Geekbrains

Разработка интернет-магазина растений с использованием современных веб-технологий

Программа: Разработчик

Специализация: Fullstack разработчик

ФИО: Денисенков Антон Анатольевич

Уфа
2025

Содержание

1. Введение

2. Глава 1. Теоретические основы

2.1. Особенности ниши растений

2.2. Стек технологий

3. Глава 2. Проектирование и разработка интернет-магазина растений

3.1. Проектирование архитектуры и базы данных

3.2. Реализация бэкенда на Express.js и MongoDB

3.3. Разработка фронтенда на React.js с использованием Material-UI

4. Заключение

5. Список используемой литературы

6. Приложения

1. Введение

Актуальность темы: Рынок онлайн-продаж растений активно развивается, но многие существующие платформы не учитывают специфику товара.

Цель работы: Разработка интернет-магазина растений с удобным интерфейсом, интеграцией платежей и адаптацией под требования ниши.

Задачи:

1. Выбрать и обосновать стек технологий.

2. Реализовать полный цикл разработки: от проектирования до деплоя.

Стек технологий:

- **Frontend:** React.js, Redux Toolkit, Material-UI, Axios, React Router.
- **Backend:** Node.js, Express.js, MongoDB, Mongoose.
- **Инфраструктура:** Docker, Nginx.
- **Дополнительно:** JWT для аутентификации.
- **Практическая значимость:** Магазин может быть использован малыми бизнесами для продажи растений с учетом их специфики.

2. Глава 1. Теоретические основы

2.1. Особенности ниши растений

- **Требования к хранению и доставке живых растений:**
  - **Соблюдение оптимальных условий хранения:** Растения, в отличие от обычных товаров, требуют особых условий хранения, чтобы сохранить свою жизнеспособность и привлекательный внешний вид. Это включает в себя:
    - **Контроль температуры:** Разные виды растений требуют разной температуры хранения. Некоторые растения чувствительны к заморозкам, другие - к высоким температурам. Неправильная температура может привести к повреждению или гибели растения.
    - **Оптимальная влажность:** Влажность воздуха также играет важную роль в хранении растений. Слишком низкая влажность может привести к обезвоживанию, а слишком высокая - к развитию грибковых заболеваний.
    - **Достаточное освещение:** Растениям необходимо освещение для фотосинтеза. Недостаток света может привести к замедлению роста и ухудшению внешнего вида. Использование искусственного освещения может быть необходимо в помещениях с недостаточным естественным светом.
    - **Правильный полив:** Растения нуждаются в регулярном поливе, но важно не переусердствовать. Перелив может привести к загниванию корней, а недостаток влаги - к обезвоживанию.
    - **Обеспечение безопасной и бережной доставки:** Доставка живых растений требует особого внимания и аккуратности. **Важно:**
      - **Надежная упаковка:** Растения должны быть надежно упакованы, чтобы предотвратить повреждения при

транспортировке. Упаковка должна защищать растения от механических повреждений, перепадов температуры и влажности.

- **Быстрая доставка:** Время доставки должно быть минимальным, чтобы растения не страдали от длительного пребывания в закрытой упаковке. Использование специализированных служб доставки, специализирующихся на доставке растений, может быть оправдано.
- **Защита от экстремальных температур:** В зависимости от сезона необходимо обеспечить защиту растений от экстремальных температур - переохлаждения зимой и перегрева летом. Использование термоупаковки или доставки в специальных условиях может быть необходимо.
- **Правильное обращение при погрузке и разгрузке:** Важно обучить персонал службы доставки правильному обращению с растениями, чтобы избежать их повреждения при погрузке и разгрузке.
- **Предоставление инструкций по распаковке и уходу после доставки:** Вместе с растением необходимо предоставить клиенту подробные инструкции по распаковке и уходу, чтобы он мог обеспечить растению оптимальные условия после доставки.

- **Необходимость детального описания товаров** (освещение, полив, температурный режим):
  - **Предоставление полной и точной информации:** Для успешной продажи растений необходимо предоставить покупателям полную и точную информацию о каждом виде растения, чтобы они могли сделать осознанный выбор и обеспечить растению оптимальные условия ухода.

Это включает в себя:

- **Подробное описание внешнего вида:** Необходимо предоставить фотографии растения с разных ракурсов, а также подробное описание его размеров, формы листьев, цветов и других характеристик.
- **Требования к освещению:** Указать, какое освещение необходимо растению - яркое, рассеянное или тень.
- **Рекомендации по поливу:** Указать, как часто и в каком количестве поливать растение.
- **Требования к температурному режиму:** Указать оптимальную температуру для выращивания растения.
- **Требования к влажности воздуха:** Указать, какая влажность воздуха необходима растению.
- **Потребности в удобрениях:** Указать, какие удобрения и как часто необходимо использовать для подкормки растения.
- **Рекомендации по пересадке:** Указать, когда и как пересаживать растение.
- **Информация о болезнях и вредителях:** Предоставить информацию о распространенных болезнях и вредителях, поражающих данный вид растения, а также о способах их профилактики и лечения.
- **Размер растения:** Обязательно указывать размер растения в горшке и его примерную высоту.

- **Создание интерактивных инструментов:** Для облегчения выбора растения можно создать интерактивные инструменты, которые помогут покупателям подобрать растения, соответствующие их условиям и потребностям, например:
  - **Фильтр по освещению, поливу и температуре:** Позволяет покупателям отфильтровать растения по требуемым условиям ухода.

- **Тест "Какое растение вам подходит?":** Помогает покупателям определить, какие растения лучше всего подходят для их дома и образа жизни
- **Калькулятор полива:** Помогает покупателям определить, как часто поливать растение в зависимости от времени года, температуры и влажности воздуха.

- **Сезонность и ограниченный срок реализации некоторых видов:**
  - **Учет сезонности:** Спрос на определенные виды растений может зависеть от времени года. Необходимо учитывать сезонность при планировании ассортимента и маркетинговых акций.
  - **Ограниченный срок годности:** Некоторые виды растений, особенно цветущие, имеют ограниченный срок реализации. Необходимо тщательно контролировать сроки годности и избегать продажи увядших растений.
  - **Предзаказы:** Для сезонных растений можно использовать систему предзаказов, чтобы гарантировать наличие растений и избежать их увядания до момента продажи.

2.2 Стек технологий

Фронтенд:

1. React.js

- **Причины выбора:**
  - **Компонентная архитектура** для создания переиспользуемых UI-элементов (например, карточки растений, фильтры): React.js основан на компонентной архитектуре, что позволяет разбивать интерфейс на независимые, переиспользуемые компоненты. Это значительно упрощает разработку, поддержку и масштабирование сложных приложений. Компоненты могут быть небольшими, отвечая за отображение одной кнопки, или сложными, представляя собой целую секцию сайта. Переиспользование компонентов сокращает дублирование кода и повышает консистентность интерфейса.Виртуальный DOM для оптимизации производительности при частых обновлениях интерфейса (динамическая корзина, фильтрация товаров).
  - **Виртуальный DOM** для оптимизации производительности при частых обновлениях интерфейса (динамическая корзина, фильтрация товаров): React использует виртуальный DOM, который представляет собой легковесную копию реального DOM. При изменении данных React сравнивает виртуальный DOM с предыдущим состоянием и обновляет только те части реального DOM, которые действительно изменились. Это значительно повышает производительность, особенно при частых обновлениях интерфейса, таких как обновление содержимого корзины или фильтрация товаров.
  - **Декларативный подход:** React использует декларативный подход к разработке, что означает, что вы описываете, как должен выглядеть интерфейс в зависимости от состояния данных, а не указываете последовательность действий для его изменения. Это упрощает

понимание и отладку кода, а также позволяет React оптимизировать процесс обновления интерфейса.

- **JSX:** Использование JSX позволяет писать HTML-подобный код непосредственно в JavaScript, что делает код более читабельным и удобным для работы.
- **Большое и активное сообщество:** Огромное сообщество React предоставляет множество библиотек, инструментов и ресурсов, которые помогают решать практически любые задачи разработки. Это позволяет быстро находить готовые решения, избегать изобретения велосипеда и получать поддержку при возникновении проблем.

- **Инструменты:**
  - **Redux Toolkit**: Управление состоянием приложения (корзина, данные пользователя, фильтры): Redux Toolkit - это набор инструментов, упрощающих использование Redux для управления состоянием приложения. Он предоставляет API для создания хранилищ, редьюсеров и экшенов, а также интегрируется с React DevTools для отладки. Redux идеально подходит для управления сложным состоянием, которое необходимо совместно использовать между различными компонентами приложения. В данном случае, он используется для управления состоянием корзины, данными пользователя и фильтрами товаров.
  - **Material-UI**: Готовые компоненты для быстрого создания адаптивного дизайна (кнопки, формы, модальные окна): Material-UI - это библиотека готовых UI-компонентов, соответствующих принципам Material Design от Google. Она предоставляет широкий спектр компонентов, таких как кнопки, формы, модальные окна, таблицы и т.д., которые можно легко настроить и использовать для быстрого создания адаптивного и современного дизайна. Использование Material-UI позволяет сократить

время разработки и обеспечить консистентный внешний вид приложения.

- **Axios**: Отправка HTTP-запросов к бэкенду (получение каталога растений, оформление заказов): Axios - это HTTP-клиент на основе Promise, который позволяет отправлять запросы к бэкенду и получать ответы. Он предоставляет удобный API для настройки запросов, обработки ошибок и работы с данными. Axios используется для получения каталога растений, оформления заказов, аутентификации пользователей и других операций, требующих взаимодействия с сервером.
- **React Router**: Навигация между страницами (главная, каталог, корзина, личный кабинет): React Router - это библиотека для управления маршрутизацией в React-приложениях. Она позволяет создавать Single Page Applications (SPA) с навигацией между различными страницами без перезагрузки страницы. React Router используется для навигации между главной страницей, каталогом, корзиной, личным кабинетом и другими страницами приложения.

Бэкенд:

1. Node.js + Express.js

- **Причины выбора:**
  - **Быстрое создание RESTful API** с поддержкой асинхронных операций: Node.js, благодаря своей неблокирующей архитектуре и событийной модели, идеально подходит для создания высокопроизводительных API, способных обрабатывать большое количество одновременных запросов без задержек. Express.js упрощает процесс создания API, предоставляя набор готовых инструментов и middlewares для маршрутизации, обработки запросов и ответов.
  - **Легкая интеграция с MongoDB** через Mongoose: Mongoose предоставляет элегантный и простой в использовании интерфейс для взаимодействия с MongoDB. Он позволяет определять схемы данных, выполнять валидацию, а также предоставляет удобные методы для выполнения CRUD-операций. Это значительно упрощает работу с базой данных и сокращает время разработки.
  - **JavaScript на бэкенде и фронтенде** позволяет команде использовать единый язык программирования, упрощая разработку и поддержку: Использование JavaScript на обеих сторонах приложения позволяет команде разработчиков использовать одни и те же знания и навыки, что повышает эффективность и сокращает затраты на обучение. Также упрощается обмен кодом и логикой между фронтендом и бэкендом.
  - **Большое и активное сообщество**, что обеспечивает доступ к множеству библиотек и ресурсов: Огромное сообщество Node.js предлагает широкий спектр библиотек и ресурсов, которые помогают решать практически любые задачи разработки. Это позволяет быстро находить готовые решения, избегать изобретения велосипеда и получать поддержку при возникновении проблем. NPM (Node Package

Manager) значительно упрощает управление зависимостями и установку библиотек.

- **Высокая производительность и масштабируемость** благодаря неблокирующей, событийной модели: Неблокирующая архитектура Node.js позволяет ему эффективно использовать ресурсы сервера и обрабатывать большое количество одновременных соединений. Это делает его идеальным выбором для приложений с высокой нагрузкой и необходимостью масштабирования. Возможность кластеризации Node.js приложений позволяет распределять нагрузку между несколькими ядрами процессора, еще больше повышая производительность.
- **Middleware-ориентированная архитектура Express.js:** Express.js позволяет легко добавлять и использовать middleware для обработки запросов и ответов. Это позволяет создавать модульную и расширяемую архитектуру, в которой каждый middleware выполняет определенную функцию, такую как аутентификация, логирование, обработка ошибок и т.д.

- **Ключевые модули:**
  - **JWT:** аутентификация пользователей (токены для защиты роутов заказов): JWT (JSON Web Tokens) используются для безопасной аутентификации пользователей и авторизации доступа к защищенным ресурсам. Токен содержит информацию о пользователе и его правах доступа, и подписывается сервером, что гарантирует его целостность и подлинность.
  - **Bcryptjs**: Хеширование паролей: Bcryptjs - это библиотека для надежного хеширования паролей. Она использует алгоритм bcrypt, который добавляет соль (случайную строку) к паролю перед

хешированием, и позволяет настраивать вычислительную сложность хеширования, чтобы затруднить взлом паролей методом перебора.

- **Настройка доступа к API для фронтенда:** CORS (Cross-Origin Resource Sharing) - это механизм, который позволяет браузеру делать запросы к ресурсам, находящимся на другом домене. Настройка CORS необходима для того, чтобы фронтенд, размещенный на одном домене, мог обращаться к API, размещенному на другом домене.

2. MongoDB

- **Причины выбора:**
  - **Гибкая структура:** Данные хранятся как JSON-подобные документы, что удобно, если структура данных часто меняется. Не нужно заранее определять строгую схему, как в традиционных базах данных.
  - **Легко масштабируется:** Можно легко увеличить объем хранилища и скорость обработки данных, добавляя новые серверы (горизонтальное масштабирование).
  - **Простота разработки с Node.js:** Хорошо работает с Node.js и библиотекой Mongoose. Mongoose упрощает взаимодействие с базой данных, позволяя меньше думать о самой базе и больше о логике приложения.
  - **Быстрая работа:** Обеспечивает высокую скорость записи и чтения данных благодаря использованию индексов.
  - **Изменение структуры без проблем:** Можно легко добавлять новые поля или менять типы данных без сложных процедур обновления (миграций схемы).

- **Основные возможности (через Mongoose):**
  - **Схемы:** Mongoose позволяет задавать структуру данных (схему) для каждой коллекции, включая типы данных и правила проверки (валидации).
  - **Индексы**: Mongoose позволяет создавать индексы для ускорения поиска данных по определенным полям.
  - **Агрегация:** Mongoose позволяет выполнять сложные запросы для обработки и анализа данных с помощью агрегационных пайплайнов.
  - **Модели:** Mongoose создает модели, которые представляют собой объекты для работы с данными в MongoDB. Через модели можно добавлять, изменять, удалять и искать данные.
  - **Безопасность данных:** MongoDB поддерживает атомарные операции, что позволяет гарантировать целостность данных, даже если несколько пользователей одновременно пытаются их изменить.
  - **Надежность:** MongoDB поддерживает репликацию, что позволяет создавать копии данных на разных серверах. Это обеспечивает отказоустойчивость и высокую доступность данных, даже если один из серверов выйдет из строя.

3. Глава 2. Проектирование архитектуры и базы данных

3.1. Проектирование архитектуры и базы данных

В данном разделе дипломной работы рассматривается процесс проектирования архитектуры интернет-магазина растений и структуры его базы данных. Ключевой целью этапа проектирования является создание надежной, масштабируемой и удобной в поддержке системы, способной эффективно обрабатывать данные о растениях, пользователях, заказах и других связанных сущностях.

Архитектура приложения

Архитектура приложения будет построена на основе трехзвенной модели:

- **Фронтенд:** Пользовательский интерфейс, реализованный на React.js, отвечающий за отображение данных, взаимодействие с пользователем и отправку запросов к бэкенду.
- **Бэкенд:** Серверная часть, реализованная на Node.js и Express.js, отвечающая за обработку запросов от фронтенда, взаимодействие с базой данных, аутентификацию и авторизацию пользователей, а также реализацию бизнес-логики приложения.
- **База данных:** MongoDB, NoSQL база данных, выбранная для хранения данных о растениях, пользователях, заказах и других сущностях.

Данный выбор архитектуры обусловлен гибкостью, масштабируемостью и простотой интеграции компонентов, что позволяет быстро разрабатывать и развертывать приложение.

Проектирование базы данных

Проектирование базы данных является критически важным этапом, определяющим эффективность хранения и обработки данных. В рамках данного этапа будут определены следующие аспекты:

- **Определение сущностей:** Определение основных сущностей, которые будут храниться в базе данных, таких как Растения, Пользователи, Заказы, Категории растений, Отзывы, и т.д.
- **Определение атрибутов сущностей:** Определение атрибутов для каждой сущности, например, для сущности "Растение" будут определены атрибуты "Название", "Описание", "Цена", "Изображения", "Вес", "Артикул", "Категория" и т.д.
- **Определение связей между сущностями:** Определение связей между сущностями, например, "Пользователь" может иметь несколько "Заказов", "Растение" может принадлежать к одной или нескольким "Категориям", и т.д.
- **Разработка схем данных:** Определение схем данных для каждой коллекции MongoDB с использованием Mongoose schemas, включая определение типов данных, правил валидации и индексов.

При проектировании базы данных будет учитываться гибкость схемы MongoDB, позволяющая легко адаптировать структуру данных к изменяющимся требованиям. Особое внимание будет уделено индексации данных для обеспечения высокой производительности запросов, особенно при поиске и фильтрации растений.

Ожидаемые результаты:

Результатом данного этапа является разработанная архитектура приложения, обеспечивающая модульность, масштабируемость и удобство поддержки. Также будет разработана логическая модель базы данных, описывающая структуру хранения данных и связи между сущностями, а также определены схемы данных MongoDB с использованием Mongoose schemas.

Этот раздел закладывает основу для последующих этапов разработки, обеспечивая четкое понимание структуры приложения и организации данных.

3.2. Реализация бэкенда на Express.js и MongoDB

В данном разделе описывается процесс реализации серверной части (бэкенда) интернет-магазина растений с использованием Node.js, Express.js и MongoDB. Бэкенд отвечает за обработку запросов от фронтенда, управление данными, аутентификацию и авторизацию пользователей, а также реализацию бизнес-логики приложения.

Основные задачи бэкенда:

- **Предоставление RESTful API:** Создание RESTful API для взаимодействия фронтенда с бэкендом. API предоставляет endpoints для выполнения различных операций, таких как получение списка растений, добавление растения в корзину, оформление заказа, регистрация и аутентификация пользователей.
- **Обработка запросов от фронтенда:** Получение и обработка HTTP-запросов от фронтенда, выполнение необходимых операций (например, чтение данных из базы данных, обновление данных, выполнение бизнес-логики) и отправка ответов обратно на фронтенд.
- **Управление данными:** Взаимодействие с базой данных MongoDB для хранения и извлечения данных о растениях, пользователях, заказах и других сущностях. Использование Mongoose для работы с MongoDB, что обеспечивает удобный и типизированный доступ к данным.
- **Аутентификация и авторизация пользователей:** Реализация механизмов аутентификации и авторизации для защиты доступа к определенным ресурсам и функциям приложения. Использование JWT (JSON Web Tokens) для безопасной аутентификации пользователей.
- **Реализация бизнес-логики:** Реализация бизнес-логики приложения, такой как расчет стоимости заказа, обработка платежей, управление запасами растений и т.д.

Используемые технологии и библиотеки:

- **Node.js:** JavaScript-среда выполнения на сервере, обеспечивающая высокую производительность и масштабируемость.
- **Express.js:** Фреймворк для Node.js, упрощающий создание RESTful API и обработку HTTP-запросов.
- **MongoDB:** NoSQL база данных, используемая для хранения данных о растениях, пользователях, заказах и других сущностях.
- **Mongoose:** ODM (Object-Document Mapper) для MongoDB, предоставляющий удобный и типизированный доступ к данным.
- **JWT (JSON Web Tokens)**: Библиотека для создания и проверки JSON Web Tokens, используемых для аутентификации пользователей.
- **Bcryptjs:** Библиотека для хеширования паролей, обеспечивающая надежную защиту пользовательских данных.
- **CORS:** Middleware для настройки доступа к API для фронтенда, обеспечивающий безопасность и контроль доступа к ресурсам.

Реализация основных модулей:

- **Модуль аутентификации и авторизации:** Реализация endpoints для регистрации, аутентификации и авторизации пользователей. Использование JWT для безопасной передачи информации о пользователе между фронтендом и бэкендом.
- **Модуль управления растениями:** Реализация endpoints для получения списка растений, добавления нового растения, обновления информации о растении и удаления растения.
- **Модуль управления заказами:** Реализация endpoints для создания заказа, получения информации о заказе, обновления статуса заказа и отмены заказа.

- **Модуль управления корзиной:** Реализация endpoints для добавления растения в корзину, удаления растения из корзины и получения содержимого корзины.
- **Модуль управления пользователями:** Реализация endpoints для:
    - Получения списка пользователей (для администраторов).
    - Создания нового пользователя .
    - Обновления информации о пользователе (профиль пользователя, для пользователей и администраторов).
    - Удаления пользователя (для администраторов).
    - Изменения пароля пользователя.
    - Управление ролями пользователей (для администраторов).
    - Возможность реализации функциональности сброса пароля.

Ожидаемые результаты:

В результате реализации данного раздела будет создан функциональный и надежный бэкенд для интернет-магазина растений, обеспечивающий:

- Предоставление RESTful API для взаимодействия фронтенда с бэкендом.
- Управление данными о растениях, пользователях и заказах.
- Аутентификацию и авторизацию пользователей.
- Реализацию бизнес-логики приложения.

Реализованный бэкенд будет протестирован на соответствие требованиям и обеспечен необходимой документацией для дальнейшей поддержки и развития.

3.3 Разработка фронтенда на React.js с использованием Material-UI

В данном разделе описывается процесс разработки пользовательского интерфейса (фронтенда) интернет-магазина растений с использованием React.js и ряда сопутствующих библиотек и инструментов. Фронтенд отвечает за отображение данных пользователю, обеспечение интерактивности и отправку запросов к бэкенду.

Основные задачи фронтенда:

- **Отображение каталога растений:** Реализация интерфейса для просмотра и фильтрации каталога растений, включая отображение детальной информации о каждом растении (изображения, описание, характеристики и т.д.).
- **Реализация корзины:** Создание функциональности корзины, позволяющей пользователям добавлять растения в корзину, просматривать содержимое корзины, изменять количество растений и удалять растения из корзины.
- **Оформление заказа:** Реализация процесса оформления заказа, включая ввод контактной информации, выбор способа доставки и оплаты, и подтверждение заказа.
- **Аутентификация и авторизация пользователей:** Предоставление интерфейса для регистрации, аутентификации и авторизации пользователей.
- **Личный кабинет пользователя:** Реализация личного кабинета пользователя, позволяющего просматривать историю заказов, изменять профиль и управлять адресами доставки.
- **Адаптивный дизайн:** Обеспечение адаптивного дизайна, обеспечивающего корректное отображение приложения на различных устройствах (компьютеры, планшеты, смартфоны).

- **Интерактивность и удобство использования:** Создание интуитивно понятного и удобного в использовании интерфейса, обеспечивающего приятный пользовательский опыт.

Используемые технологии и библиотеки:

- **React.js:** JavaScript-библиотека для создания пользовательских интерфейсов, обеспечивающая компонентный подход и высокую производительность.
- **Redux Toolkit:** Набор инструментов, упрощающий управление состоянием приложения, обеспечивающий предсказуемое и централизованное хранение данных.
- **Material-UI:** Библиотека готовых UI-компонентов, соответствующих принципам Material Design, позволяющая быстро создавать адаптивный и современный дизайн.
- **Axios:** HTTP-клиент на основе Promise, используемый для отправки запросов к бэкенду и получения данных.
- **React Router:** Библиотека для управления маршрутизацией в React-приложении, обеспечивающая навигацию между различными страницами и компонентами.

Реализация основных компонентов и страниц:

- **Главная страница:** Отображение баннеров, популярных растений и категорий растений.
- **Страница каталога растений:** Отображение списка растений с возможностью фильтрации и сортировки.

- **Страница детальной информации о растении:** Отображение подробной информации о растении, включая изображения, описание, характеристики и рекомендации по уходу.
- **Страница корзины:** Отображение содержимого корзины с возможностью изменения количества растений и удаления растений из корзины.
- **Страница оформления заказа:** Сбор информации о доставке и оплате, подтверждение заказа.
- **Страница регистрации и авторизации:** Предоставление интерфейса для регистрации и аутентификации пользователей.
- **Личный кабинет пользователя:** Отображение истории заказов, профиля пользователя и адресов доставки.

Ожидаемые результаты:

В результате реализации данного раздела будет создан современный, удобный и адаптивный пользовательский интерфейс для интернет-магазина растений, обеспечивающий:

- Удобный просмотр и фильтрацию каталога растений.
- Функциональность корзины и оформления заказа.
- Аутентификацию и авторизацию пользователей.
- Личный кабинет пользователя.
- Адаптивный дизайн для различных устройств.

Разработанный фронтенд будет интегрирован с бэкендом и протестирован на соответствие требованиям.

4. Заключение

В рамках данной дипломной работы был разработан интернет-магазин растений, представляющий собой комплексное веб-приложение, включающее в себя как клиентскую (фронтенд), так и серверную (бэкенд) части. Проект охватывает все этапы разработки, начиная от проектирования архитектуры и базы данных и заканчивая реализацией пользовательского интерфейса и серверной логики.

В ходе выполнения работы были решены следующие задачи:

- **Спроектирована архитектура приложения:** Определена трехзвенная архитектура, обеспечивающая модульность, масштабируемость и удобство поддержки.
- **Разработана база данных:** Спроектирована логическая модель базы данных MongoDB, описывающая структуру хранения данных и связи между сущностями, а также определены схемы данных с использованием Mongoose schemas.
- **Реализован бэкенд:** Разработан бэкенд на Node.js и Express.js, предоставляющий RESTful API для взаимодействия фронтенда с бэкендом, обеспечивающий управление данными о растениях, пользователях и заказах, аутентификацию и авторизацию пользователей, а также реализацию бизнес-логики приложения.
- **Разработан фронтенд:** Создан современный, удобный и адаптивный пользовательский интерфейс на React.js с использованием Material-UI, обеспечивающий удобный просмотр и фильтрацию каталога растений, функциональность корзины и оформления заказа, аутентификацию и авторизацию пользователей, личный кабинет пользователя и адаптивный дизайн для различных устройств.

В результате выполнения работы был создан полнофункциональный интернет-магазин растений, готовый к развертыванию и использованию. Разработанное приложение демонстрирует эффективное применение современных веб-технологий и подходов к разработке, таких как:

- **RESTful API:** Обеспечивает четкое разделение между фронтендом и бэкендом и упрощает интеграцию с другими системами.
- **Компонентный подход (React.js):** Обеспечивает модульность и переиспользуемость компонентов, что упрощает разработку и поддержку приложения.
- **NoSQL база данных (MongoDB):** Обеспечивает гибкость и масштабируемость хранения данных.
- **Адаптивный дизайн (Material-UI):** Обеспечивает удобство использования приложения на различных устройствах.

Полученные в ходе выполнения дипломной работы навыки и знания могут быть использованы для разработки других веб-приложений, а также для дальнейшего развития и улучшения разработанного интернет-магазина растений.

Возможные направления дальнейшего развития проекта:

- **Интеграция с платежными системами:** Реализация возможности оплаты заказов онлайн.
- **Интеграция со службами доставки:** Автоматизация процесса доставки заказов.
- **Реализация системы отзывов и рейтингов:** Предоставление пользователям возможности оставлять отзывы о растениях и магазинах.

- **Разработка мобильного приложения:** Создание мобильного приложения для iOS и Android.
- **Использование машинного обучения для персонализации рекомендаций:** Предоставление пользователям персональных рекомендаций по растениям на основе их предпочтений и истории заказов.

В заключение, данная дипломная работа позволила углубить знания в области веб-разработки и получить практический опыт создания комплексного веб-приложения. Разработанный интернет-магазин растений может быть использован как основа для реального бизнеса, а также как пример успешного применения современных веб-технологий.

5. Список используемой литературы

В данном разделе представлен список литературы, использованной при подготовке дипломной работы. Он включает в себя документацию и онлайн-ресурсы, которые послужили основой для теоретических исследований и практической реализации проекта.

Официальная документация:

1. Node.js Documentation. https://nodejs.org/dist/latest-v20.x/docs/api/

2. Express.js Documentation. https://expressjs.com/

3. MongoDB Documentation. https://www.mongodb.com/docs/

4. Mongoose Documentation. https://mongoosejs.com/docs/

5. React.js Documentation. https://react.dev/

6. Redux Toolkit Documentation. https://redux-toolkit.js.org/

7. Material-UI Documentation (MUI). https://mui.com/material-ui/

8. Axios Documentation. https://axios-http.com/docs/intro

9. React Router Documentation. https://reactrouter.com/home

Онлайн-ресурсы:

1. MDN Web Docs. https://developer.mozilla.org/ru/ (Обширный ресурс по веб-технологиям).

2. Stack Overflow. https://stackoverflow.com/ (Форум для разработчиков, полезный для решения конкретных проблем).

6. Приложения

1. **репозиторий на GitHub:**
   https://github.com/Qwerander/magazin