МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема Web-сайт «Интернет-магазин электроники»

**Исполнитель**

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П. Д. Заяц

подпись, дата

**Руководитель**

Ассистент каф. ИСИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Кантарович

должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Кантарович

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc122299324)

[1. Постановка задач и обзор аналогичных решений 6](#_Toc122299325)

[1.1 Актуальность темы 6](#_Toc122299326)

[1.2 Обзор аналогичных решений 6](#_Toc122299327)

[1.3 Постановка задачи 10](#_Toc122299328)

[1.4 Вывод 10](#_Toc122299329)

[2. Проектирование веб-сайта 11](#_Toc122299330)

[2.1 Выбор средств реализации 11](#_Toc122299331)

[2.2 Проектирование uml диаграммы 13](#_Toc122299332)

[2.3 Проектирование базы данных 13](#_Toc122299333)

[2.4 Проектирование страниц 14](#_Toc122299334)

[2.5 Выбор стилевого оформления 16](#_Toc122299335)

[2.6 Разработка пользовательских элементов 16](#_Toc122299336)

[2.7 Вывод 19](#_Toc122299337)

[3 Реализация структуры веб-сайты 20](#_Toc122299338)

[3.1 Реализация клиентской части 28](#_Toc122299341)

[3.2 Реализация серверной части 30](#_Toc122299342)

[3.6 Вывод 32](#_Toc122299344)

[4 Тестирование веб-сайта 33](#_Toc122299345)

[4.1 Адаптивный дизайн веб-сайта 33](#_Toc122299346)

[4.2 Кроссбраузерность веб-сайта 34](#_Toc122299347)

[4.3 Ручное тестирование 37](#_Toc122299348)

[4.4 Вывод 38](#_Toc122299349)

[5 Руководство пользователю 39](#_Toc122299350)

[Заключение 40](#_Toc122299351)

[Список использованных источников 41](#_Toc122299352)

[Приложение А Макет структуры веб-сайта 42](#_Toc122299353)

[Приложение Б Листинг описания базы данных 46](#_Toc122299354)

[Приложение В Листинг описания UserController 48](#_Toc122299355)

**Введение**

Электроника и бытовая техника настолько прочно вошли в жизнь людей, что уже сложно представлять своё существование без смартфона, компьютера или широкоформатного телевизора. Но даже если без этих вещей ещё можно как-то прожить, то существует ряд приборов, без которых быт в современном мире невозможен. Если учесть, что любая техника имеет срок эксплуатации, очевидно, что она подлежит периодической замене. Многие предприниматели считают эту причину одной из главных для открытия собственного дела по продаже электронных и бытовых товаров. Для успешного старта необходимо составить бизнес-план магазина электроники, оценить риски и предполагаемые затраты.

Сегодня в каждом среднем и крупном городе присутствует несколько сетевых супермаркетов электроники, отличающихся между собой лишь названием. Реклама по ТВ, радио и в интернете просто пестрит всевозможными призывами и акциями, направленными в сторону приобретения дешёвого смартфона, телевизора или игровой приставки.

Тема моей курсовой работы «Создание интернет магазина бытовой техники». Веб-сайт представляет из себя интернет магазин, который предоставляет услуги покупки товаров через интернет, с последующей доставкой.

В качестве названия было выбрано слово «Lucrum», важно чтобы название было броским и запоминающемся, а к тому же «Lucrum», в переводе с латыни означает «выгода», что определяет наш магазин.

Цель курсовой работы: разработать веб-приложение онлайн магазина бытовой техники, которое работало бы в реальном времени, с использованием REACT, HTML5, Sass, REDUX, а также с применение базы данных PostgreSQL.

Задачи:

* Проанализировать существующие языки разметки, инструменты и библиотеки для создания веб-приложения.
* Разработать макет приложения.
* Разработать структуру веб-приложения.
* Разработать функциональную часть веб-приложения.
* Подключить облачное хранилище данных.
* Протестировать веб-приложение.
* Разработать руководство пользователя.

# Постановка задачи и обзор аналогичных решений

## 1.1 Актуальность темы

Актуальность темы заключается в том, что интернет-магазин является одним из самых популярных видов сайтов, которые посещают пользователи. Как для покупателей важно, чтобы было удобно и быстро совершать покупки, так и для продавцов важно, чтобы было удобно и быстро добавлять товары в каталог, а также управлять ими.

Сегодня на рынке существует множество интернет-магазинов, которые используют различные технологии и платформы для создания своих сайтов. Но, как правило, они используют один из самых популярных фреймворков для создания интернет-магазинов – React JS.

# 1.2 Обзор аналогичных решений

Темой моей курсовой работы является создание интернет магазина бытовой техники. В интернете уже существуют некоторые аналогичные решения. В этом разделе будут приведены веб-сайты существующих магазинов бытовой техники. Для примера возьмём 3 сайта:

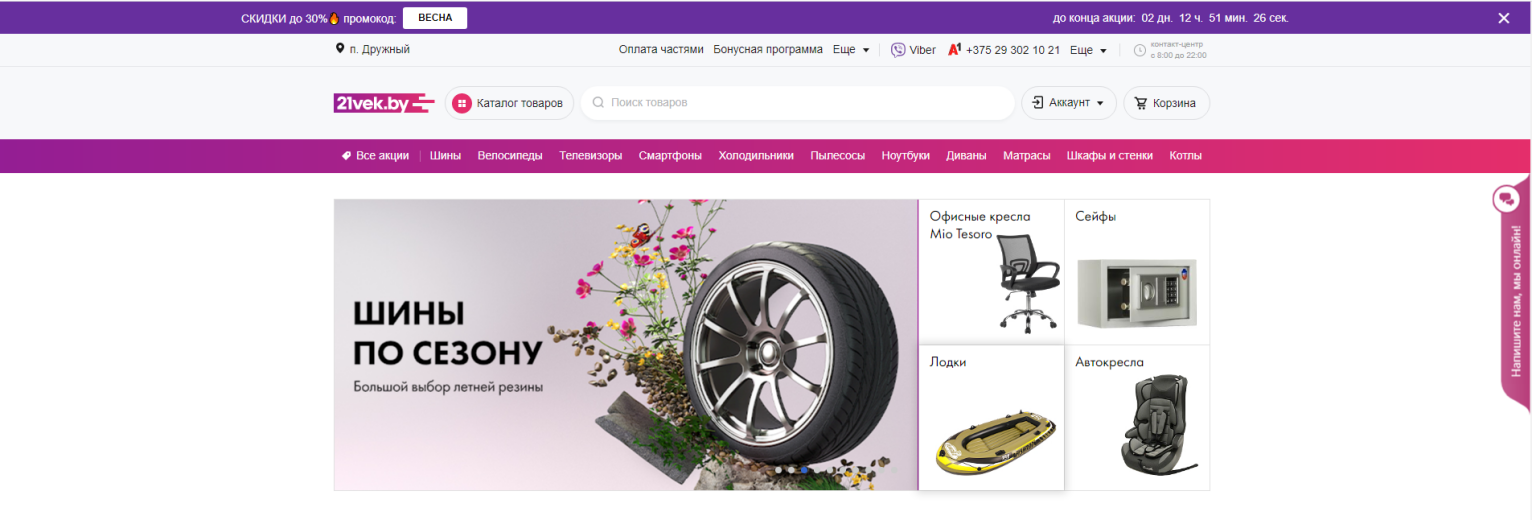
На рисунке 1.1 представлена главная страница сайта «21vek» [[1]](https://www.21vek.by/) Сайт имеет приятный интерфейс, важная информация(скидки, основные категории товаров и т.д.), оформленная в виде слайдера, содержатся на главной странице, имеется приятный дизайн.

Рисунок 1.1 – Сайт интернет магазина «21vek.by»

Стоит отметить цветовую палитру, здесь она в основном состоит из фиолетового и розового цветов, что означает то, что сайт ориентирован на женскую половину, так как для женской аудитории фиолетовый и розовый оттенки популярны.

Из минусов можно подметить обычный шрифт, которые плохо контрастирует с основным дизайном, из-за чего создаётся ощущение визуального шума сайта, что конечно отталкивает пользователя.

А также, на рисунке 1.2, можно увидеть подвал сайт, который является минусом, дизайн почти отсутствует, из-за чего кажется, что это просто текстовый набор, что само собой отталкивает потенциальных покупателей.

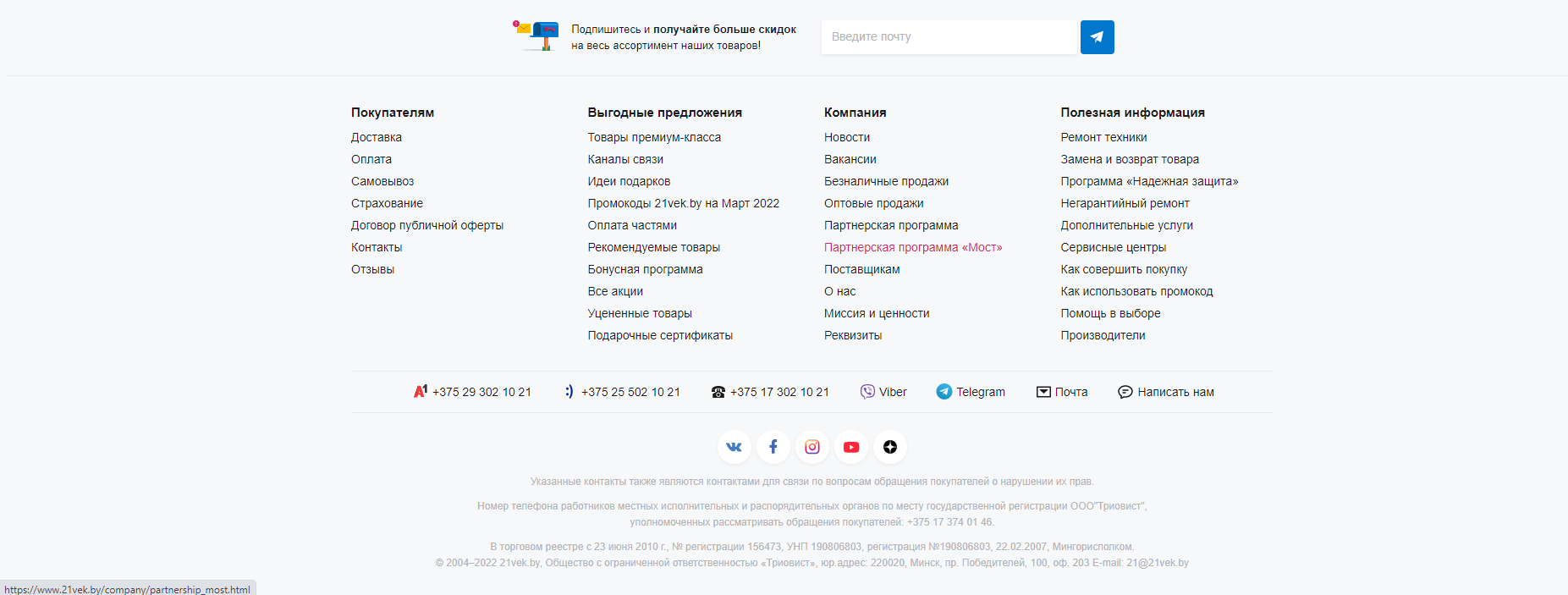
Стоит заметить и функциональные особенности сайта, на сайте реализована функция добавления в корзину и оформления заказа, где будут указаны данные доставки, цены и т.д.

Рисунок 1.2 – Подвал сайта интернет магазина «21vek.by»

На рисунке 1.3 главная страница сайт «5 элемент» [[2]](https://5element.by/) Сайт, который имеет хороший и приятный дизайн, плавную анимацию и интерактив, здесь, как и на сайте «21vek», важная информация (скидки, основные категории товаров и т.д.), оформленная в виде слайдера и содержится на главной странице, также красиво оформленный информационный панели, что добавляет удобство, в отличие от прошлого сайта, здесь недостаток в виде неприятного шрифта отсутствует, здесь шрифт хорошо вписался в общий дизайн сайта.

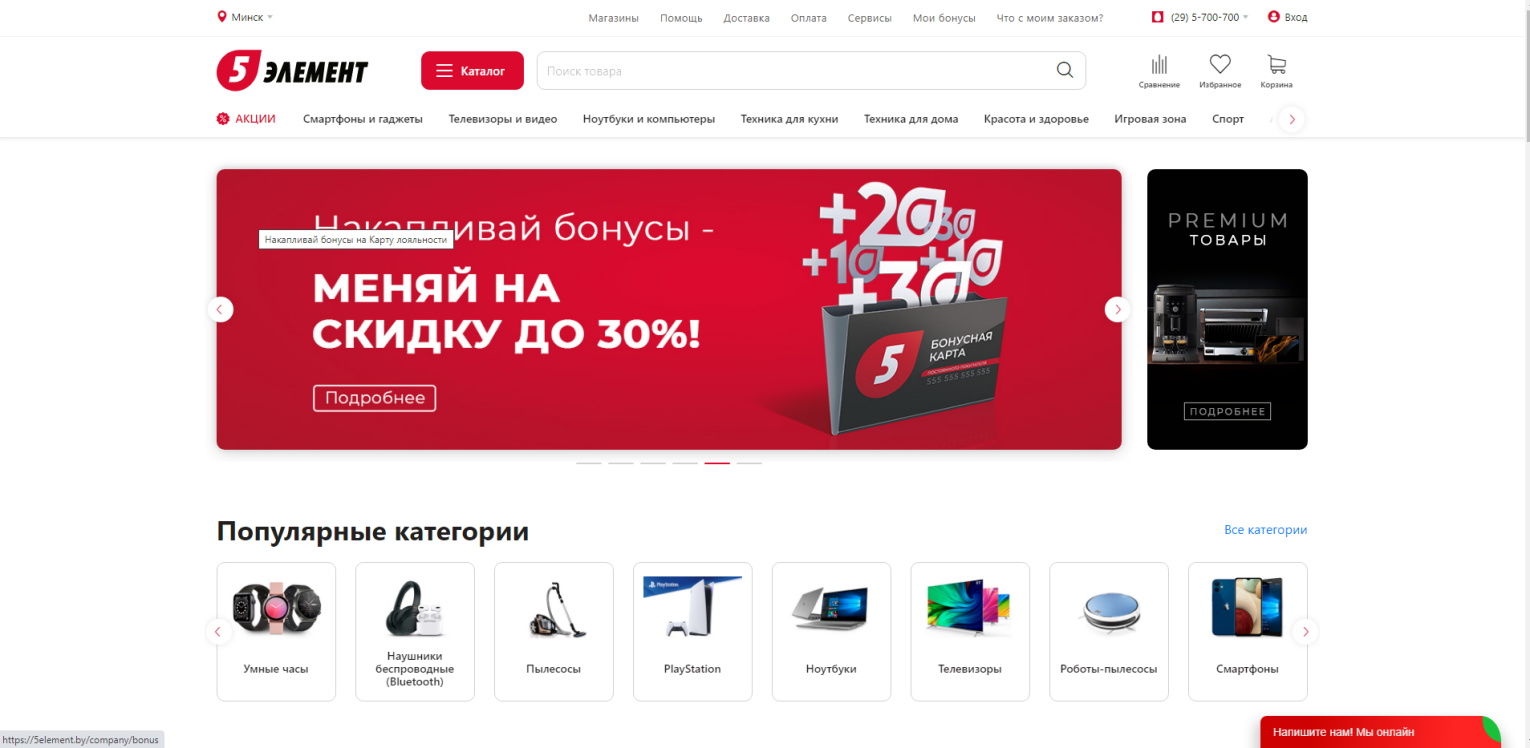


Рисунок 1.3 – Сайт интернет магазина «5 элемент»

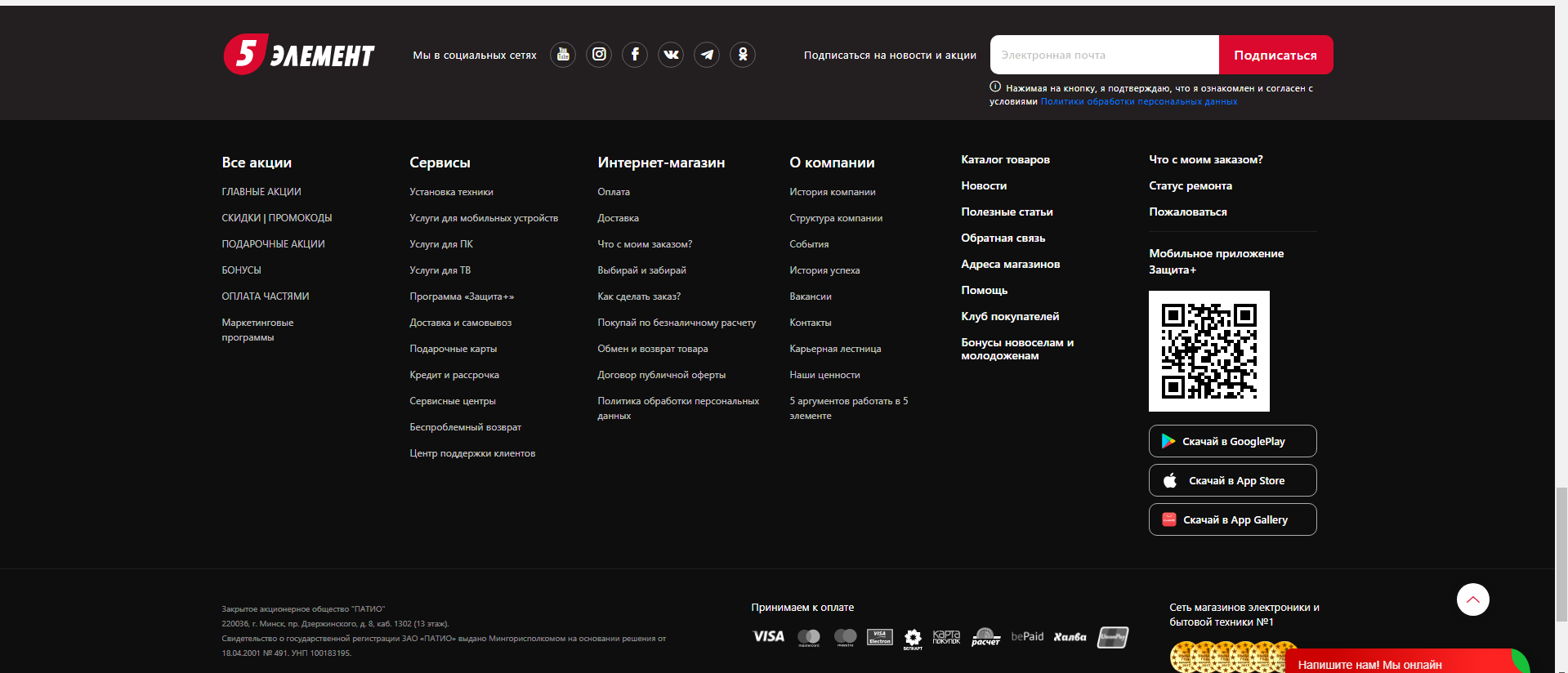
Хотелось бы подметить, что на рисунке 1.4, в отличие от «21vek.by», подвал сайта «5 элемент», очень удобный, с хорошим дизайном, приятен для глаз, белый текст на чёрном фоне, намного приятнее чёрного текста на белом фоне, функциональность подвала присутствует, тут имеется как и часто задаваемые вопросы, так и нахождение по адресу, информация о компании, QR-код, акции, сервисы и т.д.

Рисунок 1.4 – Подвал сайта интернет магазина «5 элемент»

На сайте имеется возможность товар в избранное, там будут хранится товары, которые понравились пользователю. Также имеется функция сравнения товаров, где вы можете выбрать 2 товара и сравнить их характеристики, после чего выбрать подходящий.

На рисунке 1.5 представлена главная страница зарубежного магазина «Walmart» [[3]](https://www.walmart.com/). Опять же хороший дизайн сайта, удобство, комфорт, все это есть на нем. Как и на прошлых сайтах на главной странице есть слайдер, в котором содержится важная информация (скидки, основные категории товаров и т.д.).

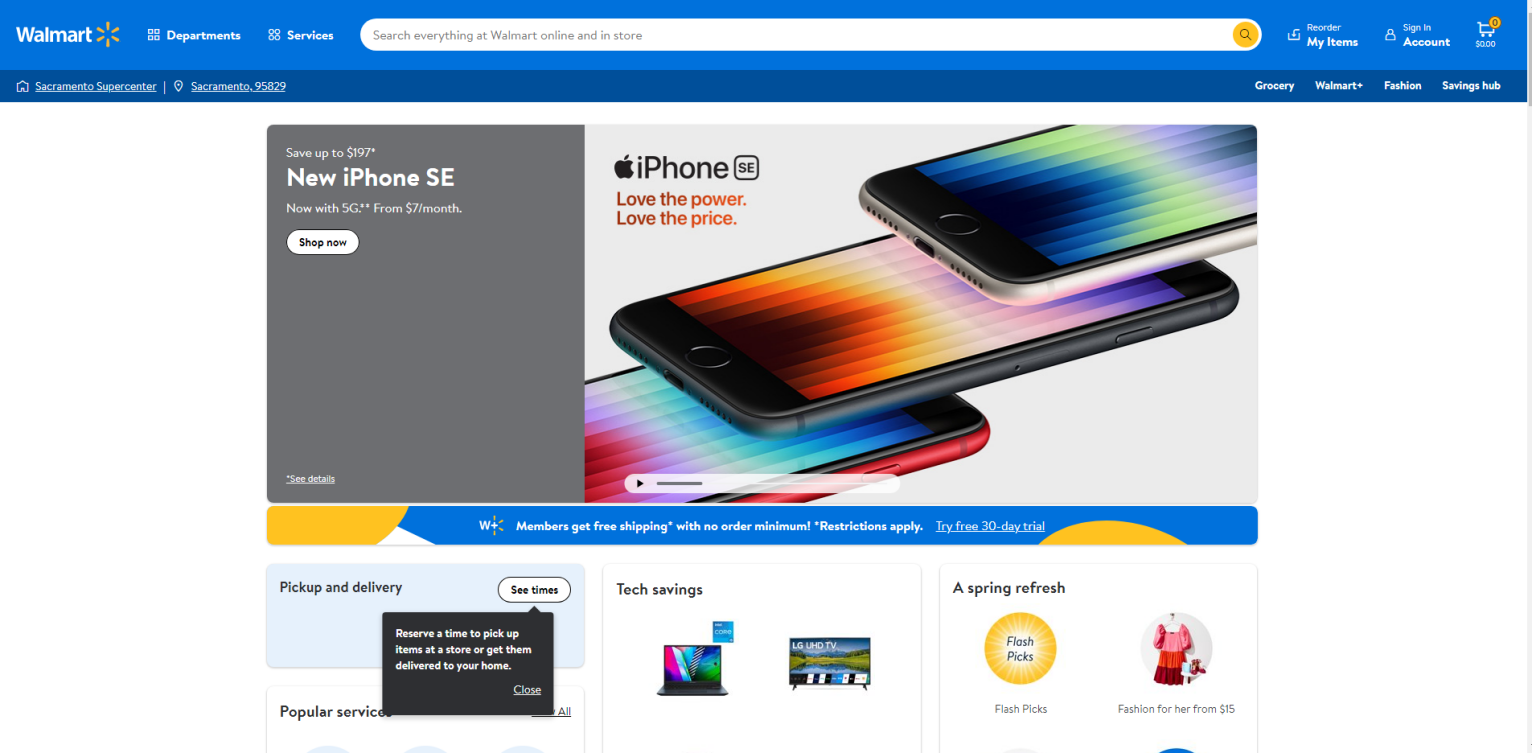
 Из особенностей сайта стоит подметить то, что основным цветом является синий, сам по себе синий цвет яркий, а также является любимым цветом большинства людей, что даёт несомненно огромную выгоду в продвижение, а также комфорт для самих пользователей.

Рисунок 1.5 – Сайт интернет магазина «Walmart»

На рисунке 1.6 представлена основой часть сайта категории, товары пестрят простыми, но в той же мере красивыми картинками, что тоже является плюсом для пользователей и самого сайта.

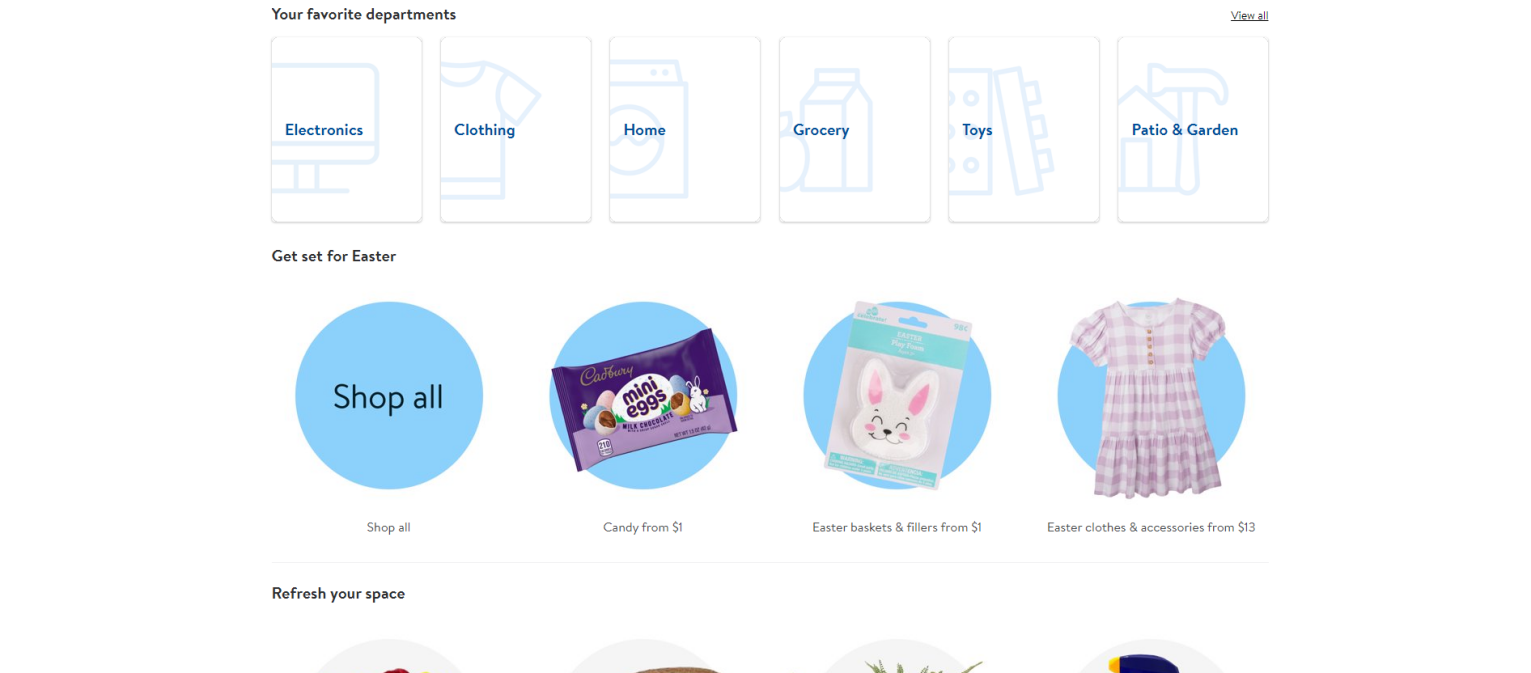


Рисунок 1.6 – Сайт интернет магазина «Walmart»

На рисунке 1.7 представлен подвал сайта, там нет такого объёма информации что и на прошлых сайтах, из-за чего некоторые пользователи не смогут найти интересующую их информацию.

Рисунок 1.7 – Подвал сайта интернет магазина «Walmart»

В интернет-магазине «Walmart» также имеются функции добавления в корзину, сравнения товаров или добавления в избранное.

# 1.3 Постановка задачи

Основными задачами курсового проекта являются:

* + Разработка многостраничного сайта
  + Создание удобного интерфейса
  + Создание дизайна, делающего сайт более привлекательным для клиентов
  + Разработка адаптивности для веб-страниц.
  + Создание базы данных

Основными задачами программного средства являются: создание привлекательного для пользователя дизайна, разработка динамического сайта, хранение контента сайта в базе данных, добавление функциональности и анимации отдельных элементов сайта.

# 1.4 Вывод

В данном разделе были рассмотрены аналогичные решения, были проанализированы их недостатки и преимущества, для создания в последующем качественного веб-сайта. Были поставлены конкретные задачи для создания интерактивного информационного ресурса. Кроме того, были рассмотрены средства реализации программного продукта, такие как языки разметки, а также редактор кода VS Code. Были проанализированы особенности языков и основные дополнительные средства редактора. Все вышеперечисленные пункты понадобятся для достижения поставленной задачи и для создания в дальнейшем качественного продукта.

Основываясь на общих выводах, на вычисленных преимуществах и недостатков, можно сказать что сайт будет выполнен в серых оттенках, будет подобран хороший контрастирующий шрифт, подвал(footer) будет оформлен с небольшим объёмом нужной информации, главная страница будет выполнена в стиле минимализма, для того чтобы пользователь мог видеть только нужную информацию; Главным преимуществом нашего сайта будет краткость, функциональность и информативность, простота и удобство - главное требование к сайту.

Основными задачами курсового проекта являются:

* + Разработка многостраничного сайта
  + Создание удобного интерфейса
  + Создание дизайна, для привлекательности сайта
  + Разработка адаптивности для веб-страниц.
  + Создание базы данных

# Проектирование программного средства

# 2.1 Выбор средств реализации

Сайт разрабатывался в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах как Internet Explorer, Microsoft Edge, Safari, Opera, Mozilla Firefox, Google Chrome.

Visual Studio Code — [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0), разработанный [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) для [Windows](https://ru.wikipedia.org/wiki/Windows), [Linux](https://ru.wikipedia.org/wiki/Linux) и [macOS](https://ru.wikipedia.org/wiki/MacOS). Позиционируется как «лёгкий» редактор кода для [кроссплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) разработки [веб](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)- и [облачных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) приложений. Включает в себя [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), инструменты для работы с [Git](https://ru.wikipedia.org/wiki/Git), [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0) и средства для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3).

SPA (Одностраничное приложение) - это веб-приложение или веб-сайт, который взаимодействует с веб-браузером, динамически перезаписывая текущий веб-сайт page с новыми данными с веб-сервера вместо метода по умолчанию, при котором браузер загружает все новые страницы. В React это достигается с помощью Browser Router, часто называемого React Router and Link.

Для реализации будущего сайта будет использована следующая архитектура:

* + Front-end.
  + Back-end.

Фронтенд (англ. frontend) — эторазработка пользовательских функций и интерфейса. К ним относится всё, что пользователи видят на сайте или в приложении, и с чем можно взаимодействовать: картинки, выпадающие списки, меню, анимация, карточки товаров, кнопки, чекбоксы, интерактивные элементы. В данном случае будет использованы следующие технологии: React JS (18.2.0), Axios(1.2.1), Mobx(6.7.0), Bootsrap(5.2.3), JWT-decode(3.1.2).

JavaScript – это язык программирования, который используется для создания динамических веб-страниц. JavaScript позволяет создавать интерактивные веб-страницы, которые могут реагировать на действия пользователя.

React JS — это библиотека JavaScript, предназначенная для разработки пользовательских интерфейсов. В отличие от других библиотек JavaScript React JS использует компонентный подход, который позволяет разрабатывать сложные интерфейсы из простых компонентов. С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, заполнение форм, применение фильтров, добавление товаров в корзину и так далее. React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом.

Axios — это библиотека для работы с HTTP-запросами. Axios позволяет создавать запросы к API, которые будут возвращать промисы, которые можно обрабатывать с помощью async/await.

Bootstrap-это бесплатный CSS-фреймворк с открытым исходным кодом, предназначенный для адаптивной, ориентированной на мобильные устройства фронтальной веб-разработки. Он содержит HTML, CSS и (опционально) JavaScript-шаблоны дизайна для типографии, форм, кнопок, навигации и других компонентов интерфейса.

MobX — это библиотека, делающая управление состоянием приложения простым и масштабируемым, применяя функционально-реактивное программирование. Философия MobX очень проста:“. Все, что может быть получено из состояния приложения, должно быть получено. Автоматически”. которое включает в себя пользовательский интерфейс, сериализацию данных, подключение к сервером и т.д.

JSON Web Token (JWT) — это открытый стандарт для создания токенов доступа, основанный на формате JSON. Токены создаются сервером Яндекса, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует токен для подтверждения своей личности перед своим сервером. Формат запроса.

Back-End — это разработка бизнес-логики продукта (сайта или веб-приложения). Бэкенд отвечает за взаимодействие пользователя с внутренними данными, которые потом отображает фронтенд. Попросту говоря, это то, что скрыто от глаз пользователя и происходит вне его браузера и компьютера. В данном случае будет использованы следующие технологии: Bcrypt(5.1.0), Cors(2.8.5), Express(4.18.2),Node(19.2.0), Sequelize(6.27.0).

Node.js — это JavaScript-окружение, которое позволяет запускать JavaScript-код на сервере. Node.js позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

Express — это фреймворк для Node.js, который позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

База данных - это совокупность взаимосвязанных данных. Для создания базы данных будет использована PostgreSQL.

Bcrypt - Адаптивная криптографическая хеш-функция формирования ключа, используемая для защищенного хранения паролей.

Cross-Origin Resource Sharing (CORS) — механизм, использующий дополнительные HTTP -заголовки, чтобы дать возможность агенту пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент.

Sequelize - это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в БД и отношений между ними с классами. При использовании. Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами. Причем Sequelize может работать с рядом СУБД - MySQL, Postgres, MariaDB, SQLite, MS SQL Server.

# 2.2 Проектирование UML диаграммы

Ролями, предоставляемыми данной системой, являются администратор, гость и пользователь. На основе предоставленного списка ролей необходимо построить UML диаграмму. UML диаграмма изображена на рисунке 2.1.

Таким образом видно, что неавторизованный пользователь имеет право авторизоваться или зарегистрироваться, в то время как уже авторизованный имеет больше возможностей, например арендовать автомобили и по желанию отменять бронь. Следующим идет администратор платформы, который обладает существенным администраторским функционалом, который позволяет ему,создать товар, тип или бренд, а также выполнять значительные изменения по части структуры объектов базы данных.

Данный этап является одним из самых значимых по время проектирования, так как именно тут решается, кто и какими ролями будет обладать в данной системе. Обзор аналогов не отвечает на этот вопрос, он просто дает некоторые подсказки, которые есть возможность воспользоваться, но не обязательно. Но данная сфера не обладает какими-либо серьезными связями между пользователями различных групп. Если все свести к минимуму, то можно заметить, что функционал достаточно тривиален, и не требует к себе особо серьезного внимания со стороны проектирования, тем не менее, это не делает этап менее значимых.

Каждая роль имеет своих участников, которые должны мочь выполнять только то, что предусмотрено для них системой, любое нарушение пределов данных и оговоренных в этом приложении ролей может привести к непредвиденным последствиям, что означает важность правильной реализации данный ролей. Реализация, таким образом, должна предусматривать, что каждый пользователь может совершать определенные действия, согласно его роли. Если пользователь пытается сделать что-то, что находиться вне его полномочий, система должна пресечь его действия, иначе они пагубно отразятся на системе.

Обычно при использовании правильных технологий, настройка прав идет достаточно просто и не требует каких-то больших временных затрат со стороны разработчика или тестировщика, но даже в таким серьезных технологиях могут встречаться изъяны, способные погубить целый проект, поэтому используя даже самые мощные и серьезные технологии нельзя относиться к этому пренебрежительно.

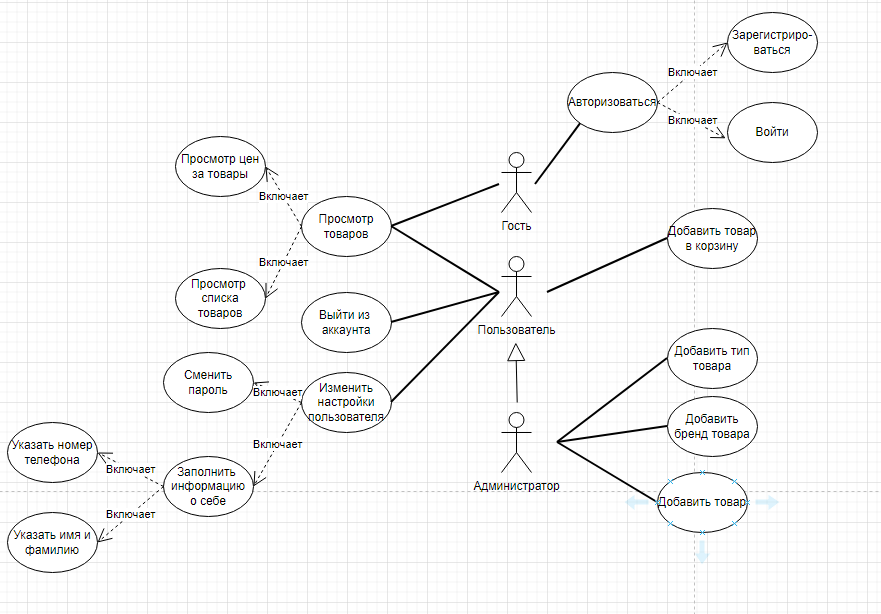


Рисунок 2.1 – Спроектированная UML диаграмма

Администратор, тот, кто является владельцем наибольших полномочий имеет право создавать товар, бренды и типы или же просто просматривать их список.

# 2.3 Проектирование базы данных

База данных – это совокупность взаимосвязанных данных, которые можно изобразить в виде схемы. На этом этапе важной деталью является проектирование базы данных, что и определит дальнейшую структуру приложения в целом. Спроектированная база данных изображена на рисунке 2.2.

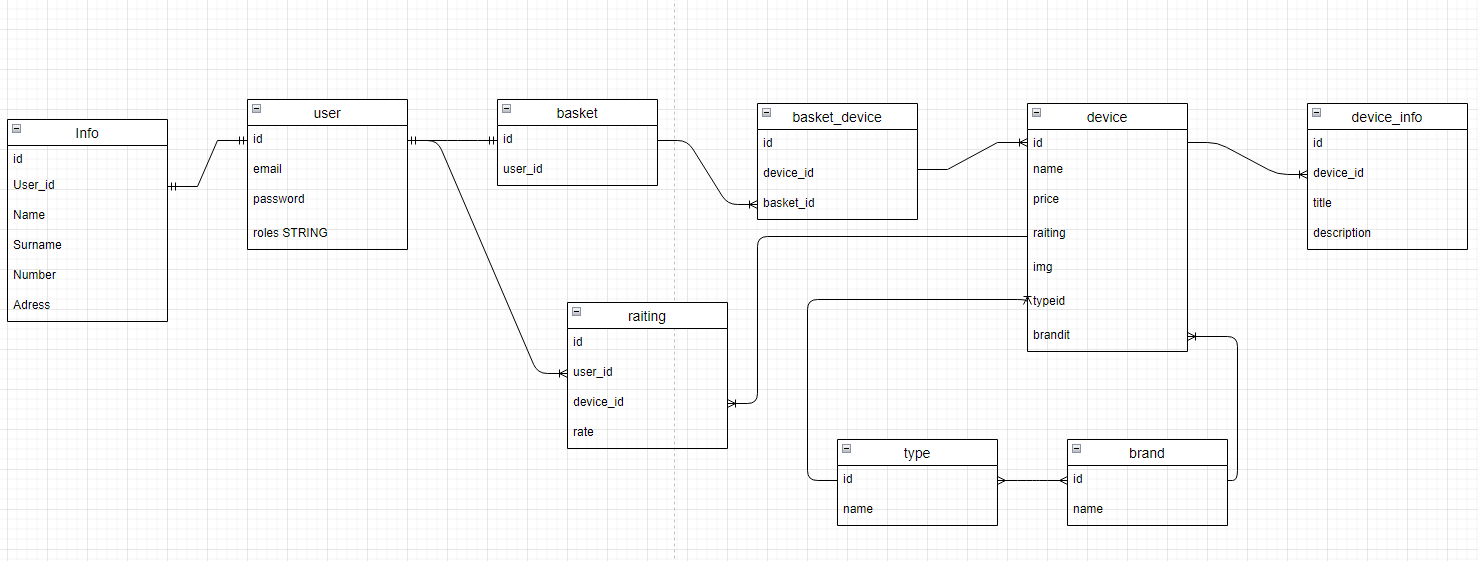


Рисунок 2.2 – Спроектированная база данных

Сущность «info» содержит в себе всю подробность о пользователе.

Сущность «user» содержит в себе необходимый минимум информации о пользователе для корректной работы сервиса.

Сущность «basket» является хранилищем пользовательских товаров.

Сущность «basket\_device» хранит в себе список товаров, которые хранятся в корзине пользователя.

Сущность «device» хранит в себе данные про товары, необходимые для отображения пользователю, а также для рассчета стоимости покупки.

Сущность «rating» является воплощением оценки пользователей тому или иному товару.

Сущность «brand» содержит в себе все бренды товаров, сохраненные на сервере.

Сущность «type» содержит построена по аналогии с таблицей «brand», за тем исключением, что в ней будут хранится уже типы товаров.

Сущность «device\_info» является хранилищем данных о определенном товаре.

# 2.4 Проектирование страниц

Первым шагом было принято создать макет сайта на основе рассмотренных аналогов, сделать макет удобным и легким. По макету будет понятно, где будут располагаться основные блоки, картинки, текст. Макет будет представлен в [приложении А.](#_Приложение_A_Прототипы)

На сайте будут представлены следующие страницы:

* + - Main (Главная страница).
    - Basket (Корзина).
    - Profile (Профиль).
    - Login (Вход).
    - Register (Регистрация).
    - ProductFull (Полное описание продукта).
    - Admin panel (Меню для Администратора).
    - Catalog (Каталог товаров).

На странице Main будет удобный слайдер, с акциями и выгодными предложениями,а так же часть категории товаров, которая может понравится покупателям, информация о компании, контакты, а также верхнее меню навигации/регистрации.

На странице Basket будут расположены карточки продуктов, форма оформления заказа, информация о оплате.

На странице Profile будут расположены личные данные пользователя. Здесь пользователь сможет изменить свои данные, изменить пароль, удалить аккаунт.

На странице Login будут расположены форма входа, ссылка на страницу регистрации.

На странице Register будут расположены форма регистрации, ссылка на страницу входа.

На странице Admin panel будет расположено меню для администратора, которое позволяет взаимодействовать с базой данных товаров.

На странице Catalog будет представлен перечень товаров, которые можно добавить в корзину, также в каталоге есть функция структурирование данных, то есть поиск товара по определенному бренду или типу.

## 2.5 Выбор стилевого оформления

Для реализации стилевого оформления сайта было выбрано минималистическое направление с элементами классического веб-дизайна. Такой стиль позволяет привлечь внимание новых пользователей. Сайт не загромождается лишними деталями (боковыми панелями и т.п.), большим количеством спецэффектов. Акцент на сайте ориентируется на основной контент.

Сайт будет состоять из 2-х основных цветов - серого и белого. Белый цвет будет использоваться для фона, а серый некоторых элементов дизайна. Эти цвета были подобраны так, потому что они будут приятны для глаз и это цвета, на которое пользователь сразу обратит внимания на сайт.

Верда́на ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Verdana*) — гуманистический [гротеск](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%BE%D1%82%D0%B5%D1%81%D0%BA_(%D1%88%D1%80%D0%B8%D1%84%D1%82)), созданный в [1996](https://ru.wikipedia.org/wiki/1996) [Мэтью Картером](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D1%80%D1%82%D0%B5%D1%80_%D0%9C%D1%8D%D1%82%D1%8C%D1%8E&action=edit&redlink=1) для компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), оптимизированный для отображения на экранах низкого разрешения. Шрифт включает в себя [хинтинги](https://ru.wikipedia.org/wiki/%D0%A5%D0%B8%D0%BD%D1%82%D0%B8%D0%BD%D0%B3), разработанные [Томасом Рикнером](https://en.wikipedia.org/wiki/Thomas_Rickner) в компании [Monotype](https://ru.wikipedia.org/wiki/Monotype). О необходимости создания подобного шрифта сообщила в рамках созданной [Стивом Балмером](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D0%B2_%D0%91%D0%B0%D0%BB%D0%BC%D0%B5%D1%80) группы по типографии в Microsoft.

Roboto (Робо́то) — [шрифтовая гарнитура](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D1%80%D0%BD%D0%B8%D1%82%D1%83%D1%80%D0%B0_(%D1%82%D0%B8%D0%BF%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0)) [без засечек](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%BE%D1%82%D0%B5%D1%81%D0%BA), которая была спроектирована американской корпорацией [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)), а затем представлена вместе с [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [Android 4.0 Ice Cream Sandwich](https://ru.wikipedia.org/wiki/Android_Ice_Cream_Sandwich) в качестве системного шрифта вышеназванной [мобильной ОС](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Гарнитура включает в себя прямое и наклонное начертания для шрифтов различных насыщенностей: сверхтонкое, тонкое, обычное, среднее, полужирное, очень жирное.

# 2.6 Разработка пользовательских элементов

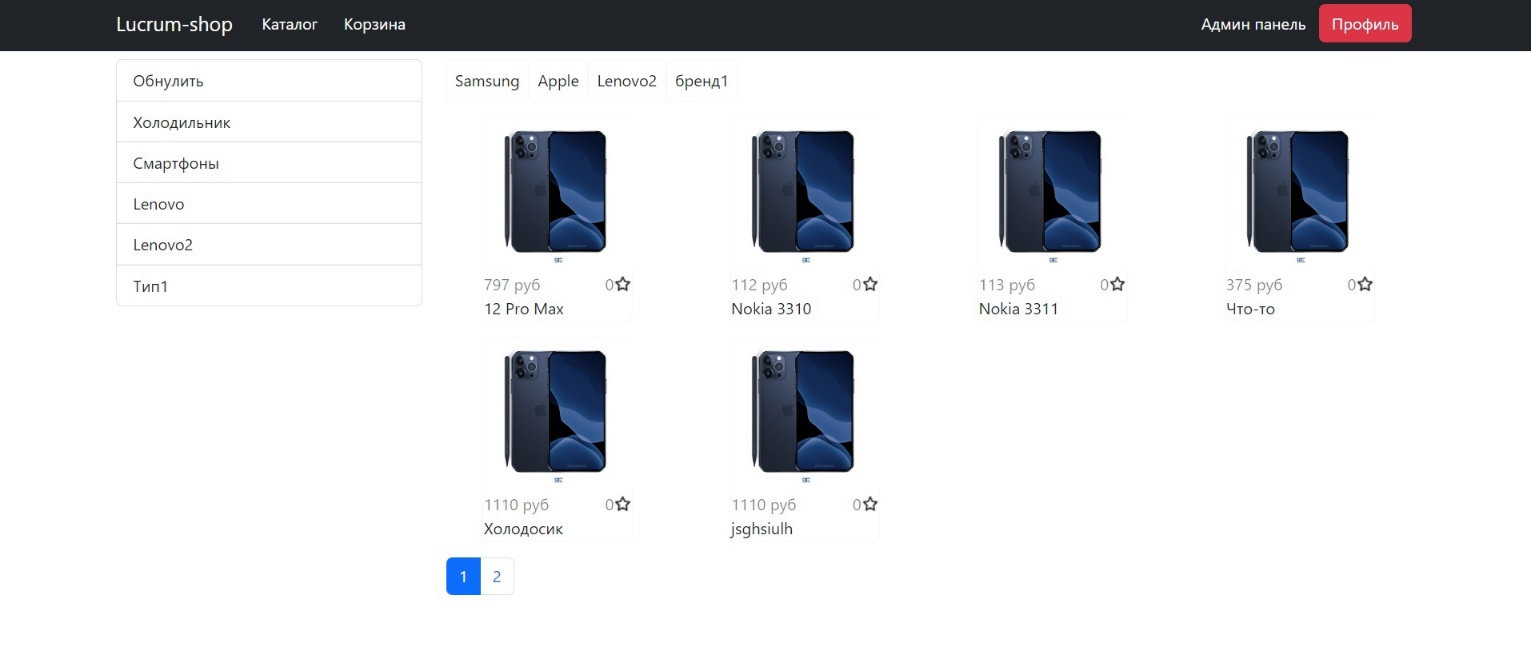
Страницы сайта разделены на «NavBar» (шапка сайта), «Section» (основной контент) и «footer» (подвал сайта). Все разделы разработаны с помощью технологии CSS Flexbox, которая предоставляет инструменты для быстрого создания сложных, гибких макетов, и функции, которые были сложны в традиционных методах CSS.

Рисунок 2.3 – Каталог сайта «Lucrum»

**Шапка сайта** включает в себя основные вкладки, такие как вход/регистрация, кнопки перехода в корзину, главное, каталог, также регистрация для незарегистрированного пользователя или меню профиля для авторизованого пользователя. Помимо этого, посетители должны иметь возможность узнавать бренд и понимать, чем занимается компания, как только они попадают на сайт. Также им важно удобство пользования ресурсом. Все это позволяет сделать грамотно спроектированный навигационная панель.

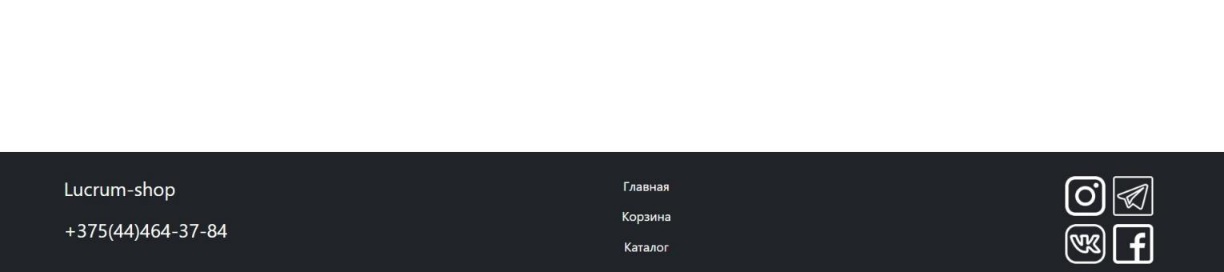


Рисунок 2.4 – Подвал сайта «Lucrum»

Как правило, **шапка, которая представлена на рисунке 2.3,** и подвал сайта, который предоставлен на рисунке 2.4, являются взаимодополняющими частями сайта и различаются местом расположения. В отличие от шапки, подвал сайта размещается внизу, после основного контента. Он содержит ссылки на интернет-магазин бытовой техники в социальных сетях, важную информацию, ссылку на корзину, каталог и главную страницу сайта.

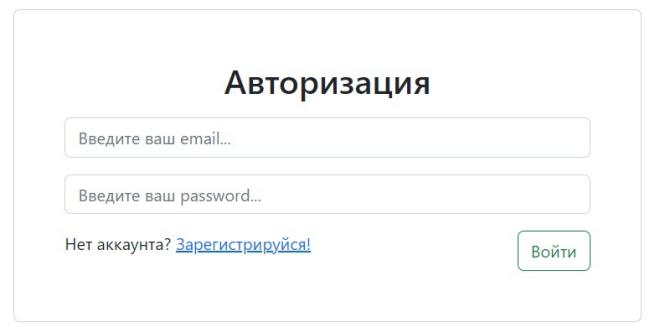
Так же на сайте будут реализованы следующие страницы: регистрация, авторизация, каталог, главная страница, админская панель, профиль. Для примера можно просмотреть следующие элементы элементы:

Рисунок 2.5 – Окно авторизации на сайте «Lucrum»

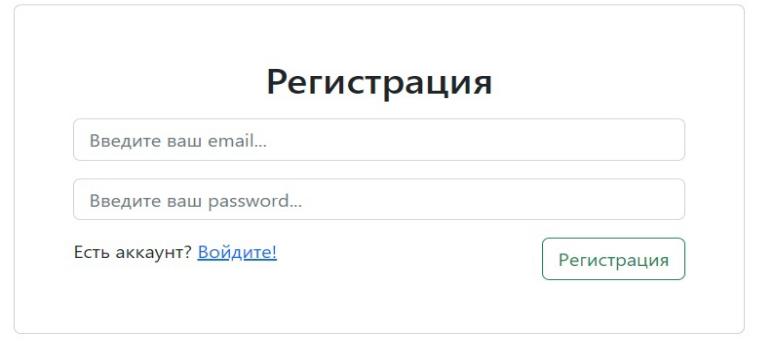
На рисунках 2.5 и 2.6, представлены окна авторизации и регистрации соответственно, файлы которые ввел пользователь отправятся на сервер и занесутся в базу данных.

Рисунок 2.6 – Окно регистрации на сайте «Lucrum»

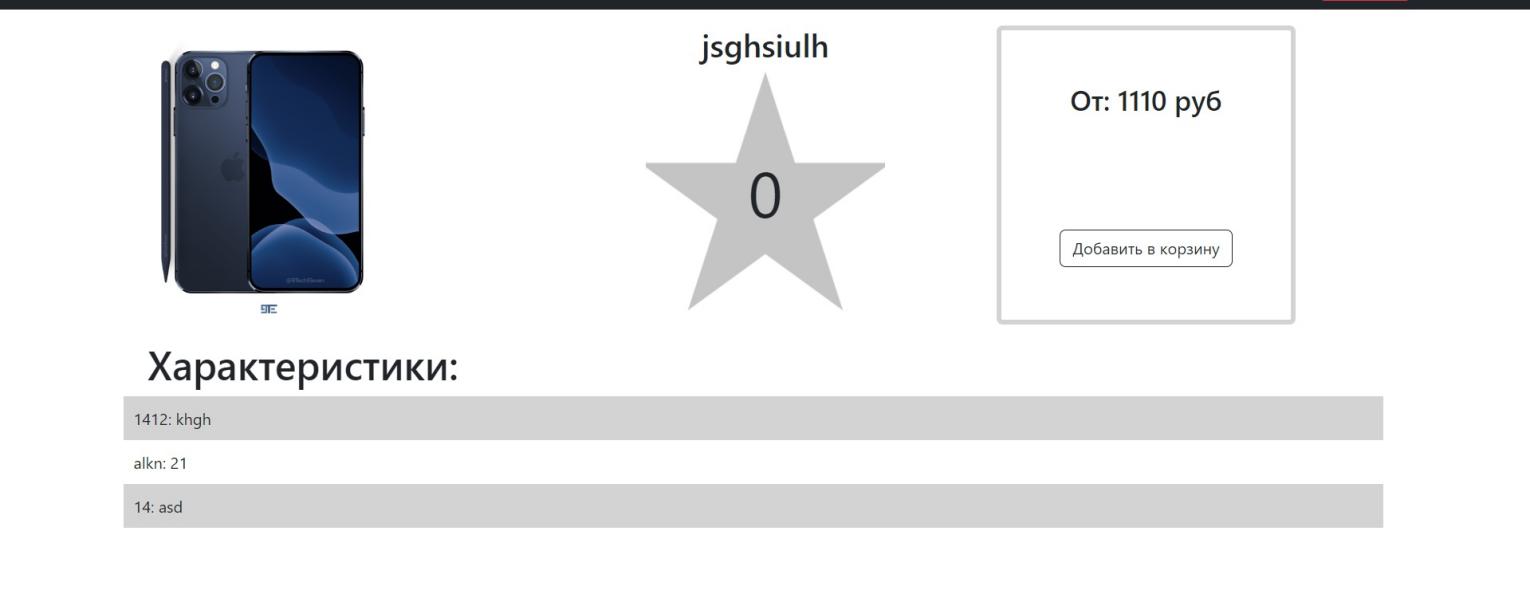
На рисунке 2.6 есть удобная ссылка на окно авторизации, это делает окно регистрации/авторизации более удобным.

Рисунок 2.7 – Описание товара на сайте «Lucrum»

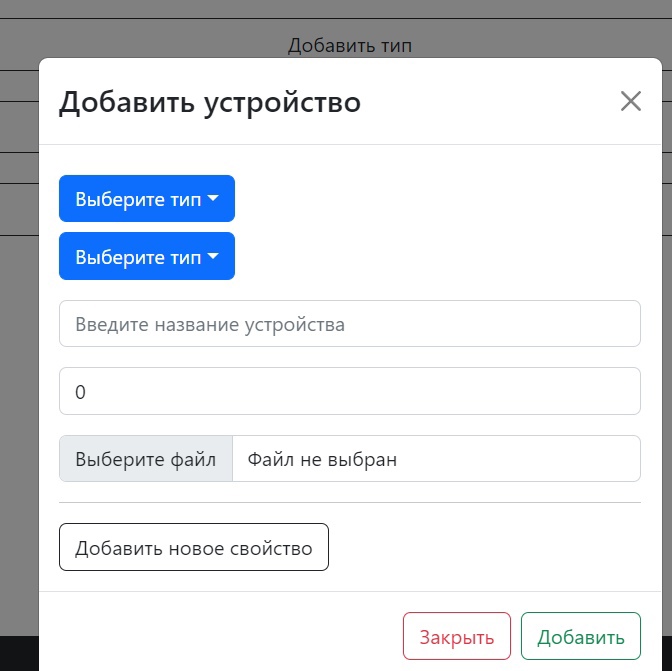
На рисунке 2.7 представлена информация о товаре. Любой пользователь может посмотреть информацию о товаре, его характеристиках и цене.

Рисунок 2.8 – Админ панель на сайте «Lucrum»

На сайте так же имеется админ панель, она представлена на рисунке 2.8, с помощью её администратор сайта может заполнить данные о товаре, бренде или типе, добавлять различные характеристики, вставлять фото или указывать тип или бренд для товара.

**2.7 Вывод**

Данный проект разработан для продвижения компании «Lucrum» и развития интернет маркетинга. Для того, чтобы завлечь пользователя на сайт, был разработан уникальный дизайн и логотип агентства, подобрана спокойной теплая палитра цветов, а также создан удобный интерфейс для быстрого поиска информации.

Были созданы макеты будущих страниц сайта. Макет – это план сайта с добавлением цветов, изображений, типографики. Его основная задача – создать стиль и настроение, показать визуализацию и общие образы. На этом этапе были внесены последние изменения во внешний вид сайта, подобрана цветовая палитра, а также шрифты, фотоматериалы и оформление отдельных элементов сайта.

Было определено стилевое оформление сайта, то есть основная цветовая гамма, шрифтовое оформление, элементы пользовательского интерфейса, а также эффекты и переходы, придающие динамики страницам.

# Реализация структуры веб-сайта

**3.1. Структура серверной части проекта**

Бэкенд — серверная сторона сайта или приложения, которая отвечает за его функционирование и хранение данных. Начинается серверная часть с подключения к базе данных, в нашем случае - подключение к Postgresql, а также создания самой базы данных, определение связей.

|  |
| --- |
| PORT=5000  DB\_NAME=lucrum\_shop  DB\_USER=postgres  DB\_PASSWORD=123  DB\_HOST=localhost  DB\_PORT=5432  SECRET\_KEY=random\_secret\_key123 |

Листинг 3.1 – Содержнаие .env файла

В листинге 3.1 содержится .env файл, в котором определены основные коснтаты, которые применяются ко всему остальному коду.

|  |
| --- |
| const {Sequelize} = require('sequelize')  module.exports = new Sequelize(  process.env.DB\_NAME,  process.env.DB\_USER,  process.env.DB\_PASSWORD,  {  dialect: 'postgres',  host: process.env.DB\_HOST,  port: process.env.DB\_PORT  }  ) |

Листинг 3.2 – Подключение к Postgresql

В листинге 3.2 представлено подключение к базе данным Postgresql, с помощью которого будет происходить управление проектом.

В приложении Б описывается база данных, то какие будут таблицы, primary key, связи и т.д.

|  |
| --- |
| const jwt = require('jsonwebtoken')  module.exports = function (req, res, next){  if ( req.method === "OPTIONS"){  next()  }  try{  const token = req.headers.authorization.split(' ')[1]  if (!token) {  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"})  }  } |

Листинг 3.3 – Описание MiddleWare

Важной частью серверной части является MiddleWare, он представлен в листинге 3.3, он нужен для обработки запросов и ответов. Допустим данный листинг производит проверку jwt токена, а значит и нахождение зарегистрированного пользователя.

|  |
| --- |
| 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 |

Листинг 3.4 – Описание ApiError

В листинге 3.4 для удобста дальнейшего пользования баззой данных, был создан класс для обработки ошибок.

В проекте реализовано много функций Controller, для примера был взят userController, он представлен в приложении В, внутри его описана основная функция регистрации и авторизации.

|  |
| --- |
| 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 |

Листинг 3.5 – Описание userRoutes

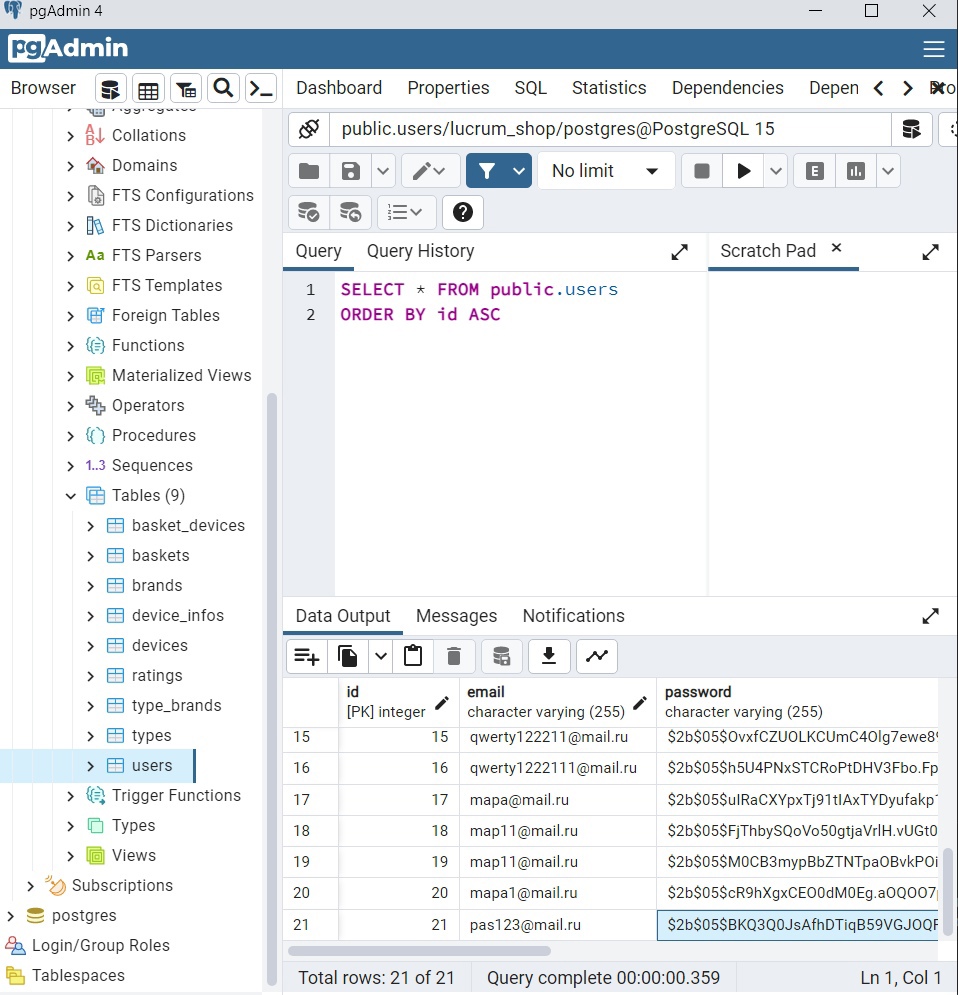
В листинге 3.5 внутри userRoutes определены запросы, которые будут отправлятся на сервер, где будут обработаны и пользователь будет получит что-то в ответ.

Рисунок 3.1 – Программа PGAdmin4

Также для удобства, была скачана программа PGAdmin4, она представлена на рисунке 3.1, внутри этой программы можно взаимодействовать с данными из базы данных напрямую и с помощью SELECT-запросов изменять ее.

**3.2. Структура клиентской части проекта**

Клиентская часть — это интерфейс с набором функций, с которым взаимодействует пользователь.

|  |
| --- |
| import { observer } from "mobx-react-lite";  import React, { useContext, useEffect, useState } from "react";  import { Spinner } from "react-bootstrap";  import { BrowserRouter } from "react-router-dom";  import { Context } from ".";  import AppRouter from "./components/AppRouter";  import Footer from "./components/Footer";  import NavBar from "./components/NavBar";  import { check } from "./http/userAPI";  const App = observer(() => {  const {user} = useContext(Context)  const {loading, setLoading} = useState(true)  useEffect(() => {  check().then(data => {  user.setUser(data)  user.setIsAuth(true)  })  }, [])  if (loading) {  return <Spinner animation={"grow"}/>  }  return (  <BrowserRouter>  <NavBar/>  <AppRouter />  <Footer/>  </BrowserRouter>  );  });  export default App; |

Листинг 3.6 – App.js файл

В листинге 3.6, внутри App.js описываются основные элементы приложения.

|  |
| --- |
| 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)  } |

Листинг 3.7 – User API

В листинге 3.7, внутри UserAPI описываются функции для отправления данных на сервер.

|  |
| --- |
| const Shop = observer(() => {  const {device} = useContext(Context)  useEffect(() => {  fetchTypes().then(data => device.setTypes(data))  fetchBrands().then(data => device.setBrands(data))  fetchDevices(null, null, 1, 24).then(data => {  device.setDevices(data.rows)  device.setTotalCount(data.count)  })  }, [])  useEffect(() => {  fetchDevices(device.selectedType.id, device.selectedBrand.id, device.page, 24).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; |

Листинг 3.8 – Струтура Shop

В листинге 3.8 описывается основная структура Shop страницы, внутри также используется bootsrap для удобства.

|  |
| --- |
| export const authRoutes = [  {  path: ADMIN\_ROUTE,  Component: Admin  },  {  path: BASKET\_ROUTE,  Component: Basket  },  ]  export const publicRoutes = [  {  path: SHOP\_ROUTE,  Component: Shop  },  {  path: MAIN\_ROUTE,  Component: Main  },  {  path: LOGIN\_ROUTE,  Component: Auth  },  {  path: REGISTRATION\_ROUTE,  Component: Auth  },  {  path: DEVICE\_ROUTE + '/:id',  Component: DevicePage  },  ] |

Листинг 3.9 – Структура Routes

Внутри листинга 3.9, описывается логика и определения уровня доступа для разных пользователей

**3.3. Выводы**

На данном этапе была реализована структура приложения, описана логика серверной и клиентской части. Был разработан основной функционал приложения. Было определено стилевое оформление сайта, использован SCSS, а также приложение было подготовлено к следующему этапу – к тестированию.

**4. Тестирование веб-сайта**

**4.1. Адаптивный дизайн веб-сайта**

Адаптивность — одно из ключевых требований, предъявляемых к современным сайтам. Ресурс должен одинаково хорошо демонстрироваться на экране компьютера, планшете, смартфоне. Для этого создается адаптивный дизайн и используется адаптивная верстка. В итоге процесс веб-разработки усложняется, требует дополнительных ресурсов. Но результат стоит вложений: сайт отлично позиционируется на всех основных типах устройств, что гарантирует полноценный охват аудитории.

Адаптивность — это способность сайта «подстраиваться» под различные технические условия (а именно, под размеры экрана пользовательского устройства). Адаптивный сайт хорошо смотрится и на десктопном мониторе (обычный ПК), и на планшетном компьютере, и на экране смартфона. Причем качество отображения не зависит ни от диагонали, ни от позиционирования экрана.

Адаптивная верстка — это такой способ создания веб-страниц, при котором они автоматически подстраиваются под размеры и ориентацию экрана устройства, а их дизайн варьируется в зависимости от действий пользователя.

Цель адаптивной верстки — добиться того, чтобы сайт оставался удобным и обеспечивал конверсию при загрузке на разных устройствах.

В проекте используются основное медиа-правило на 990px.

|  |  |
| --- | --- |
| xO-F6yTZ2ZQ | K_glNTUOy40 |
| а) | б) |

Рисунок 4.1 – Вид веб-страницы в а) мобильная версия; б) компьютерная версия

Как пример на рисунок 4.1 представлена часть основного варианта адаптивности сайта.

**4.2. Кроссбраузерность веб-сайта**

В работе периодически использовались разные браузеры: Google Chrome, Яндекс.Браузером, Mozilla Firefox, Opera. Были замечены некоторые изменения в зависимости от браузер, где-то эти отличия мало заметны и никак не влияют на просмотр и чтение, но иногда возникают определенные сложности с картинками, баннерами, кнопками команд и даже текстами. Чаще всего с такими проблемами сталкиваются владельцы смартфонов и планшетов. Большинство посетителей, встретившись с подобными неудобствами, просто покидают проблемный ресурс и переключаются на альтернативные источники информации.

Если для рядового посетителя неудобство просмотра определенного веб-ресурса скорее относится к разряду мелкой секундной проблемы, то регулярное возникновение отказов от посещения сайта для его владельцев не сулит ничего хорошего. Падает конверсия, снижается рейтинг ресурса в поисковых системах.

Вендорные префиксы - это специальные приставки к названию CSS свойства, заточенные под конкретный браузер, которые позволяют ему понимать экспериментальные CSS свойства и одновременно игнорировать записи, предназначенные для других браузеров.

# 4.3 Ручное тестирования

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления ajax запросов, их обработки frontend сервером, а также правильности отображение представлений.

Рассмотрим сценарий принятия участия в мероприятии. Для начала необходимо зарегистрироваться. При заполнении формы неправильными данными должно быть получено оповещение об ошибки. Заполнение формы неправильными данными и уведомление об ошибке изображено на рисунке 4.2.

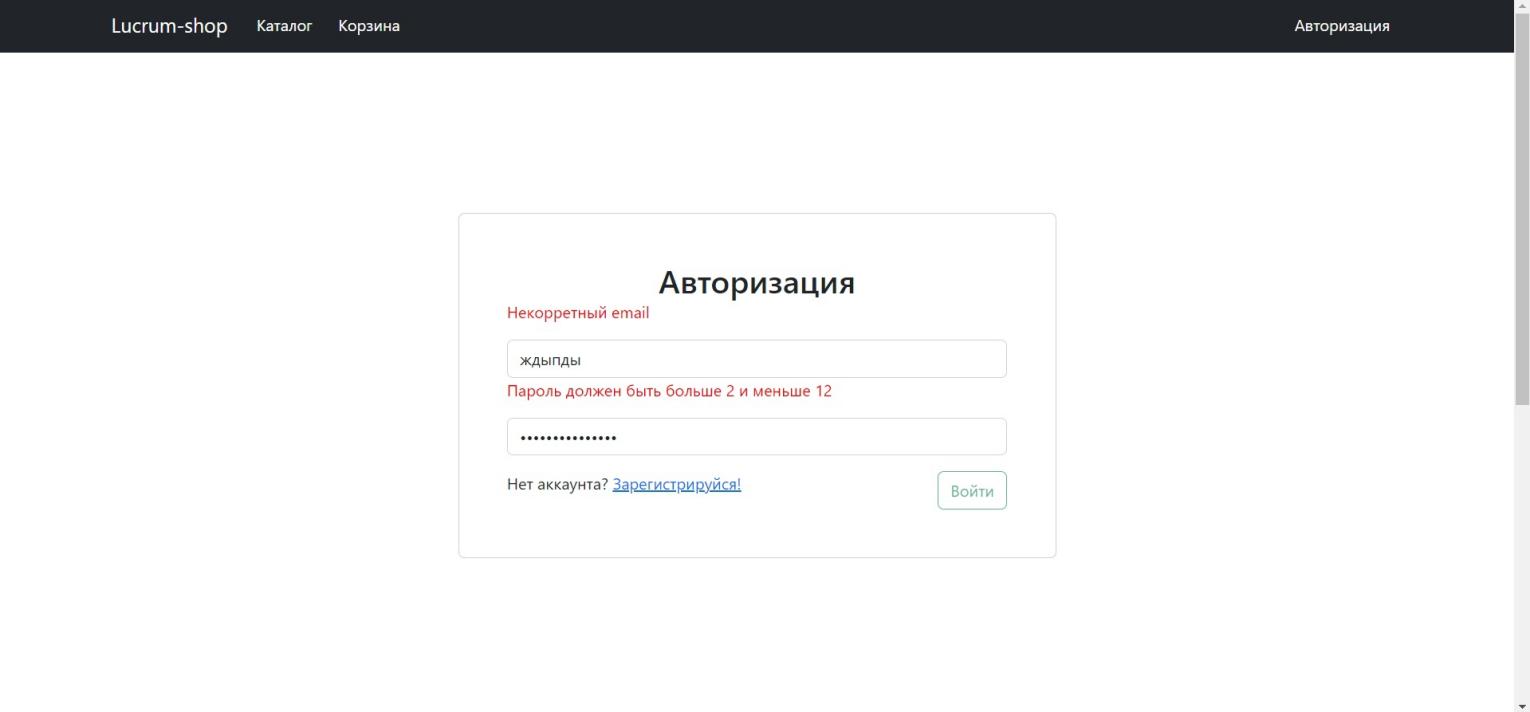


Рисунок 4.2 – Форма заполнения неверными данными

Теперь заполним формы правильными данными и завершим регистрацию. После успешного завершения регистрации должна быть изображена форма авторизации. Успешная регистрация и переход изображена на рисунке 4.3.

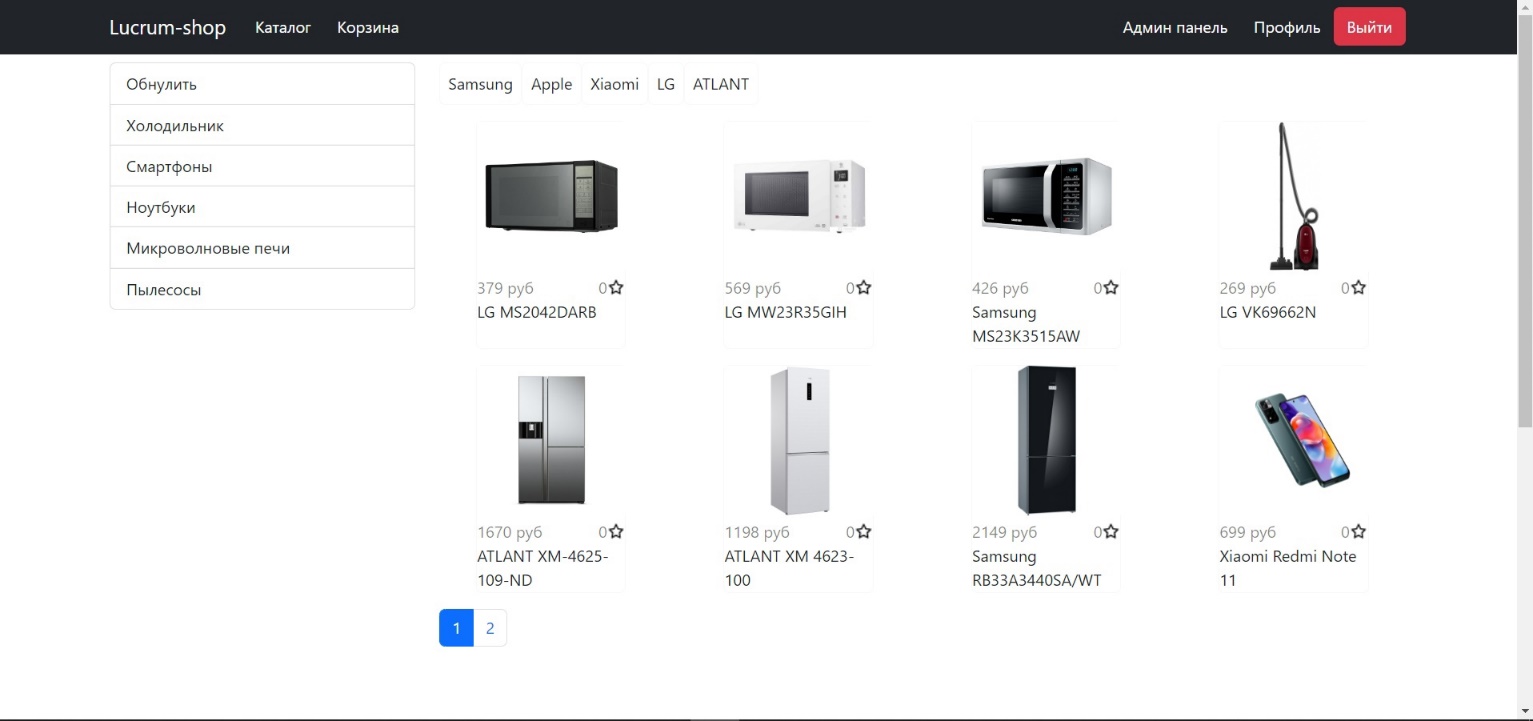


Рисунок 4.3 – Успешная регистрация

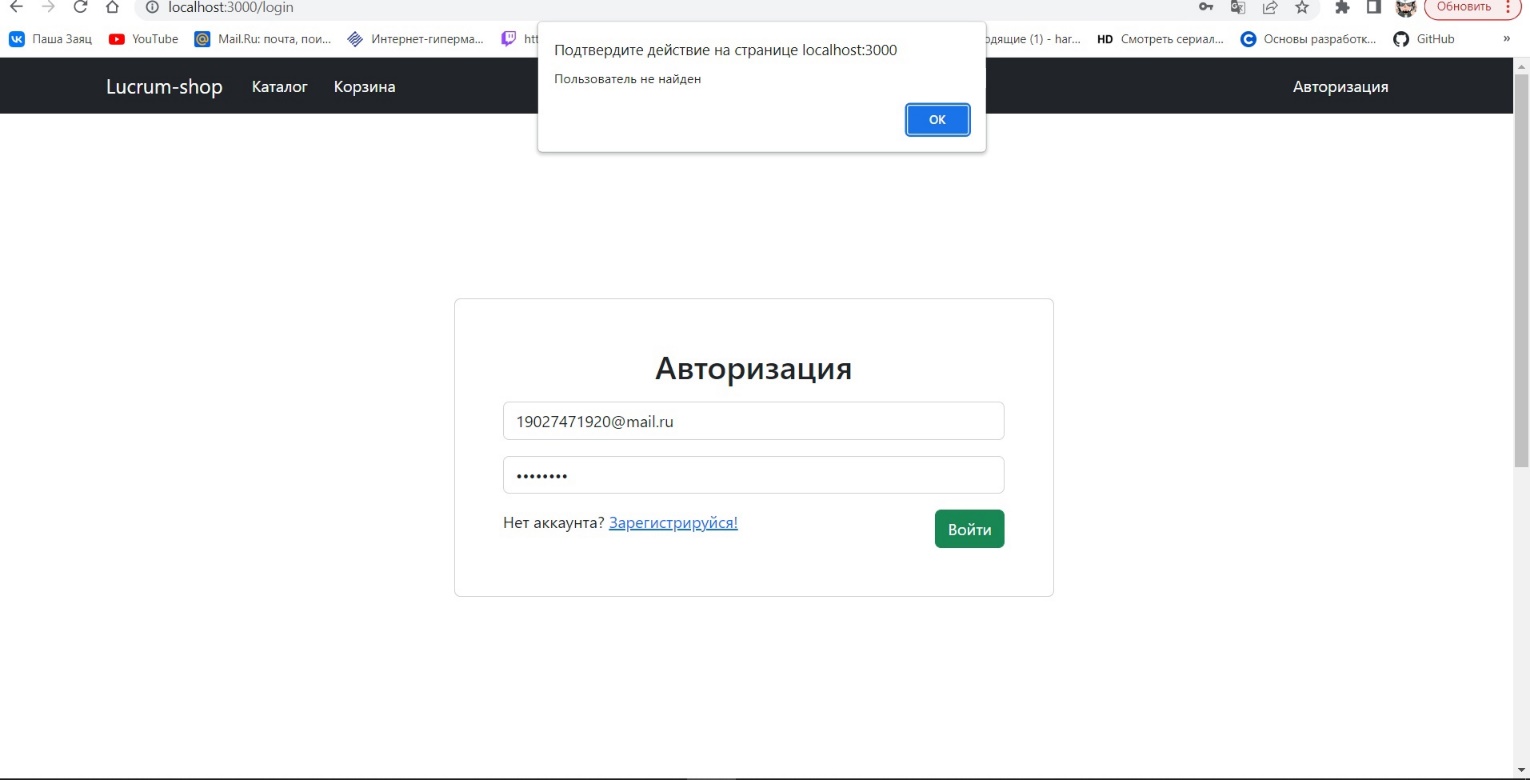
Попробуем зайти под несуществующим пользователем. При неверном введение данным, получим оповещение об ошибке. Оповещение об ошибке изображено на рисунке 4.4.

Рисунок 4.4 – Неправильный ввод данных для входа и оповещение об ошибке

Таким образом было проведено ручное тестирование интернет магазина.

**4.3. Вывод**

На данном этапе было проанализировано поведение веб-сайта при использовании с разных устройств, а также веб-ресурс был протестирован в большом количестве браузеров. По итогу выполнения данного этапа для веб-сайта был разработан адаптивный дизайн и кроссбраузерность. Не было обнаружено никаких ошибок, которые могли бы повлиять на работу сайта.

# 5. Руководство пользователю

Данное приложение предоставляет возможность приятно пользоваться функциями реального чата, также прослушивать приятную музыку, также реализован приятный интерфейс, понятен даже новичку. При входе в приложение будет предложено войти, либо же зарегистрироваться новому пользователю. После входа будут доступны все функции приложения.

На странице Main будет удобный слайдер, с акциями и выгодными предложениями, информация о компании, контакты, а также верхнее меню навигации/регистрации.

На странице Basket будут расположены карточки продуктов, форма оформления заказа, информация о оплате.

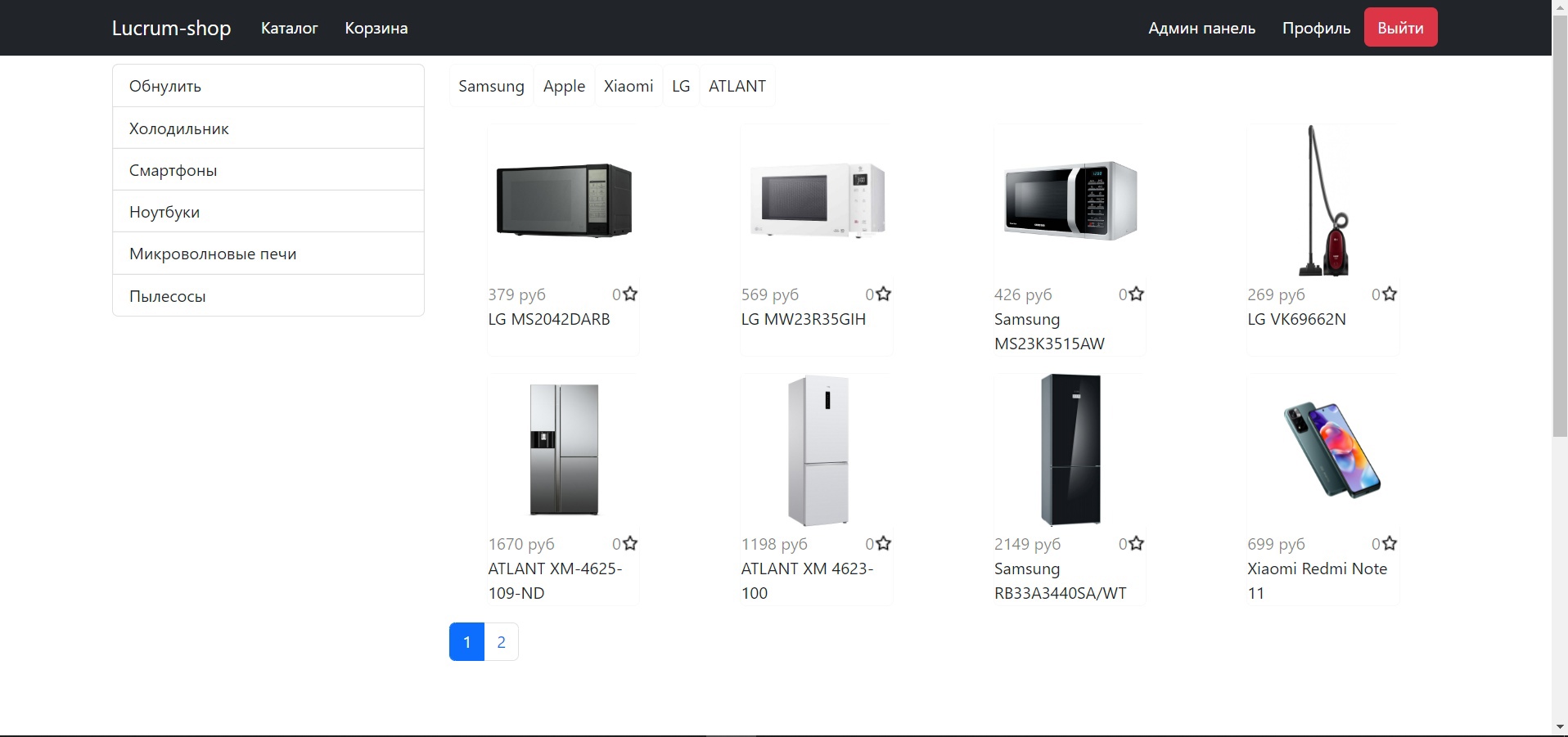
На странице Profile будут расположены личные данные пользователя. Здесь пользователь сможет изменить свои данные, удалить аккаунт.

На странице Login будут расположены форма входа, ссылка на страницу регистрации.

На странице Register будут расположены форма регистрации, ссылка на страницу входа.

На странице Catalog будет представлен перечень товаров, которые можно добавить в корзину, также в каталоге есть функция структурирование данных, то есть поиск товара по определенному бренду или типу.

Любой посетитель в самом начале имеет право регистрироваться или авторизироваться. Для того, чтобы зарегистрироваться следует осуществить переход с главной страницы на вкладку «Авторизаця». Главная страница данного приложения изображена на рисунке 5.1.

Рисунок 5.1 – Главная страница приложения

Для того, чтобы совершить регистрацию пользователю необходимо ввести некоторые обязательные поля, такие как пароль, электронную почту. Переход на страницу с формой «Регистрация» изображен на рисунке 5.2.

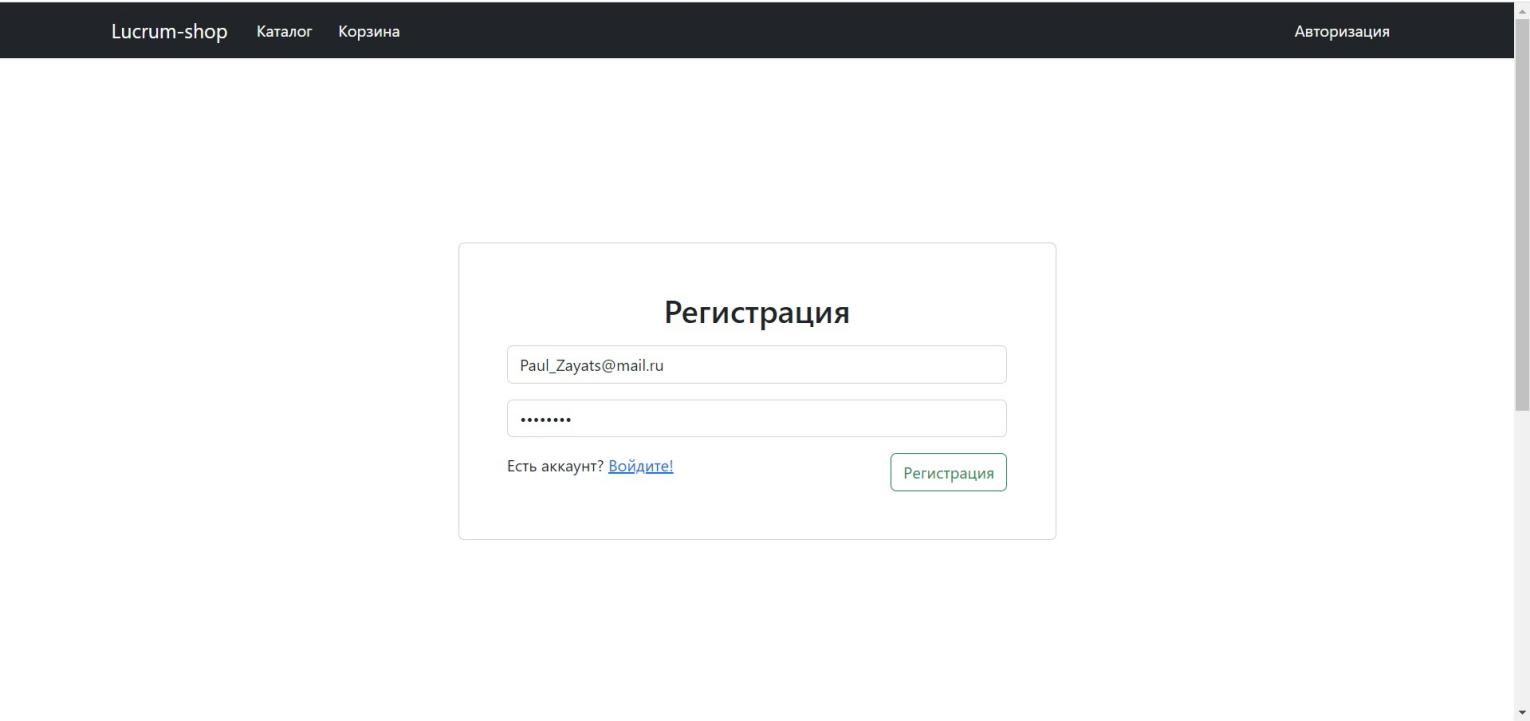


Рисунок 5.2 – Форма регистрация

Если пользователь уже зарегистрирован, то он может воспользоваться кнопкой «Log In» для того, чтобы выйти в уже существующий аккаунт. Экран входа в аккаунт изображен на рисунке 5.3.

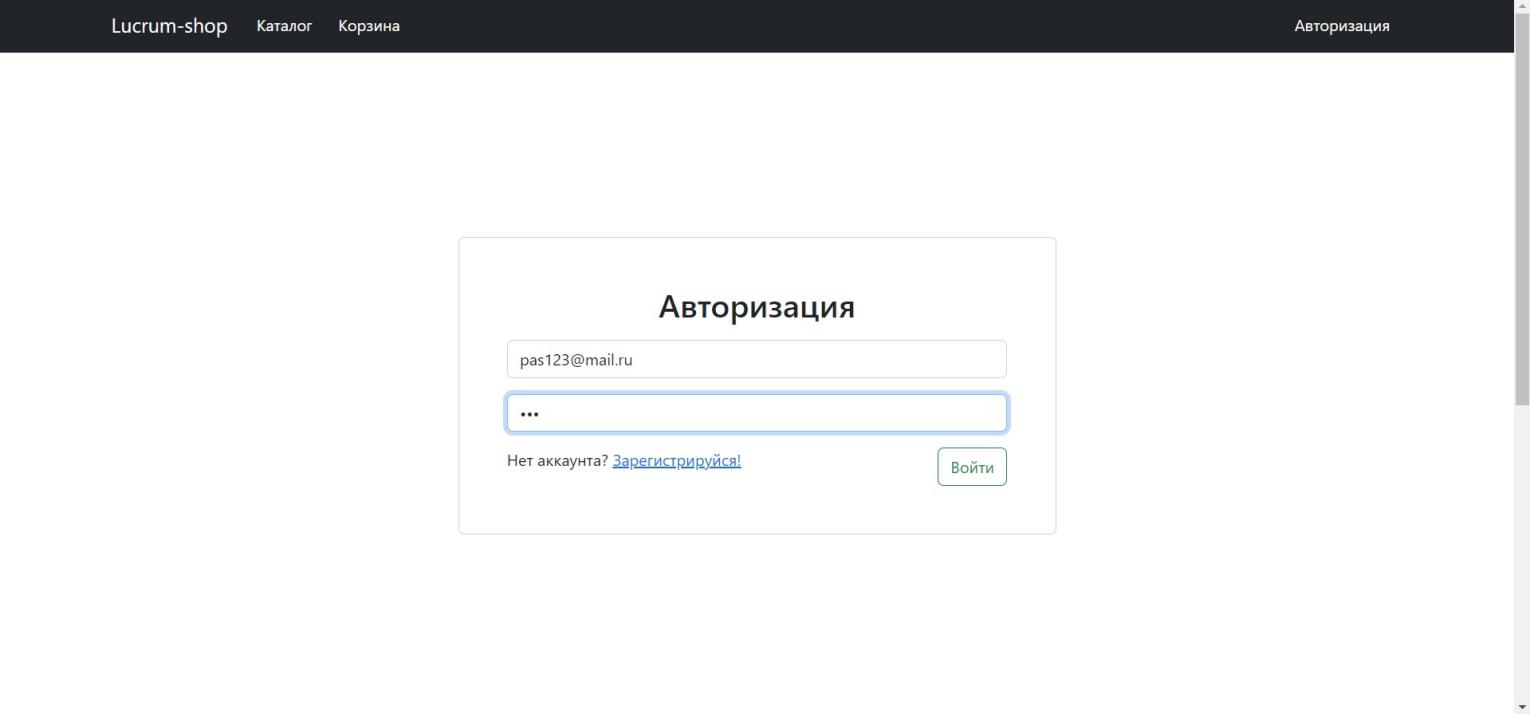
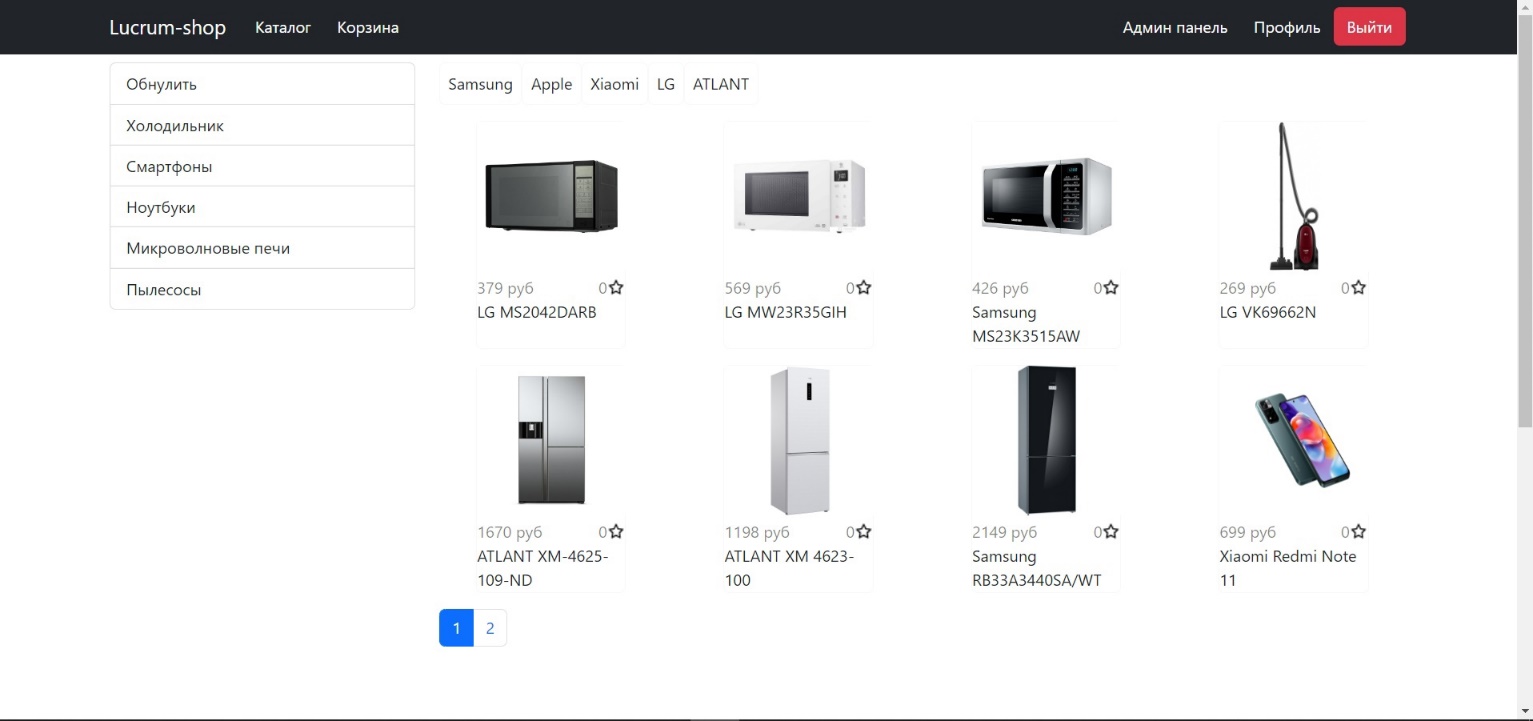


Рисунок 5.3 – Страница авторизации

После входа в аккаунт, пользователь может просмотреть каталог товаров, просмотреть характеристики и добавить товар в корзину. Каталог товаров представлен на рисунке 5.4

Рисунок 5.4 – Страница каталога товаров

Перейдя на товар, можно увидеть его подробную характеристику, рейтинг и фото, а также добавить товар в корзину. Информация о товаре представлена на рисунке 5.5

Рисунок 5.5 – Страница информации о товаре

Перейдя в корзину, пользователь видит все товары, которые он добавил, а также может видеть сумму всех товаров, и имеет возможность удалить 1 определенный товар или сразу все. Корзина товаров представлена на рисунке 5.6

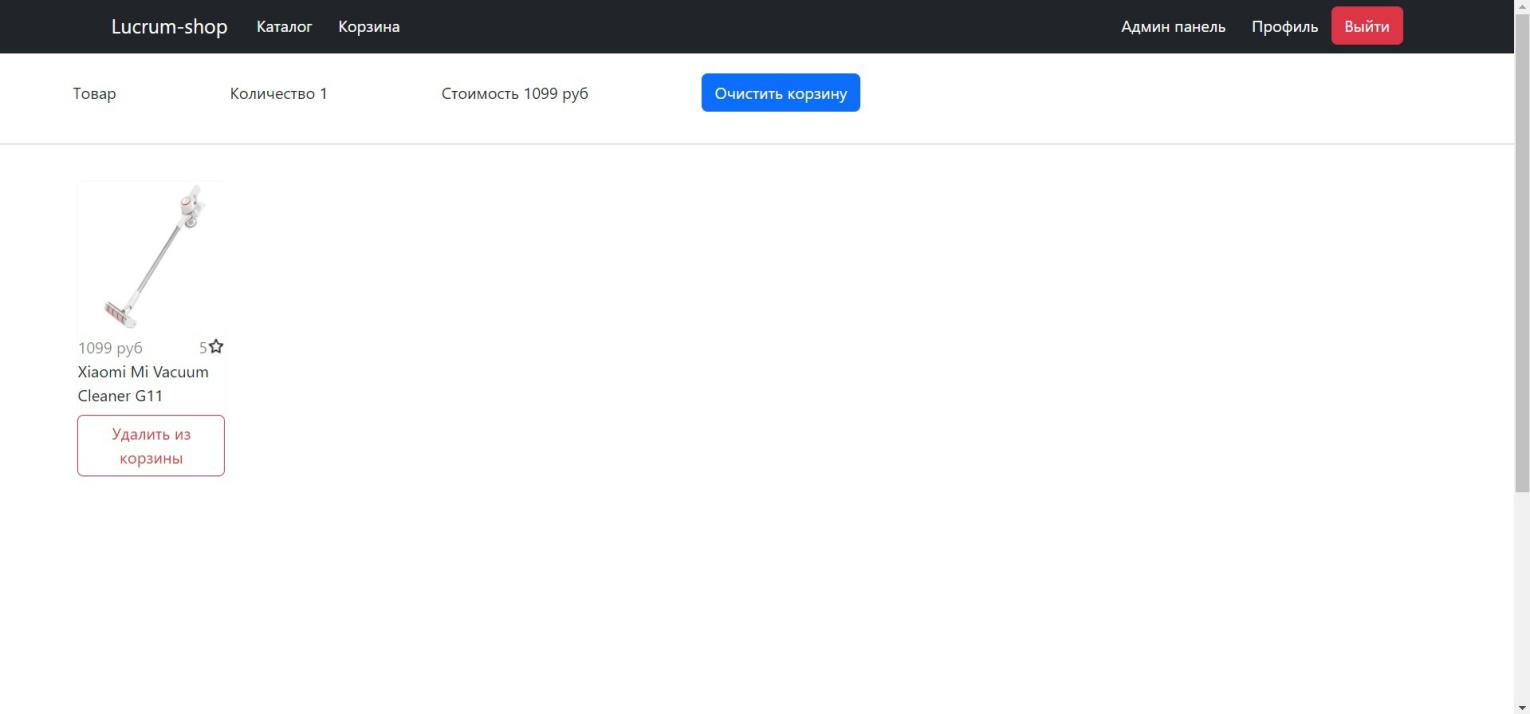


Рисунок 5.6 – Страница корзины пользователя

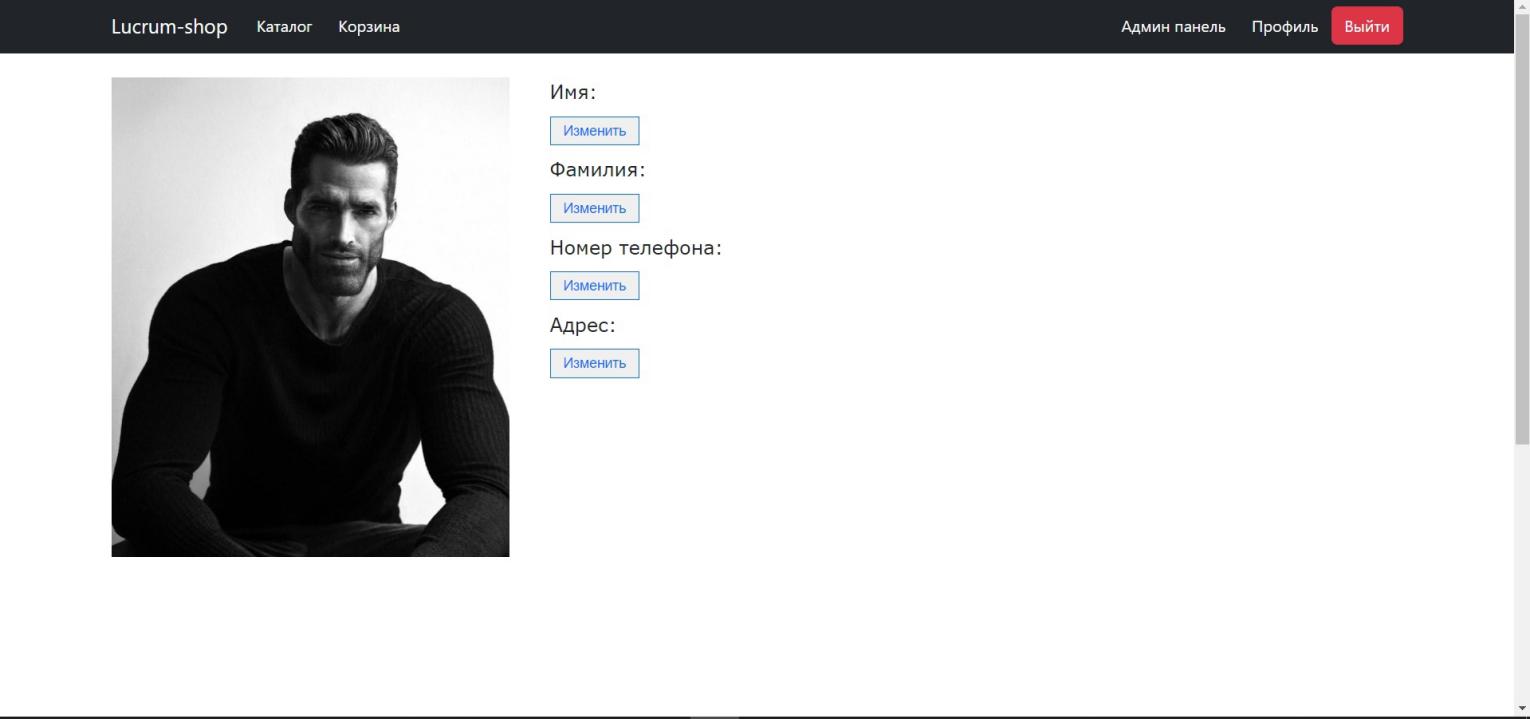
Также у пользователя есть возможность изменить некоторые свои данные, зайдя в свой профиль. Страница профиля представлена на рисунке 5.7

Рисунок 5.7 – Страница профиля пользователя

Таким образом, пользователь может пользоваться интернет магазином.

**Заключение**

Целью на курсовое проектирование было создание приложения бытовой техники «Lucrum». Создание веб-сайта, который в основном облегчит контакт с компанией и будет предоставлять информацию. Цель состояла в том, чтобы обеспечить отличный пользовательский опыт и увеличить продажи. Помимо целевой страницы на веб-сайте были реализованы страницы, такие как: «Корзина», «Каталог», «Вход/Регистрация», «Профиль», «Админ панель».

Для реализации макета веб-страницы были исследованы сайты схожей тематикой и исследованы недостатки веб-приложения. Показана и предложено иные реализации блоков веб-страницы.

Visual Studio Code стала отличной IDE для комфортного и ускоренного написания кода для веб-страницы. А именно выбор необходимых плагинов и цветовой темы.

По итогам реализации веб-сайта, было принято решения о тестировании веб-сайта. В результате тестирования была обнаружена неспособность веб-браузера(Internet Explorer) работать с практическими всеми функциями сайта.

Для достижения поставленной цели для курсового проекта функционально были реализованы следующие задачи:

* + Анализ аналогичных решений;
  + Выбор способа верстки.
  + Выбор стилевого оформления.
  + Разработка логотипа.
  + Разработка пользовательских элементов.
  + Разработка спецэффектов.
  + Разработана база данных.
  + Разработана серверная часть.
  + Разработана клиентская часть.
  + Добавление веб-сайту адаптивности.
  + Кроссбраузерность веб-сайта.
  + Создание руководства пользователя.

Результатом курсового проекта стал готовый к работе веб-сайта «Lucrum». В дальнейшем разработанный нами веб-сайт можно будет использовать в реальной жизни. Так же возможно расширение списка предоставляемых услуг, акций, новостей. Данный ресурс стабилен в работе и не требует высоких знаний для работы с ним.

Таким образом, из выше сказанного, можно сделать вывод, что в курсовом проекте раскрывается все теоретические и практические аспекты темы и достигнуты поставленные цели и задачи роботы.

**Список использованных литературных источников.**

1. Онлайн гипермаркет «21 век» [[Электронный ресурс]](https://www.21vek.by/). – Режим доступа: https://www.21vek.by/ – Дата доступа: 21.12.2022.
2. Онлайн гипермаркет «5 элемент» [[Электронный ресурс]](https://5element.by/). – Режим доступа: <https://5element.by/> – Дата доступа: 21.12.2022.
3. Онлайн гипермаркет «Walmart» [[Электронный ресурс]](https://www.walmart.com/). – Режим доступа: [hhttps://www.walmart.com/](https://5element.by/) – Дата доступа: 21.12.2022.
4. Шрифты онлайн [[Электронный ресурс]](https://fonts-online.ru/) – Режим доступа: https://fonts-online.ru/ – Дата доступа: 21.12.2022.
5. Автопрефиксер CSS онлайн [[Электронный ресурс]](https://autoprefixer.github.io/ru/) – Режим доступа: <https://autoprefixer.github.io/ru/> – Дата доступа: 21.12.2022.

# Приложение А Макет структуры веб-сайта

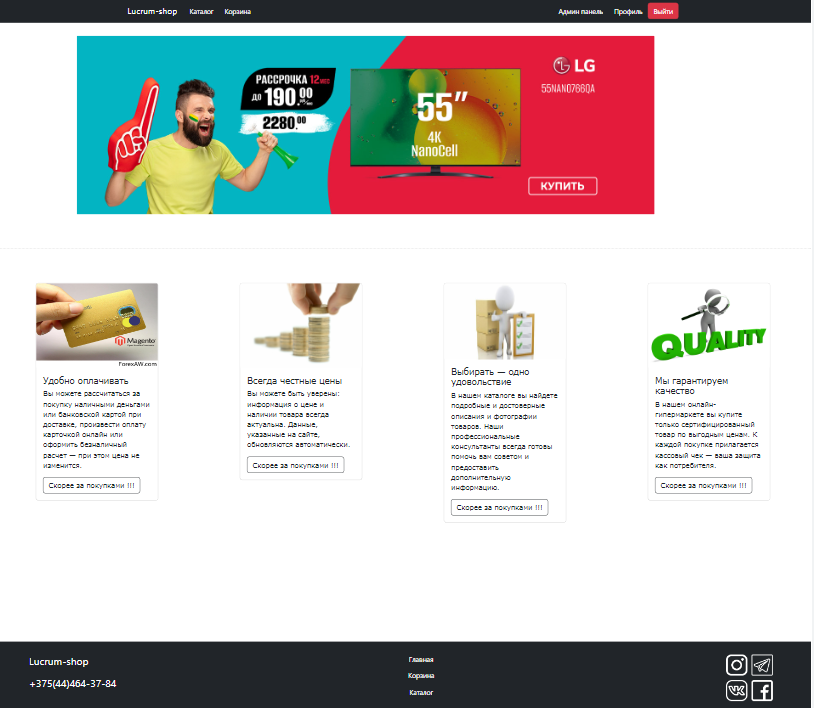


Рисунок 1-макет страницы “Main”

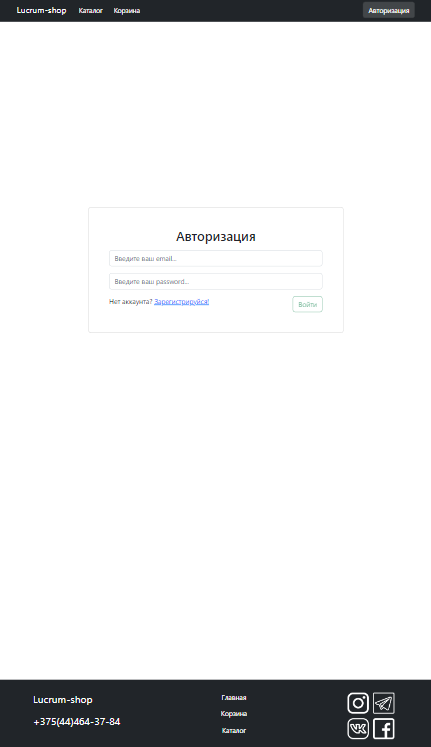


Рисунок 2-макет страницы “Auth”

Рисунок 3-макет страницы “Admin”

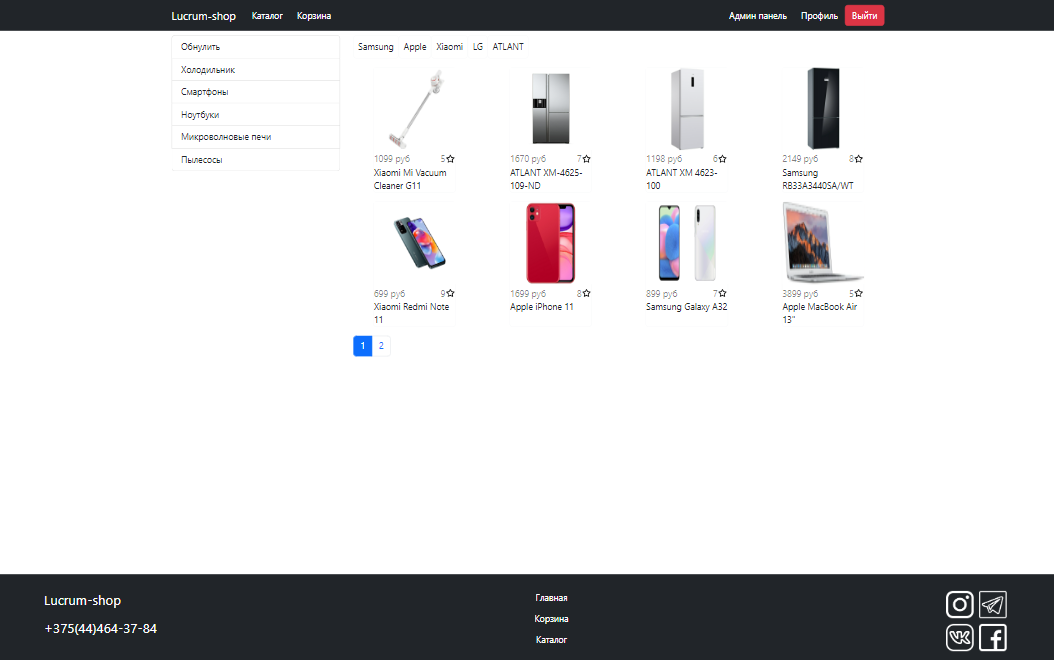
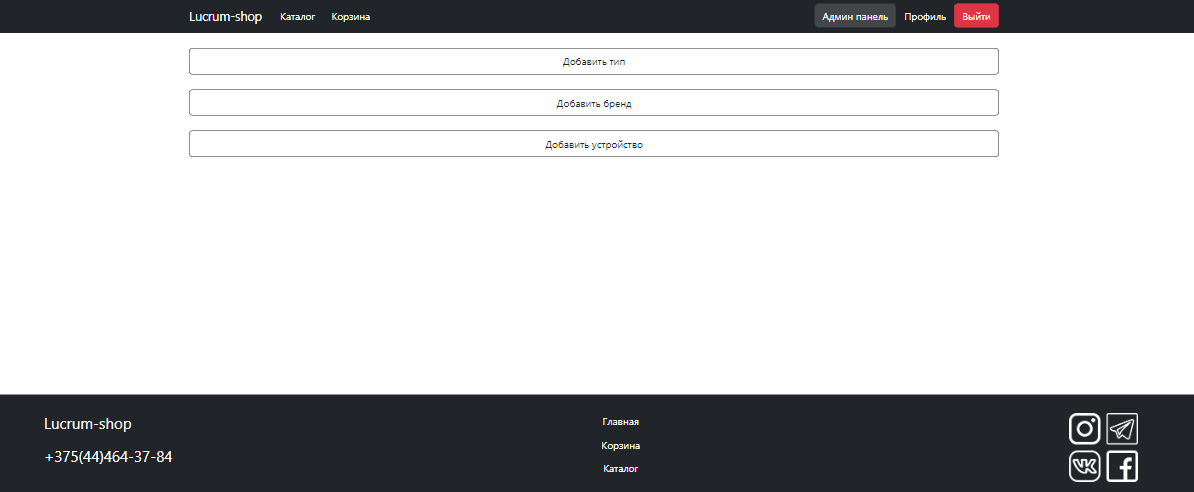


Рисунок 4-макет страницы “Shop”

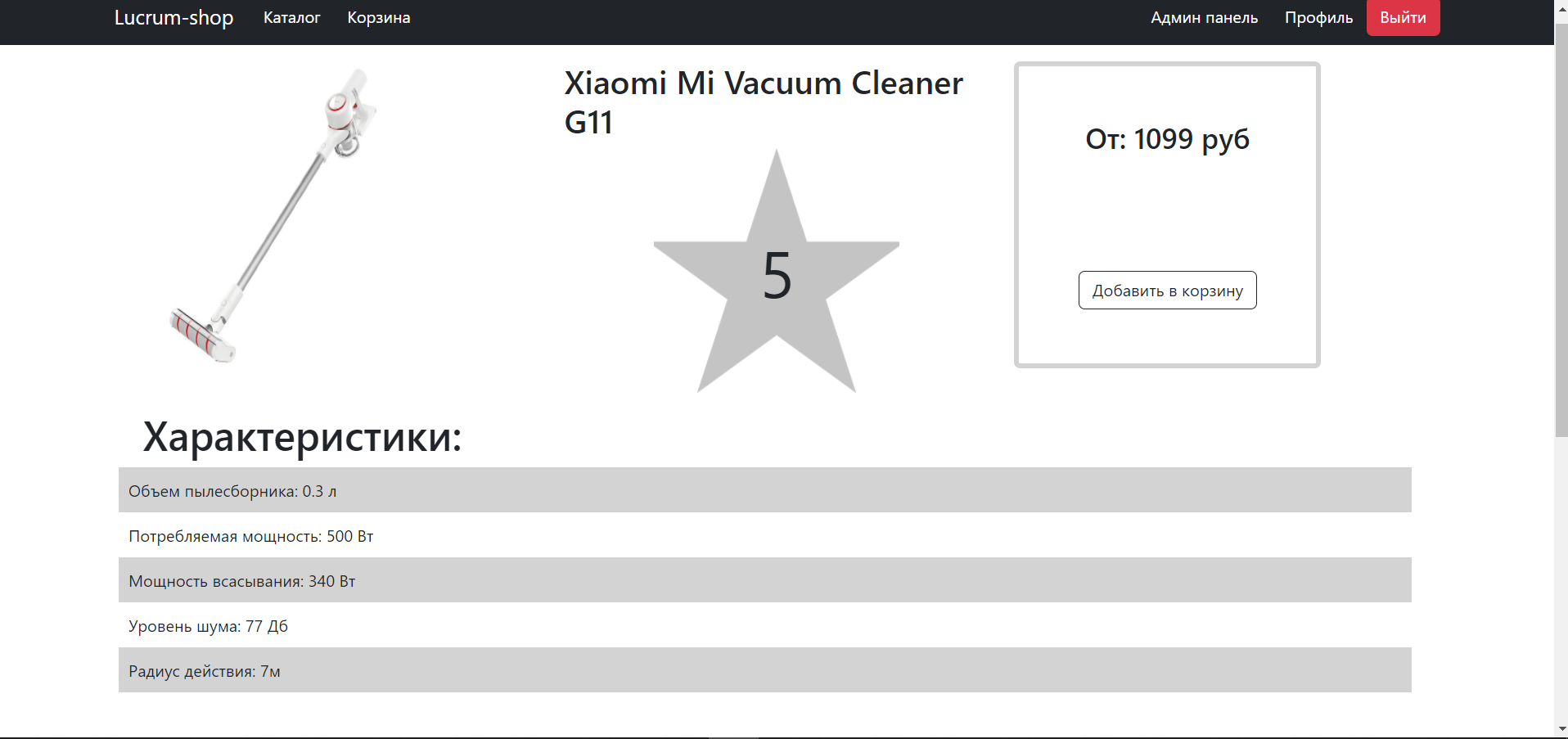


Рисунок 5-макет страницы “DevicePage”

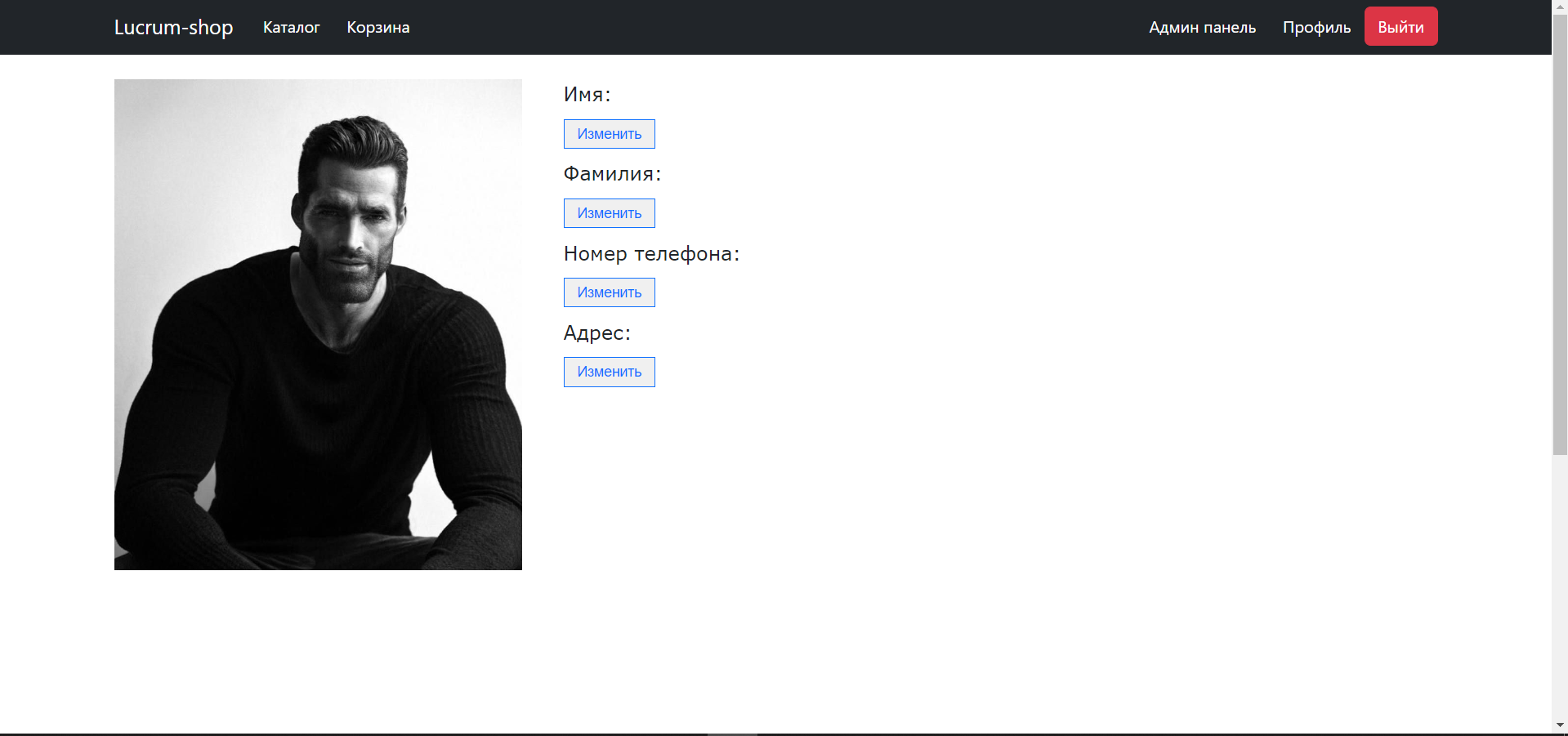
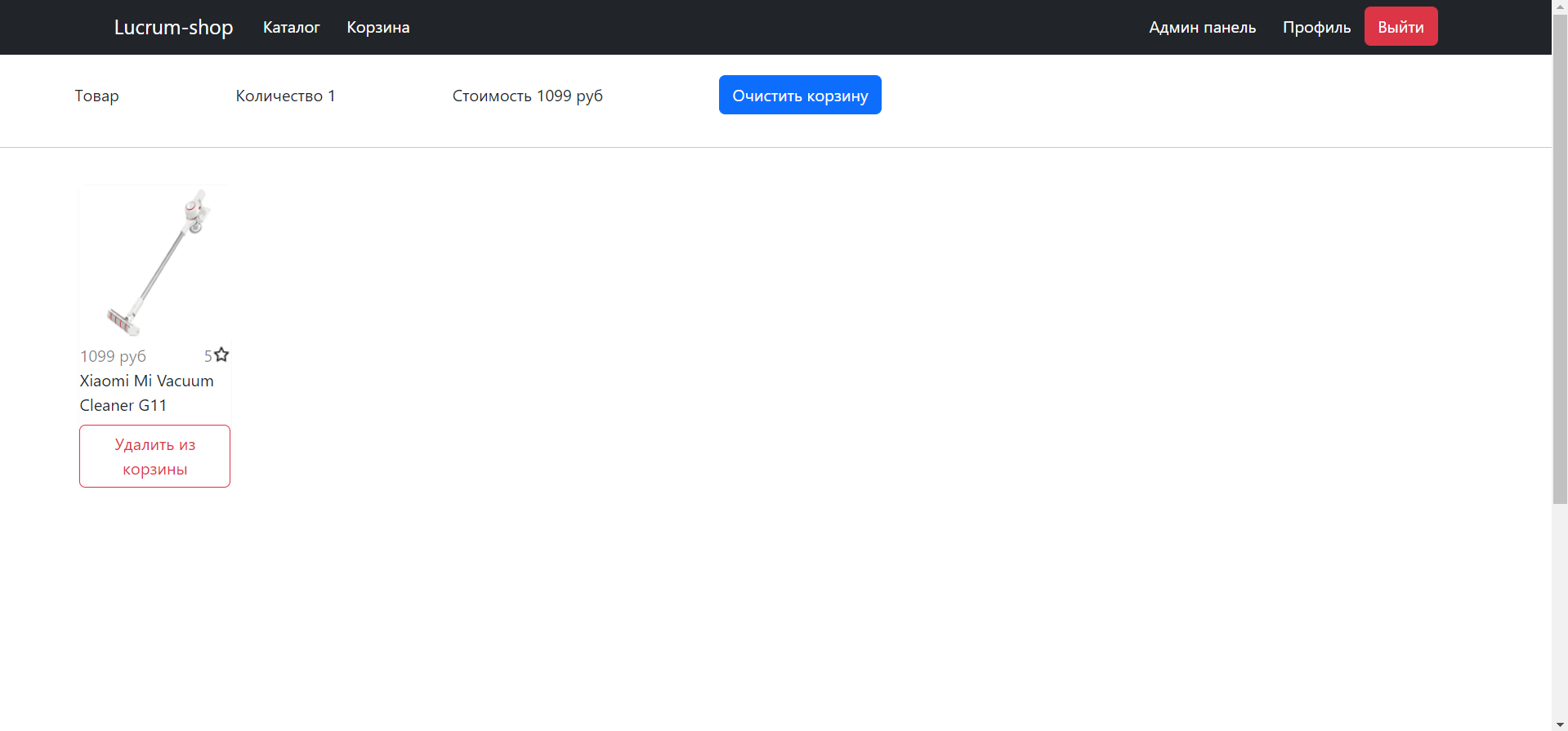


Рисунок 6-макет страницы “Profile”

Рисунок 7-макет страницы “Basket”



# Приложение Б Листинг описание базы данных

|  |
| --- |
| const sequelize = require('../db')  const {DataTypes, DECIMAL} = require('sequelize')  const User = sequelize.define('user', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  email: {type: DataTypes.STRING, unique:true,},  password: {type: DataTypes.STRING},  role: {type: DataTypes.STRING, defaultValue: "USER"},  })  const Basket = sequelize.define('basket', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  })  const BasketDevice = sequelize.define('basket\_device', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  })  const Device = sequelize.define('device', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  name: {type: DataTypes.STRING, unique: true, allowNull: false},  price: {type: DataTypes.INTEGER, allowNull: false},  rating: {type: DataTypes.INTEGER, defaultValue: 0},  img: {type: DataTypes.STRING, allowNull: false},  })  const Type = sequelize.define('type', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  name: {type: DataTypes.STRING, unique: true, allowNull: false},  })  const Brand = sequelize.define('brand', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  name: {type: DataTypes.STRING, unique: true, allowNull: false},  })  const Rating = sequelize.define('rating', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  rate: {type: DataTypes.INTEGER, allowNull: false},  })  const DeviceInfo = sequelize.define('device\_info', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  title: {type: DataTypes.STRING, allowNull: false},  description: {type: DataTypes.STRING, allowNull: false},  })  const TypeBrand = sequelize.define('type\_brand', {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  })  User.hasOne(Basket)  Basket.belongsTo(User)  User.hasMany(Rating)  Rating.belongsTo(User)  Basket.hasMany(BasketDevice)  BasketDevice.belongsTo(Basket)  Type.hasMany(Device)  Device.belongsTo(Type)  Brand.hasMany(Device)  Device.belongsTo(Brand)  Device.hasMany(Rating)  Rating.belongsTo(Device)  Device.hasMany(BasketDevice)  BasketDevice.belongsTo(Device)  Device.hasMany(DeviceInfo, {as: 'info'})  DeviceInfo.belongsTo(Device)  Type.belongsToMany(Brand, {through: TypeBrand})  Brand.belongsToMany(Type, {through: TypeBrand})  module.exports = {  User,  Basket,  BasketDevice,  Device,  Type,  Brand,  Rating,  TypeBrand,  DeviceInfo,  } |

# Приложение В Листинг описания 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('Некоректный 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}})  console.log(user)  if(!user) {  return next(ApiError.internal('Пользователь не найден'))  }  let comparePassword = bcrypt.compareSync(password, user.password)  console.log(comparePassword)  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() |