МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»
(ГАПОУ СПО ОКЭИ)

КУРСОВОЙ ПРОЕКТ

ОКЭИ 09.02.07. 4323. №5 ПЗ
(код документа)

Разработка веб-приложения на REACT для продажи БАДов

Количество листов 35
Дата готовности 27.12.23
Руководитель Гукасян А. Д.
Разработал Галимьянова Э. Р.
Защищен 27.12.23 с оценкой _____
(дата)

Оренбург 2023

Содержание

Введение

Интернет сыграл ключевую роль в трансформации бизнеса, сделав его более доступным и удобным для потребителей. Он предоставляет платформу для продвижения товаров и услуг, а также для улучшения взаимодействия между предпринимателями и клиентами.

Веб-приложение – это программное обеспечение, которое функционирует на удаленном сервере и доступно пользователю через Интернет посредством веб-браузера. Оно позволяет выполнять различные задачи, включая работу с данными, обработку информации, взаимодействие с другими пользователями и многое другое. Веб-приложения могут быть разнообразными, от простых сайтов социальных сетей до сложных систем электронной коммерции, онлайн банкинга, интернет-форумов и др. Они позволяют пользователям получать информацию, взаимодействовать с контентом, выполнять операции и проводить транзакции в режиме реального времени. Это динамичные системы, которые обычно имеют клиент-серверную архитектуру, где клиентский браузер обращается к серверу для получения данных и выполнения действий.

Основные преимущества веб-приложений включают их доступность с любого устройства с доступом к интернету, обновление программного обеспечения на удаленном сервере без необходимости обновления клиентского компонента, а также возможность централизованного управления и хранения данных.

Потребность в создании информационных систем (ИС) может обусловливаться как необходимостью автоматизации или модернизации существующих информационных процессов, так и необходимостью коренной реорганизации в деятельности предприятия. Практически любая сфера деятельности (торговля, производство, медицина, оказание услуг и тд), так или иначе создает вокруг себя информационную систему.

В современном мире создание информационных систем становится все более значимым. Они не только автоматизируют процессы, но и являются ключевым инструментом для управления и анализа данных, что особенно важно для различных сфер деятельности, от медицины до услуг и производства. Веб-приложения дополнительно повышают эффективность работы и обслуживания. Благодаря им, процессы обработки данных и взаимодействия с клиентами становятся более удобными, быстрыми и прозрачными.

Индивидуальный веб-ресурс позволяет донести огромное количество информации до конечного потребителя в максимально короткие сроки. Наиболее актуальные новости компании, информация о предоставляемых услугах, изменениях в прайсе, режиме работы и отзывы клиентов – это лишь малый перечень информации, которая располагается на личном сайте фирмы. Главной особенностью такого способа информирования является то, что вся информация представляется в сжатом и одновременно полноценном виде, чего не может позволить ни один другой ресурс. Веб-приложения обеспечивают возможность персонализации взаимодействия с пользователями и сбора аналитических данных.

Это позволяет предприятиям адаптировать свои услуги и продукты под индивидуальные потребности клиентов, что способствует улучшению качества обслуживания и удовлетворенности клиентов.

Интернет-магазины стали неотъемлемой частью онлайн-покупок, предоставляя широкий выбор товаров и услуг. Веб-приложения, используемые в этих магазинах, обеспечивают покупателей удобством, эффективностью и безопасностью в процессе онлайн-шопинга.

Веб-приложения обычно разрабатываются с использованием веб-технологий, таких как HTML, CSS и JavaScript для клиентской части, и языков программирования, таких как PHP, Python, Java или Ruby, для серверной части. Они также могут использовать базы данных для хранения данных и взаимодействия с другими сервисами.

Создавая Веб-магазин, разработанный с использованием React и Node.js, остаётся актуальным и востребованным в современной веб-разработке. Технологии React и Node.js популярны для создания интернет-магазинов из-за поддержки отзывчивости и быстродействия, компонентной архитектуры, возможности создания одностраничного приложения (SPA). Оба фреймворка — React и Node.js - известны своей масштабируемостью. Это важно для веб-магазинов, которые могут столкнуться с ростом трафика и объемом данных. Node.js обеспечивает возможность масштабирования серверной части, а React - клиентской.

Актуальность Интернет-магазинов в современном мире набирает все больше и больше оборотов. Это обеспечивается за счет удобства и быстроты покупок в Интернете. Бешенный темп жизни современных людей забирает много времени, которое люди тратят на хождения по магазинам, поиск нужных товаров и сравнению цен, а интернет-магазины помогают людям экономить время и силы, и направлять столь ценные ресурсы на более приоритетные вещи. Интернет-магазины становятся все более актуальными с каждым годом. Это связано с ростом популярности онлайн-шопинга, удобством и быстротой оформления заказов, а также доступностью товаров со всего мира.

Основные преимущества интернет-магазина:

− возможность покупки товаров 24/7 в любое удобное время, без необходимости тратить время и деньги на посещение магазина в реальной жизни;

− широкий выбор товаров: можно легко сравнить цены, бренды, характеристики, отзывы покупателей и выбрать наиболее подходящий вариант;

− доставка товаров прямо на дом или в офис, что экономит время и силы;

− акции, скидки и предложения только для онлайн-покупателей, которые помогают экономить деньги;

− легкость и удобство поиска нужного товара и оформления заказа, а также возможность прямого общения с продавцом через онлайн-чат или электронную почту.

В целом, интернет-магазины обеспечивают намного более удобную и эффективную покупку товаров, чем традиционные магазины. Поэтому, эта форма продажи становится все популярнее каждый день.

Несомненно, интернет-магазины помогают вывести малый бизнес на мировой рынок за счет того, что рассеиваются географические препятствия процессов купли-продаж, так как человеку необязательно лично присутствовать в магазине, а продавец может поставлять свои товары в различные точки города, страны и даже мира. Это обуславливается наличием развитой системы доставки, которая может быть реализована во все крупно населенные пункты. Безусловно, удобство интернет-магазина заключается и в том, что в сети интернет она работает круглые сутки и позволяет покупателю искать и покупать товар в удобное для него время. За счет всего вышеперечисленного, можно сделать вывод и о том, что Интернет-магазин помогает вывести прибыль бизнеса на новый уровень. Это происходит за счет увеличения потенциальных клиентов и увеличения количества продаж.

Интернет-магазин продажи БАДов предоставляет организации множество преимуществ, помогая привлекать клиентов, показывать им полную информацию о продуктах и предлагать удобные условия покупки. Это способствует увеличению продаж и развитию бизнеса.

Цели, поставленные для выполнения во время производственной практики: реализация сайта Интернет-магазина, предназначенного для продажи остатков ассортимента биологически активных добавок в аптечных пунктах предприятия ГАУЗ «ОАС». Дизайн сайта должен быть привлекательным, понятным и пользовательски-ориентированным. Он должен отражать бренд или концепцию сайта. Хороший дизайн включает в себя грамотное использование цветовой палитры, шрифтов, изображений и композиции элементов. Важно также обеспечить удобную навигацию и интуитивно понятный пользовательский интерфейс.

Задачи, поставленные на выполнение в ходе реализации курсового проекта:

− определить предметную область, выяснить первоначальные потребности и бизнес-задачи системы;

− разработать схемы интерфейса;

− разработать макет сайта, выбрать наиболее подходящее для целевого рынка дизайнерское решение;

− выполнить проектирование дизайна сайта с применением промежуточных эскизов, требований к эргономике в технической эстетике;

− написать систему для продажи БАДов, используя стек технологий React, Node.js, PostgreSql;

− провести тестирования юзабилити сайта.

Объектом реализуемой системы является - Интернет-магазин по продаже биологически активных добавок.

Предмет реализуемой системы является - является реализация интернет-магазина средствами React, Node.js, PostgreSql, предназначенного для продажи остатков ассортимента биологически активных добавок в аптечных пунктах предприятия ГАУЗ «ОАС».

1 Анализ предметной области

Государственное автономное учреждение здравоохранения «Областной аптечный склад», расположенное по адресу г. Оренбург, ул. Березка, 24, реализует в Оренбургской области обеспечение медикаментами и товарами медицинского назначения лечебно-профилактических учреждений и населения области, закупаемым за счет средств областного и федерального бюджетов.

Объектом реализуемой системы является – Интернет-магазин по продаже биологически активных добавок, реализуемых предприятием ГАУЗ «ОАС».

Предмет реализуемой системы является - является реализация сайта Интернет-магазина, предназначенного для продажи остатков ассортимента биологически активных добавок в аптечных пунктах предприятия ГАУЗ «ОАС». Интернет-магазины для продажи БАДов — это специализированная онлайн площадка, где можно приобрести разнообразные виды биологически активных веществ и их композиций, предназначенных для непосредственного приёма с пищей или введения в состав пищевых продуктов.

Интернет-магазин предназначен для продажи остатков ассортимента БАДов в аптечных пунктах ГАУЗ "ОАС". Это специализированная онлайн-платформа, обеспечивающая удобство и доступность приобретения разнообразных видов биологически активных веществ для поддержания здоровья. Для оптимизации процессов и улучшения доступности медицинских продуктов, ГАУЗ "ОАС" реализует интернет-магазин по продаже биологически активных добавок (БАДов). Этот шаг направлен на современную цифровизацию в сфере здравоохранения и улучшение сервиса для населения. БАДы представляют собой продукты, содержащие биологически активные вещества, которые призваны поддерживать нормальное функционирование организма. Интернет-магазин предоставляет широкий выбор таких добавок, обеспечивая клиентам возможность выбора подходящего продукта.

Интернет-магазин становится эффективным механизмом распределения медицинских продуктов, предоставляя возможность клиентам легко и быстро получать необходимые товары, поддерживая своё здоровье. Грамотно созданный интернет-магазин является удобным инструментом маркетинга по активному продвижению товаров на рынок, что в конечно счете ведет к выполнению основной задачи интернет-магазина – увеличению прибыли от продажи товаров и большее количество новых посетителей и покупателей. При создании ИС появляется неограниченная база потенциальных клиентов, которые могут ознакомиться с ассортиментом, независимо от своего местонахождения, также появляется неограниченная виртуальная площадь витрин, позволяющая продемонстрировать весь товар, описав достоинства каждого.

Цели создания дизайна интернет-магазина для продажи БАДов включают:

− привлечение внимания и удержание посетителей: дизайн должен быть привлекательным и эстетически приятным, чтобы привлечь внимание посетителей и заинтересовать их, он также должен быть удобным в использовании и легким для

навигации, чтобы посетители могли легко находить нужные им продукты и информацию.

− отображение профессионализма и надежности: дизайн должен создавать впечатление организации, как эксперта в области БАДов. Это может быть достигнуто с помощью использования качественных изображений продуктов, понятной и информативной описательной информации, а также инструментов для связи с клиентами, таких как чат поддержки или контактные формы;

− подчеркивание ключевых характеристик и преимуществ продуктов: Дизайн должен помочь подчеркнуть уникальные характеристики и преимущества БАДов, чтобы посетители узнали, почему выбор данной продукции является лучшим. Это может быть достигнуто с помощью использования ярких и привлекательных изображений, визуальных элементов и эффективного размещения информации;

− создание доверия и безопасности: дизайн должен помочь посетителям чувствовать себя уверенно и безопасно при совершении покупок. Это может быть достигнуто с помощью использования ясных инструкций по оформлению заказа, значков безопасности и сертификатов, а также отзывов и рекомендаций других клиентов.

Система разрабатывается для магазина, продающего биологически активные добавки. Для того, чтобы интернет-магазин по продаже БАДов был успешным, нужно учитывать следующие требования:

− интернет-магазин должен обеспечивать высокий уровень защиты данных о покупателях: персональных данных и другой конфиденциальной информации;

− сайт магазина должен быть удобным и легким для использования. Пользователи должны быстро и легко находить нужные разделы и товары, а оформление заказа должно быть простым и интуитивно понятным;

− интернет-магазин должен быть доступен всегда, в том числе и в праздники и выходные дни, чтобы покупатели могли сделать заказы в удобное для них время;

− магазин должен иметь широкий продукции;

− интернет-магазин должен иметь различные каналы связи, такие как телефон, онлайн-чат или электронная почта, чтобы покупатели могли задавать вопросы и получать ответы;

− магазин должен иметь удобную систему доставки и оплаты, которая бы соответствовала потребностям покупателей и позволяла им выбрать удобный способ оплаты и доставки.

Учитывая эти требования, интернет-магазин для продажи БАДов сможет привлекать больше покупателей и быть успешным в своей деятельности. Внедрение веб-сайта для предприятия позволит изменить работу аптечных пунктов, существенно повысить количество клиентов, приобретающих биологически активные добавки и расширить показ основной информации товаров. Таким образом, разработка веб-сайта с экономической точки зрения эффективна.

2 Проектирование приложения

Разработанный сайт предприятия должен быть детально структурирован. Детальная структуризация сайта предполагает профессиональную разработку информационных разделов и блоков. Также, определение главной задачи созданного сайта – информация о фирме.

Для создания веб-сайта следует так же заострить внимание на:

- создание внутренней логики продукта;
- максимально интуитивном расположении элементов;
- создание дизайна;
- написание кода веб-сайта;
- проверка функционирования системы.

Каждый этап разработки сайта напрямую влияет на конечный результат проекта, его функциональность и эргономику. Проектирование веб-сайта по продаже биологически активных добавок (БАДов) на React включает в себя множество важных ключевых этапов.

Анализ требований. Определение Целевой Аудитории (ЦА) является ключевым этапом в разработке и маркетинге любого продукта или услуги, включая веб-сайты. Важно идентифицировать правильно целевую аудиторию, определить ее потребности и предпочтения.

Исследование конкуренции. Анализ конкурентов при создании сайтов играет ключевую роль в успешной разработке и маркетинговой стратегии. Важно изучить существующие веб-сайты, продающие БАДы, чтобы понять их стратегии, дизайн и функциональность.

Определение функциональности веб-приложения. Важно разработать структуру каталога, удобную для поиска и навигации. предоставить подробные описания каждого БАДа, включая состав, инструкции по применению и возможные противопоказания, создать удобную корзину покупок и процесс оформления заказа.

Дизайн и UX. Важно разработать логичную структуру меню и навигацию для удобства пользователей, создать эстетически приятный и профессиональный дизайн, который соответствует ценностям и стилю вашего бренда.

Оптимизация для поисковых систем (SEO). Нужно использовать соответствующие метатеги, ключевые слова и другие SEO-стратегии для улучшения видимости в поисковых системах.

Безопасность и соблюдение законодательства. Необходимо обеспечить безопасность личной информации клиентов и данных транзакций, убедиться, что ваш веб-сайт соответствует всем законам и нормативам в области электронной коммерции.

Создание веб-сайта по продаже БАДов на React требует внимательного планирования, дизайна и тестирования для обеспечения успешного запуска и удовлетворения потребностей вашей целевой аудитории.

Внутренняя логика продукта крайне необходима. С помощью этого пользователь не будет долго разбираться в продукте и не уйдет сразу же с веб-

сервиса, а поможет в этой задаче оформление (дизайн) элементов сайта - грамотное подчеркивание и оформление не даст клиенту заскучать, в то время как плохой дизайн может подтолкнуть клиента уйти с сайта даже, не успевши с ним ознакомиться.

Далее все решения должны быть перенесены в код.

После для хранения информации нужно создать БД и привязать ее к нашему продукту. Данная система отличается наличием большой БД для предприятия.

После полного анализа полученных целей и задач для интернет-каталога можно приступать к верстке сайта и к его последующему тестированию.

Данный сайт будет реализован посредством использования современных веб-технологий и языков. Под термином «технологический стек» понимают сложную комбинацию, включающую языки программирования, программное обеспечение и спектр фреймворков, применяемых для разработки IT-проекта. Архитектура любого web-приложения включает две стороны – клиентскую и серверную. Клиентской частью считаются визуализирующиеся данные, доступные пользователям на дисплее, которые используются посетителями сайта.

Для реализации данного веб-сайта будет использован современный технологический стек, объединяющий в себе разнообразные языки программирования, программное обеспечение и фреймворки. Термин "технологический стек" охватывает широкий спектр компонентов, необходимых для создания IT-проекта. Архитектура любого веб-приложения включает две основные стороны: клиентскую и серверную.

Клиентская часть представляет собой визуализирующиеся данные, доступные пользователям на экране. Эти данные используются посетителями сайта во время их взаимодействия с ресурсом. Основными компонентами структуры клиентской части IT-проекта являются:

− язык программирования, ответственный за интерактивную часть веб-проекта, который в данном случае является JavaScript. JavaScript играет ключевую роль в обеспечении интерактивности веб-проекта. Этот язык программирования позволяет создавать динамичные элементы, обеспечивая пользовательский опыт более интересными и функциональными возможностями;

− язык разметки документации, обеспечивающий точное отображение содержимого сайтов в браузере, а именно HTML. HTML, язык разметки документации, является фундаментальным компонентом для точного представления содержимого сайта в браузере. Он определяет структуру информации и обеспечивает понятное восприятие контента;

− формальный (табличный) язык, который обеспечивает правильное стилизование контента – CSS. CSS, табличный язык, отвечает за эстетику и стиль визуального представления веб-страницы. Он позволяет создавать привлекательный дизайн и обеспечивать удобство восприятия контента.

Разработка серверной стороны включает в себя использование следующего технологического стека:

базы данных, в данном случае, PostgreSql, обеспечивающей хранение и эффективное управление данными проекта.

− PostgreSql является выбранным решением для базы данных, обеспечивая надежное и эффективное хранение данных проекта. Его мощные возможности позволяют эффективно управлять информацией и обеспечивать надежность;

− веб-фреймворк Express.js для языка программирования JavaScript, который обеспечивает легкость и эффективность в создании серверной части веб-приложения. Express.js, веб-фреймворк для языка программирования JavaScript, обеспечивает эффективное создание серверной части веб-приложения. Его легковесность и гибкость делают его предпочтительным выбором для обработки запросов и управления данными.

Все вышеперечисленные технологические компоненты интегрируются для создания современного и функционального веб-приложения. Их взаимодействие обеспечивает оптимальную работу как клиентской, так и серверной стороны проекта.

Реализация проекта с использованием современных веб-технологий обеспечивает не только функциональность, но и способствует легкости разработки, обеспечивая высокую производительность и удовлетворение пользовательских потребностей.

В функционале сайта должны присутствовать такие возможности, как:

− просмотр основного каталога товара;

− возможность сортировки товара по каким-либо критериям;

− возможность поиска товара по торговому наименованию;

− добавление товаров в корзину;

− предоставление основной информацию о компании;

− возможность бронировать товар или оформлять доставку товара;

− возможность просмотра информации о заказах и их статусе;

− возможность создания личного кабинета.

IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Данная методология, разработанная для описания системы в целом, представлена в приложении Б на рисунке Б1. После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией. Данная диаграмма для разрабатываемой системы представлена в приложении Б на рисунке Б1.

Для создания информационной системы необходимо решить следующие задачи:

− функциональность: анализ клиентской базы, планирование контактов с клиентами;

− удобство использования: интуитивность, мультимедийность, удобство обучения, легкость получения справочной информации;

− производительность: высокая пропускная способность, минимальное время отклика системы, высокая скорость восстановления, масштабируемость.

3 Разработка программного обеспечения

3.1 Описание технологического стека разработки

При разработке данного проекта будет применяться современный и мощный стек технологий, включающий в себя React, Node.js и PostgreSql. Этот технологический арсенал обеспечивает не только высокую производительность, но и эффективное взаимодействие всех компонентов приложения. Интеграция всех выбранных технологий позволит создать качественное и масштабируемое веб-приложение. Взаимодействие между React, Node.js и PostgreSql обеспечит оптимальную работу приложения на каждом этапе его функционирования.

Применение этого стека обеспечит не только высокую эффективность, но и возможность масштабировать проект в будущем. Каждый компонент технологического стека вносит свой вклад в создание надежного, функционального и инновационного веб-приложения. Использование такого разнообразного и сбалансированного стека технологий также упрощает процесс разработки, делая его более удобным и эффективным. Каждый инструмент в этом арсенале спроектирован с учетом лучших практик.

React является основой клиентской части веб-приложения. Эта библиотека обеспечивает создание динамичных и эффективных пользовательских интерфейсов, что делает взаимодействие пользователя с приложением более комфортным и привлекательным. React представляет собой открытую JavaScript библиотеку, которая призвана упростить процесс создания масштабных веб-приложений. Ориентированный на компонентный подход, React решает проблемы производительности приложения, сделав его более отзывчивым и эффективным за счет использования виртуального DOM.

Веб-страница представляет собой структуру (DOM), содержащую элементы, такие как текст, изображения и кнопки, с которыми пользователь взаимодействует. DOM организовано в виде иерархического дерева элементов HTML, используемого браузером для отображения содержимого страницы. JavaScript может изменять DOM, что позволяет создавать динамичные веб-приложения.

React JS обеспечивает ряд выгод, таких как ускорение процесса разработки, уменьшение числа ошибок и облегчение поддержки больших кодовых баз. Для бизнеса React JS приносит пользу, ускоряя создание новых продуктов и улучшение уже существующих. Это приводит к повышению эффективности разработки и снижению затрат на производство продукта в долгосрочной перспективе. React в качестве фреймворка для web-приложений обеспечивает масштабируемость и скорость при создании и отображении пользовательских интерфейсов.

Основные преимущества React JS:

− компонентный подход: React.js основан на концепции создания множества маленьких, переиспользуемых компонентов. Эти компоненты затем объединяются в более крупные приложения;

− виртуальный DOM - это одна из главных особенностей React.js. Это означает, что React не обновляет все элементы на странице, а только те, которые действительно изменились, что делает приложение меньше и быстрее;

− активная поддержка со стороны сообщества - React JS имеет большое сообщество разработчиков, которые постоянно обновляют и улучшают эту библиотеку, создают новые расширения и плагины.

Основы Node.js. Node.js станет основой серверной части приложения. Этот серверный фреймворк на языке JavaScript обеспечивает высокую производительность и масштабируемость, обрабатывая запросы от клиентской части и взаимодействуя с базой данных. Node.js представляет собой окружение, которое дает возможность создавать программы на языке JavaScript вне браузера. Ранее JavaScript-программы можно было запускать только в браузерах, обладающих специализированным встроенным движком для данного языка. Однако появление Node.js, основанного на движке JavaScript под названием V8, созданного Google для браузера Chrome, кардинально изменило эту динамику. Благодаря Node.js теперь JavaScript-код может выполняться практически в любой среде, что позволяет разрабатывать как клиентскую, так и серверную части веб-приложений, используя единый язык программирования.

Node.js обладает несколькими ключевыми характеристиками, которые делают его мощным и уникальным инструментом для разработки приложений. Некоторые из этих характеристик включают:

Node.js позволяет разрабатывать приложения, используя популярный язык программирования - JavaScript. Эта совместимость упрощает процесс разработки, позволяя разработчикам применять уже имеющиеся навыки и опыт в новых проектах.

Асинхронная и событийно-ориентированная природа. Асинхронность - ключевая характеристика Node.js. Это означает, что серверы, созданные с использованием Node.js, не ожидают завершения операций ввода-вывода перед тем, как продолжить выполнение других задач. Асинхронность также подразумевает наличие неблокирующего ввода-вывода, что позволяет обрабатывать другие операции во время ожидания ответов, например, от базы данных.

Однопоточность. Node.js по умолчанию работает в однопоточном режиме, что означает выполнение всех операций в одном основном потоке. Однако это не мешает Node.js эффективно обрабатывать множество одновременных соединений.

Высокая производительность. Благодаря асинхронной обработке и архитектуре V8, Node.js способен обрабатывать большое количество запросов с минимальной задержкой, что делает его отличным выбором для создания высоконагруженных приложений.

Основы PostgreSql. Физически база данных — это файлы в определённом формате. Если базу нужно сохранить, например как резервную копию, используют дамп (от англ. dump — сбрасывать). Дамп — это файл с базой, в котором сохранены все данные, то есть просто хранилище, поэтому с таким файлом невозможно работать. Чтобы администрировать базу — загружать, выгружать или изменять данные, нужна система управления базами данных (сокр. СУБД).

PostgreSQL — это бесплатная СУБД с открытым исходным кодом. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL.

PostgreSQL — объектно-реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход. Традиционно популярные СУБД — реляционные. Это значит, что данные, которые в них хранятся, представляются в виде записей, связанных друг с другом отношениями, — relations. Получаются связанные списки, которые могут иметь между собой те или иные отношения, — так и образуется таблица. Существует еще одна популярная модель — объектная. Данные представляются в виде объектов, их атрибутов, методов и классов.

Поддержка множества типов данных. Еще одна особенность PostgreSQL — поддержка большого количества типов записи информации. В ней есть поддержка XML, JSON и тд.

PostgreSQL поддерживает работу с большими объемами. В большинстве СУБД, рассчитанных на средние и небольшие проекты, есть ограничения по объему базы и количеству записей в ней. В PostgreSQL ограничений нет.

PostgreSQL работает со сложными, составными запросами. Система справляется с задачами разбора и выполнения трудоемких операций, которые подразумевают и чтение, и запись, и валидацию одновременно.

PostgreSQL поддерживает больше языков, чем аналоги. Кроме стандартного SQL, в PostgreSQL можно писать на C и C++, Java, Python, PHP, Lua и Ruby. Он поддерживает V8 — один из движков JavaScript, поэтому JS тоже можно использовать совместно с PgSQL.

3.2 Описание алгоритма работы

Первоначальный этап разработки веб-приложения включает в себя планирование и анализ системы. Планирование и анализ являются ключевыми этапами в разработке веб-сайта и играют важную роль в обеспечении успешной реализации проекта. Планирование позволяет четко определить цели веб-сайта и задачи, которые он должен выполнять. Это формирует основу для всего процесса разработки. Анализ помогает лучше понять вашу целевую аудиторию, их потребности, предпочтения и ожидания. Это позволяет адаптировать дизайн и функциональность сайта под требования пользователей. Первоначально необходимо определить цели и целевую аудиторию веб-сайта, для дальнейшей удачной разработки веб-приложения. Далее следует провести анализ конкурентов и изучить требования рынка БАДов, спланировать основные функциональные и дизайнерские требования.

Следующим этапом в разработке сайта являются проектирование и разработка дизайна. Необходимо разработать структуру информации и создать макеты страниц, разработать дизайн, учитывая корпоративный стиль и UX/UI принципы, выбрать цветовую палитру, шрифты и другие дизайнерские элементы.

Далее необходимо выбрать технологического стек. Определите, какие библиотеки и фреймворки React будут использоваться. React.js — это всего лишь способ в удобном виде представить код JavaScript и HTML, сделать его повторяемым и наглядным. Компоненты React.js пишут на особом языке — JSX, который выглядит как смесь JavaScript и HTML. В нашем случае необходимо прописать в консоли команду npm i, что позволяет использовать обширную экосистему сторонних пакетов и легко устанавливать или обновлять их. Сборщик, такой как webpack или Parcel. Он позволяет писать модульный код и объединять его в небольшие пакеты, чтобы оптимизировать время загрузки. Далее установить пакет react-router-dom, который содержит привязки для использования React Router в веб-приложениях.

Разработка фронтенда (React). Разработка фронтенда в среде React - процесс, который представляет собой создание интерактивных и динамичных пользовательских интерфейсов для веб-приложений. Необходимо создать компоненты и страницы React, отвечающие за отображение информации о продуктах и услугах. Первоначально установливается Node.js и npm (пакетный менеджер Node.js) для эффективной установки и управления зависимостями проекта, внедряется маршрутизация для создания одностраничного приложения (SPA) - React Router. При разработке веб-приложения используются обьекты управление состоянием, например, через встроенный стейт. Объект state описывает внутреннее состояние компонента. Это обеспечивает эффективное управление данными в приложении. Одним из наиболее используемых встроенных хуков является useState, который позволяет определить состояние компонента. Необходимо произвести работу с внешними данными, взаимодействуя с API через HTTP-запросы, например с помощью Fetch. Fetch() – это метод JavaScript для выполнения HTTP-запросов, а также метод setState() React для обновления состояния вашего приложения при получении ответа.

Следующим этапом является анализ организации базы данных в PostgreSQL. И заполнение таблиц данными. Схема базы данных представлена на рисунке А1.

Разработка бэкенда (Node.js). Создаётся серверное приложение на Node.js с использованием Express.js. Нужно установить и настроить среду разработки для Node.js. С помощью команды npm init создаётся файл package.json для своего приложения. Файл package. json позволяет указать диапазон версий пакетов, которые нужно ставить. Это нужно, чтобы не следить вручную за появлением новых версий пакетов. Также при разработки серверной части веб-приложения понадобятся такие библиотеки, как pg, nodemon, express. Далее создаётся основной файл вашего сервера, например, index.js, определяются маршруты для обработки HTTP-запросов. Express обеспечивает простой способ определения обработчиков для различных типов запросов. Далее необходимо создать API для обработки запросов от клиентской части приложения, определить роуты, которые обрабатывают запросы и возвращают данные.

Интеграция с базой данных (PostgreSQL). Необходимо написать SQL-запросы и скрипты для создания и обслуживания базы данных, интегрировать серверное приложение с базой данных, используя соединение с PostgreSQL.

Интеграция с базой данных PostgreSQL в Node.js является важным этапом разработки бэкенда. PostgreSQL — это мощная реляционная система управления базами данных, и вот как можно осуществить ее интеграцию в Node.js с использованием библиотеки pg.

Далее нужно оптимизировать код для максимальной производительности и быстрой загрузки страниц.

Код реализации серверной и клиентской части веб-приложения представлен в приложении Д.

И заключающим этапом является тестирование. Проводится тестирование функциональности, включая модульное, интеграционное и системное тестирование, проверяется безопасность приложения и защита от потенциальных угроз, осуществляется тестирование совместимости с различными браузерами и устройствами.

3.3 Описание интерфейса пользователя

Логика работы интернет-магазина предполагает сбор и отображение информации о товарах или услугах, которые магазин предлагает покупателям, а также предоставление удобных инструментов для заказа и оплаты покупок.

Диаграмма прецедентов использования (use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. Данная диаграмма изображена в приложении В на рисунке В1.

Как было сказано ранее, дизайн сайта должен быть интуитивно понятным, комфортным в использовании всем пользователям, для этого должна быть создана карта пользователя сайта. Модель взаимодействия пользователей с платформой (карта пользователя) представлена на рисунке 1.

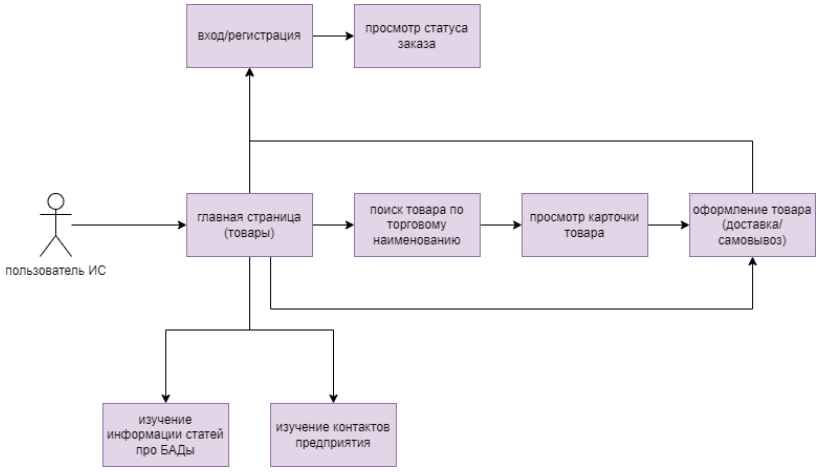

Рисунок 1 - Карта пользователя информационной системы

При разработке сайта важно выделить ключевые этапы взаимодействия пользователя с интерфейсом. Диаграмм схемы интерфейсов показывает переходы между различными страницами и компонентами интерфейса. Это важно для обеспечения плавной навигации пользователя и предотвращения потери

пользователя на сайте. На рисунке 2 представлена диаграмма схемы интерфейсов, которая позволяет проследить пересылку пользователя по сайту.

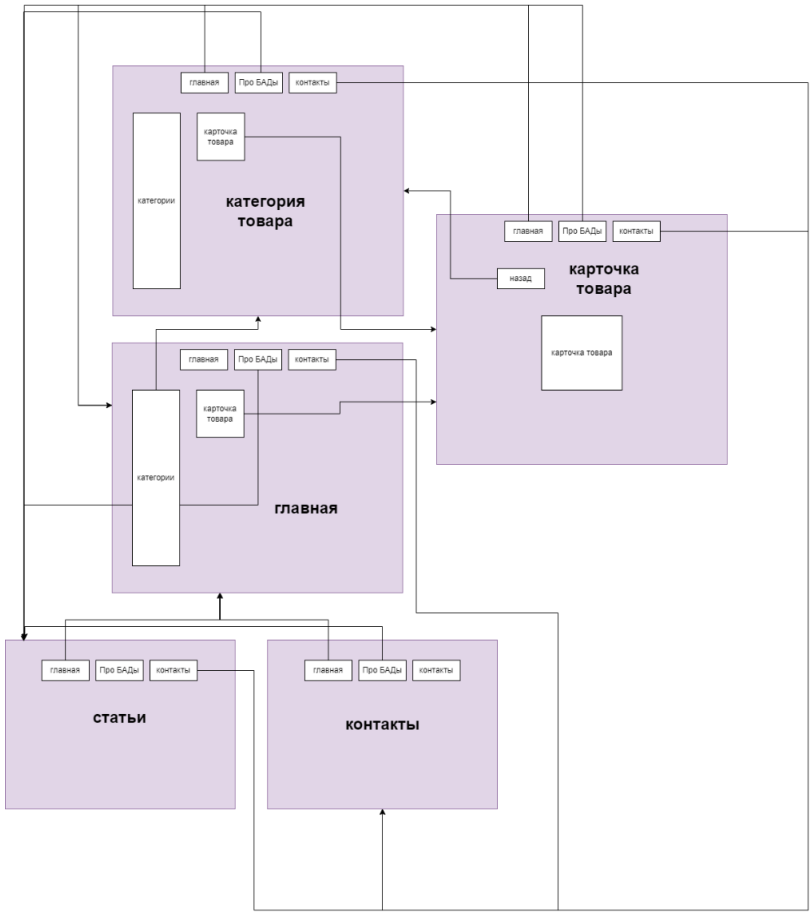

Рисунок 2 - Диаграмма схемы интерфейсов

Дизайн для сайта будет разрабатываться в программе Figma. Figma — это популярный инструмент для дизайна и прототипирования интерфейсов, который позволяет дизайнерам создавать, редактировать и совместно работать над макетами. Figma предлагает широкий набор функций, таких как векторный дизайн, библиотеки компонентов, совместная работа в реальном времени и многое другое.

Во время создания системы, необходимо опираться на представленные базовые элементы такие как: цветовая палитра, типографика, колоночная сетка, которые задают цветовую гамму, текстовое оформление и расположение объектов для дальнейшего дизайна системы.

Правильно подобранная цветовая схема позволит человеку запомнить дизайн сайта и выделить предприятие среди конкурентов. Для приятного, не перегруженного дизайна сайта нужно использовать правильно подобранные цветовые сочетания, которые смогут акцентировать внимание пользователя на нужных элементах сайта, не перегружая интерфейс веб-сайта. Сочетающиеся оттенки способствуют тому, что пользователь задержится на веб-ресурсе и совершит целевое действие. При выборе цветового решения важно опираться на тематику страницы, аудиторию и брендинг компании, для которой создается сайт.

При выборе цветовой палитры мы остановились на акцентом оттенке зелёного цвета, оттенке белого цвета и оттенке серо-коричневого цвета (представлено на рисунке 3). Выбор зеленого оттенка обуславливается тем, что согласно психологии цвета, зеленый является символом здоровья, свежести, природы, чистоты и роста. Именно такое настроение нужно передать пользователю системы, так как система направлена на продажу биологически активных добавок, что в свою очередь напрямую связано со здоровьем человека.

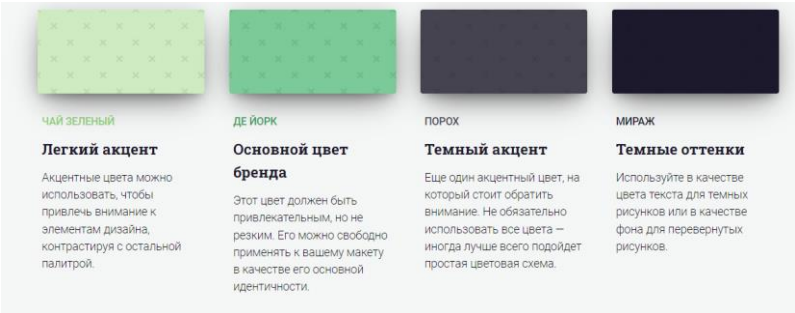

Рисунок 3 - Цветовая палитра сайта

Шрифт, используемый при создании сайта: Open Sans (его главные плюсы – разборчивость и универсальность). Размер шрифта должен обеспечивать удобное восприятие текста при любом разрешении экрана.

Система будет разрабатываться для десктопных носителей. Ширина контейнера будет составлять 1200px. В дальнейшем для системы возможно создание мобильной версии сайта. Для разработки дизайна сайта будет использоваться 12-ти колоночная сетка (рисунок 4), обеспечивающая рациональное заполнение пространства страницы объектами на сайте, а также для определения структуры, иерархии и ритм дизайна.

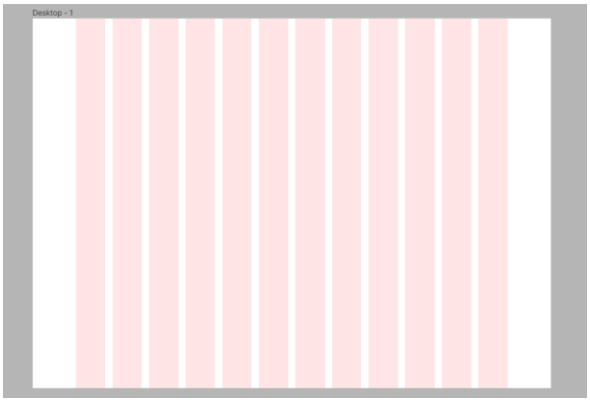

Рисунок 4 - 12-ти колоночная система для разработки дизайна ИС

Для дизайна системы выбраны плавные скругления элементов сайта, из-за того, что исследования показали, что прямоугольники со скругленными углами меньше раздражают глаза, чем прямоугольники с острыми краями, потому что для их визуальной обработки требуется меньше когнитивных усилий. Также с ранних лет в нас психологически закладывали, что острые предметы и опасны и не вызывают доверия, нам же наоборот дизайном системы нужно привить человеку

чувство безопасности и доверия к нашим продуктам. Скруглению поддаются такие объекты системы, как:

- кнопки сайта;
- карточки товаров сайта;
- изображения сайта;
- различные формы взаимодействия с пользователем.

Разрабатывая дизайн сайта важно заострить внимание на правильно подобранном шрифте, его начертании и размере в конкретном месте сайта, поскольку Текст является основным инструментом передачи информации на веб-страницах. Работая с типографикой, мы акцентируем внимание пользователей на нужных деталях. Типографика в веб-дизайне — это правила оформления текста сайта с целью донесения информации до аудитории в легчайшем для чтения и восприятия форме.

При создании дизайна интернет-магазина по продаже БАДов для предприятия ГАУЗ «ОАС» будут использоваться начертания, размеры и цвета для оформления текста на сайте, представленные на рисунке 5.

Рисунок 5 - Типографика информационной системы

Полностью разработанный макет системы для интернет-магазина, реализующего продажу остатков биологически активных добавок аптечных пунктов предприятия ГАУЗ «ОАС», с учетом всех правил UX/UI дизайн представлен в приложении Г.

4 Тестирование приложения.

4.1 План тестирования

Контроль и приемка интернет-магазинов является важным шагом в процессе их разработки и запуска. Несколько основных этапов контроля и приемки интернет-магазина представлены ниже.

Тестирование функциональности. Первым шагом является тестирование функциональности сайта. Необходимо проверить, правильно ли отображается каталог товаров, работает ли корзина и система оплаты, правильно ли обрабатываются заказы и т.д.

Тестирование соответствия требованиям. Второй шаг - проверка соответствия сайта требованиям заказчика. Это включает в себя проверку наличия всех необходимых функций и инструментов, точное соответствие макетам, дизайну и стилю сайта, соответствие цен и характеристик товаров и т.д.

Тестирование безопасности. Третий шаг – тестирование безопасности. Это включает проверку защиты от взлома, защиту персональных данных клиентов и подбор паролей.

Проверка качества контента. Четвертый шаг – проверка качества контента, который отображается на сайте. Необходимо убедиться, что описания товаров и услуг являются точными и полными, а картинки высокого качества и точно отображают товар.

Оценка производительности. Пятый шаг – оценка производительности сайта, включая скорость загрузки страниц и работу системы управления сайтом. Проверка систем защиты. Шестой шаг – проверка систем защиты сайта от взлома, спама и вредоносных программ.

Проверка SEO-оптимизации. Седьмой шаг – проверка SEO-оптимизации сайта, включая правильность использования ключевых слов, мета-описаний и других SEO-элементов.

План тестирования веб-сайта, разработанного на React, обеспечивает полный и систематический подход к проверке функциональности, производительности и безопасности приложения. Ниже представлен общий план тестирования, который может быть адаптирован под конкретные потребности проекта.

Тестирование функциональности:

− навигация и взаимодействие - проверка навигации по сайту (тестирование всех интерактивных элементов, включая кнопки, формы, меню и ссылки);

− верификация корректности отображения и взаимодействия компонентов React;

− проверка корректности ввода данных в формы;

− тестирование валидации форм и обработки ошибок;

− проверка отправки данных на сервер:

− аутентификация и авторизация;

− проверка правильности отображения данных пользователя после входа;

− проверка прав доступа в зависимости от уровня авторизации;
− проверка корректности отображения данных на страницах;
− тестирование фильтрации, сортировки и поиска данных;
− проверка динамического обновления данных без перезагрузки страницы.

Тестирование совместимости.

− проверка работоспособности в различных браузерах (Chrome, Firefox, Safari, Edge);
− тестирование на различных версиях браузеров.

Тестирование производительности.

− загрузка страниц (измерение времени загрузки главных страниц);
− проверка производительности (при использовании медленного интернет-соединения);
− мониторинг использования ресурсов браузера;
− проверка оптимизации изображений и других медиа-ресурсов.

Тестирование нагрузки.

− стресс-тестирование (проверка стабильности приложения при максимальной нагрузке, тестирование на предмет утечек памяти и проблем с производительностью).

Юзабилити-тестирование сайта — это процесс оценки удобства использования сайта для пользователей. Целью тестирования является выявление проблем, с которыми пользователи могут столкнуться при использовании сайта, и улучшение юзабилити сайта для улучшения пользовательского опыта.

Выбор методики тестирования юзабилити веб-сайта зависит от целей тестирования, доступных ресурсов и времени, а также от характеристик аудитории сайта. В процессе тестирования юзабилити были использованы следующие методы:

− тестирование сценариев использования: этот метод заключается в том, чтобы попросить пользователей выполнить определенные задачи на сайте и записать их действия и комментарии. Это позволяет оценить, насколько легко и интуитивно понятно пользователю использование сайта;
− тестирование с использованием эмоциональных метрик: этот метод заключается в том, чтобы попросить пользователей оценить свои эмоции и удовлетворенность при использовании сайта. Это позволяет оценить, насколько сайт вызывает положительные эмоции у пользователей.

4.2 Оценка результатов проведения тестирования

Проведение тестирования веб-приложения, разработанного на React, имеет критическое значение для обеспечения высокого уровня качества, надежности и безопасности приложения. Тестирование веб-приложения на React является неотъемлемой частью разработки, обеспечивая высокий стандарт качества, устойчивость и надежность приложения в процессе его жизненного цикла.

Тестирование Функциональности. Все функциональные требования были протестированы и успешно соответствуют ожиданиям. Навигация, взаимодействие

и формы работают корректно, обеспечивая удобство пользователей. Аутентификация и авторизация безопасны и эффективны.

Тестирование Совместимости. Веб-сайт прошел успешное тестирование в различных браузерах (Chrome, Firefox, Edge) и на различных устройствах (мобильные телефоны, планшеты).

Тестирование Производительности. Время загрузки главных страниц соответствует установленным стандартам. Ресурсоемкость оптимизирована, что обеспечивает плавную работу даже при медленном интернет-соединении.

Тестирование Нагрузки. Сайт успешно выдерживает стресс-тестирование при максимальной нагрузке. Проблем с производительностью и утечками памяти не обнаружено.

Взаимодействие с бэкендом и API проходит успешно. Данные передаются согласованно между frontend и backend.

Итогами проведенного тестирования юзабилити являются следующие показатели, представленные в таблице 1.

Таблица 1 – Результаты опроса пользователей сайта

Номер тестируемого	1	2	3	4	5	Итоговый балл
Вопрос о системе	Оценка по пятибалльной системе оценивания					
Насколько понятный интерфейс сайта	5	5	4	5	5	4,8
Насколько хорошо сайт ассоциируется с темой	5	5	5	5	5	5
Насколько сайт запоминающийся	5	5	5	5	4	4,8
Хотели бы вы что-то приобрести на этом сайте	5	5	5	5	5	5
Соответствует ли дизайн сайта современным трендам и ожиданиям пользователей?	5	5	5	5	5	5
Насколько хорошо организованы элементы дизайна на странице и насколько легко на них фокусироваться?	5	5	5	4	5	4,8

Веб-сайт, разработанный на React, успешно прошел все этапы тестирования, демонстрируя высокую функциональность, производительность и безопасность. Обнаруженные и устраненные в процессе тестирования ошибки не влияют на работоспособность приложения. Результаты тестирования гарантируют стабильное и эффективное функционирование веб-сайта для пользователей.

Заключение

В ходе курсовой работы акцент был сделан на важности анализа предметной области и потребностей предприятия. Определение бизнес-задач системы позволило сформировать четкие цели для создания Интернет-магазина. Активное изучение предприятия ГАУЗ "ОАС" позволило глубоко погрузиться в контекст и осознать, какие возможности и проблемы стоят перед бизнесом.

Следующим важным этапом была разработка схем интерфейса и макета сайта. Это дало возможность визуализировать концепцию будущего веб-приложения. Особое внимание уделено эргономике и технической эстетике, чтобы обеспечить максимальное удобство использования для конечного пользователя.

Применение современного стека технологий, включающего React, Node.js, и PostgreSql, подчеркнуло технологическую актуальность проекта. Эти инструменты предоставили не только высокую производительность, но и обеспечили удобство разработки и поддержки.

Тестирование веб-сайта является ключевым этапом в разработке, и его успешное проведение подтвердило функциональность, стабильность и безопасность веб-приложения. Обнаружение и устранение возможных проблем на этом этапе позволяет предотвратить негативный опыт для пользователей в будущем.

В процессе выполнения курсовой работы была успешно выполнена задача по реализации Интернет-магазина для продажи биологически активных добавок. Работа включала в себя ряд этапов, начиная с изучения информации о предприятии и его потребностях, заканчивая реализацией готового веб-приложения. В рамках курсового проекта были решены следующие задачи:

− определена предметную область, выяснены первоначальные потребности и бизнес-задачи системы;

− разработаны схемы интерфейса;

− разработан макет сайта, выбран наиболее подходящее для целевого рынка дизайнерское решение;

− выполнено проектирование дизайна сайта с применением промежуточных эскизов, требований к эргономике в технической эстетике;

− реализована систему для продажи БАДов, используя стек технологий React, применяя дополнительные инструменты Node.js, PostgreSql;

− проведено тестирование сайта.

Результатом данной работы стало веб-приложение для реализации биологически активных добавок. Сайт отличается привлекательным и удобным дизайном, обеспечивает быструю загрузку страниц и удобную навигацию. Важно отметить, что создание современного интернет-магазина – это не только предоставление продукции, но и инструмент для улучшения сервиса, привлечения новых клиентов и оптимизации бизнес-процессов. Работа нацелена на увеличение прибыли, привлечение новых посетителей и повышение эффективности продаж.

Список используемых источников

1 UML для бизнес-моделирования: зачем нужны диаграммы процессов. — Текст: электронный // evergreens.com: [сайт]. — Режим доступа: https://evergreens.com.ua/ru/articles/uml-diagrams.html (дата обращения: 26.11.2023).

2 What is PostgreSQL? — Текст: электронный // aws.amazon.com: [сайт]. — Режим доступа: https://aws.amazon.com/ru/rds/postgresql/what-is-postgresql/ (дата обращения: 05.12.2023)

3 Базово о React: что это такое и как помогает разработчику — Текст: электронный // practicum.yandex.ru: [сайт]. — Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-react-i-kak-on-rabotaet/ (дата обращения: 30.11.2023)

4 Десять советов по оптимизации скорости работы вашего сайта. — Текст: электронный // habr.com: [сайт]. — Режим доступа: https://habr.com/ru/articles/112720/ (дата обращения: 29.11.2023).

5 Диаграмма IDEF0 второго уровня. — Текст: электронный // megaobuchalka.ru: [сайт]. — Режим доступа: https://megaobuchalka.ru/8/26772.html (дата обращения: 10.11.2023).

6 Как дизайн влияет на продажи. — Текст: электронный // dzen.ru: [сайт]. — Режим доступа: https://dzen.ru/a/ZGDiXFEDcnnU9KaC (дата обращения: 7.09.2023)

7 Обзор фреймворка React.js: преимущества, недостатки и сценарии использования — Текст: электронный // serverspace.ru: [сайт]. — Режим доступа: https://serverspace.ru/about/blog/obzor-frejmvorka-react-js/ (дата обращения: 30.11.2023)

8 ОПТИМИЗАЦИЯ ИЗОБРАЖЕНИЙ ДЛЯ САЙТА. — Текст: электронный // ashmanov.com: [сайт]. — Режим доступа: https://www.ashmanov.com/education/articles/optimizatsiya-izobrazhenij-dlya-sajta/ (дата обращения: 10.12.2023)

9 ОСОБЕННОСТИ, ХАРАКТЕРИСТИКИ И ОБЛАСТИ ПРИМЕНЕНИЯ NODE.JS — Текст: электронный // scand.com: [сайт]. — Режим доступа: https://scand.com/ru/company/blog/node-js-features-uses-and-benefits-of-development/ (дата обращения: 01.12.2023)

10 Правильная реализация перехода по ссылке в рамках одного сайта. — Текст: электронный // evergreens.com: [сайт]. — Режим доступа: https://ru.stackoverflow.com/questions/740227/ (дата обращения: 17.12.2023).

11 Руководство по JavaScript. — Текст: электронный // metanit.com: [сайт]. — Режим доступа: https://metanit.com/web/javascript/ (дата обращения: 8.12.2023).

12 Хранение картинок в БД: за и против? — Текст: электронный // habr.com: [сайт]. — Режим доступа: https://qna.habr.com/q/21342 (дата обращения: 12.12.2023)

Приложение А
(обязательное)

Информационная модель системы

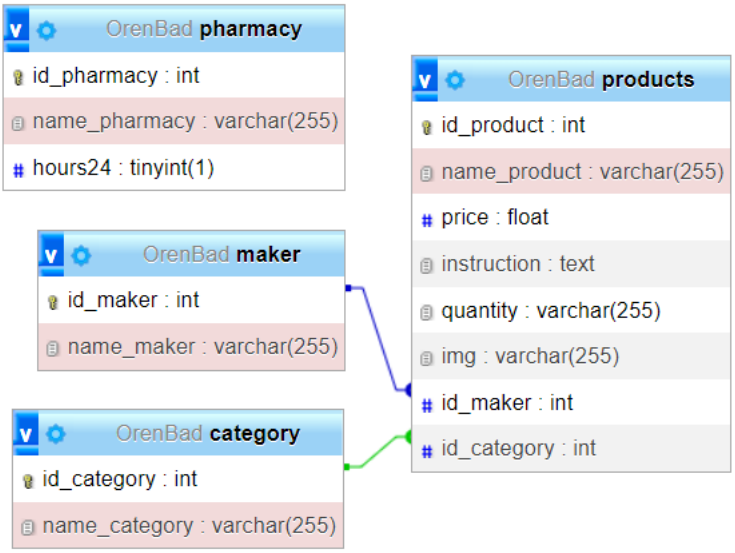

Рисунок А1 – Информационная модель системы

Приложение Б
(обязательное)

Функциональная модель системы

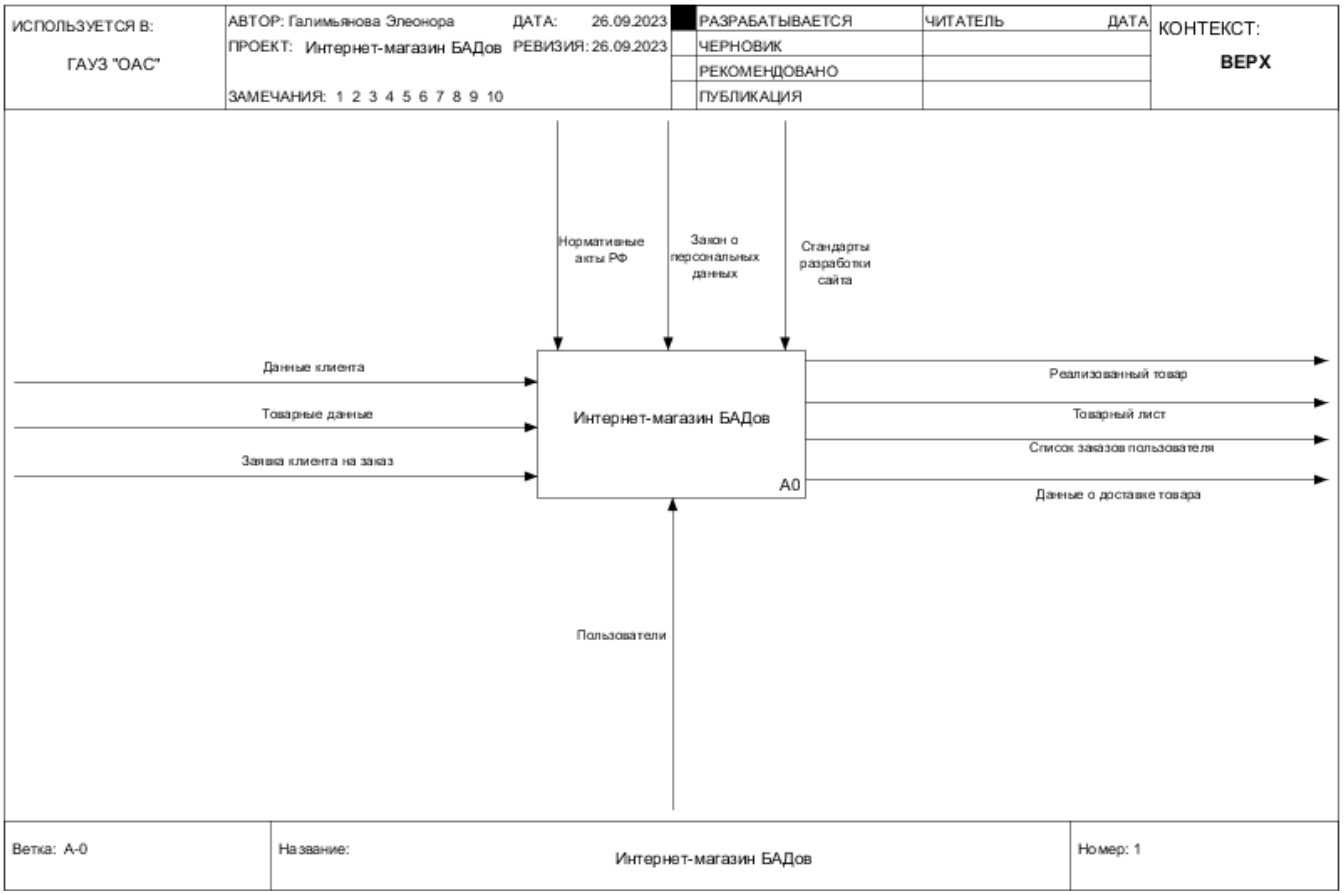

Рисунок Б1 - Функциональная модель системы первого уровня

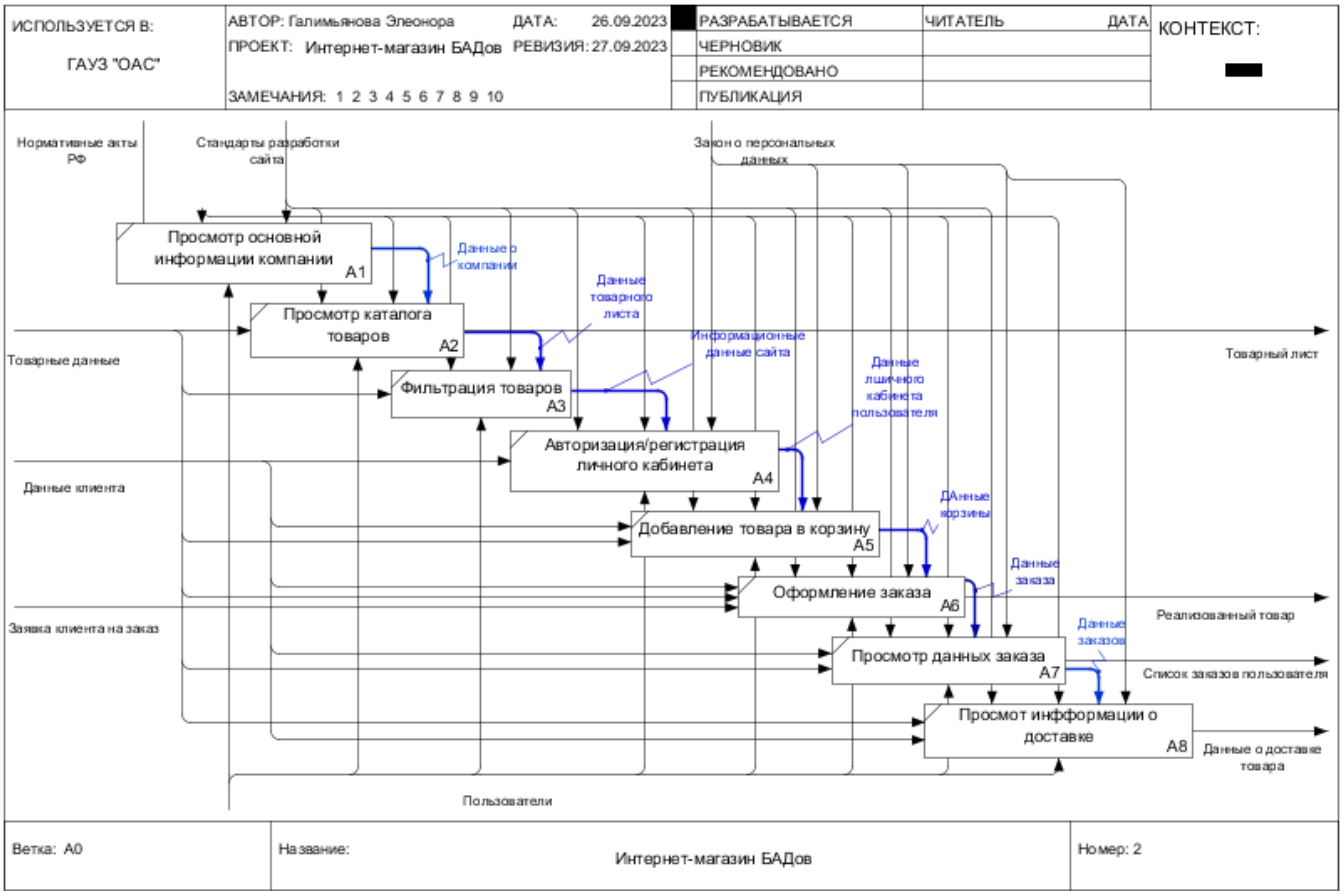

Рисунок Б2 - Функциональная модель системы второго уровня

Приложение В
(обязательное)

Диаграмма прецендентов системы

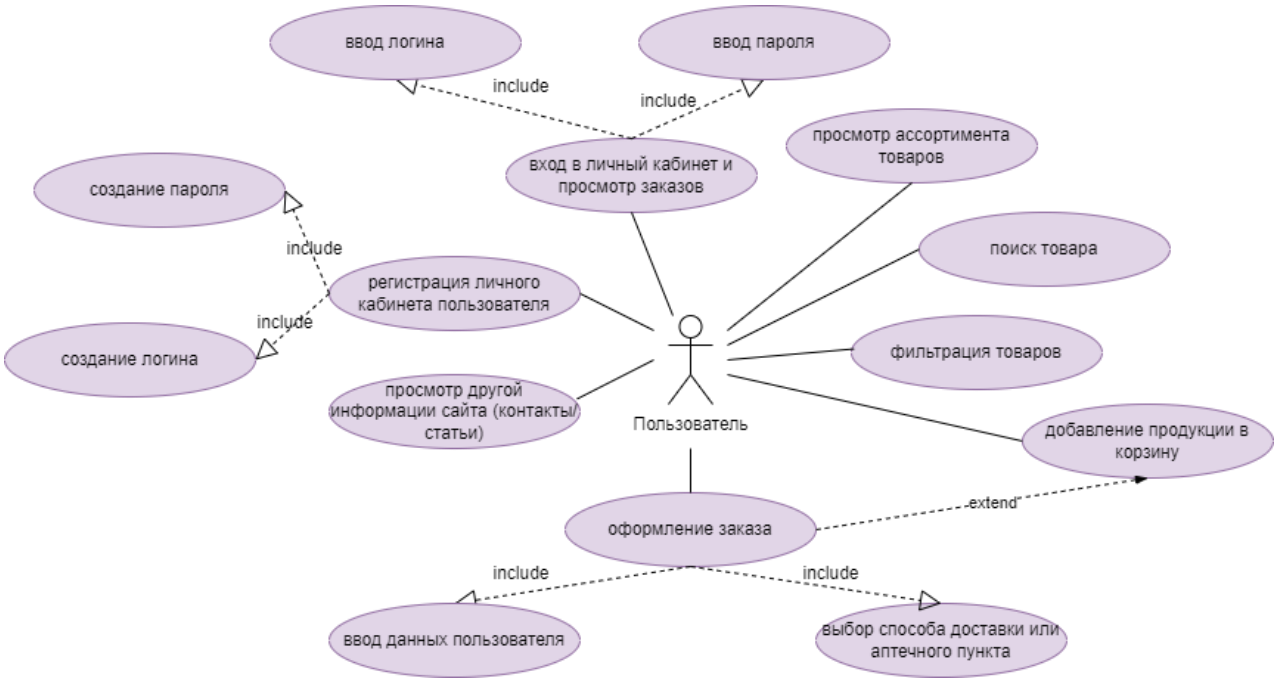

Рисунок В1 – Диаграмма прецедентов использования

Приложение Г
(обязательное)

Дизайн информационной системы

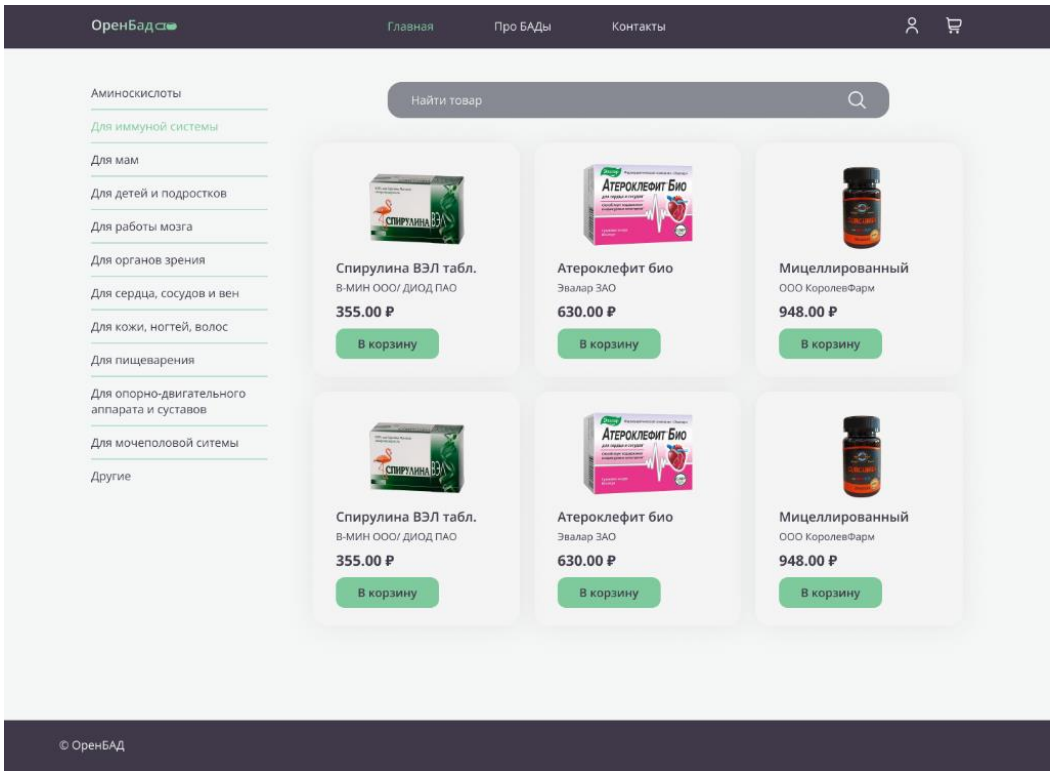

Рисунок Г1 – Дизайн главной страницы

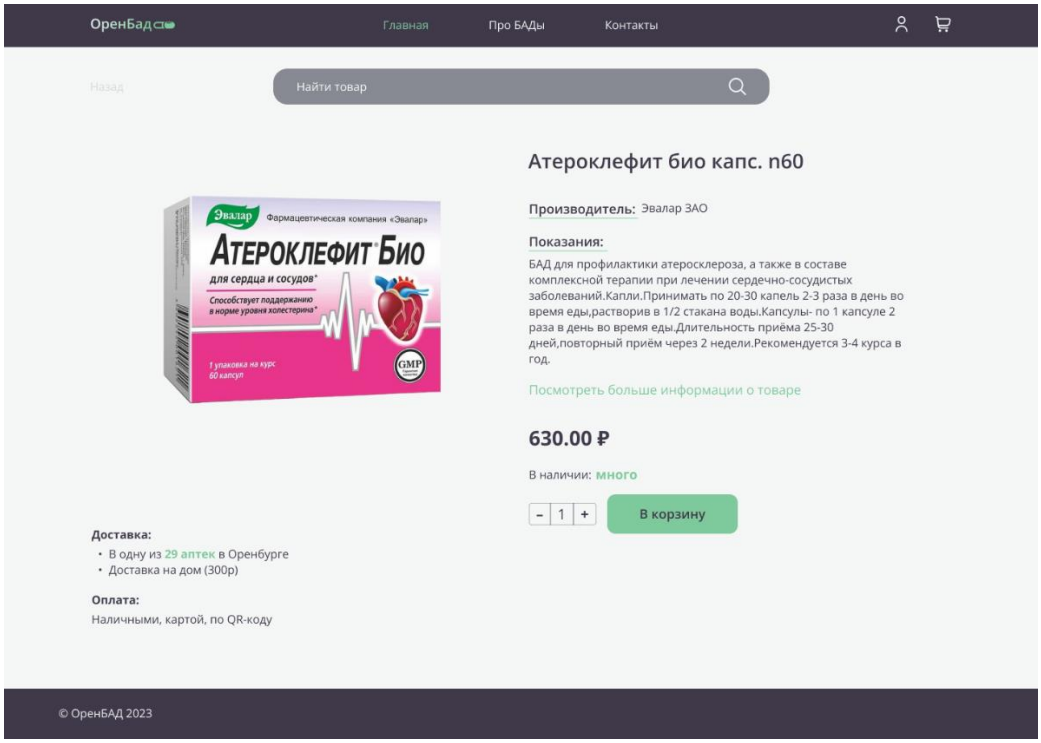

Рисунок Г2 – Дизайн страницы товара

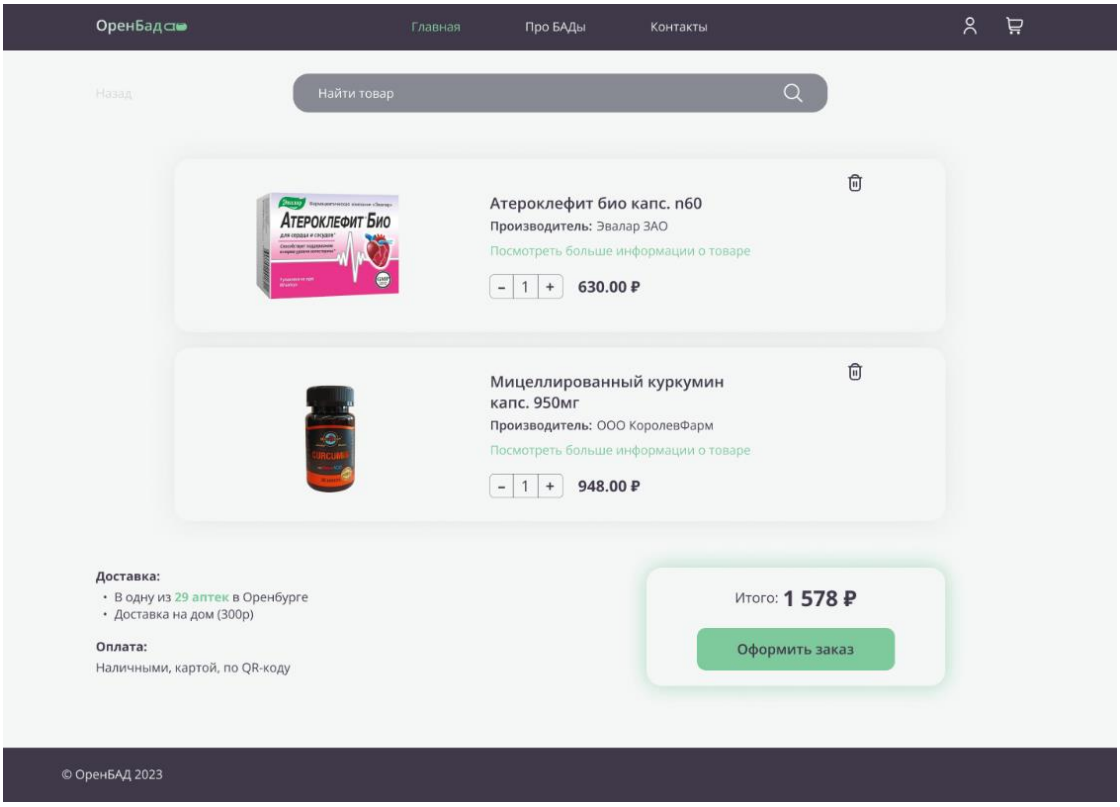

Рисунок Г3 – Дизайн корзины сайта

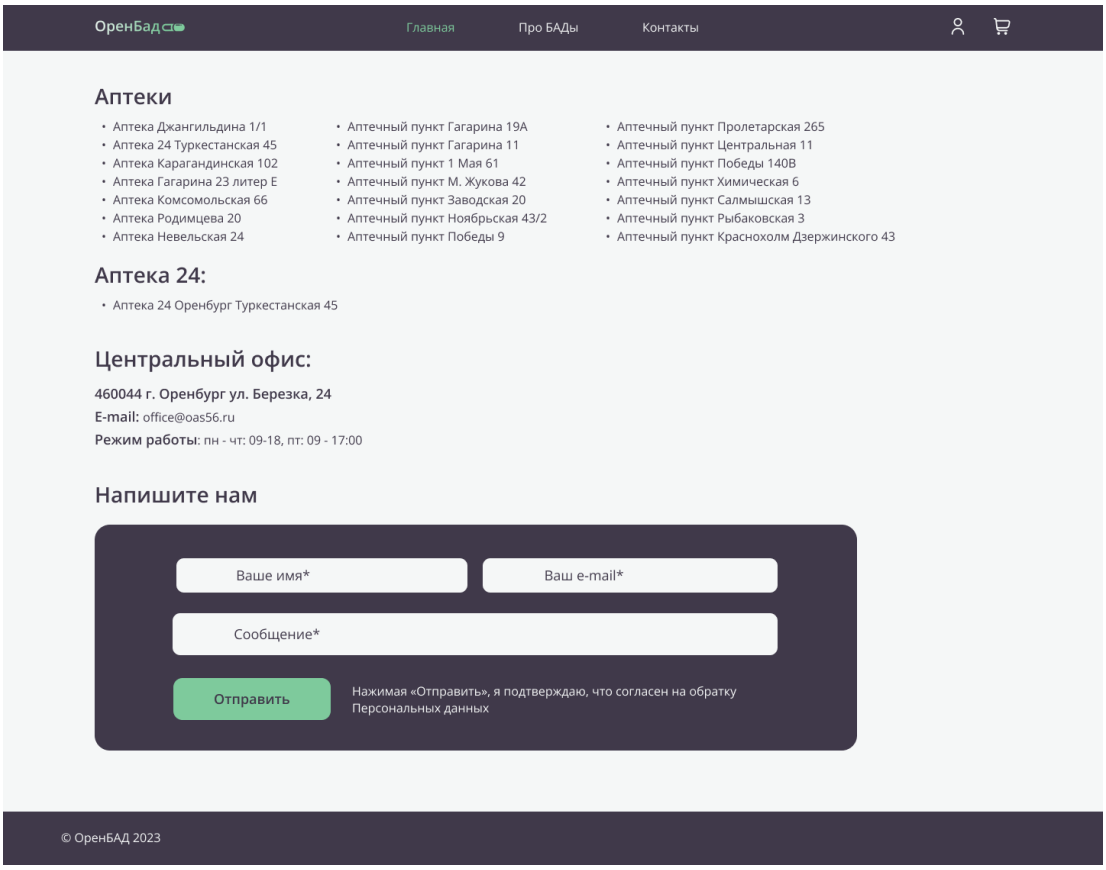

Рисунок Г4 – Дизайн страницы контактов предприятия

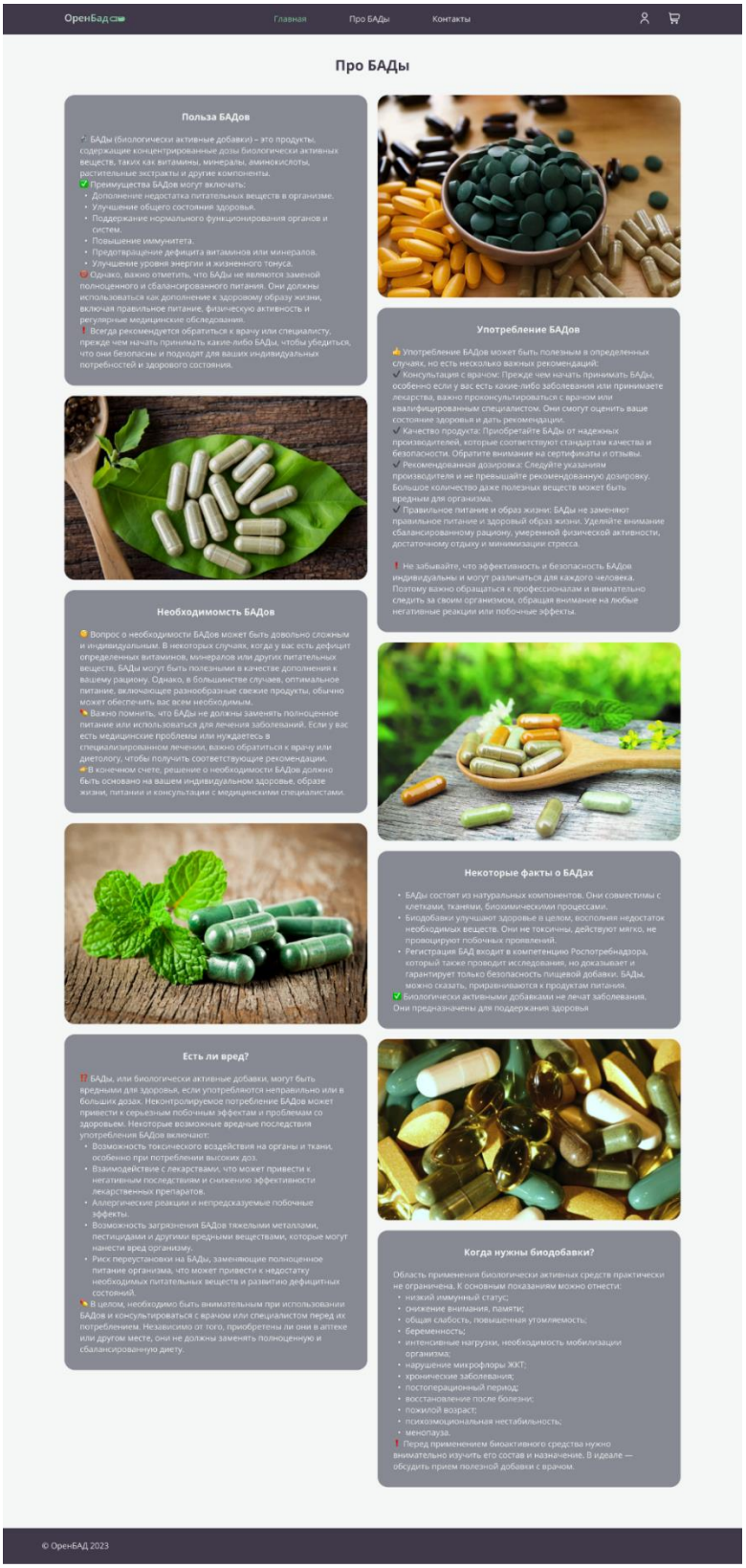

Рисунок Г5 - Дизайн страницы со статьями системы

Приложение Д
(обязательное)

Листинг программы

```
import express from "express"
import { router } from "./routes/OrenBad.routes.js"
import cors from 'cors'
const app = express()
const PORT = process.env.PORT || 8080
app.use(cors())
app.use(express.json())
app.use('/api', router)
app.listen(PORT, () => {
  console.log(`Example app listening on port ${PORT}`)})

import { db } from "../db.js";
import fs from "fs";
import path from "path";
class OrenBadControllerClass {
    async getCategory(req, res) {
        try {
            console.log("Connecting to the database...");
            const category = await db.query('SELECT * FROM public."category"');
            res.status(200).json(category.rows);
        } catch (error) {
            console.error("Error fetching category:", error);
            res.status(500).json({ error: "Internal Server Error" });}}
    async getPharmacy(req, res) {
        try {
            console.log("Connecting to the database...");
            const category = await db.query('SELECT * FROM public."pharmacy"
ORDER BY "namePharmacy"');
            res.status(200).json(category.rows);
        } catch (error) {
            console.error("Error fetching category:", error);
            res.status(500).json({ error: "Internal Server Error" }); }}
    async postProduct(req, res) {
        try {
            const { nameProduct, price, quantity, maker, category} = req.body;
            const img = req.files ? req.files.img : null;
            if (!img) {
                return res.status(400).json({ error: "No image uploaded" });
            }
```

```
                const imageName = `${Date.now()}_${img.name}`;
                const imagePath = `D:/REACT/OrenBadReact/server/img/${imageName}`;
                img.mv(imagePath, (err) => {
                  if (err) {
                    console.error("Error saving image:", err);
                    return res.status(500).json({ error: "Error saving image" });
                  }
                  // Insert product data into the database
                  db.query(
                    'INSERT INTO public."product" ( nameProduct, price, quantity, maker, category, img ) VALUES ($1, $2, $3, $4, $5, $6) RETURNING *',
                    [ nameProduct, price, quantity, maker, category, imageName],
                    (error, result) => {
                      if (error) {
                        console.error("Error inserting product:", error);
                        return res.status(500).json({ error: "Error inserting product into the database" });
                      }
                      // Remove the image from the server if needed (optional)
                      fs.unlinkSync(imagePath);

                      res.json({ success: true, product: result.rows[0] });});});
              } catch (error) {
                console.error("Error handling product upload:", error);
                res.status(500).json({ error: "Internal Server Error" });  }   }
            async getProduct(req, res) {
              try {
                console.log("Connecting to the database...");
                const products = await db.query//('SELECT * FROM public."product"');
                ('SELECT public."product".*, public."maker"."nameMaker" AS "maker" FROM public."product" LEFT JOIN public."maker" ON public."product"."maker" = public."maker"."idMaker" ');
                res.status(200).json(products.rows);
              } catch (error) {
                console.error("Error fetching products:", error);
                res.status(500).json({ error: "Internal Server Error" });}}
            async getProductId(req, res) {
              try {
                const { productId } = req.params;
                console.log("Connecting to the database...");
                const product = await db.query('SELECT public."product".*, public."maker"."nameMaker" AS "maker" FROM public."product" LEFT JOIN
```

```
public."maker" ON public."product"."maker" = public."maker"."idMaker" WHERE
product."productId" = $1', [productId]);
                if (product.rows.length === 0) {
                   return res.status(404).json({ error: "Product not found" });}
                res.status(200).json(product.rows[0]);} catch (error) {
                console.error("Error fetching product:", error);
        res.status(500).json({ error: "Internal Server Error" });  } }}
        export const OrenBadController = new OrenBadControllerClass();

        import { Router } from 'express'
        import { OrenBadController } from '../controllers/OrenBad.controller.js'
        export const router = new Router()
        router.get('/category', OrenBadController.getCategory)
        router.get('/pharmacy', OrenBadController.getPharmacy)
        router.get('/product', OrenBadController.getProduct)
        router.get('/product/:productId', OrenBadController.getProductId)
        router.post('/postProduct', OrenBadController.postProduct)

        import React from 'react'
        import ReactDOM from 'react-dom/client'
        import {
          createBrowserRouter,
          RouterProvider,
        } from "react-router-dom";
        import './index.css'
        import App from './App/App';
        import Contact from './Contact/Contact';
        import Article from './article/Article';
        import Account from './account/Account';
        import CardProduct from './CardProduct/CardProduct';
        import Registration from './Entry/Registration';
        import PostImg from './PostImg';
        const router = createBrowserRouter ([
        { path: "/",
          element: <App />},{path: "/contact",
          element: <Contact />},{path: '/article',
          element: <Article />},{path: '/account',
          element: <Account />},{ path: '/:productId',
          element: <CardProduct />},{path: '/post',
          element: <PostImg />},{path: '/entry',
          element: <Registration />}]);
        ReactDOM.createRoot(document.getElementById("root")).render(
          <RouterProvider router={router} />)
```

```
import React, { useEffect } from 'react'
import './Card.css'
export default function Card({ img, nameProduct, maker, price}) {
  return (<div class="product_card" >
      <img src={img} alt="product" />
      <div class="row_card_product">
        <div class="name_product">
          <h3 class="lettering_semi_bold">
            <div class="texturl">{nameProduct}</div></h3></div>
        <h4>{maker}</h4>
        <h2 class="lettering_bold product_rub">{price}</h2>
        <button type="button" class="card_product_btn">
          <span class="lettering_semi_bold">В корзину</span></button></div></div> )}

import React, { useEffect, useState } from 'react'
import './Cards.css'
import Card from './Card/Card'
import img from '../assets/spirulina.png'
import { useNavigate } from 'react-router-dom';

export default function Cards() {
   const navigate = useNavigate();
   const [array, setArray] = useState([])
   useEffect(() => {
      fetch('http://localhost:8080/api/product')
        .then(response => response.json())
        .then(json => {
          console.log(json); // Проверьте структуру
  setArray(json); // Оставьте эту строку без изменений})}, []);
    return (<div><div className="container_cards">
{ array.map((e) => {
return (<div onClick={ () => {navigate(`/${e.productId}`)}} key={e.productId}><Card key={e.idProduct} nameProduct={e.nameProduct} img={'../src/img/' + e.img} maker={e.maker} price={e.price} instruction={e.instruction} quantity={e.quantity}/>
                </div>  )}) }</div></div>)}

import React, { useEffect, useState } from 'react';
import './CardProduct.css';
import Header from '../components/Header/Header';
import Footer from '../components/Footer/Footer';
import Navigate from '../components/navigation/Navigate';
import FormSearch from '../components/FormSearch/FormSearch';
```

```
import Info_delivery from '../components/info_delivery/Info_delivery';
import Dropdown from '../components/dropdown/Dropdown';
import Btn from '../components/BTN/Btn';
import Counter from '../components/counter/Counter';
import { useParams } from 'react-router-dom';
export default function CardProduct() {
   const [product, setProduct] = useState({});
   let { productId } = useParams();
   useEffect(() => {
      fetch(`http://localhost:8080/api/product/${productId}`)
         .then(response => response.json())
         .then(data => {
            console.log(data); // Проверьте структуру данных
            setProduct(data); // Обновите состояние продукта
         })
         .catch(error => {
            console.error('Ошибка при получении данных о продукте:', error);
         });
   }, [productId]);
   return (
      <>  <Header />
         <div          className='container'><div          className="content"><div
className="search_row">
<Navigate categoryProduct='Для органов зрения' /><FormSearch /></div>
 <div className="mar-35"></div>
   <div className="main_block_catalog">
   <img src={`../src/img/${product.img}`} alt="product" />
   <div className="info_product">
   <h1 className='lettering_bold'>{product.nameProduct}</h1>
  <div className="information">
   <div  className="txt_row"><h3  className='lettering_semi_bold  bottom-
border'>Производитель:</h3> <p>{product.maker}</p></div>
    <div  className='block_txt'><h3  className='lettering_semi_bold  bottom-
border'>Показания:<br /> </h3> <p>{product.instruction}</p></div>
  <h3><a  className="txt_green"  href="https://www.rlsnet.ru/">Посмотреть
больше информации о товаре</a></h3>
   <h1 className='lettering_bold price'>{product.price}</h1> </div>
    <div className="add_order information"><Dropdown />
   <p  className='txt_row'>В  наличии:  <h3  className='lettering_semi_bold
txt_green'>много</h3></p>
   <div className="row_counter"><Counter />
   <Btn name_btn={"В корзину"} />
    </div> </div></div></div><Info_delivery />
</div></div><Footer /></>  )}
```