**Поволжский Государственный Университет Телекоммуникаций и Информатики**Кафедра «Программная инженерия»

|  |  |
| --- | --- |
| Сдана на проверку  «\_\_» \_\_\_\_2023 г. | Допустить к защите  «\_\_» \_\_\_\_\_ 2023 г.  Защищена с оценкой  «\_\_\_» \_\_\_\_\_\_\_\_ 2023 г. |

**КУРСОВАЯ РАБОТА**По дисциплине: «Прикладное программирование»

На тему: «Разработка клиент-серверного web-приложения Книжный магазин»  
  
Пояснительная записка

|  |  |
| --- | --- |
| Студент группы ИВТ-13\_\_\_\_\_\_\_ Фомин А. В.  (роспись) (ФИО)  Руководитель \_\_\_\_\_\_\_\_ к.т.н., доц. Ахметшина Э. Г.  (роспись) (ФИО) | 0 210203 0  (№ зачетной книжки) |

Самара 2023

**Рецензия**

**Содержание**

[Описание предметной области. Актуальность 4](#_Toc90735856)

[Описание программы. Общие сведения 5](#_Toc90735857)

[Функциональное назначение 5](#_Toc90735858)

[Логическая модель базы данных 6](#_Toc90735859)

[Физическая модель базы данных 9](#_Toc90735860)

[Диаграмма классов 11](#_Toc90735861)

[Диаграмма компонентов 14](#_Toc90735862)

[Диаграмма вариантов использования 14](#_Toc90735863)

[Диаграмма последовательности 15](#_Toc90735864)

[Демонстрация работы приложения 16](#_Toc90735865)

[Список используемых источников 25](#_Toc90735866)

[Приложение А- Листинг программного кода 26](#_Toc90735867)

# **Описание предметной области. Актуальность**

Книжный магазин - это учреждение, специализирующееся на продаже книг. В настоящее время, несмотря на развитие цифровых технологий, книги остаются востребованным продуктом, и многие люди предпочитают покупать их в специализированных магазинах. Книжный магазин предлагает широкий ассортимент книг разных жанров и авторов.

Актуальность разработки клиент-серверного web-приложения для книжного магазина заключается в следующем:

1. Онлайн-продажи. В настоящее время многие люди предпочитают делать покупки через интернет. Клиент-серверное web-приложение позволит книжному магазину предлагать свои товары онлайн, что увеличит его охват аудитории и потенциально привлечет больше клиентов.

2. Удобство и доступность. Web-приложение позволит пользователям покупать книги в любое время и из любого места с использованием устройств с доступом в интернет. Они смогут просматривать каталог книг, ознакомиться с их описанием и ценами, делать заказы.

3. Удобное управление ассортиментом и инвентаризация. Web-приложение позволит книжному магазину управлять и отслеживать свой ассортимент более эффективно. Магазин сможет обновлять информацию о новых книгах, управлять их наличием на складе и автоматически обновлять данные о доступных экземплярах на сайте. Это поможет избежать ситуации, когда клиент делает заказ на книгу, которую уже нет в наличии, что повысит удовлетворенность клиента и сократит возможные проблемы с возвратами или рекламациями.

4. Возможность расширить бизнес. Онлайн-продажи через web-приложение позволят книжному магазину выйти на новые рынки и расширить свою деятельность за пределы физического магазина. Это повысит его конкурентоспособность и может привести к дополнительной прибыли.

Таким образом разработка клиент-серверного web-приложения для книжного магазина является актуальной и востребованной задачей, которая позволит сделать покупку книг более удобной и доступной для клиентов, позволит расширить потенциал и возможности магазина, улучшить обслуживание клиентов, упростить процессы управления и контроля, а также привлечь новых пользователей и увеличить прибыль.

# **Описание программы. Общие сведения**

Приложение – веб-ресурс. Установка не требуется.

Необходимые требования: доступ к интернету и клиент для запуска(браузер).

Язык программирования: JavaScript

Платформа: Express.

Среда разработки: Visual Studio Code.

Объем проекта: 731,7 МБ (вместе с исходным кодом).

Исходный код: 1220 строк.

# **Функциональное назначение**

Функциональное назначение клиент-серверного web-приложения для книжного магазина включает в себя следующие основные функции:

1. Регистрация и авторизация пользователей:

- Регистрация новых пользователей путем заполнения формы с указанием личных данных, таких как имя, электронная почта и пароль;

- Авторизация существующих пользователей с использованием учетных данных.

2. Просмотр каталога книг:

- Отображение списка доступных книг с их названиями, авторами, ценами и описанием;

3. Поиск книг:

- Расширенный поиск книг по различным жанрам;

- Результаты поиска должны быть представлены в виде списка книг с их основной информацией.

4. Детальная информация о книге:

- Отображение подробной информации о выбранной книге, описание, автор, год издания, цена;

- Предоставление возможности добавления книги в корзину для покупки.

5. Управление корзиной:

- Просмотр выбранной книги в корзине для дальнейшего оформления заказа;

6. Администраторские возможности:

- Управление каталогом книг, включая добавление, редактирование и удаление данных о книгах,

STACK Технологий для BACKEND:

1. Node.js - это среда выполнения JavaScript, которая позволяет запускать JavaScript на сервере, а не только в браузере.
2. Express - это веб-фреймворк для Node.js, предоставляющий набор инструментов для обработки маршрутизации, запросов, обработку ошибок и управление сеансами, поддержка и запуска сервера.
3. PostgreSQL – это расширяемая система управления реляционными базами данных, которая используется для хранения и обработки структурированных данных.
4. Sequelize – это это ORM для Node.js, который позволяет разработчикам работать с реляционными базами данных с помощью объектно-ориентированной модели данных.

STACK Технологий для FRONTEND:

1. React.js - это JavaScript-библиотека для разработки пользовательского интерфейса (UI).
2. React bootstrap - это набор готовых компонентов и стилей, основанный на CSS-фреймворке Bootstrap.
3. Axios - это библиотека для отправки HTTP-запросов с помощью JavaScript на сервер.
4. React-router-dom - это набор инструментов для реализации маршрутизации в приложениях React.js.
5. MobX - это библиотека для управления состоянием приложений.

# **Логическая модель базы данных**

Логическая модель базы данных для книжного магазина состоит из пяти сущностей: Клиент (user), Корзина (basket), Книга (book), Автор (author), Жанр (genre).

|  |  |
| --- | --- |
| Название | Назначение |
| User | Описывает пользователей. Содержит атрибуты: ID(идентификатор), email, password, role(роль пользователя, admin or public). Связана связью “1 к 1” с сущностью “Basket”. |
| Basket | Описывает корзину пользователя. Содержит атрибуты: ID(идентификатор), user\_ID(идентификатор пользователя), book\_ID(идентификатор книги). Связана связью “1 к 1” с сущностью “User” и связью “1 к 1” с сущностью “Book”. |
| Book | Описывает книгу. Содержит атрибуты: ID(идентификатор), name(название книги), date(дата выпуска), price(цена), description(Аннотация), genre\_ID(идентификатор жанра книги), author\_ID(идентификатор автора книги). Связана связью “1 к 1” с сущностью “Basket”, связью “1 к M” с сущность “Author” и связью “1 к M” с сущность “Genre”. |
| Author | Описывает автора книг. Содержит атрибуты: ID(идентификатор), name(Имя), last\_name(Фамилия). Связана связью “1 к M” с сущностью “Book”. |
| Genre | Описывает жанр книг. Содержит атрибуты: ID(идентификатор), name(Название). Связана связью “1 к M” с сущностью “Book”. |

Эта логическая модель описывает основные сущности и их атрибуты в базе данных для книжного магазина. Пользователи могут добавлять книги в корзину, просматривать информацию о книгах. Каждая книга имеет своего автора и принадлежит определенному жанру. Цена книги указывается для определения стоимости покупки. Все данные связаны с помощью идентификаторов, что облегчает поиск и управление информацией о книгах в базе данных.

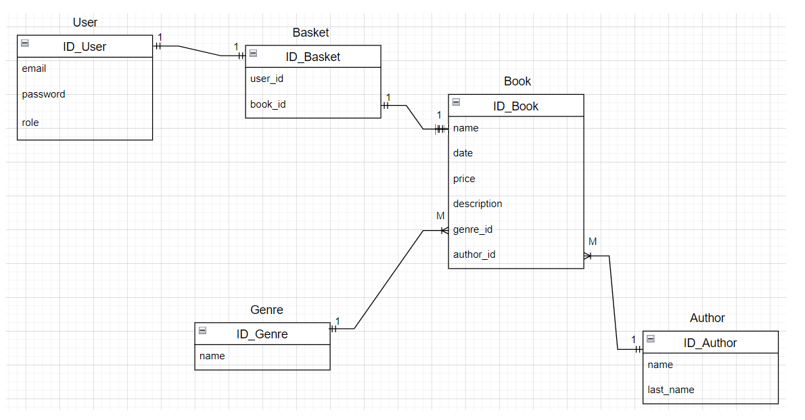


Рис 1. Логическая модель БД

# **Физическая модель базы данных**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя поля | Тип данных | Тип ключа |
| 1 | ID\_User | INT(5) | Первичный |
| 2 | email | VARCHAR(50) |  |
| 3 | password | VARCHAR(12) |  |
| 4 | role | VARCHAR(7) |  |

User

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя поля | Тип данных | Тип ключа |
| 1 | ID\_Basket | INT(5) | Первичный |
| 2 | user\_ID | INT(5) | Внешний |
| 3 | book\_ID | INT(5) | Внешний |

Basket

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя поля | Тип данных | Тип ключа |
| 1 | ID\_Book | INT(5) | Первичный |
| 2 | name | VARCHAR(20) |  |
| 3 | date | DATE(10) |  |
| 4 | price | INT(7) |  |
| 5 | description | VARCHAR(50) |  |
| 6 | genre\_ID | INT(5) | Внешний |
| 7 | author\_ID | INT(5) | Внешний |

Book

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя поля | Тип данных | Тип ключа |
| 1 | ID\_Genre | INT(5) | Первичный |
| 2 | name | VARCHAR(30) |  |

Genre

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя поля | Тип данных | Тип ключа |
| 1 | ID\_Author | INT(5) | Первичный |
| 2 | name | VARCHAR(30) |  |
| 3 | last\_name | VARCHAR(30) |  |

Author

Таблица 5

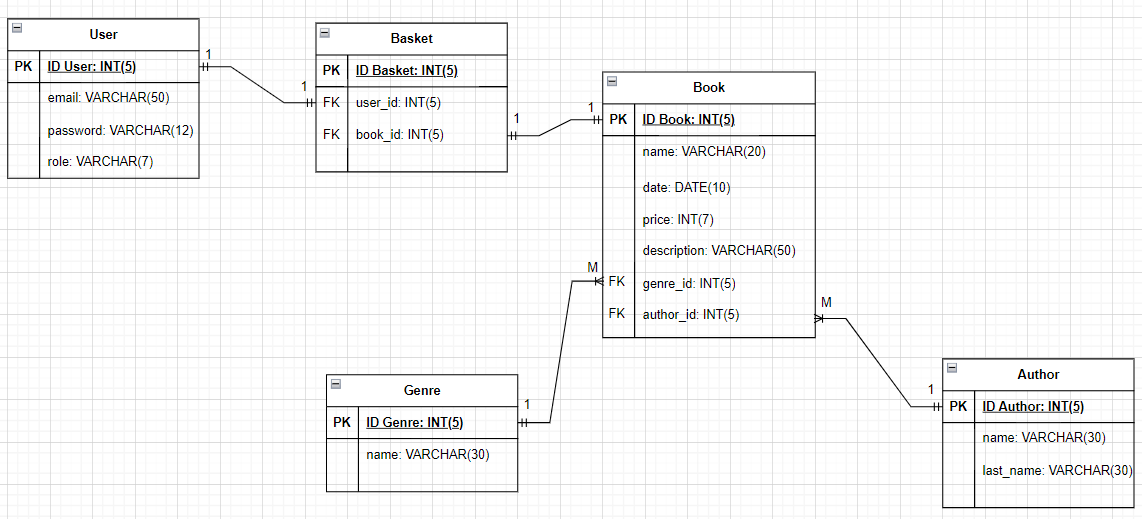


Рис. 2. Физическая ER-диаграмма БД

# **Диаграмма классов**

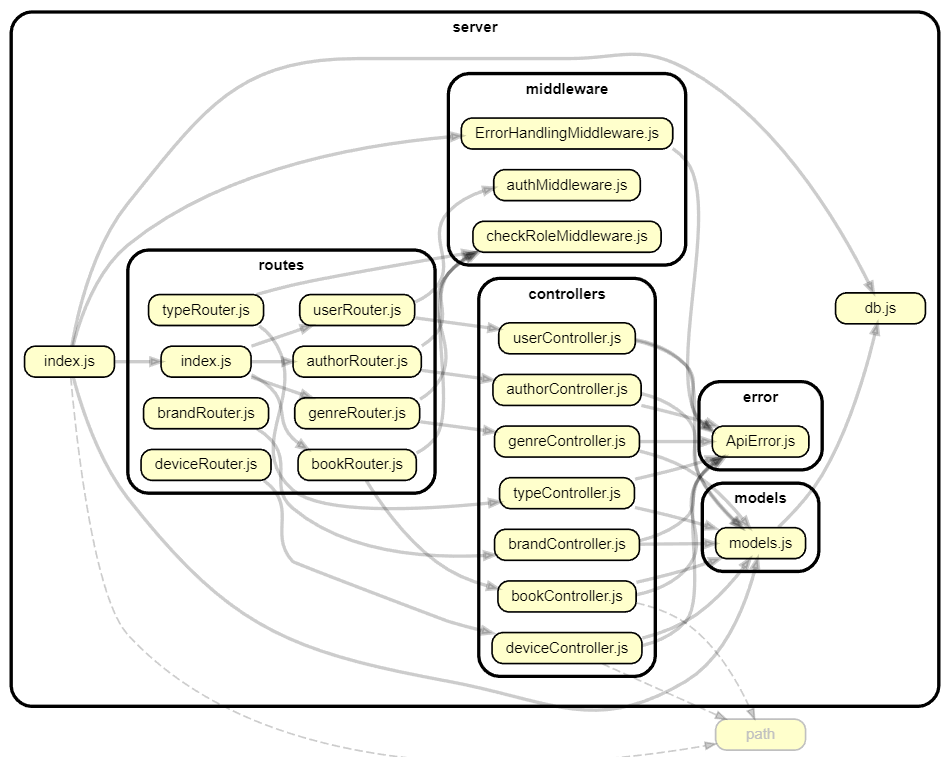
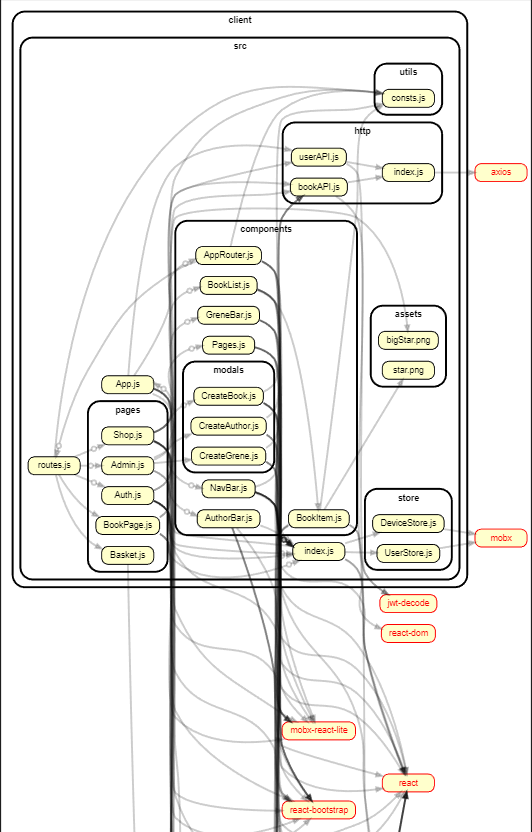
п

Рис.3 Диаграмма классов для сервера



# 

# Рис. 4 Диаграмма классов для client

# **Диаграмма компонентов**

Рис. 5 Диаграмма компонентов

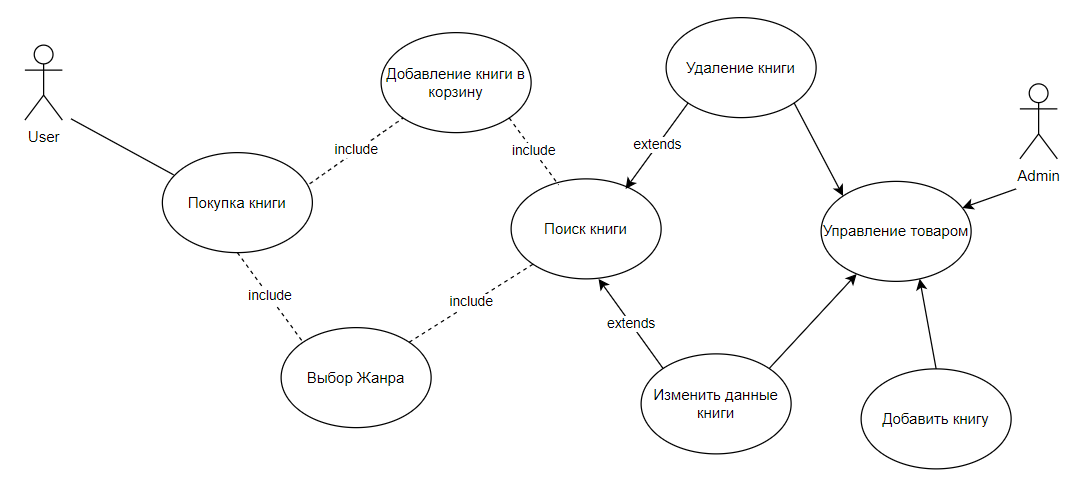
**Диаграмма вариантов использования**

Рис.6 Диаграмма вариантов использования

# **Диаграмма последовательности**

# 

Рис.7 Диаграмма последовательности

# 

# **Демонстрация работы приложения**

Рис.8 Страница авторизации

Рис.9 Страница магазина от лица пользователя



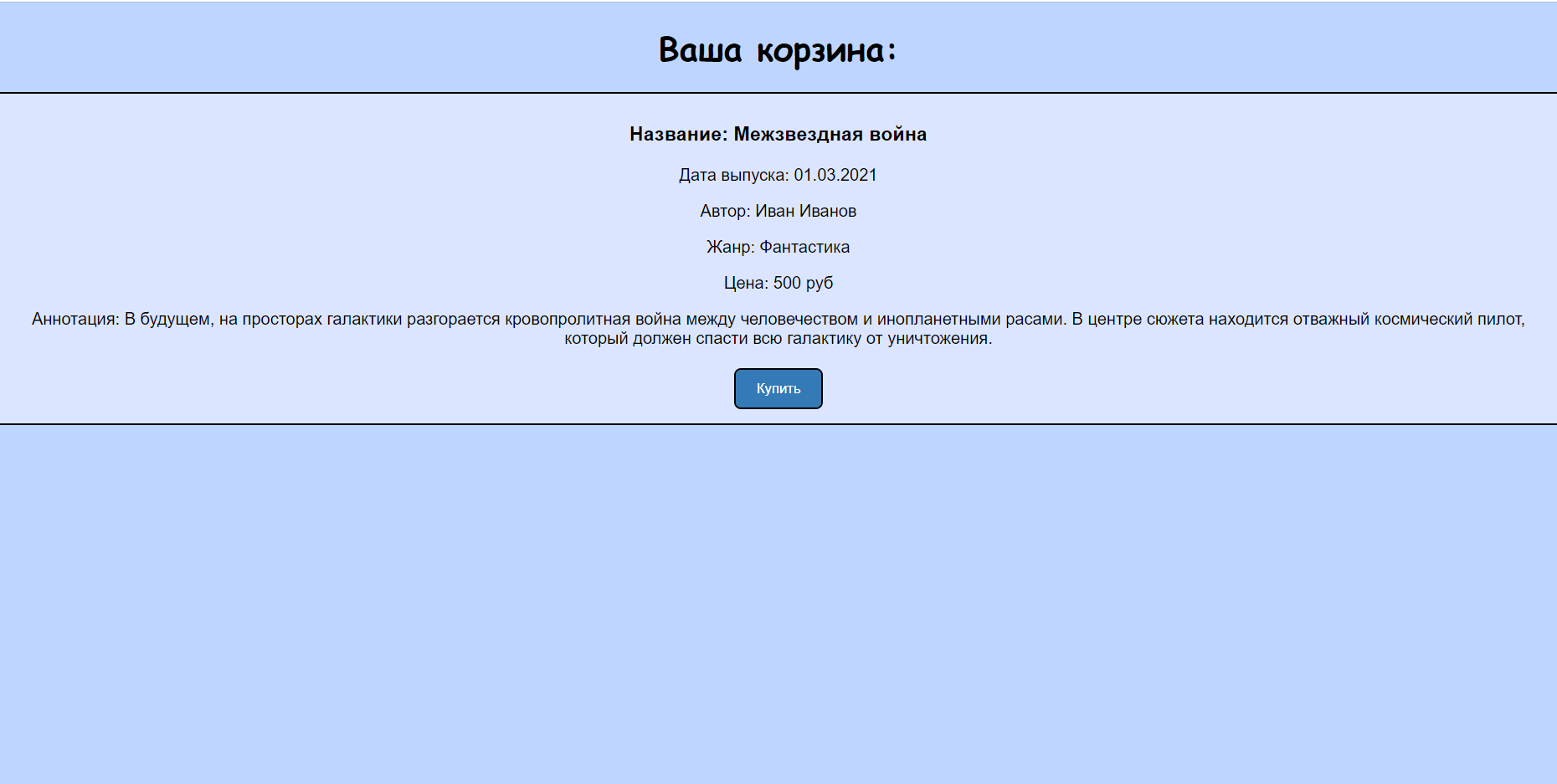
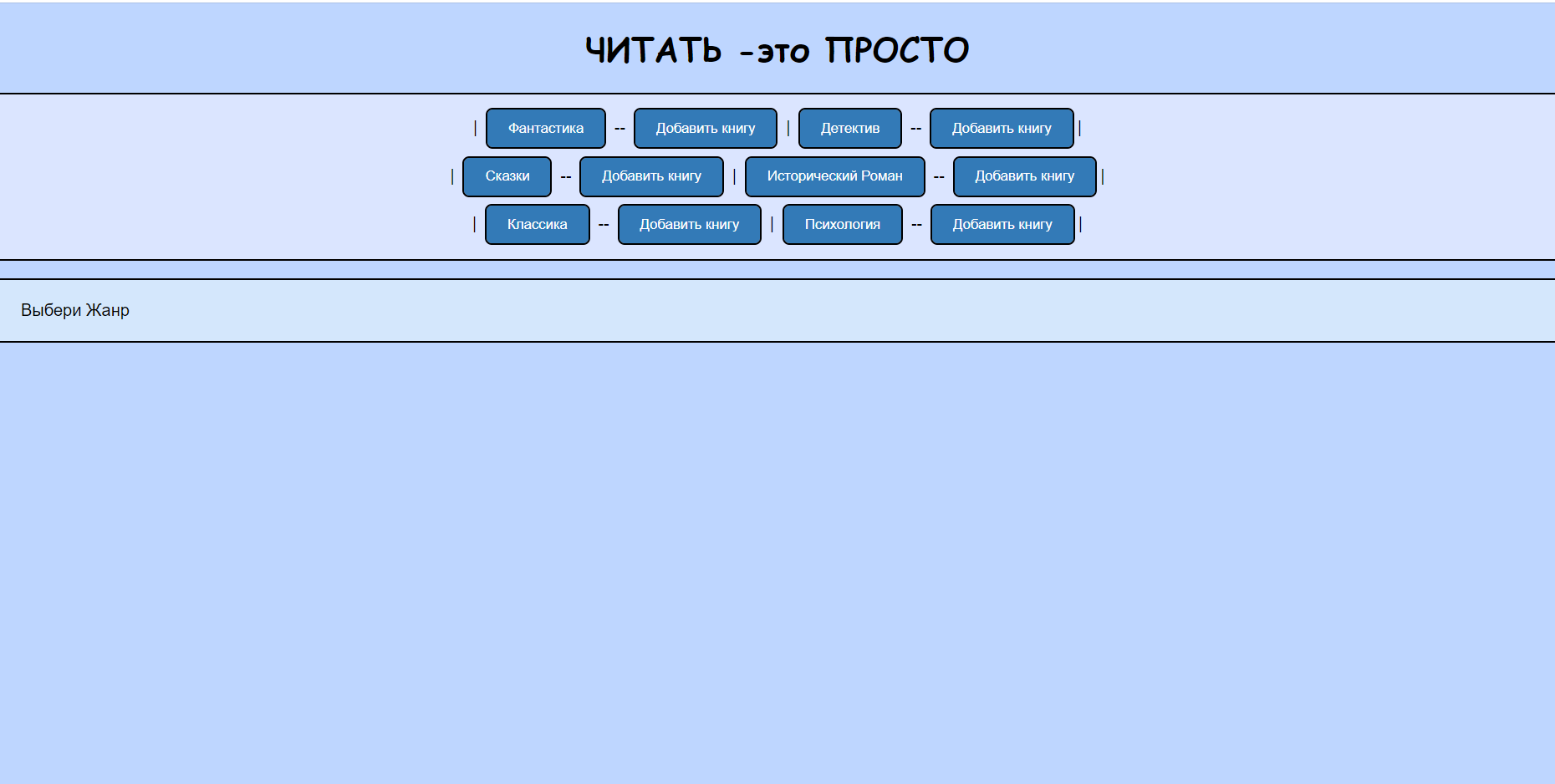
 Рис.10 При нажатии на кнопку ‘купить’ попадаем на страницу корзины

Рис.11 Стариница корзины



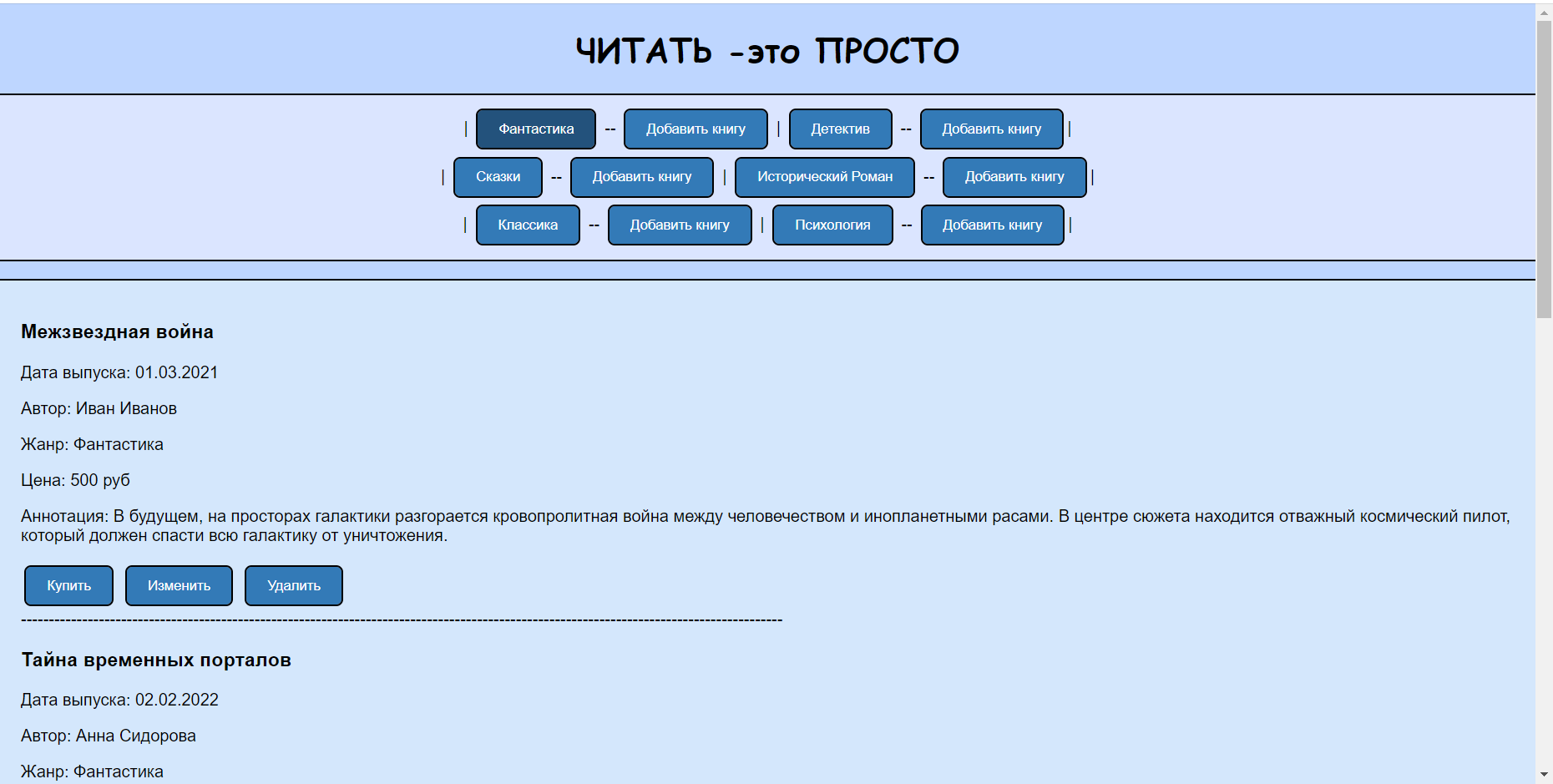
Рис.12 Страница магазина со стороны админа

 Рис.13

Рис.14 Добавление книги(Нажатие на кнопку)

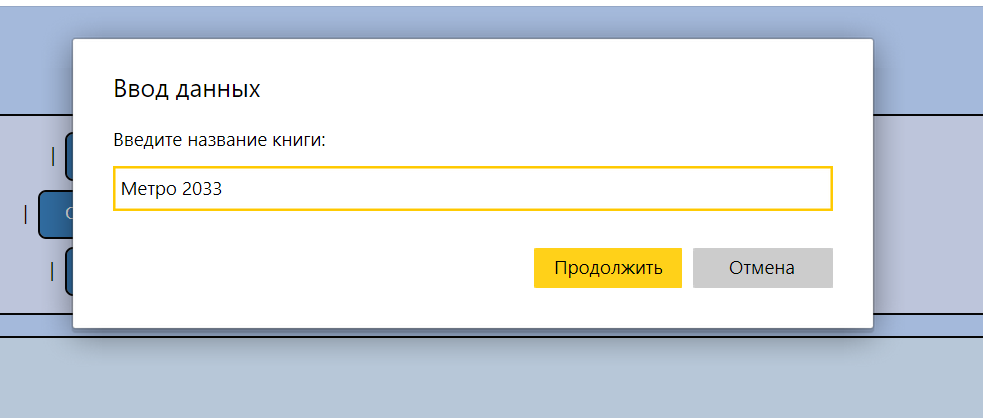


Рис.15 Ввод названия книги

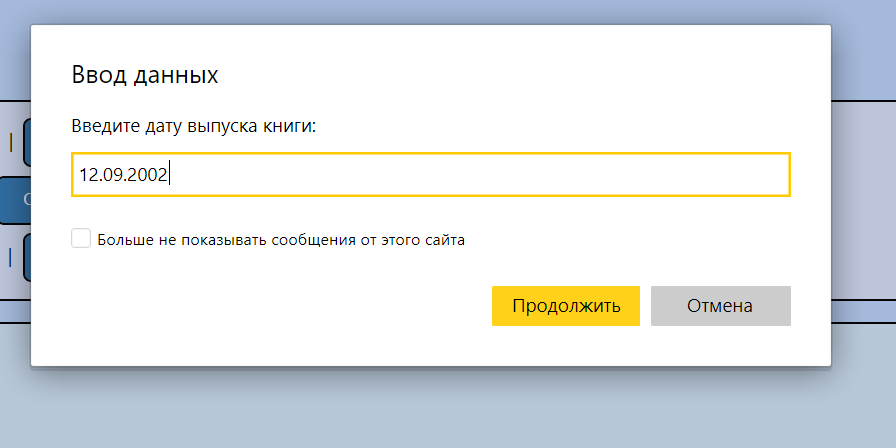


Рис.16 Ввод даты выпуска книги

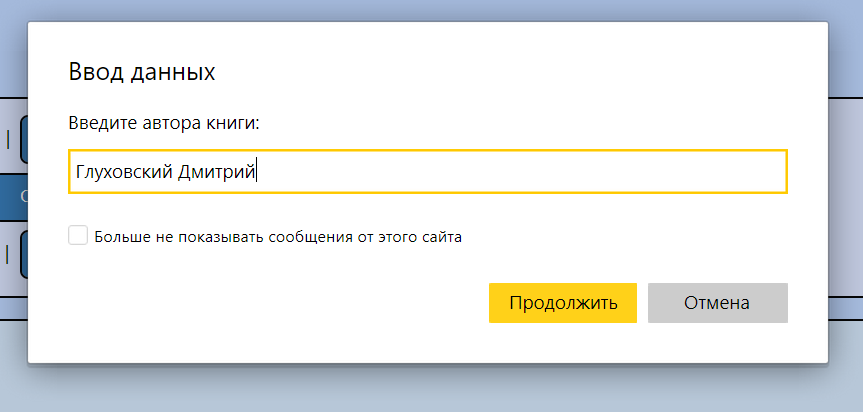


Рис.17 Ввод автора книги

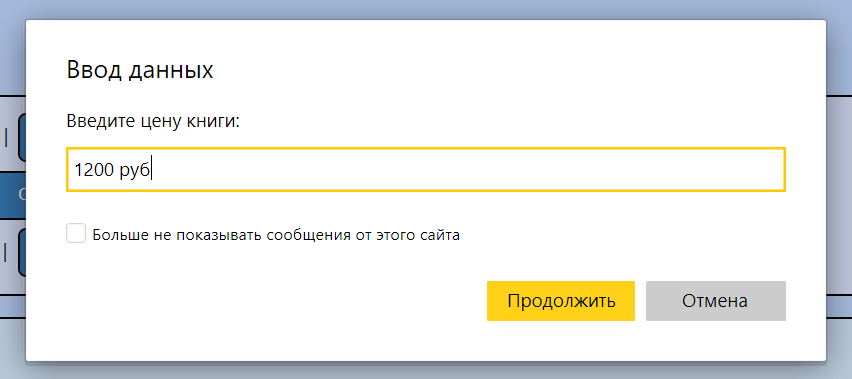


Рис.18 Ввод цены на книгу

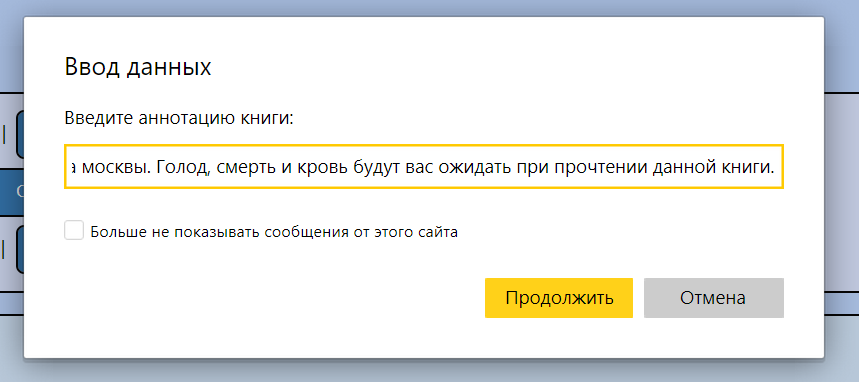


Рис.19 Ввод аннотации на книгу

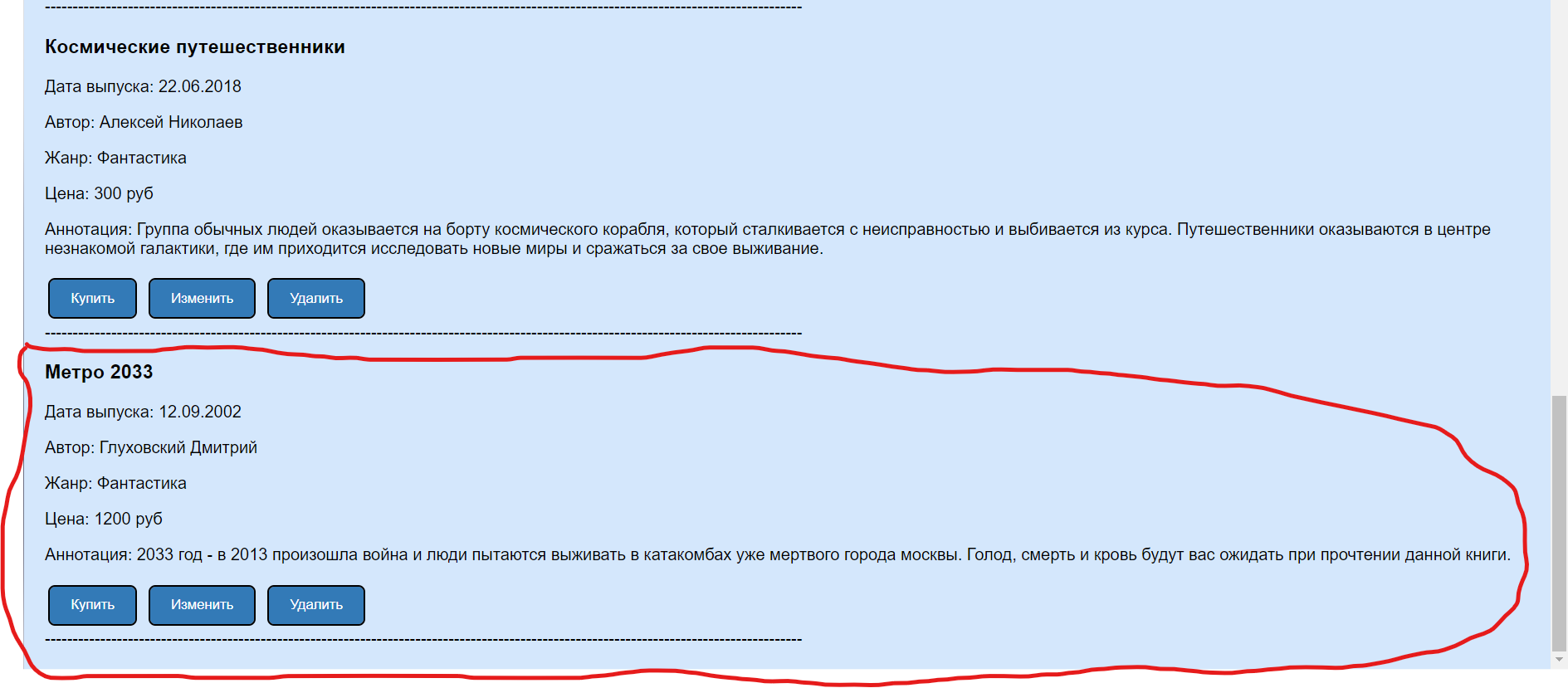


Рис.20 Книга добавленна в список(фантастика)



Рис.20 Список книг(фантастика)

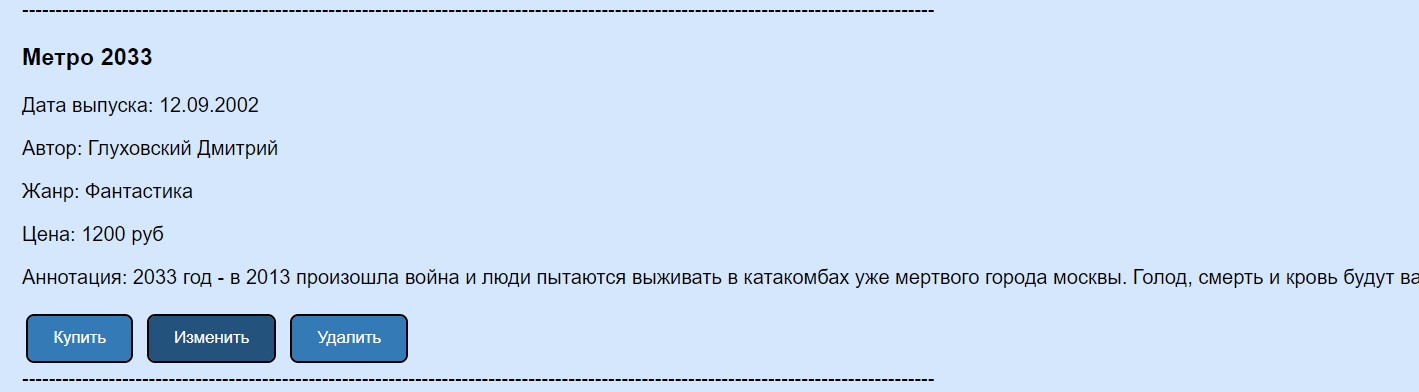


Рис.21 Изменение книги (нажатие на кнопку)

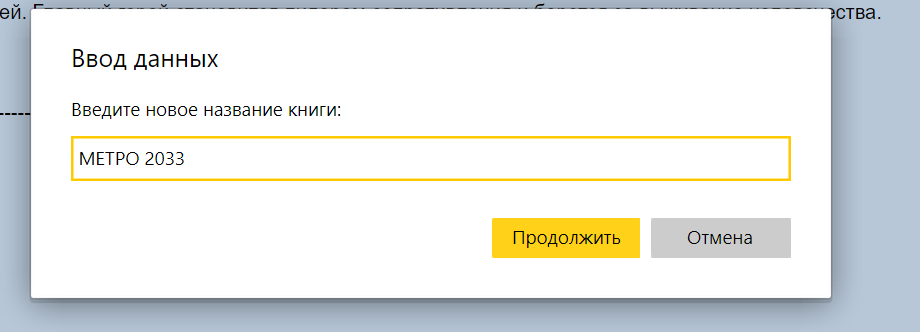


Рис.22 Изменение название книги

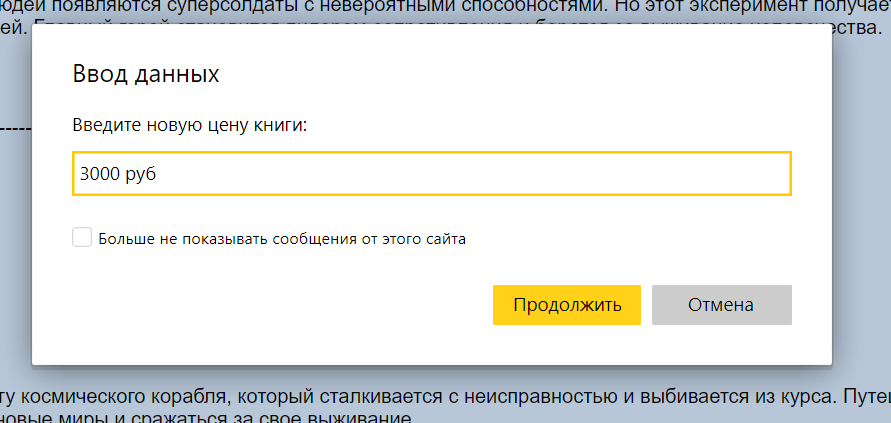


Рис.22 Изменение цены книги

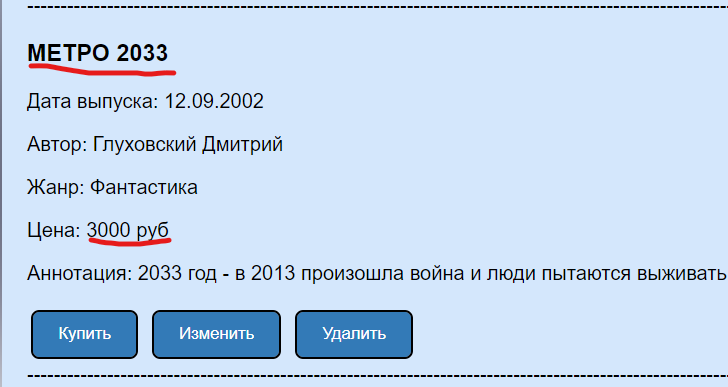


Рис.23 Изменение книга



Рис.24 Удаление книги(нажатие на кнопку)

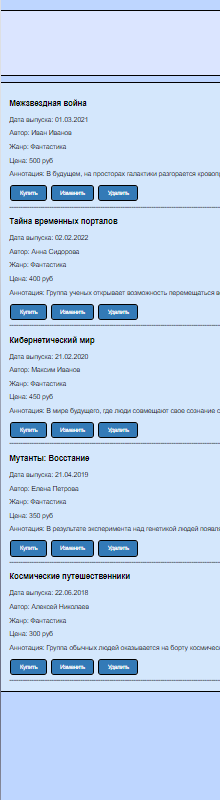
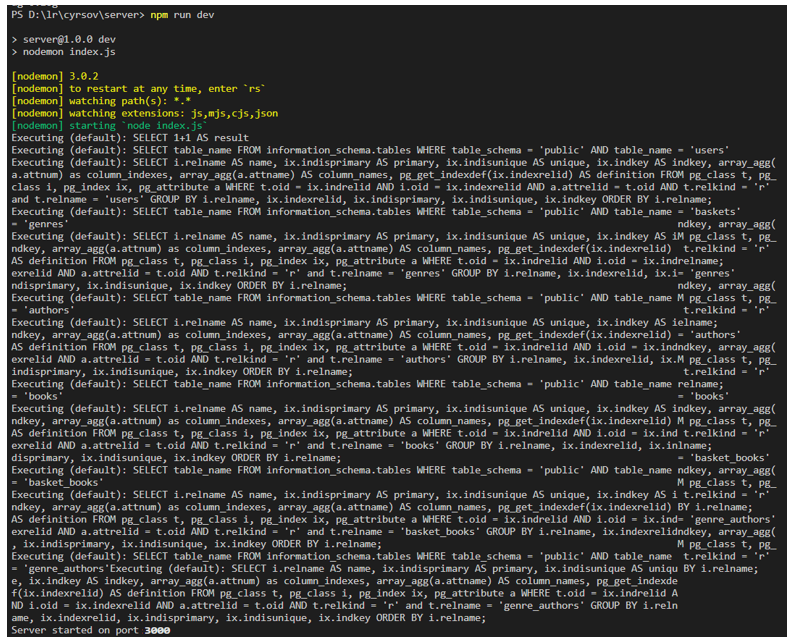


Рис.25 Книга удалена



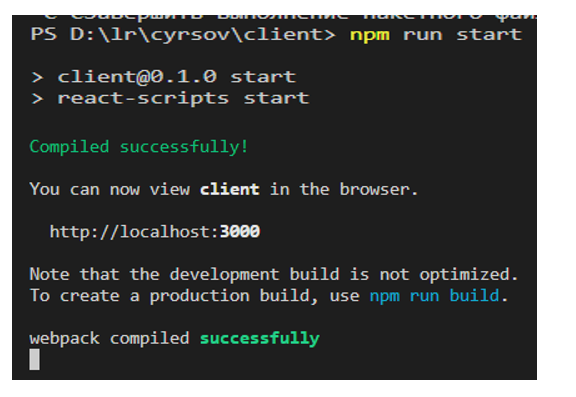
Рис.26 Запуска сервера с инициализацией DB

Рис.27 Запуска клиента на порту(3000)

# **Список используемых источников**

1. "Node.js in Action" - авторы: Mike Cantelon, Marc Harter, TJ Holowaychuk, Nathan Rajlich; издательство: Manning Publications; город: Нью-Йорк; год: 2013; число страниц: 416. Лия Веру - "Секреты CSS. Идеальные решения ежедневных задач", Питер, 2017, 336 стр. (ориг. название: "CSS Secrets. Better solutions to everyday web design problems", O'Reilly)
2. "Learning React: Functional Web Development with React and Redux" - авторы: Alex Banks, Eve Porcello; издательство: O'Reilly Media; город: Себастьянполис; год: 2017; число страниц: 350.Джон Дакетт - "HTML и CSS. Разработка и дизайн веб-сайтов", Эксмо, 2017, 480 стр. (ориг. название: "HTML и CSS: Design and Build Websites", John Wiley & Sons)
3. "Node.js Design Patterns" - автор: Mario Casciaro; издательство: Packt Publishing; город: Бирмингем; год: 2016; число страниц: 454.
4. “Node.js Design Patterns" - автор: Mario Casciaro; издательство: Packt Publishing; город: Бирмингем; год: 2016; число страниц: 454.
5. "Web Development with Node and Express: Leveraging the JavaScript Stack" - автор: Ethan Brown; издательство: O'Reilly Media; город: Себастьянполис; год: 2018; число страниц: 334.
6. "Mastering Node.js" - авторы: Sandro Pasquali, Paweł Kozłowski; издательство: Packt Publishing; город: Бирмингем; год: 2017; число страниц: 502.

# **Приложение А – Листинг программного кода**

**Server**

**Server.Controllers. authorController**

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

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

class AuthorController {

    async create(req,res) {

        const {name} =  req.body

        const author = await Author.create({name})

        return res.json(author)

    }

    async getAll(req,res) {

        const authors = await Author.findAll()

        return res.json(authors)

    }

}

module.exports = new AuthorController()

**Server.Controllers. bookController**

const uuid = require('uuid')

const path = require('path');

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

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

const { where } = require('sequelize');

class BookController {

    async create(req,res) {

        try{

            const {name,price,description,genreId,authorId} = req.body

            const {img} = req.files

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

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

            const book = await Book.create({name, price, img: fileName, description, genreId, authorId})

            return res.json(book)

        } catch(e){

            next(ApiError.badRequest(e.message))

        }

    }

    async getAll(req,res) {

        let {genreId, authorId, limit, page} = req.query

        page = page || 1

        limit = limit || 9

        let offset = page \* limit - limit

        let books;

        if (!genreId && !authorId) {

            books = await Book.findAndCountAll({limit, offset})

        }

        if (genreId && !authorId) {

            books = await Book.findAndCountAll({where:{genreId}, limit, offset})

        }

        if (!genreId && authorId) {

            books = await Book.findAndCountAlll({where:{authorId}, limit, offset})

        }

        if (genreId && authorId) {

            books = await Book.findAndCountAll({where:{genreId, authorId}, limit, offset})

        }

        return res.json(books)

    }

    async getOne(req,res) {

        const {id} = req.params

        const book = await Book.findOne(

            {where: {id}},

        )

        return res.json(book)

    }

}

module.exports = new BookController()

**Server.Controllers. genreController**

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

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

class GenreController {

    async create(req,res) {

        const {name} =  req.body

        const genre = await Genre.create({name})

        return res.json(genre)

    }

    async getAll(req,res) {

        const genres = await Genre.findAll()

        return res.json(genres)

    }

}

module.exports = new GenreController()

**Server.Controllers. userController**

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.badRequest("Пользователь с таким именем не найден"))

        }

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

        if(!comparePassword){

            return next(ApiError.badRequest("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()

**Server.Error. ApiError**

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

**Server. MiddleWare.auth**

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"})

    }

};

**Server.MiddleWare.checkRole**

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"})

        }

    };

}

**Server.MiddleWare.ErrorHanding**

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!"})

}

**Server.Routes.Author**

const Router = require('express')

const router = new Router()

const authorController = require('../controllers/authorController')

const checkRole = require('../middleware/checkRoleMiddleware')

router.post('/',checkRole('ADMIN'), authorController.create)

router.get('/',authorController.getAll)

module.exports = router

**Server. Routes.Book**

const Router = require('express')

const router = new Router()

const bookController = require('../controllers/bookController')

const checkRole = require('../middleware/checkRoleMiddleware')

router.post('/', checkRole('ADMIN'), bookController.create)

router.get('/',bookController.getAll)

router.get('/:id',bookController.getOne)

module.exports = router

**Server.Routes.Genre**

const Router = require('express')

const router = new Router()

const genreController = require('../controllers/genreController')

const checkRole = require('../middleware/checkRoleMiddleware')

router.post('/', checkRole('ADMIN'), genreController.create)

router.get('/',genreController.getAll)

module.exports = router

**Server.Routes.index**

const Router = require('express')

const router = new Router()

const bookRouter = require('./bookRouter')

const userRouter = require('./userRouter')

const genreRouter = require('./genreRouter')

const authorRouter = require('./authorRouter')

router.use('/user',userRouter)

router.use('/genre',genreRouter)

router.use('/author',authorRouter)

router.use('/book',bookRouter)

module.exports = router

**Server.Routes.user**

const Router = require('express')

const router = new Router()

const userController = require('../controllers/userController')

const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration', userController.registration)

router.post('/login', userController.login)

router.get('/auth', authMiddleware,userController.check)

module.exports = router

**Server.ORM.DB**

const {Sequelize} = require('sequelize')

module.exports = new Sequelize(

    process.env.DB\_NAME, //Название БД

    process.env.DB\_USER, //Имя пользователя

    process.env.DB\_PASSWORD, //Пароль

    {

        dialect: 'postgres',

        host: process.env.DB\_HOST,

        port: process.env.DB\_PORT

    }

)

**Server.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 ошибки

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()

**Client**

**Client.Components.modals.CreateBook**

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/bookAPI";

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

const CreateBook = 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('genreId', device.selectedBrand.id)

        formData.append('authorId', 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 "}</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 "}</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 "

                    />

                    <Form.Control

                        value={price}

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

                        className="mt-3"

                        placeholder="Введите "

                        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 "

                                />

                            </Col>

                            <Col md={4}>

                                <Form.Control

                                    value={i.description}

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

                                    placeholder="Введите"

                                />

                            </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;

**Client.components.AppRouter**

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;

**Client.components.AuthorBar**

import React, {useContext} from 'react';

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

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

import Col from "react-bootstrap/Col";

import ListGroup from "react-bootstrap/ListGroup";

const TypeBar = observer(() => {

    const {device} = useContext(Context)

    return (

        <ListGroup>

            {device.types.map(type =>

                <ListGroup.Item

                    style={{cursor: 'pointer'}}

                    active={type.id === device.selectedType.id}

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

                    key={type.id}

                >

                    {type.name}

                </ListGroup.Item>

            )}

        </ListGroup>

    );

});

export default TypeBar;

**Client.components.BookItem**

import React from 'react';

import {Card, Col} from "react-bootstrap";

import Image from "react-bootstrap/Image";

import star from '../assets/star.png'

import {useHistory} from "react-router-dom"

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

const BookItem = ({book}) => {

    const history = useHistory()

    return (

        <Col md={3} className={"mt-3"} onClick={() => history.push(DEVICE\_ROUTE + '/' + device.id)}>

            <Card style={{width: 150, cursor: 'pointer'}} border={"light"}>

                <Image width={150} height={150} src={process.env.REACT\_APP\_API\_URL + device.img}/>

                <div className="text-black-50 mt-1 d-flex justify-content-between align-items-center">

                    <div>S</div>

                    <div className="d-flex align-items-center">

                        <div>{device.rating}</div>

                        <Image width={18} height={18} src={star}/>

                    </div>

                </div>

                <div>{device.name}</div>

            </Card>

        </Col>

    );

};

export default DeviceItem;

**Client.components.BookList**

import React, {useContext} from 'react';

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

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

import {Row} from "react-bootstrap";

import BookItem from "./BookItem";

const BookList = observer(() => {

    const {device} = useContext(Context)

    return (

        <Row className="d-flex">

            {book.books.map(book =>

                <BookItem key={book.id} book={book}/>

            )}

        </Row>

    );

});

export default BookList;

**Client.components.GenreBar**

import React, {useContext} from 'react';

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

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

import {Card, Row} from "react-bootstrap";

const GenreBar = observer(() => {

    const {book} = useContext(Context)

    return (

        <Row className="d-flex">

            {books.genres.map(genre =>

                <Card

                    style={{cursor:'pointer'}}

                    key={brand.id}

                    className="p-3"

                    onClick={() => book.setSelectedGenre(genre)}

                    border={genre.id === book.selectedGenre.id ? 'danger' : 'light'}

                >

                    {Genre.name}

                </Card>

            )}

        </Row>

    );

});

export default GenreBar;

**Client.components.NavBar**

import React, {useContext} from 'react';

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

import Navbar from "react-bootstrap/Navbar";

import Nav from "react-bootstrap/Nav";

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

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

import {Button} from "react-bootstrap";

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

import Container from "react-bootstrap/Container";

import {useHistory} from 'react-router-dom'

const NavBar = observer(() => {

    const {user} = useContext(Context)

    const history = useHistory()

    const logOut = () => {

        user.setUser({})

        user.setIsAuth(false)

    }

    return (

        <Navbar bg="dark" variant="dark">

            <Container>

                <NavLink style={{color:'white'}} to={SHOP\_ROUTE}>КупиДевайс</NavLink>

                {user.isAuth ?

                    <Nav className="ml-auto" style={{color: 'white'}}>

                        <Button

                            variant={"outline-light"}

                            onClick={() => history.push(ADMIN\_ROUTE)}

                        >

                            Админ панель

                        </Button>

                        <Button

                            variant={"outline-light"}

                            onClick={() => logOut()}

                            className="ml-2"

                        >

                            Выйти

                        </Button>

                    </Nav>

                    :

                    <Nav className="ml-auto" style={{color: 'white'}}>

                        <Button variant={"outline-light"} onClick={() => history.push(LOGIN\_ROUTE)}>Авторизация</Button>

                    </Nav>

                }

            </Container>

        </Navbar>

    );

});

export default NavBar;

**Client.components.Pages**

import React, {useContext} from 'react';

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

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

import {Pagination} from "react-bootstrap";

const Pages = observer(() => {

    const {device} = useContext(Context)

    const pageCount = Math.ceil(device.totalCount / device.limit)

    const pages = []

    for (let i = 0; i < pageCount; i++) {

        pages.push(i + 1)

    }

    return (

        <Pagination className="mt-3">

            {pages.map(page =>

                <Pagination.Item

                    key={page}

                    active={device.page === page}

                    onClick={() => device.setPage(page)}

                >

                    {page}

                </Pagination.Item>

            )}

        </Pagination>

    );

});

export default Pages;

**Client.http.BookApi**

import {$authHost, $host} from "./index";

import jwt\_decode from "jwt-decode";

export const createType = async (type) => {

    const {data} = await $authHost.post('api/type', type)

    return data

}

export const fetchTypes = async () => {

    const {data} = await $host.get('api/type')

    return data

}

export const createBrand = async (brand) => {

    const {data} = await $authHost.post('api/brand', brand)

    return data

}

export const fetchBrands = async () => {

    const {data} = await $host.get('api/brand', )

    return data

}

export const createDevice = async (device) => {

    const {data} = await $authHost.post('api/device', device)

    return data

}

export const fetchDevices = async (typeId, brandId, page, limit= 5) => {

    const {data} = await $host.get('api/device', {params: {

            typeId, brandId, page, limit

        }})

    return data

}

export const fetchOneDevice = async (id) => {

    const {data} = await $host.get('api/device/' + id)

    return data

}

**Client.http.Index**

import axios from "axios";

const $host = axios.create({

    baseURL: process.env.REACT\_APP\_API\_URL

})

const $authHost = axios.create({

    baseURL: process.env.REACT\_APP\_API\_URL

})

const authInterceptor = config => {

    config.headers.authorization = `Bearer ${localStorage.getItem('token')}`

    return config

}

$authHost.interceptors.request.use(authInterceptor)

export {

    $host,

    $authHost

}

**Client.http.userApi**

import {$authHost, $host} from "./index";

import jwt\_decode from "jwt-decode";

export const registration = async (email, password) => {

    const {data} = await $host.post('api/user/registration', {email, password, role: 'ADMIN'})

    localStorage.setItem('token', data.token)

    return jwt\_decode(data.token)

}

export const login = async (email, password) => {

    const {data} = await $host.post('api/user/login', {email, password})

    localStorage.setItem('token', data.token)

    return jwt\_decode(data.token)

}

export const check = async () => {

    const {data} = await $authHost.get('api/user/auth' )

    localStorage.setItem('token', data.token)

    return jwt\_decode(data.token)

}

**Client.pages.Admin**

import React, {useState} from 'react';

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

import CreateBrand from "../components/modals/CreateGrene";

import CreateDevice from "../components/modals/CreateBook";

import CreateType from "../components/modals/CreateAuthor";

const Admin = () => {

    const [brandVisible, setBrandVisible] = useState(false)

    const [typeVisible, setTypeVisible] = useState(false)

    const [deviceVisible, setDeviceVisible] = useState(false)

    return (

        <Container className="d-flex flex-column">

            <Button

                variant={"outline-dark"}

                className="mt-4 p-2"

                onClick={() => setTypeVisible(true)}

            >

            </Button>

            <Button

                variant={"outline-dark"}

                className="mt-4 p-2"

                onClick={() => setBrandVisible(true)}

            >

            </Button>

            <Button

                variant={"outline-dark"}

                className="mt-4 p-2"

                onClick={() => setDeviceVisible(true)}

            >

            </Button>

            <CreateBrand show={brandVisible} onHide={() => setBrandVisible(false)}/>

            <CreateDevice show={deviceVisible} onHide={() => setDeviceVisible(false)}/>

            <CreateType show={typeVisible} onHide={() => setTypeVisible(false)}/>

        </Container>

    );

};

export default Admin;

**Client.pages.Auth**

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

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

import Card from "react-bootstrap/Card";

import Button from "react-bootstrap/Button";

import Row from "react-bootstrap/Row";

import {NavLink, useLocation, useHistory} from "react-router-dom";

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

import {login, registration} from "../http/userAPI";

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

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

const Auth = observer(() => {

    const {user} = useContext(Context)

    const location = useLocation()

    const history = useHistory()

    const isLogin = location.pathname === LOGIN\_ROUTE

    const [email, setEmail] = useState('')

    const [password, setPassword] = useState('')

    const click = async () => {

        try {

            let data;

            if (isLogin) {

                data = await login(email, password);

            } else {

                data = await registration(email, password);

            }

            user.setUser(user)

            user.setIsAuth(true)

            history.push(SHOP\_ROUTE)

        } catch (e) {

            alert(e.response.data.message)

        }

    }

    return (

        <Container

            className="d-flex justify-content-center align-items-center"

            style={{height: window.innerHeight - 54}}

        >

            <Card style={{width: 600}} className="p-5">

                <h2 className="m-auto">{isLogin ? 'Авторизация' : "Регистрация"}</h2>

                <Form className="d-flex flex-column">

                    <Form.Control

                        className="mt-3"

                        placeholder="Введите ваш email..."

                        value={email}

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

                    />

                    <Form.Control

                        className="mt-3"

                        placeholder="Введите ваш пароль..."

                        value={password}

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

                        type="password"

                    />

                    <Row className="d-flex justify-content-between mt-3 pl-3 pr-3">

                        {isLogin ?

                            <div>

                                Нет аккаунта? <NavLink to={REGISTRATION\_ROUTE}>Зарегистрируйся!</NavLink>

                            </div>

                            :

                            <div>

                                Есть аккаунт? <NavLink to={LOGIN\_ROUTE}>Войдите!</NavLink>

                            </div>

                        }

                        <Button

                            variant={"outline-success"}

                            onClick={click}

                        >

                            {isLogin ? 'Войти' : 'Регистрация'}

                        </Button>

                    </Row>

                </Form>

            </Card>

        </Container>

    );

});

export default Auth;

**Client.pages.Basket**

import React from 'react';

const Basket = () => {

    return (

        <div>

            basket

        </div>

    );

};

export default Basket;

**Client.pages.Bookpage**

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

import {Button, Card, Col, Container, Image, Row} from "react-bootstrap";

import bigStar from '../assets/bigStar.png'

import {useParams} from 'react-router-dom'

import {fetchOneDevice} from "../http/bookAPI";

const DevicePage = () => {

    const [device, setDevice] = useState({info: []})

    const {id} = useParams()

    useEffect(() => {

        fetchOneDevice(id).then(data => setDevice(data))

    }, [])

    return (

        <Container className="mt-3">

            <Row>

                <Col md={4}>

                    <Image width={300} height={300} src={process.env.REACT\_APP\_API\_URL + device.img}/>

                </Col>

                <Col md={4}>

                    <Row className="d-flex flex-column align-items-center">

                        <h2>{device.name}</h2>

                        <div

                            className="d-flex align-items-center justify-content-center"

                            style={{background: `url(${bigStar}) no-repeat center center`, width:240, height: 240, backgroundSize: 'cover', fontSize:64}}

                        >

                            {device.rating}

                        </div>

                    </Row>

                </Col>

                <Col md={4}>

                    <Card

                        className="d-flex flex-column align-items-center justify-content-around"

                        style={{width: 300, height: 300, fontSize: 32, border: '5px solid lightgray'}}

                    >

                        <h3>От: {device.price} руб.</h3>

                        <Button variant={"outline-dark"}>Добавить в корзину</Button>

                    </Card>

                </Col>

            </Row>

            <Row className="d-flex flex-column m-3">

                <h1></h1>

                {device.info.map((info, index) =>

                    <Row key={info.id} style={{background: index % 2 === 0 ? 'lightgray' : 'transparent', padding: 10}}>

                        {info.title}: {info.description}

                    </Row>

                )}

            </Row>

        </Container>

    );

};

export default DevicePage;

**Client.pages.shop**

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

import {Container} from "react-bootstrap";

import Row from "react-bootstrap/Row";

import Col from "react-bootstrap/Col";

import TypeBar from "../components/AuthorBar";

import BrandBar from "../components/GreneBar";

import DeviceList from "../components/BookList";

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

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

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

import Pages from "../components/Pages";

const Shop = observer(() => {

    const {device} = useContext(Context)

    useEffect(() => {

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

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

        fetchDevices(null, null, 1, 2).then(data => {

            device.setDevices(data.rows)

            device.setTotalCount(data.count)

        })

    }, [])

    useEffect(() => {

        fetchDevices(device.selectedType.id, device.selectedBrand.id, device.page, 2).then(data => {

            device.setDevices(data.rows)

            device.setTotalCount(data.count)

        })

    }, [device.page, device.selectedType, device.selectedBrand,])

    return (

        <Container>

            <Row className="mt-2">

                <Col md={3}>

                    <TypeBar/>

                </Col>

                <Col md={9}>

                    <BrandBar/>

                    <DeviceList/>

                    <Pages/>

                </Col>

            </Row>

        </Container>

    );

});

export default Shop;

**Client.App**

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;

**Client.index**

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')

);

**Client.Routes**

import Admin from "./pages/Admin";

import {ADMIN\_ROUTE, BASKET\_ROUTE, BOOK\_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 BookPage from "./pages/BookPage";

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: BOOK\_ROUTE + '/:id',

        Component: BookPage

    },

]