PANEVROPSKI UNIVERZITET APEIRON
FAKULTET INFORMACIONIH TEHNOLOGIJA

Redovne studije

Smjer „Programiranje i softversko inžinjerstvo”

„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

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:

[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

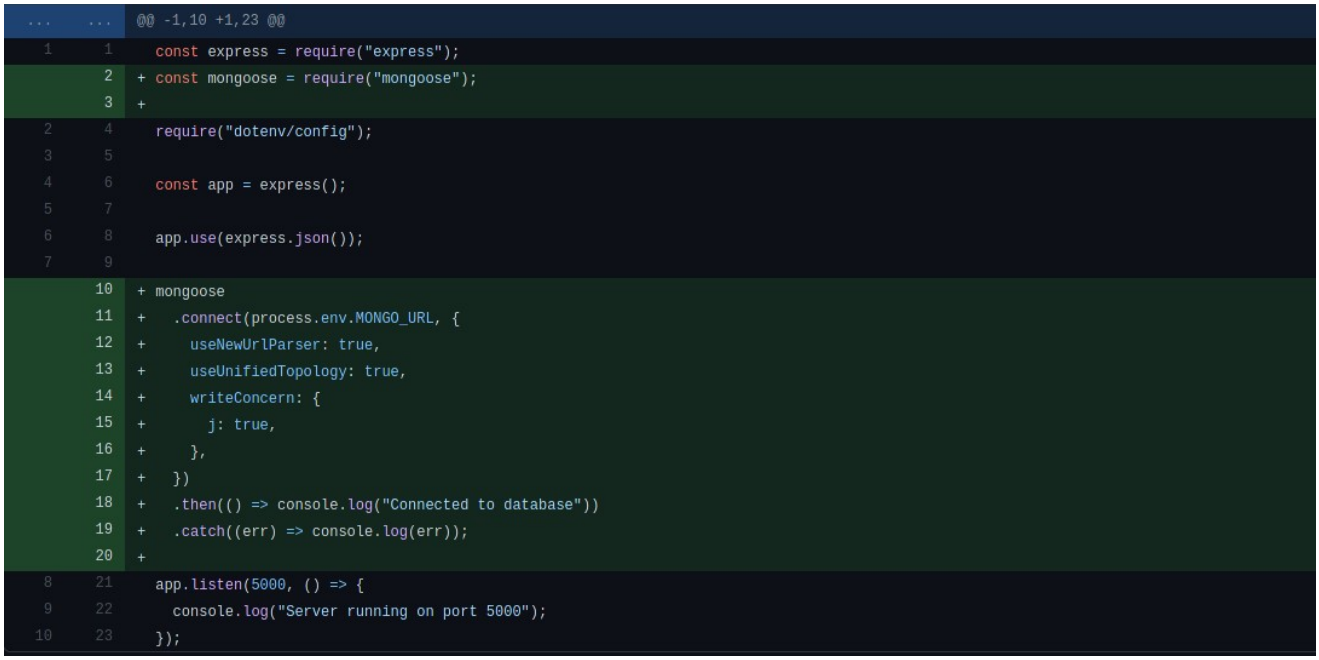
Razvojni proces započinje korištenjem komande “mkdir backend”, “cd backend” i “npm init -y” za inicijalizaciju node projekta. Zatim se instaliraju paketi express i dotenv[4] sa komandom „npm install express dotenv“. Dotenv paket se koristi za .env fajl u kojem su sačuvani senzitivini podaci kao šifre. 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 kreairanje 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 @@
  const express = require("express");
+ const mongoose = require("mongoose");
+
  require("dotenv/config");

  const app = express();

  app.use(express.json());

+ mongoose
+   .connect(process.env.MONGO_URL, {
+     useNewUrlParser: true,
+     useUnifiedTopology: true,
+     writeConcern: {
+       j: true,
+     },
+   })
+   .then(() => console.log("Connected to database"))
+   .catch((err) => console.log(err));
+
  app.listen(5000, () => {
    console.log("Server running on port 5000");
  });
```

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.

backend/models/category.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const categorySchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Category = mongoose.model("Category", categorySchema);
```

backend/models/order.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const orderSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Order = mongoose.model("Order", orderSchema);
```

backend/models/product.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const productSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Product = mongoose.model("Product", productSchema);
```

backend/models/user.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const userSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.User = mongoose.model("User", userSchema);
```

backend/routes/categories.js

```
@@ -0,0 +1,5 @@
+ const express = require("express");
+ const { Category } = require("../models/category.model");
+ const router = express.Router();
+
+ module.exports = router;
```

backend/routes/orders.js

```
@@ -0,0 +1,5 @@
+ const express = require("express");
+ const { Order } = require("../models/order.model");
+ const router = express.Router();
+
+ module.exports = router;
```

backend/routes/product.js

```
@@ -0,0 +1,5 @@
+ const express = require("express");
+ const { Product } = require("../models/product.model");
+ const router = express.Router();
+
+ module.exports = router;
```

backend/routes/users.js

```
@@ -0,0 +1,5 @@
+ const express = require("express");
+ const { User } = require("../models/user.model");
+ const router = express.Router();
+
+ module.exports = router;
```

7 backend/models/category.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const categorySchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Category = mongoose.model("Category", categorySchema);
```

7 backend/models/order.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const orderSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Order = mongoose.model("Order", orderSchema);
```

7 backend/models/product.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const productSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.Product = mongoose.model("Product", productSchema);
```

7 backend/models/user.model.js

```
@@ -0,0 +1,7 @@
+ const mongoose = require("mongoose");
+
+ const userSchema = mongoose.Schema({
+   name: String,
+ });
+
+ exports.User = mongoose.model("User", userSchema);
```

12 backend/app.js

```
@@ -1,11 +1,23 @@
  const express = require("express");
  const mongoose = require("mongoose");
+ const cors = require("cors");

  require("dotenv/config");

  const app = express();

+ app.options("*", cors());
+
+ const productsRouter = require("./routes/product");
+ const categoriesRouter = require("./routes/categories");
+ const usersRouter = require("./routes/users");
+ const ordersRouter = require("./routes/orders");
+
  app.use(express.json());
+ app.use("/api/products", productsRouter);
+ app.use("/api/categories", categoriesRouter);
+ app.use("/api/users", usersRouter);
+ app.use("/api/orders", ordersRouter);

  mongoose
    .connect(process.env.MONGO_URL, {
```

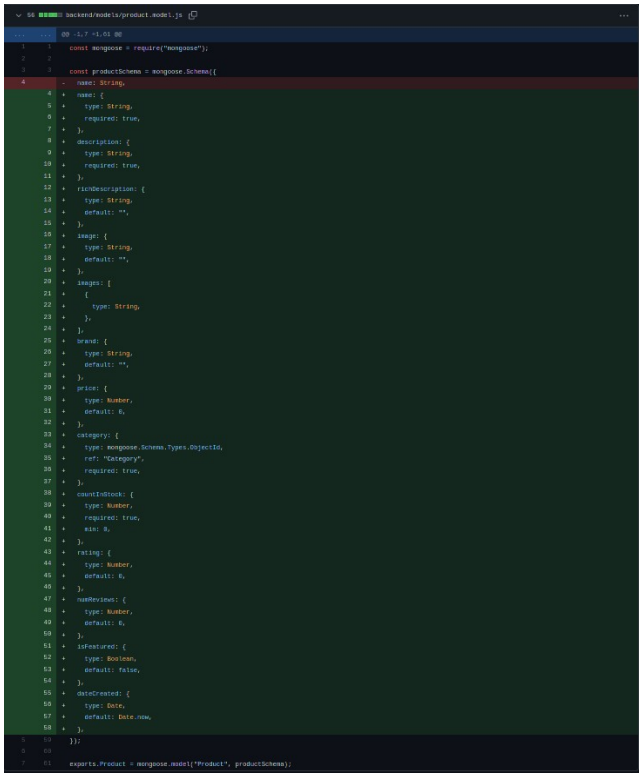
3.1.4. podešavanje šeme za produkte i kategorije - set up product and category schema

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

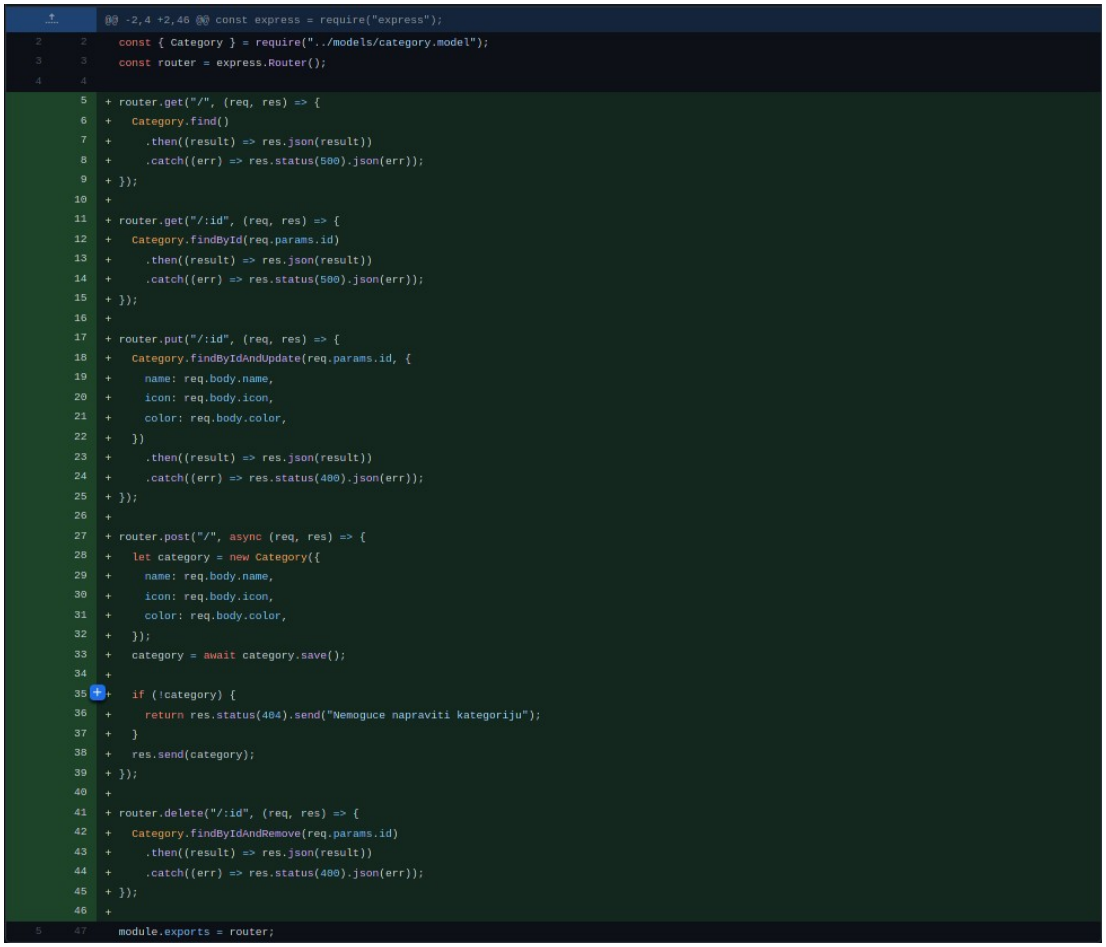
```
@@ -1,7 +1,16 @@
 const mongoose = require("mongoose");

 const categorySchema = mongoose.Schema({
-  name: String,
+  name: {
+    type: String,
+    required: true,
+  },
+  icon: {
+    type: String,
+  },
+  color: {
+    type: String,
+  },
 });

 exports.Category = mongoose.model("Category", categorySchema);
```

Produkti su malo kompleksiniji te će da imaju više parametara. Svaki produkt treba da ima ime, opis, detaljni opis, glavnu sliku (url), ostale slike (niz urlova), 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:



3.1.5. završavanje rute i šema za produkte i kateogrije - 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.



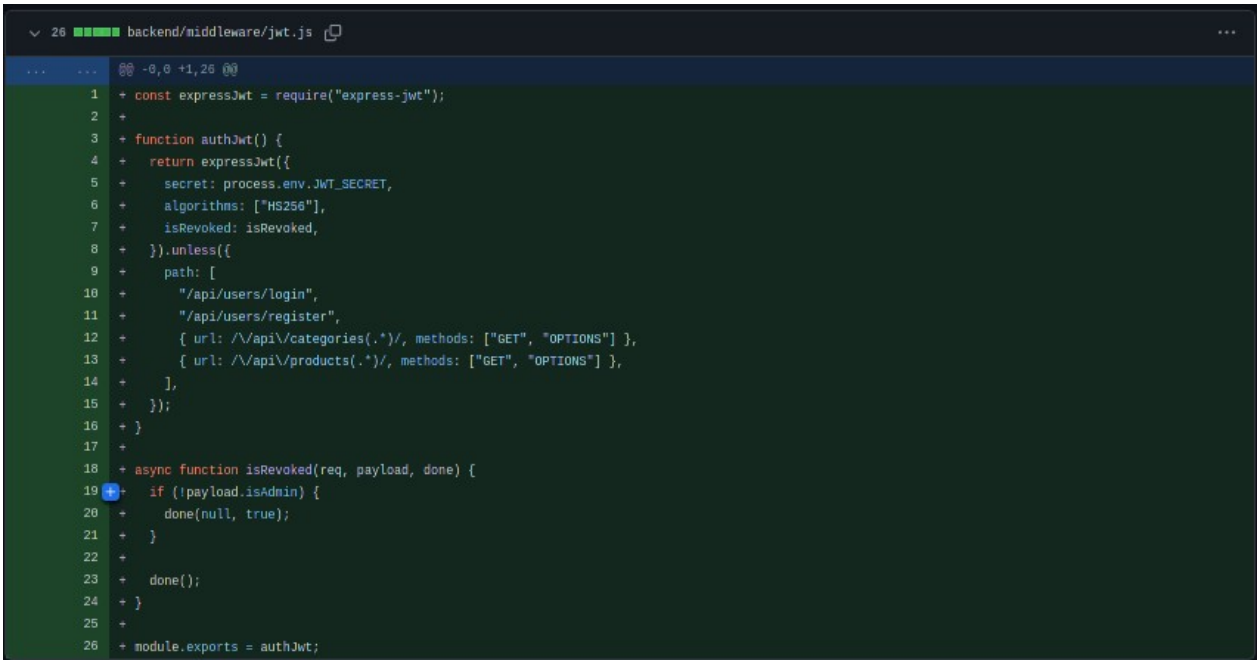
```
@@ -2,4 +2,46 @@ const express = require("express");
  const { Category } = require("../models/category.model");
  const router = express.Router();

+ router.get("/", (req, res) => {
+   Category.find()
+     .then((result) => res.json(result))
+     .catch((err) => res.status(500).json(err));
+ });
+
+ router.get("/:id", (req, res) => {
+   Category.findById(req.params.id)
+     .then((result) => res.json(result))
+     .catch((err) => res.status(500).json(err));
+ });
+
+ router.put("/:id", (req, res) => {
+   Category.findByIdAndUpdate(req.params.id, {
+     name: req.body.name,
+     icon: req.body.icon,
+     color: req.body.color,
+   })
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
+ router.post("/", async (req, res) => {
+   let category = new Category({
+     name: req.body.name,
+     icon: req.body.icon,
+     color: req.body.color,
+   });
+   category = await category.save();
+
+   if (!category) {
+     return res.status(404).send("Nemoguce napraviti kategoriju");
+   }
+   res.send(category);
+ });
+
+ router.delete("/:id", (req, res) => {
+   Category.findByIdAndRemove(req.params.id)
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
  module.exports = router;
```

91 backend/routes/product.js

```
@@ -1,5 +1,93 @@
const express = require("express");
const { Category } = require("../models/category.model");
const { Product } = require("../models/product.model");
const router = express.Router();

router.get("/", [req, res] => {
  let filter = {};
  if (req.query.categories) {
    filter = { category: req.query.categories.split(",") };
  }

  Product.find(filter)
    .populate("category")
    .then((result) => res.json(result))
    .catch((err) => res.status(400).json(err));
});

router.get("/:id", (req, res) => {
  Product.findById(req.params.id)
    .populate("category")
    .then((result) => res.json(result))
    .catch((err) => res.status(400).json(err));
});

router.post("/", async (req, res) => {
  const category = await Category.findById(req.body.category);
  if (!category) return res.status(400).send("Bad category");

  const product = new Product({
    name: req.body.name,
    description: req.body.description,
    richDescription: req.body.richDescription,
    image: req.body.image,
    brand: req.body.brand,
    price: req.body.price,
    category: req.body.category,
    countInStock: req.body.countInStock,
    rating: req.body.rating,
    numReviews: req.body.numReviews,
    isFeatured: req.body.isFeatured,
  });
  product
    .save()
    .then((result) => res.json(result))
    .catch((err) => res.status(400).json(err));
});

router.put("/", async (req, res) => {
  const category = await Category.findById(req.body.category);
  if (!category) return res.status(400).send("Bad category");

  Product.findByIdAndUpdate(
    req.params.id,
    {
      name: req.body.name,
      description: req.body.description,
      richDescription: req.body.richDescription,
      image: req.body.image,
      brand: req.body.brand,
      price: req.body.price,
      category: req.body.category,
      countInStock: req.body.countInStock,
      rating: req.body.rating,
      numReviews: req.body.numReviews,
      isFeatured: req.body.isFeatured,
    },
    {
      new: true,
    }
  )
    .then((result) => res.json(result))
    .catch((err) => res.status(400).json(err));
});

router.delete("/:id", (req, res) => {
  Product.findByIdAndRemove(req.params.id)
    .then((result) => res.json(result))
    .catch((err) => res.status(400).json(err));
});

router.get("/get/count", async (req, res) => {
  const productCount = await Product.countDocuments((count) => count);

  if (!productCount) {
    res.status(400).json("Error");
  }
  res.json(productCount);
});

router.get("/get/featured", (req, res) => {
  Product.find({ isFeatured: true })
    .then((result) => res.json(result))
    .catch((err) => res.json(err));
});

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.



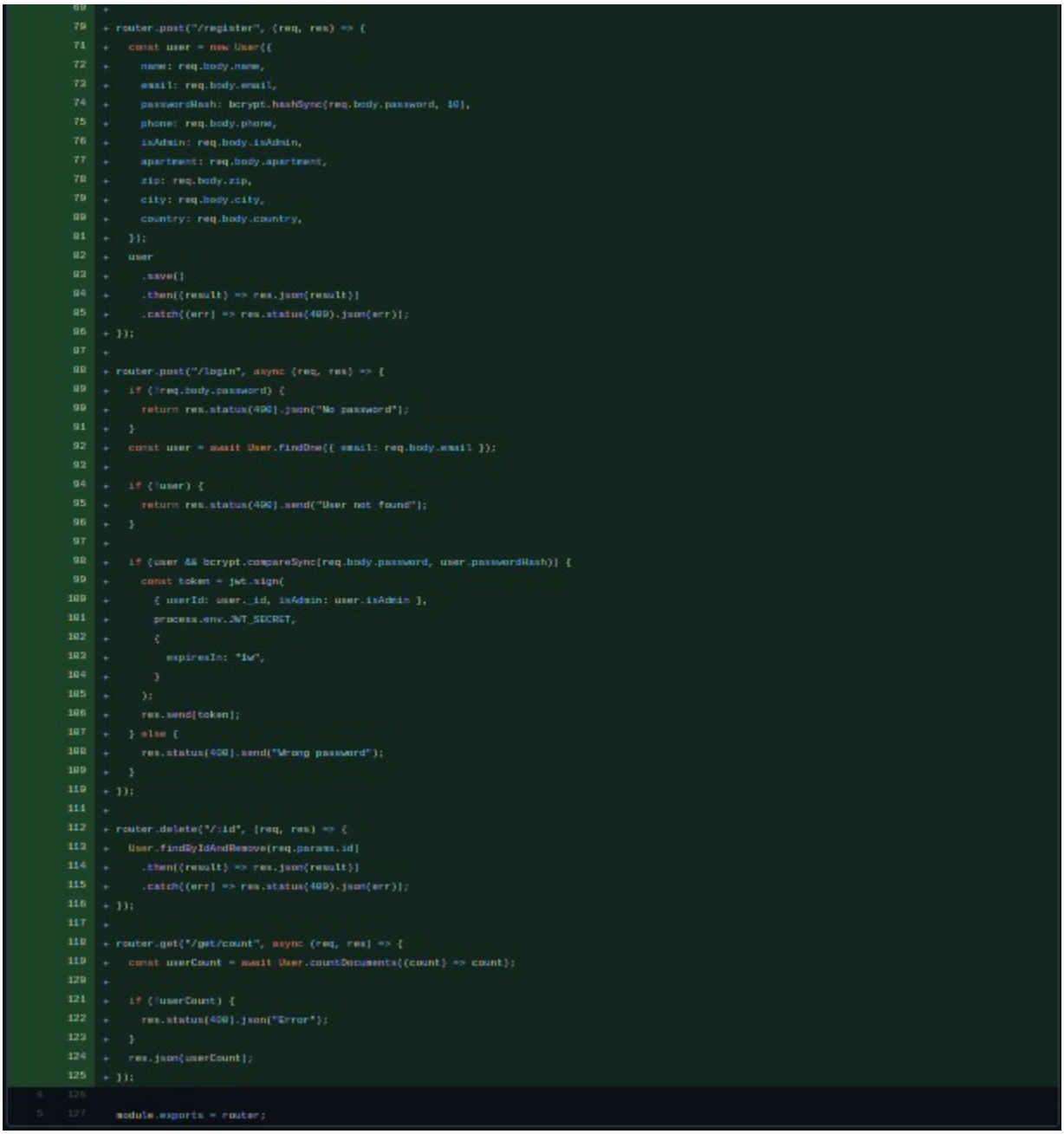
[6] https://www.npmjs.com/package/bcryptjs
[7] https://www.npmjs.com/package/jsonwebtoken

41 backend/models/user.model.js

```
@@ -1,7 +1,46 @@
 const mongoose = require("mongoose");

 const userSchema = mongoose.Schema({
-  name: String,
+  name: {
+    type: String,
+    required: true,
+  },
+  email: {
+    type: String,
+    required: true,
+  },
+  passwordHash: {
+    type: String,
+    required: true,
+  },
+  phone: {
+    type: String,
+    required: true,
+  },
+  isAdmin: {
+    type: Boolean,
+    required: true,
+  },
+  street: {
+    type: String,
+    default: "",
+  },
+  apartment: {
+    type: String,
+    default: "",
+  },
+  zip: {
+    type: String,
+    default: "",
+  },
+  city: {
+    type: String,
+    default: "",
+  },
+  country: {
+    type: String,
+    default: "",
+  },
 });

 exports.User = mongoose.model("User", userSchema);
```

122 backend/routes/users.js

```
@@ -1,5 +1,127 @@
 const express = require("express");
 const { User } = require("../models/user.model");
 const router = express.Router();
+const bcrypt = require("bcryptjs");
+const jwt = require("jsonwebtoken");
+
+router.get("/", (req, res) => {
+  User.find()
+    .select("-passwordHash")
+    .then((result) => res.json(result))
+    .catch((err) => res.status(400).json(err));
+});
+
+router.post("/", (req, res) => {
+  const user = new User({
+    name: req.body.name,
+    email: req.body.email,
+    passwordHash: bcrypt.hashSync(req.body.password, 10),
+    phone: req.body.phone,
+    isAdmin: req.body.isAdmin,
+    apartment: req.body.apartment,
+    zip: req.body.zip,
+    city: req.body.city,
+    country: req.body.country,
+  });
+  user
+    .save()
+    .then((result) => res.json(result))
+    .catch((err) => res.status(400).json(err));
+});
+
+router.get("/:id", (req, res) => {
+  User.findById(req.params.id)
+    .select("-passwordHash")
+    .then((result) => res.json(result))
+    .catch((err) => res.status(400).json(err));
+});
+
+router.put("/:id", async (req, res) => {
+  const userExist = await User.findById(req.params.id);
+  let newPassword;
+  if (req.body.password) {
+    newPassword = bcrypt.hashSync(req.body.password, 10);
+  } else {
+    newPassword = userExist.passwordHash;
+  }
+
+  const user = await User.findByIdAndUpdate(
+    req.params.id,
+    {
+      name: req.body.name,
+      email: req.body.email,
+      passwordHash: newPassword,
+      phone: req.body.phone,
+      isAdmin: req.body.isAdmin,
+      street: req.body.street,
+      apartment: req.body.apartment,
+      zip: req.body.zip,
+      city: req.body.city,
+      country: req.body.country,
+    },
+    { new: true }
+  );
+
+  if (!user) return res.status(400).send("the user cannot be created!");
+
+  res.send(user);
+});
+
```



```
+
+ router.post("/register", (req, res) => {
+   const user = new User({
+     name: req.body.name,
+     email: req.body.email,
+     passwordHash: bcrypt.hashSync(req.body.password, 10),
+     phone: req.body.phone,
+     isAdmin: req.body.isAdmin,
+     apartment: req.body.apartment,
+     zip: req.body.zip,
+     city: req.body.city,
+     country: req.body.country,
+   });
+   user
+     .save()
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
+ router.post("/login", async (req, res) => {
+   if (!req.body.password) {
+     return res.status(400).json("No password");
+   }
+   const user = await User.findOne({ email: req.body.email });
+
+   if (!user) {
+     return res.status(400).send("User not found");
+   }
+
+   if (user && bcrypt.compareSync(req.body.password, user.passwordHash)) {
+     const token = jwt.sign(
+       { userId: user._id, isAdmin: user.isAdmin },
+       process.env.JWT_SECRET,
+       {
+         expiresIn: "1w",
+       }
+     );
+     res.send(token);
+   } else {
+     res.status(400).send("Wrong password");
+   }
+ });
+
+ router.delete("/:id", (req, res) => {
+   User.findByIdAndRemove(req.params.id)
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
+ router.get("/get/count", async (req, res) => {
+   const userCount = await User.countDocuments((count) => count);
+
+   if (!userCount) {
+     res.status(400).json("Error");
+   }
+   res.json(userCount);
+ });

module.exports = router;
```

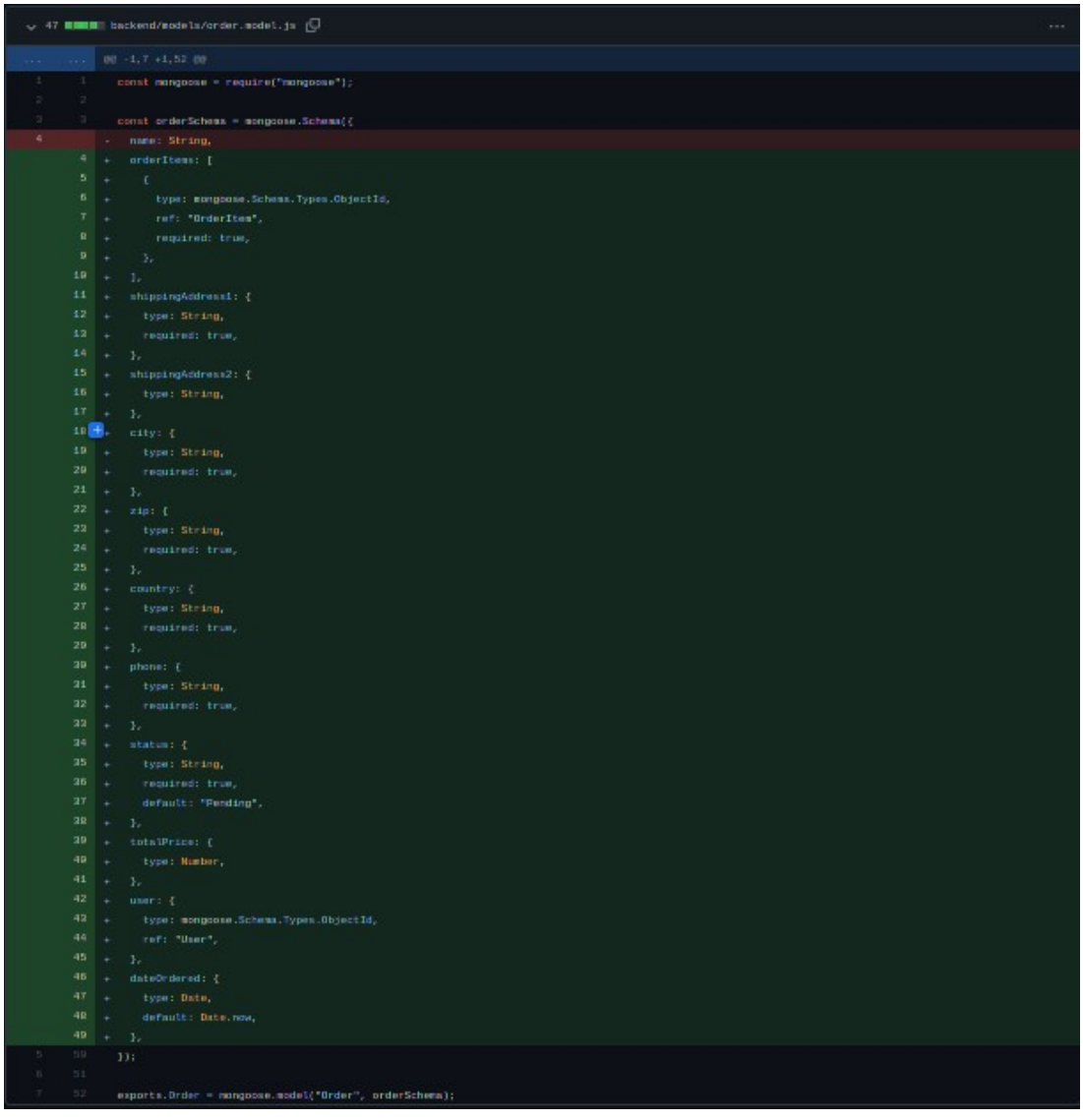
3.1.7. dodavnje order i orderitem šema - added order and orderitems schema

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.

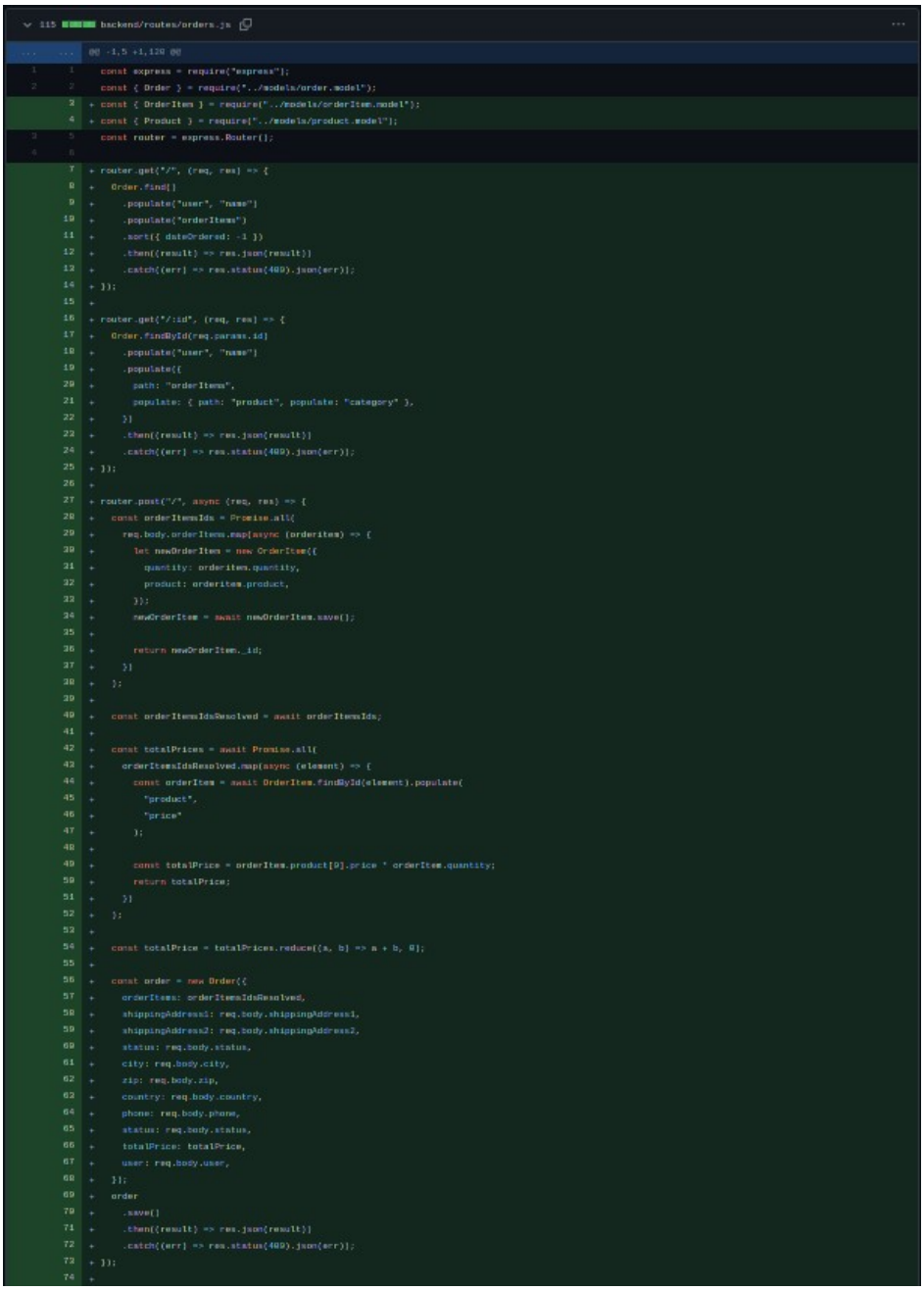
16 backend/models/orderItem.model.js

```
@@ -0,0 +1,16 @@
+ const mongoose = require("mongoose");
+
+ const orderItemSchema = mongoose.Schema({
+   quantity: {
+     type: Number,
+     required: true,
+   },
+   product: [
+     {
+       type: mongoose.Schema.Types.ObjectId,
+       ref: "Product",
+     },
+   ],
+ });
+
+ exports.OrderItem = mongoose.model("OrderItem", orderItemSchema);
```



3.1.8. završavanje order ruta - finished orders routes

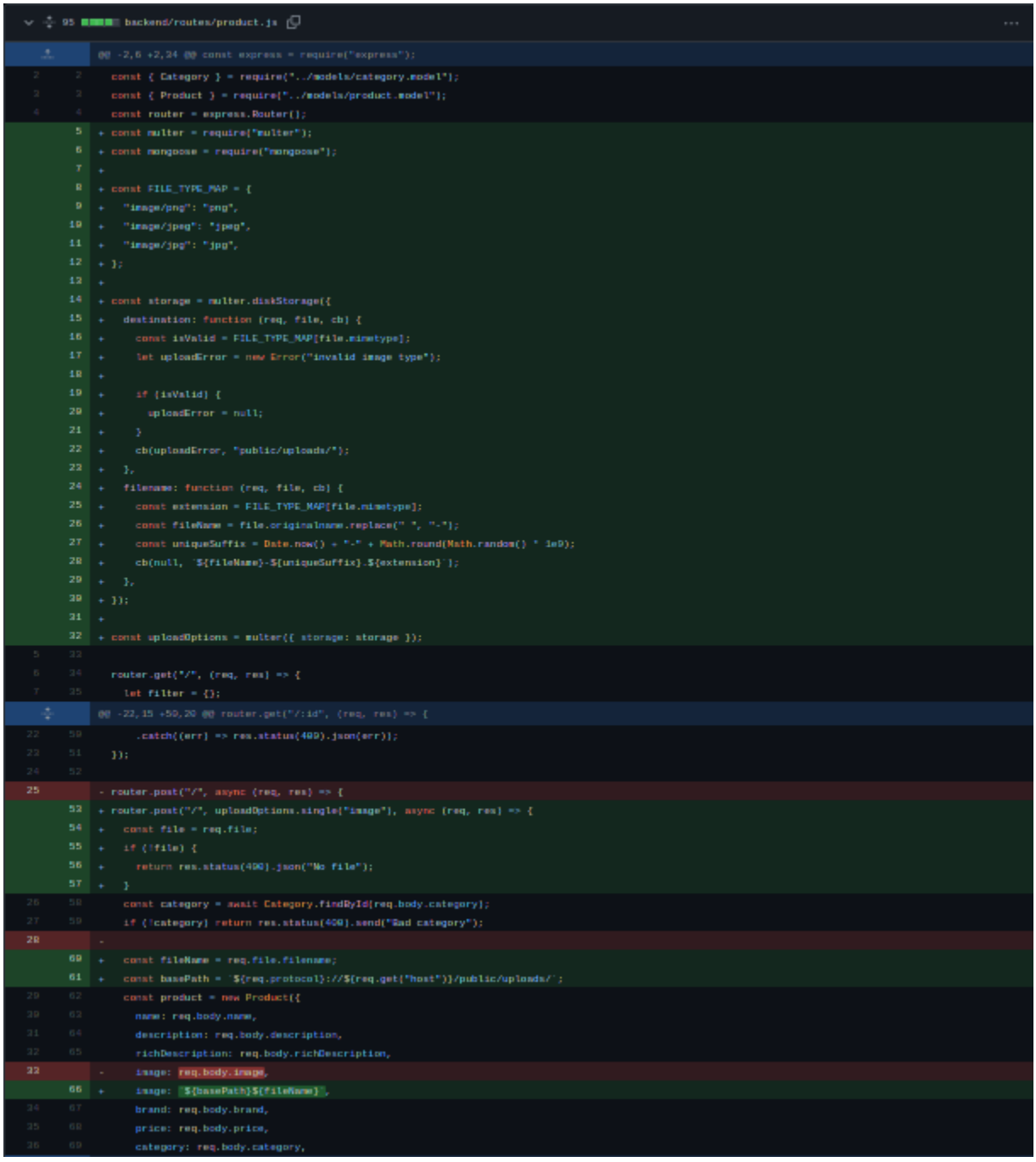
Kada se završe šeme za narudbže, potrebno je kreirati i rute na backendu u kojima se koriste te šeme. Dodaju se sve rute kao u kategorijama, a zatim se dodaju i ruta get „/get/usersorders/:userid“ na kojoj se mogu dohvatiti sve narudžbe za korisnika sa datim id-om u parametru.



```
+ router.put("/:id", async (req, res) => {
+   Order.findByIdAndUpdate(
+     req.params.id,
+     { status: req.body.status },
+     { new: true }
+   )
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
+ router.delete("/:id", (req, res) => {
+   Order.findByIdAndRemove(req.params.id)
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
+ router.get("/get/totalsales", async (req, res) => {
+   const totalSales = await Order.aggregate([
+     { $group: { _id: null, totalsales: { $sum: "$totalPrice" } } },
+   ]);
+   if (!totalSales) {
+     return res.status(400).json("Error");
+   }
+   res.json(totalSales);
+ });
+
+ router.get("/get/count", async (req, res) => {
+   const orderCount = await Order.countDocuments((count) => count);
+   if (!orderCount) {
+     res.status(500).json("error");
+   }
+   res.json(orderCount);
+ });
+
+ router.get("/get/userorders/:userid", (req, res) => {
+   Order.find({ user: req.params.userid })
+     .populate({
+       path: "orderItems",
+       populate: { path: "product", populate: "category" },
+     })
+     .sort({ dateOrdered: -1 })
+     .then((result) => res.json(result))
+     .catch((err) => res.status(400).json(err));
+ });
+
  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

```
@@ -2,6 +2,34 @@ const express = require("express");
 const { Category } = require("../models/category.model");
 const { Product } = require("../models/product.model");
 const router = express.Router();
+const multer = require("multer");
+const mongoose = require("mongoose");
+
+const FILE_TYPE_MAP = {
+  "image/png": "png",
+  "image/jpeg": "jpeg",
+  "image/jpg": "jpg",
+};
+
+const storage = multer.diskStorage({
+  destination: function (req, file, cb) {
+    const isValid = FILE_TYPE_MAP[file.mimetype];
+    let uploadError = new Error("invalid image type");
+
+    if (isValid) {
+      uploadError = null;
+    }
+    cb(uploadError, "public/uploads/");
+  },
+  filename: function (req, file, cb) {
+    const extension = FILE_TYPE_MAP[file.mimetype];
+    const fileName = file.originalname.replace(" ", "-");
+    const uniqueSuffix = Date.now() + "-" + Math.round(Math.random() * 1e9);
+    cb(null, `${fileName}-${uniqueSuffix}.${extension}`);
+  },
+});
+
+const uploadOptions = multer({ storage: storage });

 router.get("/", (req, res) => {
   let filter = {};
@@ -22,15 +50,20 @@ router.get("/:id", (req, res) => {
     .catch((err) => res.status(400).json(err));
 });

-router.post("/", async (req, res) => {
+router.post("/", uploadOptions.single("image"), async (req, res) => {
+  const file = req.file;
+  if (!file) {
+    return res.status(400).json("No file");
+  }
   const category = await Category.findById(req.body.category);
   if (!category) return res.status(400).send("Bad category");
-
+  const fileName = req.file.filename;
+  const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
   const product = new Product({
     name: req.body.name,
     description: req.body.description,
     richDescription: req.body.richDescription,
-    image: req.body.image,
+    image: `${basePath}${fileName}`,
     brand: req.body.brand,
     price: req.body.price,
     category: req.body.category,
```

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

```
@@ -45,17 +78,25 @@ router.post("/", async (req, res) => {
      .catch((err) => res.status(400).json(err));
  });

- router.put("/", async (req, res) => {
+ router.put("/", uploadOptions.single("image"), async (req, res) => {
+   if (!mongoose.isValidObjectId(req.params.id)) {
+     return res.status(400).send("Bad id");
+   }
+
    const category = await Category.findById(req.body.category);
    if (!category) return res.status(400).send("Bad category");

+   const product = await Product.findById(req.params.id);
+   if (!category) return res.status(400).send("Bad product");
+
+   const file = req.file;
+   let imagepath;
+
+   if (file) {
+     const fileName = req.file.filename;
+     const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
+     imagepath = `${basePath}${fileName}`;
+   } else {
+     imagepath = product.image;
+   }
+
    Product.findByIdAndUpdate(
      req.params.id,
      {
        name: req.body.name,
        description: req.body.description,
        richDescription: req.body.richDescription,
-       image: req.body.image,
+       image: imagepath,
        brand: req.body.brand,
        price: req.body.price,
        category: req.body.category,
```

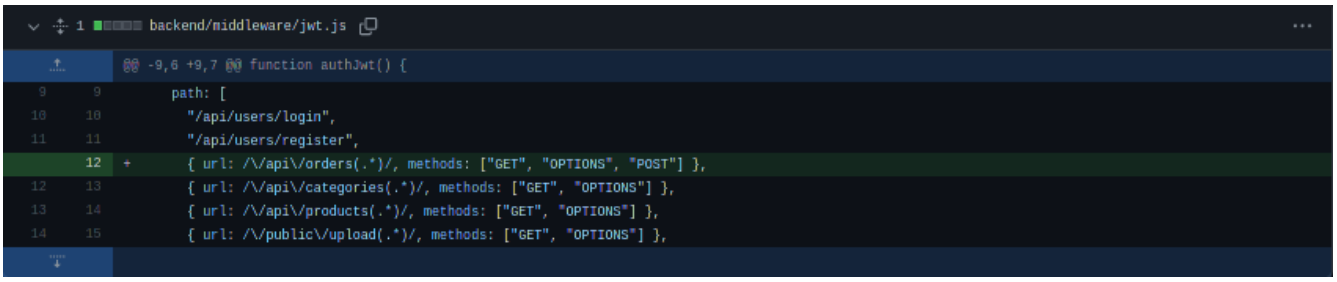
```
@@ -93,4 +144,38 @@ router.get("/get/featured", (req, res) => {
      .catch((err) => res.json(err));
  });

+ router.put(
+   "/gallery-images/:id",
+   uploadOptions.array("images", 10),
+   async (req, res) => {
+     if (!mongoose.isValidObjectId(req.params.id)) {
+       return res.status(400).send("Bad id");
+     }
+     const files = req.files;
+
+     let imagesPaths = [];
+
+     if (files) {
+       const basePath = `${req.protocol}://${req.get("host")}/public/uploads/`;
+       files.map((file) => {
+         imagesPaths.push(`${basePath}${file.filename}`);
+       });
+     }
+
+     const product = await Product.findByIdAndUpdate(
+       req.params.id,
+       {
+         images: imagesPaths,
+       },
+       {
+         new: true,
+       }
+     );
+     if (!product) {
+       return res.status(500).send("Error, not updated");
+     }
+     res.json(product);
+   }
+ );
+
  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.

3.1.11. dodavanje mogućnosti za obične korisinike 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.



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.

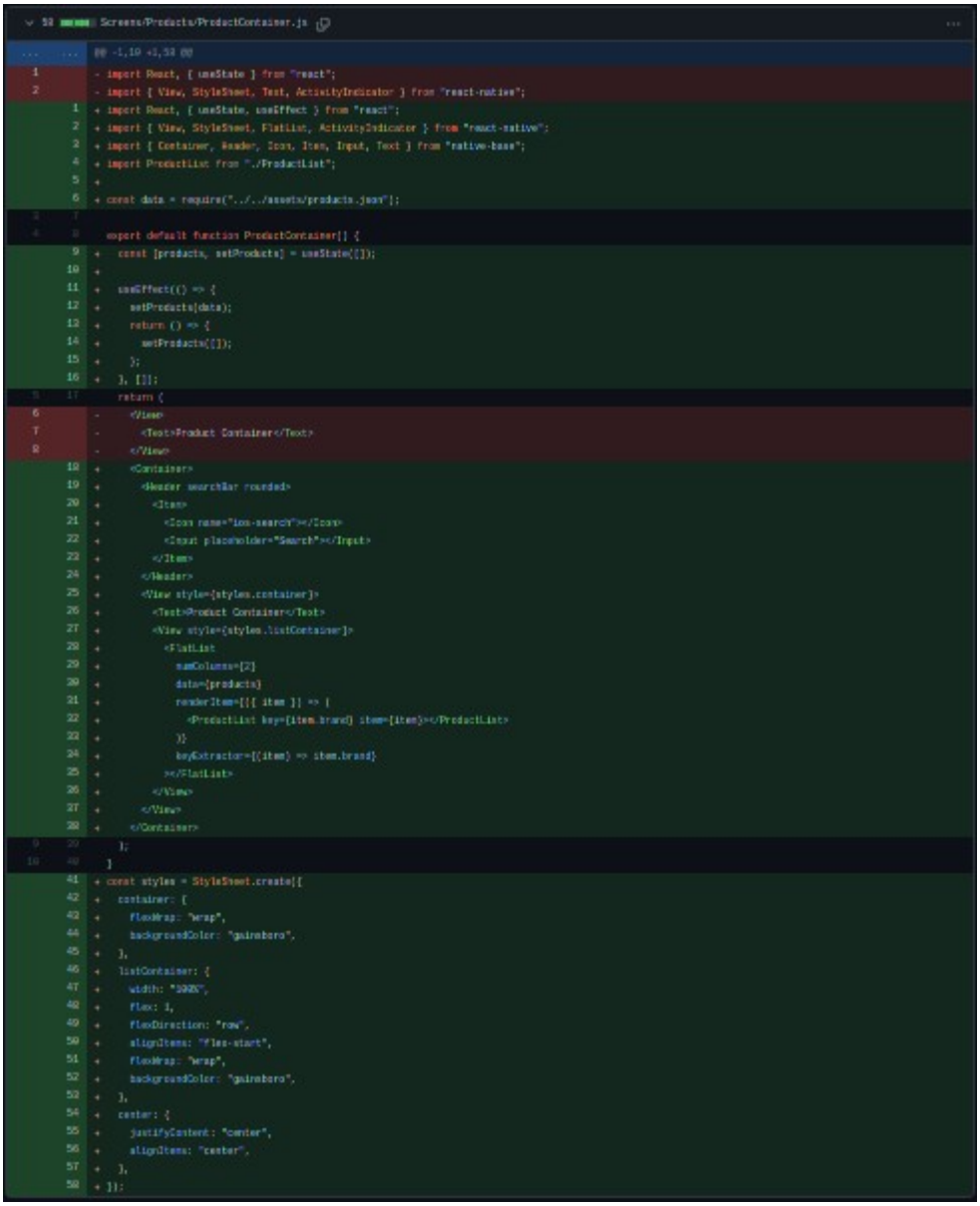
3.2.2. dodavanje productcontainer, list, card i header komponenti - added productcontainer, list, card and header

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

Screens/Products/ProductCard.js

```
@@ -0,0 +1,79 @@
+ import React from "react";
+ import {
+   StyleSheet,
+   View,
+   Dimensions,
+   Image,
+   Text,
+   Button,
+ } from "react-native";
+
+ var { width } = Dimensions.get("window");
+
+ export default function ProductCard(props) {
+   const { name, price, image, countInStock } = props;
+   return (
+     <View style={styles.container}>
+       <Image
+         style={styles.image}
+         resizeMode="contain"
+         source={{
+           uri: image
+             ? image
+             : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23848_960_720.png",
+         }}
+       ></Image>
+       <View style={styles.card}></View>
+       <Text style={styles.title}>
+         {name.length > 15 ? name.substring(0, 15 - 3) + "..." : name}
+       </Text>
+       <Text style={styles.price}>${price}</Text>
+
+       {countInStock > 0 ? (
+         <View style={{ marginBottom: 60 }}>
+           <Button title={"Add"} color={"green"}></Button>
+         </View>
+       ) : (
+         <Text style={{ marginTop: 20 }}>Currently Unavailable</Text>
+       )}
+     </View>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     width: width / 2 - 20,
+     height: width / 1.7,
+     padding: 10,
+     borderRadius: 10,
+     marginTop: 55,
+     marginBottom: 5,
+     marginLeft: 10,
+     alignItems: "center",
+     elevation: 8,
+     backgroundColor: "white",
+   },
+   image: {
+     width: width / 2 - 20 - 10,
+     height: width / 2 - 20 - 30,
+     backgroundColor: "transparent",
+     position: "absolute",
+     top: -45,
+   },
+   card: {
+     marginBottom: 10,
+     height: width / 2 - 20 - 90,
+     backgroundColor: "transparent",
+     width: width / 2 - 20 - 10,
+   },
+   title: {
+     fontWeight: "bold",
+     fontSize: 14,
+     textAlign: "center",
+   },
+   price: {
+     fontSize: 20,
+     color: "orange",
+     marginTop: 10,
+   },
+ });
```



Screens/Products/ProductContainer.js

```
@@ -1,10 +1,53 @@
- import React, { useState } from "react";
- import { View, StyleSheet, Text, ActivityIndicator } from "react-native";
+ import React, { useState, useEffect } from "react";
+ import { View, StyleSheet, FlatList, ActivityIndicator } from "react-native";
+ import { Container, Header, Icon, Item, Input, Text } from "native-base";
+ import ProductList from "./ProductList";
+
+ const data = require("../../assets/products.json");

  export default function ProductContainer() {
+   const [products, setProducts] = useState([]);
+
+   useEffect(() => {
+     setProducts(data);
+     return () => {
+       setProducts([]);
+     };
+   }, []);
    return (
-     <View>
-       <Text>Product Container</Text>
-     </View>
+     <Container>
+       <Header searchBar rounded>
+         <Item>
+           <Icon name="ios-search"></Icon>
+           <Input placeholder="Search"></Input>
+         </Item>
+       </Header>
+       <View style={styles.container}>
+         <Text>Product Container</Text>
+         <View style={styles.listContainer}>
+           <FlatList
+             numColumns={2}
+             data={products}
+             renderItem={({ item }) => (
+               <ProductList key={item.brand} item={item}></ProductList>
+             )}
+             keyExtractor={(item) => item.brand}
+           ></FlatList>
+         </View>
+       </View>
+     </Container>
    );
  }
+ const styles = StyleSheet.create({
+   container: {
+     flexWrap: "wrap",
+     backgroundColor: "gainsboro",
+   },
+   listContainer: {
+     width: "100%",
+     flex: 1,
+     flexDirection: "row",
+     alignItems: "flex-start",
+     flexWrap: "wrap",
+     backgroundColor: "gainsboro",
+   },
+   center: {
+     justifyContent: "center",
+     alignItems: "center",
+   },
+ });
```

Shared/Header.js

```
@@ -0,0 +1,24 @@
+ import React from "react";
+ import { StyleSheet, View, Image, SafeAreaView } from "react-native";
+
+ export default function Header() {
+   return (
+     <SafeAreaView style={styles.header}>
+       <Image
+         source={require("../assets/shop.jpg")}
+         resizeMode="contain"
+         style={{ height: 50 }}
+       ></Image>
+     </SafeAreaView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   header: {
+     width: "100%",
+     flexDirection: "row",
+     alignContent: "center",
+     justifyContent: "center",
+     padding: 20,
+   },
+ });
```

App.js

```
@@ -2,10 +2,12 @@ import { StatusBar } from "expo-status-bar";
  import React from "react";
  import { StyleSheet, Text, View } from "react-native";
  import ProductContainer from "./Screens/Products/ProductContainer";
+ import Header from "./Shared/Header";

  export default function App() {
    return (
      <View style={styles.container}>
+       <Header></Header>
        <ProductContainer></ProductContainer>
      </View>
    );
```

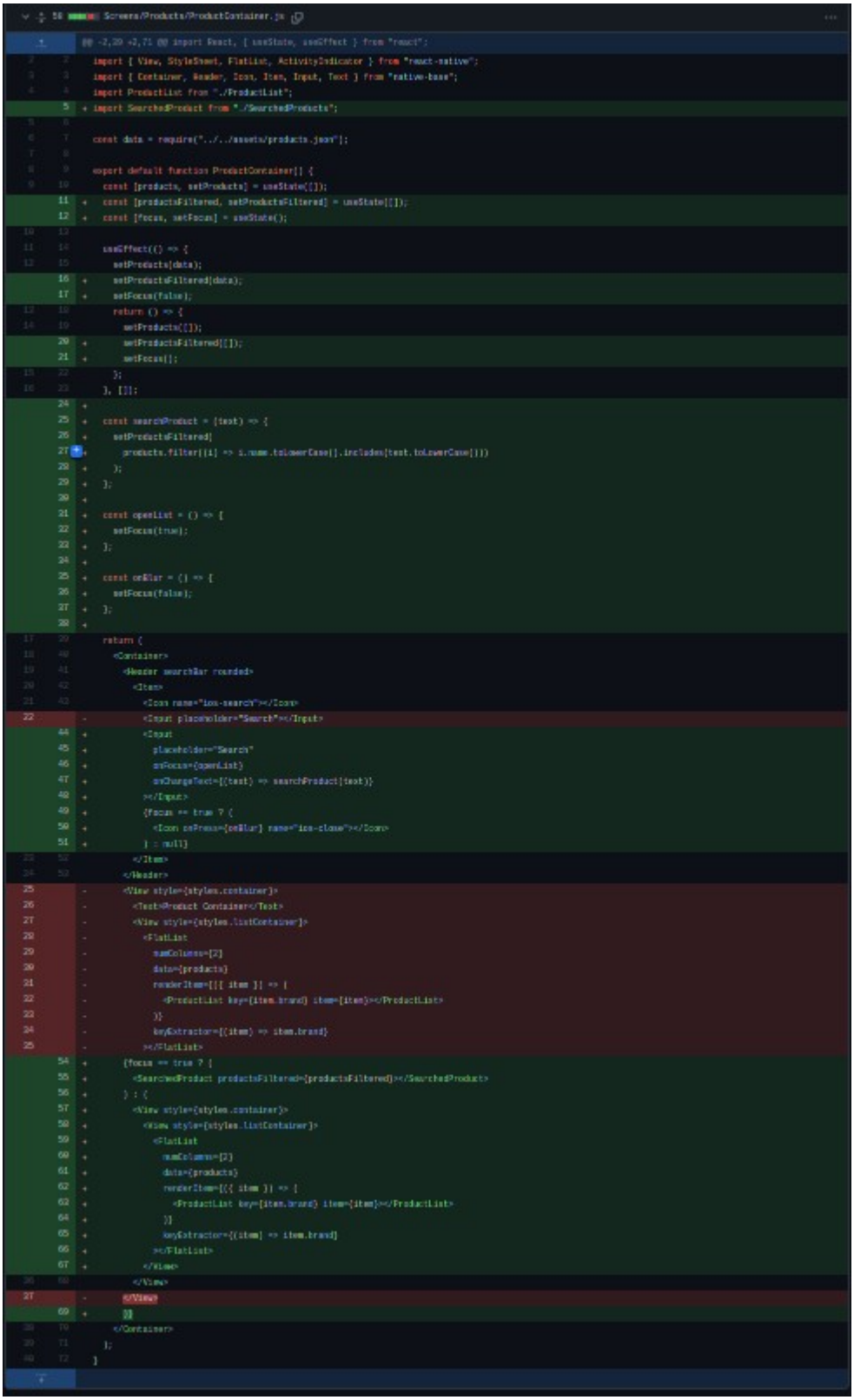
3.2.3. dodavanje polje za pretragu i funkcionalnost - added search field and functionality

Potrebno je da korisnicu mogu da pretraže produkte 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 produkti koji sadrže taj string te se prikazuju na

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

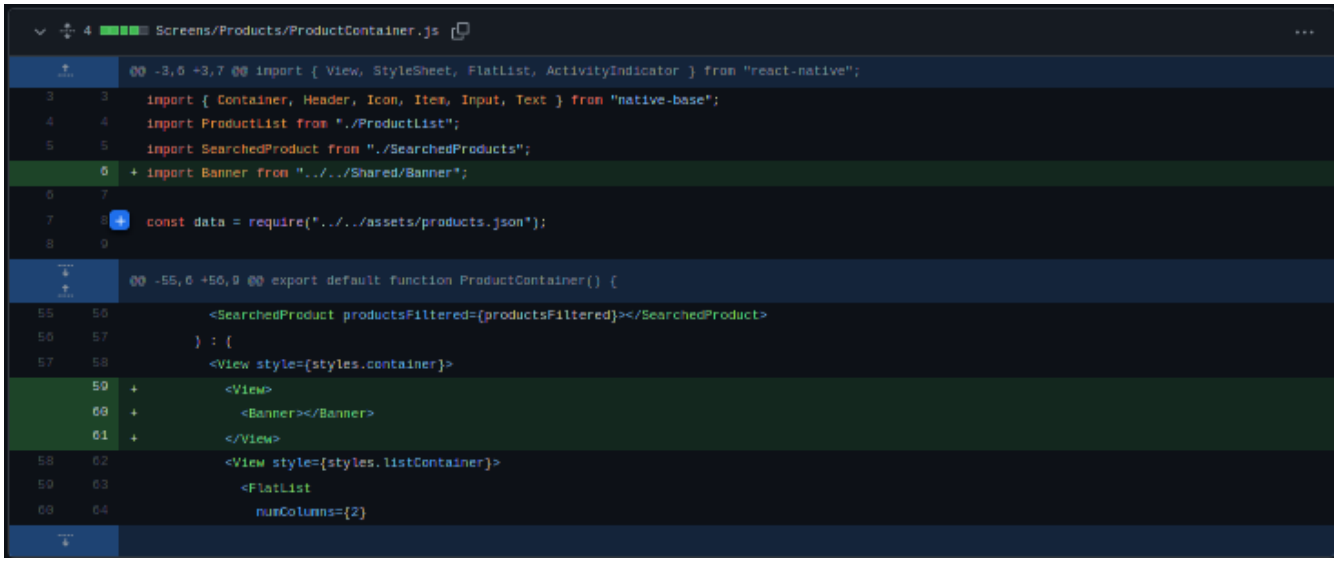
47 Screens/Products/SearchedProducts.js

```
@@ -0,0 +1,47 @@
+ import React from "react";
+ import { View, StyleSheet, Dimensions } from "react-native";
+ import { Content, Left, Body, ListItem, Thumbnail, Text } from "native-base";
+
+ var { width } = Dimensions.get("window");
+
+ export default function SearchedProducts(props) {
+   const { productsFiltered } = props;
+   return (
+     <Content style={{ width: width }}>
+       {productsFiltered.length > 0 ? (
+         productsFiltered.map((item) => {
+           return (
+             <ListItem key={item._id.$oid} avatar>
+               <Left>
+                 <Thumbnail
+                   source={{
+                     uri: item.image
+                       ? item.image
+                       : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23649_960_720.png",
+                   }}
+                 ></Thumbnail>
+               </Left>
+               <Body>
+                 <Text>{item.name}</Text>
+                 <Text note>{item.description}</Text>
+               </Body>
+             </ListItem>
+           );
+         })
+       ) : (
+         <View style={styles.center}>
+           <Text style={{ alignSelf: "center" }}>
+             No products match the selected search
+           </Text>
+         </View>
+       )}
+     </Content>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   center: {
+     justifyContent: "center",
+     alignItems: "center",
+   },
+ });
```



3.2.4. dodavanje bannera 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.



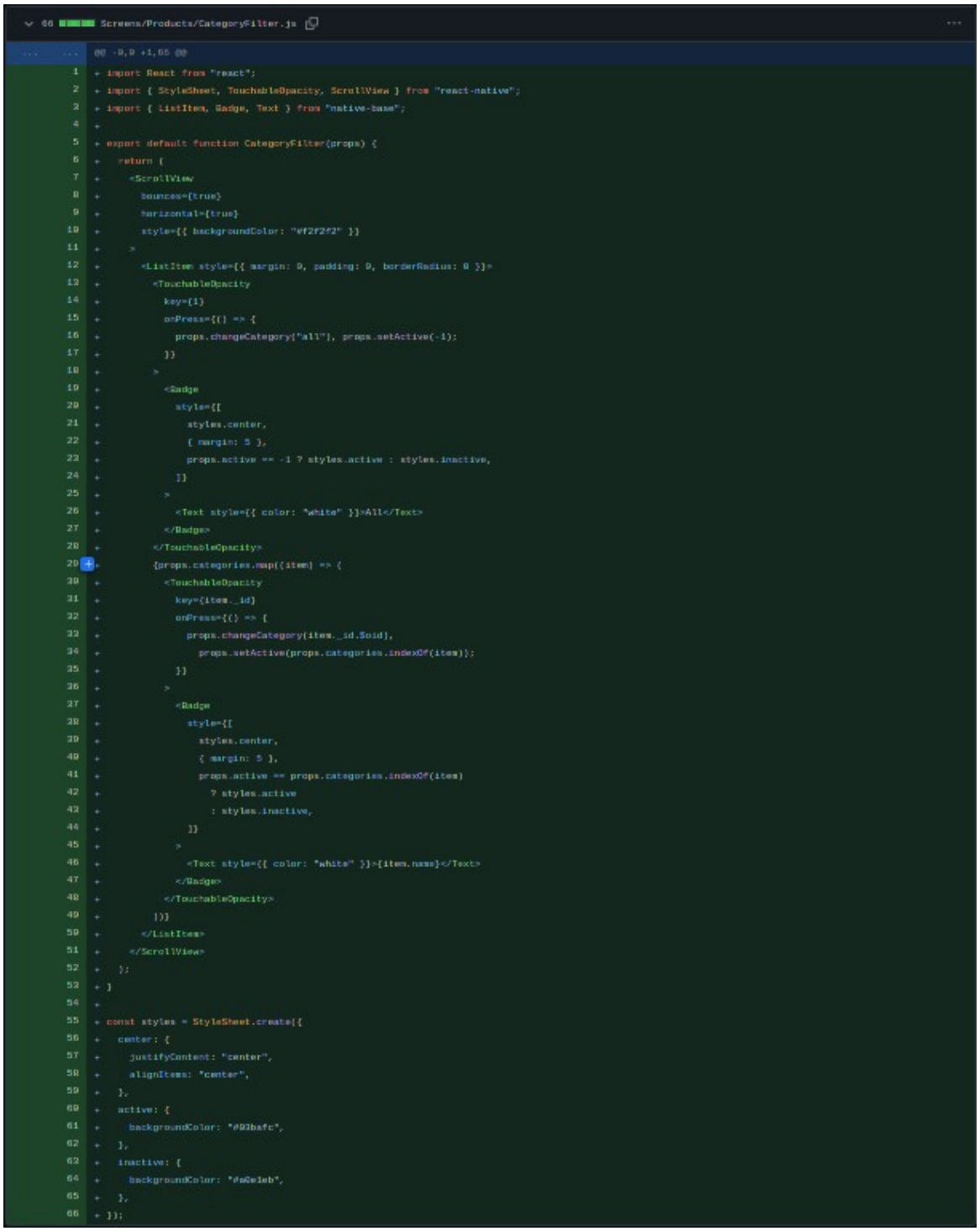
[11] https://www.npmjs.com/package/react-native-swiper

65 Shared/Banner.js

```
@@ -0,0 +1,65 @@
+ import React, { useState, useEffect } from "react";
+ import { Image, StyleSheet, Dimensions, View, ScrollView } from "react-native";
+ import Swiper from "react-native-swiper/src";
+
+ var { width } = Dimensions.get("window");
+
+ export default function Banner() {
+   const [bannerData, setBannerData] = useState([]);
+
+   useEffect(() => {
+     setBannerData([
+       "https://images.vexels.com/media/users/3/128443/preview2/ff9af1e1edfa2c4a40c43b8c2840ce52-macbook-pro-touch-bar-banner.jpg",
+       "https://pbs.twimg.com/media/D7P_yLdX4AAvJWO.jpg",
+       "https://www.yardproduct.com/blog/wp-content/uploads/2016/01/gardening-banner.jpg",
+     ]);
+     return () => {
+       setBannerData([]);
+     };
+   }, []);
+
+   return (
+     <ScrollView>
+       <View style={StyleSheet.container}>
+         <View style={styles.swiper}>
+           <Swiper
+             style={{ height: width / 2 }}
+             showButtons={false}
+             autoplay={true}
+             autoplayTimeout={3}
+           >
+             {bannerData.map((item) => {
+               return (
+                 <Image
+                   key={item}
+                   style={styles.imageBanner}
+                   resizeMode="contain"
+                   source={{ uri: item }}
+                 ></Image>
+               );
+             })}
+           </Swiper>
+           <View style={{ height: 28 }}></View>
+         </View>
+       </View>
+     </ScrollView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     flex: 1,
+     backgroundColor: "gainsboro",
+   },
+   swiper: {
+     width: width,
+     alignItems: "center",
+     marginTop: 16,
+   },
+   imageBanner: {
+     height: width / 2,
+     width: width - 40,
+     borderRadius: 10,
+     marginHorizontal: 20,
+   },
+ });
```

3.2.5. dodavanje kategorija - added categories

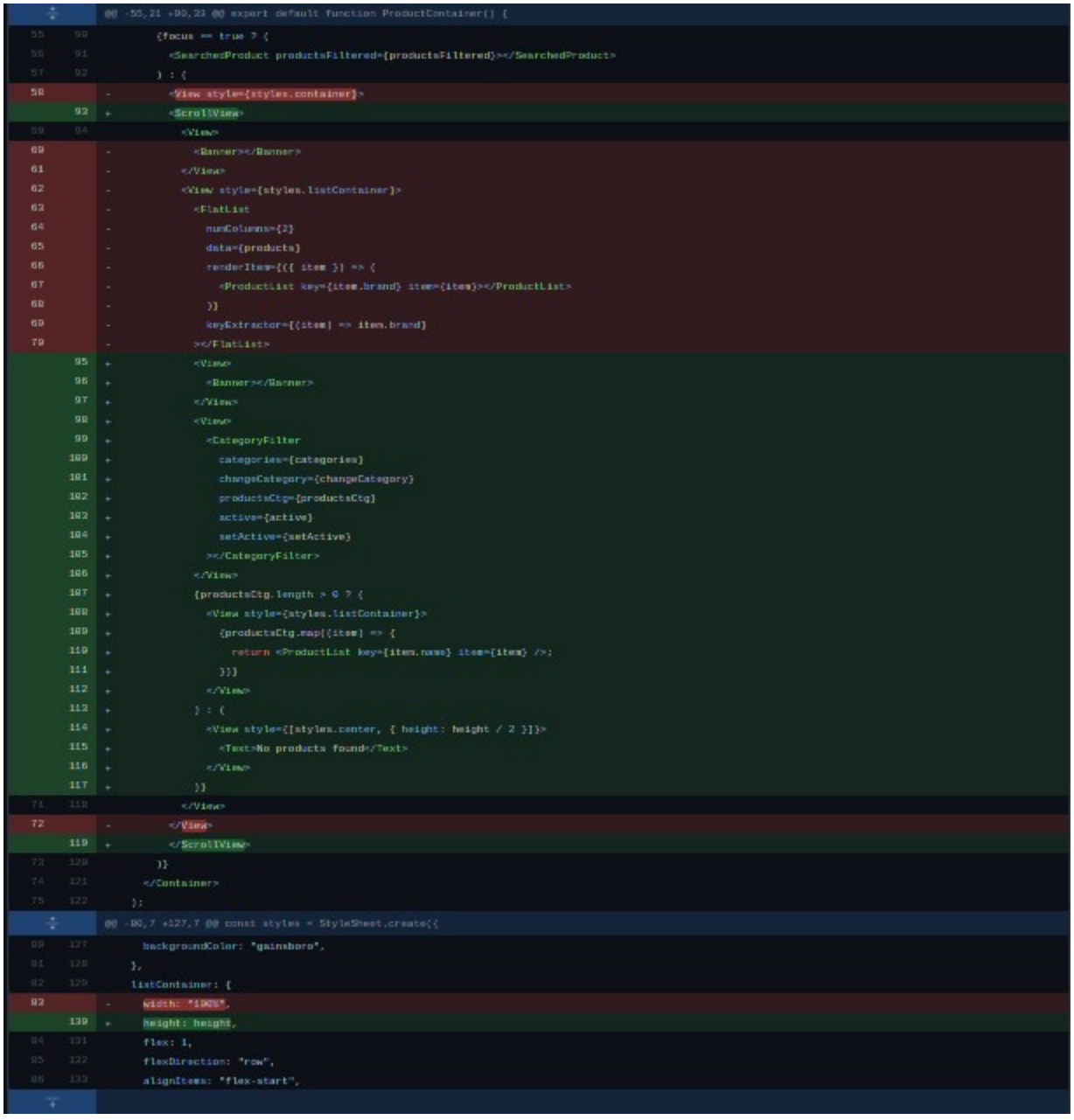
Svaki produkt ima svoju kategoriju, te s tim korisnik može da ih filteruje ukoliko to želi. Potrebno je da se doda selektor za kategoriju iznad svih produkta (CategoryFilter.js) i po korisničkom izboru prikazuje određena kategorija produkta ili svi produkti.



```
66 Screens/Products/CategoryFilter.js
@@ -0,0 +1,65 @@
+ import React from "react";
+ import { StyleSheet, TouchableOpacity, ScrollView } from "react-native";
+ import { ListItem, Badge, Text } from "native-base";
+
+ export default function CategoryFilter(props) {
+   return (
+     <ScrollView
+       bounces={true}
+       horizontal={true}
+       style={{ backgroundColor: "#f2f2f2" }}
+     >
+       <ListItem style={{ margin: 0, padding: 0, borderRadius: 0 }}>
+         <TouchableOpacity
+           key={1}
+           onPress={() => {
+             props.changeCategory("all"), props.setActive(-1);
+           }}
+         >
+           <Badge
+             style={[
+               styles.center,
+               { margin: 5 },
+               props.active == -1 ? styles.active : styles.inactive,
+             ]}
+           >
+             <Text style={{ color: "white" }}>All</Text>
+           </Badge>
+         </TouchableOpacity>
+         {props.categories.map((item) => {
+           <TouchableOpacity
+             key={item._id}
+             onPress={() => {
+               props.changeCategory(item._id.$oid),
+                 props.setActive(props.categories.indexOf(item));
+             }}
+           >
+             <Badge
+               style={[
+                 styles.center,
+                 { margin: 5 },
+                 props.active == props.categories.indexOf(item)
+                   ? styles.active
+                   : styles.inactive,
+               ]}
+             >
+               <Text style={{ color: "white" }}>{item.name}</Text>
+             </Badge>
+           </TouchableOpacity>
+         })}
+       </ListItem>
+     </ScrollView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   center: {
+     justifyContent: "center",
+     alignItems: "center",
+   },
+   active: {
+     backgroundColor: "#03bafc",
+   },
+   inactive: {
+     backgroundColor: "#a0e0eb",
+   },
+ });
```

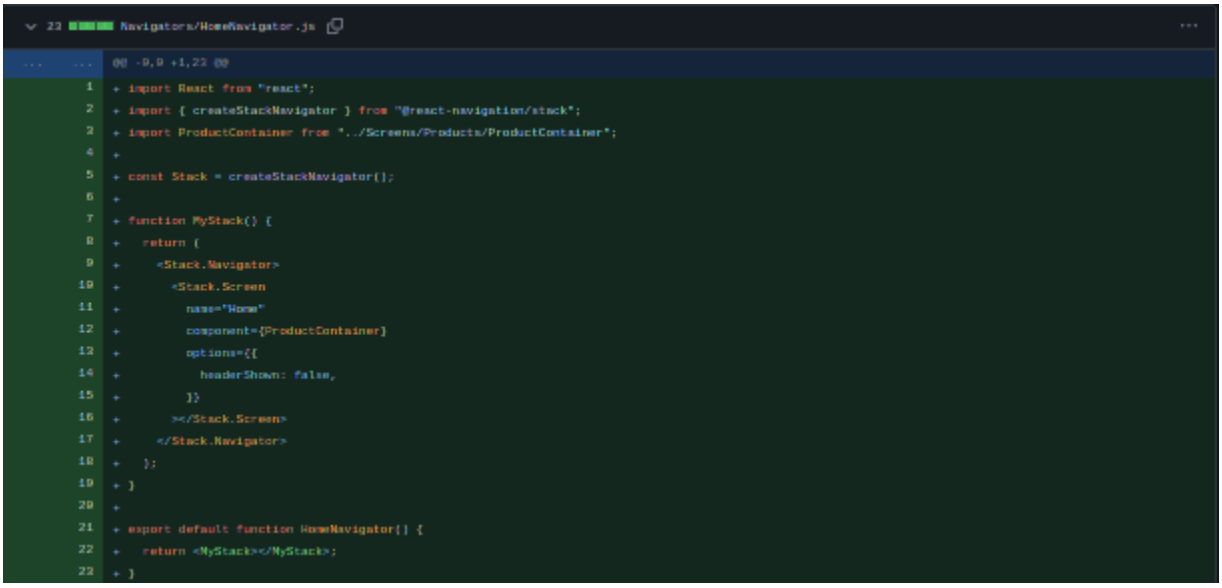
66 Screens/Products/CategoryFilter.js

```
@@ -0,0 +1,65 @@
+ import React from "react";
+ import { StyleSheet, TouchableOpacity, ScrollView } from "react-native";
+ import { ListItem, Badge, Text } from "native-base";
+
+ export default function CategoryFilter(props) {
+   return (
+     <ScrollView
+       bounces={true}
+       horizontal={true}
+       style={{ backgroundColor: "#f2f2f2" }}
+     >
+       <ListItem style={{ margin: 0, padding: 0, borderRadius: 0 }}>
+         <TouchableOpacity
+           key={1}
+           onPress={() => {
+             props.changeCategory["all"], props.setActive(-1);
+           }}
+         >
+           <Badge
+             style={[
+               styles.center,
+               { margin: 5 },
+               props.active == -1 ? styles.active : styles.inactive,
+             ]}
+           >
+             <Text style={{ color: "white" }}>All</Text>
+           </Badge>
+         </TouchableOpacity>
+         {props.categories.map((item) => {
+           <TouchableOpacity
+             key={item._id}
+             onPress={() => {
+               props.changeCategory(item._id.$oid],
+                 props.setActive(props.categories.indexOf(item));
+             }}
+           >
+             <Badge
+               style={[
+                 styles.center,
+                 { margin: 5 },
+                 props.active == props.categories.indexOf(item)
+                   ? styles.active
+                   : styles.inactive,
+               ]}
+             >
+               <Text style={{ color: "white" }}>{item.name}</Text>
+             </Badge>
+           </TouchableOpacity>
+         })}
+       </ListItem>
+     </ScrollView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   center: {
+     justifyContent: "center",
+     alignItems: "center",
+   },
+   active: {
+     backgroundColor: "#03bafc",
+   },
+   inactive: {
+     backgroundColor: "#a0e1eb",
+   },
+ });
```



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.



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

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

02 Navigators/Main.js

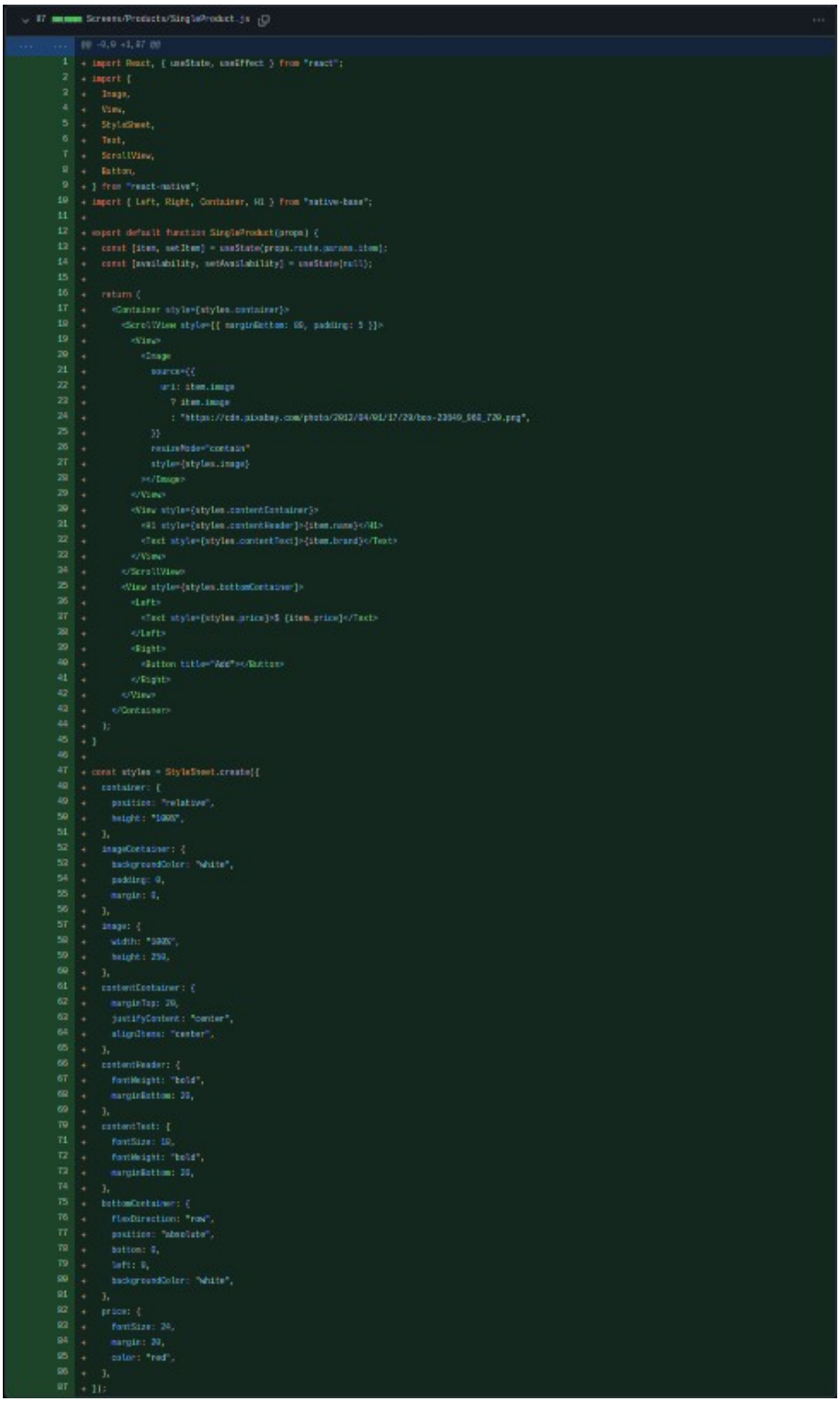
```
@@ -0,0 +1,62 @@
+ import React from "react";
+ import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
+ import { View } from "react-native";
+ import Icon from "react-native-vector-icons/FontAwesome";
+ import HomeNavigator from "./HomeNavigator";
+
+ const Tab = createBottomTabNavigator();
+
+ export default function Main() {
+   return (
+     <Tab.Navigator
+       initialRouteName="Home"
+       tabBarOptions={{
+         keyboardHidesTabBar: true,
+         showLabel: false,
+         activeTintColor: "#e91e63",
+       }}
+     >
+       <Tab.Screen
+         name="Home"
+         component={HomeNavigator}
+         options={{
+           tabBarIcon: ({ color }) => (
+             <Icon
+               name="home"
+               style={{ position: "relative" }}
+               color={color}
+               size={30}
+             ></Icon>
+           ),
+         }}
+       ></Tab.Screen>
+       <Tab.Screen
+         name="Cart"
+         component={HomeNavigator}
+         options={{
+           tabBarIcon: ({ color }) => (
+             <Icon name="shopping-cart" color={color} size={30}></Icon>
+           ),
+         }}
+       ></Tab.Screen>
+       <Tab.Screen
+         name="Admin"
+         component={HomeNavigator}
+         options={{
+           tabBarIcon: ({ color }) => (
+             <Icon name="cog" color={color} size={30}></Icon>
+           ),
+         }}
+       ></Tab.Screen>
+       <Tab.Screen
+         name="User"
+         component={HomeNavigator}
+         options={{
+           tabBarIcon: ({ color }) => (
+             <Icon name="user" color={color} size={30}></Icon>
+           ),
+         }}
+       ></Tab.Screen>
+     </Tab.Navigator>
+   );
+ }
```

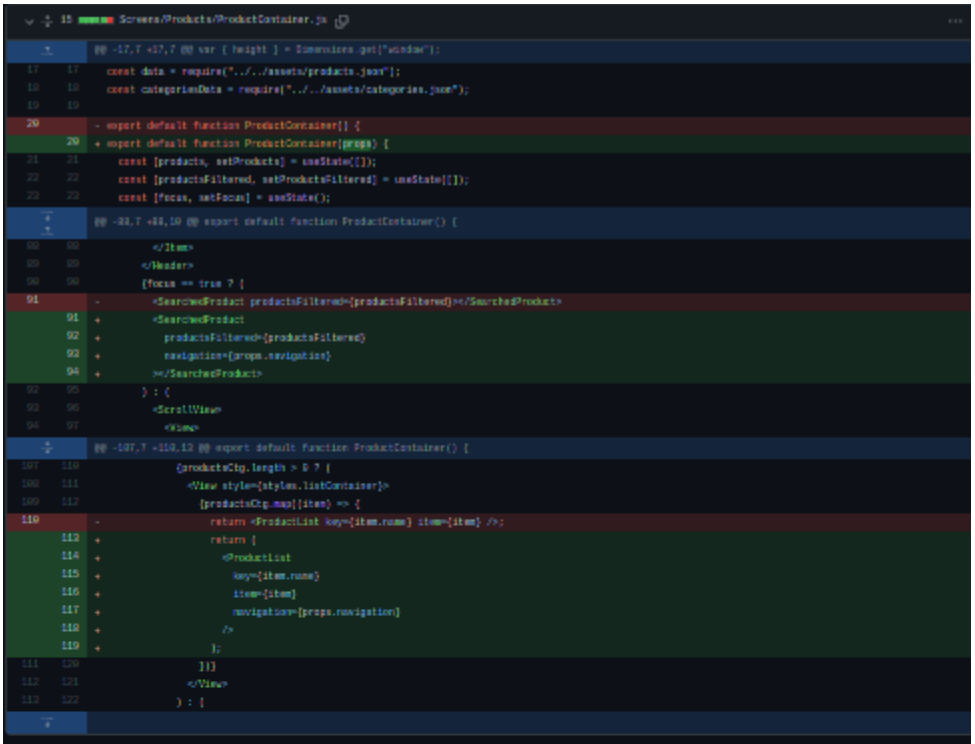
App.js

```
@@ -2,13 +2,15 @@ import React from "react";
  import { StyleSheet, Text, View } from "react-native";
  import ProductContainer from "./Screens/Products/ProductContainer";
  import Header from "./Shared/Header";
+ import { NavigationContainer } from "@react-navigation/native";
+ import Main from "./Navigators/Main";

  export default function App() {
    return (
-     <View style={styles.container}>
+     <NavigationContainer>
        <Header></Header>
-       <ProductContainer></ProductContainer>
-     </View>
+       <Main></Main>
+     </NavigationContainer>
    );
  }
```

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.

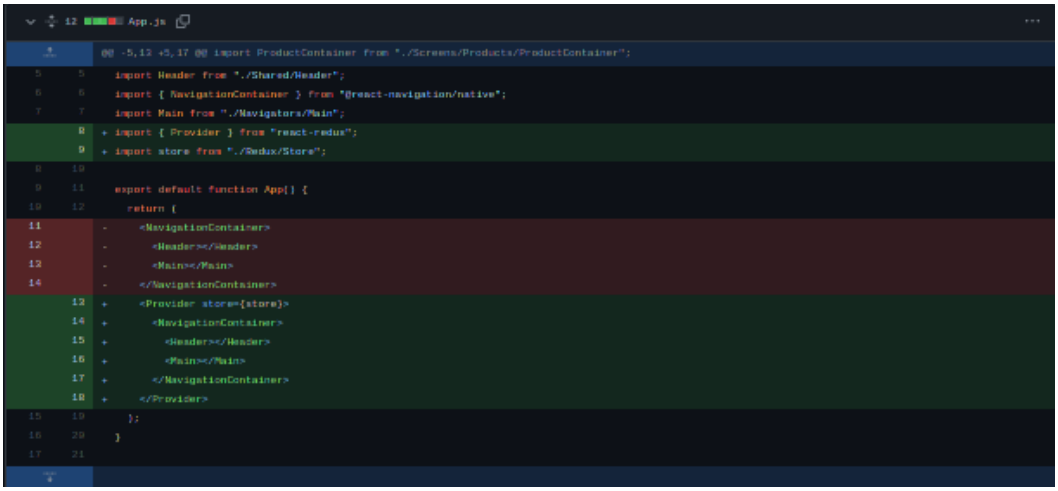




```
8 Navigators/HomeNavigator.js
@@ -1,6 +1,7 @@
 import React from "react";
 import { createStackNavigator } from "@react-navigation/stack";
 import ProductContainer from "../Screens/Products/ProductContainer";
+import SingleProduct from "../Screens/Products/SingleProduct";

 const Stack = createStackNavigator();

@@ -14,6 +15,13 @@ function MyStack() {
           headerShown: false,
         }}
       ></Stack.Screen>
+      <Stack.Screen
+        name="Product Detail"
+        component={SingleProduct}
+        options={{
+          headerShown: false,
+        }}
+      ></Stack.Screen>
     </Stack.Navigator>
   );
 }
```

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 paylaod kojim mijenja state. Nakon kreiranja svega kompltna aplikacija se wrapuje u Provider komponentu iz react-redux biblioteke.



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

21 Redux/Actions/cartActions.js

```
@@ -0,0 +1,21 @@
+ import { ADD_TO_CART, REMOVE_FROM_CART, CLEAR_CART } from "../constants";
+
+ export const addToCart = (payload) => {
+   return {
+     type: ADD_TO_CART,
+     payload,
+   };
+ };
+
+ export const removeFromCart = (payload) => {
+   return {
+     type: REMOVE_FROM_CART,
+     payload,
+   };
+ };
+
+ export const removeFromCart = () => {
+   return {
+     type: CLEAR_CART,
+   };
+ };
```

15 Redux/Reducers/cartItem.js

```
@@ -0,0 +1,15 @@
+ import { ADD_TO_CART, REMOVE_FROM_CART, CLEAR_CART } from "../constants";
+
+ const cartItems = (state = [], action) => {
+   switch (action.type) {
+     case ADD_TO_CART:
+       return [...state, action.payload];
+     case REMOVE_FROM_CART:
+       return state.filter((cartItem) => cartItem !== action.payload);
+     case CLEAR_CART:
+       return (state = []);
+   }
+   return state;
+ };
+
+ export default cartItems;
```

15 Redux/Store.js

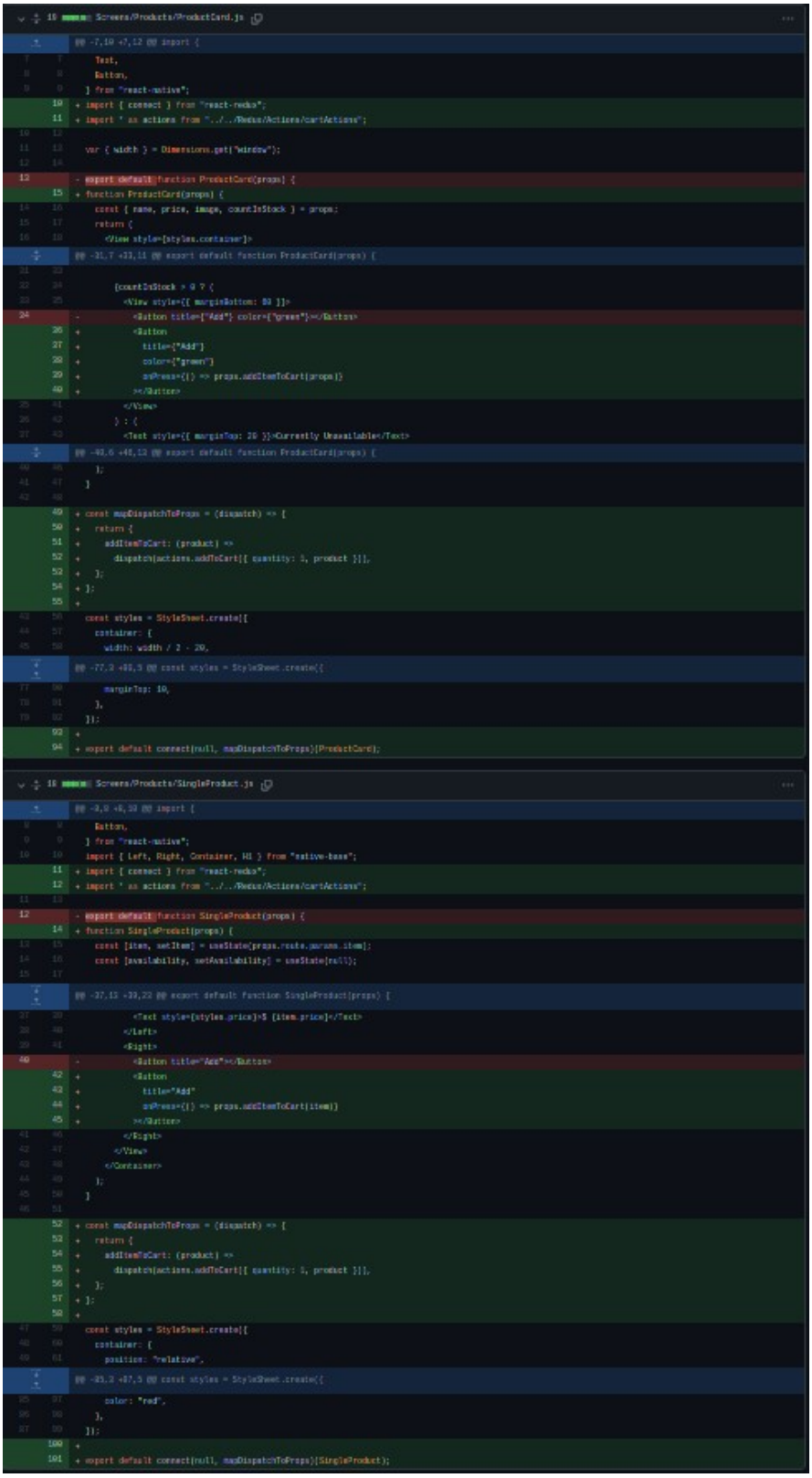
```
@@ -0,0 +1,15 @@
+ import { createStore, combineReducers, applyMiddleware } from "redux";
+ import thunkMiddleware from "redux-thunk";
+ import { composeWithDevTools } from "redux-devtools-extension/developmentOnly";
+ import cartItems from "./Reducers/cartItem";
+
+ const reducers = combineReducers({
+   cartItems: cartItems,
+ });
+
+ const store = createStore(
+   reducers,
+   composeWithDevTools(applyMiddleware(thunkMiddleware))
+ );
+
+ export default store;
```

3 Redux/constants.js

```
@@ -0,0 +1,3 @@
+ export const ADD_TO_CART = "ADD_TO_CART";
+ export const REMOVE_FROM_CART = "REMOVE_FROM_CART";
+ export const CLEAR_CART = "CLEAR_CART";
```

3.2.9. dodavanje funkcionalosti 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.



44 Shared/CartIcon.js

```
@@ -0,0 +1,44 @@
+ import React from "react";
+ import { StyleSheet } from "react-native";
+ import { Badge, Text } from "native-base";
+
+ import { connect } from "react-redux";
+
+ function CartIcon(props) {
+   return (
+     <>
+       {props.cartItems.length ? (
+         <Badge style={styles.badge}>
+           <Text style={styles.text}>{props.cartItems.length}</Text>
+         </Badge>
+       ) : null}
+     </>
+   );
+ }
+
+ const mapStateToProps = (state) => {
+   const { cartItems } = state;
+   return {
+     cartItems: cartItems,
+   };
+ };
+
+ const styles = StyleSheet.create({
+   badge: {
+     width: 25,
+     position: "absolute",
+     flex: 1,
+     justifyContent: "center",
+     alignItems: "center",
+     alignContent: "center",
+     top: -4,
+     right: -15,
+   },
+   text: {
+     fontSize: 12,
+     width: 100,
+     fontWeight: "bold",
+   },
+ });
+
+ export default connect(mapStateToProps)(CartIcon);
```

42 Screens/Cart/CartItem.js

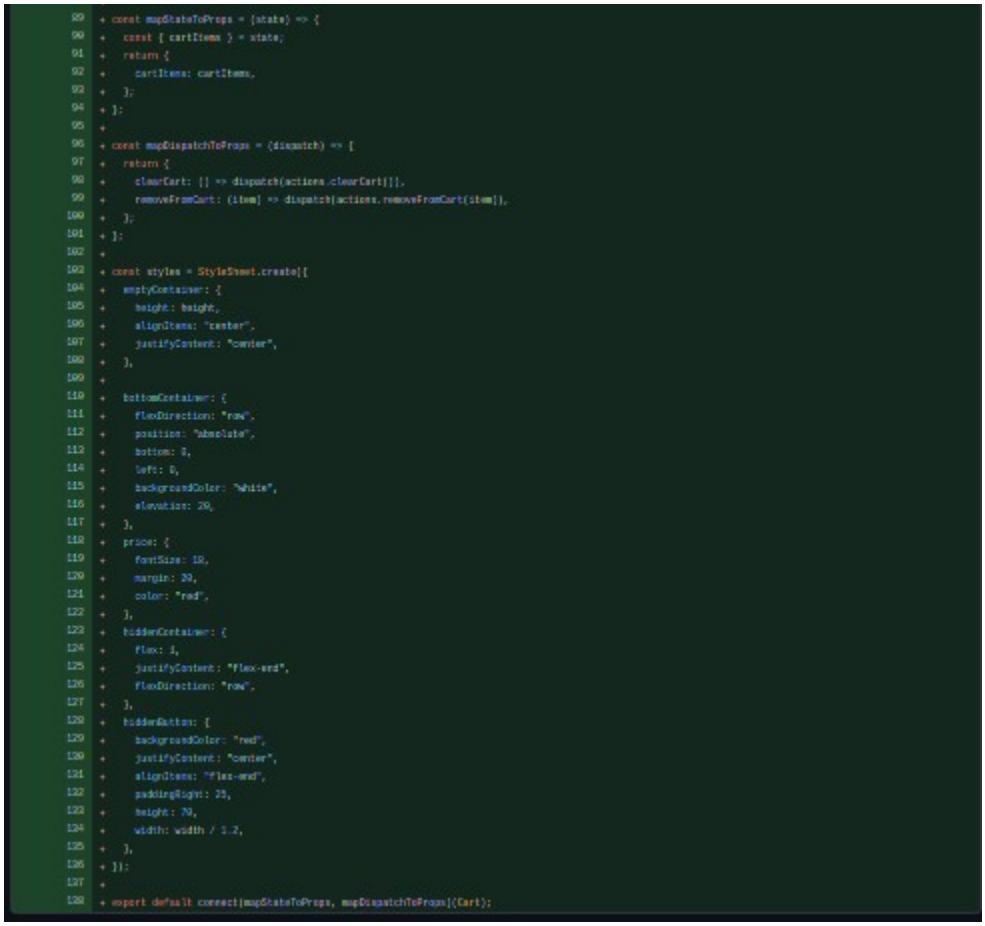
```
@@ -0,0 +1,42 @@
+ import React, { useState } from "react";
+ import { StyleSheet } from "react-native";
+ import { Text, Right, Left, ListItem, Thumbnail, Body } from "native-base";
+
+ export default function CartItem(props) {
+   const item = props.item.item.product;
+   const [quantity, setQuantity] = useState(props.item.quantity);
+   return (
+     <ListItem style={styles.listItem} key={Math.random()} avatar>
+       <Left>
+         <Thumbnail
+           source={{
+             uri: item.image
+               ? item.image
+               : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23649_960_720.png",
+           }}
+         ></Thumbnail>
+       </Left>
+       <Body style={styles.body}>
+         <Left>
+           <Text>{item.name}</Text>
+         </Left>
+         <Right>
+           <Text>$ {item.price}</Text>
+         </Right>
+       </Body>
+     </ListItem>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   listItem: {
+     alignItems: "center",
+     backgroundColor: "White",
+     justifyContent: "center",
+   },
+   body: {
+     margin: 10,
+     alignItems: "center",
+     flexDirection: "row",
+   },
+ });
```

6 Screens/Cart/Checkout.js

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

133 [redacted] Screens/Cart/Cart.js

```
@@ -0,0 +1,133 @@
+ import React from "react";
+ import {
+   View,
+   Dimensions,
+   StyleSheet,
+   Button,
+   TouchableOpacity,
+ } from "react-native";
+ import {
+   Container,
+   Text,
+   Left,
+   Right,
+   H1,
+   ListItem,
+   Thumbnail,
+   Body,
+ } from "native-base";
+ import CartItem from "./CartItem";
+ import { SwipeListView } from "react-native-swipe-list-view";
+
+ import Icon from "react-native-vector-icons/FontAwesome";
+
+ import { connect } from "react-redux";
+ import * as actions from "../../Redux/Actions/cartActions";
+
+ var { height, width } = Dimensions.get("window");
+
+ function Cart(props) {
+   var total = 0;
+   props.cartItems.forEach((cart) => {
+     return (total += cart.product.price);
+   });
+
+   return (
+     <>
+       {props.cartItems.length ? (
+         <Container>
+           <H1 style={{ alignSelf: "center" }}>Cart</H1>
+           <SwipeListView
+             data={props.cartItems}
+             renderItem={(data) => <CartItem item={data}></CartItem>}
+             renderHiddenItem={(data) => (
+               <View
+                 style={styles.hiddenContainer}
+                 onPress={() => props.removeFromCart(data.item)}
+               >
+                 <TouchableOpacity
+                   style={styles.hiddenButton}
+                   onPress={() => props.removeFromCart(data.item)}
+                 >
+                   <Icon name="trash" color={"white"} size={30}></Icon>
+                 </TouchableOpacity>
+               </View>
+             )}
+             disableRightSwipe={true}
+             previewOpenDelay={3000}
+             friction={1000}
+             tension={40}
+             leftOpenValue={75}
+             stopLeftSwipe={75}
+             rightOpenValue={-75}
+           ></SwipeListView>
+           <View style={styles.bottomContainer}>
+             <Left>
+               <Text style={styles.price}>$ {total}</Text>
+             </Left>
+             <Right>
+               <Button title="Clear" onPress={() => props.clearCart()}></Button>
+             </Right>
+             <Right>
+               <Button
+                 title="Checkout"
+                 onPress={() => props.navigation.navigate("Checkout")}
+               ></Button>
+             </Right>
+           </View>
+         </Container>
+       ) : (
+         <Container style={styles.emptyContainer}>
+           <Text>Looks like your cart is empty</Text>
+           <Text>Add products to your cart to get started</Text>
+         </Container>
+       )}
+     </>
+   );
+ }
+
```



```
+ const mapStateToProps = (state) => {
+   const { cartItems } = state;
+   return {
+     cartItems: cartItems,
+   };
+ };
+
+ const mapDispatchToProps = (dispatch) => {
+   return {
+     clearCart: () => dispatch(actions.clearCart()),
+     removeFromCart: (item) => dispatch(actions.removeFromCart(item)),
+   };
+ };
+
+ const styles = StyleSheet.create({
+   emptyContainer: {
+     height: height,
+     alignItems: "center",
+     justifyContent: "center",
+   },
+
+   bottomContainer: {
+     flexDirection: "row",
+     position: "absolute",
+     bottom: 0,
+     left: 0,
+     backgroundColor: "white",
+     elevation: 20,
+   },
+   price: {
+     fontSize: 18,
+     margin: 20,
+     color: "red",
+   },
+   hiddenContainer: {
+     flex: 1,
+     justifyContent: "flex-end",
+     flexDirection: "row",
+   },
+   hiddenButton: {
+     backgroundColor: "red",
+     justifyContent: "center",
+     alignItems: "flex-end",
+     paddingRight: 25,
+     height: 70,
+     width: width / 1.2,
+   },
+ });
+
+ export default connect(mapStateToProps, mapDispatchToProps)(Cart);
```

31 Navigators/CartNavigator.js

```
@@ -0,0 +1,31 @@
+ import React from "react";
+ import { createStackNavigator } from "@react-navigation/stack";
+ import Cart from "../Screens/Cart/Cart";
+ import Checkout from "../Screens/Cart/Checkout";
+
+ const Stack = createStackNavigator();
+
+ function MyStack() {
+   return (
+     <Stack.Navigator>
+       <Stack.Screen
+         name="Cart"
+         component={Cart}
+         options={{
+           headerShown: false,
+         }}
+       ></Stack.Screen>
+       <Stack.Screen
+         name="Checkout"
+         component={Checkout}
+         options={{
+           title: "Checkout",
+         }}
+       ></Stack.Screen>
+     </Stack.Navigator>
+   );
+ }
+
+ export default function CartNavigator() {
+   return <MyStack></MyStack>;
+ }
```

9 Navigators/Main.js

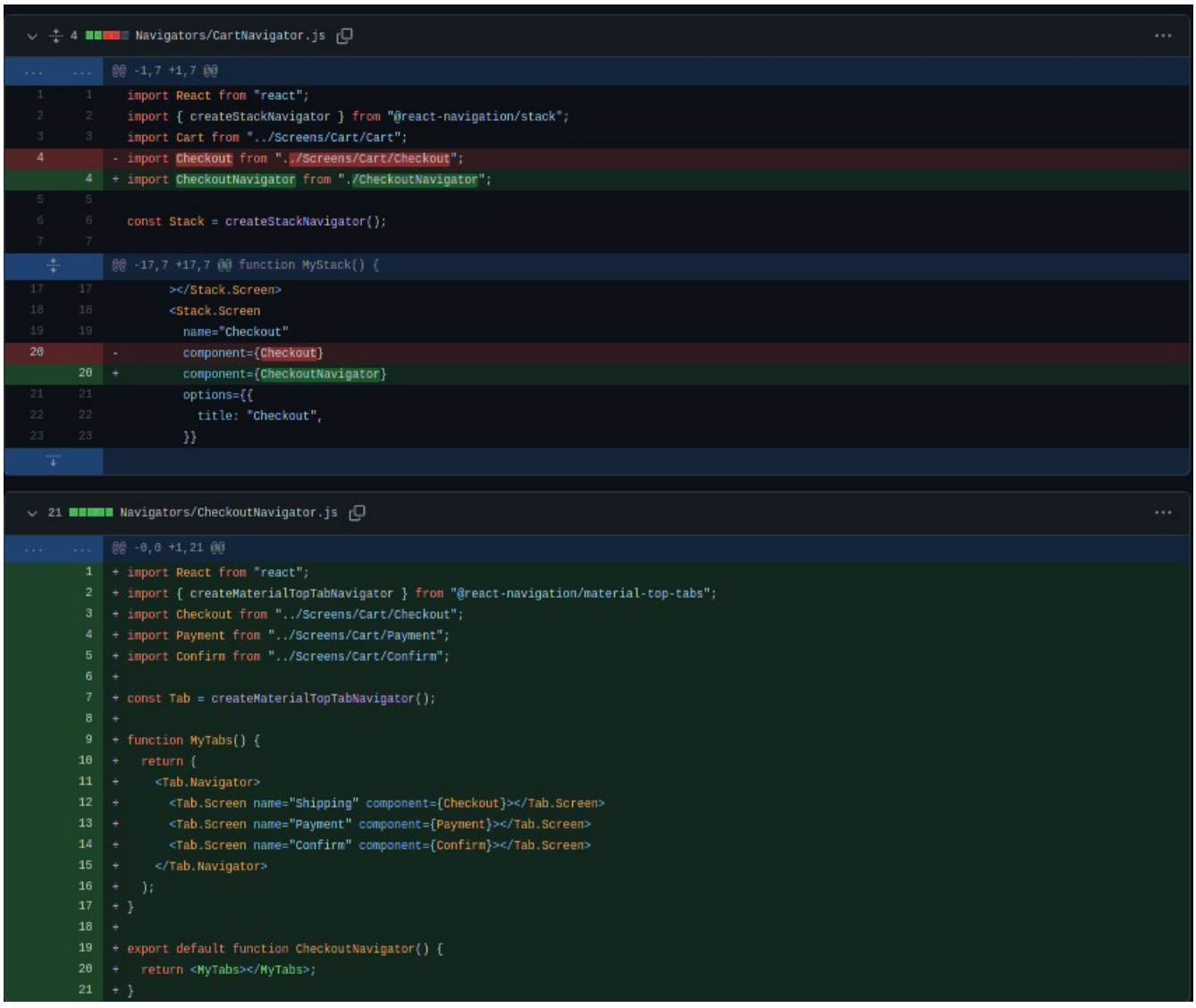
```
@@ -2,6 +2,8 @@ import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
 import { View } from "react-native";
 import Icon from "react-native-vector-icons/FontAwesome";
 import HomeNavigator from "./HomeNavigator";
+ import CartNavigator from "./CartNavigator";
+ import CartIcon from "../Shared/CartIcon";

 const Tab = createBottomTabNavigator();

@@ -22,10 +24,13 @@ export default function Main() {
         ></Tab.Screen>
         <Tab.Screen
           name="Cart"
-          component={HomeNavigator}
+          component={CartNavigator}
           options={{
             tabBarIcon: ({ color }) => (
-              <Icon name="shopping-cart" color={color} size={30}></Icon>
+              <View>
+                <Icon name="shopping-cart" color={color} size={30}></Icon>
+                <CartIcon></CartIcon>
+              </View>
             ),
           }}
         ></Tab.Screen>
```

2 Redux/Actions/cartActions.js

```
@@ -14,7 +14,7 @@ export const removeFromCart = (payload) => {
   };
 };

- export const removeFromCart = () => {
+ export const clearCart = () => {
   return {
     type: CLEAR_CART,
   };
```

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 produkte. Za to sve se dodaju navigacija i komponente.

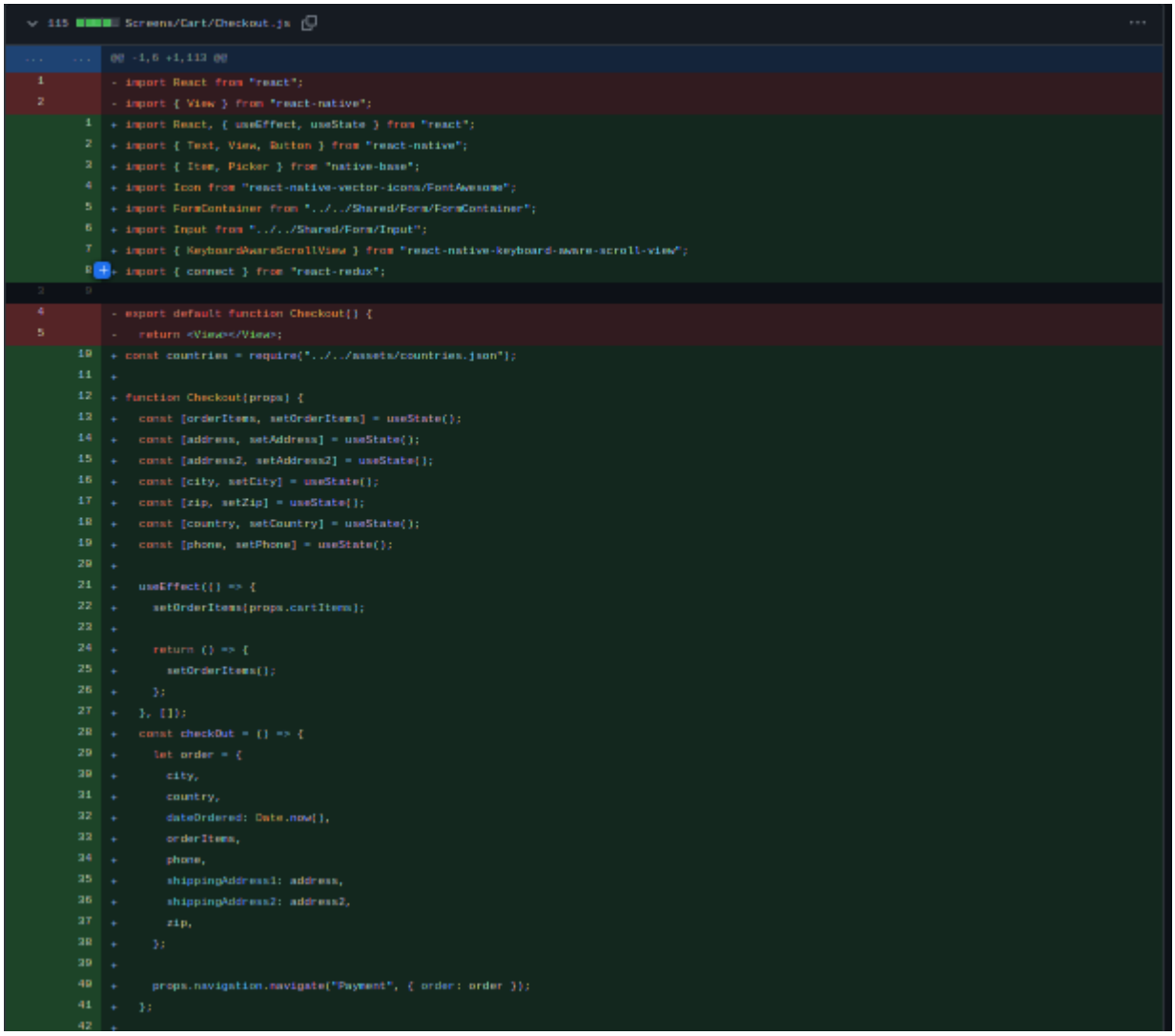


Navigators/CartNavigator.js

```
@@ -1,7 +1,7 @@
 import React from "react";
 import { createStackNavigator } from "@react-navigation/stack";
 import Cart from "../Screens/Cart/Cart";
-import Checkout from "../Screens/Cart/Checkout";
+import CheckoutNavigator from "./CheckoutNavigator";

 const Stack = createStackNavigator();

@@ -17,7 +17,7 @@ function MyStack() {
       ></Stack.Screen>
       <Stack.Screen
         name="Checkout"
-        component={Checkout}
+        component={CheckoutNavigator}
         options={{
           title: "Checkout",
         }}
```

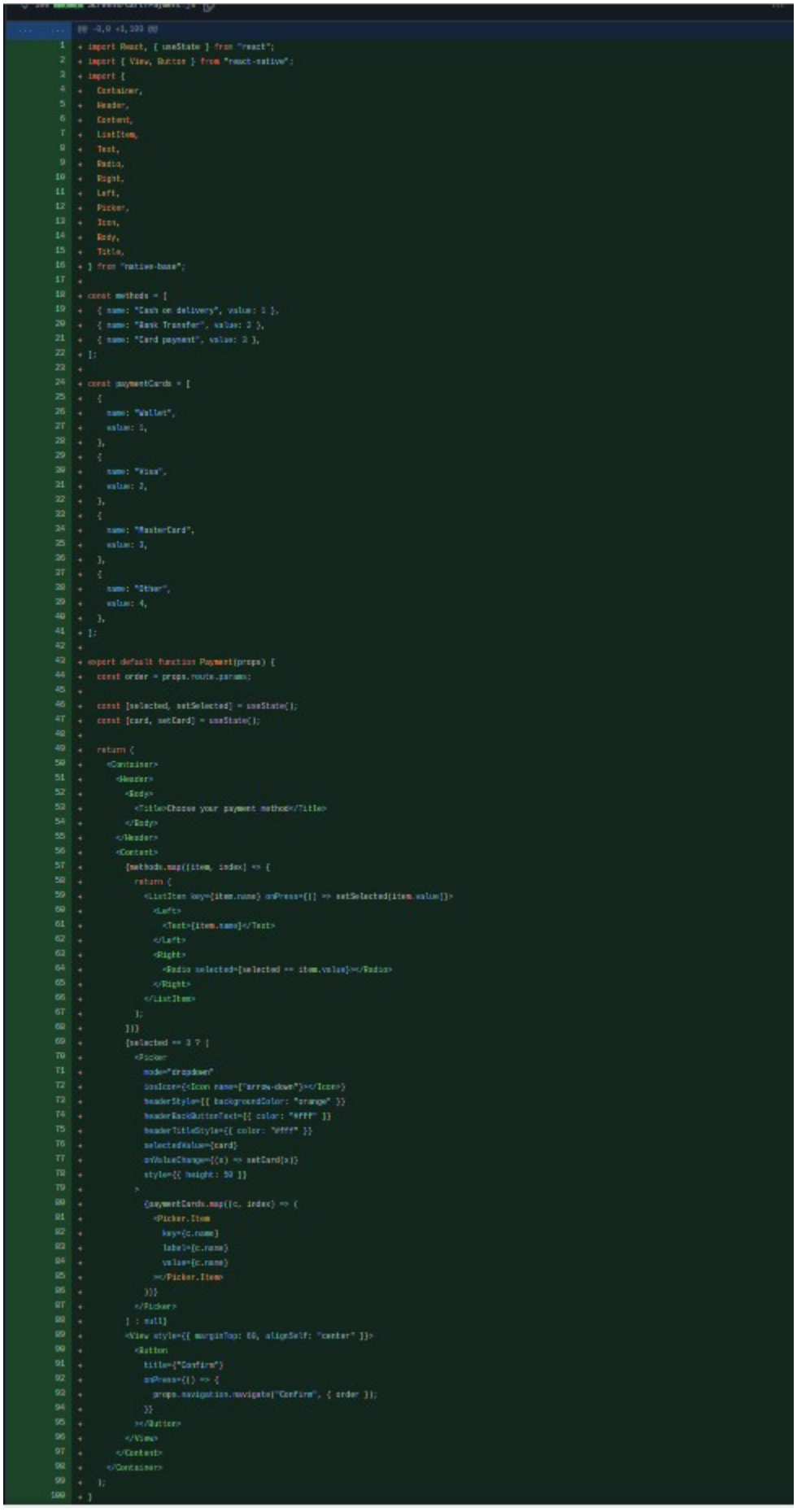
Navigators/CheckoutNavigator.js

```
@@ -0,0 +1,21 @@
+import React from "react";
+import { createMaterialTopTabNavigator } from "@react-navigation/material-top-tabs";
+import Checkout from "../Screens/Cart/Checkout";
+import Payment from "../Screens/Cart/Payment";
+import Confirm from "../Screens/Cart/Confirm";
+
+const Tab = createMaterialTopTabNavigator();
+
+function MyTabs() {
+  return (
+    <Tab.Navigator>
+      <Tab.Screen name="Shipping" component={Checkout}></Tab.Screen>
+      <Tab.Screen name="Payment" component={Payment}></Tab.Screen>
+      <Tab.Screen name="Confirm" component={Confirm}></Tab.Screen>
+    </Tab.Navigator>
+  );
+}
+
+export default function CheckoutNavigator() {
+  return <MyTabs></MyTabs>;
+}
```


115 Screens/Cart/Checkout.js
@@ -1,6 +1,113 @@
- import React from "react";
- import { View } from "react-native";
+ import React, { useEffect, useState } from "react";
+ import { Text, View, Button } from "react-native";
+ import { Item, Picker } from "native-base";
+ import Icon from "react-native-vector-icons/FontAwesome";
+ import FormContainer from "../../Shared/Form/FormContainer";
+ import Input from "../../Shared/Form/Input";
+ import { KeyboardAwareScrollView } from "react-native-keyboard-aware-scroll-view";
+ import { connect } from "react-redux";
- export default function Checkout() {
-   return <View></View>;
+ const countries = require("../../assets/countries.json");
+
+ function Checkout(props) {
+   const [orderItems, setOrderItems] = useState();
+   const [address, setAddress] = useState();
+   const [address2, setAddress2] = useState();
+   const [city, setCity] = useState();
+   const [zip, setZip] = useState();
+   const [country, setCountry] = useState();
+   const [phone, setPhone] = useState();
+
+   useEffect(() => {
+   setOrderItems(props.cartItems);
+
+   return () => {
+   setOrderItems();
+   };
+   }, []);
+   const checkOut = () => {
+   let order = {
+   city,
+   country,
+   dateOrdered: Date.now(),
+   orderItems,
+   phone,
+   shippingAddress1: address,
+   shippingAddress2: address2,
+   zip,
+   };
+
+   props.navigation.navigate("Payment", { order: order });
+   };
+

```
+
+  return (
+    <KeyboardAwareScrollView
+      viewIsInsideTabBar={true}
+      extraHeight={200}
+      enableOnAndroid={true}
+    >
+      <FormContainer title={"Shipping Address"}>
+        <Input
+          placeholder={"Phone"}
+          name={"phone"}
+          value={phone}
+          keyboardType={"numeric"}
+          onChangeText={(text) => setPhone(text)}
+        />
+        <Input
+          placeholder={"Shipping Address 1"}
+          name={"ShippingAddress1"}
+          value={address}
+          onChangeText={(text) => setAddress(text)}
+        />
+        <Input
+          placeholder={"Shipping Address 2"}
+          name={"ShippingAddress2"}
+          value={address2}
+          onChangeText={(text) => setAddress2(text)}
+        />
+        <Input
+          placeholder={"City"}
+          name={"city"}
+          value={city}
+          onChangeText={(text) => setCity(text)}
+        />
+        <Input
+          placeholder={"Zip Code"}
+          name={"zip"}
+          value={zip}
+          keyboardType={"numeric"}
+          onChangeText={(text) => setZip(text)}
+        />
+        <Item picker>
+          <Picker
+            mode="dropdown"
+            iosIcon={<Icon name="arrow-down" color={"#007aff"} />}
+            style={{ width: undefined, height: 50 }}
+            selectedValue={country}
+            placeholder="Select your country"
+            placeholderStyle={{ color: "#007aff" }}
+            placeholderIconColor="#007aff"
+            onValueChange={(e) => setCountry(e)}
+          >
+            {countries.map((c) => {
+              return <Picker.Item key={c.code} label={c.name} value={c.name} />;
+            })}
+          </Picker>
+        </Item>
+        <View style={{ marginTop: 20, width: "80%", alignItems: "center" }}>
+          <Button title="Confirm" onPress={() => checkOut()} />
+        </View>
+      </FormContainer>
+    </KeyboardAwareScrollView>
+  );
}
+
+ const mapStateToProps = (state) => {
+   const { cartItems } = state;
+   return {
+     cartItems: cartItems,
+   };
+ };
+
+ export default connect(mapStateToProps)(Checkout);
```

```
@@ -0,0 +1,193 @@
+ import React from "react";
+ import { View, StyleSheet, Dimensions, ScrollView, Button } from "react-native";
+ import { Text, Left, Right, ListItem, Thumbnail, Body } from "native-base";
+ import { connect } from "react-redux";
+ import * as actions from "../../Redux/Actions/cartActions";
+
+ var { width, height } = Dimensions.get("window");
+
+ function Confirm(props) {
+   const confirmOrder = () => {
+     setTimeout(() => {
+       props.clearCart();
+       props.navigation.navigate("Cart");
+     });
+   };
+
+   const confirm = props.route.params;
+   return (
+     <ScrollView contentContainerStyle={styles.container}>
+       <View style={styles.titleContainer}>
+         <Text style={{ fontSize: 20, fontWeight: "bold" }}>Confirm Order</Text>
+       </View>
+       {props.route.params ? (
+         <View
+           style={{
+             borderWidth: 1,
+             borderColor: "orange",
+           }}
+         >
+           <Text style={styles.title}>Shipping to:</Text>
+           <View style={{ padding: 8 }}>
+             <Text>Address: {confirm.order.order.shippingAddress1}</Text>
+             <Text>Address 2: {confirm.order.order.shippingAddress2}</Text>
+             <Text>City: {confirm.order.order.city}</Text>
+             <Text>Zip Code: {confirm.order.order.zip}</Text>
+             <Text>Country: {confirm.order.order.country}</Text>
+           </View>
+           <Text style={styles.title}>Items:</Text>
+           {confirm.order.order.orderItems.map((i) => {
+             return (
+               <ListItem style={styles.listItem} key={i.product.name} avatar>
+                 <Left>
+                   <Thumbnail source={{ uri: i.product.image }}></Thumbnail>
+                 </Left>
+                 <Body style={styles.body}>
+                   <Left>
+                     <Text>{i.product.name}</Text>
+                   </Left>
+                   <Right>
+                     <Text>$ {i.product.price}</Text>
+                   </Right>
+                 </Body>
+               </ListItem>
+             );
+           })}
+         </View>
+       ) : null}
+       <View style={{ alignItems: "center", margin: 20 }}>
+         <Button title={"Place order"} onPress={confirmOrder}></Button>
+       </View>
+     </ScrollView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     height: height,
+     padding: 8,
+     alignContent: "center",
+     backgroundColor: "white",
+   },
+   titleContainer: {
+     justifyContent: "center",
+     alignItems: "center",
+     margin: 8,
+   },
+   title: {
+     alignSelf: "center",
+     margin: 8,
+     fontSize: 16,
+     fontWeight: "bold",
+   },
+   listItem: {
+     alignItems: "center",
+     backgroundColor: "white",
+     justifyContent: "center",
+     width: width / 1.2,
+   },
+   body: {
+     margin: 10,
+     alignItems: "center",
+     flexDirection: "row",
+   },
+ });
+
+ const mapDispatchToProps = (dispatch) => {
+   return { clearCart: () => dispatch(actions.clearCart()) };
+ };
+
+ export default connect(null, mapDispatchToProps)(Confirm);
```



2 Screens/Products/ProductContainer.js

```
@@ -94,7 +94,7 @@ export default function ProductContainer(props) {
          ></SearchedProduct>
        ) : (
          <ScrollView>
-           <View>
+           <View style={{ paddingBottom: 100, backgroundColor: "gainsboro" }}>
              <View>
                <Banner></Banner>
              </View>
```

20 Shared/Form/FormContainer.js

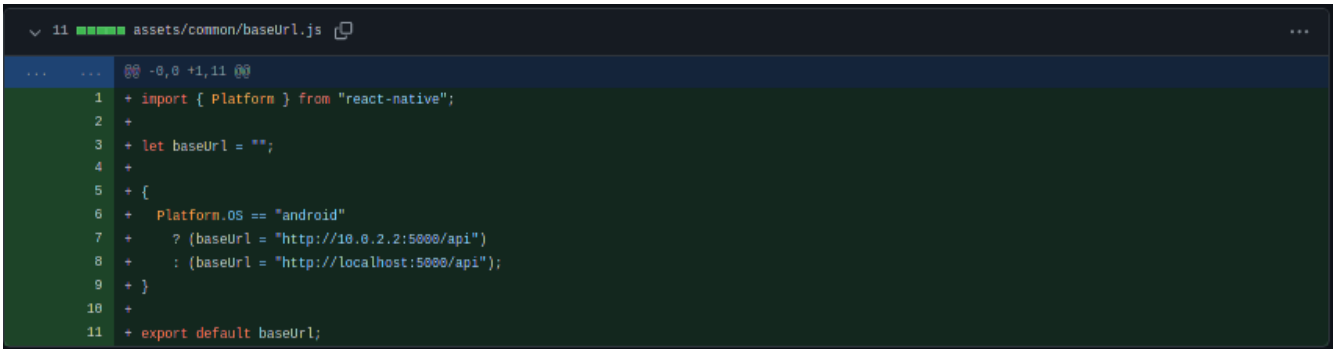
```
@@ -0,0 +1,20 @@
+ import React from "react";
+ import { ScrollView, Dimensions, StyleSheet, Text } from "react-native";
+
+ var { width } = Dimensions.get("window");
+
+ export default function FormContainer(props) {
+   return (
+     <ScrollView contentContainerStyle={styles.container}>
+       <Text style={styles.title}>{props.title}</Text>
+       {props.children}
+     </ScrollView>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     marginTop: 20,
+     marginBottom: 400,
+     width: width,
+     justifyContent: "center",
+     alignItems: "center",
+   },
+   title: {
+     fontSize: 20,
+   },
+ });
```

32 Shared/Form/Input.js

```
@@ -0,0 +1,32 @@
+ import React from "react";
+ import { TextInput, StyleSheet } from "react-native";
+
+ export default function Input(props) {
+   return (
+     <TextInput
+       style={styles.input}
+       placeholder={props.placeholder}
+       name={props.name}
+       id={props.id}
+       value={props.value}
+       autoCorrect={props.autoCorrect}
+       onChangeText={props.onChangeText}
+       onFocus={props.onFocus}
+       secureTextEntry={props.secureTextEntry}
+       keyboardType={props.keyboardType}
+     ></TextInput>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   input: {
+     width: "80%",
+     height: 50,
+     backgroundColor: "white",
+     margin: 10,
+     borderRadius: 20,
+     padding: 10,
+     borderWidth: 2,
+     borderColor: "orange",
+   },
+ });
```

3.2.11. spajanje frontenda sa backendom - connected frontend with backend

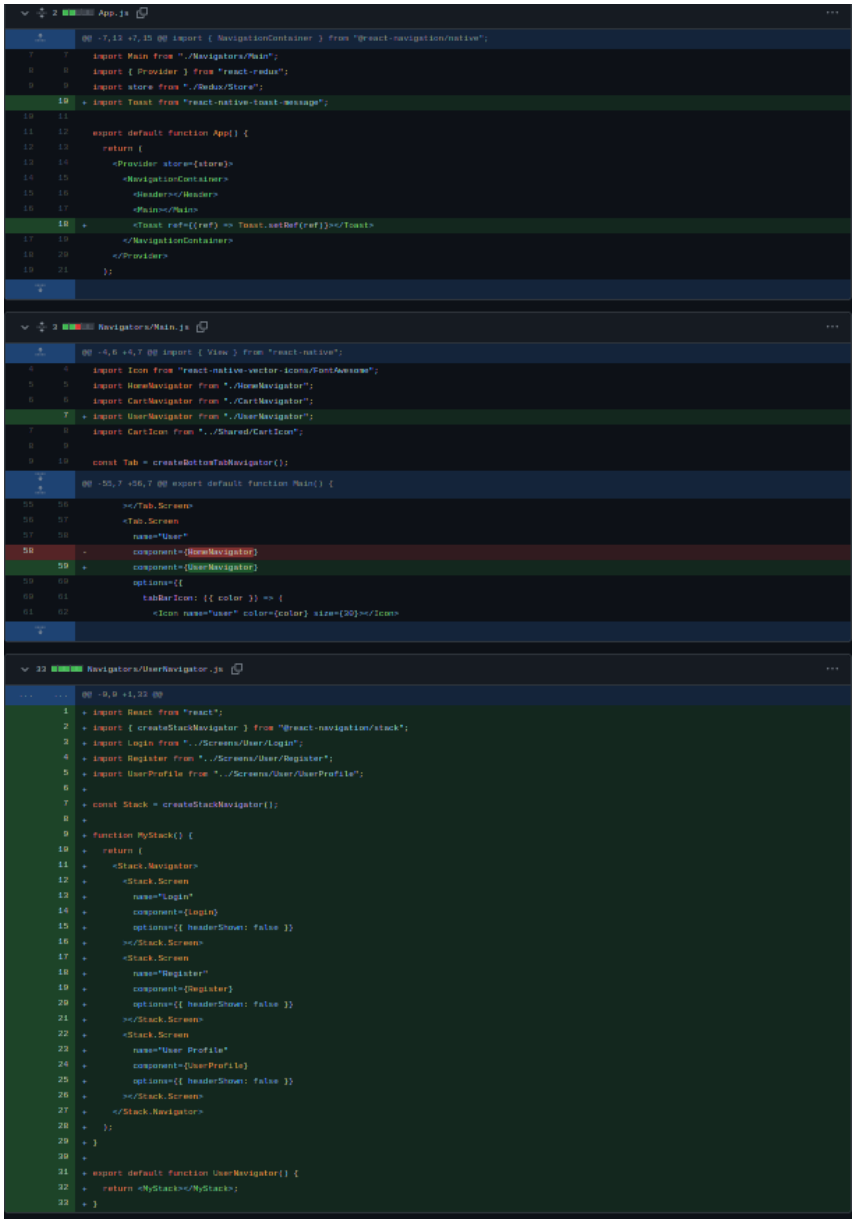
U ovom koraku se dodaje biblioteka „axios“ koja se koristi za sve API call-ove. Mijenja se ProductContainer i daju mu se podaci za produkte i kategorije sa backenda.



3.2.12. dodavanje logina i registeracije 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

```
@@ -7,12 +7,15 @@ import { NavigationContainer } from "@react-navigation/native";
import Main from "./Navigators/Main";
import { Provider } from "react-redux";
import store from "./Redux/Store";
+ import Toast from "react-native-toast-message";

export default function App() {
  return (
    <Provider store={store}>
      <NavigationContainer>
        <Header></Header>
        <Main></Main>
+       <Toast ref={(ref) => Toast.setRef(ref)}></Toast>
      </NavigationContainer>
    </Provider>
  );
```

Navigators/Main.js

```
@@ -4,6 +4,7 @@ import { View } from "react-native";
import Icon from "react-native-vector-icons/FontAwesome";
import HomeNavigator from "./HomeNavigator";
import CartNavigator from "./CartNavigator";
+ import UserNavigator from "./UserNavigator";
import CartIcon from "../Shared/CartIcon";

const Tab = createBottomTabNavigator();
@@ -55,7 +56,7 @@ export default function Main() {
        ></Tab.Screen>
        <Tab.Screen
          name="User"
-         component={HomeNavigator}
+         component={UserNavigator}
          options={{
            tabBarIcon: { color }) => (
              <Icon name="user" color={color} size={30}></Icon>
```

Navigators/UserNavigator.js

```
@@ -0,0 +1,33 @@
+ import React from "react";
+ import { createStackNavigator } from "@react-navigation/stack";
+ import Login from "../Screens/User/Login";
+ import Register from "../Screens/User/Register";
+ import UserProfile from "../Screens/User/UserProfile";
+
+ const Stack = createStackNavigator();
+
+ function MyStack() {
+   return (
+     <Stack.Navigator>
+       <Stack.Screen
+         name="Login"
+         component={Login}
+         options={{ headerShown: false }}
+       ></Stack.Screen>
+       <Stack.Screen
+         name="Register"
+         component={Register}
+         options={{ headerShown: false }}
+       ></Stack.Screen>
+       <Stack.Screen
+         name="User Profile"
+         component={UserProfile}
+         options={{ headerShown: false }}
+       ></Stack.Screen>
+     </Stack.Navigator>
+   );
+ }
+
+ export default function UserNavigator() {
+   return <MyStack></MyStack>;
+ }
```

11 Screens/Products/ProductCard.js

```
@@ -9,6 +9,7 @@ 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");

@@ -36,7 +37,15 @@ 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

```
@@ -10,6 +10,7 @@ import {
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);

@@ -41,7 +42,15 @@ 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>
```

Screens/User/Login.js

```
@@ -0,0 +1,68 @@
+ import React, { useState, useEffect } from "react";
+ import { View, Text, StyleSheet, Button } from "react-native";
+ import FormContainer from "../../Shared/Form/FormContainer";
+ import Input from "../../Shared/Form/Input";
+ import Error from "../../Shared/Error";
+
+ export default function Login(props) {
+   const [email, setEmail] = useState("");
+   const [password, setPassword] = useState("");
+   const [error, setError] = useState("");
+
+   const handleSubmit = () => {
+     const user = {
+       email,
+       password,
+     };
+
+     if (!email || !password) {
+       setError("Please fill in your credentials");
+     } else {
+       setError("");
+     }
+   };
+
+   return (
+     <FormContainer title={"Login"}>
+       <Input
+         placeholder={"Enter Email"}
+         name={"email"}
+         id={"email"}
+         value={email}
+         onChangeText={(text) => setEmail(text.toLowerCase())}
+       ></Input>
+       <Input
+         placeholder={"Enter Password"}
+         name={"password"}
+         id={"password"}
+         value={password}
+         secureTextEntry={true}
+         onChangeText={(text) => setPassword(text)}
+       ></Input>
+       <View style={styles.buttonGroup}>
+         {error ? <Error message={error}></Error> : null}
+         <Button title="Login" onPress={handleSubmit}></Button>
+       </View>
+       <View style={[{ marginTop: 40 }, styles.buttonGroup]}>
+         <Text style={styles.middleText}>Don't have an account yet?</Text>
+         <Button
+           title="Register"
+           onPress={() => {
+             props.navigation.navigate("Register");
+           }}
+         ></Button>
+       </View>
+     </FormContainer>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   buttonGroup: {
+     width: "80%",
+     alignItems: "center",
+   },
+   middleText: {
+     marginBottom: 20,
+     alignSelf: "center",
+   },
+ });
```

10 Screens/User/UserProfile.js

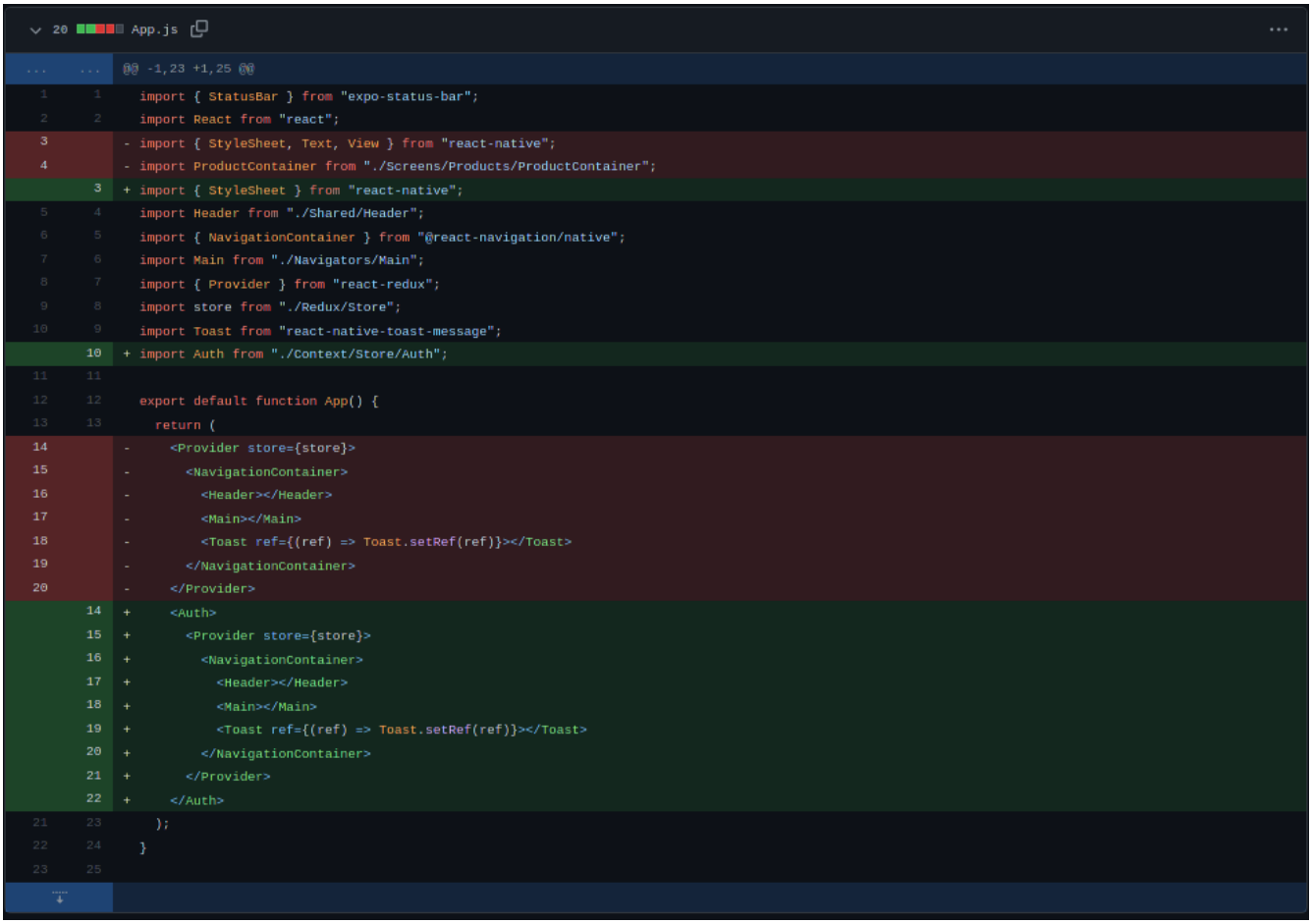
```
@@ -0,0 +1,10 @@
+ import React from "react";
+ import { View, Text } from "react-native";
+
+ export default function UserProfile(props) {
+   return (
+     <View>
+       <Text>User profile page</Text>
+     </View>
+   );
+ }
```

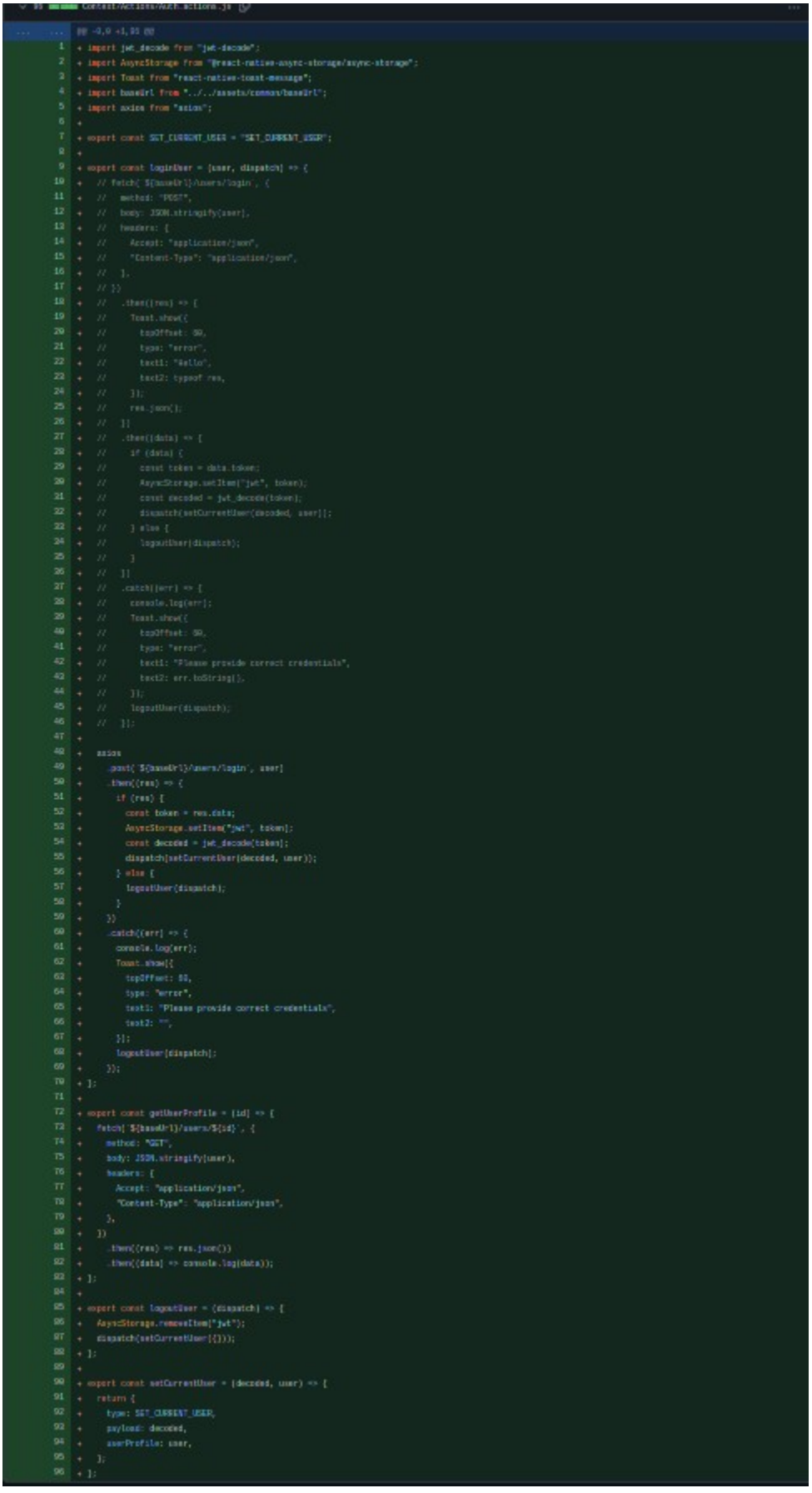
21 Shared/Error.js

```
@@ -0,0 +1,21 @@
+ import React from "react";
+ import { StyleSheet, View, Text } from "react-native";
+
+ export default function Error(props) {
+   return (
+     <View style={styles.container}>
+       <Text style={styles.text}>{props.message}</Text>
+     </View>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     width: "100%",
+     alignItems: "center",
+     margin: 10,
+   },
+   text: {
+     color: "red",
+   },
+ });
```

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 korisniki u state-u. Ponovo se javlja problem od ranije koji je riješen reduxom, ali u ovom slučaju će se riješit 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.





Context/Reducers/Auth.reducer.js

```
@@ -0,0 +1,16 @@
+ import { SET_CURRENT_USER } from "../Actions/Auth.actions";
+ import isEmpty from "../../assets/common/isEmpty";
+
+ export default function (state, action) {
+   switch (action.type) {
+     case SET_CURRENT_USER:
+       return {
+         ...state,
+         isAuthenticated: !isEmpty(action.payload),
+         user: action.payload,
+         userProfile: action.userProfile,
+       };
+     default:
+       return state;
+   }
+ }
```

Context/Store/Auth.js

```
@@ -0,0 +1,41 @@
+ import React, { useEffect, useReducer, useState } from "react";
+ import jwt_decode from "jwt-decode";
+ import AsyncStorage from "@react-native-async-storage/async-storage";
+
+ import authReducer from "../Reducers/Auth.reducer";
+ import { setCurrentUser } from "../Actions/Auth.actions";
+ import AuthGlobal from "./AuthGlobal";
+
+ const Auth = (props) => {
+   const [stateUser, dispatch] = useReducer(authReducer, {
+     isAuthenticated: null,
+     user: {},
+   });
+   const [showChild, setShowChild] = useState(false);
+
+   useEffect(() => {
+     setShowChild(true);
+     if (AsyncStorage.jwt) {
+       const decoded = AsyncStorage.jwt ? AsyncStorage.jwt : "";
+       if (setShowChild) {
+         dispatch(setCurrentUser(jwt_decode(decoded)));
+       }
+     }
+     return () => setShowChild(false);
+   }, []);
+
+   if (!showChild) {
+     return null;
+   } else {
+     return (
+       <AuthGlobal.Provider
+         value={[
+           stateUser,
+           dispatch,
+         ]}
+       >
+         {props.children}
+       </AuthGlobal.Provider>
+     );
+   }
+ };
+
+ export default Auth;
```

Context/Store/AuthGlobal.js

```
@@ -0,0 +1,3 @@
+ import React from "react";
+
+ export default React.createContext();
```

3 Context/Store/AuthGlobal.js

```
@@ -0,0 +1,3 @@
+ import React from "react";
+
+ export default React.createContext();
```

23 Screens/User/Login.js

```
@@ -1,14 +1,31 @@
- import React, { useState, useEffect } from "react";
+ import React, { useState, useEffect, useContext } from "react";
  import { View, Text, StyleSheet, Button } from "react-native";
+ import { useFocusEffect } from "@react-navigation/native";
  import FormContainer from "../../Shared/Form/FormContainer";
  import Input from "../../Shared/Form/Input";
  import Error from "../../Shared/Error";
-
+ import AuthGlobal from "../../Context/Store/AuthGlobal";
+ import { loginUser } from "../../Context/Actions/Auth.actions";
  export default function Login(props) {
+   const context = useContext(AuthGlobal);
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");

+   useEffect(() => {
+     if (context.stateUser.isAuthenticated) {
+       props.navigation.navigate("User Profile");
+     }
+     return () => {};
+   }, [context.stateUser.isAuthenticated]);
+
+   useFocusEffect(() => {
+     if (context.stateUser.isAuthenticated) {
+       props.navigation.navigate("User Profile");
+     }
+     return () => {};
+   });
+
    const handleSubmit = () => {
      const user = {
       email,
@@ -18,7 +35,7 @@ export default function Login(props) {
      if (!email || !password) {
        setError("Please fill in your credentials");
      } else {
-       setError("");
+       loginUser(user, context.dispatch);
      }
    };
```

73 Screens/User/UserProfile.js

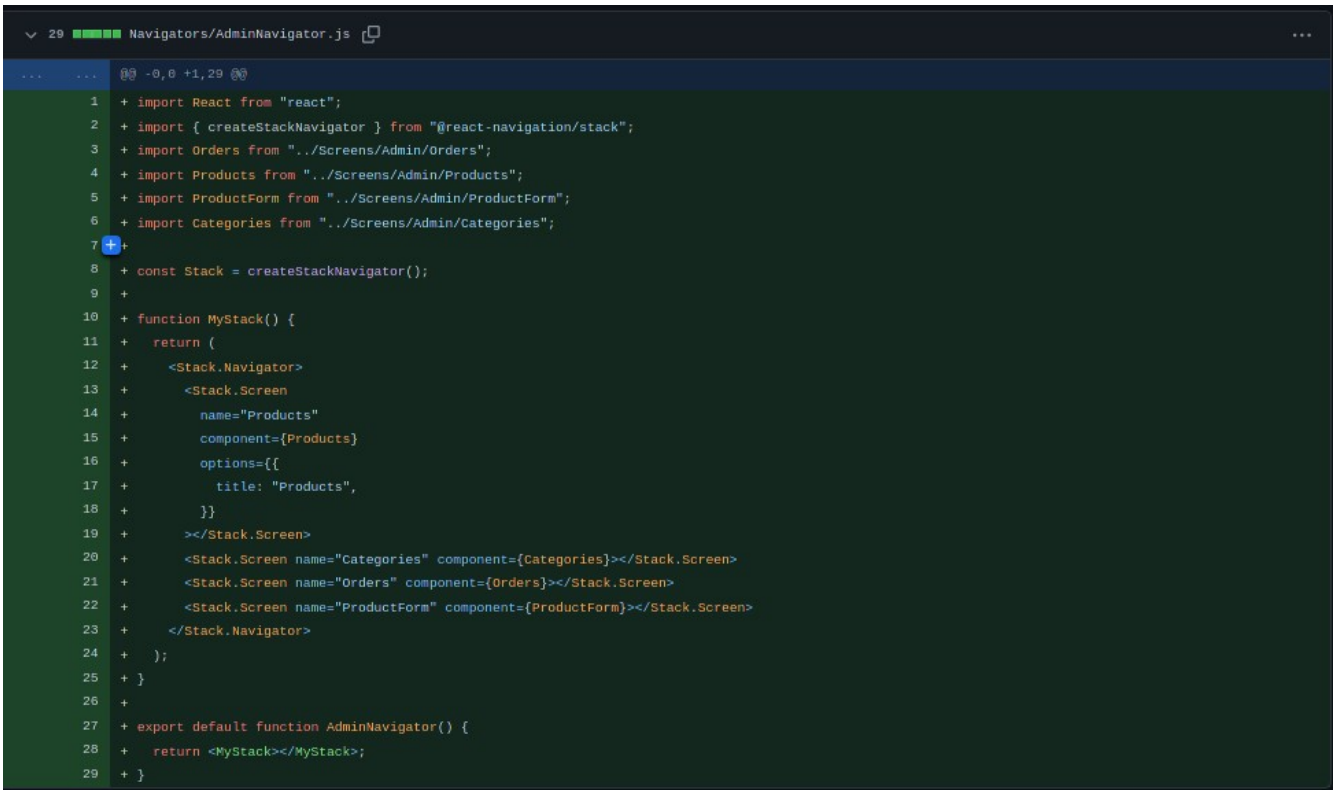
```
@@ -1,10 +1,73 @@
- import React from "react";
- import { View, Text } from "react-native";
+ import React, { useContext, useState, useCallback, useEffect } from "react";
+ import { View, Text, ScrollView, Button, StyleSheet } from "react-native";
+ import { Container, Content } from "native-base";
+ import { useFocusEffect } from "@react-navigation/native";
+ import AsyncStorage from "@react-native-async-storage/async-storage";
+ import axios from "axios";
+ import baseUrl from "../../assets/common/baseUrl";
+ import AuthGlobal from "../../Context/Store/AuthGlobal";
+ import { logoutUser } from "../../Context/Actions/Auth.actions";

  export default function UserProfile(props) {
+   const context = useContext(AuthGlobal);
+   const [userProfile, setUserProfile] = useState();
+   const [jwt, setJwt] = useState("na");
+
+   useEffect(() => {
+     if (!context.stateUser.isAuthenticated) {
+       props.navigation.navigate("Login");
+     }
+
+     AsyncStorage.getItem("jwt")
+       .then((res) => {
+         setJwt(res);
+         axios
+           .get(`${baseUrl}/users/${context.stateUser.user.userId}`, {
+             headers: { Authorization: `Bearer ${res}` },
+           })
+           .then((user) => setUserProfile(user.data));
+       })
+       .catch((err) => console.log(err));
+     return () => {
+       setUserProfile();
+     };
+   }, [context.stateUser.isAuthenticated]);
+
    return (
-     <View>
-       <Text>User profile page</Text>
-     </View>
+     <Container style={styles.container}>
+       <ScrollView contentContainerStyle={styles.subContainer}>
+         <Text style={{ fontSize: 30 }}>
+           {userProfile ? userProfile.name : ""}
+         </Text>
+         <View style={{ marginTop: 20 }}>
+           <Text style={{ margin: 10 }}>
+             Email: {userProfile ? userProfile.email : ""}
+           </Text>
+           <Text style={{ margin: 10 }}>
+             Phone: {userProfile ? userProfile.phone : ""}
+           </Text>
+         </View>
+         <View style={{ marginTop: 80 }}>
+           <Button
+             title="Logout"
+             onPress={() => [
+               AsyncStorage.removeItem("jwt"),
+               logoutUser(context.dispatch),
+             ]}
+           ></Button>
+         </View>
+       </ScrollView>
+     </Container>
    );
  }
+
+ const styles = StyleSheet.create({
+   container: {
+     flex: 1,
+     alignItems: "center",
+   },
+   subContainer: {
+     alignItems: "center",
+     marginTop: 20,
+   },
+ });
```

13 assets/common/isEmpty.js

```
@@ -0,0 +1,13 @@
+ const isEmpty = (value) => {
+   if (
+     value === undefined ||
+     value === null ||
+     (typeof value === "object" && Object.keys(value).length === 0) ||
+     (typeof value === "string" && value.trim().length === 0)
+   ) {
+     return true;
+   } else {
+     return false;
+   }
+ };
+ 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.



28 Navigators/Main.js

```
@@ -1,15 +1,19 @@
- import React from "react";
+ import React, { useContext } from "react";
  import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
  import { View } from "react-native";
  import Icon from "react-native-vector-icons/FontAwesome";
  import HomeNavigator from "./HomeNavigator";
  import CartNavigator from "./CartNavigator";
  import UserNavigator from "./UserNavigator";
+ import AdminNavigator from "./AdminNavigator";
  import CartIcon from "../Shared/CartIcon";
+ import AuthGlobal from "../Context/Store/AuthGlobal";

  const Tab = createBottomTabNavigator();

  export default function Main() {
+   const context = useContext(AuthGlobal);
+
    return (
      <Tab.Navigator
        initialRouteName="Home"
@@ -45,15 +49,17 @@ export default function Main() {
            ),
          }}
        ></Tab.Screen>
-       <Tab.Screen
-         name="Admin"
-         component={HomeNavigator}
-         options={{
-           tabBarIcon: ({ color }) => (
-             <Icon name="cog" color={color} size={30}></Icon>
-           ),
-         }}
-       ></Tab.Screen>
+       {context.stateUser.user.isAdmin ? (
+         <Tab.Screen
+           name="Admin"
+           component={AdminNavigator}
+           options={{
+             tabBarIcon: ({ color }) => (
+               <Icon name="cog" color={color} size={30}></Icon>
+             ),
+           }}
+         ></Tab.Screen>
+       ) : null}
        <Tab.Screen
          name="User"
          component={UserNavigator}
```

10 Screens/Admin/Categories.js

```
@@ -0,0 +1,10 @@
+ import React from "react";
+ import { View, Text } from "react-native";
+
+ export default function Categories(props) {
+   return (
+     <View>
+       <Text>Categories screen</Text>
+     </View>
+   );
+ }
```

123 Screens/Admin/ListItem.js

```
@@ -0,0 +1,123 @@
+ import React, { useState } from "react";
+ import {
+   View,
+   StyleSheet,
+   Text,
+   Image,
+   TouchableHighlight,
+   TouchableOpacity,
+   Dimensions,
+   Button,
+   Modal,
+ } from "react-native";
+ import Icon from "react-native-vector-icons/FontAwesome";
+
+ var { width } = Dimensions.get("window");
+
+ export default function ListItem(props) {
+   const [modalVisible, setModalVisible] = useState(false);
+
+   return (
+     <View>
+       <Modal
+         animationType="fade"
+         transparent={true}
+         visible={modalVisible}
+         onRequestClose={() => setModalVisible(false)}
+       >
+         <View style={styles.centerView}>
+           <View style={styles.modalView}>
+             <TouchableOpacity
+               underlayColor="#E8E8E8"
+               onPress={() => setModalVisible(false)}
+               style={{
+                 alignSelf: "flex-end",
+                 position: "absolute",
+                 top: 5,
+                 right: 10,
+               }}
+             >
+               <Icon name="close" size={20}></Icon>
+             </TouchableOpacity>
+             <Button
+               title="Edit"
+               onPress={() => [
+                 props.navigation.navigate("ProductForm"),
+                 setModalVisible(false),
+               ]}
+             ></Button>
+             <Button title="Delete"></Button>
+           </View>
+         </View>
+       </Modal>
+       <TouchableOpacity
+         onPress={() => {
+           props.navigation.navigate("Product Detail", { item: props });
+         }}
+         onLongPress={() => setModalVisible(true)}
+         style={[
+           styles.container,
+           { backgroundColor: props.index % 2 === 0 ? "white" : "gainsboro" },
+         ]}
+       >
```

```
+         <Image
+           source={{
+             uri: props.image
+               ? props.image
+               : "https://cdn.pixabay.com/photo/2012/04/01/17/29/box-23640_960_720.png",
+           }}
+           style={styles.image}
+           resizeMode="contain"
+         ></Image>
+         <Text style={styles.item}>{props.brand}</Text>
+         <Text style={styles.item} numberOfLines={1} ellipsizeMode="tail">
+           {props.name}
+         </Text>
+         <Text style={styles.item} numberOfLines={1} ellipsizeMode="tail">
+           {props.category.name}
+         </Text>
+         <Text style={styles.item}>$ {props.price}</Text>
+       </TouchableOpacity>
+     </View>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   container: {
+     flexDirection: "row",
+     padding: 5,
+     width: width,
+   },
+   image: {
+     borderRadius: 50,
+     width: width / 6,
+     height: 20,
+     margin: 2,
+   },
+   item: {
+     flexWrap: "wrap",
+     margin: 3,
+     width: width / 6,
+   },
+   centerView: {
+     flex: 1,
+     justifyContent: "center",
+     alignItems: "center",
+     marginTop: 22,
+   },
+   modalView: {
+     margin: 20,
+     backgroundColor: "white",
+     borderRadius: 20,
+     padding: 35,
+     alignItems: "center",
+     shadowColor: "#000",
+     shadowOffset: {
+       width: 0,
+       height: 2,
+     },
+     shadowOpacity: 0.25,
+     shadowRadius: 3.84,
+     elevation: 5,
+   },
+ });
```

10 Screens/Admin/Orders.js

```
@@ -0,0 +1,10 @@
+ import React from "react";
+ import { View, Text } from "react-native";
+
+ export default function Orders(props) {
+   return (
+     <View>
+       <Text>Orders screen</Text>
+     </View>
+   );
+ }
```

10 Screens/Admin/ProductForm.js

```
@@ -0,0 +1,10 @@
+ import React from "react";
+ import { View, Text } from "react-native";
+
+ export default function ProductForm(props) {
+   return (
+     <View>
+       <Text>ProductForm screen</Text>
+     </View>
+   );
+ }
```

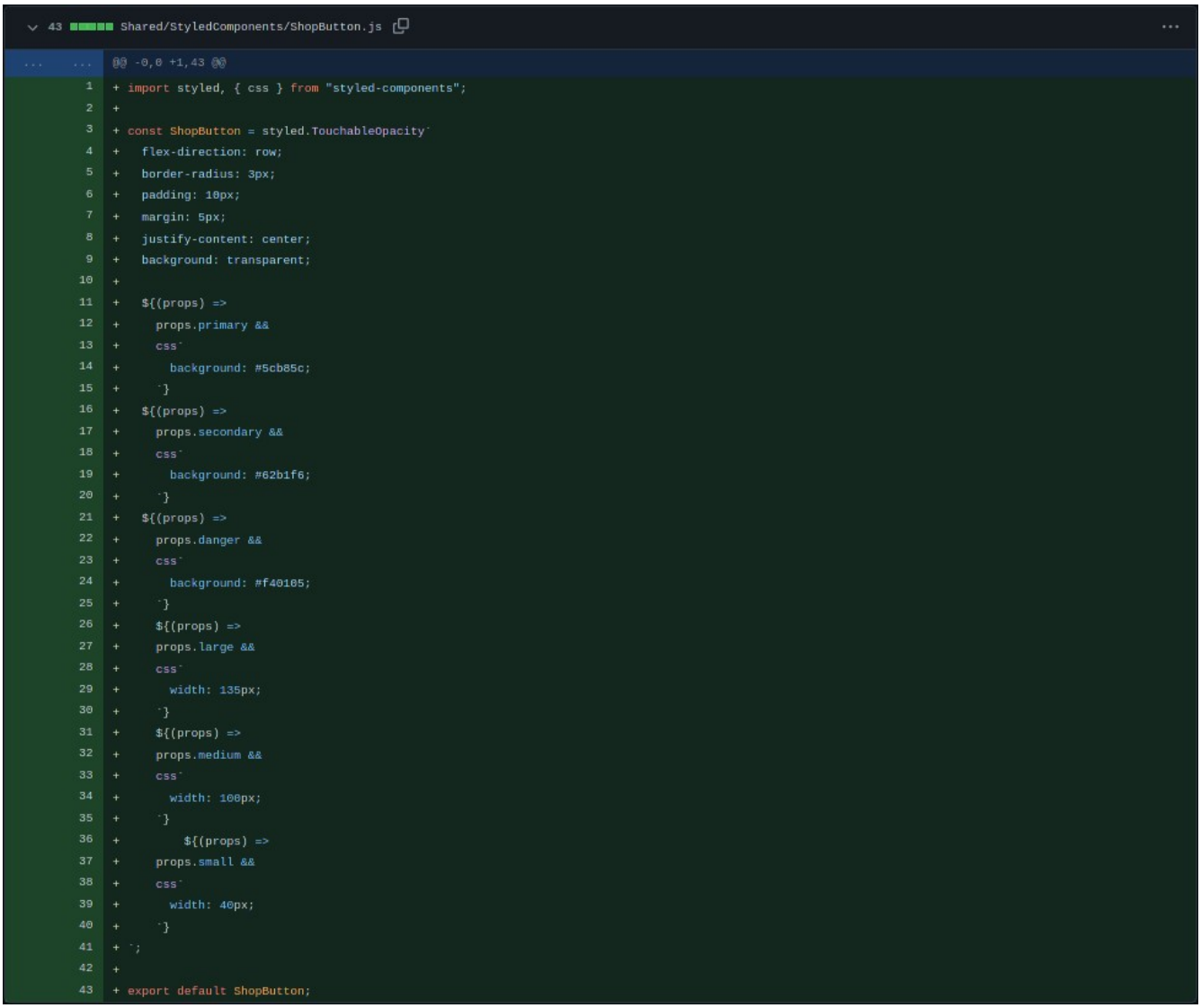
127 Screens/Admin/Products.js

```
@@ -0,0 +1,127 @@
+ import React, { useState, useCallback } from "react";
+ import {
+   View,
+   Text,
+   FlatList,
+   ActivityIndicator,
+   StyleSheet,
+   Dimensions,
+   Button,
+ } from "react-native";
+ import { Header, Item, Input } from "native-base";
+ import Icon from "react-native-vector-icons/FontAwesome";
+ import { useFocusEffect } from "@react-navigation/native";
+ import axios from "axios";
+ import baseUrl from "../../assets/common/baseUrl";
+ import AsyncStorage from "@react-native-async-storage/async-storage";
+ import ListItem from "./ListItem";
+
+ var { height, width } = Dimensions.get("window");
+
+ const listHeader = () => {
+   return (
+     <View elevation={1} style={styles.listHeader}>
+       <View style={styles.headerItem}></View>
+       <View style={styles.headerItem}>
+         <Text style={{ fontWeight: "bold" }}>Brand</Text>
+       </View>
+       <View style={styles.headerItem}>
+         <Text style={{ fontWeight: "bold" }}>Name</Text>
+       </View>
+       <View style={styles.headerItem}>
+         <Text style={{ fontWeight: "bold" }}>Category</Text>
+       </View>
+       <View style={styles.headerItem}>
+         <Text style={{ fontWeight: "bold" }}>Price</Text>
+       </View>
+     </View>
+   );
+ };
+
+ export default function Products(props) {
+   const [productList, setProductList] = useState();
+   const [productFilter, setProductFilter] = useState([]);
+   const [loading, setLoading] = useState(true);
+   const [token, setToken] = useState();
+
+   useFocusEffect(
+     useCallback(() => {
+       AsyncStorage.getItem("jwt")
+         .then((res) => setToken(res))
+         .catch((err) => console.log(err));
+
+       axios.get( ${baseUrl}/products ).then((res) => {
+         setProductList(res.data);
+         setProductFilter(res.data);
+         setLoading(false);
+       });
+       return () => {
+         setProductList();
+         setProductFilter();
+         setLoading(true);
+       };
+     }, [])
+   );
+
+   const searchProduct = (text) => {
+     if (text === "") {
+       setProductFilter(productList);
+     }
+     setProductFilter(
+       productList.filter((i) =>
+         i.name.toLowerCase().includes(text.toLowerCase())
+       )
+     );
+   };
+
```

```
+   return (
+     <View>
+       <View>
+         <Header searchBar rounded>
+           <Item style={{ padding: 5 }}>
+             <Icon name="search"></Icon>
+             <Input
+               placeholder="Search"
+               onChangeText={(text) => searchProduct(text)}
+             ></Input>
+           </Item>
+         </Header>
+       </View>
+       {loading ? (
+         <View style={styles.spinner}>
+           <ActivityIndicator size="large" color="red"></ActivityIndicator>
+         </View>
+       ) : (
+         <FlatList
+           ListHeaderComponent={listHeader}
+           data={productFilter}
+           renderItem={({ item, index }) => (
+             <ListItem
+               {...item}
+               navigation={props.navigation}
+               index={index}
+             ></ListItem>
+           )}
+           keyExtractor={(item) => item.id}
+         ></FlatList>
+       )}
+     </View>
+   );
+ }
+
+ const styles = StyleSheet.create({
+   listHeader: {
+     flexDirection: "row",
+     padding: 5,
+     backgroundColor: "gainsboro",
+   },
+   headerItem: {
+     margin: 3,
+     width: width / 6,
+   },
+   spinner: {
+     height: height / 2,
+     alignItems: "center",
+     justifyContent: "center",
+   },
+ });
```

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 kreairanja komponenti potrebno je izmijeniti već postojeće buttone u aplikaciji sa ShopButton.js komponentom.



```
43 Shared/StyledComponents/ShopButton.js
@@ -0,0 +1,43 @@
+ import styled, { css } from "styled-components";
+
+ const ShopButton = styled.TouchableOpacity`
+   flex-direction: row;
+   border-radius: 3px;
+   padding: 10px;
+   margin: 5px;
+   justify-content: center;
+   background: transparent;
+
+   ${(props) =>
+     props.primary &&
+     css`
+       background: #5cb85c;
+     `}
+   ${(props) =>
+     props.secondary &&
+     css`
+       background: #62b1f6;
+     `}
+   ${(props) =>
+     props.danger &&
+     css`
+       background: #f40105;
+     `}
+     ${(props) =>
+     props.large &&
+     css`
+       width: 135px;
+     `}
+     ${(props) =>
+     props.medium &&
+     css`
+       width: 100px;
+     `}
+        ${(props) =>
+     props.small &&
+     css`
+       width: 40px;
+     `}
+ `;
+
+ export default ShopButton;
```

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

28 Shared/StyledComponents/TrafficLight.js

```
@@ -0,0 +1,28 @@
+ import styled, { css } from "styled-components";
+
+ const TrafficLight = styled.View`
+   border-radius: 50px;
+   width: 10px;
+   height: 10px;
+   padding: 10px;
+
+   ${(props) =>
+     props.available &&
+     css`
+       background: #afec1a;
+     `}
+
+   ${(props) =>
+     props.limited &&
+     css`
+       background: #dde033;
+     `}
+
+     ${(props) =>
+     props.unavailable &&
+     css`
+       background: #ec241a;
+     `}
+ `;
+
+ 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 @@
- import React from "react";
- import { View, Text } from "react-native";
+ import React, { useState, useEffect } from "react";
+ import {
+   View,
+   Text,
+   Image,
+   StyleSheet,
+   TouchableOpacity,
+   Platform,
+ } from "react-native";
+ import { Item, Picker } from "native-base";
+ import FormContainer from "../../Shared/Form/FormContainer";
+ import Input from "../../Shared/Form/Input";
+ import ShopButton from "../../Shared/StyledComponents/ShopButton";
+ import Error from "../../Shared/Error";
+ import Icon from "react-native-vector-icons/FontAwesome";
+ import Toast from "react-native-toast-message";
+ import AsyncStorage from "@react-native-async-storage/async-storage";
+ import baseUrl from "../../assets/common/baseUrl";
+ import axios from "axios";
+ import * as ImagePicker from "expo-image-picker";
+ import mime from "mime";

  export default function ProductForm(props) {
+   const [pickerValue, setPickerValue] = useState();
+   const [brand, setBrand] = useState();
+   const [name, setName] = useState();
+   const [price, setPrice] = useState();
+   const [description, setDescription] = useState();
+   const [image, setImage] = useState();
+   const [mainImage, setMainImage] = useState();
+   const [category, setCategory] = useState();
+   const [categories, setCategories] = useState([]);
+   const [token, setToken] = useState();
+   const [error, setError] = useState();
+   const [countInStock, setCountInStock] = useState();
+   const [rating, setRating] = useState(0);
+   const [isFeatured, setIsFeatured] = useState(false);
+   const [richDescription, setRichDescription] = useState();
+   const [numReviews, setNumReviews] = useState(0);
+   const [item, setItem] = useState(null);
+   const [updatedImage, setUpdatedImage] = useState(false);
+
+   useEffect(() => {
+     if (!props.route.params) {
+       setItem(null);
+     } else {
+       setItem(props.route.params.item);
+       setBrand(props.route.params.item.brand);
+       setName(props.route.params.item.name);
+       setPrice(props.route.params.item.price.toString());
+       setDescription(props.route.params.item.description);
+       setMainImage(props.route.params.item.image);
+       setCategory(props.route.params.item.category._id);
+       setCountInStock(props.route.params.item.countInStock.toString());
+     }
```

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

```
+    AsyncStorage.getItem("jwt")
+      .then((res) => setToken(res))
+      .catch((err) => console.log(err));
+
+    axios
+      .get(`${baseUrl}/categories`)
+      .then((res) => setCategories(res.data))
+      .catch((err) => alert("Error loading categories"));
+
+    (async () => {
+      if (Platform.OS !== "web") {
+        const { status } = await ImagePicker.requestCameraPermissionsAsync();
+        if (status !== "granted") {
+          alert("Sorry, we need camera permissions.");
+        }
+      }
+    })();
+    return () => {
+      setCategories([]);
+    };
+  }, []);
+
+  const pickImage = async () => {
+    let result = await ImagePicker.launchImageLibraryAsync({
+      mediaTypes: ImagePicker.MediaTypeOptions.All,
+      allowsEditing: true,
+      aspect: [4, 3],
+      quality: 1,
+    });
+
+    if (!result.cancelled) {
+      setMainImage(result.uri);
+      setImage(result.uri);
+      setUpdatedImage(true);
+    }
+  };
+
+  const addProducts = () => {
+    if (
+      !name ||
+      !brand ||
+      !price ||
+      !description ||
+      !category ||
+      !countInStock
+    ) {
+      setError("Please fill in the form correctly.");
+      return;
+    }
+    let formData = new FormData();
+
+    if (item !== null && !image) {
+      formData.append("image", item.image);
+    } else {
+      const newImageUri = "file:///" + image.split("file:/").join("");
+      formData.append("image", {
+        uri: newImageUri,
+        type: mime.getType(newImageUri),
+        name: newImageUri.split("/").pop(),
+      });
+    }
```

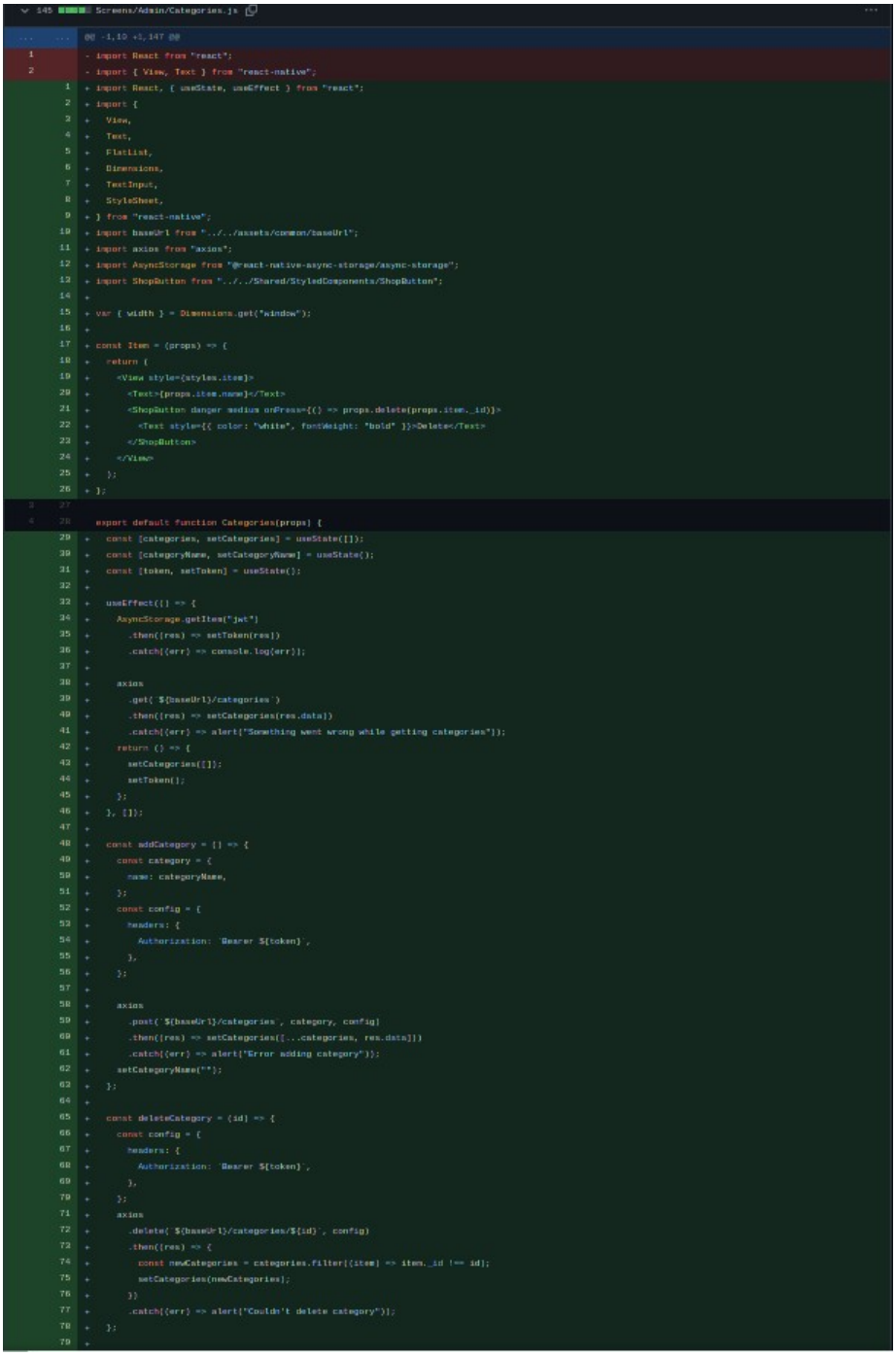
```
+    formData.append("name", name);
+    formData.append("brand", brand);
+    formData.append("price", price);
+    formData.append("description", description);
+    formData.append("category", category);
+    formData.append("countInStock", countInStock);
+    formData.append("richDescription", richDescription);
+    formData.append("rating", rating);
+    formData.append("numReviews", numReviews);
+    formData.append("isFeatured", isFeatured);
+    const config = {
+      headers: {
+        "Content-Type": "multipart/form-data",
+        Authorization: `Bearer ${token}`,
+      },
+    };
+    if (item !== null) {
+      axios
+        .put(`${baseUrl}/products/${item._id}`, formData, config)
+        .then((res) => {
+          if (res.status == 200 || res.status == 201) {
+            Toast.show({
+              topOffset: 60,
+              type: "success",
+              text1: "Product edited",
+              text2: "",
+            });
+            setTimeout(() => {
+              props.navigation.navigate("Products");
+            }, 500);
+          }
+        })
+        .catch((err) => {
+          Toast.show({
+            topOffset: 60,
+            type: "error",
+            text1: "Something went wrong",
+            text2: err.toString(),
+          });
+        });
+    } else {
+      axios
+        .post(`${baseUrl}/products`, formData, config)
+        .then((res) => {
+          if (res.status == 200 || res.status == 201) {
+            Toast.show({
+              topOffset: 60,
+              type: "success",
+              text1: "New Product added",
+              text2: "",
+            });
+            setTimeout(() => {
+              props.navigation.navigate("Products");
+            }, 500);
+          }
+        })
+        .catch((err) => {
+          Toast.show({
+            topOffset: 60,
+            type: "error",
+            text1: "Something went wrong",
+            text2: err.toString(),
+          });
+        });
+    }
+  };
+
```

```
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>
```

```
+       <Item picker>
+         <Picker
+           mode="dropdown"
+           iosIcon={<Icon color={"#007aff"} name="arrow-down"></Icon>}
+           style={{ width: undefined, height: 30 }}
+           placeholder="Select your Category"
+           selectedValue={pickerValue}
+           placeholderStyle={{ color: "#007aff" }}
+           placeholderIconColor="#007aff"
+           onValueChange={(e) => [setPickerValue(e), setCategory(e)]}
+         >
+           {categories.map((c) => (
+             <Picker.Item key={c._id} label={c.name} value={c._id}></Picker.Item>
+           ))}
+         </Picker>
+       </Item>
+       {error ? <Error message={error}></Error> : null}
+       <View style={styles.buttonContainer}>
+         <ShopButton large primary onPress={addProducts}>
+           <Text style={styles.buttonText}>Confirm</Text>
+         </ShopButton>
+       </View>
+     </FormContainer>
    );
  }
+
+ const styles = StyleSheet.create({
+   label: {
+     width: "80%",
+     marginTop: 10,
+   },
+   buttonContainer: {
+     width: "80%",
+     marginBottom: 80,
+     marginTop: 20,
+     alignItems: "center",
+   },
+   buttonText: {
+     color: "white",
+   },
+   imageContainer: {
+     width: 200,
+     height: 200,
+     borderStyle: "solid",
+     borderWidth: 8,
+     padding: 8,
+     justifyContent: "center",
+     borderRadius: 100,
+     borderColor: "#E0E0E0",
+     elevation: 10,
+   },
+   image: {
+     width: "100%",
+     height: "100%",
+     borderRadius: 100,
+   },
+   imagePicker: {
+     position: "absolute",
+     right: 5,
+     bottom: 5,
+     backgroundColor: "grey",
+     padding: 8,
+     borderRadius: 100,
+     elevation: 20,
+   },
+ });
```

3.2.17. dodavanje mogućnosti za brisanje i dodavanje kategorija iz admin panela - added abilty 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.



```
    return (
-     <View>
-       <Text>Categories screen</Text>
+     <View style={{ position: "relative", height: "100%" }}>
+       <View style={{ marginBottom: 60 }}>
+         <FlatList
+           data={categories}
+           renderItem={({ item, index }) => (
+             <Item item={item} index={index} delete={deleteCategory}></Item>
+           )}
+           keyExtractor={(item) => item.id}
+         ></FlatList>
+       </View>
+       <View style={styles.bottomBar}>
+         <View>
+           <Text>Add Category</Text>
+         </View>
+         <View style={{ width: width / 2.5 }}>
+           <TextInput
+             value={categoryName}
+             onChangeText={(text) => setCategoryName(text)}
+             style={styles.input}
+           ></TextInput>
+         </View>
+         <View>
+           <ShopButton medium primary onPress={addCategory}>
+             <Text style={{ color: "white", fontWeight: "bold" }}>Submit</Text>
+           </ShopButton>
+         </View>
+       </View>
      </View>
    );
  }
+
+ const styles = StyleSheet.create({
+   bottomBar: {
+     backgroundColor: "white",
+     width: width,
+     height: 60,
+     padding: 2,
+     flexDirection: "row",
+     alignItems: "center",
+     justifyContent: "space-between",
+     position: "absolute",
+     bottom: 0,
+     left: 0,
+   },
+   input: {
+     height: 40,
+     borderColor: "gray",
+     borderWidth: 1,
+   },
+   item: {
+     shadowColor: "#000",
+     shadowOffset: {
+       width: 0,
+       height: 2,
+     },
+     shadowOpacity: 0.2,
+     shadowRadius: 1,
+     elevation: 1,
+     padding: 5,
+     margin: 5,
+     backgroundColor: "white",
+     flexDirection: "row",
+     alignItems: "center",
+     justifyContent: "space-between",
+     borderRadius: 5,
+   },
+ });
```

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 @@
+ import React, { useState, useEffect } from "react";
+ import { View, Text, StyleSheet } from "react-native";
+ import { Picker } from "native-base";
+ import Icon from "react-native-vector-icons/FontAwesome";
+ import TrafficLight from "./StyledComponents/TrafficLight";
+ import ShopButton from "./StyledComponents/ShopButton";
+ import Toast from "react-native-toast-message";
+
+ import AsyncStorage from "@react-native-async-storage/async-storage";
+ import axios from "axios";
+ import baseUrl from "../assets/common/baseUrl";
+
+ const codes = [
+   {
+     name: "pending",
+     code: "3",
+   },
+   {
+     name: "shipped",
+     code: "2",
+   },
+   {
+     name: "delivered",
+     code: "1",
+   },
+ ];
+
+ export default function OrderCard(props) {
+   const [orderStatus, setOrderStatus] = useState();
+   const [statusText, setStatusText] = useState();
+   const [statusChange, setStatusChange] = useState();
+   const [token, setToken] = useState();
+   const [cardColor, setCardColor] = useState();
+
+   useEffect(() => {
+     if (props.editMode) {
+       AsyncStorage.getItem("jwt")
+         .then((res) => setToken(res))
+         .catch((err) => console.log(err));
+     }
+     if (props.status == "3") {
+       setOrderStatus(<TrafficLight unavailable></TrafficLight>);
+       setStatusText("pending");
+       setCardColor("#E74C3C");
+     } else if (props.status == "2") {
+       setOrderStatus(<TrafficLight limited></TrafficLight>);
+       setStatusText("shipped");
+       setCardColor("#F1C40F");
+     } else if (props.status == "1") {
+       setOrderStatus(<TrafficLight available></TrafficLight>);
+       setStatusText("delivered");
+       setCardColor("#E74C3C");
+     }
+     return () => {
+       setOrderStatus();
+       setStatusText();
+       setCardColor();
+     };
+   }, []);
```

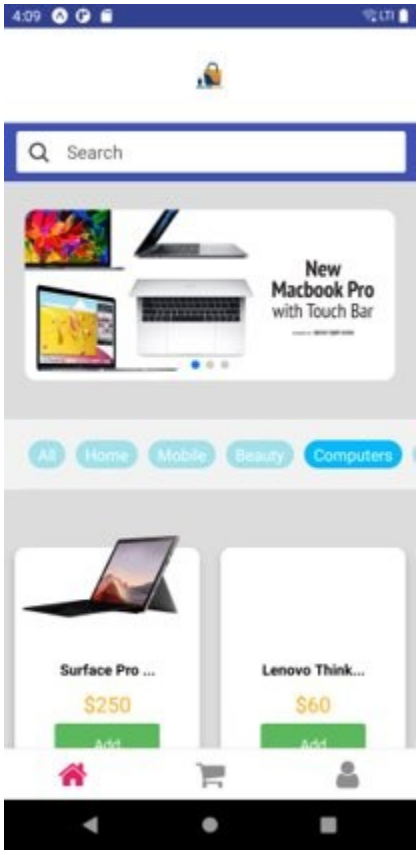
```
+
+  const updateOrder = () => {
+    const config = {
+      headers: {
+        Authorization: `Bearer ${token}`,
+      },
+    };
+
+    const order = {
+      city: props.city,
+      country: props.country,
+      dateOrdered: props.dateOrdered,
+      orderItems: props.orderItems,
+      phone: props.phone,
+      shippingAddress1: props.shippingAddress1,
+      shippingAddress2: props.shippingAddress2,
+      status: statusChange,
+      totalPrice: props.totalPrice,
+      user: props.user,
+      zip: props.zip,
+    };
+
+    axios
+      .put(`${baseUrl}/orders/${props._id}`, order, config)
+      .then((res) => {
+        if (res.status == 200 || res.status == 201) {
+          Toast.show({
+            topOffset: 60,
+            type: "success",
+            text1: "Order successfuly updated",
+            text2: "",
+          });
+          if (statusChange == "3") {
+            setOrderStatus(<TrafficLight unavailable></TrafficLight>);
+            setStatusText("pending");
+            setCardColor("#E74C3C");
+          } else if (statusChange == "2") {
+            setOrderStatus(<TrafficLight limited></TrafficLight>);
+            setStatusText("shipped");
+            setCardColor("#F1C40F");
+          } else if (statusChange == "1") {
+            setOrderStatus(<TrafficLight available></TrafficLight>);
+            setStatusText("delivered");
+            setCardColor("#2ECC71");
+          }
+        }
+      })
+      .catch((err) => {
+        Toast.show({
+          topOffset: 60,
+          type: "error",
+          text1: "Something went wrong",
+          text2: "Please try again",
+        });
+      });
+  };
+
```

```
+   return (
+     <View style={[styles.container, { backgroundColor: cardColor }]}>
+       <View style={styles.container}>
+         <Text>Order Number: #{props._id}</Text>
+       </View>
+       <View style={{ marginTop: 10 }}>
+         <Text>
+           Status: {statusText} {orderStatus}
+         </Text>
+         <Text>
+           Address: {props.shippingAddress1} {props.shippingAddress2}
+         </Text>
+         <Text>City: {props.city}</Text>
+         <Text>Country: {props.country}</Text>
+         <Text>Date ordered: {props.dateOrdered.split("T")[0]}</Text>
+         <View style={styles.priceContainer}>
+           <Text>Price: </Text>
+           <Text style={styles.price}>$ {props.totalPrice}</Text>
+         </View>
+         {props.editMode ? (
+           <View>
+             <Picker
+               mode="dropdown"
+               iosIcon={<Icon color={"#007aff"} name="arrow-down"></Icon>}
+               style={{ width: undefined, height: 20 }}
+               selectedValue={statusChange}
+               placeholder="Change status"
+               placeholderIconColor={{ color: "#007aff" }}
+               onValueChange={(e) => setStatusChange(e)}
+             >
+               {codes.map((c) => (
+                 <Picker.Item
+                   key={c.code}
+                   label={c.name}
+                   value={c.code}
+                 ></Picker.Item>
+               ))}
+             </Picker>
+             <ShopButton secondary large onPress={updateOrder}>
+               <Text>Update</Text>
+             </ShopButton>
+           </View>
+         ) : null}
+       </View>
+     </View>
+   );
+ }
+
```

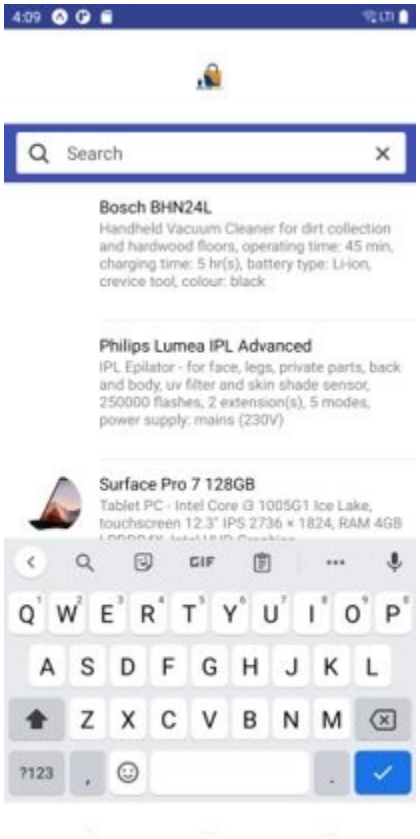
```
+ const styles = StyleSheet.create({
+   container: {
+     padding: 20,
+     margin: 10,
+     borderRadius: 10,
+   },
+   title: {
+     backgroundColor: "#62B1F6",
+     padding: 5,
+   },
+   priceContainer: {
+     marginTop: 10,
+     alignSelf: "flex-end",
+     flexDirection: "row",
+   },
+   price: {
+     color: "white",
+     fontWeight: "bold",
+   },
+ });
```

4. Izled aplikacije

4.1. Početna strana

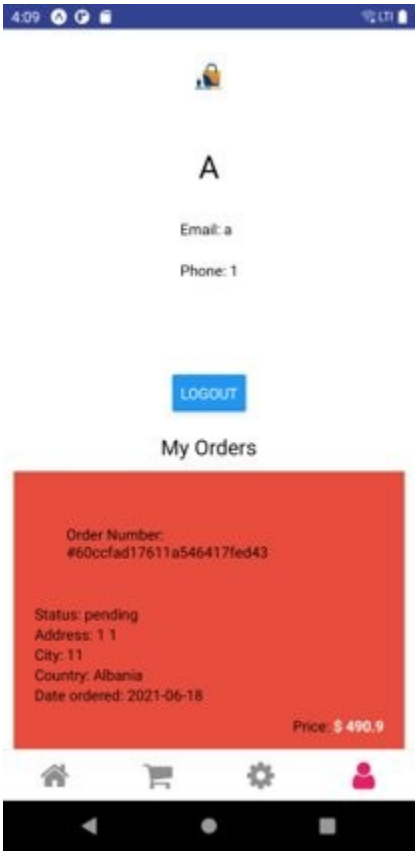


4.2. Search bar

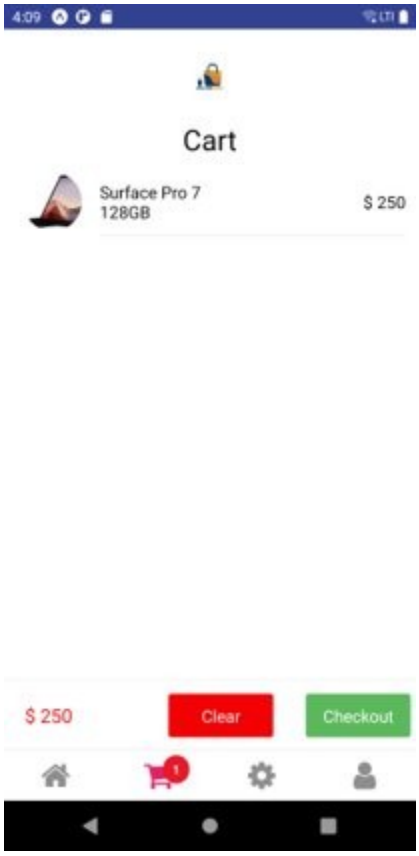


4.3. Pojedinačni produkt

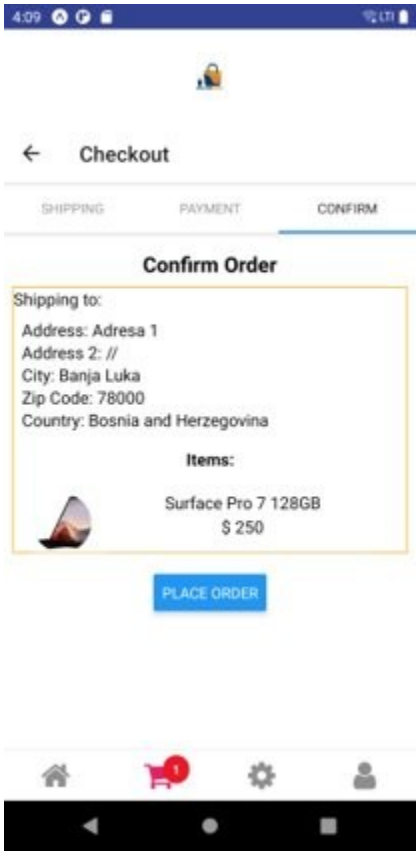
4.4. Korisnički profil



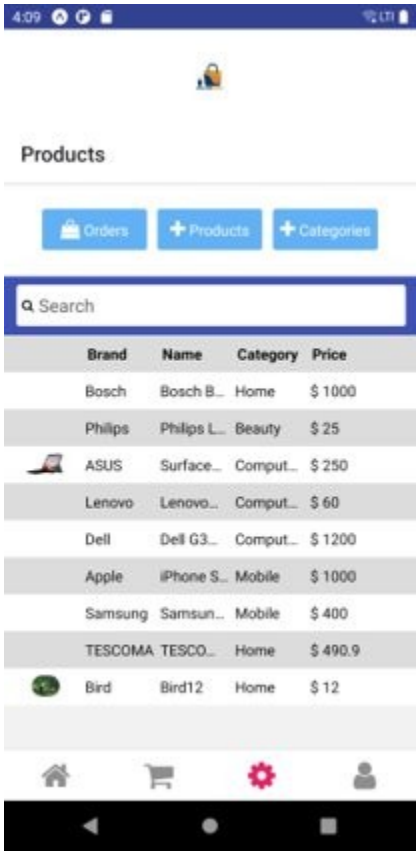
4.5. Korpa



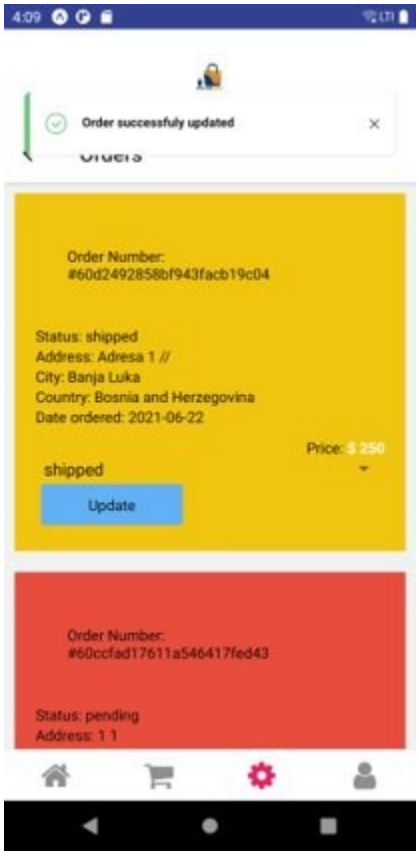
4.6. Checkout


4:09
Checkout
SHIPPING
PAYMENT
CONFIRM
Confirm Order
Shipping to:
Address: Adresa 1
Address 2: //
City: Banja Luka
Zip Code: 78000
Country: Bosnia and Herzegovina
Items:
Surface Pro 7 128GB
$ 250
PLACE ORDER

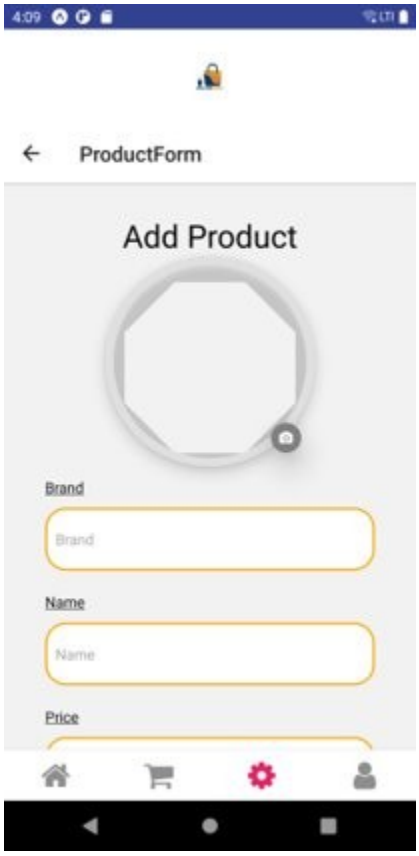
4.7. Admin Panel



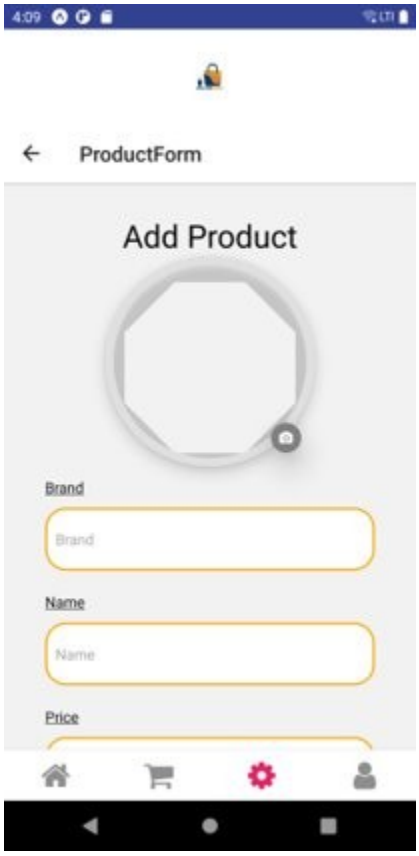
4.8. Narudžbe – Admin strana



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 deploya 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.