СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4

1.1. Анализ производственно-хозяйственной деятельности компании 4

1.2. Анализ инструментального программного обеспечения 6

1.3. Алгоритмы реализации задач приложения 9

1.4. Выбор языка программирования для реализации программного продукта 10

2. ПРАКТИЧЕСКАЯ РАБОТА 14

2.1. Проектирование пользовательского интерфейса 14

2.2. Функциональность разработки сайта 17

2.2.1. Система аутентификации 18

2.2.2. Каталог товаров 19

2.2.3. Корзина и заказы 23

2.2.4. Административная панель 26

2.2.5. Профиль пользователя 28

2.2.6. Отзывы на странице товара 29

2.3. Разработка базы данных 30

2.4. Тестирование и оптимизация 32

2.4.1. Тестирование компонентов 33

2.4.2. Интеграционное тестирование 34

2.4.3. Отладка и мониторинг 35

2.4.4. Оптимизация производительности 35

3. МАРКЕТИНГОВАЯ ЧАСТЬ 37

3.1. AptekaOnline как информационный продукт: управленческий взгляд 37

3.2. Анализ рынка и определение целевой аудитории 38

3.3. Жизненный цикл информационного продукта AptekaOnline 41

3.3.1. Этап зарождения идеи и концептуализации 41

3.3.2. Этап проектирования и разработки 42

3.3.3. Этап запуска и внедрения 42

3.3.4. Этап роста и развития 43

3.3.5. Этап зрелости и оптимизации 43

3.3.6. Перспективы развития и трансформации 44

3.4. Выводы и рекомендации 44

# ВВЕДЕНИЕ

Актуальность разработки интернет-аптеки обусловлена растущим спросом на онлайн-покупку лекарственных препаратов. Современные пользователи все чаще предпочитают заказывать лекарства через интернет, что позволяет сэкономить время и получить доступ к более широкому ассортименту товаров. Кроме того, онлайн-платформа предоставляет возможность получить подробную информацию о препаратах, сравнить цены и прочитать отзывы других покупателей. Проект также учитывает важность безопасности данных, особенно в контексте медицинской информации. Особое внимание уделено удобству использования интерфейса, что делает процесс покупки лекарств максимально простым и понятным.

Цель проекта — создание полноценной системы, которая обеспечивает удобный доступ к лекарственным препаратам через интернет. Для достижения этой цели были поставлены и решены следующие задачи:

- Реализация системы регистрации и авторизации пользователей через номер телефона.

- Создание каталога товаров с возможностью поиска, фильтрации и детального просмотра характеристик препаратов.

- Разработка функционала корзины покупок и системы оформления заказов.

- Обеспечение административной панели для управления контентом, товарами и пользователями.

- Интеграция механизмов безопасности для защиты персональных данных и истории заказов.

- Оптимизация пользовательского интерфейса для повышения удобства и снижения порога входа для новых пользователей.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## **1.1. Анализ производственно-хозяйственной деятельности компании**

ООО «Технологии программирования» представляет собой динамично развивающуюся компанию в сфере разработки программного обеспечения, которая, несмотря на статус малого предприятия, демонстрирует высокие показатели эффективности благодаря грамотному внедрению современных информационных технологий. Основная специализация компании сосредоточена на создании и обслуживании баз данных, а также разработке специализированного программного обеспечения для бизнеса. За годы работы компания выработала уникальную модель управления, сочетающую преимущества малого бизнеса с профессиональным подходом к реализации ИТ-проектов.

Организационная структура компании отличается минимальной иерархией, что является характерной чертой успешных малых предприятий в ИТ-сфере. Такая структура управления позволяет оперативно реагировать на изменения рынка и быстро адаптироваться к новым технологическим вызовам. Основные производственные процессы распределены между несколькими функциональными подразделениями, каждое из которых отвечает за определенный участок работы. Финансовый отдел обеспечивает стабильность экономических показателей, в то время как производственный отдел, состоящий из опытных программистов, занимается непосредственной разработкой программных продуктов. Особое внимание уделяется отделу продаж и маркетинга, который играет ключевую роль в продвижении услуг компании на рынке.

Техническая оснащенность предприятия соответствует современным стандартам ИТ-индустрии. Компания активно использует передовые технологии разработки программного обеспечения, включая современные системы управления базами данных и инструменты автоматизации бизнес-процессов. Все рабочие места сотрудников оснащены необходимым программным обеспечением, а корпоративная сеть построена с учетом требований информационной безопасности. Особое значение придается вопросам стандартизации и сертификации - компания строго соблюдает требования как национальных (ГОСТ, СНиП), так и международных стандартов (ISO серии 9001 и 45001), что подтверждает высокое качество предоставляемых услуг.

Производственный процесс в компании организован с учетом специфики разработки программного обеспечения. Основное внимание уделяется этапам проектирования архитектуры решений, непосредственного программирования, тестирования и внедрения готовых продуктов. Для управления проектами используются современные методологии, такие как Agile и Scrum, которые позволяют гибко реагировать на изменение требований заказчиков.

В последние годы компания активно инвестирует в информатизацию своих бизнес-процессов. Внедрение специализированного программного обеспечения для управления проектами, документооборотом и ресурсами позволило значительно повысить эффективность работы. Автоматизация рутинных операций дала возможность перераспределить трудовые ресурсы на более сложные и важные задачи, что положительно сказалось на качестве конечного продукта. Особое внимание уделяется системам аналитики, которые помогают руководству принимать обоснованные управленческие решения на основе актуальных данных.

Перспективы развития компании связаны с дальнейшим совершенствованием технологической базы и расширением спектра предоставляемых услуг. В планах руководства - активное внедрение облачных технологий, развитие компетенций в области больших данных и машинного обучения, а также выход на новые рыночные сегменты. При этом компания намерена сохранить свои конкурентные преимущества, связанные с мобильностью и гибкостью, которые характерны для малых предприятий. Опыт успешной работы в условиях высокой конкуренции позволяет с оптимизмом смотреть в будущее и ставить амбициозные цели по дальнейшему развитию бизнеса.

## 1.2. Анализ инструментального программного обеспечения

В ходе анализа инструментального программного обеспечения для реализации проекта были тщательно рассмотрены различные технологические стеки, чтобы выбрать оптимальный баланс между производительностью, удобством разработки и возможностями масштабирования. Для клиентской части первоначально рассматривались такие популярные фреймворки, как Angular и Vue.js , помимо React . Angular, например, предлагает мощный инструментарий с встроенными функциями (например, DI, RxJS), но его относительно высокий порог вхождения и строгая архитектура делают его менее гибким для небольших и средних проектов. Vue.js, в свою очередь, известен простотой интеграции и постепенного внедрения, однако его экосистема уступает React по разнообразию готовых решений. React, благодаря своей компонентной архитектуре, виртуальному DOM и активному сообществу, обеспечивает высокую производительность и возможность повторного использования кода, что особенно важно при разработке динамических интерфейсов. В сочетании с CSS-фреймворками, такими как TailwindCSS или Material-UI, React позволяет создавать адаптивные и визуально насыщенные интерфейсы за короткие сроки, что стало ключевым фактором в его выборе.

Для управления HTTP-запросами альтернативами Axios могли стать встроенный Fetch API или библиотеки вроде SuperAgent . Fetch API, несмотря на отсутствие зависимости от внешних библиотек, имеет ограниченный функционал (например, отсутствие встроенной поддержки автоматической обработки ошибок или таймеров). SuperAgent, хотя и предлагает удобный синтаксис, менее популярен и не поддерживает современные стандарты, такие как AbortController, так же эффективно, как Axios. Последний же предоставляет унифицированный интерфейс для работы с REST-запросами, автоматически сериализует/десериализует JSON и поддерживает перехватчики (interceptors), что упрощает централизованную обработку запросов и ответов.

Маршрутизация с помощью React Router была выбрана вместо встроенных решений в фреймворках вроде Next.js . Хотя Next.js предлагает преимущества в виде SSR (серверного рендеринга) и автоматической оптимизации, для текущего проекта, где акцент сделан на SPA (одностраничное приложение) без необходимости SEO-оптимизации, React Router оказался более легковесным и гибким решением. Он позволяет динамически управлять маршрутами, поддерживает lazy-loading компонентов и интегрируется с React Hooks, что упрощает навигацию без перезагрузки страницы.

На серверной стороне Node.js рассматривался в сравнении с традиционными стеками, такими как Python (Django/Flask) , Java (Spring Boot) и Ruby on Rails . Node.js выигрывает за счет своей асинхронной, неблокирующей архитектуры, которая идеально подходит для обработки множества параллельных запросов, характерных для высоконагруженных приложений. В отличие от синхронных фреймворков, таких как Django, Node.js не требует создания отдельных потоков для каждого запроса, что снижает потребление ресурсов. Среди фреймворков для Node.js выбран Express вместо Koa или NestJS . Koa, созданный теми же разработчиками, что и Express, предлагает более современный подход с использованием async/await, но имеет меньшее количество готовых middleware. NestJS, напротив, ориентирован на крупные приложения с использованием TypeScript и ООП-подхода, но для текущего проекта его сложность избыточна. Express сочетает минимализм, гибкость и обширное сообщество, что позволяет быстро настраивать маршруты, middleware и интеграции.

Для хранения данных рассматривались как реляционные (PostgreSQL , MySQL ), так и нереляционные (MongoDB , Firebase ) базы данных. SQLite была выбрана из-за своей легковесности, отсутствия необходимости в отдельном сервере и простоты развертывания — идеально для MVP или проектов среднего масштаба, где важна скорость запуска и минимальные требования к инфраструктуре. PostgreSQL, например, предложила бы более продвинутые функции (например, поддержку JSONB, геопространственные данные), но потребовала бы настройки сервера и управления подключениями. MongoDB, в свою очередь, подошла бы для гибких схем данных, но усложнила бы реализацию транзакций и отношений между сущностями, характерных для данного проекта.

В части аутентификации сравнивались JWT , OAuth 2.0 и классические сессии с использованием кук. JWT выбран за счет его stateless-природы, что упрощает масштабирование серверов, и возможности передачи данных без необходимости хранения состояния на сервере. OAuth 2.0, хотя и обеспечивает высокий уровень безопасности и интеграции с третьими сторонами (например, вход через Google), избыточен для проекта, где не требуется поддержка сторонних сервисов. Сессии на куках, в свою очередь, уязвимы к CSRF-атакам и требуют хранения данных на сервере (например, в Redis), что увеличивает нагрузку.

Инструментальный стек также анализировался в контексте альтернатив. Git был выбран как де-факто стандарт контроля версий, в отличие от SVN или Mercurial, благодаря распределенной архитектуре и интеграции с платформами вроде GitHub/GitLab. npm стал предпочтительнее Yarn или pnpm из-за встроенной поддержки в экосистеме Node.js и отсутствия необходимости в дополнительных зависимостях, хотя Yarn предлагает более быструю установку пакетов. VSCode выбран среди редакторов (например, WebStorm, Sublime Text) благодаря бесплатности, широким возможностям кастомизации, встроенной поддержке Git и богатому набору расширений для отладки, тестирования и автодополнения.

Итоговый выбор технологий обусловлен их способностью решать задачи проекта с минимальной «боевой» нагрузкой на команду разработчиков, учитывая баланс между производительностью, простотой внедрения и долгосрочной поддержкой. Например, использование SQLite вместо PostgreSQL снизило порог входа для новых разработчиков, а React сократил время на создание прототипов благодаря готовым компонентам. В совокупности выбранный стек обеспечивает гибкость для будущих модернизаций: при необходимости переход на NestJS для серверной части или замена SQLite на PostgreSQL может быть выполнен без полной переписки кода, что подтверждает правильность сделанного выбора.

## 1.3. Алгоритмы реализации задач приложения

Алгоритм авторизации пользователей:

- Пользователь вводит учетные данные (логин и пароль) в форму входа.

- Клиентская часть выполняет валидацию данных (проверка на пустые поля, минимальную длину пароля и т. д.).

- При успешной валидации отправляется HTTP-запрос (POST) на сервер с введенными данными.

- Сервер проверяет наличие пользователя в базе данных и сверяет хеш пароля.

- В случае успеха генерируется JWT-токен, который возвращается клиенту.

- Токен сохраняется в локальном хранилище браузера (localStorage или cookies) для последующей аутентификации.

Алгоритм работы с заказами:

- Пользователь добавляет товары в корзину, которая хранится в состоянии клиентского приложения.

- При оформлении заказа данные корзины отправляются на сервер (POST-запрос).

- Сервер проверяет JWT-токен для подтверждения авторизации пользователя.

- В случае успеха создается новая запись в таблице заказов базы данных с указанием пользователя, списка товаров и статуса.

- Клиенту возвращается подтверждение о successful создании заказа.

Алгоритм работы администратора:

- При входе в панель администратора проверяются права доступа (роль пользователя из JWT-токена).

- Сервер возвращает список всех заказов из базы данных.

- На клиенте реализована фильтрация и сортировка заказов (по дате, статусу, ID пользователя).

- Администратор может изменять статус заказа (например, «В обработке», «Отправлен», «Доставлен»), что обновляется в базе данных.

## 1.4. Выбор языка программирования для реализации программного продукта

Выбор языка программирования для проекта проводился с учетом множества факторов, включая требования к производительности, масштабируемости, безопасности, удобству разработки, наличию экосистемы и возможности единого стека для клиентской и серверной частей. Основным языком был выбран JavaScript , а точнее его расширенная версия — TypeScript , которая добавляет статическую типизацию и другие современные возможности. Это решение было принято после тщательного сравнения с альтернативными языками, такими как Python , Java , C# , Go и Ruby , каждый из которых имеет свои сильные стороны, но уступает в контексте специфики данного проекта.

JavaScript/TypeScript выделились за счет уникального сочетания универсальности и экосистемы. JavaScript, как нативный язык веб-браузеров, обеспечивает прямую интеграцию с HTML/CSS, что критично для клиентской части. Однако его динамическая типизация, хотя и гибкая, часто приводит к труднообнаруживаемым ошибкам на этапе выполнения. Здесь на помощь приходит TypeScript , который добавляет статическую типизацию, поддержку интерфейсов, классов и современных стандартов ECMAScript (например, async/await, декораторы). Это снижает количество ошибок, упрощает рефакторинг и делает код более читаемым, особенно в командах. Например, в сравнении с Python, где динамическая типизация также доминирует, TypeScript обеспечивает более строгий контроль на этапе разработки, что особенно важно для масштабных проектов.

Для серверной части выбор в пользу Node.js с использованием TypeScript позволил создать единую структуру, где и клиент, и сервер работают на одном языке. Это исключает необходимость переключения между различными синтаксисами (например, Python для бэкенда и JavaScript для фронтенда), упрощает обмен данными через JSON и позволяет повторно использовать код (например, валидацию форм или бизнес-логику). Альтернативные решения, такие как Python (Django/Flask) или Java (Spring Boot) , предлагают мощные серверные фреймворки, но требуют отдельных команд для фронтенда и бэкенда, что увеличивает затраты времени на координацию. Например, Python, несмотря на простоту синтаксиса и богатство библиотек для машинного обучения, менее эффективен в задачах, связанных с высокой нагрузкой и асинхронной обработкой данных, где Node.js демонстрирует лучшие результаты благодаря своей событийно-ориентированной архитектуре.

Java и C# , в свою очередь, славятся своей надежностью и производительностью, особенно в корпоративных системах. Однако их статичность и сложность настройки (например, необходимость компиляции и управления JVM) делают их менее подходящими для быстрой итерации, характерной для стартапов или MVP-проектов. TypeScript, напротив, позволяет использовать преимущества JIT-компиляции V8 (движка Node.js), которая обеспечивает высокую скорость выполнения кода, а также интеграцию с Hot Module Replacement (HMR) в инструментах вроде Webpack, что ускоряет разработку.

Go (Golang) часто позиционируется как конкурент Node.js в сценариях высокой производительности благодаря своей конкурентности и компиляции в машинный код. Однако его молодая экосистема и отсутствие встроенной поддержки асинхронного программирования на уровне стандартной библиотеки (в отличие от Node.js) усложняют реализацию типичных веб-задач, таких как обработка HTTP-запросов или работа с базами данных. TypeScript, в сочетании с Express, предоставляет готовые решения для этих задач, что сокращает время разработки.

В части безопасности TypeScript также демонстрирует преимущество перед языками с динамической типизацией. Статическая проверка типов позволяет выявлять потенциальные уязвимости (например, injection-атаки через некорректную обработку данных) еще на этапе написания кода. Например, в сравнении с Ruby (Ruby on Rails), где динамизм языка иногда приводит к runtime-ошибкам, TypeScript устраняет подобные риски за счет строгой системы типов. Кроме того, использование JWT для аутентификации, описанное ранее, гармонично сочетается с возможностями TypeScript, такими как строгая типизация объектов токенов и централизованное управление middleware.

Масштабируемость в TypeScript обеспечивается не только за счет Node.js, но и благодаря инструментам вроде Webpack , Babel и ESLint , которые позволяют оптимизировать сборку, поддерживать совместимость с устаревшими браузерами и соблюдать стандарты кодирования. Например, в сравнении с PHP (Laravel), где масштабирование часто требует перехода на микросервисную архитектуру с нуля, TypeScript и Node.js изначально проектировались для горизонтального масштабирования, что упрощает добавление новых сервисов или контейнеризацию через Docker.

Удобство разработки усиливается интеграцией с VSCode , который предоставляет мощные инструменты для TypeScript: автодополнение, навигацию по определениям, проверку типов в реальном времени и интеграцию с Jest/Mocha для тестирования. Это повышает продуктивность разработчиков по сравнению с языками, где подобные функции требуют установки сторонних плагинов (например, Python в сочетании с PyCharm).

Наконец, сообщество и поддержка играют ключевую роль. JavaScript/TypeScript имеют самую большую экосистему npm, где доступно более 2 миллионов пакетов. Это позволяет быстро находить решения для любых задач — от работы с графикой (например, Three.js) до управления состоянием (Redux, MobX). В сравнении с Go или Ruby, где количество пакетов существенно меньше, TypeScript обеспечивает гибкость и скорость реализации функционала.

Итоговый выбор JavaScript/TypeScript обусловлен их способностью объединить клиентскую и серверную части в единый стек, сократить время разработки за счет готовых решений и повысить надежность кода через статическую типизацию. Хотя альтернативные языки, такие как Java или Python, могут быть предпочтительнее в узкоспециализированных сценариях (например, для проектов с интенсивной обработкой данных), для данного проекта, где важны скорость вывода на рынок, гибкость и поддержка сообщества, TypeScript стал оптимальным решением. Его сочетание с Node.js, React и современными инструментами позволяет создать высокопроизводительное приложение, которое легко масштабировать и поддерживать на протяжении всего жизненного цикла.

# 2. ПРАКТИЧЕСКАЯ РАБОТА

## 2.1. Проектирование пользовательского интерфейса

Перед реализацией кода, был разработан макет интерфейса в Figma, отражающий продуманную пользовательскую логику. Основное внимание уделялось созданию удобного и интуитивно понятного интерфейса, который обеспечивал бы легкое взаимодействие с пользователями.

Главная страница представляет собой чистый и интуитивно понятный каталог с возможностью фильтрации по категориям, цене и наличию товаров.

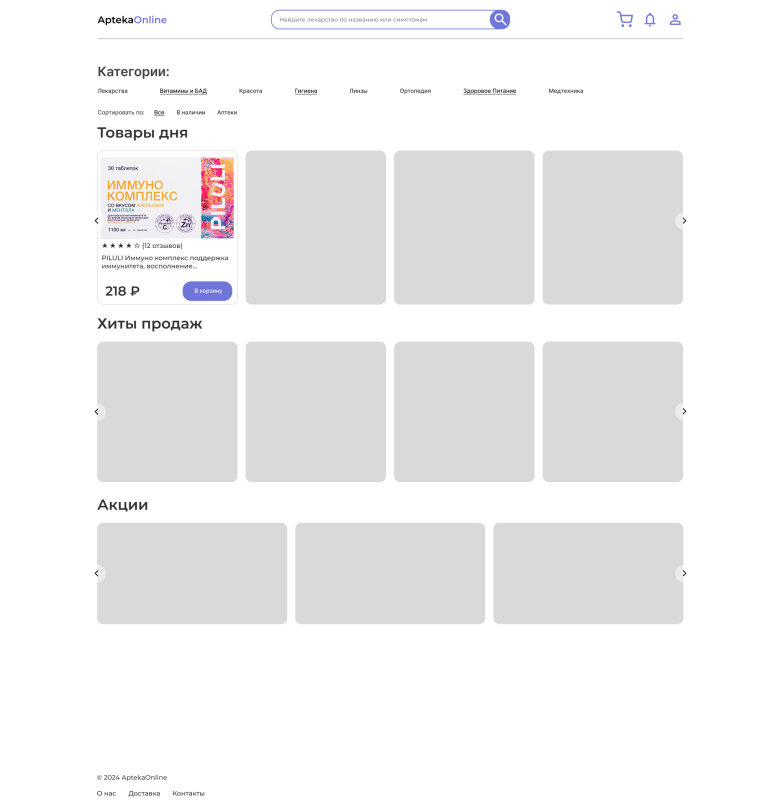


Рисунок 1 – Главная страница

Страница выполнена в спокойной цветовой гамме с преобладанием синих и белых тонов, что создает ощущение чистоты и надежности, ассоциирующееся с медицинской тематикой.

При выборе конкретного товара на главной странице система мгновенно перенаправляет пользователя на отдельную страницу, посвящённую данному товару.

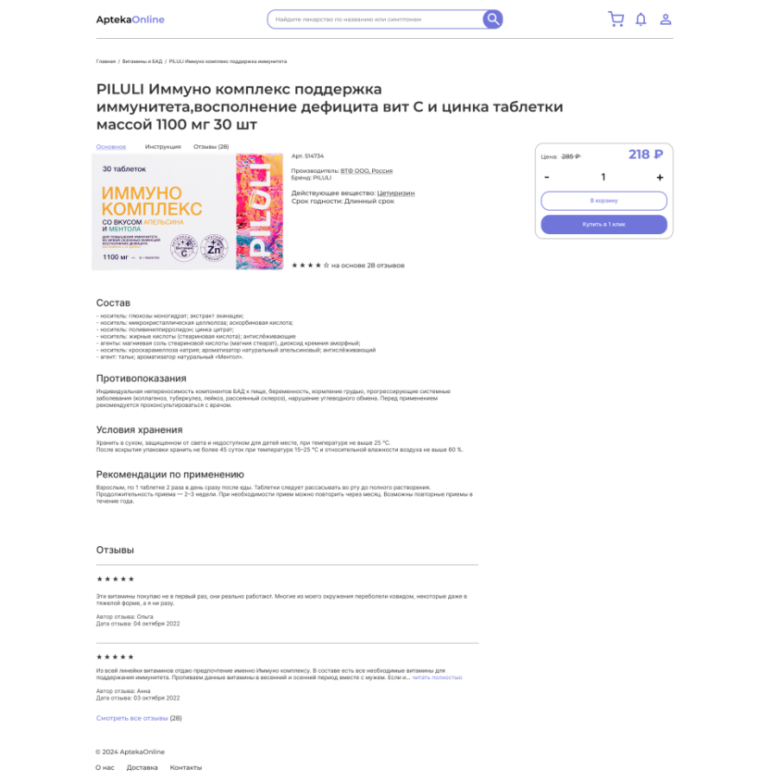


Рисунок 2 – Страница товара

Страница товара содержит всю необходимую информацию о препарате, включая изображение, описание

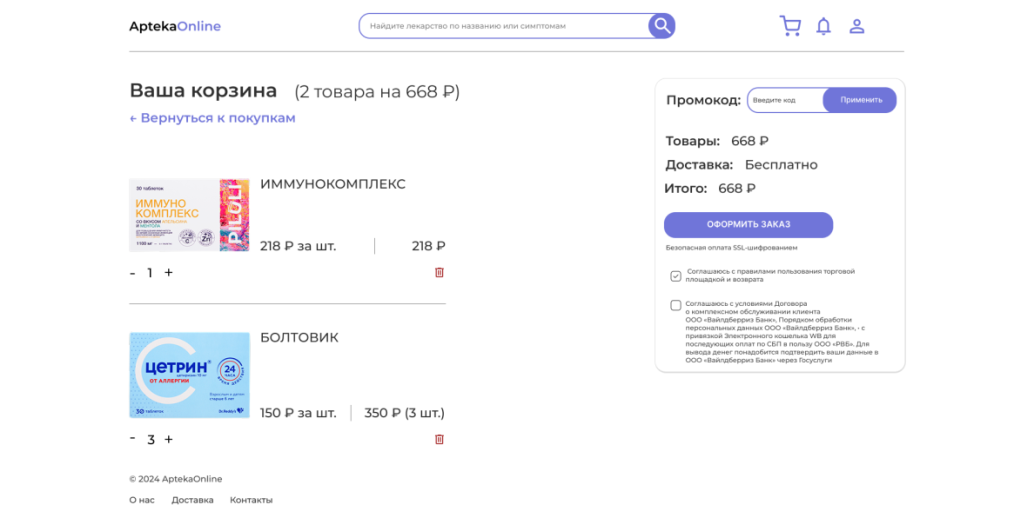


Рисунок 3 – Корзина

Интерфейс корзины позволяет легко управлять составом заказа, изменять количество товаров и переходить к оформлению.

Из шапки страницы авторизированные пользователи могут зайти в личный кабинет.

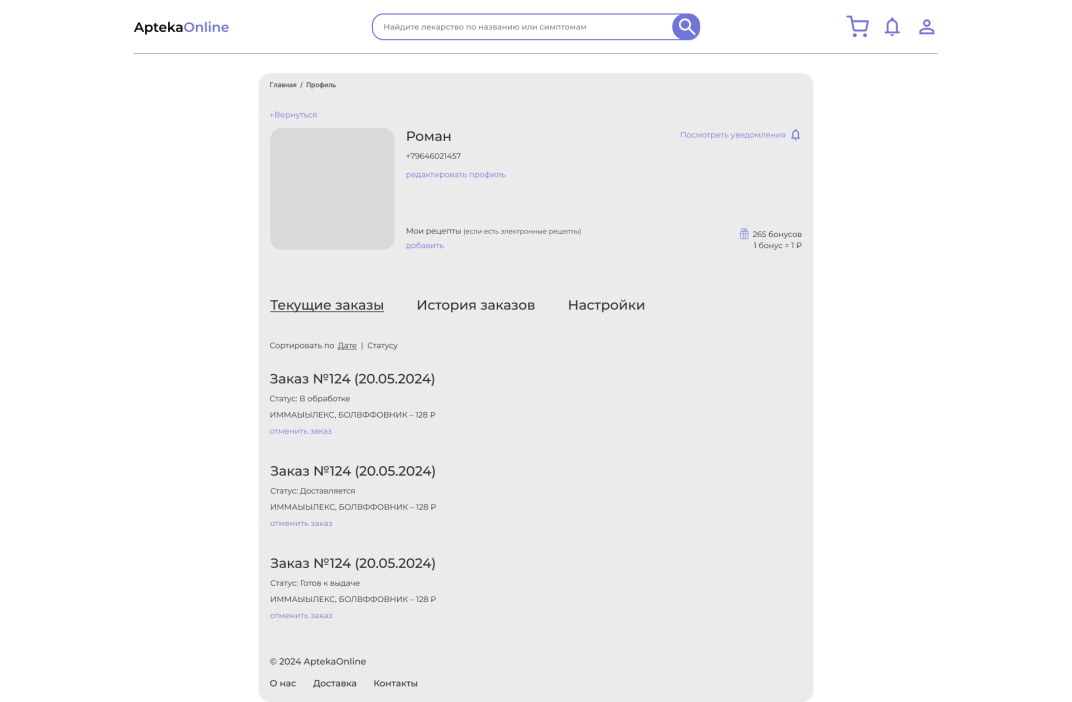


Рисунок 4 – Личный кабинет

Личный кабинет разделен на две логические части: управление персональными данными и просмотр истории заказов. Также в системе реализована административная панель для управления ассортиментом и обработки заказов, доступная только авторизованным сотрудникам с соответствующими правами.

Для реализации проекта была выбрана комбинированная модель разработки, сочетающая элементы каскадной и гибкой методологий. На начальном этапе по принципам Waterfall были выполнены проектирование базы данных, создание базовой структуры проекта и разработка основных модулей аутентификации. Это позволило заложить прочный фундамент для последующей работы.

После завершения базовой реализации процесс перешел в итеративную фазу по методологии Agile. Разработка велась спринтами, каждый из которых был посвящен реализации определенного функционального блока. Первый спринт включал создание системы фильтрации товаров с возможностью сортировки по различным параметрам. Второй спринт был посвящен разработке корзины покупок с использованием локального хранилища для временного сохранения выбранных товаров. Третий спринт охватывал создание системы оформления заказов и просмотра истории покупок. Четвертый спринт включал доработку пользовательского интерфейса и реализацию дополнительных функций, таких как поиск по каталогу и система уведомлений.

## 2.2. Функциональность разработки сайта

Разработка функциональности сайта интернет-аптеки осуществлялась с учетом современных требований к веб-приложениям и специфики фармацевтического бизнеса. Основной целью было создание интуитивно понятного и удобного интерфейса, обеспечивающего быстрый доступ к необходимой информации и простоту совершения покупок.

При разработке функциональности особое внимание было уделено безопасности и надежности системы. Все компоненты реализованы с использованием современных практик разработки и проверенных технологий. Система построена на принципах модульности, что позволяет легко расширять функциональность и поддерживать код в актуальном состоянии.

Особое внимание было уделено оптимизации производительности и скорости загрузки страниц. Реализованы механизмы кэширования данных, оптимизированы запросы к серверу, внедрена постраничная загрузка контента. Это позволяет обеспечить быструю работу сайта даже при большом количестве пользователей и товаров.

Разработка функциональности велась с учетом потребностей различных категорий пользователей: от обычных покупателей до администраторов системы. Для каждой категории реализован свой набор функций и свой уровень доступа к информации. Это обеспечивает удобство использования системы для всех участников процесса.

### 2.2.1. Система аутентификации

Система аутентификации реализована в компонентах `LoginRegisterModal.jsx`, `LoginForm.jsx` и `RegisterForm.jsx`.



Рисунок 5 – Реализация логики аутентификации пользователей

Основная логика аутентификации построена на использовании JWT-токенов, которые генерируются при успешной авторизации пользователя.

Формы авторизации и регистрации реализованы в виде модальных окон, что обеспечивает удобный доступ к функциям аутентификации с любой страницы сайта. Интерфейс форм выполнен в минималистичном стиле с четкой структурой полей ввода. При регистрации пользователь должен указать номер телефона и пароль, подтвердить пароль, а также дать согласие на обработку персональных данных.

Процесс регистрации включает проверку корректности номера телефона и пароля. Все данные проходят валидацию перед отправкой на сервер, что обеспечивает целостность и безопасность данных.

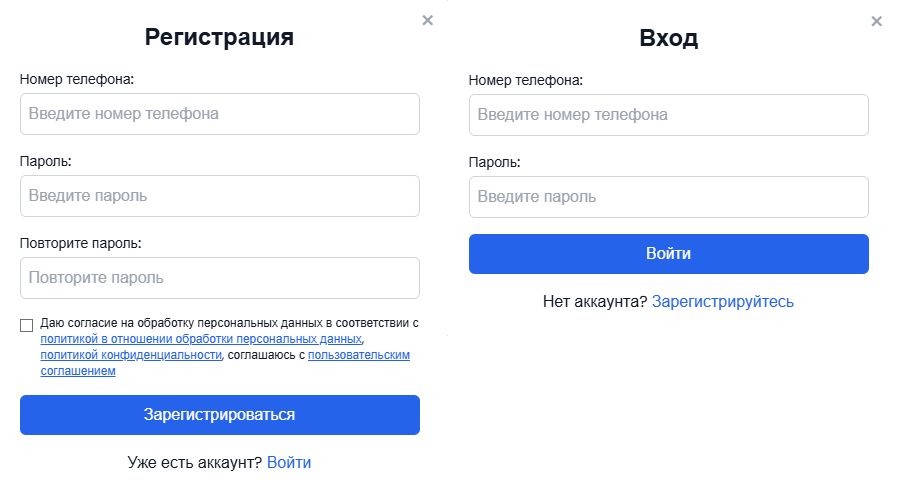


Рисунок 6 – Интерфейс модальных окон входа и регистрации

Система автоматически проверяет корректность введенных данных и выводит понятные сообщения об ошибках. При успешной авторизации или регистрации модальное окно закрывается, а пользователь получает доступ к персонализированным функциям сайта.

### 2.2.2. Каталог товаров

Каталог товаров реализован в компонентах `HomePage.jsx` и `ProductList.jsx`. Основная логика работы с товарами построена на взаимодействии с API сервера и отображении данных в удобном для пользователя формате.

Данный фрагмент кода представляет собой основную логику работы с каталогом товаров. При загрузке страницы или изменении параметров фильтрации (поиск, категории, цена, наличие товара) происходит автоматический запрос к серверу для получения списка товаров. В запросе передаются все необходимые параметры: ключевое слово для поиска, номер текущей страницы, выбранные категории товаров, диапазон цен, наличие товара на складе и способ сортировки.

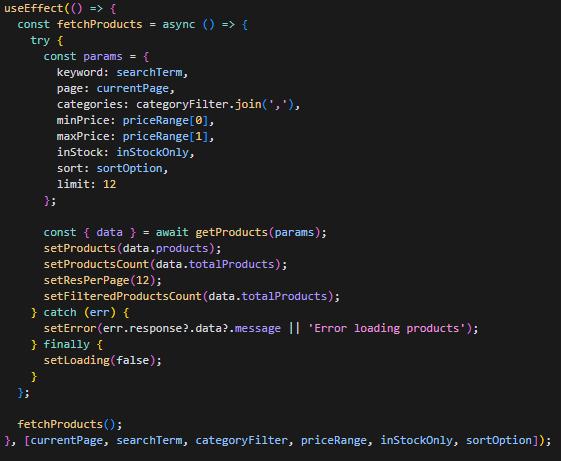


Рисунок 7 – Реализация логики загрузки и фильтрации товаров в каталоге

После получения данных с сервера, они сохраняются в состоянии компонента для отображения пользователю. В случае возникновения ошибки при загрузке данных, пользователю показывается соответствующее сообщение.

Карточка товара представляет собой компактный информационный блок, содержащий все необходимые данные о продукте. В верхней части карточки размещено изображение товара, которое при наведении курсора увеличивается для детального просмотра. Под изображением расположена основная информация: название товара, цена и категория. В нижней части карточки находится кнопка добавления в корзину.

При нажатии на карточку пользователь перенаправляется на страницу с полным описанием товара, где представлена расширенная информация, включая подробные характеристики, инструкцию по применению и отзывы покупателей.

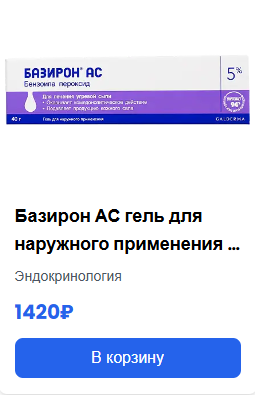


Рисунок 8 – Интерфейс карточки товара

Администраторы могут добавлять новые товары через специальную форму в административной панели. При добавлении товара происходит валидация всех полей, включая проверку формата цены, наличия обязательных полей и корректности изображения.

Поиск товаров реализован в компоненте `SearchPage.jsx` и имеет следующие особенности:

- Автоматическое предложение вариантов при вводе текста

- Сохранение истории поиска для быстрого доступа к часто искомым товарам

- Возможность фильтрации результатов по цене, категории и наличию

При вводе поискового запроса или изменении параметров фильтрации происходит автоматический запрос к серверу.



Рисунок 9 – Реализация логики загрузки и фильтрации товаров

Система также поддерживает пагинацию результатов поиска, что позволяет просматривать большое количество товаров постранично.

Поле поиска расположено в верхней части страницы и имеет интуитивно понятный дизайн.

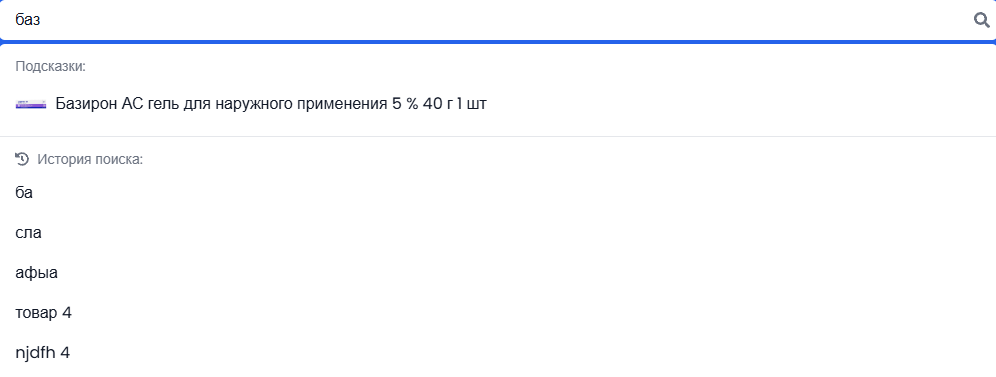


Рисунок 10 - Интерфейс поиска товаров

Результаты поиска отображаются в виде карточек товаров с основной информацией.

### 2.2.3. Корзина и заказы

Система корзины реализована в компоненте `CartPage.jsx`, а процесс оформления заказа - в `CheckoutModal.jsx`. Пользователи могут добавлять товары в корзину, изменять их количество и оформлять заказы.

В корзине используются модули, такие как `CartReducer` для управления состоянием товаров, `CartActions` для выполнения действий, а также `CartSelector` для получения данных о выбранных товарах. Система также включает модуль `CartTotal` для расчета итоговой стоимости заказа, который автоматически обновляется при изменении содержимого корзины.

Корзина взаимодействует с бэкендом через API-запросы, которые позволяют добавлять, удалять и изменять количество товаров. Система автоматически обновляет общую стоимость заказа при изменении количества товаров.

Интерфейс корзины и оформления заказа реализован с учетом удобства пользователя и эффективности процесса покупки. Корзина отображает список выбранных товаров с возможностью изменения их количества или удаления.

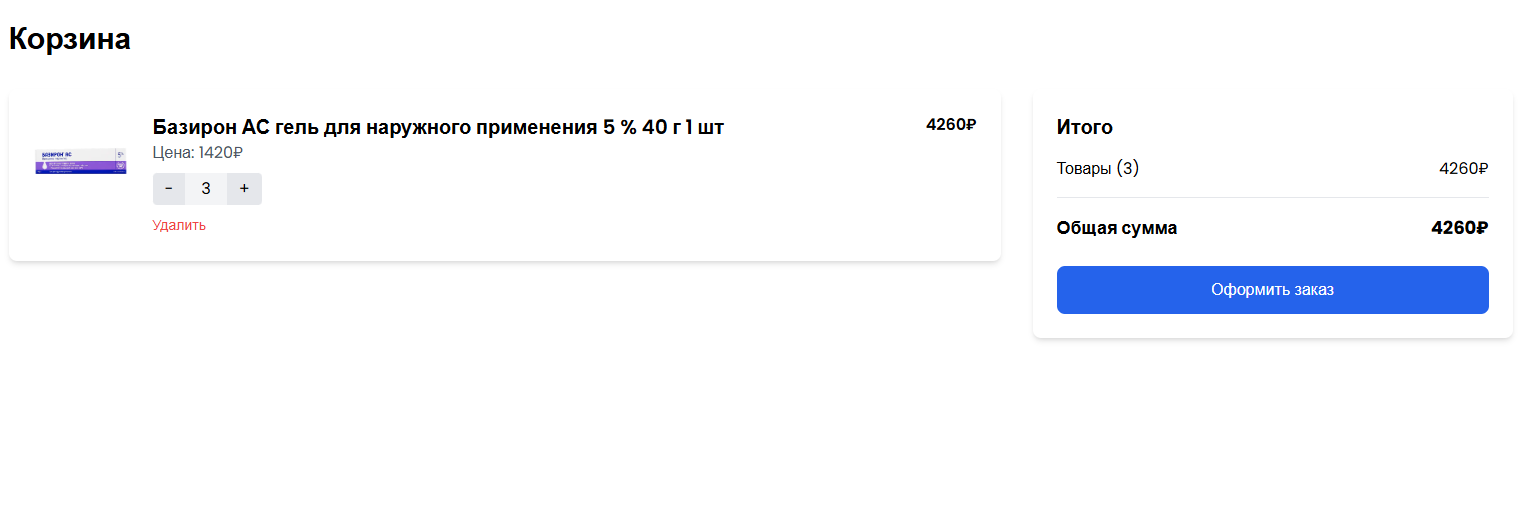


Рисунок 11 - Интерфейс корзины

При добавлении товаров в корзину система автоматически проверяет их наличие на складе и актуальные цены. Это позволяет избежать ситуаций, когда пользователь пытается заказать товар, которого уже нет в наличии или цена которого изменилась.

При оформлении заказа пользователь проходит через несколько этапов. Сначала он вводит свои контактные данные и адрес доставки. Для удобства выбора адреса доставки реализована интеграция с Яндекс.Картами, которая отображает карту с расположением аптек и пунктов выдачи. На карте отмечены все доступные точки выдачи с информацией о режиме работы и контактных данных. Это позволяет пользователю выбрать наиболее удобный для него пункт получения заказа.

Данный фрагмент кода представляет собой основную логику оформления заказа из корзины. При подтверждении заказа система формирует объект с данными заказа, который включает список выбранных товаров с их количеством и ценой, общую стоимость товаров. После успешного создания заказа корзина очищается, модальное окно закрывается, и пользователь перенаправляется на страницу профиля.

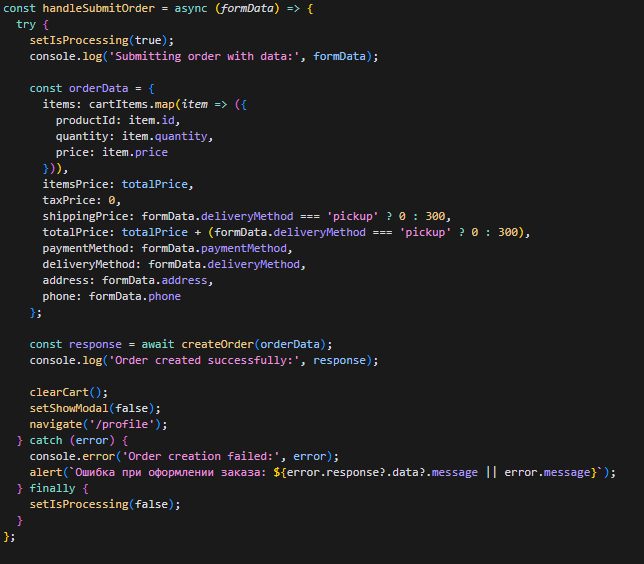


Рисунок 11 - Реализация логики оформления заказа из корзины

Интерфейс корзины и оформления заказа интегрирован с системой управления состоянием, что обеспечивает мгновенное обновление данных при любых изменениях. Система также сохраняет состояние корзины между сессиями, позволяя пользователю продолжить оформление заказа в любое время. Автоматически проверяется корректность введенных данных и доступность выбранных товаров. В процессе оформления заказа пользователь может вернуться к корзине для внесения изменений или продолжить оформление.

Вся информация о заказе сохраняется в базе данных и становится доступной как пользователю, так и администраторам через соответствующие интерфейсы.

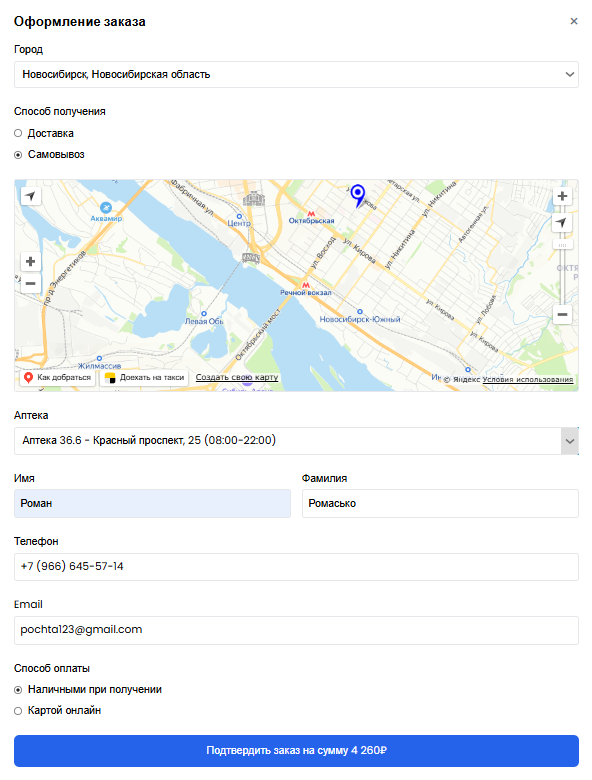


Рисунок 12 - Модальное окно оформления заказа

Процесс оформления заказа построен таким образом, чтобы обеспечить максимальное удобство для пользователя.

### 2.2.4. Административная панель

В административной панели можно добавлять новые товары, редактировать существующие, а также удалять их. При добавлении товара происходит обработка изображения (сохранение на сервере), валидация обязательных полей (название, цена), и запись информации в базу данных. Все эти операции выполняются в рамках одной транзакции, что обеспечивает целостность данных.

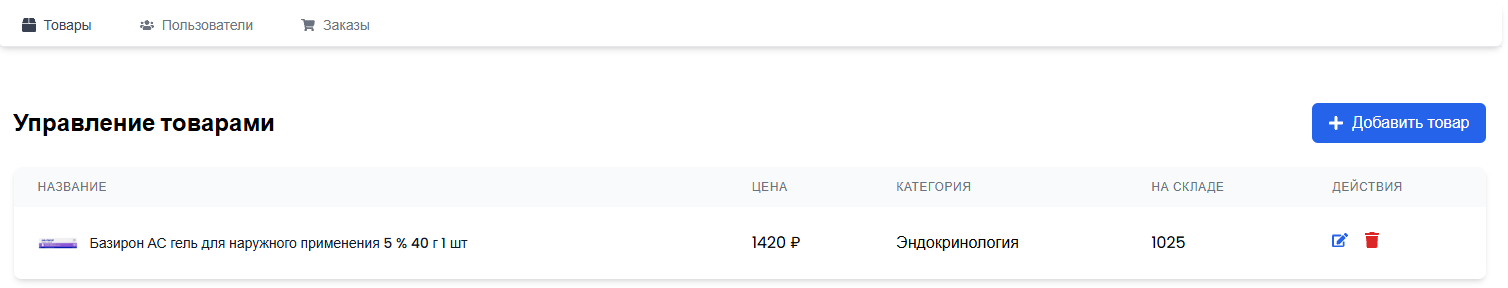


Рисунок 13 – Блок управления товарами

Для редактирования товара реализована загрузка и удаление изображений, а также обновление всех связанных данных.

Управление пользователями осуществляется через специальный интерфейс, который предоставляет доступ к базе данных пользователей. Система позволяет просматривать и редактировать информацию о пользователях.

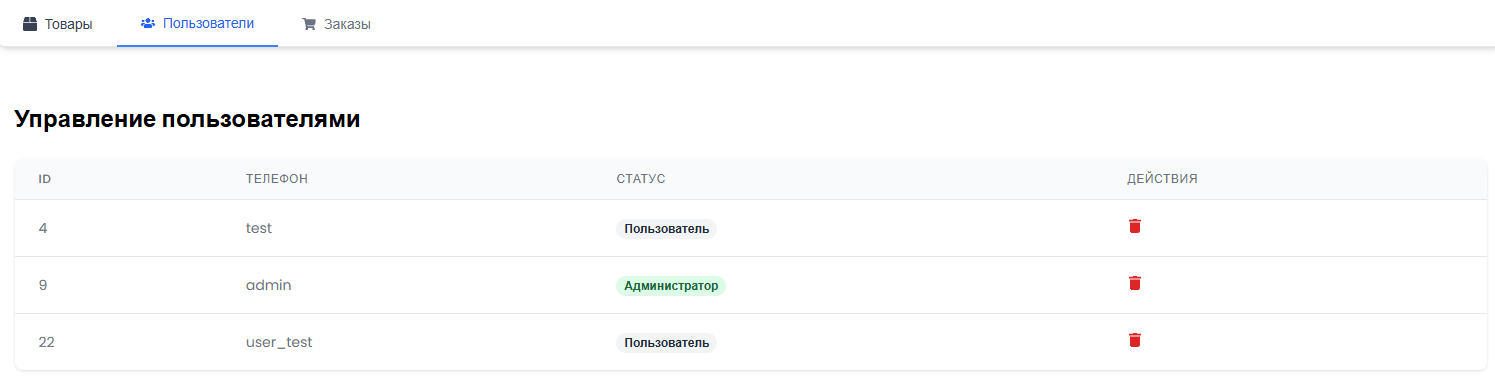


Рисунок 14 – Блок управления пользователями

При необходимости можно удалить пользователя, что приведёт к автоматическому завершению всех его активных сессий.

Администратор может просматривать все заказы пользователей, видеть детали каждого заказа, включая контактные данные пользователя, состав заказа и его стоимость. Для этого используется запрос к базе данных, объединяющий таблицы заказов и пользователей. Также реализована возможность изменять статус заказа (например, "В обработке", "Выполнен", "Отменён") и удалять заказы.

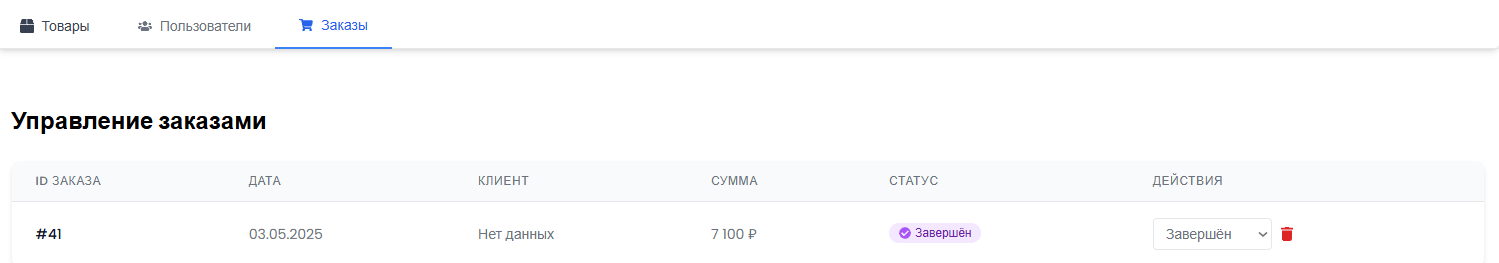


Рисунок 15 – Блок управления заказами

Все изменения статуса и удаления выполняются через соответствующие API-запросы к серверу, которые обновляют записи в базе данных.

Статистика и аналитика в административной панели реализованы с использованием WebSocket-соединений, что обеспечивает обновление данных в реальном времени без необходимости перезагрузки страницы. Система собирает данные о продажах, пользователях, заказов и товаров.

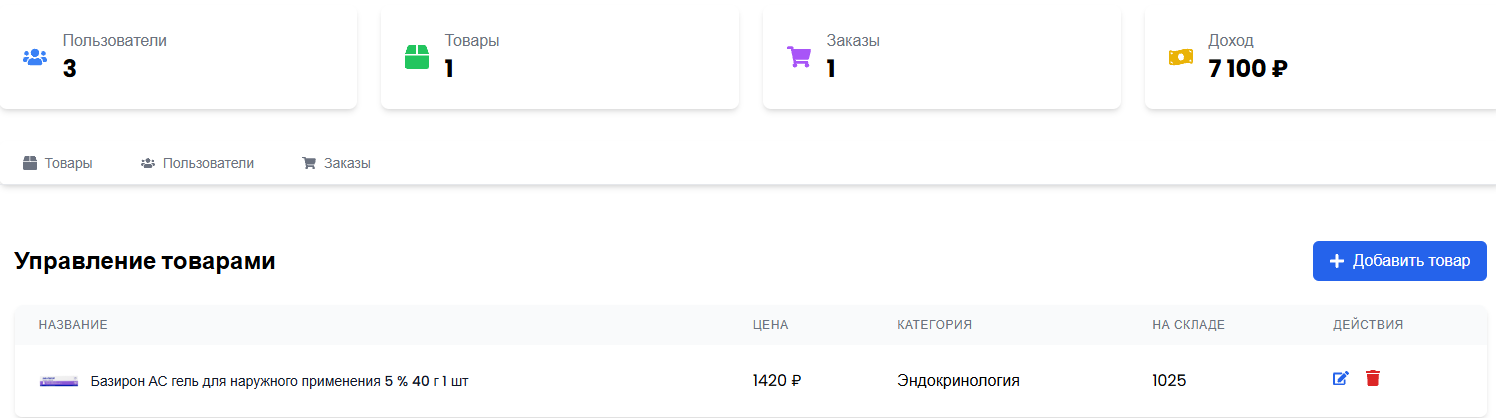


Рисунок 16 – Блок статистики

Для обработки больших объемов данных используются оптимизированные SQL-запросы с агрегацией и группировкой, что позволяет получать актуальную статистику даже при большом количестве транзакций.

Все действия администраторов в системе логируются для обеспечения безопасности. Каждое изменение в базе данных сопровождается записью в лог-файл с указанием времени, пользователя и типа операции. Это позволяет отслеживать все изменения в системе и при необходимости восстанавливать предыдущие состояния данных.

### 2.2.5. Профиль пользователя

Профиль пользователя представляет собой личный кабинет, где отображается история заказов и личная информация. Пользователь может просматривать свои заказы, их статусы и детали. Также в профиле отображается имя пользователя, которое используется при оставлении комментариев к товарам. Это позволяет другим пользователям видеть, кто оставил отзыв, что повышает доверие к отзывам.

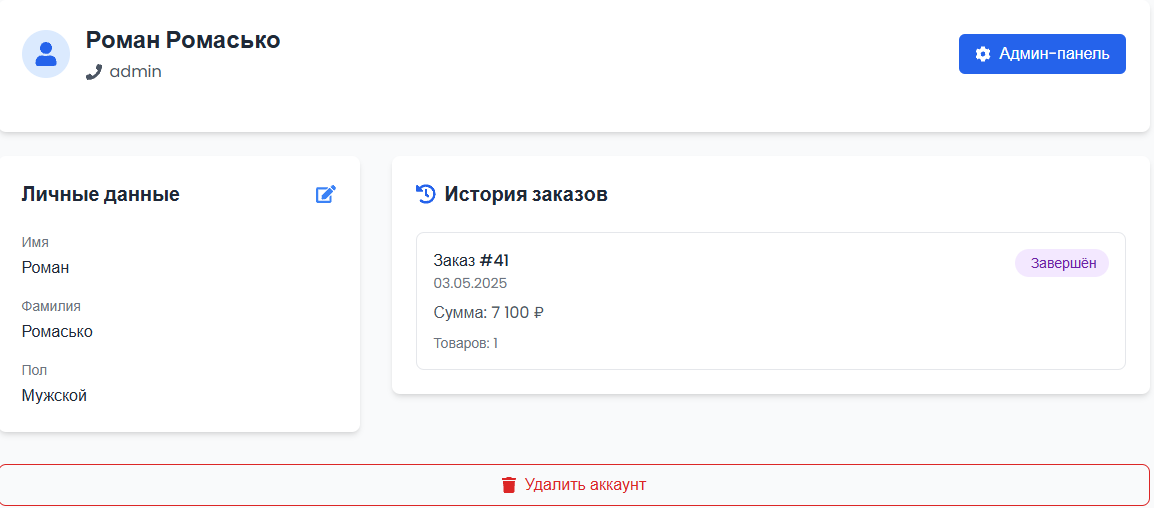


Рисунок 17 - Интерфейс профиля пользователя

Профиль пользователя взаимодействует с бэкендом через API-запросы, которые позволяют получать данные о заказах и личной информации. Запросы к базе данных включают получение истории заказов, их статусов и деталей, а также обновление личной информации пользователя.

### 2.2.6. Отзывы на странице товара

Пользователь может ознакомиться с оценками и комментариями других клиентов, а также оставить собственный отзыв, если авторизован и ранее не писал отзыв на этот товар. Для каждого отзыва отображается имя пользователя, дата публикации, текст комментария и визуальный рейтинг в виде звёзд. Средняя оценка товара рассчитывается автоматически и отображается рядом с количеством отзывов.

Добавление отзыва осуществляется через специальную форму, где пользователь выбирает оценку (от 1 до 5 звёзд) и оставляет текстовый комментарий. После отправки отзыв сразу появляется в общем списке.

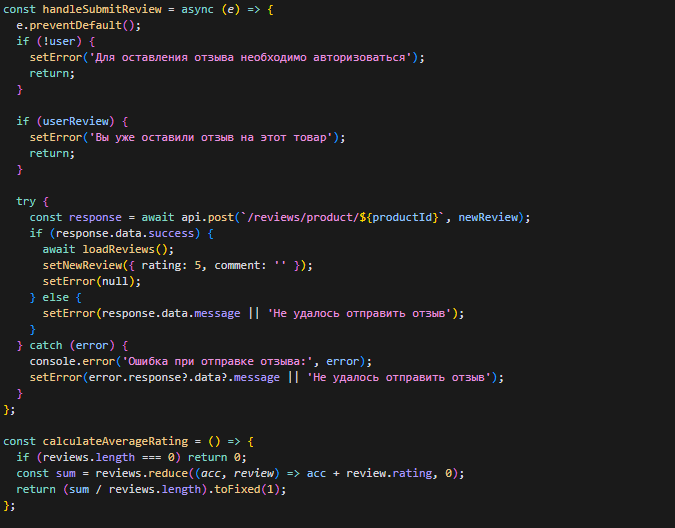


Рисунок 18 - Реализация логики добавления и расчета рейтинга отзывов

Загрузка и обновление отзывов происходит асинхронно через API-запросы к серверу. Все изменения отображаются мгновенно без перезагрузки страницы.

Блок отзывов имеет современный и интуитивно понятный дизайн, выполненный в едином стиле с остальными элементами интерфейса.

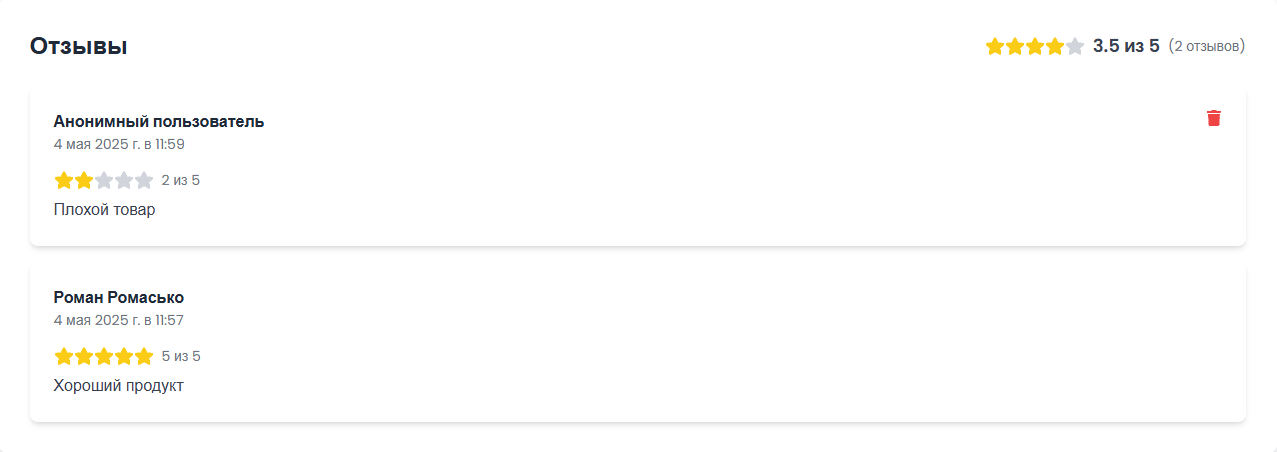


Рисунок 19 - Блок отзывов на странице товара

Если пользователь уже оставил отзыв, повторное добавление невозможно. Администратор и автор отзыва могут удалить свой комментарий.

## 2.3. Разработка базы данных

База данных интернет-аптеки разработана с использованием SQLite, что обеспечивает надежное хранение и быстрый доступ к данным. Выбор SQLite обусловлен его надежностью, простотой использования и отсутствием необходимости в отдельном сервере базы данных. Структура базы данных спроектирована с учетом всех необходимых бизнес-процессов и включает в себя таблицы для хранения информации о товарах, пользователях, заказах, отзывах и элементах корзины.

При проектировании базы данных особое внимание было уделено нормализации данных для устранения избыточности и обеспечения целостности информации. Каждая таблица содержит только необходимые поля, а связи между таблицами реализованы через внешние ключи. Это позволяет эффективно управлять данными и избежать дублирования информации. Например, информация о пользователях хранится в отдельной таблице, а в таблице заказов содержится только ссылка на пользователя.

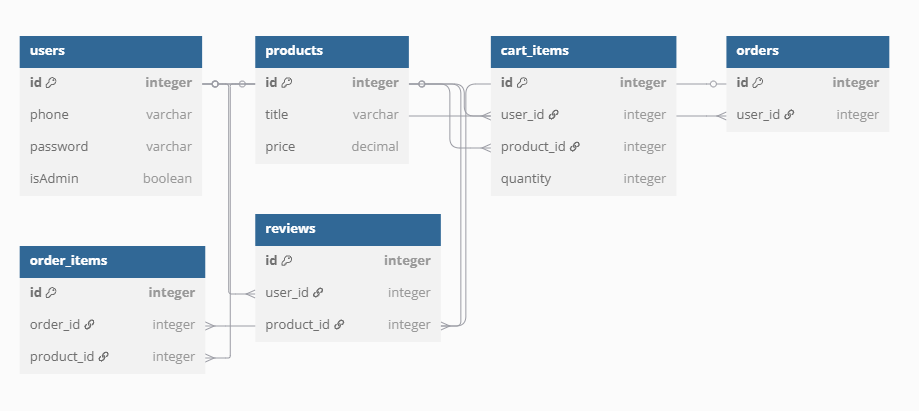


Рисунок 20 - Схема связей между таблицами в базе данных

Основная таблица products содержит полную информацию о товарах, включая их характеристики, цены, наличие на складе и категории.

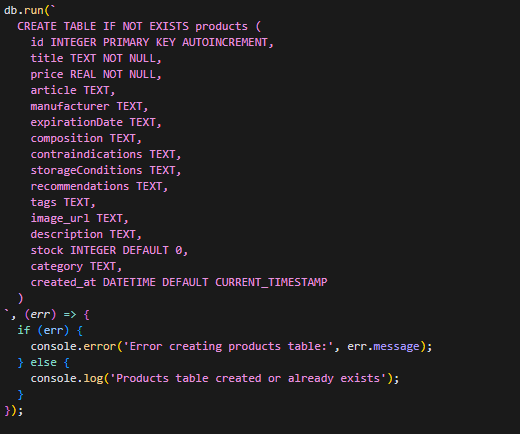


Рисунок 21 - Код инициализации таблицы товаров в базе данных

База данных реализует связи между таблицами через внешние ключи, что обеспечивает целостность данных. Например, таблица orders связана с таблицей users через поле user\_id, а таблица cart\_items связана с таблицами products и users. Это позволяет эффективно отслеживать заказы пользователей и управлять корзиной покупок. Система связей обеспечивает корректное обновление данных при изменении информации о товарах или пользователях.

Для оптимизации производительности в базе данных созданы индексы по часто используемым полям, таким как название товара и идентификатор пользователя. Это значительно ускоряет выполнение запросов поиска и фильтрации, что особенно важно при работе с большим каталогом товаров. Индексация также улучшает производительность при сортировке и группировке данных.

База данных включает механизмы для обеспечения безопасности данных. Все пароли пользователей хранятся в хешированном виде с использованием современного алгоритма шифрования. Доступ к данным осуществляется через подготовленные SQL-запросы, что предотвращает возможность SQL-инъекций и обеспечивает безопасность хранения информации.

Реализованная структура базы данных обеспечивает эффективное хранение и обработку данных, необходимых для работы интернет-аптеки. Система легко масштабируется и может быть дополнена новыми таблицами и связями при необходимости расширения функциональности.

## 2.4. Тестирование и оптимизация

Тестирование и оптимизация являются критически важными этапами разработки любого программного обеспечения, особенно когда речь идет о системах электронной коммерции, где от надежности и производительности напрямую зависит успех бизнеса. В процессе разработки интернет-аптеки особое внимание было уделено комплексному тестированию всех компонентов системы и их оптимизации для обеспечения стабильной и эффективной работы.

Тестирование проводилось на всех уровнях разработки: от проверки отдельных компонентов до комплексного тестирования всей системы. Такой подход позволил выявить и устранить потенциальные проблемы на ранних этапах разработки, что значительно снизило риски возникновения критических ошибок в процессе эксплуатации. Особое внимание было уделено тестированию критически важных бизнес-процессов, таких как аутентификация пользователей, работа с корзиной, оформление заказов и обработка платежей.

В процессе оптимизации основное внимание было сосредоточено на повышении производительности системы и улучшении пользовательского опыта. Были реализованы механизмы кэширования данных, оптимизированы запросы к базе данных, внедрена постраничная загрузка контента. Эти меры позволили значительно ускорить работу системы и снизить нагрузку на серверное оборудование.

Важным аспектом тестирования и оптимизации стало обеспечение безопасности системы. Были проведены проверки на уязвимости, протестированы механизмы защиты от различных типов атак, реализованы дополнительные уровни безопасности для защиты персональных данных пользователей и информации о заказах.

### 2.4.1. Тестирование компонентов

Тестирование компонентов проводилось с использованием встроенных средств разработки React и Node.js. Основное внимание было уделено проверке критически важных компонентов системы, таких как аутентификация, работа с корзиной и оформление заказов.

Для тестирования системы аутентификации были созданы различные сценарии входа в систему. Проверялась корректность работы с некорректными данными, такими как неправильный формат номера телефона или слишком короткий пароль. Система успешно обрабатывала все случаи, выдавая соответствующие сообщения об ошибках. При вводе корректных данных авторизация происходила без задержек, а JWT-токен сохранялся в localStorage.

Тестирование работы с корзиной включало проверку добавления и удаления товаров, изменения их количества и расчета общей стоимости. Особое внимание уделялось синхронизации данных между клиентской и серверной частями. При добавлении товара в корзину система проверяла его наличие на складе и актуальную цену. В случае отсутствия товара или изменения цены пользователь получал соответствующее уведомление.

### 2.4.2. Интеграционное тестирование

Интеграционное тестирование проводилось путем проверки взаимодействия между фронтендом и бэкендом. Основной фокус был направлен на проверку корректности передачи данных и обработки ответов от сервера

При тестировании процесса оформления заказа проверялась последовательность действий: от добавления товаров в корзину до получения подтверждения о создании заказа. Система успешно обрабатывала все этапы, включая проверку наличия товаров, расчет стоимости и сохранение данных в базе. В случае возникновения ошибок (например, при попытке заказать отсутствующий товар) пользователь получал понятное сообщение об ошибке.

Тестирование работы с базой данных включало проверку корректности SQL-запросов и обработки результатов. Особое внимание уделялось проверке целостности данных при выполнении транзакций. Например, при оформлении заказа проверялось, что уменьшение количества товаров на складе и создание записи о заказе происходят в рамках одной транзакции.

### 2.4.3. Отладка и мониторинг

Отладка системы проводилась с использованием встроенных средств разработки и логирования. Для отслеживания ошибок и проблем использовался консольный вывод и базовое логирование в файл.

В процессе отладки были выявлены и исправлены следующие проблемы:

- Некорректная обработка сетевых ошибок при медленном соединении

- Проблемы с отображением данных при большом количестве товаров в корзине

- Ошибки валидации пользовательского ввода в формах

- Проблемы с кэшированием данных на клиентской стороне

Для мониторинга производительности системы использовались встроенные инструменты браузера и Node.js. Это позволило выявить узкие места в работе приложения и оптимизировать их. Например, было обнаружено, что загрузка большого количества товаров при первом открытии каталога вызывает задержки. Для решения этой проблемы была реализована постраничная загрузка данных.

### 2.4.4. Оптимизация производительности

В каталоге товаров реализована постраничная загрузка (пагинация). Вместо загрузки всех товаров сразу клиентское приложение отправляет запросы с параметрами страницы и лимита, получая только необходимое количество товаров для отображения на текущем экране. Это снижает нагрузку на сервер и базу данных, а также ускоряет отображение каталога для пользователя. На сервере реализована обработка параметров пагинации, что позволяет гибко управлять размером выборки и быстро получать нужные данные с помощью SQL-запросов с ограничением по количеству записей.

Для ускорения работы с данными в базе используются агрегирующие и объединяющие SQL-запросы (COUNT, SUM, JOIN), что позволяет получать всю необходимую информацию за один запрос.

При добавлении или редактировании товара изображение проходит проверку на размер и формат, после чего сохраняется в отдельной директории на сервере. Это предотвращает загрузку слишком больших файлов, экономит дисковое пространство и ускоряет работу с изображениями при отображении каталога.

На клиенте используется компонентный подход React: каждый компонент отвечает только за свою часть интерфейса, что позволяет эффективно обновлять только изменившиеся элементы страницы. Для управления состоянием корзины используются отдельные модули (`CartReducer`, `CartActions`, `CartSelector`), что обеспечивает быстрый доступ к данным и минимизирует количество лишних перерисовок интерфейса.

В системе реализована синхронизация данных между клиентом и сервером: при изменении содержимого корзины или оформлении заказа все изменения сразу отправляются на сервер, а клиент получает актуальные данные в ответ. Для мониторинга производительности и выявления узких мест использовались встроенные инструменты браузера и Node.js. В процессе тестирования были обнаружены задержки при загрузке большого количества товаров, после чего была внедрена постраничная загрузка и оптимизированы SQL-запросы.

Таким образом, реализованные меры оптимизации позволяют системе интернет-аптеки работать быстро, стабильно и эффективно даже при росте объёма данных и числа пользователей, обеспечивая высокий уровень пользовательского опыта.

# 3. МАРКЕТИНГОВАЯ ЧАСТЬ

## 3.1. AptekaOnline как информационный продукт: управленческий взгляд

Анализ AptekaOnline как информационного продукта требует комплексного подхода, учитывающего как технические, так и управленческие аспекты. Архитектура платформы отражает современные тенденции в разработке информационных систем и обеспечивает необходимую гибкость для адаптации к изменяющимся требованиям рынка. На схеме представлена структура основных компонентов системы, их взаимосвязи и принципы взаимодействия.

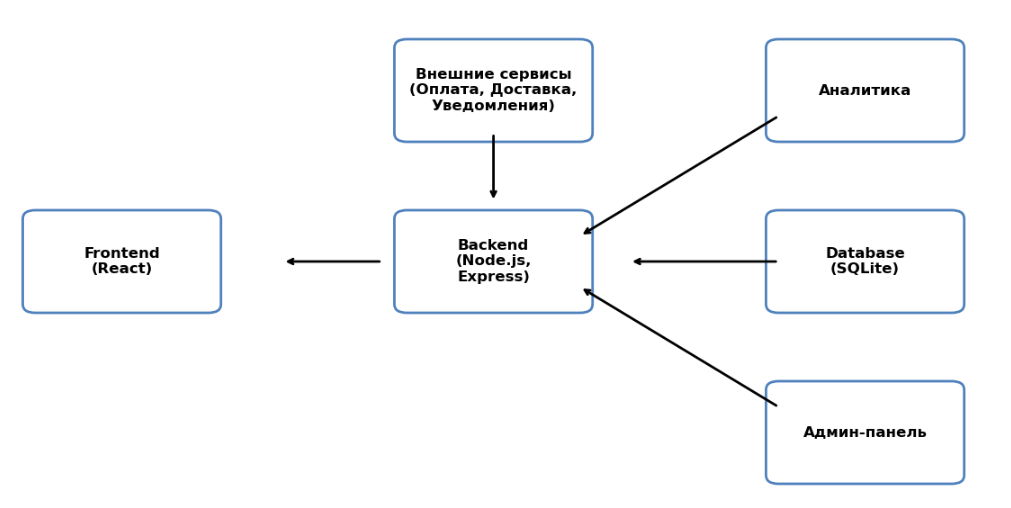


Рисунок 22 – Архитектура информационного продукта AptekaOnline

AptekaOnline — это не просто сайт для онлайн-продажи лекарственных препаратов, а комплексная цифровая экосистема, построенная по принципам современного управления информационными продуктами. С точки зрения менеджмента, AptekaOnline разрабатывается как масштабируемый, гибкий и устойчивый к изменениям рыноковый продукт, способный адаптироваться к новым требованиям законодательства, технологическим трендам и ожиданиям пользователей.

В основе платформы лежит модульная архитектура, что позволяет быстро внедрять новые сервисы, интегрировать сторонние решения и оптимизировать бизнес-процессы. Вся деятельность по развитию AptekaOnline строится на принципах agile-менеджмента: регулярный сбор обратной связи, быстрая реакция на изменения рынка, итеративное внедрение новых функций и постоянное совершенствование пользовательского опыта. AptekaOnline позиционируется как долгосрочный проект с возможностью масштабирования на новые регионы, расширения ассортимента и интеграции с внешними сервисами, что обеспечивает устойчивое развитие и конкурентоспособность на рынке.

## 3.2. Анализ рынка и определение целевой аудитории



Рисунок 23 – Доля крупнейших игроков на рынке интернет-аптек России в 2023 году

Рынок интернет-аптек в России в последние годы демонстрирует устойчивую динамику роста, что подтверждается данными аналитических агентств и профильных изданий. По информации Data Insight, объем рынка онлайн-продаж лекарственных средств в 2023 году превысил 120 млрд рублей, а темпы роста по сравнению с предыдущим годом составили более 30%. Такой рост обусловлен несколькими ключевыми факторами: во-первых, продолжается цифровизация сферы здравоохранения, во-вторых, изменяются потребительские привычки, и все больше людей предпочитают заказывать лекарства онлайн, а в-третьих, государственное регулирование становится более гибким, что позволяет легализовать дистанционную продажу рецептурных и безрецептурных препаратов.

Среди крупнейших игроков рынка выделяются такие платформы, как «Аптека.ру», «Здравсити», «Еаптека», а также ряд региональных сервисов, которые активно развиваются в своих нишах. «Аптека.ру» занимает лидирующие позиции по объему заказов и широте ассортимента, предлагая пользователям более 30 тысяч наименований товаров и развитую сеть пунктов самовывоза по всей стране. «Здравсити» делает акцент на быстрой доставке и интеграции с крупными аптечными сетями, что позволяет обеспечивать высокую скорость выполнения заказов и широкий охват аудитории. «Еаптека» выделяется развитой программой лояльности, наличием мобильного приложения и возможностью получения консультаций фармацевта онлайн. Конкуренция на рынке крайне высока, поскольку большинство крупных платформ инвестируют значительные средства в развитие IT-инфраструктуры, маркетинг и расширение партнерской сети.

Фактические преимущества лидирующих платформ заключаются, прежде всего, в масштабируемости бизнеса, наличии широкой партнерской сети, интеграции с логистическими и платежными сервисами, а также в высоком уровне автоматизации процессов. Например, «Аптека.ру» и «Здравсити» обеспечивают доставку в течение нескольких часов в крупных городах, что становится важным конкурентным преимуществом для пользователей, которым необходимы лекарства в кратчайшие сроки. Кроме того, крупные игроки активно внедряют персонализированные сервисы, такие как рекомендации на основе истории покупок, push-уведомления о скидках и акциях, а также интеграцию с электронными рецептами, что значительно повышает удобство для конечного пользователя.

Важным трендом становится развитие мобильных приложений, которые позволяют пользователям быстро оформлять заказы, отслеживать статус доставки и получать консультации специалистов. По данным исследования Mediascope, доля заказов через мобильные устройства в 2023 году превысила 60%, что свидетельствует о смещении пользовательских предпочтений в сторону мобильных решений. В условиях высокой конкуренции ключевыми факторами успеха становятся скорость и удобство доставки, широта ассортимента, прозрачность цен, а также наличие дополнительных сервисов, таких как консультации фармацевта и поддержка пользователей. В результате рынок интернет-аптек продолжает активно развиваться, а требования к качеству сервиса и технологическим возможностям платформ постоянно растут.

Целевая аудитория AptekaOnline формируется из взрослых пользователей в возрасте от 25 до 65 лет, проживающих преимущественно в городах с развитой инфраструктурой и имеющих опыт онлайн-покупок. Особое внимание уделяется семьям с детьми, пожилым людям, а также лицам с хроническими заболеваниями, которым важно регулярно приобретать лекарства и получать консультации специалистов. Для этих групп критически важны удобство поиска, прозрачность цен, возможность быстрой доставки и гарантия подлинности препаратов. В рамках стратегии позиционирования AptekaOnline делает акцент на безопасности, легальности, прозрачности всех операций и высоком уровне клиентского сервиса, что позволяет формировать лояльную аудиторию и успешно конкурировать с крупными игроками рынка. Важной частью маркетинговой стратегии становится работа с обратной связью, персонализация предложений и развитие программ лояльности, что способствует удержанию клиентов и росту повторных покупок.

## 3.3. Жизненный цикл информационного продукта AptekaOnline

Анализ жизненного цикла информационного продукта AptekaOnline позволяет проследить его развитие от концепции до текущего состояния и определить перспективы дальнейшего роста. На каждом этапе жизненного цикла применяются специфические подходы к управлению и развитию продукта, учитывающие особенности фармацевтического рынка и требования пользователей. Дорожная карта развития AptekaOnline отражает ключевые этапы и направления развития платформы, включая технические улучшения, расширение функциональности и оптимизацию бизнес-процессов.



Рисунок 24 – Дорожная карта развития AptekaOnline

Особое внимание уделяется интеграции с внешними сервисами, развитию мобильных приложений и внедрению аналитических инструментов для повышения эффективности работы платформы.

### 3.3.1. Этап зарождения идеи и концептуализации

*Начальный этап жизненного цикла* информационного продукта AptekaOnline характеризовался формированием концепции и определением ключевых параметров проекта. В ходе анализа рынка интернет-аптек была выявлена потребность в современном решении, объединяющем удобство онлайн-покупок с профессиональной консультационной поддержкой. На данном этапе проводился глубокий анализ потребностей целевой аудитории, исследование конкурентного ландшафта, формирование базовой концепции продукта и определение ключевых функциональных требований. В результате проведенного анализа сформировалась концепция информационного продукта, сочетающего функции электронной коммерции с элементами информационно-консультационной системы. Особое внимание было уделено созданию интуитивно понятного интерфейса и обеспечению безопасности персональных данных пользователей.

### 3.3.2. Этап проектирования и разработки

*Проектирование информационного продукта* AptekaOnline осуществлялось с учетом современных тенденций веб-разработки и требований к пользовательскому опыту. Архитектура системы была спроектирована с использованием микросервисного подхода, что обеспечило масштабируемость и гибкость решения. В процессе проектирования проводилась разработка архитектуры системы, проектирование базы данных, создание прототипов пользовательского интерфейса и формирование технического задания. Особое внимание было уделено обеспечению безопасности данных, оптимизации производительности и созданию удобного пользовательского интерфейса. Были определены основные модули системы и их взаимодействие, что позволило создать надежную и эффективную платформу.

### 3.3.3. Этап запуска и внедрения

*Запуск информационного продукта* AptekaOnline осуществлялся поэтапно, с предварительным тестированием всех компонентов системы. Особое внимание было уделено обеспечению стабильности работы и безопасности данных. В процессе запуска проводилось функциональное и нагрузочное тестирование, проверка безопасности и развертывание системы. В процессе внедрения были реализованы механизмы мониторинга и логирования, обеспечивающие контроль над работой системы и возможность быстрого реагирования на возникающие проблемы. Это позволило обеспечить стабильную работу платформы с момента запуска.

### 3.3.4. Этап роста и развития

При успешном запуске информационный продукт AptekaOnline вступит в фазу активного роста. При масштабировании проекта ожидается увеличение пользовательской базы и объема продаж. При достижении целевой аудитории прогнозируется рост количества зарегистрированных пользователей, увеличение объема продаж, расширение ассортимента и улучшение пользовательских метрик. В данный период планируется постоянное совершенствование функциональности, оптимизация производительности и расширение возможностей системы. Платформа будет развиваться, адаптируясь к потребностям пользователей и рыночным трендам.

### 3.3.5. Этап зрелости и оптимизации

При достижении стабильной пользовательской базы и необходимых показателей эффективности информационный продукт AptekaOnline сможет вступить в фазу зрелости, которая будет характеризоваться стабильной работой и высоким уровнем удовлетворенности пользователей. Основными направлениями оптимизации станут улучшение производительности, расширение функциональности, оптимизация бизнес-процессов и внедрение новых технологий. При успешном развитии система будет адаптироваться к изменяющимся потребностям рынка и технологическим возможностям. Планируется регулярное проведение работ по улучшению пользовательского опыта и оптимизации внутренних процессов.

### 3.3.6. Перспективы развития и трансформации

При успешной реализации всех предыдущих этапов информационный продукт AptekaOnline получит потенциал для дальнейшего развития и трансформации. В планах развития предусмотрено внедрение новых технологий и расширение функциональных возможностей. Среди перспективных направлений развития могут быть реализованы внедрение искусственного интеллекта, расширение мобильных возможностей, интеграция с новыми сервисами и развитие аналитических инструментов. При достижении необходимого уровня развития система будет готова к адаптации к новым технологическим трендам и изменениям в потребностях пользователей. Это создаст предпосылки для долгосрочной конкурентоспособности и устойчивого развития платформы.

AptekaOnline демонстрирует успешную реализацию ключевых этапов жизненного цикла информационного продукта. Для дальнейшего роста и выхода на этап зрелости рекомендуется продолжать расширять ассортимент товаров и внедрять новые сервисы, такие как рекомендации и персонализированные предложения, активно собирать и анализировать обратную связь пользователей для улучшения интерфейса и функционала, усиливать маркетинговую активность, развивать партнерские программы с аптеками и поставщиками, следить за изменениями в законодательстве и своевременно адаптировать платформу, внедрять современные методы аналитики и оптимизации бизнес-процессов, а также развивать мобильную версию и интеграцию с внешними сервисами, такими как доставка и оплата.