongoose");
  2 +
  3 + const categorySchema = mongoose.Schema({
  4 +   name: String,
  5 + });
  6 +
  7 + exports.Category = mongoose.model("Category", categorySchema);

  7 backend/models/order.model.js
  ...
  1 + const mongoose = require("mongoose");
  2 +
  3 + const ordersSchema = mongoose.Schema({
  4 +   name: String,
  5 + });
  6 +
  7 + exports.Order = mongoose.model("Order", orderSchema);

  7 backend/models/product.model.js
  ...
  1 + const mongoose = require("mongoose");
  2 +
  3 + const productSchema = mongoose.Schema({
  4 +   name: String,
  5 + });
  6 +
  7 + exports.Product = mongoose.model("Product", productSchema);

  7 backend/models/user.model.js
  ...
  1 + const mongoose = require("mongoose");
  2 +
  3 + const userSchema = mongoose.Schema({
  4 +   name: String,
  5 + });
  6 +
  7 + exports.User = mongoose.model("User", userSchema);

```

```

  5 backend/routes/categories.js
  ...
  1 + const express = require("express");
  2 + const { Category } = require("../models/category.model");
  3 + const router = express.Router();
  4 +
  5 + module.exports = router;

  5 backend/routes/orders.js
  ...
  1 + const express = require("express");
  2 + const { Order } = require("../models/order.model");
  3 + const router = express.Router();
  4 +
  5 + module.exports = router;

  5 backend/routes/product.js
  ...
  1 + const express = require("express");
  2 + const { Product } = require("../models/product.model");
  3 + const router = express.Router();
  4 +
  5 + module.exports = router;

  5 backend/routes/users.js
  ...
  1 + const express = require("express");
  2 + const { User } = require("../models/user.model");
  3 + const router = express.Router();
  4 +
  5 + module.exports = router;

```

7 backend/models/category.model.js

```
1 + const mongoose = require("mongoose");
2 +
3 + const categorySchema = mongoose.Schema({
4 +   name: String,
5 + });
6 +
7 + exports.Category = mongoose.model("Category", categorySchema);
```

7 backend/models/order.model.js

```
1 + const mongoose = require("mongoose");
2 +
3 + const orderSchema = mongoose.Schema({
4 +   name: String,
5 + });
6 +
7 + exports.Order = mongoose.model("Order", orderSchema);
```

7 backend/models/product.model.js

```
1 + const mongoose = require("mongoose");
2 +
3 + const productSchema = mongoose.Schema({
4 +   name: String,
5 + });
6 +
7 + exports.Product = mongoose.model("Product", productSchema);
```

7 backend/models/user.model.js

```
1 + const mongoose = require("mongoose");
2 +
3 + const userSchema = mongoose.Schema({
4 +   name: String,
5 + });
6 +
7 + exports.User = mongoose.model("User", userSchema);
```

12 backend/app.js

```
1 const express = require("express");
2 const mongoose = require("mongoose");
3 + const cors = require("cors");
4
5 require("dotenv/config");
6
7 const app = express();
8
9 + app.options("*", cors());
10 +
11 + const productsRouter = require("../routes/product");
12 + const categoriesRouter = require("../routes/categories");
13 + const usersRouter = require("../routes/users");
14 + const ordersRouter = require("../routes/orders");
15 +
16 app.use(express.json());
17 + app.use("/api/products", productsRouter);
18 + app.use("/api/categories", categoriesRouter);
19 + app.use("/api/users", usersRouter);
20 + app.use("/api/orders", ordersRouter);
21
22 mongoose
23 .connect(process.env.MONGO_URL, {
```

3.1.4. podešavanje šeme za produkte i kategorije - set up product and category schema

Nakon kreiranja osnove za šeme, potrebno je dodijeliti parametre koji ih opisuju. Za svaku kategoriju potrebno je ime, ikonica i boja koja ju predstavlja.

```
11 backend/models/category.model.js
... .. @@ -1,7 +1,16 @@
1 1 const mongoose = require("mongoose");
2 2
3 3 const categorySchema = mongoose.Schema({
4 4 - name: String,
5 5 + name: {
6 6 + type: String,
7 7 + required: true,
8 8 + },
9 9 + icon: {
10 10 + type: String,
11 11 + },
12 12 + color: {
13 13 + type: String,
14 14 + },
15 15 });
16 16 exports.Category = mongoose.model("Category", categorySchema);
```

Produkti su malo kompleksniji te će da imaju više parametara. Svaki produkt treba da ima ime, opis, detaljni opis, glavnu sliku (url), ostale slike (niz url-ova), brend, cijenu, kategoriju kojoj pripada, količinu na stanju, ocjenu, broj ocjena, datum i boolean koji prikazuje da li je produkt istaknut ili ne. To sve izgleda ovako:

```
11 backend/models/product.model.js
... .. @@ -1,7 +1,51 @@
1 1 const mongoose = require("mongoose");
2 2
3 3 const productSchema = mongoose.Schema({
4 4 - name: String,
5 5 + name: {
6 6 + type: String,
7 7 + required: true,
8 8 + },
9 9 + description: {
10 10 + type: String,
11 11 + required: true,
12 12 + },
13 13 + richDescription: {
14 14 + type: String,
15 15 + default: "",
16 16 + },
17 17 + slug: {
18 18 + type: String,
19 19 + default: "",
20 20 + },
21 21 + slugs: [
22 22 + {
23 23 + type: String,
24 24 + },
25 25 + ],
26 26 + brands: [
27 27 + {
28 28 + type: String,
29 29 + default: "",
30 30 + },
31 31 + ],
32 32 + price: {
33 33 + type: Number,
34 34 + default: 0,
35 35 + },
36 36 + category: {
37 37 + type: mongoose.Schema.Types.ObjectId,
38 38 + ref: "Category",
39 39 + required: true,
40 40 + },
41 41 + availability: {
42 42 + type: Number,
43 43 + required: true,
44 44 + min: 0,
45 45 + },
46 46 + rating: {
47 47 + type: Number,
48 48 + default: 0,
49 49 + },
50 50 + numReviews: [
51 51 + {
52 52 + type: Number,
53 53 + default: 0,
54 54 + },
55 55 + ],
56 56 + manufacturer: {
57 57 + type: Boolean,
58 58 + default: false,
59 59 + },
60 60 + createdAt: {
61 61 + type: Date,
62 62 + default: Date.now,
63 63 + },
64 64 + });
65 65 exports.Product = mongoose.model("Product", productSchema);
```

3.1.5. završavanje rute i šema za produkte i kategorije - products and categories routes and schemas finished

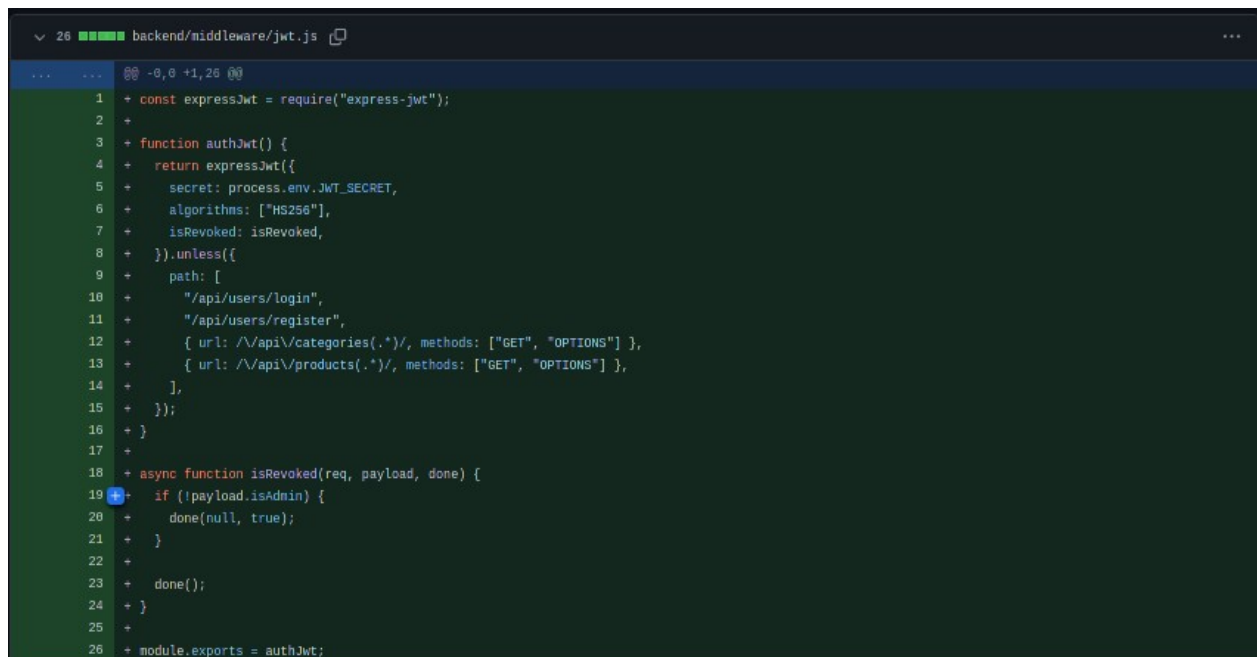
Nakon kreiranja šema potrebno je kreirati i rute u kojima će se koristiti te šeme. Za kategorije postoje „/“ ruta sa get requestom koja dohvata sve sve kategorije iz baze i vraća ih kao JSON, get „/:id“ prima id kao parametar i vraća jednu kategoriju sa tim id-om, post „/“ ruta prima sve potrebne podatke za kategoriju u body-u pa ih snima kao novu kategoriju u bazi, put „/:id“ prima id kao parametar i podatke koje koristi za izmjenu u body-u te ih mijenja u bazi na objektu sa id-om koji je u parametru i delete „/:id“ pronalazi objekat sa id-om iz parametra i briše ga iz baze. Sve isto se radi i za produkte, samo što pored toga se dodaje i get „/get/count“ ruta koja vraća ukupan broj produkta koja se koristi za za statistiku i get „/get/featured“ ruta koja se koristi za istaknute produkte na frontendu.

```
1  const express = require("express");
2  const { Category } = require("../models/category.model");
3  const router = express.Router();
4
5  + router.get("/", (req, res) => {
6  +   Category.find()
7  +   .then((result) => res.json(result))
8  +   .catch((err) => res.status(500).json(err));
9  + });
10 +
11 + router.get("/:id", (req, res) => {
12 +   Category.findById(req.params.id)
13 +   .then((result) => res.json(result))
14 +   .catch((err) => res.status(500).json(err));
15 + });
16 +
17 + router.put("/:id", (req, res) => {
18 +   Category.findByIdAndUpdate(req.params.id, {
19 +     name: req.body.name,
20 +     icon: req.body.icon,
21 +     color: req.body.color,
22 +   })
23 +   .then((result) => res.json(result))
24 +   .catch((err) => res.status(400).json(err));
25 + });
26 +
27 + router.post("/", async (req, res) => {
28 +   let category = new Category({
29 +     name: req.body.name,
30 +     icon: req.body.icon,
31 +     color: req.body.color,
32 +   });
33 +   category = await category.save();
34 +
35 +   if (!category) {
36 +     return res.status(404).send("Nemoguće napraviti kategoriju");
37 +   }
38 +   res.send(category);
39 + });
40 +
41 + router.delete("/:id", (req, res) => {
42 +   Category.findByIdAndRemove(req.params.id)
43 +   .then((result) => res.json(result))
44 +   .catch((err) => res.status(400).json(err));
45 + });
46 +
47  module.exports = router;
```

```
backend/routes/product.js
... .. @ -1,5 +1,5 @
1 + const express = require("express");
2 + const [ Category ] = require("../models/category.model");
3 + const [ Product ] = require("../models/product.model");
4 + const router = express.Router();
5
6 + router.get("/", (req, res) => {
7 +   let filter = {};
8 +   if (req.query.categories) {
9 +     filter = { category: req.query.categories.split(",") };
10 +   }
11 +
12 +   Product.find(filter)
13 +     .populate("category")
14 +     .then((result) => res.json(result))
15 +     .catch((err) => res.status(400).json(err));
16 + });
17 +
18 + router.get("/:id", (req, res) => {
19 +   Product.findById(req.params.id)
20 +     .populate("category")
21 +     .then((result) => res.json(result))
22 +     .catch((err) => res.status(400).json(err));
23 + });
24 +
25 + router.post("/", async (req, res) => {
26 +   const category = await Category.findById(req.body.category);
27 +   if (!category) return res.status(400).send("Bad category");
28 +
29 +   const product = new Product({
30 +     name: req.body.name,
31 +     description: req.body.description,
32 +     richDescription: req.body.richDescription,
33 +     image: req.body.image,
34 +     brand: req.body.brand,
35 +     price: req.body.price,
36 +     category: req.body.category,
37 +     countInStock: req.body.countInStock,
38 +     rating: req.body.rating,
39 +     numReviews: req.body.numReviews,
40 +     isFeatured: req.body.isFeatured,
41 +   });
42 +   product
43 +     .save()
44 +     .then((result) => res.json(result))
45 +     .catch((err) => res.status(400).json(err));
46 + });
47 +
48 + router.put("/:id", async (req, res) => {
49 +   const category = await Category.findById(req.body.category);
50 +   if (!category) return res.status(400).send("Bad category");
51 +
52 +   Product.findById(req.params.id)
53 +     .select("name description richDescription image brand price category countInStock rating numReviews isFeatured")
54 +     .then((product) => {
55 +       product.name = req.body.name;
56 +       product.description = req.body.description;
57 +       product.richDescription = req.body.richDescription;
58 +       product.image = req.body.image;
59 +       product.brand = req.body.brand;
60 +       product.price = req.body.price;
61 +       product.category = req.body.category;
62 +       product.countInStock = req.body.countInStock;
63 +       product.rating = req.body.rating;
64 +       product.numReviews = req.body.numReviews;
65 +       product.isFeatured = req.body.isFeatured;
66 +     })
67 +     .then((product) => {
68 +       product.save()
69 +         .then((result) => res.json(result))
70 +         .catch((err) => res.status(400).json(err));
71 +     });
72 + });
73 +
74 + router.delete("/:id", (req, res) => {
75 +   Product.findById(req.params.id)
76 +     .then((result) => res.json(result))
77 +     .catch((err) => res.status(400).json(err));
78 + });
79 +
80 +
81 + router.get("/get/count", async (req, res) => {
82 +   const productCount = await Product.countDocuments({ count: { $gt: 0 } });
83 +
84 +   if (!productCount) {
85 +     res.status(400).json({ error: "Error" });
86 +   }
87 +   res.json(productCount);
88 + });
89 +
90 + router.get("/get/featured", (req, res) => {
91 +   Product.find({ isFeatured: true })
92 +     .then((result) => res.json(result))
93 +     .catch((err) => res.json(err));
94 + });
95 +
96 + module.exports = router;
```

3.1.6. korisnik model, rute i JWT autentikacija - users model, routes and jwt auth

Instaliraju se paketi bcryptjs^[6], express-jwt i jsonwebtoken^[7]. Zadnja dva paketa se koriste za autentikaciju korisnika a bcryptjs se koristi za hešovanje i provjeravanje korisničkih šifri. Kreira se user model koji će da ima parametre ime, email, heš šifre, broj telefona, ulicu, stan, poštanski broj, grad, državu i isAdmin parametar koji ako je true daje korisniku administratorske komande. Za rute se kreiraju iste rute kao i na kategorijama te se dodaju login i register rute koje objašnjavaju same sebe. Nakon uspješnog logina korisnik dobija JWT koji kasnije koristi za autentikaciju. Pored svega ovoga se kreira i middleware koji se koristi za autentikaciju sa JWT-om koji je vraćen iz logina. Korisnik može da pristupi samo rutama „/api/users/login“, „/api/users/register“, „/api/categories(.*)“ i „/api/products(.*)“ bez autentikacije. Sve ostale rute su za sada nedostupne.



```
backend/middleware/jwt.js
1 + const expressJwt = require("express-jwt");
2 +
3 + function authJwt() {
4 +   return expressJwt({
5 +     secret: process.env.JWT_SECRET,
6 +     algorithms: ["HS256"],
7 +     isRevoked: isRevoked,
8 +   }).unless({
9 +     path: [
10 +       "/api/users/login",
11 +       "/api/users/register",
12 +       { url: /\api\categories(.*)/, methods: ["GET", "OPTIONS"] },
13 +       { url: /\api\products(.*)/, methods: ["GET", "OPTIONS"] },
14 +     ],
15 +   });
16 + }
17 +
18 + async function isRevoked(req, payload, done) {
19 +   if (!payload.isAdmin) {
20 +     done(null, true);
21 +   }
22 +   done();
23 + }
24 +
25 +
26 + module.exports = authJwt;
```

[6] <https://www.npmjs.com/package/bcryptjs>

[7] <https://www.npmjs.com/package/jsonwebtoken>

41 backend/models/user.model.js

```
... -1,7 +1,46 @@
1 1 const mongoose = require("mongoose");
2 2
3 3 const userSchema = mongoose.Schema({
4 4   - name: String,
5 5   + name: {
6 6     + type: String,
7 7     + required: true,
8 8   },
9 9   + email: {
10 10    + type: String,
11 11    + required: true,
12 12   },
13 13   + passwordHash: {
14 14    + type: String,
15 15    + required: true,
16 16   },
17 17   + phone: {
18 18    + type: String,
19 19    + required: true,
20 20   },
21 21   + isAdmin: {
22 22    + type: Boolean,
23 23    + required: true,
24 24   },
25 25   + street: {
26 26    + type: String,
27 27    + default: "",
28 28   },
29 29   + apartment: {
30 30    + type: String,
31 31    + default: "",
32 32   },
33 33   + zip: {
34 34    + type: String,
35 35    + default: "",
36 36   },
37 37   + city: {
38 38    + type: String,
39 39    + default: "",
40 40   },
41 41   + country: {
42 42    + type: String,
43 43    + default: "",
44 44   },
45 45 });
46 46 exports.User = mongoose.model("User", userSchema);
```



```
122 backend/routes/users.js
... .. 00 -1,5 +1,127 00
1 1 const express = require("express");
2 2 const { User } = require("../models/user.model");
3 3 const router = express.Router();
4 4 + const bcrypt = require("bcryptjs");
5 5 + const jwt = require("jsonwebtoken");
6 6 +
7 7 + router.get("/", (req, res) => {
8 8 +   User.find()
9 9 +     .select("-passwordHash")
10 10 +     .then((result) => res.json(result))
11 11 +     .catch((err) => res.status(400).json(err));
12 12 + });
13 13 +
14 14 + router.post("/", (req, res) => {
15 15 +   const user = new User({
16 16 +     name: req.body.name,
17 17 +     email: req.body.email,
18 18 +     passwordHash: bcrypt.hashSync(req.body.password, 10),
19 19 +     phone: req.body.phone,
20 20 +     isAdmin: req.body.isAdmin,
21 21 +     apartment: req.body.apartment,
22 22 +     zip: req.body.zip,
23 23 +     city: req.body.city,
24 24 +     country: req.body.country,
25 25 +   });
26 26 +   user
27 27 +     .save()
28 28 +     .then((result) => res.json(result))
29 29 +     .catch((err) => res.status(400).json(err));
30 30 + });
31 31 +
32 32 + router.get("/:id", (req, res) => {
33 33 +   User.findById(req.params.id)
34 34 +     .select("-passwordHash")
35 35 +     .then((result) => res.json(result))
36 36 +     .catch((err) => res.status(400).json(err));
37 37 + });
38 38 +
39 39 + router.put("/:id", async (req, res) => {
40 40 +   const userExist = await User.findById(req.params.id);
41 41 +   let newPassword;
42 42 +   if (req.body.password) {
43 43 +     newPassword = bcrypt.hashSync(req.body.password, 10);
44 44 +   } else {
45 45 +     newPassword = userExist.passwordHash;
46 46 +   }
47 47 +
48 48 +   const user = await User.findByIdAndUpdate(
49 49 +     req.params.id,
50 50 +     [
51 51 +       name: req.body.name,
52 52 +       email: req.body.email,
53 53 +       passwordHash: newPassword,
54 54 +       phone: req.body.phone,
55 55 +       isAdmin: req.body.isAdmin,
56 56 +       street: req.body.street,
57 57 +       apartment: req.body.apartment,
58 58 +       zip: req.body.zip,
59 59 +       city: req.body.city,
60 60 +       country: req.body.country,
61 61 +     ],
62 62 +     { new: true }
63 63 +   );
64 64 +
65 65 +   if (!user) return res.status(400).send("the user cannot be created!");
66 66 +
67 67 +   res.send(user);
68 68 + });
69 69 +
```

```
69 +
70 + router.post("/register", (req, res) => {
71 +   const user = new User({
72 +     name: req.body.name,
73 +     email: req.body.email,
74 +     passwordHash: bcrypt.hashSync(req.body.password, 10),
75 +     phone: req.body.phone,
76 +     isAdmin: req.body.isAdmin,
77 +     apartment: req.body.apartment,
78 +     zip: req.body.zip,
79 +     city: req.body.city,
80 +     country: req.body.country,
81 +   });
82 +   user
83 +     .save()
84 +     .then((result) => res.json(result))
85 +     .catch((err) => res.status(400).json(err));
86 + });
87 +
88 + router.post("/login", async (req, res) => {
89 +   if (!req.body.password) {
90 +     return res.status(400).json("No password");
91 +   }
92 +   const user = await User.findOne({ email: req.body.email });
93 +
94 +   if (!user) {
95 +     return res.status(400).send("User not found");
96 +   }
97 +
98 +   if (user && bcrypt.compareSync(req.body.password, user.passwordHash)) {
99 +     const token = jwt.sign(
100 +       { userId: user._id, isAdmin: user.isAdmin },
101 +       process.env.JWT_SECRET,
102 +       {
103 +         expiresIn: "1w",
104 +       }
105 +     );
106 +     res.send(token);
107 +   } else {
108 +     res.status(400).send("Wrong password");
109 +   }
110 + });
111 +
112 + router.delete("/:id", (req, res) => {
113 +   User.findByIdAndRemove(req.params.id)
114 +     .then((result) => res.json(result))
115 +     .catch((err) => res.status(400).json(err));
116 + });
117 +
118 + router.get("/get/count", async (req, res) => {
119 +   const userCount = await User.countDocuments({ count }) => count;
120 +
121 +   if (!userCount) {
122 +     res.status(400).json("Error");
123 +   }
124 +   res.json(userCount);
125 + });
126 +
127 + module.exports = router;
```

3.1.7. dodavnje order i orderitem šema - added order and orderitems schema

Kako svaka narudžba može da ima više produkta a svaki produkt može da ima različitu količinu, pravi se dokument orderItem koji će da veže produkte i narudžbe. On će da ima

kvantitet i ID produkta. Narudžba će da ima ID orderItems dokumenta koji je vezan za nju, adresu1, adresu2, grad, poštanski broj, državu, broj telefona, status, ukupnu cijenu, korisnika koju je naručio i datum narudžbe.

```
16 backend/models/orderItem.model.js
... 00 -0,0 +1,16 @0
1 + const mongoose = require("mongoose");
2 +
3 + const orderItemSchema = mongoose.Schema({
4 +   quantity: {
5 +     type: Number,
6 +     required: true,
7 +   },
8 +   product: [
9 +     {
10 +       type: mongoose.Schema.Types.ObjectId,
11 +       ref: "Product",
12 +     },
13 +   ],
14 + });
15 +
16 + exports.OrderItem = mongoose.model("OrderItem", orderItemSchema);
```

```
47 backend/models/order.model.js
... 00 -1,7 +1,52 @0
1 const mongoose = require("mongoose");
2
3 const orderSchema = mongoose.Schema({
4   - name: String,
5   + orderItems: [
6   +   {
7   +     type: mongoose.Schema.Types.ObjectId,
8   +     ref: "OrderItems",
9   +     required: true,
10 +   },
11 + ],
12 + shippingAddress1: {
13 +   type: String,
14 +   required: true,
15 + },
16 + shippingAddress2: {
17 +   type: String,
18 + },
19 + city: {
20 +   type: String,
21 +   required: true,
22 + },
23 + zip: {
24 +   type: String,
25 +   required: true,
26 + },
27 + country: {
28 +   type: String,
29 +   required: true,
30 + },
31 + phone: [
32 +   {
33 +     type: String,
34 +     required: true,
35 +   },
36 + ],
37 + status: {
38 +   type: String,
39 +   required: true,
40 +   default: "Pending",
41 + },
42 + totalPrice: {
43 +   type: Number,
44 + },
45 + user: {
46 +   type: mongoose.Schema.Types.ObjectId,
47 +   ref: "User",
48 + },
49 + dateOrdered: {
50 +   type: Date,
51 +   default: Date.now,
52 + },
53 + });
54
55 exports.Order = mongoose.model("Order", orderSchema);
```

3.1.8. završavanje order ruta - finished orders routes

Kada se završe šeme za narudbže, potrebno je kreirati i rute na backendu u kojima se koriste te šeme. Dodaju se sve rute kao u kategorijama, a zatim se dodaju i ruta get „/get/usersorders/:userid“ na kojoj se mogu dohvatiti sve narudžbe za korisnika sa datim id-om u parametru.

```
115 backend/routes/orders.js
... .. 00 -1,5 +1,120 00
1  const express = require("express");
2  const { Order } = require("../models/order.model");
3  + const { OrderItem } = require("../models/orderItem.model");
4  + const { Product } = require("../models/product.model");
5  const router = express.Router();
6
7  + router.get("/", (req, res) => {
8  +   Order.find()
9  +   .populate("user", "name")
10 +   .populate("orderItems")
11 +   .sort({ dateOrdered: -1 })
12 +   .then(result) => res.json(result)
13 +   .catch(err) => res.status(400).json(err);
14 + });
15
16 + router.get("/:id", (req, res) => {
17 +   Order.findById(req.params.id)
18 +   .populate("user", "name")
19 +   .populate({
20 +     path: "orderItems",
21 +     populate: { path: "product", populate: "category" },
22 +   })
23 +   .then(result) => res.json(result)
24 +   .catch(err) => res.status(400).json(err);
25 + });
26
27 + router.post("/", async (req, res) => {
28 +   const orderItemsIds = Promise.all(
29 +     req.body.orderItems.map(async (orderItem) => {
30 +       let newOrderItem = new OrderItem({
31 +         quantity: orderItem.quantity,
32 +         product: orderItem.product,
33 +       });
34 +       newOrderItem = await newOrderItem.save();
35 +       return newOrderItem._id;
36 +     })
37 +   );
38 +   const orderItemsIdsResolved = await orderItemsIds;
39 +   const totalPrices = await Promise.all(
40 +     orderItemsIdsResolved.map(async (element) => {
41 +       const orderItem = await OrderItem.findById(element).populate(
42 +         "product",
43 +         "price"
44 +       );
45 +       const totalPrice = orderItem.product[0].price * orderItem.quantity;
46 +       return totalPrice;
47 +     })
48 +   );
49 +   const totalPrice = totalPrices.reduce((a, b) => a + b, 0);
50 +   const order = new Order({
51 +     orderItems: orderItemsIdsResolved,
52 +     shippingAddress1: req.body.shippingAddress1,
53 +     shippingAddress2: req.body.shippingAddress2,
54 +     status: req.body.status,
55 +     city: req.body.city,
56 +     zip: req.body.zip,
57 +     country: req.body.country,
58 +     phone: req.body.phone,
59 +     status: req.body.status,
60 +     totalPrice: totalPrice,
61 +     user: req.body.user,
62 +   });
63 +   order
64 +     .save()
65 +     .then(result) => res.json(result)
66 +     .catch(err) => res.status(400).json(err);
67 + });
68 +
```

```

75 + router.put("/:id", async (req, res) => {
76 +   Order.findByIdAndUpdate(
77 +     req.params.id,
78 +     { status: req.body.status },
79 +     { new: true }
80 +   )
81 +   .then((result) => res.json(result))
82 +   .catch((err) => res.status(400).json(err));
83 + });
84 +
85 + router.delete("/:id", (req, res) => {
86 +   Order.findByIdAndRemove(req.params.id)
87 +   .then((result) => res.json(result))
88 +   .catch((err) => res.status(400).json(err));
89 + });
90 +
91 + router.get("/get/totalsales", async (req, res) => {
92 +   const totalSales = await Order.aggregate([
93 +     { $group: { _id: null, totalsales: { $sum: "$totalPrice" } } },
94 +   ]);
95 +   if (!totalSales) {
96 +     return res.status(400).json("Error");
97 +   }
98 +   res.json(totalSales);
99 + });
100 +
101 + router.get("/get/count", async (req, res) => {
102 +   const orderCount = await Order.countDocuments({ count => count });
103 +   if (!orderCount) {
104 +     res.status(500).json("error");
105 +   }
106 +   res.json(orderCount);
107 + });
108 +
109 + router.get("/get/userorders/:userId", (req, res) => {
110 +   Order.find({ user: req.params.userId })
111 +   .populate({
112 +     path: "orderItems",
113 +     populate: { path: "product", populate: "category" },
114 +   })
115 +   .sort({ dateOrdered: -1 })
116 +   .then((result) => res.json(result))
117 +   .catch((err) => res.status(400).json(err));
118 + });
119 +
120 module.exports = router;

```

3.1.9. dodavanje prenosa fajlova - added file upload

Instalira se paket multer^[8] koji se koristi za upload fajlova u expressu. Podešava se za slike produkta i koristi se u unosu novih produkta i uređivanju već postojećih produkta.

```
backend/routes/product.js
1 1  const express = require("express");
2 2  const { Category } = require("../models/category.model");
3 3  const { Product } = require("../models/product.model");
4 4  const router = express.Router();
5 5  + const multer = require("multer");
6 6  + const mongoose = require("mongoose");
7 7  +
8 8  + const FILE_TYPE_MAP = {
9 9  +   "image/png": "png",
10 10 +   "image/jpeg": "jpeg",
11 11 +   "image/jpg": "jpg",
12 12 + };
13 13 +
14 14 + const storage = multer.diskStorage({
15 15 +   destination: function (req, file, cb) {
16 16 +     const isValid = FILE_TYPE_MAP[file.mimetype];
17 17 +     let uploadError = new Error("invalid image type");
18 18 +
19 19 +     if (!isValid) {
20 20 +       uploadError = null;
21 21 +     }
22 22 +     cb(uploadError, "public/uploads/");
23 23 +   },
24 24 +   filename: function (req, file, cb) {
25 25 +     const extension = FILE_TYPE_MAP[file.mimetype];
26 26 +     const fileName = file.originalname.replace(" ", "-");
27 27 +     const uniqueSuffix = Date.now() + "-" + Math.round(Math.random() * 1e9);
28 28 +     cb(null, `${fileName}-${uniqueSuffix}.${extension}`);
29 29 +   },
30 30 + });
31 31 +
32 32 + const uploadOptions = multer({ storage: storage });
33 33
34 34  router.get("/", (req, res) => {
35 35  let filter = {};
36 36
37 37  router.get("/:id", (req, res) => {
38 38  .catch((err) => res.status(400).json(err));
39 39  });
40 40
41 41  - router.post("/", async (req, res) => {
42 42  + router.post("/", uploadOptions.single("image"), async (req, res) => {
43 43  +   const file = req.file;
44 44  +   if (!file) {
45 45  +     return res.status(400).json("No file");
46 46  +   }
47 47
48 48  const category = await Category.findById(req.body.category);
49 49  if (!category) return res.status(400).send("Bad category");
50 50
51 51  -
52 52  + const fileName = req.file.filename;
53 53  + const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
54 54
55 55  const product = new Product({
56 56  name: req.body.name,
57 57  description: req.body.description,
58 58  richDescription: req.body.richDescription,
59 59  - image: req.body.image,
60 60  + image: `${basePath}${fileName}`,
61 61
62 62  brand: req.body.brand,
63 63  price: req.body.price,
64 64  category: req.body.category,
```

[8] <https://www.npmjs.com/package/multer>

```

45 78     .catch(err => res.status(400).json(err));
46 79   });
47 80
48 81 - router.put("/", async (req, res) => {
81 82 + router.put("/", uploadOptions.single("image"), async (req, res) => {
82 83 +   if (!mongoose.isValidObjectId(req.params.id)) {
83 84 +     return res.status(400).send("Bad id");
84 85 +   }
85 86 +
49 86   const category = await Category.findById(req.body.category);
50 87   if (!category) return res.status(400).send("Bad category");
51 88
89 89 +   const product = await Product.findById(req.params.id);
90 90 +   if (!category) return res.status(400).send("Bad product");
91 91 +
92 92 +   const file = req.file;
93 93 +   let imagePath;
94 94 +
95 95 +   if (file) {
96 96 +     const fileName = req.file.filename;
97 97 +     const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
98 98 +     imagePath = `${basePath}${fileName}`;
99 99 +   } else {
100 100 +     imagePath = product.image;
101 101 +   }
102 102 +
52 103   Product.findByIdAndUpdate(
53 104     req.params.id,
54 105     {
55 106       name: req.body.name,
56 107       description: req.body.description,
57 108       richDescription: req.body.richDescription,
58 109 -       image: req.body.image,
59 109 +       image: imagePath,
60 110       brand: req.body.brand,
61 111       price: req.body.price,
62 112       category: req.body.category,

```

```

93 144     .catch(err => res.json(err));
94 145   });
95 146
147 147 + router.put(
148 148 +   "/gallery-images/:id",
149 149 +   uploadOptions.array("images", 10),
150 150 +   async (req, res) => {
151 151 +     if (!mongoose.isValidObjectId(req.params.id)) {
152 152 +       return res.status(400).send("Bad id");
153 153 +     }
154 154 +     const files = req.files;
155 155 +
156 156 +     let imagePaths = [];
157 157 +
158 158 +     if (files) {
159 159 +       const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
160 160 +       files.map((file) => {
161 161 +         imagePaths.push(`${basePath}${file.filename}`);
162 162 +       });
163 163 +     }
164 164 +
165 165 +     const product = await Product.findByIdAndUpdate(
166 166 +       req.params.id,
167 167 +       {
168 168 +         images: imagePaths,
169 169 +       },
170 170 +       {
171 171 +         new: true,
172 172 +       }
173 173 +     );
174 174 +     if (!product) {
175 175 +       return res.status(500).send("Error, not updated");
176 176 +     }
177 177 +     res.json(product);
178 178 +   }
179 179 + );
180 180 +
96 181 module.exports = router;

```

3.1.10. ispravka, dodavanje parametra id na put request u products.js - forgot to add params id on put request for products.js

Ispravka u fajlu products.js gdje nedostaje parametar id na put request ruti. Greška se nalazi na liniji broj 81.

```
backend/routes/product.js
@@ -78,7 +78,7 @@ router.post("/", uploadOptions.single("image"), async (req, res) => {
78 78     .catch((err) => res.status(400).json(err));
79 79   });
80 80
81 - router.put("/", uploadOptions.single("image"), async (req, res) => {
+ router.put("/:id", uploadOptions.single("image"), async (req, res) => {
82 82     if (!mongoose.isValidObjectId(req.params.id)) {
83 83       return res.status(400).send("Bad id");
84 84     }
}
```

3.1.11. dodavanje mogućnosti za obične korisnike da koriste narudžbe - added ability for regular users to use and post their orders

U konfiguracijama za JWT middleware dodaje se i ruta „/api/orders(.*)“ te se time omogućava običnim korisnicima da imaju interakciju sa narudžbama.

```
backend/middleware/jwt.js
@@ -9,6 +9,7 @@ function authJwt() {
9 9     path: [
10 10       "/api/users/login",
11 11       "/api/users/register",
12 12 +     { url: /\api\/orders(.*)/, methods: ["GET", "OPTIONS", "POST"] },
13 13     { url: /\api\/categories(.*)/, methods: ["GET", "OPTIONS"] },
14 14     { url: /\api\/products(.*)/, methods: ["GET", "OPTIONS"] },
15 15     { url: /\public\/upload(.*)/, methods: ["GET", "OPTIONS"] },
}
```


3.2. Frontend

3.2.1. Kreiranje nove expo aplikacije - Created a new Expo app

S obzirom da se ovaj projekat kreira pomoću Expo^[9] platforme, potrebno je globalno instalirati expo cli sa komandom „sudo npm install -g expo-cli“. Nakon instalacije se pokreće komanda „expo init frontend“ i selektuje se blank template. Nakon kreiranja stvorena je osnova za React Native projekat.

3.2.2. dodavanje productcontainer, list, card i header komponenti - added productcontainer, list, card and header

Instalira se paket „native-base“^[10] iz kojeg se koriste neke komponente da bi aplikacija ljepše izgledala. Kreira se folder components i u njemu se smještaju sve komponente koje će se koristiti u aplikaciji. U tom folderu kreira se ProductCard.js koji je kartica za svaki produkt koji se prikazuje na početnoj strani, ProductContainer.js u koji se stavlja lista produkta na početnoj strani, ProductList.js sadrži sve produkte u karticama i Header.js za sada sadrži samo logo aplikacije. Sve komponente se stilizuju. U App.js se ubacuju Header i ProductContainer.

[9] <https://www.npmjs.com/package/expo-cli>

[10] <https://www.npmjs.com/package/native-base>

```
Screen/Product/ProductCard.js
...
1 + import React from "react";
2 + import {
3 +   StyleSheet,
4 +   View,
5 +   Dimensions,
6 +   Image,
7 +   Text,
8 +   Button,
9 + } from "react-native";
10 +
11 + var { width } = Dimensions.get("window");
12 +
13 + export default function ProductCard(props) {
14 +   const { name, price, image, countInStock } = props;
15 +   return (
16 +     <View style={styles.container}>
17 +       <Image
18 +         style={styles.image}
19 +         resizeMode="contain"
20 +         source={{
21 +           uri: image
22 +           ? image
23 +           : "https://cdn.pocooay.com/photos/2012/04/01/17/26/box-23648_969_728.png",
24 +         }}
25 +       </Image>
26 +       <View style={styles.card}><View>
27 +         <Text style={styles.title}>
28 +           {name.length > 15 ? name.substring(0, 15 - 2) + "...": name}
29 +         </Text>
30 +         <Text style={styles.price}>${price}</Text>
31 +
32 +         {countInStock > 0 ? (
33 +           <View style={{ marginBottom: 10 }}>
34 +             <Button title="Add" color="green"></Button>
35 +           </View>
36 +         ) : (
37 +           <Text style={{ marginTop: 10 }}>Currently Unavailable</Text>
38 +         )}
39 +       </View>
40 +     </View>
41 +   );
42 +
43 +   const styles = StyleSheet.create([
44 +     container: {
45 +       width: width / 2 - 20,
46 +       height: width / 1.7,
47 +       padding: 10,
48 +       borderRadius: 10,
49 +       marginTop: 10,
50 +       marginBottom: 5,
51 +       marginLeft: 10,
52 +       alignItems: "center",
53 +       elevation: 8,
54 +       backgroundColor: "white",
55 +     },
56 +     image: {
57 +       width: width / 2 - 20 - 10,
58 +       height: width / 2 - 20 - 30,
59 +       backgroundColor: "transparent",
60 +       position: "absolute",
61 +       top: -45,
62 +     },
63 +     card: {
64 +       marginBottom: 10,
65 +       height: width / 2 - 20 - 90,
66 +       backgroundColor: "transparent",
67 +       width: width / 2 - 20 - 10,
68 +     },
69 +     title: {
70 +       fontWeight: "bold",
71 +       fontSize: 14,
72 +       textAlign: "center",
73 +     },
74 +     price: {
75 +       fontSize: 20,
76 +       color: "orange",
77 +       marginTop: 10,
78 +     },
79 +   ]);

```

```
▼ 58 Screens/Products/ProductContainer.js
... .. 88 -1,10 +1,51 88
1 - import React, { useState } from "react";
2 - import { View, StyleSheet, Text, ActivityIndicator } from "react-native";
3 + import React, { useState, useEffect } from "react";
4 + import { View, StyleSheet, FlatList, ActivityIndicator } from "react-native";
5 + import { Container, Header, Icon, Item, Input, Text } from "react-native-base";
6 + import ProductList from "../ProductList";
7 +
8 + const data = require("../assets/products.json");
9
10 + export default function ProductContainer() {
11 +   const [products, setProducts] = useState([]);
12 +
13 +   useEffect(() => {
14 +     setProducts(data);
15 +     return () => {
16 +       setProducts([]);
17 +     };
18 +   }, []);
19 +
20 +   return (
21 +     <View>
22 +       <Text>Product Container</Text>
23 +     </View>
24 +     <Container>
25 +       <Header searchBar rounded>
26 +         <Item>
27 +           <Icon name="ios-search"/></Icon>
28 +           <Input placeholder="Search"/></Input>
29 +         </Item>
30 +       </Header>
31 +       <View style={styles.container}>
32 +         <Text>Product Container</Text>
33 +         <View style={styles.listContainer}>
34 +           <FlatList
35 +             numColumns={2}
36 +             data={products}
37 +             renderItem={({ item } => (
38 +               <ProductList key={item.brand} item={item}></ProductList>
39 +             ))}
40 +             keyExtractor={({ item } => item.brand}
41 +           /></FlatList>
42 +       </View>
43 +     </View>
44 +   </Container>
45 + );
46
47 + }
48
49 + }
50 +
51 + }
52 +
53 + }
54 +
55 + }
56 +
57 + }
58 + }];
59
60 + }];
61
62 + }];
63
64 + }];
65
66 + }];
67
68 + }];
69
70 + }];
71
72 + }];
73
74 + }];
75
76 + }];
77
78 + }];
79
80 + }];
81
82 + }];
83
84 + }];
85
86 + }];
87
88 + }];
89
90 + }];
91
92 + }];
93
94 + }];
95
96 + }];
97
98 + }];
99
100 + }];
```

```
24 Shared/Header.js
...
1 + import React from "react";
2 + import { StyleSheet, View, Image, SafeAreaView } from "react-native";
3 +
4 + export default function Header() {
5 +   return (
6 +     <SafeAreaView style={styles.header}>
7 +       <Image
8 +         source={require("../assets/shop.jpg")}
9 +         resizeMode="contain"
10 +         style={{ height: 50 }}
11 +       ></Image>
12 +     </SafeAreaView>
13 +   );
14 + }
15 +
16 + const styles = StyleSheet.create({
17 +   header: {
18 +     width: "100%",
19 +     flexDirection: "row",
20 +     alignItems: "center",
21 +     justifyContent: "center",
22 +     padding: 20,
23 +   },
24 + });
```

```
App.js
+
2 import { StatusBar } from "expo-status-bar";
3 import React from "react";
4 import { StyleSheet, Text, View } from "react-native";
5 import ProductContainer from "../Screens/Products/ProductContainer";
6 import Header from "../Shared/Header";
7
8 export default function App() {
9   return (
10 +     <View style={styles.container}>
11 +       <Header></Header>
12 +       <ProductContainer></ProductContainer>
13 +     </View>
14   );
15 }
```

3.2.3. dodavanje polje za pretragu i funkcionalnost - added search field and functionality

Potrebno je da korisnicu mogu da pretraže proizvode unoseći njihovo ime ili naziv njihovog brenda. Zbog toga se dodaje search polje koje je zapravo Input komponenta iz native-base te se iz nje izvlači string i filteruju proizvodi koji sadrže taj string te se prikazuju na

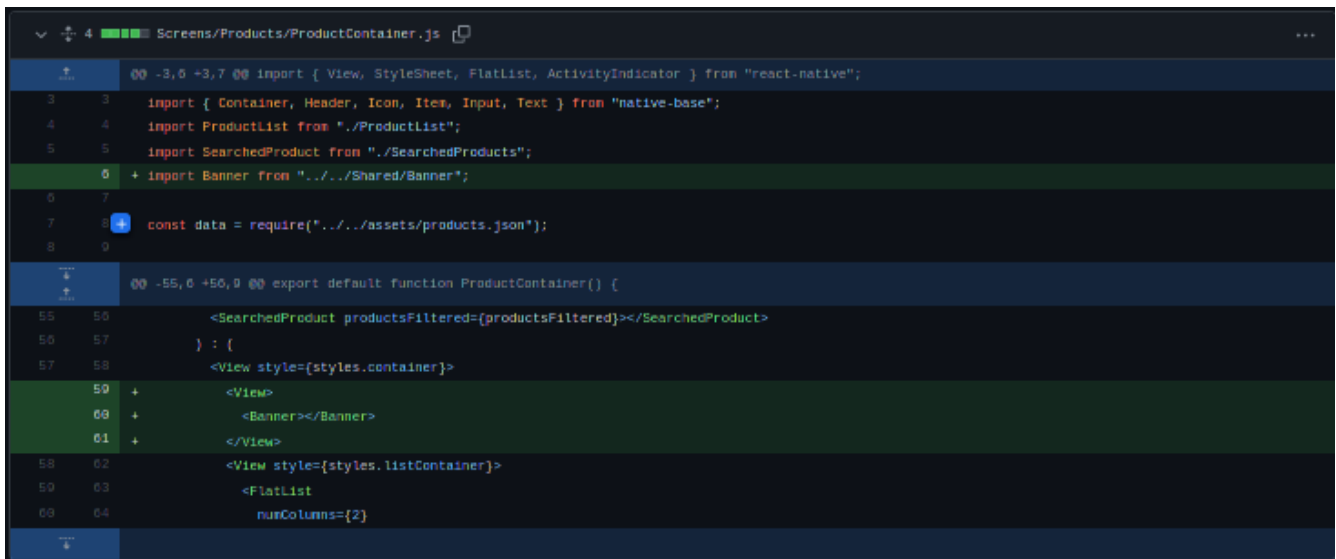
SearchedProducts komponenti. SearchProducts komponenta se prikazuje samo ako korisnik klikne na search.

```
47 Screens/Products/SearchedProducts.js
...
1 + import React from "react";
2 + import { View, StyleSheet, Dimensions } from "react-native";
3 + import { Content, Left, Body, ListItem, Thumbnail, Text } from "native-base";
4 +
5 + var { width } = Dimensions.get("window");
6 +
7 + export default function SearchedProducts(props) {
8 +   const { productsFiltered } = props;
9 +   return (
10 +     <Content style={{ width: width }}>
11 +       {productsFiltered.length > 0 ? (
12 +         productsFiltered.map((item) => {
13 +           return (
14 +             <ListItem key={item._id.Soid} avatar>
15 +               <Left>
16 +                 <Thumbnail
17 +                   source={{
18 +                     uri: item.image
19 +                     ? item.image
20 +                     : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23649_960_720.png",
21 +                   }}
22 +                 ></Thumbnail>
23 +               </Left>
24 +               <Body>
25 +                 <Text>{item.name}</Text>
26 +                 <Text note>{item.description}</Text>
27 +               </Body>
28 +             </ListItem>
29 +           );
30 +         })
31 +       ) : (
32 +         <View style={styles.center}>
33 +           <Text style={{ alignSelf: "center" }}>
34 +             No products match the selected search
35 +           </Text>
36 +         </View>
37 +       )
38 +     </Content>
39 +   );
40 + }
41 +
42 + const styles = StyleSheet.create({
43 +   center: {
44 +     justifyContent: "center",
45 +     alignItems: "center",
46 +   },
47 + });
```

```
Screena/Products/ProductContainer.js
1 import React, { useState, useEffect } from "react";
2 import { View, StyleSheet, FlatList, ActivityIndicator } from "react-native";
3 import { Container, Header, Icon, Item, Input, Text } from "native-base";
4 import ProductList from "../ProductList";
5 import SearchProduct from "../SearchProduct";
6
7 const data = require("../assets/products.json");
8
9 export default function ProductContainer() {
10   const [products, setProducts] = useState([]);
11   const [productsFiltered, setProductsFiltered] = useState([]);
12   const [focus, setFocus] = useState();
13
14   useEffect(() => {
15     setProducts(data);
16     setProductsFiltered(data);
17     setFocus(false);
18   }, []);
19
20   const openList = () => {
21     setFocus(true);
22   };
23
24   const onClose = () => {
25     setFocus(false);
26   };
27
28   return (
29     <Container>
30       <Header searchBar rounded>
31         <Item>
32           <Icon name="ios-search"/></Icon>
33           <Text placeholder="Search"/></Text>
34           <Input>
35             <Text
36               placeholder="Search"
37               onFocus={openList}
38               onChange={e => setFocus(e.target.value)}
39             />
40           </Input>
41           <Icon onPress={onClose} name="ios-close"/></Icon>
42         </Item>
43       </Header>
44
45       <View style={styles.container}>
46         <Text>Product Container</Text>
47         <View style={styles.listContainer}>
48           <FlatList>
49             <ListHeader>
50               <Text>Products</Text>
51             </ListHeader>
52             <ListContent>
53               <FlatList
54                 data={products}
55                 renderItem={({ item }) => (
56                   <ProductList key={item.brand} item={item}></ProductList>
57                 )}
58                 keyExtractor={({ item }) => item.brand}
59               />
60             </ListContent>
61           </FlatList>
62
63           {focus ? (
64             <SearchProduct productsFiltered={productsFiltered}></SearchProduct>
65           ) : (
66             <View style={styles.container}>
67               <Text>Product Container</Text>
68               <View style={styles.listContainer}>
69                 <FlatList>
70                   <ListHeader>
71                     <Text>Products</Text>
72                   </ListHeader>
73                   <ListContent>
74                     <FlatList
75                       data={products}
76                       renderItem={({ item }) => (
77                         <ProductList key={item.brand} item={item}></ProductList>
78                       )}
79                       keyExtractor={({ item }) => item.brand}
80                     />
81                   </ListContent>
82                 </FlatList>
83               </View>
84             </View>
85           )}
86         </View>
87       </Container>
88     );
89   }
90 }
91
92 const styles = StyleSheet.create({
93   container: {
94     padding: 10,
95   },
96   listContainer: {
97     padding: 10,
98   },
99 });
```

3.2.4. dodavanje bannerera sa istaknutim produktima - added banner

U nekim dijelovima ovaj banner se naziva i carousel, ali on predstavlja dio gdje se prikazuju istaknuti artikli. S obzirom da ih ima više potrebno je da se automatski mijenjaju i da korisnik može da ih swipuje, zbog toga se instalira paket „react-native-swiper“^[11]. Kreira se komponenta Banner.js i dodaje se u ProductContainer.js fajl.



```
00 -3,0 +3,7 00 import { View, StyleSheet, FlatList, ActivityIndicator } from "react-native";
3 3 import { Container, Header, Icon, Item, Input, Text } from "native-base";
4 4 import ProductList from "../ProductList";
5 5 import SearchedProduct from "../SearchedProducts";
6 6 + import Banner from "../../Shared/Banner";
7 7
7 8 + const data = require("../../assets/products.json");
8 9
00 -55,0 +50,0 00 export default function ProductContainer() {
55 56 <SearchedProduct productsFiltered={productsFiltered}></SearchedProduct>
56 57 ) : {
57 58 <View style={styles.container}>
59 59 + <View>
60 60 + <Banner></Banner>
61 61 + </View>
58 62 <View style={styles.listContainer}>
59 63 <FlatList
60 64 numColumns={2}
```

[11] <https://www.npmjs.com/package/react-native-swiper>

```
65 Shared/Banner.js
... .. 00 -8,0 ~1,05 00
1 + import React, { useState, useEffect } from "react";
2 + import { Image, StyleSheet, Dimensions, View, ScrollView } from "react-native";
3 + import Swiper from "react-native-swiper/src";
4 +
5 + var { width } = Dimensions.get("window");
6 +
7 + export default function Banner() {
8 +   const [bannerData, setBannerData] = useState([]);
9 +
10 +   useEffect(() => {
11 +     setBannerData([
12 +       "https://images.vexels.com/media/users/3/126443/preview2/ff9af1e1edfa2c4a46c43b8c2849ce52-macbook-pro-touch-bar-banner.jpg",
13 +       "https://pbs.twimg.com/media/D7P_yLdXAAAvJMO.jpg",
14 +       "https://www.yardproduct.com/blog/wp-content/uploads/2010/01/gardening-banner.jpg",
15 +     ]);
16 +     return () => {
17 +       setBannerData([]);
18 +     };
19 +   }, []);
20 +
21 +   return (
22 +     <ScrollView>
23 +       <View style={StyleSheet.container}>
24 +         <View style={styles.swiper}>
25 +           <Swiper
26 +             style={{ height: width / 2 }}
27 +             showButtons={false}
28 +             autoplay={true}
29 +             autoplayTimeout={3}
30 +           >
31 +             {bannerData.map((item) => {
32 +               return (
33 +                 <Image
34 +                   key={item}
35 +                   style={styles.imageBanner}
36 +                   resizeMode="contain"
37 +                   source={{ uri: item }}
38 +                 ></Image>
39 +               );
40 +             })}
41 +           </Swiper>
42 +           <View style={{ height: 20 }}></View>
43 +         </View>
44 +       </ScrollView>
45 +     );
46 +   }
47 + }
48 +
49 + const styles = StyleSheet.create({
50 +   container: {
51 +     flex: 1,
52 +     backgroundColor: "gainsboro",
53 +   },
54 +   swiper: {
55 +     width: width,
56 +     alignItems: "center",
57 +     marginTop: 10,
58 +   },
59 +   imageBanner: {
60 +     height: width / 2,
61 +     width: width - 40,
62 +     borderRadius: 10,
63 +     marginHorizontal: 20,
64 +   },
65 + });
```


3.2.5. dodavanje kategorija - added categories

Svaki produkt ima svoju kategoriju, te s tim korisnik može da ih filteruje ukoliko to želi. Potrebno je da se doda selektor za kategoriju iznad svih produkta (CategoryFilter.js) i po korisničkom izboru prikazuje određena kategorija produkta ili svi produkti.

```
66 Screens/Products/CategoryFilter.js
...
1 + import React from "react";
2 + import { StyleSheet, TouchableOpacity, ScrollView } from "react-native";
3 + import { ListItem, Badge, Text } from "native-base";
4 +
5 + export default function CategoryFilter(props) {
6 +   return (
7 +     <ScrollView
8 +       bounce={true}
9 +       horizontal={true}
10 +       style={{ backgroundColor: "#F2F2F2" }}
11 +     >
12 +       <ListItem style={{ margin: 0, padding: 0, borderRadius: 0 }}>
13 +         <TouchableOpacity
14 +           key={1}
15 +           onPress={() => {
16 +             props.changeCategory("all"), props.setActive(-1);
17 +           }}
18 +         >
19 +           <Badge
20 +             style={{
21 +               styles.center,
22 +               { margin: 5 },
23 +               props.active == -1 ? styles.active : styles.inactive,
24 +             }}
25 +           >
26 +             <Text style={{ color: "white" }}>All</Text>
27 +           </Badge>
28 +         </TouchableOpacity>
29 +         {props.categories.map((item) => {
30 +           <TouchableOpacity
31 +             key={item.id}
32 +             onPress={() => {
33 +               props.changeCategory(item.id.Soid),
34 +               props.setActive(props.categories.indexOf(item));
35 +             }}
36 +           >
37 +             <Badge
38 +               style={{
39 +                 styles.center,
40 +                 { margin: 5 },
41 +                 props.active == props.categories.indexOf(item)
42 +                   ? styles.active
43 +                   : styles.inactive,
44 +               }}
45 +             >
46 +               <Text style={{ color: "white" }}>{item.name}</Text>
47 +             </Badge>
48 +           </TouchableOpacity>
49 +         )}
50 +       </ListItem>
51 +     </ScrollView>
52 +   );
53 + }
54 +
55 + const styles = StyleSheet.create({
56 +   center: {
57 +     justifyContent: "center",
58 +     alignItems: "center",
59 +   },
60 +   active: {
61 +     backgroundColor: "#92bafc",
62 +   },
63 +   inactive: {
64 +     backgroundColor: "#a0e1eb",
65 +   },
66 + });
```

```
00 -0,0 +1,05 00
1 + import React from "react";
2 + import { StyleSheet, TouchableOpacity, ScrollView } from "react-native";
3 + import { ListItem, Badge, Text } from "native-base";
4 +
5 + export default function CategoryFilter(props) {
6 +   return (
7 +     <ScrollView
8 +       bounce={true}
9 +       horizontal={true}
10 +       style={{ backgroundColor: "#f2f2f2" }}
11 +     >
12 +       <ListItem style={{ margin: 0, padding: 0, borderRadius: 0 }}>
13 +         <TouchableOpacity
14 +           key={1}
15 +           onPress={() => {
16 +             props.changeCategory("all"), props.setActive(-1);
17 +           }}
18 +         >
19 +           <Badge
20 +             style={{
21 +               styles.center,
22 +               { margin: 5 },
23 +               props.active == -1 ? styles.active : styles.inactive,
24 +             }}
25 +           >
26 +             <Text style={{ color: "white" }}>All</Text>
27 +           </Badge>
28 +         </TouchableOpacity>
29 +         {props.categories.map((item) => {
30 +           <TouchableOpacity
31 +             key={item._id}
32 +             onPress={() => {
33 +               props.changeCategory(item._id.$oid),
34 +               props.setActive(props.categories.indexOf(item));
35 +             }}
36 +           >
37 +             <Badge
38 +               style={{
39 +                 styles.center,
40 +                 { margin: 5 },
41 +                 props.active == props.categories.indexOf(item)
42 +                   ? styles.active
43 +                   : styles.inactive,
44 +               }}
45 +             >
46 +               <Text style={{ color: "white" }}>{item.name}</Text>
47 +             </Badge>
48 +           </TouchableOpacity>
49 +         )}
50 +       </ListItem>
51 +     </ScrollView>
52 +   );
53 + }
54 +
55 + const styles = StyleSheet.create({
56 +   center: {
57 +     justifyContent: "center",
58 +     alignItems: "center",
59 +   },
60 +   active: {
61 +     backgroundColor: "#99baf6",
62 +   },
63 +   inactive: {
64 +     backgroundColor: "#a9c1eb",
65 +   },
66 + });
```

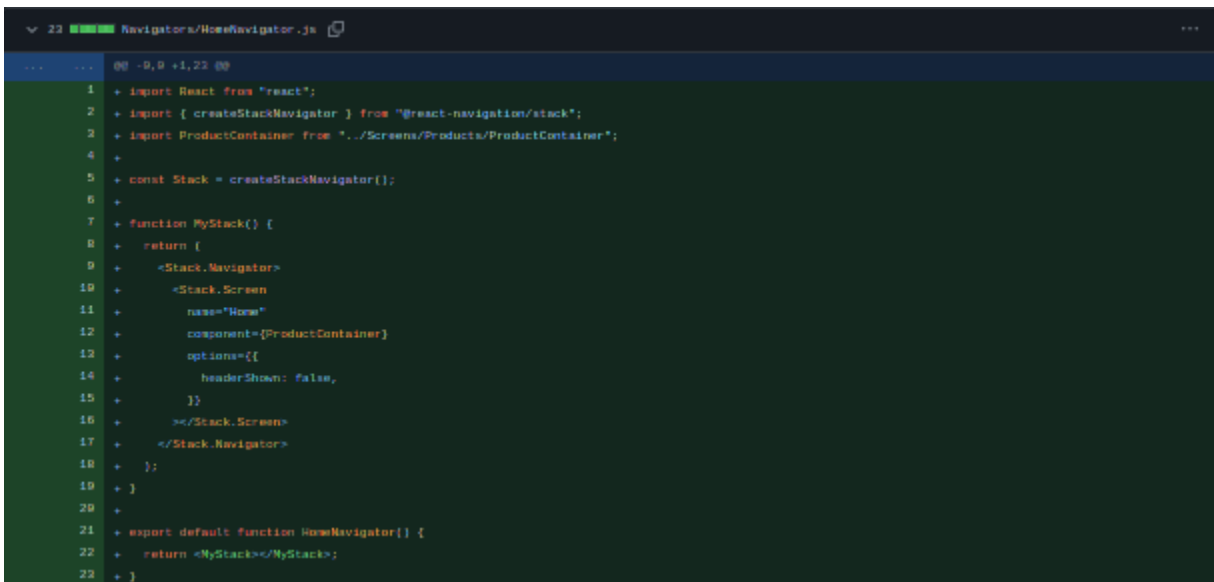
```

00 -90,21 +90,23 @@ export default function ProductContainer() {
90 90     {focus == true ? {
91 91         <SearchedProduct productsFiltered={productsFiltered}></SearchedProduct>
92 92     } : {
93 93         <View style={styles.container}>
94 94             <ScrollView>
95 95                 <View>
96 96                     <Banner></Banner>
97 97                 </View>
98 98                 <View style={styles.listContainer}>
99 99                     <FlatList
100 100                         numColumns={2}
101 101                         data={products}
102 102                         renderItem={({ item }) => {
103 103                             <ProductList key={item.brand} item={item}></ProductList>
104 104                         }}
105 105                         keyExtractor={({item}) => item.brand}
106 106                     ></FlatList>
107 107                 <View>
108 108                     <Banner></Banner>
109 109                 </View>
110 110                 <View>
111 111                     <CategoryFilter
112 112                         categories={categories}
113 113                         changeCategory={changeCategory}
114 114                         productsCtg={productsCtg}
115 115                         active={active}
116 116                         setActive={setActive}
117 117                     ></CategoryFilter>
118 118                 </View>
119 119                 {productsCtg.length > 0 ? (
120 120                     <View style={styles.listContainer}>
121 121                         {productsCtg.map((item) => {
122 122                             return <ProductList key={item.name} item={item} />;
123 123                         })}
124 124                     </View>
125 125                 ) : (
126 126                     <View style={styles.center, { height: height / 2 }}>
127 127                         <Text>No products found</Text>
128 128                     </View>
129 129                 )}
130 130             </View>
131 131         </ScrollView>
132 132     }
133 133 </Container>
134 134 };
135 135
136 136 @@ -90,7 +127,7 @@ const styles = StyleSheet.create({
137 137     backgroundColor: "gainsboro",
138 138 },
139 139 listContainer: {
140 140     width: "100%",
141 141     height: height,
142 142     flex: 1,
143 143     flexDirection: "row",
144 144     alignItems: "flex-start",

```

3.2.6. dodavanje navigacije - added navigation

Aplikacija za sada ima samo prikaz početne strane. Pored početne strane potrebno je još stvari kao što je korpa, korisnički panel i admin panel za admina. Tu navigaciju moguće je dodati korišćenjem paketa „react-navigation“^[12]. Kroz aplikaciju se koriste različite vrste navigacije pa je zbog toga potrebno instalirati paket „@react-native-navigation/bottom-tabs“, „@react-native-navigation/native“ i „@react-native-navigation/stack“. Pored ovih paketa instalira se i paket „react-native-vector-icons“^[13] da bi ljepše stilizovao bottom navigation bar. Nakon svega instaliranog se dodaju spomenute rute u Main.js fajl koji se kreira u Navigators folderu, te se importuje Main navigator u App.js.



```
1 + import React from "react";
2 + import { createStackNavigator } from "@react-navigation/stack";
3 + import ProductContainer from "../Screens/Products/ProductContainer";
4 +
5 + const Stack = createStackNavigator();
6 +
7 + function MyStack() {
8 +   return (
9 +     <Stack.Navigator>
10 +     <Stack.Screen
11 +       name="Home"
12 +       component={ProductContainer}
13 +       options={{
14 +         headerShown: false,
15 +       }}
16 +     ></Stack.Screen>
17 +   </Stack.Navigator>
18 + );
19 + }
20 +
21 + export default function HomeNavigator() {
22 +   return <MyStack/>
23 + }
```

[12] <https://www.npmjs.com/package/react-navigation>

[13] <https://www.npmjs.com/package/react-native-vector-icons>

```
Navigator/Main.js
00 -0,0 +1,02 00
1 + import React from "react";
2 + import { createBottomTabNavigator } from "react-navigation/bottom-tabs";
3 + import { View } from "react-native";
4 + import Icon from "react-native-vector-icons/FontAwesome";
5 + import HomeNavigator from "../HomeNavigator";
6 +
7 + const Tab = createBottomTabNavigator();
8 +
9 + export default function Main() {
10 +   return (
11 +     <Tab.Navigator
12 +       initialRouteName="Home"
13 +       tabBarOptions={{
14 +         keyboardHidesTabBar: true,
15 +         showLabel: false,
16 +         activeTintColor: "#912E03",
17 +       }}
18 +     >
19 +       <Tab.Screen
20 +         name="Home"
21 +         component={HomeNavigator}
22 +         options={{
23 +           tabBarIcon: ({ color }) => [
24 +             <Icon
25 +               name="home"
26 +               style={{ position: "relative" }}
27 +               color={color}
28 +               size={30}
29 +             ></Icon>
30 +           ],
31 +         }}
32 +       ></Tab.Screen>
33 +       <Tab.Screen
34 +         name="Cart"
35 +         component={HomeNavigator}
36 +         options={{
37 +           tabBarIcon: ({ color }) => [
38 +             <Icon name="shopping-cart" color={color} size={30}></Icon>
39 +           ],
40 +         }}
41 +       ></Tab.Screen>
42 +       <Tab.Screen
43 +         name="Admin"
44 +         component={HomeNavigator}
45 +         options={{
46 +           tabBarIcon: ({ color }) => [
47 +             <Icon name="cog" color={color} size={30}></Icon>
48 +           ],
49 +         }}
50 +       ></Tab.Screen>
51 +       <Tab.Screen
52 +         name="User"
53 +         component={HomeNavigator}
54 +         options={{
55 +           tabBarIcon: ({ color }) => [
56 +             <Icon name="user" color={color} size={30}></Icon>
57 +           ],
58 +         }}
59 +       ></Tab.Screen>
60 +     </Tab.Navigator>
61 +   );
62 + }
```

```
App.js
00 -2,12 +3,15 00
1 import React from "react";
2 import { StyleSheet, Text, View } from "react-native";
3 import ProductContainer from "../Screens/Products/ProductContainer";
4 import Header from "../Shared/Header";
5 import { NavigationContainer } from "react-navigation/native";
6 import Main from "../Navigators/Main";
7
8 export default function App() {
9   return (
10     <View style={styles.container}>
11       <NavigationContainer>
12         <Header/></Header>
13         <ProductContainer/></ProductContainer>
14       </NavigationContainer>
15     </View>
16   );
17 }
```

3.2.7. dodavanje pojedinačnih produkta, navigacija i komponente - added single product navigation and components

Kada se odabere neki produkt sa početne strane ili iz searcha, potrebno je da korisniku bude prikazan ekran sa opisom tog produkta i svim slikama. Za to se dodaje SingleProduct.js komponenta i u HomeNavigatoru se dodaje ruta za single product. Takođe se dodaje funkcionalnost na početnoj strani i searchu za navigaciju.

```
Screen/Products/SingleProduct.js
1  + import React, { useState, useEffect } from "react";
2  + import {
3  +   Image,
4  +   View,
5  +   StyleSheet,
6  +   Text,
7  +   ScrollView,
8  +   Button,
9  + } from "react-native";
10 + import { Left, Right, Container, H1 } from "native-base";
11 +
12 + export default function SingleProduct(props) {
13 +   const [item, setItem] = useState(props.route.params.item);
14 +   const [availability, setAvailability] = useState(false);
15 +
16 +   return (
17 +     <Container style={styles.container}>
18 +       <ScrollView style={{ marginBottom: 10, padding: 1 }}>
19 +         <Image
20 +           <Image
21 +             source={{
22 +               uri: item.image
23 +               ? item.image
24 +               : "https://cdn.shopify.com/photos/2012/04/01/17/29/bca-22949_562_720.jpg",
25 +             }}
26 +             resizeMode="contain"
27 +             style={styles.image}
28 +           />
29 +         </Image>
30 +         <View style={styles.contentContainer}>
31 +           <H1 style={styles.contentHeader}>{item.name}</H1>
32 +           <Text style={styles.contentText}>{item.brand}</Text>
33 +         </View>
34 +       </ScrollView>
35 +       <View style={styles.buttonContainer}>
36 +         <Left>
37 +           <Text style={styles.price}>${ item.price}</Text>
38 +         </Left>
39 +         <Right>
40 +           <Button title="Add"></Button>
41 +         </Right>
42 +       </View>
43 +     </Container>
44 +   );
45 + }
46 +
47 + const styles = StyleSheet.create([
48 +   container: {
49 +     position: "relative",
50 +     height: "100%",
51 +   },
52 +   imageContainer: {
53 +     backgroundColor: "white",
54 +     padding: 0,
55 +     margin: 4,
56 +   },
57 +   image: {
58 +     width: "100%",
59 +     height: 250,
60 +   },
61 +   contentContainer: {
62 +     marginTop: 20,
63 +     justifyContent: "center",
64 +     alignItems: "center",
65 +   },
66 +   contentHeader: {
67 +     fontWeight: "bold",
68 +     marginBottom: 20,
69 +   },
70 +   contentText: {
71 +     fontSize: 18,
72 +     fontWeight: "bold",
73 +     marginBottom: 20,
74 +   },
75 +   buttonContainer: {
76 +     flexDirection: "row",
77 +     position: "absolute",
78 +     bottom: 4,
79 +     left: 4,
80 +     backgroundColor: "white",
81 +   },
82 +   price: {
83 +     fontSize: 24,
84 +     margin: 20,
85 +     color: "red",
86 +   },
87 + ]);
```

```

15 Screens/Products/ProductContainer.js
17 17 @@ -17,7 +17,7 @@ var [ height ] = Dimensions.get("window");
18 18   const data = require("../assets/products.json");
19 19   const categoriesData = require("../assets/categories.json");
20 20   - export default function ProductContainer() {
21 21   + export default function ProductContainer(props) {
22 22     const [products, setProducts] = useState([]);
23 23     const [productsFiltered, setProductsFiltered] = useState([]);
24 24     const [focus, setFocus] = useState();
25 25
26 26   @@ -31,7 +31,10 @@ export default function ProductContainer() {
27 27
28 28     </Item>
29 29     </Header>
30 30     {focus == true ?
31 31     - <SearchProduct productsFiltered={productsFiltered}/></SearchProducts>
32 32     + <SearchProduct
33 33     +   productsFiltered={productsFiltered}
34 34     +   navigation={props.navigation}
35 35     + ></SearchProducts>
36 36     } : (
37 37     <ScrollView
38 38     </Item>
39 39
40 40   @@ -107,7 +110,12 @@ export default function ProductContainer() {
41 41     (products.length > 0 ?
42 42     </New style={styles.listContainer}>
43 43     {products.map((item) => {
44 44     -   return <ProductList key={item.name} item={item} />;
45 45     +   return (
46 46     +     <ProductList
47 47     +       key={item.name}
48 48     +       item={item}
49 49     +       navigation={props.navigation}
50 50     +     />
51 51     +   );
52 52     })}
53 53     </View>
54 54     ) : {

```

```

7 Screens/Products/ProductList.js
7 7   export default function ProductList(props) {
8 8     const [ item ] = props;
9 9     return (
10 10     - <TouchableOpacity style={{ width: "50%" }}>
11 11     + <TouchableOpacity
12 12     +   style={{ width: "50%" }}
13 13     +   onPress={() =>
14 14     +     props.navigation.navigate("Product Detail", { item: item })
15 15     +   }
16 16     + >
17 17     </View style={{ width: width / 2, backgroundColor: "grey600" }}>
18 18     <ProductCard {...item}></ProductCard>
19 19     </View>

```

```

8 Screens/Products/SearchedProducts.js
11 11   @@ -11,7 +11,12 @@ export default function SearchedProducts(props) {
12 12     (productsFiltered.length > 0 ?
13 13     productsFiltered.map((item) => [
14 14     -   <ListItem key={item._id.$id} asset>
15 15     +   <ListItem
16 16     +     key={item._id.$id}
17 17     +     avatar
18 18     +     onPress={() =>
19 19     +       props.navigation.navigate("Product Detail", { item: item })
20 20     +     }
21 21     + >
22 22     </ListItem>
23 23     </Thumbnail
24 24     search={

```

```
▼ 8 Navigators/HomeNavigator.js
... 1 1 import React from "react";
2 2 import { createStackNavigator } from "@react-navigation/stack";
3 3 import ProductContainer from "../Screens/Products/ProductContainer";
4 4 + import SingleProduct from "../Screens/Products/SingleProduct";
5 5
6 6 const Stack = createStackNavigator();
7 7
8 8 @@@ -14,6 +15,13 @@@ function MyStack() {
14 15     headerShown: false,
15 16     }}
16 17     ></Stack.Screen>
18 18 +     <Stack.Screen
19 19 +       name="Product Detail"
20 20 +       component={SingleProduct}
21 21 +       options={{
22 22 +         headerShown: false,
23 23 +       }}
24 24 +     ></Stack.Screen>
17 25   </Stack.Navigator>
18 26   );
19 27   }
```

3.2.8. dodavanje reduxa i podešavanje akcija i statea za korpu - added redux and set up cart state and actions

Ukoliko se podaci o korpi budu držali u top level state-u, kod ove aplikacije bi postao poprilično nečitljiv i vrlo lako bi se pravile greške koje je kasnije teško popraviti. Da bi se to pojednostavilo dodaje se alatka za state management pod nazivom „Redux“^[14]. Ona omogućava da se state odvoji od specifične komponente. Da bi se redux dodao u aplikaciju potrebno je instalirati pakete „react-redux“, „redux“ i „redux-thunk“. Kreira se store u kojem će se čuvati podaci o korpi te se kreiraju akcije koje korisnik može da izvrši, a to su: ADD_TO_CART, REMOVE_FROM_CART i CLEAR_CART. Te akcije se šalju u reducer koji prima action.type i payload kojim mijenja state. Nakon kreiranja svega kompletna aplikacija se wrapuje u Provider komponentu iz react-redux biblioteke.

```
▼ 12 App.js
... 5 5 import ProductContainer from "../Screens/Products/ProductContainer";
6 6 import Header from "../Shared/Header";
7 7 import { NavigationContainer } from "react-navigation/native";
8 8 import Main from "../Navigators/Main";
9 9 + import { Provider } from "react-redux";
10 10 + import store from "../Redux/Store";
11 11
12 12 export default function App() {
13 13   return (
14 14     <NavigationContainer>
15 15 +     <Provider store={store}>
16 16 +     <NavigationContainer>
17 17 +     <Header/></Header>
18 18 +     <Main/></Main>
19 19 +     </NavigationContainer>
20 20 +   </Provider>
21 21 );
22 22 }
23 23 }
```

[14] <https://www.npmjs.com/package/redux>


```
21 Redux/Actions/cartActions.js
... 00 -9,9 +1,21 00
1 + import { ADD_TO_CART, REMOVE_FROM_CART, CLEAR_CART } from "../constants";
2 +
3 + export const addToCart = (payload) => {
4 +   return {
5 +     type: ADD_TO_CART,
6 +     payload,
7 +   };
8 + };
9 +
10 + export const removeFromCart = (payload) => {
11 +   return {
12 +     type: REMOVE_FROM_CART,
13 +     payload,
14 +   };
15 + };
16 +
17 + export const removeFromCart = () => {
18 +   return {
19 +     type: CLEAR_CART,
20 +   };
21 + };
```

```
15 Redux/Reducers/cartItems.js
... 00 -9,9 +1,15 00
1 + import { ADD_TO_CART, REMOVE_FROM_CART, CLEAR_CART } from "../constants";
2 +
3 + const cartItems = (state = [], action) => {
4 +   switch (action.type) {
5 +     case ADD_TO_CART:
6 +       return [...state, action.payload];
7 +     case REMOVE_FROM_CART:
8 +       return state.filter((cartItem) => cartItem !== action.payload);
9 +     case CLEAR_CART:
10 +       return (state = []);
11 +   }
12 +   return state;
13 + };
14 +
15 + export default cartItems;
```

```
15 Redux/Store.js
... 00 -9,9 +1,15 00
1 + import { createStore, combineReducers, applyMiddleware } from "redux";
2 + import thunkMiddleware from "redux-thunk";
3 + import { composeWithDevTools } from "redux-devtools-extension/developmentOnly";
4 + import cartItems from "./Reducers/cartItems";
5 +
6 + const reducers = combineReducers({
7 +   cartItems: cartItems,
8 + });
9 +
10 + const store = createStore(
11 +   reducers,
12 +   composeWithDevTools(applyMiddleware(thunkMiddleware))
13 + );
14 +
15 + export default store;
```

```
2 Redux/constants.js
... 00 -9,9 +1,3 00
1 + export const ADD_TO_CART = "ADD_TO_CART";
2 + export const REMOVE_FROM_CART = "REMOVE_FROM_CART";
3 + export const CLEAR_CART = "CLEAR_CART";
```

3.2.9. dodavanje funkcionalnosti korpi - added functionality to cart

U ovom koraku potrebno je kreirati prikaz za korpu i takođe uraditi funkcionalnost dodavanja i brisanja produkta. Na prikazu pojedinačnih produkta potrebno je da dugme add to cart dispatchuje akciju ADD_TO_CART koja za payload ima objekat koji sadrži podatke o produktu i kvantitet.

```
1  import { connect } from "react-redux";
2  import { bindActionCreators } from "redux";
3  import { actions } from "../Redux/Actions/CartActions";
4
5  const mapStateToProps = (state) => {
6    return {
7      cart: state.cart,
8    };
9  };
10
11  const mapDispatchToProps = (dispatch) => {
12    return {
13      addToCart: bindActionCreators(actions.addToCart, dispatch),
14    };
15  };
16
17  @connect(mapStateToProps, mapDispatchToProps)
18  export default function ProductCard(props) {
19    const { name, price, image, cartInStock } = props;
20
21    return (
22      <View style={styles.container}>
23        <Text style={styles.name}>{name}</Text>
24        <Text style={styles.price}>{price}</Text>
25        <Text style={styles.image}>{image}</Text>
26        <Text style={styles.inStock}>{cartInStock}</Text>
27        <Button title="Add" color="green"></Button>
28      </View>
29    );
30  }
31
32  export default connect(mapStateToProps, mapDispatchToProps)(ProductCard);
```

```
1  import { connect } from "react-redux";
2  import { bindActionCreators } from "redux";
3  import { actions } from "../Redux/Actions/CartActions";
4
5  const mapStateToProps = (state) => {
6    return {
7      cart: state.cart,
8    };
9  };
10
11  const mapDispatchToProps = (dispatch) => {
12    return {
13      addToCart: bindActionCreators(actions.addToCart, dispatch),
14    };
15  };
16
17  @connect(mapStateToProps, mapDispatchToProps)
18  export default function SingleProduct(props) {
19    const { item, cartItem } = useState(props.rest.item);
20    const [availability, setAvailability] = useState(true);
21
22    return (
23      <View style={styles.container}>
24        <Text style={styles.name}>{item.name}</Text>
25        <Text style={styles.price}>${item.price}</Text>
26        <Text style={styles.image}>{item.image}</Text>
27        <Text style={styles.inStock}>{cartItem}</Text>
28        <Text style={styles.availability}>{availability}</Text>
29        <Button title="Add" color="green"></Button>
30      </View>
31    );
32  }
33
34  export default connect(mapStateToProps, mapDispatchToProps)(SingleProduct);
```

```
1 + import React from "react";
2 + import { StyleSheet } from "react-native";
3 + import { Badge, Text } from "native-base";
4 +
5 + import { connect } from "react-redux";
6 +
7 + function CartIcon(props) {
8 +   return (
9 +     <
10 +       {props.cartItems.length ? (
11 +         <Badge style={styles.badge}>
12 +           <Text style={styles.text}>{props.cartItems.length}</Text>
13 +         </Badge>
14 +       ) : null}
15 +     </>
16 +   );
17 + }
18 +
19 + const mapStateToProps = (state) => {
20 +   const { cartItems } = state;
21 +   return {
22 +     cartItems: cartItems,
23 +   };
24 + };
25 +
26 + const styles = StyleSheet.create({
27 +   badge: {
28 +     width: 25,
29 +     position: "absolute",
30 +     flex: 1,
31 +     justifyContent: "center",
32 +     alignItems: "center",
33 +     alignContent: "center",
34 +     top: -4,
35 +     right: -15,
36 +   },
37 +   text: {
38 +     fontSize: 12,
39 +     width: 100,
40 +     fontWeight: "bold",
41 +   },
42 + });
43 +
44 + export default connect(mapStateToProps)(CartIcon);
```

42 Screens/Cart/CartItem.js

```
... .. @@ -0,0 +1,42 @@
1 + import React, { useState } from "react";
2 + import { StyleSheet } from "react-native";
3 + import { Text, Right, Left, ListItem, Thumbnail, Body } from "native-base";
4 +
5 + export default function CartItem(props) {
6 +   const item = props.item.item.product;
7 +   const [quantity, setQuantity] = useState(props.item.quantity);
8 +   return (
9 +     <ListItem style={styles.listItem} key={Math.random()} avatar>
10 +       <Left>
11 +         <Thumbnail
12 +           source={{
13 +             uri: item.image
14 +             ? item.image
15 +             : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23649_900_720.png",
16 +           }}
17 +         >>/Thumbnail>
18 +       </Left>
19 +       <Body style={styles.body}>
20 +         <Left>
21 +           <Text>{item.name}</Text>
22 +         </Left>
23 +         <Right>
24 +           <Text>${ item.price}</Text>
25 +         </Right>
26 +       </Body>
27 +     </ListItem>
28 +   );
29 + }
30 +
31 + const styles = StyleSheet.create({
32 +   listItem: {
33 +     alignItems: "center",
34 +     backgroundColor: "white",
35 +     justifyContent: "center",
36 +   },
37 +   body: {
38 +     margin: 10,
39 +     alignItems: "center",
40 +     flexDirection: "row",
41 +   },
42 + });
```

6 Screens/Cart/Checkout.js

```
... .. @@ -0,0 +1,6 @@
1 + import React from "react";
2 + import { View } from "react-native";
3 +
4 + export default function Checkout() {
5 +   return <View></View>;
6 + }
```

```
Screen/Cart/Cart.js
... 88 -4,0 -1,133 88
1 + import React from "react";
2 + import {
3 +   View,
4 +   Dimensions,
5 +   StyleSheet,
6 +   Button,
7 +   TouchableOpacity,
8 + } from "react-native";
9 + import {
10 +   Container,
11 +   Text,
12 +   Left,
13 +   Right,
14 +   H1,
15 +   ListItem,
16 +   Thumbnail,
17 +   Body,
18 + } from "react-native-base";
19 + import CartItem from "../CartItem";
20 + import { SwipeListView } from "react-native-swipe-list-view";
21 +
22 + import Icon from "react-native-vector-icons/FontAwesome";
23 +
24 + import { connect } from "react-redux";
25 + import * as actions from "../../Redux/Actions/CartActions";
26 +
27 + var { height, width } = Dimensions.get("window");
28 +
29 + function Cart(props) {
30 +   var total = 0;
31 +   props.cartItems.forEach((cart) => {
32 +     return (total += cart.product.price);
33 +   });
34 +
35 +   return (
36 +     <>
37 +       <Container style={styles.container} ><Cart/>
38 +       <SwipeListView
39 +         data={props.cartItems}
40 +         renderItem={(data) => <CartItem item={data}/><CartItem>
41 +         renderHiddenItem={(data) => {
42 +           <View style={styles.hiddenContainer}
43 +             onPress={() => props.removeFromCart(data.item)}
44 +             >
45 +             <TouchableOpacity style={styles.hiddenButton}
46 +               onPress={() => props.removeFromCart(data.item)}
47 +             >
48 +               <Icon name="trash" color="white" size={30}/><Icon>
49 +             </TouchableOpacity>
50 +           </View>
51 +         }
52 +         disableRightSwipe={true}
53 +         previewOpenDelay={3000}
54 +         friction={1000}
55 +         bounces={40}
56 +         leftOpenValue={75}
57 +         stopLeftSwipe={75}
58 +         rightOpenValue={-75}
59 +       ></SwipeListView>
60 +       <View style={styles.bottomContainer}
61 +         <Left>
62 +           <Text style={styles.price}>$ {total}</Text>
63 +         </Left>
64 +         <Right>
65 +           <Button title="Clear" onPress={() => props.clearCart()}></Button>
66 +         </Right>
67 +         <Right>
68 +           <Button title="Checkout"
69 +             onPress={() => props.navigation.navigate("Checkout")}
70 +           ></Button>
71 +         </Right>
72 +       </View>
73 +       <Text>Looks like your cart is empty</Text>
74 +       <Text>Add products to your cart to get started</Text>
75 +     </Container>
76 +   );
77 + }
78 +
79 + const mapStateToProps = state => ({
80 +   cartItems: state.cart.items,
81 +   total: state.cart.total,
82 + });
83 +
84 + export default connect(mapStateToProps)(Cart);
```

```
99 + create mapStateToProps = (state) => {
99 +   create { cartItems } = state;
99 +   return {
99 +     cartItems: cartItems,
99 +   };
99 + };
99 +
99 + create mapDispatchToProps = (dispatch) => [
99 +   return {
99 +     clearCart: () => dispatch(actions.clearCart()),
99 +     removeFromCart: (item) => dispatch(actions.removeFromCart(item)),
99 +   };
99 + ];
99 +
99 + create styles = StyleSheet.create([
99 +   emptyContainer: {
99 +     height: height,
99 +     alignItems: "center",
99 +     justifyContent: "center",
99 +   },
99 +   bottomContainer: {
99 +     flexDirection: "row",
99 +     position: "absolute",
99 +     bottom: 5,
99 +     left: 5,
99 +     backgroundColor: "white",
99 +     elevation: 20,
99 +   },
99 +   price: {
99 +     fontSize: 18,
99 +     margin: 20,
99 +     color: "red",
99 +   },
99 +   headerContainer: {
99 +     flex: 1,
99 +     justifyContent: "flex-end",
99 +     flexDirection: "row",
99 +   },
99 +   headerButton: {
99 +     backgroundColor: "red",
99 +     justifyContent: "center",
99 +     alignItems: "flex-end",
99 +     paddingRight: 10,
99 +     height: 70,
99 +     width: width / 1.2,
99 +   },
99 + ]);
99 +
99 + export default connect(mapStateToProps, mapDispatchToProps)(Cart);
```

```
▼ 31 Navigators/CartNavigator.js
... 00 -9,0 +1,21 00
1 + import React from "react";
2 + import { createStackNavigator } from "@react-navigation/stack";
2 + import Cart from "../Screens/Cart/Cart";
4 + import Checkout from "../Screens/Cart/Checkout";
5 +
6 + const Stack = createStackNavigator();
7 +
8 + function MyStack() {
9 +   return (
10 +     <Stack.Navigator>
11 +     <Stack.Screen
12 +       name="Cart"
13 +       component={Cart}
14 +       options={{
15 +         headerShown: false,
16 +       }}
17 +     >>/Stack.Screen>
18 +     <Stack.Screen
19 +       name="Checkout"
20 +       component={Checkout}
21 +       options={{
22 +         title: "Checkout",
23 +       }}
24 +     >>/Stack.Screen>
25 +   </Stack.Navigator>
26 + );
27 + }
28 +
29 + export default function CartNavigator() {
30 +   return <MyStack></MyStack>;
31 + }
```

```
▼ 9 Navigators/Main.js
... 00 -2,6 +2,8 00 import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
2 import { View } from "react-native";
4 import Icon from "react-native-vector-icons/FontAwesome";
5 import HomeNavigator from "../HomeNavigator";
6 + import CartNavigator from "../CartNavigator";
7 + import CartIcon from "../Shared/CartIcon";
8
9 const Tab = createBottomTabNavigator();
10
11 export default function Main() {
12
13   </Tab.Screen>
14   <Tab.Screen
15     name="Cart"
16     component={HomeNavigator}
17     component={CartNavigator}
18     options={{
19       tabBarIcon: ({ color }) => {
20         <Icon name="shopping-cart" color={color} size={30}></Icon>
21         <View>
22           <Icon name="shopping-cart" color={color} size={30}></Icon>
23           <CartIcon></CartIcon>
24         </View>
25       },
26     }}
27   >>/Tab.Screen>
28 }
```

```
▼ 2 Redux/Actions/cartActions.js
... 00 -14,7 +14,7 00 export const removeFromCart = (payload) => {
14
15 };
16
17 export const removeFromCart = () => {
17 + export const clearCart = () => {
18   return {
19     type: CLEAR_CART,
20   };
21 }
```

3.2.10. završavanje checkout stranice - finished checkout page

Nakon popunjavanja korpe, korisnik treba da kompletira svoju narudžbu kroz checkout proces. Za to je potrebno dodati posebne screenove gdje će korisnik da unese svoje lične podatke za dostavu i gdje će da plati za proizvode. Za to sve se dodaju navigacija i komponente.

```
▼ 4 Navigators/CartNavigator.js
... -1,7 +1,7
1 1 import React from "react";
2 2 import { createStackNavigator } from "@react-navigation/stack";
3 3 import Cart from "../Screens/Cart/Cart";
4 4 - import Checkout from "../Screens/Cart/Checkout";
4 4 + import CheckoutNavigator from "../CheckoutNavigator";
5 5
6 6 const Stack = createStackNavigator();
7 7
... -17,7 +17,7
function MyStack() {
17 17 ></Stack.Screen>
18 18 <Stack.Screen
19 19   name="Checkout"
20 20 -   component={Checkout}
20 20 +   component={CheckoutNavigator}
21 21   options={{
22 22     title: "Checkout",
23 23   }}
}

▼ 21 Navigators/CheckoutNavigator.js
... -0,0 +1,21
1 1 + import React from "react";
2 2 + import { createMaterialTopTabNavigator } from "@react-navigation/material-top-tabs";
3 3 + import Checkout from "../Screens/Cart/Checkout";
4 4 + import Payment from "../Screens/Cart/Payment";
5 5 + import Confirm from "../Screens/Cart/Confirm";
6 6 +
7 7 + const Tab = createMaterialTopTabNavigator();
8 8 +
9 9 + function MyTabs() {
10 10 +   return {
11 11 +     <Tab.Navigator>
12 12 +       <Tab.Screen name="Shipping" component={Checkout}></Tab.Screen>
13 13 +       <Tab.Screen name="Payment" component={Payment}></Tab.Screen>
14 14 +       <Tab.Screen name="Confirm" component={Confirm}></Tab.Screen>
15 15 +     </Tab.Navigator>
16 16 +   };
17 17 + }
18 18 +
19 19 + export default function CheckoutNavigator() {
20 20 +   return <MyTabs></MyTabs>;
21 21 + }
```



```
115 Screens/Cart/Checkout.js
... 00 -1,0 +1,113 00
1 - import React from "react";
2 - import { View } from "react-native";
3 + import React, { useEffect, useState } from "react";
4 + import { Text, View, Button } from "react-native";
5 + import { Item, Picker } from "native-base";
6 + import Icon from "react-native-vector-icons/FontAwesome";
7 + import FormContainer from "../../Shared/Form/FormContainer";
8 + import Input from "../../Shared/Form/Input";
9 + import { KeyboardAwareScrollView } from "react-native-keyboard-aware-scroll-view";
10 + import { connect } from "react-redux";
11
12
13
14 - export default function Checkout() {
15 -   return <View></View>;
16
17 + const countries = require("../../assets/countries.json");
18 +
19 + function Checkout(props) {
20 +   const [orderItems, setOrderItems] = useState();
21 +   const [address, setAddress] = useState();
22 +   const [address2, setAddress2] = useState();
23 +   const [city, setCity] = useState();
24 +   const [zip, setZip] = useState();
25 +   const [country, setCountry] = useState();
26 +   const [phone, setPhone] = useState();
27 +
28 +   useEffect(() => {
29 +     setOrderItems(props.cartItems);
30 +
31 +     return () => {
32 +       setOrderItems();
33 +     };
34 +   }, []);
35 +
36 +   const checkOut = () => {
37 +     let order = {
38 +       city,
39 +       country,
40 +       dateOrdered: Date.now(),
41 +       orderItems,
42 +       phone,
43 +       shippingAddress1: address,
44 +       shippingAddress2: address2,
45 +       zip,
46 +     };
47 +
48 +     props.navigation.navigate("Payment", { order: order });
49 +   };
50 +
51 + }
```

```

42 +
43 +   return (
44 +     <KeyboardAwareScrollView
45 +       viewIsInsideTabBar={true}
46 +       extraHeight={200}
47 +       enableOnAndroid={true}
48 +     >
49 +       <FormContainer title={"Shipping Address"}>
50 +         <Input
51 +           placeholder={"Phone"}
52 +           name={"phone"}
53 +           value={phone}
54 +           keyboardType={"numeric"}
55 +           onChangeText={(text) => setPhone(text)}
56 +         />
57 +         <Input
58 +           placeholder={"Shipping Address 1"}
59 +           name={"ShippingAddress1"}
60 +           value={address}
61 +           onChangeText={(text) => setAddress(text)}
62 +         />
63 +         <Input
64 +           placeholder={"Shipping Address 2"}
65 +           name={"ShippingAddress2"}
66 +           value={address2}
67 +           onChangeText={(text) => setAddress2(text)}
68 +         />
69 +         <Input
70 +           placeholder={"City"}
71 +           name={"city"}
72 +           value={city}
73 +           onChangeText={(text) => setCity(text)}
74 +         />
75 +         <Input
76 +           placeholder={"Zip Code"}
77 +           name={"zip"}
78 +           value={zip}
79 +           keyboardType={"numeric"}
80 +           onChangeText={(text) => setZip(text)}
81 +         />
82 +         <Item picker>
83 +           <Picker
84 +             mode="dropdown"
85 +             iosIcon={<Icon name="arrow-down" color={"#007aff"} />}
86 +             style={{ width: undefined, height: 50 }}
87 +             selectedValue={country}
88 +             placeholder="Select your country"
89 +             placeholderStyle={{ color: "#007aff" }}
90 +             placeholderIconColor="#007aff"
91 +             onValueChange={(e) => setCountry(e)}
92 +           >
93 +             {countries.map((c) => {
94 +               return <Picker.Item key={c.code} label={c.name} value={c.name} />;
95 +             })}
96 +           </Picker>
97 +         </Item>
98 +         <View style={{ marginTop: 20, width: "100%", alignItems: "center" }}>
99 +           <Button title="Confirm" onPress={() => checkout()} />
100 +         </View>
101 +       </FormContainer>
102 +     </KeyboardAwareScrollView>
103 +   );
104 + }
105 +
106 + const mapStateToProps = (state) => {
107 +   const { cartItems } = state;
108 +   return {
109 +     cartItems: cartItems,
110 +   };
111 + };
112 +
113 + export default connect(mapStateToProps)(Checkout);

```

```

... 88 -4,0 -1,200 00
1 + import React from "react";
2 + import { View, StyleSheet, Dimensions, ScrollView, Button } from "react-native";
3 + import { Text, Left, Right, ListItem, Thumbnail, Body } from "native-base";
4 + import { connect } from "react-redux";
5 + import * as actions from "../Redux/Actions/cartActions";
6 +
7 + var { width, height } = Dimensions.get("window");
8 +
9 +
10 + function Confirm(props) {
11 +   const confirmOrder = () => {
12 +     setTimeout(() => {
13 +       props.clearCart();
14 +       props.navigation.navigate("Cart");
15 +     });
16 +   };
17 +   const confirm = props.route.params;
18 +   return (
19 +     <ScrollView contentContainerStyle={styles.container}>
20 +       <View style={styles.titleContainer}>
21 +         <Text style={{ fontSize: 20, fontWeight: "bold" }}>Confirm Order</Text>
22 +       </View>
23 +       {props.route.params ? (
24 +         <View
25 +           style={{
26 +             borderWidth: 1,
27 +             borderColor: "orange",
28 +           }}
29 +         >
30 +           <Text style={styles.title}>Shipping to:</Text>
31 +           <View style={{ padding: 8 }}>
32 +             <Text>Address: {confirm.order.order.shippingAddress}</Text>
33 +             <Text>Address 2: {confirm.order.order.shippingAddress2}</Text>
34 +             <Text>City: {confirm.order.order.city}</Text>
35 +             <Text>Zip Code: {confirm.order.order.zip}</Text>
36 +             <Text>Country: {confirm.order.order.country}</Text>
37 +           </View>
38 +           <Text style={styles.title}>Items:</Text>
39 +           {confirm.order.order.orderItems.map((i) => {
40 +             return (
41 +               <ListItem style={styles.listItem} key={i.product.name} >
42 +                 <Left>
43 +                   <Thumbnail source={{ uri: i.product.image }}></Thumbnail>
44 +                   </Left>
45 +                   <Body style={styles.body}>
46 +                     <Left>
47 +                       <Text>{i.product.name}</Text>
48 +                     </Left>
49 +                     <Right>
50 +                       <Text>${i.product.price}</Text>
51 +                     </Right>
52 +                   </Body>
53 +                 </ListItem>
54 +               );
55 +             });
56 +           </View>
57 +         ) : null}
58 +       <View style={{ alignItems: "center", margin: 20 }}>
59 +         <Button title="Place order" onPress={confirmOrder}></Button>
60 +       </View>
61 +     </ScrollView>
62 +   );
63 + }
64 +
65 + const styles = StyleSheet.create([
66 +   container: {
67 +     height: height,
68 +     padding: 8,
69 +     alignItems: "center",
70 +     backgroundColor: "white",
71 +   },
72 +   titleContainer: {
73 +     justifyContent: "center",
74 +     alignItems: "center",
75 +     margin: 8,
76 +   },
77 +   title: {
78 +     alignSelf: "center",
79 +     margin: 8,
80 +     fontSize: 20,
81 +     fontWeight: "bold",
82 +   },
83 +   listItem: {
84 +     alignItems: "center",
85 +     backgroundColor: "white",
86 +     justifyContent: "center",
87 +     width: width / 1.2,
88 +   },
89 +   body: {
90 +     margin: 20,
91 +     alignItems: "center",
92 +     flexDirection: "row",
93 +   },
94 + ]);
95 +
96 + const mapDispatchToProps = (dispatch) => {
97 +   return { clearCart: () => dispatch(actions.clearCart()) };
98 + };
99 +
100 + export default connect(null, mapDispatchToProps)(Confirm);

```

```

100 // @-4,0 -1,333 @0
1 + import React, { useState } from "react";
2 + import { View, Button } from "react-native";
3 + import {
4 +   Container,
5 +   Header,
6 +   CardView,
7 +   ListItem,
8 +   Text,
9 +   Radio,
10 +   Right,
11 +   Left,
12 +   Picker,
13 +   Icon,
14 +   Body,
15 +   Title,
16 + } from "react-native";
17 +
18 + const methods = [
19 +   { name: "Cash on delivery", value: 1 },
20 +   { name: "Bank Transfer", value: 2 },
21 +   { name: "Card payment", value: 3 },
22 + ];
23 +
24 + const paymentCards = [
25 +   {
26 +     name: "Wallet",
27 +     value: 1,
28 +   },
29 +   {
30 +     name: "Visa",
31 +     value: 2,
32 +   },
33 +   {
34 +     name: "MasterCard",
35 +     value: 3,
36 +   },
37 +   {
38 +     name: "Other",
39 +     value: 4,
40 +   },
41 + ];
42 +
43 + export default function Payment(props) {
44 +   const order = props.route.params;
45 +
46 +   const [selected, setSelected] = useState();
47 +   const [card, setCard] = useState();
48 +
49 +   return (
50 +     <Container>
51 +       <Header>
52 +         <Body>
53 +           <Title>Choose your payment method</Title>
54 +         </Body>
55 +       </Header>
56 +       <Content>
57 +         {methods.map((item, index) => {
58 +           return (
59 +             <ListItem key={item.name} onPress={() => setSelected(item.value)}>
60 +               <Left>
61 +                 <Text>{item.name}</Text>
62 +               </Left>
63 +               <Right>
64 +                 <Radio selected={selected === item.value}></Radio>
65 +               </Right>
66 +             </ListItem>
67 +           );
68 +         })}
69 +         {selected == 3 ? (
70 +           <Picker>
71 +             <mode>"dropdown"
72 +             <onIconPress>({iconName: "arrow-down"})=></onIconPress>
73 +             <headerStyle>{{ backgroundColor: "orange" }}
74 +             <headerTintColor>{{ color: "white" }}
75 +             <headerTitleStyle>{{ color: "white" }}
76 +             <selectedValue>{card}
77 +             <onValueChange>{(x) => setCard(x)}
78 +             <style>{{ height: 50 }}
79 +           </Picker>
80 +           {paymentCards.map((c, index) => {
81 +             <Picker.Item>
82 +               <key>{c.name}
83 +               <label>{c.name}
84 +               <value>{c.name}
85 +             </Picker.Item>
86 +           )}
87 +         </Picker>
88 +         < : null)
89 +       </View style={{ marginTop: 60, alignSelf: "center" }}>
90 +       <Button>
91 +         <title>{"Confirm"}
92 +         <onPress>() => {
93 +           props.navigation.navigate("Confirm", { order });
94 +         }
95 +       </Button>
96 +     </View>
97 +   </Content>
98 + </Container>
99 + );
100 + }

```

```
Screen/Products/ProductContainer.js
1 import React from "react";
2 import { SearchedProduct } from "../SearchedProduct";
3 import { ScrollView } from "react-native";
4 export default function ProductContainer(props) {
5   return (
6     <ScrollView>
7       <View style={{ padding: 10, backgroundColor: "#e0e0e0" }}>
8         <Text>Product Container</Text>
9       </View>
10    </ScrollView>
11  );
12}

Shared/Form/FormContainer.js
1 import React from "react";
2 import { ScrollView, Dimensions, StyleSheet, Text } from "react-native";
3 import { SearchedProduct } from "../SearchedProduct";
4 var { width } = Dimensions.get("window");
5 export default function FormContainer(props) {
6   return (
7     <ScrollView>
8       <View style={styles.container}>
9         <Text style={styles.title}>{props.title}</Text>
10        {props.children}
11      </ScrollView>
12    );
13}
14
15 const styles = StyleSheet.create({
16   container: {
17     margin: 10,
18     marginBottom: 40,
19     width: width,
20     justifyContent: "center",
21     alignItems: "center",
22   },
23   title: {
24     fontSize: 20,
25   },
26 });

Shared/Form/Input.js
1 import React from "react";
2 import { TextInput, StyleSheet } from "react-native";
3 export default function Input(props) {
4   return (
5     <TextInput
6       style={styles.input}
7       placeholder={props.placeholder}
8       name={props.name}
9       value={props.value}
10      autoCorrect={props.autoCorrect}
11      onChangeText={props.onChangeText}
12      onFocus={props.onFocus}
13      secureTextEntry={props.secureTextEntry}
14      keyboardType={props.keyboardType}
15    >
16    {props.children}
17  </TextInput>
18  );
19}
20
21 const styles = StyleSheet.create({
22   input: {
23     width: "80%",
24     height: 30,
25     backgroundColor: "white",
26     margin: 10,
27     borderRadius: 10,
28     padding: 10,
29     borderWidth: 2,
30     borderColor: "orange",
31   },
32 });
```

3.2.11. spajanje frontenda sa backendom - connected frontend with backend

U ovom koraku se dodaje biblioteka „axios“ koja se koristi za sve API call-ove. Mijenja se ProductContainer i daju mu se podaci za produkte i kategorije sa backenda.

```
1 import React, { useState, useEffect } from "react";
2 import React, { useState, useCallback } from "react";
3 import { useFocusEffect } from "react-navigation/native";
4 import {
5   View,
6   StyleSheet,
7 } from "react-native";
8 import ProductList from "../ProductList";
9 import SearchedProduct from "../SearchedProducts";
10 import Banner from "../Shared/Banner";
11 import CategoryFilter from "../CategoryFilter";
12 import baseUrl from "../../assets/common/baseUrl";
13 import axios from "axios";
14
15 var { height } = Dimensions.get("window");
16 const data = require("../assets/products.json");
17 const categoriesData = require("../assets/categories.json");
18
19 export default function ProductContainer(props) {
20   const [products, setProducts] = useState([]);
21   const [productActg, setProductActg] = useState(0);
22   const [active, setActive] = useState(0);
23   const [initialState, setInitialState] = useState(0);
24   const [loading, setLoading] = useState(true);
25
26   useEffect(() => {
27     - setProducts(data);
28     - setProductsFiltered(data);
29     - setFocus(false);
30     - setCategories(categoriesData);
31     - setProductCtg(data);
32     - setActive(1);
33     - setInitialState(data);
34     - return () => {
35       - setProducts([]);
36       - setProductsFiltered([]);
37       - setFocus();
38       - setCategories([]);
39       - setActive();
40       - setInitialState();
41     };
42   }, []);
43
44   useFocusEffect(
45     useCallback(() => {
46       setFocus(false);
47       setActive(-1);
48
49       axios
50         .get(`${baseUrl}/products`)
51         .then(res => {
52           setProducts(res.data);
53           setProductsFiltered(res.data);
54           setProductCtg(res.data);
55           setInitialState(res.data);
56           setLoading(false);
57         })
58         .catch(err => console.log(err));
59
60       axios
61         .get(`${baseUrl}/categories`)
62         .then(res => {
63           setCategories(res.data);
64         })
65         .catch(err => console.log(err));
66       return () => {
67         setProducts([]);
68         setProductsFiltered([]);
69         setFocus();
70         setCategories([]);
71         setActive();
72         setInitialState();
73       };
74     }, []);
75   );
76
77   const searchProduct = (text) => {
78     setProductsFiltered(

```



```
assets/common/baseUrl.js
1 + import { Platform } from "react-native";
2 +
3 + let baseUrl = "";
4 +
5 + {
6 +   Platform.OS == "android"
7 +   ? (baseUrl = "http://10.0.2.2:5000/api")
8 +   : (baseUrl = "http://localhost:5000/api");
9 + }
10 +
11 + export default baseUrl;
```

3.2.12. dodavanje logina i registracije za korisnike i dodavanje toasta za produkte i login - added user login, register and added toast for adding products and login

U ovom koraku se dodaje par komponenti, a to su Login, Register i UserProfile. Sve ove komponente se smještaju u UserNavigator. Pored ovog se dodaje i toast message preko paketa „react-native-toast-message“^[15] koji se prikazuje nakon dodavanja produkta u korpu i nakon logina.

[15] <https://www.npmjs.com/package/react-native-toast-message>


```
App.js
1  import { NavigationContainer } from "react-navigation/native";
2  import Main from "../Navigators/Main";
3  import { Provider } from "react-redux";
4  import store from "../Redux/Store";
5  import Toast from "react-native-toast-message";
6
7  export default function App() {
8    return (
9      <Provider store={store}>
10       <NavigationContainer>
11         <Header/>
12         <Main/>
13         <Toast ref={ref} => Toast.setRef(ref)/>
14       </NavigationContainer>
15     </Provider>
16   );
17 }

Navigators/Main.js
1  import { View } from "react-native";
2  import Icon from "react-native-vector-icons/FontAwesome";
3  import HomeNavigator from "../HomeNavigator";
4  import CartNavigator from "../CartNavigator";
5  import UserNavigator from "../UserNavigator";
6  import CartIcon from "../Shared/CartIcon";
7
8  const Tab = createBottomTabNavigator();
9
10 export default function Main() {
11   <Tab.Screen>
12     <Tab.Screen>
13       name="User"
14     </Tab.Screen>
15     <component={HomeNavigator}>
16     <component={UserNavigator}>
17     options={[
18       tabBarIcon: ({ color }) => (
19         <Icon name="user" color={color} size={30}/>
20       )
21     ]}
22   </Tab.Screen>
23 }

Navigators/UserNavigator.js
1  import React from "react";
2  import { createStackNavigator } from "react-navigation/stack";
3  import Login from "../Screens/User/Login";
4  import Register from "../Screens/User/Register";
5  import UserProfile from "../Screens/User/UserProfile";
6
7  const Stack = createStackNavigator();
8
9  function MyStack() {
10   return (
11     <Stack.Navigator>
12       <Stack.Screen>
13         name="Login"
14         component={Login}
15         options={{ headerShown: false }}
16       </Stack.Screen>
17       <Stack.Screen>
18         name="Register"
19         component={Register}
20         options={{ headerShown: false }}
21       </Stack.Screen>
22       <Stack.Screen>
23         name="UserProfile"
24         component={UserProfile}
25         options={{ headerShown: false }}
26       </Stack.Screen>
27     </Stack.Navigator>
28   );
29 }
30
31 export default function UserNavigator() {
32   return <MyStack/>
33 }
```

```
11 Screens/Products/ProductCard.js
import {
  } from "react-native";
import { connect } from "react-redux";
import * as actions from "../../Redux/Actions/cartActions";
import Toast from "react-native-toast-message";

var { width } = Dimensions.get("window");

function ProductCard(props) {
  <Button
    title="Add"
    color="green"
    onPress={() => props.addItemToCart(props)}
    onPress={() => {
      props.addItemToCart(props),
      Toast.show({
        topOffset: 60,
        type: "success",
        text1: `${name} added to Cart`,
        text2: "Go to your cart to complete order",
      });
    }}
  ></Button>
</View>
} : {

11 Screens/Products/SingleProduct.js
import { Left, Right, Container, H1 } from "native-base";
import { connect } from "react-redux";
import * as actions from "../../Redux/Actions/cartActions";
import Toast from "react-native-toast-message";

function SingleProduct(props) {
  const [item, setItem] = useState(props.route.params.item);

  function SingleProduct(props) {
    <Right>
      <Button
        title="Add"
        onPress={() => props.addItemToCart(item)}
        onPress={() => {
          props.addItemToCart(item);
          Toast.show({
            topOffset: 60,
            type: "success",
            text1: `${item.name} added to Cart`,
            text2: "Go to your cart to complete order",
          });
        }}
      ></Button>
    </Right>
  </View>
}
```

```
48 Screens/User/Login.js
... .. 00 -0,0 +1,00 00
1 + import React, { useState, useEffect } from "react";
2 + import { View, Text, StyleSheet, Button } from "react-native";
3 + import FormContainer from "../../Shared/Form/FormContainer";
4 + import Input from "../../Shared/Form/Input";
5 + import Error from "../../Shared/Error";
6 +
7 + export default function login(props) {
8 +   const [email, setEmail] = useState("");
9 +   const [password, setPassword] = useState("");
10 +   const [error, setError] = useState("");
11 +
12 +   const handleSubmit = () => {
13 +     const user = {
14 +       email,
15 +       password,
16 +     };
17 +
18 +     if (!email || !password) {
19 +       setError("Please fill in your credentials");
20 +     } else {
21 +       setError("");
22 +     }
23 +   };
24 +
25 +   return (
26 +     <FormContainer title="Login">
27 +       <Input
28 +         placeholder="Enter Email"
29 +         name="email"
30 +         id="email"
31 +         value={email}
32 +         onChangeText={(text) => setEmail(text.toLowerCase())}
33 +       ></Input>
34 +       <Input
35 +         placeholder="Enter Password"
36 +         name="password"
37 +         id="password"
38 +         value={password}
39 +         secureTextEntry={true}
40 +         onChangeText={(text) => setPassword(text)}
41 +       ></Input>
42 +       <View style={styles.buttonGroup}>
43 +         {error ? <Error message={error}></Error> : null}
44 +         <Button title="Login" onPress={handleSubmit}></Button>
45 +       </View>
46 +       <View style={{ marginTop: 40 }, styles.buttonGroup}>
47 +         <Text style={styles.middleText}>Don't have an account yet?</Text>
48 +         <Button
49 +           title="Register"
50 +           onPress={() => {
51 +             props.navigation.navigate("Register");
52 +           }}
53 +         ></Button>
54 +       </View>
55 +     </FormContainer>
56 +   );
57 + }
58 +
59 + const styles = StyleSheet.create({
60 +   buttonGroup: {
61 +     width: "80%",
62 +     alignItems: "center",
63 +   },
64 +   middleText: {
65 +     marginBottom: 20,
66 +     alignSelf: "center",
67 +   },
68 + });
```

```

10 // @flow strict
11
12 + import type { message } from 'react';
13 + import { class, view, stylesheets, mixins } from 'react-mixins';
14 + import AuthenticationService from './../AuthService/AuthService';
15 + import User from './../AuthService/User';
16 + import User from './../AuthService';
17 + import { registerWithEmailAndPassword } from 'react-native-firebase';
18 + import User from 'react';
19 + import User from './../AuthService/AuthService';
20 + import User from 'react-native-messaging';
21
22 +
23 +
24 +
25 +
26 +
27 +
28 +
29 +
30 +
31 +
32 +
33 +
34 +
35 +
36 +
37 +
38 +
39 +
40 +
41 +
42 +
43 +
44 +
45 +
46 +
47 +
48 +
49 +
50 +
51 +
52 +
53 +
54 +
55 +
56 +
57 +
58 +
59 +
60 +
61 +
62 +
63 +
64 +
65 +
66 +
67 +
68 +
69 +
70 +
71 +
72 +
73 +
74 +
75 +
76 +
77 +
78 +
79 +
80 +
81 +
82 +
83 +
84 +
85 +
86 +
87 +
88 +
89 +
90 +
91 +
92 +
93 +
94 +
95 +
96 +
97 +
98 +
99 +
100 +
101 +
102 +
103 +
104 +
105 +
106 +
107 +
108 +
109 +
110 +
111 +
112 +
113 +
114 +
115 +
116 +
117 +
118 +
119 +
120 +
121 +
122 +
123 +
124 +
125 +
126 +
127 +
128 +
129 +
130 +
131 +
132 +
133 +
134 +
135 +
136 +
137 +
138 +
139 +
140 +
141 +
142 +
143 +
144 +
145 +
146 +
147 +
148 +
149 +
150 +
151 +
152 +
153 +
154 +
155 +
156 +
157 +
158 +
159 +
160 +
161 +
162 +
163 +
164 +
165 +
166 +
167 +
168 +
169 +
170 +
171 +
172 +
173 +
174 +
175 +
176 +
177 +
178 +
179 +
180 +
181 +
182 +
183 +
184 +
185 +
186 +
187 +
188 +
189 +
190 +
191 +
192 +
193 +
194 +
195 +
196 +
197 +
198 +
199 +
200 +
201 +
202 +
203 +
204 +
205 +
206 +
207 +
208 +
209 +
210 +
211 +
212 +
213 +
214 +
215 +
216 +
217 +
218 +
219 +
220 +
221 +
222 +
223 +
224 +
225 +
226 +
227 +
228 +
229 +
230 +
231 +
232 +
233 +
234 +
235 +
236 +
237 +
238 +
239 +
240 +
241 +
242 +
243 +
244 +
245 +
246 +
247 +
248 +
249 +
250 +
251 +
252 +
253 +
254 +
255 +
256 +
257 +
258 +
259 +
260 +
261 +
262 +
263 +
264 +
265 +
266 +
267 +
268 +
269 +
270 +
271 +
272 +
273 +
274 +
275 +
276 +
277 +
278 +
279 +
280 +
281 +
282 +
283 +
284 +
285 +
286 +
287 +
288 +
289 +
290 +
291 +
292 +
293 +
294 +
295 +
296 +
297 +
298 +
299 +
300 +
301 +
302 +
303 +
304 +
305 +
306 +
307 +
308 +
309 +
310 +
311 +
312 +
313 +
314 +
315 +
316 +
317 +
318 +
319 +
320 +
321 +
322 +
323 +
324 +
325 +
326 +
327 +
328 +
329 +
330 +
331 +
332 +
333 +
334 +
335 +
336 +
337 +
338 +
339 +
340 +
341 +
342 +
343 +
344 +
345 +
346 +
347 +
348 +
349 +
350 +
351 +
352 +
353 +
354 +
355 +
356 +
357 +
358 +
359 +
360 +
361 +
362 +
363 +
364 +
365 +
366 +
367 +
368 +
369 +
370 +
371 +
372 +
373 +
374 +
375 +
376 +
377 +
378 +
379 +
380 +
381 +
382 +
383 +
384 +
385 +
386 +
387 +
388 +
389 +
390 +
391 +
392 +
393 +
394 +
395 +
396 +
397 +
398 +
399 +
400 +
401 +
402 +
403 +
404 +
405 +
406 +
407 +
408 +
409 +
410 +
411 +
412 +
413 +
414 +
415 +
416 +
417 +
418 +
419 +
420 +
421 +
422 +
423 +
424 +
425 +
426 +
427 +
428 +
429 +
430 +
431 +
432 +
433 +
434 +
435 +
436 +
437 +
438 +
439 +
440 +
441 +
442 +
443 +
444 +
445 +
446 +
447 +
448 +
449 +
450 +
451 +
452 +
453 +
454 +
455 +
456 +
457 +
458 +
459 +
460 +
461 +
462 +
463 +
464 +
465 +
466 +
467 +
468 +
469 +
470 +
471 +
472 +
473 +
474 +
475 +
476 +
477 +
478 +
479 +
480 +
481 +
482 +
483 +
484 +
485 +
486 +
487 +
488 +
489 +
490 +
491 +
492 +
493 +
494 +
495 +
496 +
497 +
498 +
499 +
500 +
501 +
502 +
503 +
504 +
505 +
506 +
507 +
508 +
509 +
510 +
511 +
512 +
513 +
514 +
515 +
516 +
517 +
518 +
519 +
520 +
521 +
522 +
523 +
524 +
525 +
526 +
527 +
528 +
529 +
530 +
531 +
532 +
533 +
534 +
535 +
536 +
537 +
538 +
539 +
540 +
541 +
542 +
543 +
544 +
545 +
546 +
547 +
548 +
549 +
550 +
551 +
552 +
553 +
554 +
555 +
556 +
557 +
558 +
559 +
560 +
561 +
562 +
563 +
564 +
565 +
566 +
567 +
568 +
569 +
570 +
571 +
572 +
573 +
574 +
575 +
576 +
577 +
578 +
579 +
580 +
581 +
582 +
583 +
584 +
585 +
586 +
587 +
588 +
589 +
590 +
591 +
592 +
593 +
594 +
595 +
596 +
597 +
598 +
599 +
600 +
601 +
602 +
603 +
604 +
605 +
606 +
607 +
608 +
609 +
610 +
611 +
612 +
613 +
614 +
615 +
616 +
617 +
618 +
619 +
620 +
621 +
622 +
623 +
624 +
625 +
626 +
627 +
628 +
629 +
630 +
631 +
632 +
633 +
634 +
635 +
636 +
637 +
638 +
639 +
640 +
641 +
642 +
643 +
644 +
645 +
646 +
647 +
648 +
649 +
650 +
651 +
652 +
653 +
654 +
655 +
656 +
657 +
658 +
659 +
660 +
661 +
662 +
663 +
664 +
665 +
666 +
667 +
668 +
669 +
670 +
671 +
672 +
673 +
674 +
675 +
676 +
677 +
678 +
679 +
680 +
681 +
682 +
683 +
684 +
685 +
686 +
687 +
688 +
689 +
690 +
691 +
692 +
693 +
694 +
695 +
696 +
697 +
698 +
699 +
700 +
701 +
702 +
703 +
704 +
705 +
706 +
707 +
708 +
709 +
710 +
711 +
712 +
713 +
714 +
715 +
716 +
717 +
718 +
719 +
720 +
721 +
722 +
723 +
724 +
725 +
726 +
727 +
728 +
729 +
730 +
731 +
732 +
733 +
734 +
735 +
736 +
737 +
738 +
739 +
740 +
741 +
742 +
743 +
744 +
745 +
746 +
747 +
748 +
749 +
750 +
751 +
752 +
753 +
754 +
755 +
756 +
757 +
758 +
759 +
760 +
761 +
762 +
763 +
764 +
765 +
766 +
767 +
768 +
769 +
770 +
771 +
772 +
773 +
774 +
775 +
776 +
777 +
778 +
779 +
780 +
781 +
782 +
783 +
784 +
785 +
786 +
787 +
788 +
789 +
790 +
791 +
792 +
793 +
794 +
795 +
796 +
797 +
798 +
799 +
800 +
801 +
802 +
803 +
804 +
805 +
806 +
807 +
808 +
809 +
810 +
811 +
812 +
813 +
814 +
815 +
816 +
817 +
818 +
819 +
820 +
821 +
822 +
823 +
824 +
825 +
826 +
827 +
828 +
829 +
830 +
831 +
832 +
833 +
834 +
835 +
836 +
837 +
838 +
839 +
840 +
841 +
842 +
843 +
844 +
845 +
846 +
847 +
848 +
849 +
850 +
851 +
852 +
853 +
854 +
855 +
856 +
857 +
858 +
859 +
860 +
861 +
862 +
863 +
864 +
865 +
866 +
867 +
868 +
869 +
870 +
871 +
872 +
873 +
874 +
875 +
876 +
877 +
878 +
879 +
880 +
881 +
882 +
883 +
884 +
885 +
886 +
887 +
888 +
889 +
890 +
891 +
892 +
893 +
894 +
895 +
896 +
897 +
898 +
899 +
900 +
901 +
902 +
903 +
904 +
905 +
906 +
907 +
908 +
909 +
910 +
911 +
912 +
913 +
914 +
915 +
916 +
917 +
918 +
919 +
920 +
921 +
922 +
923 +
924 +
925 +
926 +
927 +
928 +
929 +
930 +
931 +
932 +
933 +
934 +
935 +
936 +
937 +
938 +
939 +
940 +
941 +
942 +
943 +
944 +
945 +
946 +
947 +
948 +
949 +
950 +
951 +
952 +
953 +
954 +
955 +
956 +
957 +
958 +
959 +
960 +
961 +
962 +
963 +
964 +
965 +
966 +
967 +
968 +
969 +
970 +
971 +
972 +
973 +
974 +
975 +
976 +
977 +
978 +
979 +
980 +
981 +
982 +
983 +
984 +
985 +
986 +
987 +
988 +
989 +
990 +
991 +
992 +
993 +
994 +
995 +
996 +
997 +
998 +
999 +
1000 +

```

18 Screens/User/UserProfile.js

```
1 + import React from "react";
2 + import { View, Text } from "react-native";
3 +
4 + export default function UserProfile(props) {
5 +   return (
6 +     <View>
7 +       <Text>User profile page</Text>
8 +     </View>
9 +   );
10 + }
```

21 Shared/Error.js

```
1 + import React from "react";
2 + import { StyleSheet, View, Text } from "react-native";
3 +
4 + export default function Error(props) {
5 +   return (
6 +     <View style={styles.container}>
7 +       <Text style={styles.text}>{props.message}</Text>
8 +     </View>
9 +   );
10 + }
11 +
12 + const styles = StyleSheet.create({
13 +   container: {
14 +     width: "100%",
15 +     alignItems: "center",
16 +     margin: 10,
17 +   },
18 +   text: {
19 +     color: "red",
20 +   },
21 + });
```

3.2.13. dodavanje funkcionalnosti za registrovanje i login, dodavanje profila korisnika - added functionality to register and login, added user profile page

Nakon kreiranja osnove za korisnički sistem, potrebno je čuvati podatke o korisniku u state-u. Ponovo se javlja problem od ranije koji je riješen reduxom, ali u ovom slučaju će se riješiti na jedan sličan način, a to je Context API. Princip je veoma sličan reduxu, a razlika je u tome što je Context dio react biblioteke. Takođe se koristi AsyncStorage za čuvanje JWT-a koji se dobije sa backenda.

```
App.js
1  import { StatusBar } from "expo-status-bar";
2  import React from "react";
3  - import { StyleSheet, Text, View } from "react-native";
4  - import ProductContainer from "../Screens/Products/ProductContainer";
5  + import { StyleSheet } from "react-native";
6  import Header from "../Shared/Header";
7  import { NavigationContainer } from "@react-navigation/native";
8  import Main from "../Navigators/Main";
9  import { Provider } from "react-redux";
10 import store from "../Redux/Store";
11 import Toast from "react-native-toast-message";
12 + import Auth from "../Context/Store/Auth";
13
14 export default function App() {
15   return (
16     - <Provider store={store}>
17     - <NavigationContainer>
18     -   <Header></Header>
19     -   <Main></Main>
20     -   <Toast ref={{ref} => Toast.setRef(ref)}></Toast>
21     - </NavigationContainer>
22     - </Provider>
23
24     + <Auth>
25     + <Provider store={store}>
26     + <NavigationContainer>
27     +   <Header></Header>
28     +   <Main></Main>
29     +   <Toast ref={{ref} => Toast.setRef(ref)}></Toast>
30     + </NavigationContainer>
31     + </Provider>
32     + </Auth>
33   );
34 }
35
```

```
... 100 -0,0 -1,00 00
1 + import jwt_decode from "jwt-decode";
2 + import AsyncStorage from "react-native-async-storage/async-storage";
3 + import Toast from "react-native-toast-message";
4 + import baseUrl from "../../assets/consts/baseUrl";
5 + import axios from "axios";
6 +
7 + export const SET_CURRENT_USER = "SET_CURRENT_USER";
8 +
9 + export const loginUser = (user, dispatch) => {
10 + // Fetch: ${baseUrl}/users/login, {
11 + //   method: "POST",
12 + //   body: JSON.stringify(user),
13 + //   headers: [
14 + //     {Accept: "application/json"},
15 + //     {Content-Type: "application/json"},
16 + //   ],
17 + // })
18 + // .then(res) => {
19 + //   Toast.show({
20 + //     topOffset: 50,
21 + //     type: "error",
22 + //     text1: "Hello",
23 + //     text2: typeof res,
24 + //   });
25 + //   res.json();
26 + // }
27 + // .then(data) => {
28 + //   if (data) {
29 + //     const token = data.token;
30 + //     AsyncStorage.setItem("jwt", token);
31 + //     const decoded = jwt_decode(token);
32 + //     dispatch(setCurrentUser(decoded, user));
33 + //   } else {
34 + //     loginUser(dispatch);
35 + //   }
36 + // }
37 + // .catch(err) => {
38 + //   console.log(err);
39 + //   Toast.show({
40 + //     topOffset: 50,
41 + //     type: "error",
42 + //     text1: "Please provide correct credentials",
43 + //     text2: err.toString(),
44 + //   });
45 + //   loginUser(dispatch);
46 + // }
47 +
48 + axios
49 + .post(`${baseUrl}/users/login`, user)
50 + .then(res) => {
51 +   if (res) {
52 +     const token = res.data;
53 +     AsyncStorage.setItem("jwt", token);
54 +     const decoded = jwt_decode(token);
55 +     dispatch(setCurrentUser(decoded, user));
56 +   } else {
57 +     loginUser(dispatch);
58 +   }
59 + }
60 + .catch(err) => {
61 +   console.log(err);
62 +   Toast.show({
63 +     topOffset: 50,
64 +     type: "error",
65 +     text1: "Please provide correct credentials",
66 +     text2: "",
67 +   });
68 +   loginUser(dispatch);
69 + }
70 + };
71 +
72 + export const getUserProfile = (id) => {
73 +   fetch(`${baseUrl}/users/${id}`, {
74 +     method: "GET",
75 +     body: JSON.stringify(user),
76 +     headers: [
77 +       {Accept: "application/json"},
78 +       {Content-Type: "application/json"},
79 +     ],
80 +   })
81 +   .then(res) => res.json()
82 +   .then(data) => console.log(data);
83 + };
84 +
85 + export const logoutUser = (dispatch) => {
86 +   AsyncStorage.removeItem("jwt");
87 +   dispatch(setCurrentUser({}));
88 + };
89 +
90 + export const setCurrentUser = (decoded, user) => {
91 +   return {
92 +     type: SET_CURRENT_USER,
93 +     payload: decoded,
94 +     userProfile: user,
95 +   };
96 + };

```

```
Context/Reducers/Auth_reducer.js
...
1 + import { SET_CURRENT_USER } from "../Actions/Auth.actions";
2 + import isEmpty from "../../assets/common/isEmpty";
3 +
4 + export default function (state, action) {
5 +   switch (action.type) {
6 +     case SET_CURRENT_USER:
7 +       return {
8 +         ...state,
9 +         isAuthenticated: !isEmpty(action.payload),
10 +         user: action.payload,
11 +         userProfile: action.userProfile,
12 +       };
13 +     default:
14 +       return state;
15 +   }
16 + }
```

```
Context/Store/Auth.js
...
1 + import React, { useEffect, useReducer, useState } from "react";
2 + import jwt_decode from "jwt-decode";
3 + import AsyncStorage from "@react-native-async-storage/async-storage";
4 +
5 + import authReducer from "../Reducers/Auth_reducer";
6 + import { setCurrentUser } from "../Actions/Auth.actions";
7 + import AuthGlobal from "../AuthGlobal";
8 +
9 + const Auth = (props) => {
10 +   const [state, dispatch] = useReducer(authReducer, {
11 +     isAuthenticated: null,
12 +     user: {},
13 +   });
14 +   const [showChild, setShowChild] = useState(false);
15 +
16 +   useEffect(() => {
17 +     setShowChild(true);
18 +     if (AsyncStorage.getItem()) {
19 +       const decoded = AsyncStorage.getItem ? AsyncStorage.getItem : "";
20 +       if (setShowChild) {
21 +         dispatch(setCurrentUser(setCurrentUser(jwt_decode(decoded))));
22 +       }
23 +     }
24 +     return () => setShowChild(false);
25 +   }, []);
26 +
27 +   if (!showChild) {
28 +     return null;
29 +   } else {
30 +     return (
31 +       <AuthGlobal.Provider
32 +         value={{
33 +           state,
34 +           dispatch,
35 +         }}
36 +     >
37 +       {props.children}
38 +     </AuthGlobal.Provider>
39 +   );
40 + }
41 + };
42 +
43 + export default Auth;
```

```
Context/Store/AuthGlobal.js
...
1 + import React from "react";
2 +
3 + export default React.createContext({});
```



```
Context/Store/AuthGlobal.js
...
1 + import React from "react";
2 +
3 + export default React.createContext();

Screens/User/Login.js
...
1 - import React, { useState, useEffect } from "react";
1 + import React, { useState, useEffect, useContext } from "react";
2 import { View, Text, StyleSheet, Button } from "react-native";
3 + import { useFocusEffect } from "@react-navigation/native";
3 import FormContainer from "../../Shared/Form/FormContainer";
4 import Input from "../../Shared/Form/Input";
5 import Error from "../../Shared/Error";
6 -
7 + import AuthGlobal from "../../Context/Store/AuthGlobal";
8 + import { loginUser } from "../../Context/Actions/Auth.actions";
7 export default function Login(props) {
9
10 + const context = useContext(AuthGlobal);
8 const [email, setEmail] = useState("");
9 const [password, setPassword] = useState("");
10 const [error, setError] = useState("");
11
15 + useEffect(() => {
16 +   if (context.stateUser.isAuthenticated) {
17 +     props.navigation.navigate("User Profile");
18 +   }
19 +   return () => {};
20 + }, [context.stateUser.isAuthenticated]);
21 +
22 + useFocusEffect(() => {
23 +   if (context.stateUser.isAuthenticated) {
24 +     props.navigation.navigate("User Profile");
25 +   }
26 +   return () => {};
27 + });
28 +
12 const handleSubmit = () => {
13   const user = {
14     email,
18   if (!email || !password) {
19     setError("Please fill in your credentials");
20   } else {
21     setError("");
22     loginUser(user, context.dispatch);
23   }
24 };

```

```
72 Screens/User/UserProfile.js
...
1  - import React from "react";
2  - import { View, Text } from "react-native";
3  + import React, { useContext, useState, useCallback, useEffect } from "react";
4  + import { View, Text, ScrollView, Dimensions, StyleSheet } from "react-native";
5  + import { Container, Content } from "native-base";
6  + import { useFocusEffect } from "react-navigation/native";
7  + import AsyncStorage from "react-native-async-storage/async-storage";
8  + import axios from "axios";
9  + import baseUrl from "../../assets/remote/baseUrl";
10 + import AuthGlobal from "../../Context/Store/AuthGlobal";
11 + import { loginUser } from "../../Context/Actions/Auth.actions";
12
13 export default function UserProfile(props) {
14   + const context = useContext(AuthGlobal);
15   + const [userProfile, setUserProfile] = useState({});
16   + const [jet, setJet] = useState("sa");
17   +
18   + useEffect(() => {
19     + if (!context.stateUser.isAuthenticated) {
20       + props.navigation.navigate("Login");
21     }
22   +
23   + AsyncStorage.getItem("jet")
24     + .then((res) => {
25       + setJet(res);
26     +
27     + axios
28       + .get(`${baseUrl}/users/${context.stateUser.user.userId}`, {
29         + headers: { Authorization: `Bearer ${res}` },
30       + })
31       + .then((user) => setUserProfile(user.data));
32     +
33     + .catch((err) => console.log(err));
34     + return () => {
35       + setUserProfile({});
36     + };
37     + }, [context.stateUser.isAuthenticated]);
38   +
39   + return (
40     + <View>
41     +   <Text>User profile page</Text>
42     + </View>
43     + <Container style={styles.container}>
44     +   <ScrollView contentContainerStyle={styles.subContainer}>
45     +     <Text style={{ fontSize: 20 }}>
46     +       {userProfile ? userProfile.name : ""}
47     +     </Text>
48     +     <View style={{ marginTop: 20 }}>
49     +       <Text style={{ margin: 10 }}>
50     +         Email: {userProfile ? userProfile.email : ""}
51     +       </Text>
52     +       <Text style={{ margin: 10 }}>
53     +         Phone: {userProfile ? userProfile.phone : ""}
54     +       </Text>
55     +     </View>
56     +     <View style={{ marginTop: 60 }}>
57     +       <Button
58         +         title="Logout"
59         +         onPress={() => {
60           +           AsyncStorage.removeItem("jet"),
61           +           loginUser(context.dispatch),
62         +         }}
63       +     </Button>
64     +   </View>
65     + </ScrollView>
66     + </Container>
67   );
68 }
69
70 +
71 + const styles = StyleSheet.create([
72 +   container: [
73 +     flex: 1,
74 +     alignItems: "center",
75 +   ],
76 +   subContainer: [
77 +     alignItems: "center",
78 +     marginTop: 20,
79 +   ],
80 + ]);
81
82
83 assets/common/isEmpty.js
...
1  + const isEmpty = (value) => {
2  +   if (
3  +     value === undefined ||
4  +     value === null ||
5  +     (typeof value === "object" && Object.keys(value).length === 0) ||
6  +     (typeof value === "string" && value.trim().length === 0)
7  +   ) {
8  +     return true;
9  +   } else {
10 +     return false;
11 +   }
12 + };
13 + export default isEmpty;
```

3.2.14. dodavanje admin panela i produkt liste u njemu - added admin panel and product list in it

Admin treba da uređuje produkte, dodaje nove produkte i upravlja narudžbama. Za to sve se dodaje potrebna navigacija i potrebne komponente.

```
29 Navigators/AdminNavigator.js
...
1 + import React from "react";
2 + import { createStackNavigator } from "@react-navigation/stack";
3 + import Orders from "../Screens/Admin/Orders";
4 + import Products from "../Screens/Admin/Products";
5 + import ProductForm from "../Screens/Admin/ProductForm";
6 + import Categories from "../Screens/Admin/Categories";
7 +
8 + const Stack = createStackNavigator();
9 +
10 + function MyStack() {
11 +   return (
12 +     <Stack.Navigator>
13 +       <Stack.Screen
14 +         name="Products"
15 +         component={Products}
16 +         options={{
17 +           title: "Products",
18 +         }}
19 +     ></Stack.Screen>
20 +     <Stack.Screen name="Categories" component={Categories}></Stack.Screen>
21 +     <Stack.Screen name="Orders" component={Orders}></Stack.Screen>
22 +     <Stack.Screen name="ProductForm" component={ProductForm}></Stack.Screen>
23 +   </Stack.Navigator>
24 + );
25 + }
26 +
27 + export default function AdminNavigator() {
28 +   return <MyStack></MyStack>;
29 + }
```

```
26 Navigators/Main.js
...
1 - import React from "react";
1 + import React, { useContext } from "react";
2 import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
3 import { View } from "react-native";
4 import Icon from "react-native-vector-icons/FontAwesome";
5 import HomeNavigator from "../HomeNavigator";
6 import CartNavigator from "../CartNavigator";
7 import UserNavigator from "../UserNavigator";
8 + import AdminNavigator from "../AdminNavigator";
8 import CartIcon from "../Shared/CartIcon";
10 + import AuthGlobal from "../Context/Store/AuthGlobal";
9
10 const Tab = createBottomTabNavigator();
11
12 export default function Main() {
15 + const context = useContext(AuthGlobal);
10 +
13 return (
14 <Tab.Navigator
15 initialRouteName="Home"
16 >
17 <Tab.Screen
18 name="Home"
19 component={HomeNavigator}
20 options={{
21 tabBarIcon: ({ color }) => (
22 <Icon name="home" color={color} size={30}></Icon>
23 ),
24 }}
25 ></Tab.Screen>
26 <Tab.Screen
27 name="Admin"
28 component={AdminNavigator}
29 options={{
30 tabBarIcon: ({ color }) => (
31 <Icon name="cog" color={color} size={30}></Icon>
32 ),
33 }}
34 ></Tab.Screen>
35 <Tab.Screen
36 name="User"
37 component={UserNavigator}
38 ></Tab.Screen>
39 </Tab.Navigator>
40 );
41 }
42
43 export default Main;
```

```
10 Screens/Admin/Categories.js
...
1 + import React from "react";
2 + import { View, Text } from "react-native";
3 +
4 + export default function Categories(props) {
5 + return (
6 + <View>
7 + <Text>Categories screen</Text>
8 + </View>
9 + );
10 + }
```

```
...  ...  @@ -0,0 +1,123 @@
1  + import React, { useState } from "react";
2  + import {
3  +   View,
4  +   StyleSheet,
5  +   Text,
6  +   Image,
7  +   TouchableHighlight,
8  +   TouchableOpacity,
9  +   Dimensions,
10 +   Button,
11 +   Modal,
12 + } from "react-native";
13 + import Icon from "react-native-vector-icons/FontAwesome";
14 +
15 + var { width } = Dimensions.get("window");
16 +
17 + export default function ListItems(props) {
18 +   const [modalVisible, setModalVisible] = useState(false);
19 +
20 +   return (
21 +     <View>
22 +       <Modal
23 +         animationType="fade"
24 +         transparent={true}
25 +         visible={modalVisible}
26 +         onRequestClose={() => setModalVisible(false)}
27 +       >
28 +         <View style={styles.centerView}>
29 +           <View style={styles.modalView}>
30 +             <TouchableOpacity
31 +               underlayColor="#E8E8E8"
32 +               onPress={() => setModalVisible(false)}
33 +               style={{
34 +                 alignSelf: "flex-end",
35 +                 position: "absolute",
36 +                 top: 5,
37 +                 right: 10,
38 +               }}
39 +             >
40 +               <Icon name="close" size={20}></Icon>
41 +             </TouchableOpacity>
42 +             <Button
43 +               title="Edit"
44 +               onPress={() => [
45 +                 props.navigation.navigate("ProductForm"),
46 +                 setModalVisible(false),
47 +               ]}
48 +             ></Button>
49 +             <Button title="Delete"></Button>
50 +           </View>
51 +         </View>
52 +       </Modal>
53 +       <TouchableOpacity
54 +         onPress={() => {
55 +           props.navigation.navigate("Product Detail", { item: props });
56 +         }}
57 +         onLongPress={() => setModalVisible(true)}
58 +         style={{
59 +           styles.container,
60 +           [ backgroundColor: props.index % 2 === 0 ? "white" : "gainsboro" ],
61 +         }}
62 +       >
```

```

63 +     <Image
64 +       source={{
65 +         uri: props.image
66 +         ? props.image
67 +         : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23640_960_720.png",
68 +       }}
69 +       style={styles.image}
70 +       resizeMode="contain"
71 +     ></Image>
72 +     <Text style={styles.item}>{props.brand}</Text>
73 +     <Text style={styles.item} numberOfLines={1} ellipsizeMode="tail">
74 +       {props.name}
75 +     </Text>
76 +     <Text style={styles.item} numberOfLines={1} ellipsizeMode="tail">
77 +       {props.category.name}
78 +     </Text>
79 +     <Text style={styles.item}>${ props.price}</Text>
80 +   </TouchableOpacity>
81 + </View>
82 +   });
83 + ]
84 +
85 + const styles = StyleSheet.create({
86 +   container: {
87 +     flexDirection: "row",
88 +     padding: 5,
89 +     width: width,
90 +   },
91 +   image: {
92 +     borderRadius: 50,
93 +     width: width / 6,
94 +     height: 20,
95 +     margin: 2,
96 +   },
97 +   item: {
98 +     flexWrap: "wrap",
99 +     margin: 3,
100 +     width: width / 6,
101 +   },
102 +   containerView: {
103 +     flex: 1,
104 +     justifyContent: "center",
105 +     alignItems: "center",
106 +     marginTop: 22,
107 +   },
108 +   modalView: {
109 +     margin: 20,
110 +     backgroundColor: "white",
111 +     borderRadius: 20,
112 +     padding: 35,
113 +     alignItems: "center",
114 +     shadowColor: "#000",
115 +     shadowOffset: {
116 +       width: 0,
117 +       height: 2,
118 +     },
119 +     shadowOpacity: 0.25,
120 +     shadowRadius: 3.84,
121 +     elevation: 5,
122 +   },
123 + });

```

```
10 Screens/Admin/Orders.js
...
1 + import React from "react";
2 + import { View, Text } from "react-native";
3 +
4 + export default function Orders(props) {
5 +   return (
6 +     <View>
7 +       <Text>Orders screen</Text>
8 +     </View>
9 +   );
10 + }

10 Screens/Admin/ProductForm.js
...
1 + import React from "react";
2 + import { View, Text } from "react-native";
3 +
4 + export default function ProductForm(props) {
5 +   return (
6 +     <View>
7 +       <Text>ProductForm screen</Text>
8 +     </View>
9 +   );
10 + }
```

```
127 Screens/Admin/Products.js
... .. 00 -9,9 +1,127 00
1 + import React, { useState, useCallback } from "react";
2 + import {
3 +   View,
4 +   Text,
5 +   FlatList,
6 +   ActivityIndicator,
7 +   StyleSheet,
8 +   Dimensions,
9 +   Button,
10 + } from "react-native";
11 + import { Header, Item, Input } from "native-base";
12 + import Icon from "react-native-vector-icons/FontAwesome";
13 + import { useFocusEffect } from "@react-navigation/native";
14 + import axios from "axios";
15 + import baseUrl from "../assets/common/baseUrl";
16 + import AsyncStorage from "@react-native-async-storage/async-storage";
17 + import ListItem from "../ListItem";
18 +
19 + var { height, width } = Dimensions.get("window");
20 +
21 + const listHeader = () => {
22 +   return (
23 +     <View elevation={1} style={styles.listHeader}>
24 +       <View style={styles.headerItem}></View>
25 +       <View style={styles.headerItem}>
26 +         <Text style={{ fontWeight: "bold" }}>Brand</Text>
27 +       </View>
28 +       <View style={styles.headerItem}>
29 +         <Text style={{ fontWeight: "bold" }}>Name</Text>
30 +       </View>
31 +       <View style={styles.headerItem}>
32 +         <Text style={{ fontWeight: "bold" }}>Category</Text>
33 +       </View>
34 +       <View style={styles.headerItem}>
35 +         <Text style={{ fontWeight: "bold" }}>Prices</Text>
36 +       </View>
37 +     </View>
38 +   );
39 + };
40 +
41 + export default function Products(props) {
42 +   const [productList, setProductList] = useState();
43 +   const [productFilter, setProductFilter] = useState();
44 +   const [loading, setloading] = useState(true);
45 +   const [token, setToken] = useState();
46 +
47 +   useFocusEffect(
48 +     useCallback(() => {
49 +       AsyncStorage.getItem("jwt")
50 +         .then((res) => setToken(res))
51 +         .catch((err) => console.log(err));
52 +
53 +       axios.get(`${baseUrl}/products`).then((res) => {
54 +         setProductList(res.data);
55 +         setProductFilter(res.data);
56 +         setloading(false);
57 +       });
58 +       return () => {
59 +         setProductList();
60 +         setProductFilter();
61 +         setloading(true);
62 +       };
63 +     }, []),
64 +   );
65 +
66 +   const searchProduct = (text) => {
67 +     if (text === "") {
68 +       setProductFilter(productList);
69 +     }
70 +     setProductFilter(
71 +       productList.filter((i) =>
72 +         i.name.toLowerCase().includes(text.toLowerCase())
73 +       )
74 +     );
75 +   };
76 + }
```



```

77 +   return {
78 +     <View>
79 +       <View>
80 +         <Header searchBar rounded>
81 +           <Item style={{ padding: 5 }}>
82 +             <Icon name="search"></Icon>
83 +             <Input
84 +               placeholder="Search"
85 +               onChangeText={({text} => searchProduct(text))
86 +             ></Input>
87 +           </Item>
88 +         </Header>
89 +       </View>
90 +     {loading ? (
91 +       <View style={styles.spinner}>
92 +         <ActivityIndicator size="large" color="red"></ActivityIndicator>
93 +       </View>
94 +     ) : (
95 +       <FlatList
96 +         ListHeaderComponent={listHeader}
97 +         data={productFilter}
98 +         renderItem={({ item, index }) => (
99 +           <ListItem
100 +             {...item}
101 +             navigation={props.navigation}
102 +             index={index}
103 +           ></ListItem>
104 +         )}
105 +         keyExtractor={({item} => item.id}
106 +       ></FlatList>
107 +     )}
108 +   </View>
109 + };
110 + }
111 +
112 + const styles = StyleSheet.create({
113 +   listHeader: {
114 +     flexDirection: "row",
115 +     padding: 5,
116 +     backgroundColor: "gainsboro",
117 +   },
118 +   headerItem: {
119 +     margin: 3,
120 +     width: width / 6,
121 +   },
122 +   spinner: {
123 +     height: height / 2,
124 +     alignItems: "center",
125 +     justifyContent: "center",
126 +   },
127 + });

```

3.2.15. dodavanje stilizovanih dugmadi i semafora za dostupnost - added styled buttons and traffic light for availability

Da bi aplikacija ljepše izgledala koristi se styled-components^[16] paket za kreiranja posebnih komponenti. U ovom slučaju se kreira button koji kroz propove dobija karakteristike kao što su tip i veličina. Pored buttona kreira se i traffic light koji se koristi za produkte u trgovini, ako je produkt dostupan imat će zeleni traffic light kraj sebe, niska količina na stanju – žuti i ako nije na stanju imat će crveni traffic light. Nakon kreiranja komponenti potrebno je izmijeniti već postojeće buttone u aplikaciji sa ShopButton.js komponentom.

```
43 Shared/StyledComponents/ShopButton.js
...
1 + import styled, { css } from "styled-components";
2 +
3 + const ShopButton = styled.TouchableOpacity`
4 +   flex-direction: row;
5 +   border-radius: 3px;
6 +   padding: 10px;
7 +   margin: 5px;
8 +   justify-content: center;
9 +   background: transparent;
10 +
11 +   ${({props =>
12 +     props.primary &&
13 +     css`
14 +       background: #5cb85c;
15 +     `
16 +   }
17 +   ${({props =>
18 +     props.secondary &&
19 +     css`
20 +       background: #62b1f6;
21 +     `
22 +   }
23 +   ${({props =>
24 +     props.danger &&
25 +     css`
26 +       background: #f40105;
27 +     `
28 +   }
29 +   ${({props =>
30 +     props.large &&
31 +     css`
32 +       width: 135px;
33 +     `
34 +   }
35 +   ${({props =>
36 +     props.medium &&
37 +     css`
38 +       width: 100px;
39 +     `
40 +   }
41 +   ${({props =>
42 +     props.small &&
43 +     css`
44 +       width: 40px;
45 +     `
46 +   }
47 + `;
48 +
49 + export default ShopButton;
```

[16] <https://www.npmjs.com/package/styled-components>

```
1 + import styled, { css } from "styled-components";
2 +
3 + const TrafficLight = styled.View`
4 +   border-radius: 50px;
5 +   width: 10px;
6 +   height: 10px;
7 +   padding: 10px;
8 +
9 +   ${({props}) =>
10 +     props.available &&
11 +     css`
12 +       background: #afec1a;
13 +     `
14 +
15 +   ${({props}) =>
16 +     props.limited &&
17 +     css`
18 +       background: #dde033;
19 +     `
20 +
21 +   ${({props}) =>
22 +     props.unavailable &&
23 +     css`
24 +       background: #ec241a;
25 +     `
26 + `;
27 +
28 + export default TrafficLight;
```

3.2.16. dodavanje mogućnosti za dodavanje, uređivanje i brisanje produkta iz admin panela - added ability to add, edit and delete items from admin panel

Kreiranjem ProductForm.js komponente aplikaciji se dodaje mogućnost za dodavanje i izmjenjivanje postojećih produkta u bazi. Za ovaj korak se instalira paket „expo-image-picker“^[17] kojim je moguće da tražiti perimisije za pristupi kameri i slikama. Asinhrona funkcija launchImageLibraryAsync otvara galeriju te se iz nje selektuje i uploaduje slika. Ostatak komponente prima inpute kao što su naziv, brend, cijena i slično.

```
313 Screens/Admin/ProductForm.js
... -1,10 +1,313 @0
1 - import React from "react";
2 - import { View, Text } from "react-native";
+ import React, { useState, useEffect } from "react";
3 + import {
4 +   View,
5 +   Text,
6 +   Image,
7 +   StyleSheet,
8 +   TouchableOpacity,
9 +   Platform,
10 + } from "react-native";
11 + import { Item, Picker } from "native-base";
12 + import FormContainer from "../../Shared/Form/FormContainer";
13 + import Input from "../../Shared/Form/Input";
14 + import ShopButton from "../../Shared/StyledComponents/ShopButton";
15 + import Error from "../../Shared/Error";
16 + import Icon from "react-native-vector-icons/FontAwesome";
17 + import Toast from "react-native-toast-message";
18 + import AsyncStorage from "react-native-async-storage/async-storage";
19 + import baseUrl from "../../assets/common/baseUrl";
20 + import axios from "axios";
21 + import * as ImagePicker from "expo-image-picker";
22 + import mime from "mime";
23
24 export default function ProductForm(props) {
25 +   const [pickerValue, setPickerValue] = useState();
26 +   const [brand, setBrand] = useState();
27 +   const [name, setName] = useState();
28 +   const [price, setPrice] = useState();
29 +   const [description, setDescription] = useState();
30 +   const [image, setImage] = useState();
31 +   const [mainImage, setMainImage] = useState();
32 +   const [category, setCategory] = useState();
33 +   const [categories, setCategories] = useState([]);
34 +   const [token, setToken] = useState();
35 +   const [error, setError] = useState();
36 +   const [countInStock, setCountInStock] = useState();
37 +   const [rating, setRating] = useState(0);
38 +   const [isFeatured, setIsFeatured] = useState(false);
39 +   const [richDescription, setRichDescription] = useState();
40 +   const [numReviews, setNumReviews] = useState(0);
41 +   const [item, setItem] = useState(null);
42 +   const [updatedImage, setUpdatedImage] = useState(false);
43 +
44 +   useEffect(() => {
45 +     if (!props.route.params) {
46 +       setItem(null);
47 +     } else {
48 +       setItem(props.route.params.item);
49 +       setBrand(props.route.params.item.brand);
50 +       setName(props.route.params.item.name);
51 +       setPrice(props.route.params.item.price.toString());
52 +       setDescription(props.route.params.item.description);
53 +       setMainImage(props.route.params.item.image);
54 +       setCategory(props.route.params.item.category_id);
55 +       setCountInStock(props.route.params.item.countInStock.toString());
56 +     }
57 +   });
58 +
59 +   const handleSubmit = async () => {
60 +     if (!props.route.params) {
61 +       // Dodavanje novog produkta
62 +       const formData = {
63 +         name,
64 +         brand,
65 +         price,
66 +         description,
67 +         image,
68 +         mainImage,
69 +         category,
70 +         countInStock,
71 +         rating,
72 +         isFeatured,
73 +         richDescription,
74 +         numReviews,
75 +       };
76 +       try {
77 +         await axios.post(baseUrl + "/products", formData);
78 +         Toast.show({
79 +           text: "Proizvod je dodan",
80 +           type: "success",
81 +         });
82 +       } catch (error) {
83 +         setError(error);
84 +       }
85 +     } else {
86 +       // Uređivanje postojećeg produkta
87 +       const formData = {
88 +         name,
89 +         brand,
90 +         price,
91 +         description,
92 +         image,
93 +         mainImage,
94 +         category,
95 +         countInStock,
96 +         rating,
97 +         isFeatured,
98 +         richDescription,
99 +         numReviews,
100 +       };
101 +       try {
102 +         await axios.put(baseUrl + "/products/" + item._id, formData);
103 +         Toast.show({
104 +           text: "Proizvod je uređen",
105 +           type: "success",
106 +         });
107 +       } catch (error) {
108 +         setError(error);
109 +       }
110 +     }
111 +   };
112 +
113 +   const handleImageUpload = async () => {
114 +     try {
115 +       const result = await ImagePicker.launchImageLibraryAsync({
116 +         mediaTypes: ImagePicker.MediaTypeOptions.Images,
117 +         allowsEditing: true,
118 +         aspectRatio: "square",
119 +         quality: 1,
120 +       });
121 +       if (!result.canceled) {
122 +         const uri = result.assets[0].uri;
123 +         const mimeType = mime.getType(uri);
124 +         const formData = new FormData();
125 +         formData.append("image", {
126 +           uri,
127 +           type: mimeType,
128 +         });
129 +         await axios.post(baseUrl + "/upload-image", formData);
130 +         Toast.show({
131 +           text: "Slika je uploadovana",
132 +           type: "success",
133 +         });
134 +       }
135 +     } catch (error) {
136 +       setError(error);
137 +     }
138 +   };
139 +
140 +   const handleImageDelete = async () => {
141 +     try {
142 +       await axios.delete(baseUrl + "/upload-image/" + item.image);
143 +       Toast.show({
144 +         text: "Slika je obrisana",
145 +         type: "success",
146 +       });
147 +     } catch (error) {
148 +       setError(error);
149 +     }
150 +   };
151 +
152 +   return (
153 +     <View style={styles.container}>
154 +       <FormContainer>
155 +         <Text>Dodaj novi proizvod</Text>
156 +         <Text>Ime</Text>
157 +         <Text>Cijena</Text>
158 +         <Text>Opis</Text>
159 +         <Text>Glavna slika</Text>
160 +         <Text>Kategorija</Text>
161 +         <Text>Broj na zali</Text>
162 +         <Text>Ocjenjivanje</Text>
163 +         <Text>Izdvajanje</Text>
164 +         <Text>Opis</Text>
165 +         <Text>Broj recenzija</Text>
166 +         <Text>Dodaj proizvod</Text>
167 +         <Text>Uredi proizvod</Text>
168 +         <Text>Obriši proizvod</Text>
169 +         <Text>Dodaj sliku</Text>
170 +         <Text>Obriši sliku</Text>
171 +       </FormContainer>
172 +     </View>
173 +   );
174 + }
175 +
176 + const styles = StyleSheet.create({
177 +   container: {
178 +     padding: 10,
179 +   },
180 + });
```

[17] <https://www.npmjs.com/package/expo-image-picker>

```

57 + AsyncStorage.getItem("jwt")
58 +   .then((res) => setToken(res))
59 +   .catch((err) => console.log(err));
60 +
61 + axios
62 +   .get(`${baseUrl}/categories`)
63 +   .then((res) => setCategories(res.data))
64 +   .catch((err) => alert("Error loading categories"));
65 +
66 + (async () => {
67 +   if (Platform.OS !== "web") {
68 +     const { status } = await ImagePicker.requestCameraPermissionsAsync();
69 +     if (status !== "granted") {
70 +       alert("Sorry, we need camera permissions.");
71 +     }
72 +   }
73 + })();
74 + return () => {
75 +   setCategories([]);
76 + };
77 + }, []);
78 +
79 + const pickImage = async () => {
80 +   let result = await ImagePicker.launchImageLibraryAsync({
81 +     mediaTypes: ImagePicker.MediaTypeOptions.All,
82 +     allowsEditing: true,
83 +     aspect: [4, 3],
84 +     quality: 1,
85 +   });
86 +
87 +   if (!result.cancelled) {
88 +     setMainImage(result.uri);
89 +     setImage(result.uri);
90 +     setUpdatedImage(true);
91 +   }
92 + };
93 +
94 + const addProducts = () => {
95 +   if (
96 +     !name ||
97 +     !brand ||
98 +     !price ||
99 +     !description ||
100 +     !category ||
101 +     !countInStock
102 +   ) {
103 +     setError("Please fill in the form correctly.");
104 +     return;
105 +   }
106 +   let formData = new FormData();
107 +
108 +   if (item !== null && !image) {
109 +     formData.append("image", item.image);
110 +   } else {
111 +     const newImageUri = "file:/// " + image.split("file:").join("");
112 +     formData.append("image", {
113 +       uri: newImageUri,
114 +       type: mime.getType(newImageUri),
115 +       name: newImageUri.split("/").pop(),
116 +     });
117 +   }

```

```

118 +   formData.append("name", name);
119 +   formData.append("brand", brand);
120 +   formData.append("price", price);
121 +   formData.append("description", description);
122 +   formData.append("category", category);
123 +   formData.append("countInStock", countInStock);
124 +   formData.append("richDescription", richDescription);
125 +   formData.append("rating", rating);
126 +   formData.append("numReviews", numReviews);
127 +   formData.append("isFeatured", isFeatured);
128 +   const config = {
129 +     headers: {
130 +       "Content-Type": "multipart/form-data",
131 +       Authorization: "Bearer ${token}",
132 +     },
133 +   };
134 +   if (item !== null) {
135 +     axios
136 +       .put(`${baseUrl}/products/${item._id}`, formData, config)
137 +       .then((res) => {
138 +         if (res.status == 200 || res.status == 2001) {
139 +           Toast.show({
140 +             topOffset: 60,
141 +             type: "success",
142 +             text1: "Product edited",
143 +             text2: "",
144 +           });
145 +           setTimeout(() => {
146 +             props.navigation.navigate("Products");
147 +           }, 500);
148 +         }
149 +       })
150 +       .catch((err) => {
151 +         Toast.show({
152 +           topOffset: 60,
153 +           type: "error",
154 +           text1: "Something went wrong",
155 +           text2: err.toString(),
156 +         });
157 +       });
158 +   } else {
159 +     axios
160 +       .post(`${baseUrl}/products`, formData, config)
161 +       .then((res) => {
162 +         if (res.status == 200 || res.status == 2001) {
163 +           Toast.show({
164 +             topOffset: 60,
165 +             type: "success",
166 +             text1: "New Product added",
167 +             text2: "",
168 +           });
169 +           setTimeout(() => {
170 +             props.navigation.navigate("Products");
171 +           }, 500);
172 +         }
173 +       })
174 +       .catch((err) => {
175 +         Toast.show({
176 +           topOffset: 60,
177 +           type: "error",
178 +           text1: "Something went wrong",
179 +           text2: err.toString(),
180 +         });
181 +       });
182 +   }
183 + });
184 +

```

```

5 185     return {
6         -     <View>
7         -     <Text>ProductForm screen</Text>
8         -     </View>
186 +     <FormContainer title="Add Product">
187 +     <View style={styles.imageContainer}>
188 +     <Image source={{ uri: mainImage }} style={styles.image}></Image>
189 +     <TouchableOpacity onPress={pickImage} style={styles.imagePicker}>
190 +     <Icon style={{ color: "white" }} name="camera"></Icon>
191 +     </TouchableOpacity>
192 +     </View>
193 +     <View style={styles.label}>
194 +     <Text style={{ textDecorationLine: "underline" }}>Brand</Text>
195 +     </View>
196 +     <Input
197 +     placeholder="Brand"
198 +     name="brand"
199 +     id="brand"
200 +     value={brand}
201 +     onChangeText={({text} => setBrand(text))
202 +     >></Input>
203 +     <View style={styles.label}>
204 +     <Text style={{ textDecorationLine: "underline" }}>Name</Text>
205 +     </View>
206 +     <Input
207 +     placeholder="Name"
208 +     name="name"
209 +     id="name"
210 +     value={name}
211 +     onChangeText={({text} => setName(text))
212 +     >></Input>
213 +     <View style={styles.label}>
214 +     <Text style={{ textDecorationLine: "underline" }}>Price</Text>
215 +     </View>
216 +     <Input
217 +     placeholder="Price"
218 +     name="price"
219 +     id="price"
220 +     value={price}
221 +     keyboardType={"numeric"}
222 +     onChangeText={({text} => setPrice(text))
223 +     >></Input>
224 +     <View style={styles.label}>
225 +     <Text style={{ textDecorationLine: "underline" }}>Count in Stock</Text>
226 +     </View>
227 +     <Input
228 +     placeholder="Stock"
229 +     name="stock"
230 +     id="stock"
231 +     value={countInStock}
232 +     keyboardType={"numeric"}
233 +     onChangeText={({text} => setCountInStock(text))
234 +     >></Input>
235 +     <View style={styles.label}>
236 +     <Text style={{ textDecorationLine: "underline" }}>Description</Text>
237 +     </View>
238 +     <Input
239 +     placeholder="Description"
240 +     name="description"
241 +     id="description"
242 +     value={description}
243 +     onChangeText={({text} => setDescription(text))
244 +     >></Input>
245 +     <View style={styles.label}>
246 +     <Text style={{ textDecorationLine: "underline" }}>Category</Text>
247 +     </View>

```



```

248 +     <Item picker>
249 +     <Picker
250 +       mode="dropdown"
251 +       iosIcon={<Icon color={"#887aff"} name="arrow-down"></Icon>}
252 +       style={{ width: undefined, height: 30 }}
253 +       placeholder="Select your Category"
254 +       selectedValue={pickerValue}
255 +       placeholderStyle={{ color: "#887aff" }}
256 +       placeholderIconColor="#887aff"
257 +       onChange={(e) => [setPickerValue(e), setCategory(e)]}
258 +     >
259 +       {categories.map((c) => {
260 +         <Picker.Item key={c._id} label={c.name} value={c._id}></Picker.Item>
261 +       })}
262 +     </Picker>
263 +   </Item>
264 +   {error ? <Error message={error}></Error> : null}
265 +   <View style={styles.buttonContainer}>
266 +     <ShopButton large primary onPress={addProducts}>
267 +       <Text style={styles.buttonText}>Confirm</Text>
268 +     </ShopButton>
269 +   </View>
270 + </FormContainer>
271 +   });
272 + }
273 +
274 + const styles = StyleSheet.create({
275 +   label: {
276 +     width: "88%",
277 +     marginTop: 18,
278 +   },
279 +   buttonContainer: {
280 +     width: "88%",
281 +     marginBottom: 88,
282 +     marginTop: 26,
283 +     alignItems: "center",
284 +   },
285 +   buttonText: {
286 +     color: "white",
287 +   },
288 +   imageContainer: {
289 +     width: 288,
290 +     height: 288,
291 +     borderStyle: "solid",
292 +     borderWidth: 8,
293 +     padding: 8,
294 +     justifyContent: "center",
295 +     borderRadius: 188,
296 +     borderColor: "#E8E8E8",
297 +     elevation: 18,
298 +   },
299 +   image: {
300 +     width: "180%",
301 +     height: "100%",
302 +     borderRadius: 188,
303 +   },
304 +   imagePicker: {
305 +     position: "absolute",
306 +     right: 5,
307 +     bottom: 5,
308 +     backgroundColor: "grey",
309 +     padding: 8,
310 +     borderRadius: 188,
311 +     elevation: 28,
312 +   },
313 + });

```


3.2.17. dodavanje mogućnosti za brisanje i dodavanje kategorija iz admin panela - added ability to add and delete categories in admin panel

Nakon što admin može da doda produkta, takođe treba da ima opciju gdje konfiguriše kategorije. Za to je kreirana komponenta Categories.js koja prikazuje postojeće kategorije, ima opciju da briše izabranu kategoriju i može da doda novu kategoriju u listu kategorija.

```
00 -1,10 +1,147 00
1 - import React from "react";
2 - import { View, Text } from "react-native";
+ import React, { useState, useEffect } from "react";
+ import {
+   View,
+   Text,
+   FlatList,
+   Dimensions,
+   TextInput,
+   StyleSheet,
+ } from "react-native";
10 + import BaseURL from "../../assets/constant/baseURL";
11 + import axios from "axios";
12 + import AsyncStorage from "react-native-async-storage/async-storage";
13 + import ShopButton from "../../Shared/StyledComponents/ShopButton";
14 +
15 + var { width } = Dimensions.get("window");
16 +
17 + const Item = (props) => {
18 +   return (
19 +     <View style={styles.item}>
20 +       <Text>{props.item.name}</Text>
21 +       <ShopButton danger radius onPress={() => props.delete(props.item._id)}>
22 +         <Text style={{ color: "white", fontWeight: "bold" }}>Delete</Text>
23 +     </ShopButton>
24 +   </View>
25 + );
26 + };
27
28 export default function Categories(props) {
29 +   const [categories, setCategories] = useState([]);
30 +   const [categoryName, setCategoryName] = useState("");
31 +   const [token, setToken] = useState("");
32 +
33 +   useEffect(() => {
34 +     AsyncStorage.getItem("jwt")
35 +       .then((res) => setToken(res))
36 +       .catch((err) => console.log(err));
37 +
38 +     axios
39 +       .get(`${BaseURL}/categories`)
40 +       .then((res) => setCategories(res.data))
41 +       .catch((err) => alert("Something went wrong while getting categories"));
42 +     return () => {
43 +       setCategories([]);
44 +       setToken("");
45 +     };
46 +   }, []);
47 +
48 +   const addCategory = () => {
49 +     const category = {
50 +       name: categoryName,
51 +     };
52 +     const config = {
53 +       headers: {
54 +         Authorization: `Bearer ${token}`,
55 +       },
56 +     };
57 +
58 +     axios
59 +       .post(`${BaseURL}/categories`, category, config)
60 +       .then((res) => setCategories([...categories, res.data]))
61 +       .catch((err) => alert("Error adding category"));
62 +     setCategoryName("");
63 +   };
64 +
65 +   const deleteCategory = (id) => {
66 +     const config = {
67 +       headers: {
68 +         Authorization: `Bearer ${token}`,
69 +       },
70 +     };
71 +
72 +     axios
73 +       .delete(`${BaseURL}/categories/${id}`, config)
74 +       .then((res) => {
75 +         const newCategories = categories.filter((item) => item._id !== id);
76 +         setCategories(newCategories);
77 +       })
78 +       .catch((err) => alert("Couldn't delete category"));
79 +   };
80 + }
```

```
5 80 return {
6 - <View>
7 - <Text>Categories screen</Text>
91 + <View style={{ position: "relative", height: "100%" }}>
92 + <View style={{ margin: 60 }}>
93 + <FlatList
94 +   data={categories}
95 +   renderItem={({ item, index }) => {
96 +     <Item item={item} index={index} delete={deleteCategory}></Item>
97 +   }}
98 +   keyExtractor={({item}) => item.id)
99 + ></FlatList>
100 + </View>
101 + <View style={styles.bottomBar}>
102 + <View>
103 + <Text>Add Category</Text>
104 + </View>
105 + <View style={{ width: width / 2.5 }}>
106 + <TextInput
107 +   value={categoryName}
108 +   onChangeText={({text}) => setCategoryName(text)}
109 +   style={styles.input}
110 + ></TextInput>
111 + </View>
112 + <View>
113 + <ShopButton style={styles.primary} onPress={addCategory}>
114 +   <Text style={{ color: "white", fontWeight: "bold" }}>Submit</Text>
115 + </ShopButton>
116 + </View>
117 + </View>
118 + </View>
119 + }
120 + };
121 + ];
122 +
123 +
124 +
125 +
126 +
127 +
128 +
129 +
130 +
131 +
132 +
133 +
134 +
135 +
136 +
137 +
138 +
139 +
140 +
141 +
142 +
143 +
144 +
145 +
146 +
147 +
148 +
149 +
150 +
151 +
152 +
153 +
154 +
155 +
156 +
157 +
158 +
159 +
160 +
161 +
162 +
163 +
164 +
165 +
166 +
167 +
168 +
169 +
170 +
171 +
172 +
173 +
174 +
175 +
176 +
177 +
178 +
179 +
180 +
181 +
182 +
183 +
184 +
185 +
186 +
187 +
188 +
189 +
190 +
191 +
192 +
193 +
194 +
195 +
196 +
197 +
198 +
199 +
200 +
201 +
202 +
203 +
204 +
205 +
206 +
207 +
208 +
209 +
210 +
211 +
212 +
213 +
214 +
215 +
216 +
217 +
218 +
219 +
220 +
221 +
222 +
223 +
224 +
225 +
226 +
227 +
228 +
229 +
230 +
231 +
232 +
233 +
234 +
235 +
236 +
237 +
238 +
239 +
240 +
241 +
242 +
243 +
244 +
245 +
246 +
247 +
248 +
249 +
250 +
251 +
252 +
253 +
254 +
255 +
256 +
257 +
258 +
259 +
260 +
261 +
262 +
263 +
264 +
265 +
266 +
267 +
268 +
269 +
270 +
271 +
272 +
273 +
274 +
275 +
276 +
277 +
278 +
279 +
280 +
281 +
282 +
283 +
284 +
285 +
286 +
287 +
288 +
289 +
290 +
291 +
292 +
293 +
294 +
295 +
296 +
297 +
298 +
299 +
300 +
301 +
302 +
303 +
304 +
305 +
306 +
307 +
308 +
309 +
310 +
311 +
312 +
313 +
314 +
315 +
316 +
317 +
318 +
319 +
320 +
321 +
322 +
323 +
324 +
325 +
326 +
327 +
328 +
329 +
330 +
331 +
332 +
333 +
334 +
335 +
336 +
337 +
338 +
339 +
340 +
341 +
342 +
343 +
344 +
345 +
346 +
347 +
348 +
349 +
350 +
351 +
352 +
353 +
354 +
355 +
356 +
357 +
358 +
359 +
360 +
361 +
362 +
363 +
364 +
365 +
366 +
367 +
368 +
369 +
370 +
371 +
372 +
373 +
374 +
375 +
376 +
377 +
378 +
379 +
380 +
381 +
382 +
383 +
384 +
385 +
386 +
387 +
388 +
389 +
390 +
391 +
392 +
393 +
394 +
395 +
396 +
397 +
398 +
399 +
400 +
401 +
402 +
403 +
404 +
405 +
406 +
407 +
408 +
409 +
410 +
411 +
412 +
413 +
414 +
415 +
416 +
417 +
418 +
419 +
420 +
421 +
422 +
423 +
424 +
425 +
426 +
427 +
428 +
429 +
430 +
431 +
432 +
433 +
434 +
435 +
436 +
437 +
438 +
439 +
440 +
441 +
442 +
443 +
444 +
445 +
446 +
447 +
448 +
449 +
450 +
451 +
452 +
453 +
454 +
455 +
456 +
457 +
458 +
459 +
460 +
461 +
462 +
463 +
464 +
465 +
466 +
467 +
468 +
469 +
470 +
471 +
472 +
473 +
474 +
475 +
476 +
477 +
478 +
479 +
480 +
481 +
482 +
483 +
484 +
485 +
486 +
487 +
488 +
489 +
490 +
491 +
492 +
493 +
494 +
495 +
496 +
497 +
498 +
499 +
500 +
501 +
502 +
503 +
504 +
505 +
506 +
507 +
508 +
509 +
510 +
511 +
512 +
513 +
514 +
515 +
516 +
517 +
518 +
519 +
520 +
521 +
522 +
523 +
524 +
525 +
526 +
527 +
528 +
529 +
530 +
531 +
532 +
533 +
534 +
535 +
536 +
537 +
538 +
539 +
540 +
541 +
542 +
543 +
544 +
545 +
546 +
547 +
548 +
549 +
550 +
551 +
552 +
553 +
554 +
555 +
556 +
557 +
558 +
559 +
560 +
561 +
562 +
563 +
564 +
565 +
566 +
567 +
568 +
569 +
570 +
571 +
572 +
573 +
574 +
575 +
576 +
577 +
578 +
579 +
580 +
581 +
582 +
583 +
584 +
585 +
586 +
587 +
588 +
589 +
590 +
591 +
592 +
593 +
594 +
595 +
596 +
597 +
598 +
599 +
600 +
601 +
602 +
603 +
604 +
605 +
606 +
607 +
608 +
609 +
610 +
611 +
612 +
613 +
614 +
615 +
616 +
617 +
618 +
619 +
620 +
621 +
622 +
623 +
624 +
625 +
626 +
627 +
628 +
629 +
630 +
631 +
632 +
633 +
634 +
635 +
636 +
637 +
638 +
639 +
640 +
641 +
642 +
643 +
644 +
645 +
646 +
647 +
648 +
649 +
650 +
651 +
652 +
653 +
654 +
655 +
656 +
657 +
658 +
659 +
660 +
661 +
662 +
663 +
664 +
665 +
666 +
667 +
668 +
669 +
670 +
671 +
672 +
673 +
674 +
675 +
676 +
677 +
678 +
679 +
680 +
681 +
682 +
683 +
684 +
685 +
686 +
687 +
688 +
689 +
690 +
691 +
692 +
693 +
694 +
695 +
696 +
697 +
698 +
699 +
700 +
701 +
702 +
703 +
704 +
705 +
706 +
707 +
708 +
709 +
710 +
711 +
712 +
713 +
714 +
715 +
716 +
717 +
718 +
719 +
720 +
721 +
722 +
723 +
724 +
725 +
726 +
727 +
728 +
729 +
730 +
731 +
732 +
733 +
734 +
735 +
736 +
737 +
738 +
739 +
740 +
741 +
742 +
743 +
744 +
745 +
746 +
747 +
748 +
749 +
750 +
751 +
752 +
753 +
754 +
755 +
756 +
757 +
758 +
759 +
760 +
761 +
762 +
763 +
764 +
765 +
766 +
767 +
768 +
769 +
770 +
771 +
772 +
773 +
774 +
775 +
776 +
777 +
778 +
779 +
780 +
781 +
782 +
783 +
784 +
785 +
786 +
787 +
788 +
789 +
790 +
791 +
792 +
793 +
794 +
795 +
796 +
797 +
798 +
799 +
800 +
801 +
802 +
803 +
804 +
805 +
806 +
807 +
808 +
809 +
810 +
811 +
812 +
813 +
814 +
815 +
816 +
817 +
818 +
819 +
820 +
821 +
822 +
823 +
824 +
825 +
826 +
827 +
828 +
829 +
830 +
831 +
832 +
833 +
834 +
835 +
836 +
837 +
838 +
839 +
840 +
841 +
842 +
843 +
844 +
845 +
846 +
847 +
848 +
849 +
850 +
851 +
852 +
853 +
854 +
855 +
856 +
857 +
858 +
859 +
860 +
861 +
862 +
863 +
864 +
865 +
866 +
867 +
868 +
869 +
870 +
871 +
872 +
873 +
874 +
875 +
876 +
877 +
878 +
879 +
880 +
881 +
882 +
883 +
884 +
885 +
886 +
887 +
888 +
889 +
890 +
891 +
892 +
893 +
894 +
895 +
896 +
897 +
898 +
899 +
900 +
901 +
902 +
903 +
904 +
905 +
906 +
907 +
908 +
909 +
910 +
911 +
912 +
913 +
914 +
915 +
916 +
917 +
918 +
919 +
920 +
921 +
922 +
923 +
924 +
925 +
926 +
927 +
928 +
929 +
930 +
931 +
932 +
933 +
934 +
935 +
936 +
937 +
938 +
939 +
940 +
941 +
942 +
943 +
944 +
945 +
946 +
947 +
948 +
949 +
950 +
951 +
952 +
953 +
954 +
955 +
956 +
957 +
958 +
959 +
960 +
961 +
962 +
963 +
964 +
965 +
966 +
967 +
968 +
969 +
970 +
971 +
972 +
973 +
974 +
975 +
976 +
977 +
978 +
979 +
980 +
981 +
982 +
983 +
984 +
985 +
986 +
987 +
988 +
989 +
990 +
991 +
992 +
993 +
994 +
995 +
996 +
997 +
998 +
999 +
1000 +
```

3.2.18. dodavanje narudžbi u adminal panel i korisničkom profilu - added orders to admin panel and user profile

Svaki korisnik želi da prati status svojih narudžbi, te se zbog toga dodaje ta funkcionalost uz korisnikov profil. Administar je onaj koji upravlja narudžbama i mijenja njihov status, tj. prihvata, odbija i isporučuje. Koristi se komponenta OrderCard.js i za admin panel i za korisnički profil.

```
184 Shared/orderCard.js
... @@ -0,0 +1,184 @@
1 + import React, { useState, useEffect } from "react";
2 + import { View, Text, StyleSheet } from "react-native";
3 + import { Picker } from "native-base";
4 + import Icon from "react-native-vector-icons/FontAwesome";
5 + import TrafficLight from "../StyledComponents/TrafficLight";
6 + import ShopButton from "../StyledComponents/ShopButton";
7 + import Toast from "react-native-toast-message";
8 +
9 + import AsyncStorage from "react-native-async-storage/async-storage";
10 + import axios from "axios";
11 + import baseUrl from "../assets/common/baseUrl";
12 +
13 + const codes = [
14 +   {
15 +     name: "pending",
16 +     code: "3",
17 +   },
18 +   {
19 +     name: "shipped",
20 +     code: "2",
21 +   },
22 +   {
23 +     name: "delivered",
24 +     code: "1",
25 +   },
26 + ];
27 +
28 + export default function OrderCard(props) {
29 +   const [orderStatus, setOrderStatus] = useState();
30 +   const [statusText, setStatusText] = useState();
31 +   const [statusChange, setStatusChange] = useState();
32 +   const [token, setToken] = useState();
33 +   const [cardColor, setCardColor] = useState();
34 +
35 +   useEffect(() => {
36 +     if (props.editMode) {
37 +       AsyncStorage.getItem("jwt")
38 +         .then((res) => setToken(res))
39 +         .catch((err) => console.log(err));
40 +     }
41 +     if (props.status == "3") {
42 +       setOrderStatus(<TrafficLight unavailable></TrafficLight>);
43 +       setStatusText("pending");
44 +       setCardColor("#E74C3C");
45 +     } else if (props.status == "2") {
46 +       setOrderStatus(<TrafficLight limited></TrafficLight>);
47 +       setStatusText("shipped");
48 +       setCardColor("#F1C40F");
49 +     } else if (props.status == "1") {
50 +       setOrderStatus(<TrafficLight available></TrafficLight>);
51 +       setStatusText("delivered");
52 +       setCardColor("#E74C3C");
53 +     }
54 +     return () => {
55 +       setOrderStatus();
56 +       setStatusText();
57 +       setCardColor();
58 +     };
59 +   }, []);
```

```

60 +
61 +   const updateOrder = () => {
62 +     const config = {
63 +       headers: {
64 +         Authorization: `Bearer ${token}`,
65 +       },
66 +     };
67 +
68 +     const order = {
69 +       city: props.city,
70 +       country: props.country,
71 +       dateOrdered: props.dateOrdered,
72 +       orderItems: props.orderItems,
73 +       phone: props.phone,
74 +       shippingAddress1: props.shippingAddress1,
75 +       shippingAddress2: props.shippingAddress2,
76 +       status: statusChange,
77 +       totalPrice: props.totalPrice,
78 +       user: props.user,
79 +       zip: props.zip,
80 +     };
81 +
82 +     axios
83 +       .put(`${baseUrl}/orders/${props._id}`, order, config)
84 +       .then((res) => {
85 +         if (res.status == 200 || res.status == 201) {
86 +           Toast.show({
87 +             topOffset: 60,
88 +             type: "success",
89 +             text1: "Order successfully updated",
90 +             text2: "",
91 +           });
92 +           if (statusChange == "3") {
93 +             setOrderStatus(<TrafficLight unavailable></TrafficLight>);
94 +             setStatusText("pending");
95 +             setCardColor("#E74C3C");
96 +           } else if (statusChange == "2") {
97 +             setOrderStatus(<TrafficLight limited></TrafficLight>);
98 +             setStatusText("shipped");
99 +             setCardColor("#F1C40F");
100 +           } else if (statusChange == "1") {
101 +             setOrderStatus(<TrafficLight available></TrafficLight>);
102 +             setStatusText("delivered");
103 +             setCardColor("#2ECC71");
104 +           }
105 +         }
106 +       })
107 +       .catch((err) => {
108 +         Toast.show({
109 +           topOffset: 60,
110 +           type: "error",
111 +           text1: "Something went wrong",
112 +           text2: "Please try again",
113 +         });
114 +       });
115 +   };
116 +

```

```

117 +   return (
118 +     <View style={styles.container, { backgroundColor: cardColor }}>
119 +       <View style={styles.container}>
120 +         <Text>Order Number: #{props._id}</Text>
121 +       </View>
122 +       <View style={{ marginTop: 10 }}>
123 +         <Text>
124 +           Status: {statusText} {orderStatus}
125 +         </Text>
126 +         <Text>
127 +           Address: {props.shippingAddress1} {props.shippingAddress2}
128 +         </Text>
129 +         <Text>City: {props.city}</Text>
130 +         <Text>Country: {props.country}</Text>
131 +         <Text>Date ordered: {props.dateOrdered.split("T")[0]}</Text>
132 +         <View style={styles.priceContainer}>
133 +           <Text>Price: </Text>
134 +           <Text style={styles.price}>${ props.totalPrice}</Text>
135 +         </View>
136 +         {props.editMode ? (
137 +           <View>
138 +             <Picker
139 +               mode="dropdown"
140 +               iosIcon={<Icon color={"#007aff"} name="arrow-down"></Icon>}
141 +               style={{ width: undefined, height: 20 }}
142 +               selectedValue={statusChange}
143 +               placeholder="Change status"
144 +               placeholderIconColor={{ color: "#007aff" }}
145 +               onValueChange={(e) => setStatusChange(e)}
146 +             >
147 +               {codes.map((c) => (
148 +                 <Picker.Item
149 +                   key={c.code}
150 +                   label={c.name}
151 +                   value={c.code}
152 +                 ></Picker.Item>
153 +               ))}
154 +             </Picker>
155 +             <ShopButton secondary large onPress={updateOrder}>
156 +               <Text>Update</Text>
157 +             </ShopButton>
158 +           </View>
159 +         ) : null}
160 +       </View>
161 +     </View>
162 +   );
163 + }
164 +

```

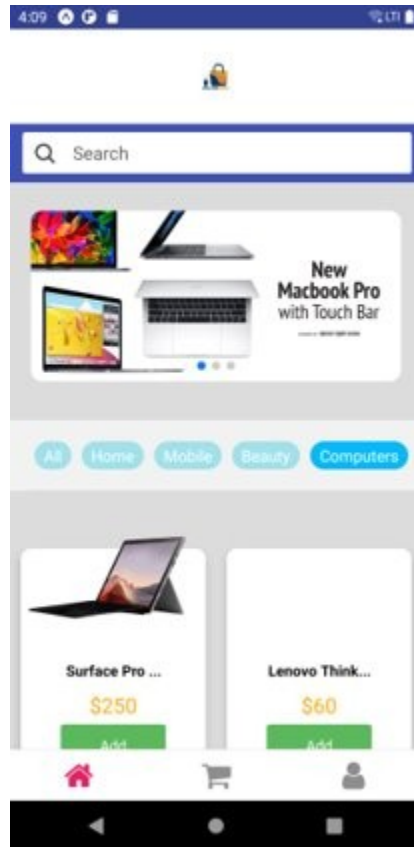
```

165 + const styles = StyleSheet.create({
166 +   container: {
167 +     padding: 20,
168 +     margin: 10,
169 +     borderRadius: 10,
170 +   },
171 +   title: {
172 +     backgroundColor: "#62B1F6",
173 +     padding: 5,
174 +   },
175 +   priceContainer: {
176 +     marginTop: 10,
177 +     alignSelf: "flex-end",
178 +     flexDirection: "row",
179 +   },
180 +   price: {
181 +     color: "white",
182 +     fontWeight: "bold",
183 +   },
184 + });

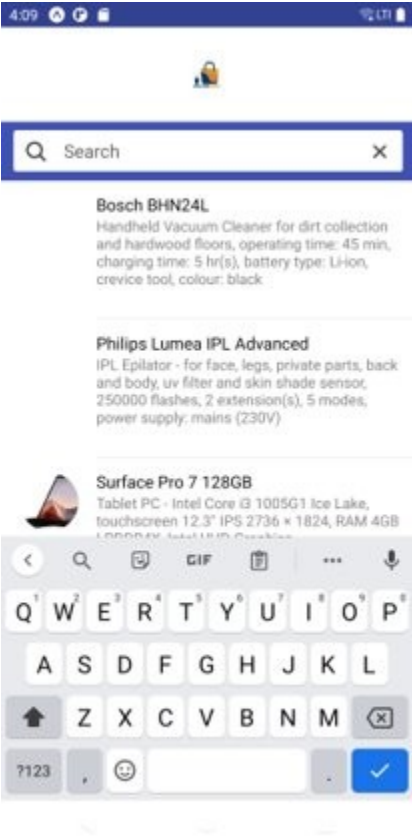
```

4. Izled aplikacije

4.1. Početna strana



4.2. Search bar



4.3. Pojedinačni produkt

4:09

Surface Pro 7 128GB

ASUS

Availability: Available

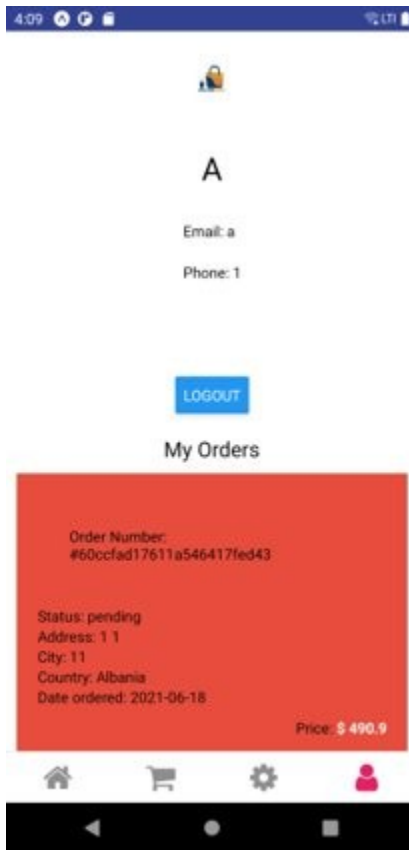
Tablet PC - Intel Core i3 1005G1 Ice Lake, touchscreen 12.3" IPS 2736 x 1824, RAM 4GB LPDDR4X, Intel UHD Graphics

\$ 250

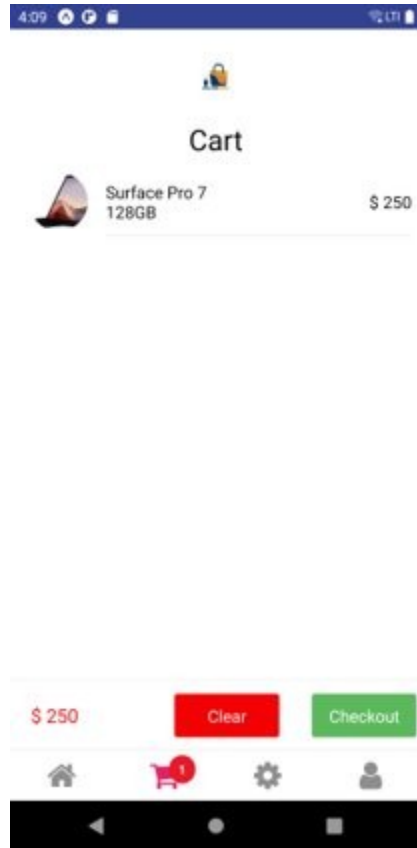
Add

Home Cart Profile

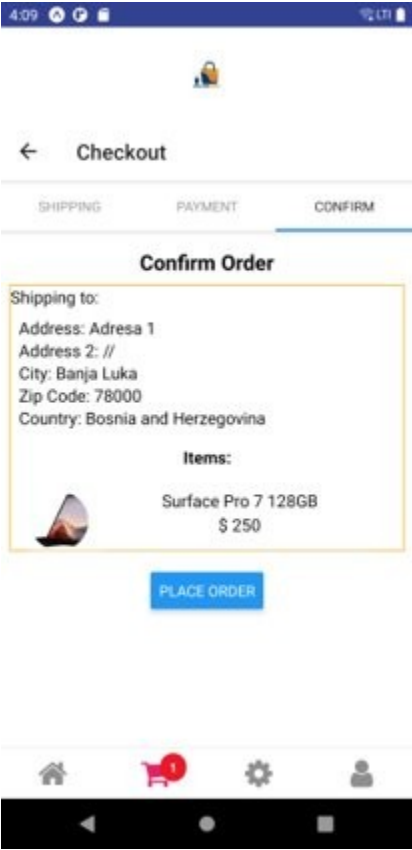
4.4. Korisnički profil



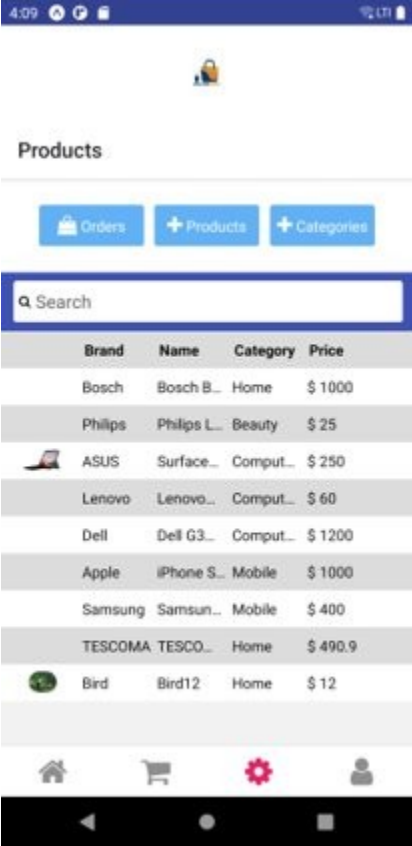
4.5. Korpa



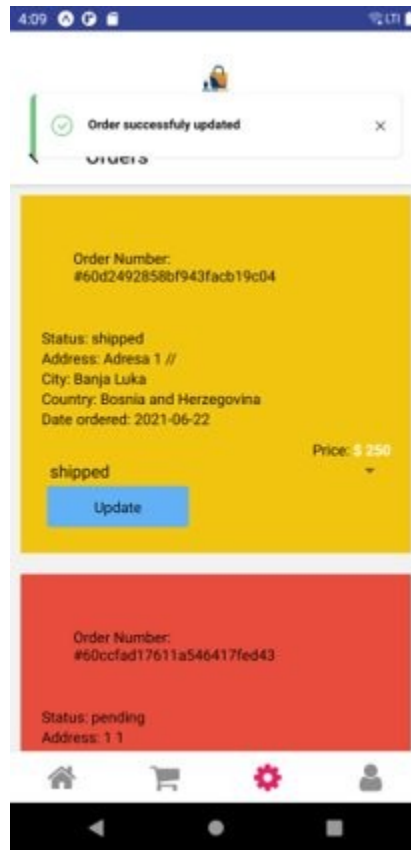
4.6. Checkout



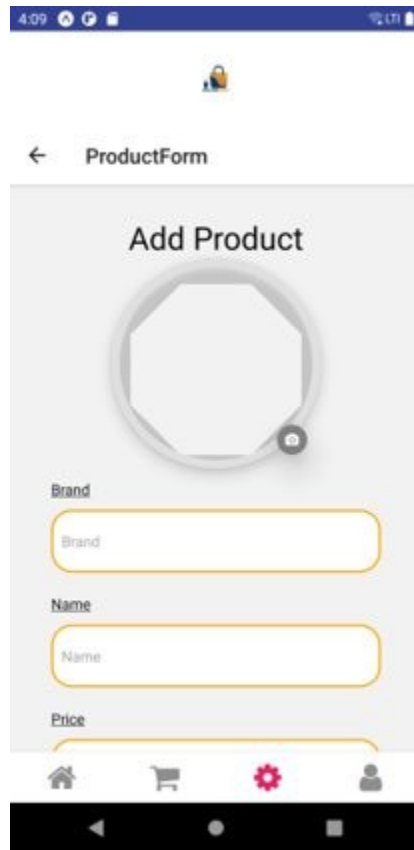
4.7. Admin Panel



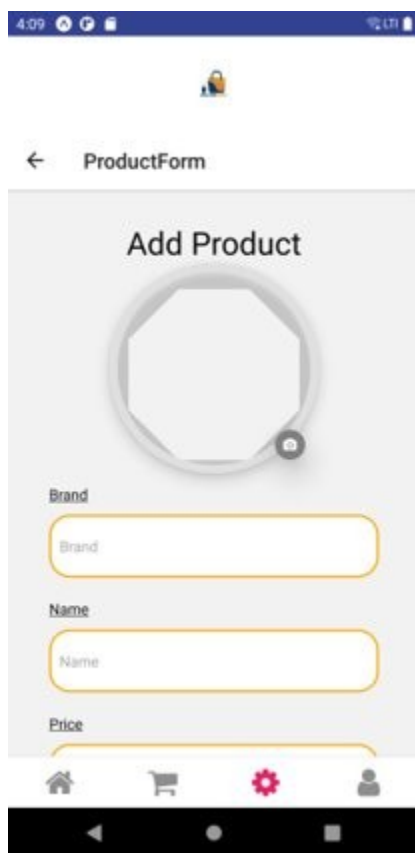
4.8. Narudžbe – Admin strana



4.9. Kreiranje novog produkta



4.10. Kategorije u admin panelu



5. Zaključak

Korištenjem MongoDB, ExpressJS, React Native i NodeJS tehnologija je uspješno izrađena mobilna aplikacija koja se može koristiti na Android i iOS uređajima bez ikakvih problema. Prednost korištenja navedenih tehnologija je to što je trenutnu aplikaciju vrlo lagano konvertovati i u web aplikaciju, koristeći React biblioteku. Danas je moguće pronaći jako puno softver inženjera koji poznaju tehnologije korištene u ovom projektu, te dovođenje novih ljudi u ovakav ili sličan projekat ne predstavlja veliki problem.

Razvojni proces je protekao vrlo brzo, što je takođe jedna od prednosti ovih tehnologija. Mnogi resursi koji se koriste u ovoj aplikaciju, mogu da se dijele i sa drugim aplikacijama, kao što su komponente i middleware-i.

Ukoliko se deploja backend u produkciju, tu je veliki izbor popularnih servisa poput AWS ili Azure, a da bi se podesilo lokalno razvojno okruženje sve što treba da se uradi je da se klonira projekat iz GitHub repozitorijuma i pokrene komanda „npm install“ (ukoliko je već instaliran nodejs i npm lokalno).

Gotov projekat se nalazi na sledećim linkovima:

- https://github.com/simicjovo/mern_native_shop
- https://github.com/simicjovo/mern_native_shop_frontend

6. Literatura

- [1] https://en.wikipedia.org/wiki/React_Native, pristupljeno 4.9.2021.
- [2] <https://en.wikipedia.org/wiki/Express.js>, pristupljeno 4.9.2021.
- [3] <https://en.wikipedia.org/wiki/MongoDB>, pristupljeno 4.9.2021.
- [4] <https://www.npmjs.com/package/dotenv>, pristupljeno 4.9.2021.
- [5] <https://www.npmjs.com/package/mongoose>, pristupljeno 4.9.2021.
- [6] <https://www.npmjs.com/package/bcryptjs>, pristupljeno 4.9.2021.
- [7] <https://www.npmjs.com/package/jsonwebtoken>, pristupljeno 4.9.2021.
- [8] <https://www.npmjs.com/package/multer>, pristupljeno 4.9.2021.
- [9] <https://www.npmjs.com/package/expo-cli>, pristupljeno 4.9.2021.
- [10] <https://www.npmjs.com/package/native-base>, pristupljeno 4.9.2021.
- [11] <https://www.npmjs.com/package/react-native-swiper>, pristupljeno 4.9.2021.
- [12] <https://www.npmjs.com/package/react-navigation>, pristupljeno 4.9.2021.
- [13] <https://www.npmjs.com/package/react-native-vector-icons>, pristupljeno 4.9.2021.
- [14] <https://www.npmjs.com/package/redux>, pristupljeno 4.9.2021.
- [15] <https://www.npmjs.com/package/react-native-toast-message>, pristupljeno 4.9.2021.
- [16] <https://www.npmjs.com/package/styled-components>, pristupljeno 4.9.2021.
- [17] <https://www.npmjs.com/package/expo-image-picker>, pristupljeno 4.9.2021.