Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1-3

за 5 семестр

По дисциплине: «СЯРД»

Тема: «Онлайн-магазин товаров»

Выполнил:

Студент 3 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Войцехович Г.Ю.

2022

Лабораторная работа №1-3

Онлайн-магазин товаров

**Цель работы:** разработать простейшее веб-приложение, использующее базы данных.

Для создания BackEnd использовал:

1. PostgreSQL
2. Sequelize, jsonwebtoken, express, cors, dotenv, uuid, nodemon.

Для создания FontEnd испоьзовал:

1. React, react-dom, react-router-dom, react-scripts, mobx, jwt-decoded, axios, bootstrap.

Запуск сервера backend:

cd server

npm run dev

Запуск клиентской части:

Cd client

npm start

server files:

index.js

require('dotenv').config()

const express = require('express')

const sequelize = require('./db')

const models = require('./models/models')

const cors = require('cors')

const fileUpload = require('express-fileupload')

const router = require('./routes/index')

const errorHandler = require('./middleware/ErrorHandlingMiddleware')

const path = require('path')

const PORT = process.env.PORT || 5000

const app = express()

app.use(cors())

app.use(express.json())

app.use(express.static(path.resolve(\_\_dirname, 'static')))

app.use(fileUpload({}))

app.use('/api', router)

// Обработка ошибок, последний Middleware

app.use(errorHandler)

const start = async () => {

try {

await sequelize.authenticate()

await sequelize.sync()

app.listen(PORT, () => console.log(`Server started on port ${PORT}`))

} catch (e) {

console.log(e)

}

}

start()

db.js

const {Sequelize} = require('sequelize')

module.exports = new Sequelize(

process.env.DB\_NAME, // Название БД

process.env.DB\_USER, // Пользователь

process.env.DB\_PASSWORD, // ПАРОЛЬ

{

dialect: 'postgres',

host: process.env.DB\_HOST,

port: process.env.DB\_PORT

}

)

userController.js

const ApiError = require('../error/ApiError');

const bcrypt = require('bcrypt')

const jwt = require('jsonwebtoken')

const {User, Basket} = require('../models/models')

const generateJwt = (id, email, role) => {

return jwt.sign(

{id, email, role},

process.env.SECRET\_KEY,

{expiresIn: '24h'}

)

}

class UserController {

async registration(req, res, next) {

const {email, password, role} = req.body

if (!email || !password) {

return next(ApiError.badRequest('Некорректный email или password'))

}

const candidate = await User.findOne({where: {email}})

if (candidate) {

return next(ApiError.badRequest('Пользователь с таким email уже существует'))

}

const hashPassword = await bcrypt.hash(password, 5)

const user = await User.create({email, role, password: hashPassword})

const basket = await Basket.create({userId: user.id})

const token = generateJwt(user.id, user.email, user.role)

return res.json({token})

}

async login(req, res, next) {

const {email, password} = req.body

const user = await User.findOne({where: {email}})

if (!user) {

return next(ApiError.internal('Пользователь не найден'))

}

let comparePassword = bcrypt.compareSync(password, user.password)

if (!comparePassword) {

return next(ApiError.internal('Указан неверный пароль'))

}

const token = generateJwt(user.id, user.email, user.role)

return res.json({token})

}

async check(req, res, next) {

const token = generateJwt(req.user.id, req.user.email, req.user.role)

return res.json({token})

}

}

module.exports = new UserController()

typeController.js

const {Type} = require('../models/models')

const ApiError = require('../error/ApiError');

class TypeController {

async create(req, res) {

const {name} = req.body

const type = await Type.create({name})

return res.json(type)

}

async getAll(req, res) {

const types = await Type.findAll()

return res.json(types)

}

}

module.exports = new TypeController()

deviceController.js

const uuid = require('uuid')

const path = require('path');

const {Device, DeviceInfo} = require('../models/models')

const ApiError = require('../error/ApiError');

class DeviceController {

async create(req, res, next) {

try {

let {name, price, brandId, typeId, info} = req.body

const {img} = req.files

let fileName = uuid.v4() + ".jpg"

img.mv(path.resolve(\_\_dirname, '..', 'static', fileName))

const device = await Device.create({name, price, brandId, typeId, img: fileName});

if (info) {

info = JSON.parse(info)

info.forEach(i =>

DeviceInfo.create({

title: i.title,

description: i.description,

deviceId: device.id

})

)

}

return res.json(device)

} catch (e) {

next(ApiError.badRequest(e.message))

}

}

async getAll(req, res) {

let {brandId, typeId, limit, page} = req.query

page = page || 1

limit = limit || 9

let offset = page \* limit - limit

let devices;

if (!brandId && !typeId) {

devices = await Device.findAndCountAll({limit, offset})

}

if (brandId && !typeId) {

devices = await Device.findAndCountAll({where:{brandId}, limit, offset})

}

if (!brandId && typeId) {

devices = await Device.findAndCountAll({where:{typeId}, limit, offset})

}

if (brandId && typeId) {

devices = await Device.findAndCountAll({where:{typeId, brandId}, limit, offset})

}

return res.json(devices)

}

async getOne(req, res) {

const {id} = req.params

const device = await Device.findOne(

{

where: {id},

include: [{model: DeviceInfo, as: 'info'}]

},

)

return res.json(device)

}

}

module.exports = new DeviceController()

brandController.js

const {Brand} = require('../models/models')

const ApiError = require('../error/ApiError');

class BrandController {

async create(req, res) {

const {name} = req.body

const brand = await Brand.create({name})

return res.json(brand)

}

async getAll(req, res) {

const brands = await Brand.findAll()

return res.json(brands)

}

}

module.exports = new BrandController()

ApiErrors.js

class ApiError extends Error{

constructor(status, message) {

super();

this.status = status

this.message = message

}

static badRequest(message) {

return new ApiError(404, message)

}

static internal(message) {

return new ApiError(500, message)

}

static forbidden(message) {

return new ApiError(403, message)

}

}

module.exports = ApiError

authMiddleware.js

const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {

if (req.method === "OPTIONS") {

next()

}

try {

const token = req.headers.authorization.split(' ')[1] // Bearer asfasnfkajsfnjk

if (!token) {

return res.status(401).json({message: "Не авторизован"})

}

const decoded = jwt.verify(token, process.env.SECRET\_KEY)

req.user = decoded

next()

} catch (e) {

res.status(401).json({message: "Не авторизован"})

}

};

checkRoleMiddleware.js

const jwt = require('jsonwebtoken')

module.exports = function(role) {

return function (req, res, next) {

if (req.method === "OPTIONS") {

next()

}

try {

const token = req.headers.authorization.split(' ')[1] // Bearer asfasnfkajsfnjk

if (!token) {

return res.status(401).json({message: "Не авторизован"})

}

const decoded = jwt.verify(token, process.env.SECRET\_KEY)

if (decoded.role !== role) {

return res.status(403).json({message: "Нет доступа"})

}

req.user = decoded;

next()

} catch (e) {

res.status(401).json({message: "Не авторизован"})

}

};

}

errorHandlingMiddleware.js

const ApiError = require('../error/ApiError');

module.exports = function (err, req, res, next) {

if (err instanceof ApiError) {

return res.status(err.status).json({message: err.message})

}

return res.status(500).json({message: "Непредвиденная ошибка!"})

}

Client files:

Routes.js

import Admin from "./pages/Admin";

import {ADMIN\_ROUTE, BASKET\_ROUTE, DEVICE\_ROUTE, LOGIN\_ROUTE, REGISTRATION\_ROUTE, SHOP\_ROUTE} from "./utils/consts";

import Basket from "./pages/Basket";

import Shop from "./pages/Shop";

import Auth from "./pages/Auth";

import DevicePage from "./pages/DevicePage";

export const authRoutes = [

{

path: ADMIN\_ROUTE,

Component: Admin

},

{

path: BASKET\_ROUTE,

Component: Basket

},

]

export const publicRoutes = [

{

path: SHOP\_ROUTE,

Component: Shop

},

{

path: LOGIN\_ROUTE,

Component: Auth

},

{

path: REGISTRATION\_ROUTE,

Component: Auth

},

{

path: DEVICE\_ROUTE + '/:id',

Component: DevicePage

},

]

Index.js

import React, {createContext} from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import UserStore from "./store/UserStore";

import DeviceStore from "./store/DeviceStore";

export const Context = createContext(null)

ReactDOM.render(

<Context.Provider value={{

user: new UserStore(),

device: new DeviceStore(),

}}>

<App />

</Context.Provider>,

document.getElementById('root')

);

App.js

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

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

import AppRouter from "./components/AppRouter";

import NavBar from "./components/NavBar";

import {observer} from "mobx-react-lite";

import {Context} from "./index";

import {check} from "./http/userAPI";

import {Spinner} from "react-bootstrap";

const App = observer(() => {

const {user} = useContext(Context)

const [loading, setLoading] = useState(true)

useEffect(() => {

check().then(data => {

user.setUser(true)

user.setIsAuth(true)

}).finally(() => setLoading(false))

}, [])

if (loading) {

return <Spinner animation={"grow"}/>

}

return (

<BrowserRouter>

<NavBar />

<AppRouter />

</BrowserRouter>

);

});

export default App;

consts.js

export const ADMIN\_ROUTE = '/admin'

export const LOGIN\_ROUTE = '/login'

export const REGISTRATION\_ROUTE = '/registration'

export const SHOP\_ROUTE = '/'

export const BASKET\_ROUTE = '/basket'

export const DEVICE\_ROUTE = '/device'

Реализацию каждой страницы опустил.

AppRouter.js

import React, {useContext} from 'react';

import {Switch, Route, Redirect} from 'react-router-dom'

import {authRoutes, publicRoutes} from "../routes";

import {SHOP\_ROUTE} from "../utils/consts";

import {Context} from "../index";

import {observer} from "mobx-react-lite";

const AppRouter = observer(() => {

const {user} = useContext(Context)

console.log(user)

return (

<Switch>

{user.isAuth && authRoutes.map(({path, Component}) =>

<Route key={path} path={path} component={Component} exact/>

)}

{publicRoutes.map(({path, Component}) =>

<Route key={path} path={path} component={Component} exact/>

)}

<Redirect to={SHOP\_ROUTE}/>

</Switch>

);

});

export default AppRouter;

CreateBrand.js

import React, {useState} from 'react';

import Modal from "react-bootstrap/Modal";

import {Button, Form} from "react-bootstrap";

import {createBrand, createType} from "../../http/deviceAPI";

const CreateBrand = ({show, onHide}) => {

const [value, setValue] = useState('')

const addBrand = () => {

createBrand({name: value}).then(data => {

setValue('')

onHide()

})

}

return (

<Modal

show={show}

onHide={onHide}

centered

>

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-vcenter">

Добавить тип

</Modal.Title>

</Modal.Header>

<Modal.Body>

<Form>

<Form.Control

value={value}

onChange={e => setValue(e.target.value)}

placeholder={"Введите название типа"}

/>

</Form>

</Modal.Body>

<Modal.Footer>

<Button variant="outline-danger" onClick={onHide}>Закрыть</Button>

<Button variant="outline-success" onClick={addBrand}>Добавить</Button>

</Modal.Footer>

</Modal>

);

};

export default CreateBrand;

CreateDevice.js

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

import Modal from "react-bootstrap/Modal";

import {Button, Dropdown, Form, Row, Col} from "react-bootstrap";

import {Context} from "../../index";

import {createDevice, fetchBrands, fetchDevices, fetchTypes} from "../../http/deviceAPI";

import {observer} from "mobx-react-lite";

const CreateDevice = observer(({show, onHide}) => {

const {device} = useContext(Context)

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

const [price, setPrice] = useState(0)

const [file, setFile] = useState(null)

const [info, setInfo] = useState([])

useEffect(() => {

fetchTypes().then(data => device.setTypes(data))

fetchBrands().then(data => device.setBrands(data))

}, [])

const addInfo = () => {

setInfo([...info, {title: '', description: '', number: Date.now()}])

}

const removeInfo = (number) => {

setInfo(info.filter(i => i.number !== number))

}

const changeInfo = (key, value, number) => {

setInfo(info.map(i => i.number === number ? {...i, [key]: value} : i))

}

const selectFile = e => {

setFile(e.target.files[0])

}

const addDevice = () => {

const formData = new FormData()

formData.append('name', name)

formData.append('price', `${price}`)

formData.append('img', file)

formData.append('brandId', device.selectedBrand.id)

formData.append('typeId', device.selectedType.id)

formData.append('info', JSON.stringify(info))

createDevice(formData).then(data => onHide())

}

return (

<Modal

show={show}

onHide={onHide}

centered

>

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-vcenter">

Добавить устройство

</Modal.Title>

</Modal.Header>

<Modal.Body>

<Form>

<Dropdown className="mt-2 mb-2">

<Dropdown.Toggle>{device.selectedType.name || "Выберите тип"}</Dropdown.Toggle>

<Dropdown.Menu>

{device.types.map(type =>

<Dropdown.Item

onClick={() => device.setSelectedType(type)}

key={type.id}

>

{type.name}

</Dropdown.Item>

)}

</Dropdown.Menu>

</Dropdown>

<Dropdown className="mt-2 mb-2">

<Dropdown.Toggle>{device.selectedBrand.name || "Выберите тип"}</Dropdown.Toggle>

<Dropdown.Menu>

{device.brands.map(brand =>

<Dropdown.Item

onClick={() => device.setSelectedBrand(brand)}

key={brand.id}

>

{brand.name}

</Dropdown.Item>

)}

</Dropdown.Menu>

</Dropdown>

<Form.Control

value={name}

onChange={e => setName(e.target.value)}

className="mt-3"

placeholder="Введите название устройства"

/>

<Form.Control

value={price}

onChange={e => setPrice(Number(e.target.value))}

className="mt-3"

placeholder="Введите стоимость устройства"

type="number"

/>

<Form.Control

className="mt-3"

type="file"

onChange={selectFile}

/>

<hr/>

<Button

variant={"outline-dark"}

onClick={addInfo}

>

Добавить новое свойство

</Button>

{info.map(i =>

<Row className="mt-4" key={i.number}>

<Col md={4}>

<Form.Control

value={i.title}

onChange={(e) => changeInfo('title', e.target.value, i.number)}

placeholder="Введите название свойства"

/>

</Col>

<Col md={4}>

<Form.Control

value={i.description}

onChange={(e) => changeInfo('description', e.target.value, i.number)}

placeholder="Введите описание свойства"

/>

</Col>

<Col md={4}>

<Button

onClick={() => removeInfo(i.number)}

variant={"outline-danger"}

>

Удалить

</Button>

</Col>

</Row>

)}

</Form>

</Modal.Body>

<Modal.Footer>

<Button variant="outline-danger" onClick={onHide}>Закрыть</Button>

<Button variant="outline-success" onClick={addDevice}>Добавить</Button>

</Modal.Footer>

</Modal>

);

});

export default CreateDevice;

CreateType.js

import React, {useState} from 'react';

import Modal from "react-bootstrap/Modal";

import {Form, Button} from "react-bootstrap";

import {createType} from "../../http/deviceAPI";

const CreateType = ({show, onHide}) => {

const [value, setValue] = useState('')

const addType = () => {

createType({name: value}).then(data => {

setValue('')

onHide()

})

}

return (

<Modal

show={show}

onHide={onHide}

centered

>

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-vcenter">

Добавить тип

</Modal.Title>

</Modal.Header>

<Modal.Body>

<Form>

<Form.Control

value={value}

onChange={e => setValue(e.target.value)}

placeholder={"Введите название типа"}

/>

</Form>

</Modal.Body>

<Modal.Footer>

<Button variant="outline-danger" onClick={onHide}>Закрыть</Button>

<Button variant="outline-success" onClick={addType}>Добавить</Button>

</Modal.Footer>

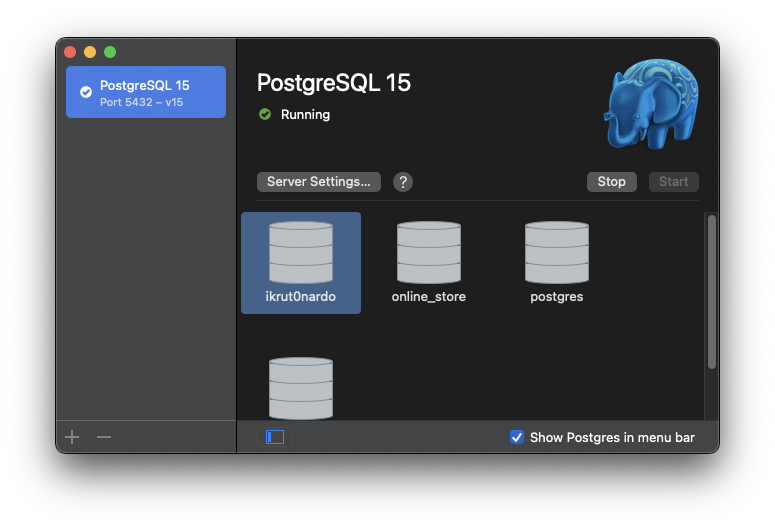
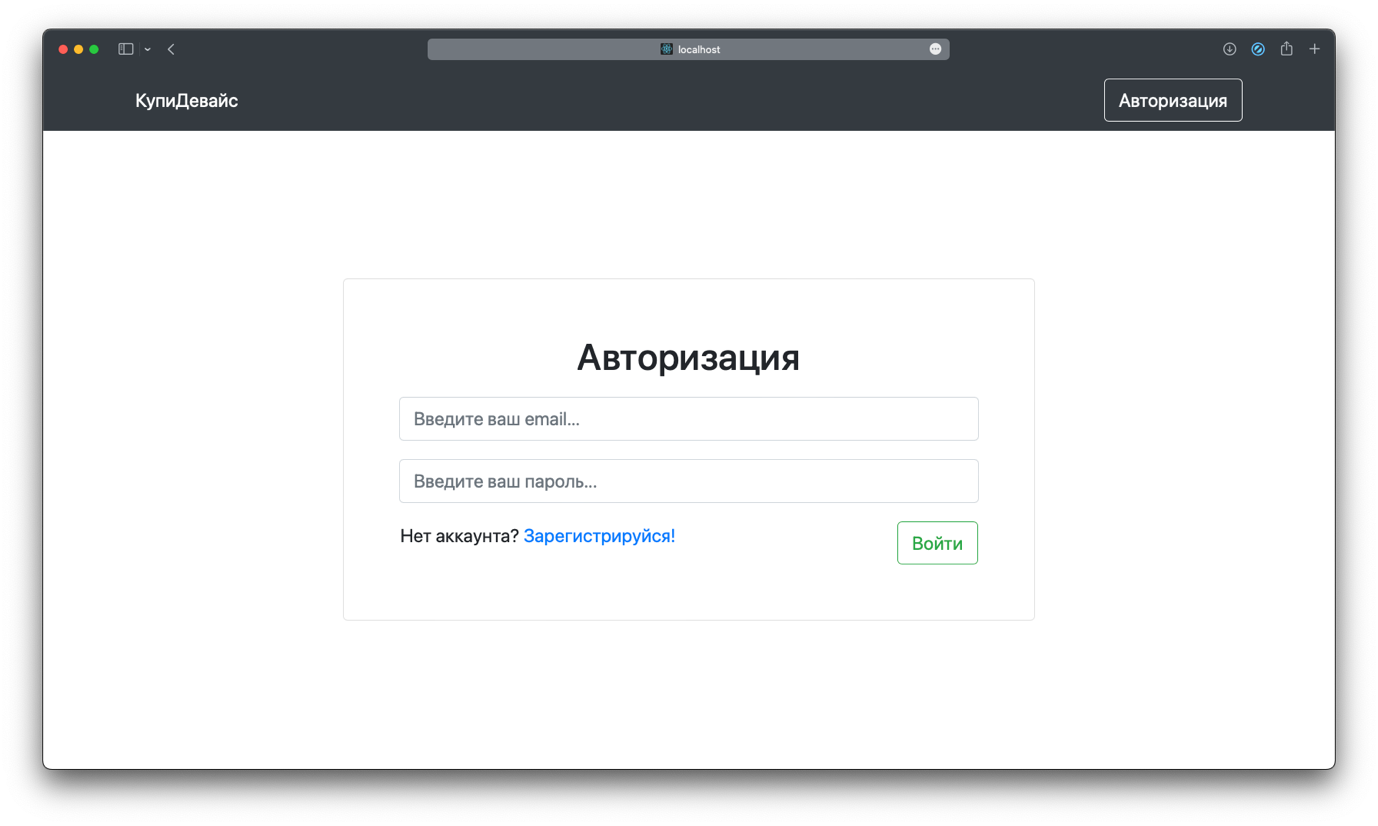
</Modal>

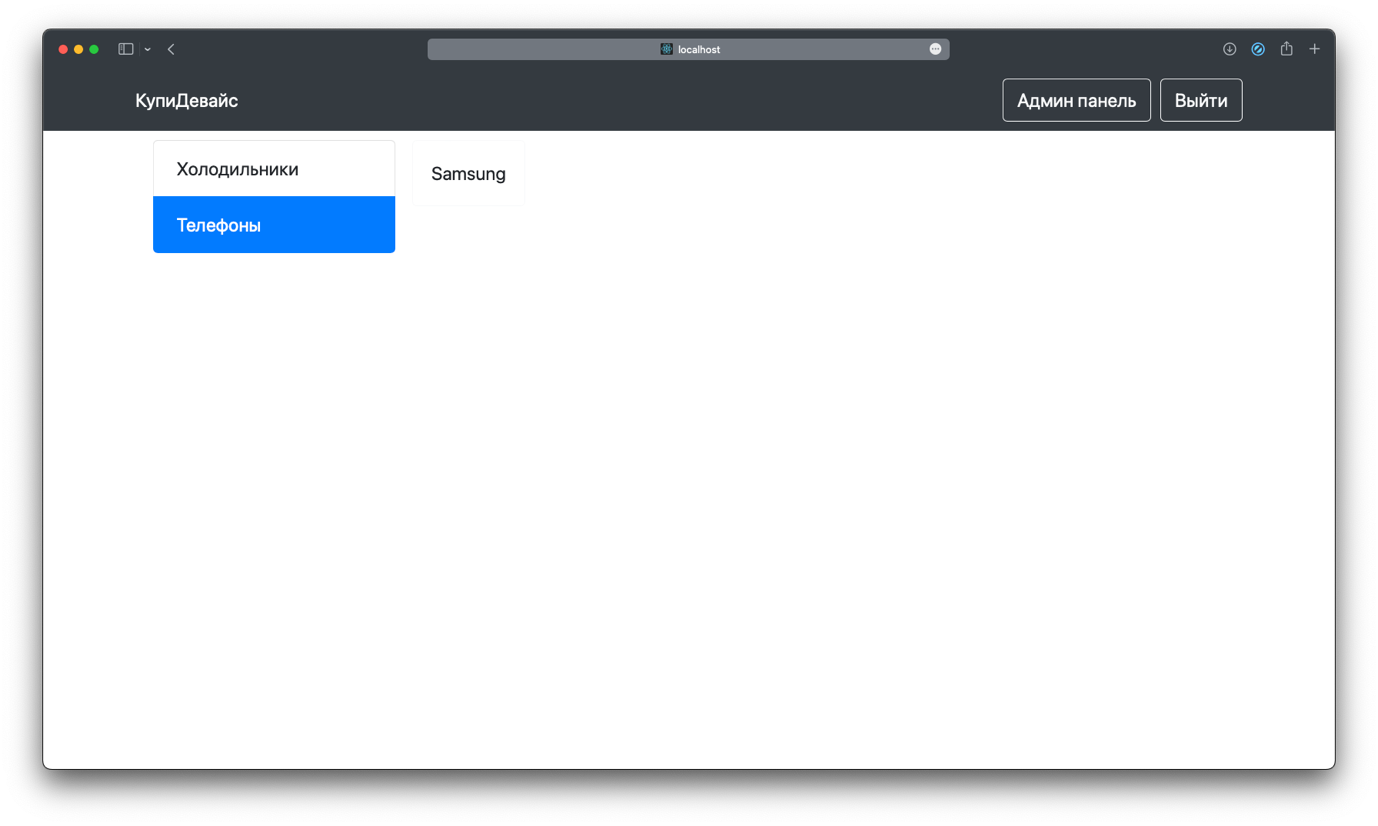
);

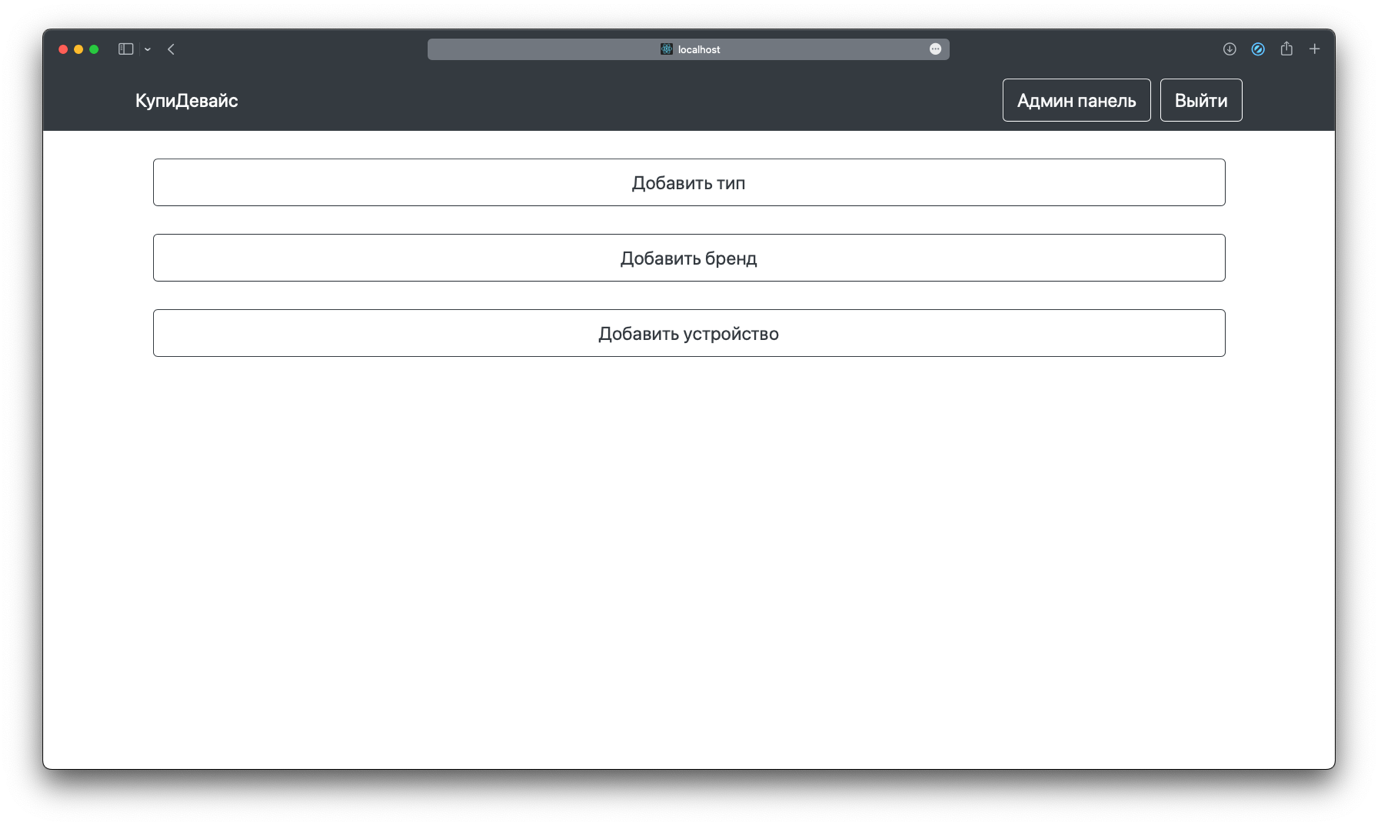
};

export default CreateType;

Графический материал:







Вывод: разработал простейшее веб-приложение, использующее базы данных.