Geekbrains

**Разработка веб-приложения для управления гардеробом: 'Мой гардероб'**

Программа: Разработчик

Программист

Муминов Темурбек Комилович

Санкт-Петербург

2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc169135434)

[Глава 1. Основы веб-разработки приложений 6](#_Toc169135435)

[1.1 Что такое веб-приложение и его особенности 6](#_Toc169135436)

[1.2 Архитектура веб-приложений 8](#_Toc169135437)

[1.3 Технологии для разработки веб-приложений 12](#_Toc169135438)

[1.4 Этапы разработки веб-приложений 15](#_Toc169135439)

[Глава 2. Анализ существующих решений и потребностей пользователей 18](#_Toc169135440)

[2.1 Обзор существующих приложений для управления гардеробом 18](#_Toc169135441)

[2.2 Анализ потребностей пользователей 22](#_Toc169135442)

[2.3 Определение ключевых функциональных требований 24](#_Toc169135443)

[Глава 3. Проектирование и разработка веб-приложения 'Мой гардероб' 28](#_Toc169135444)

[3.1 Разработка требований к функционалу 28](#_Toc169135445)

[3.2 Проектирование интерфейса 31](#_Toc169135446)

[3.3 Архитектура приложения 34](#_Toc169135447)

[Глава 4. Реализация функционала приложения 38](#_Toc169135448)

[4.1 Создание и управление шкафами 38](#_Toc169135449)

[4.2 Добавление и организация предметов 40](#_Toc169135450)

[4.3 Система фильтрации и поиска 42](#_Toc169135451)

[4.4 Реализация авторизации и регистрации пользователей 43](#_Toc169135452)

[ЗАКЛЮЧЕНИЕ 47](#_Toc169135453)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 50](#_Toc169135454)

# ВВЕДЕНИЕ

Проект представляет собой веб-приложение "Мой гардероб", предназначенное для управления личными вещами пользователей. В современном мире, где быстрый темп жизни требует высокой эффективности и порядка, необходимость в удобных инструментах для организации личных вещей становится все более актуальной. "Мой гардероб" предлагает пользователям простой и интуитивно понятный интерфейс для создания виртуальных шкафов, добавления в них одежды и аксессуаров, а также для удобной организации и поиска предметов по различным параметрам.

Обоснование темы проекта заключается в том, что многие люди сталкиваются с проблемой беспорядка в гардеробе и трудностями с поиском нужных вещей. В условиях ограниченного времени и стремления к упрощению повседневных задач, создание инструмента, который бы позволил эффективно управлять своим гардеробом, представляет собой значительный шаг к улучшению качества жизни. "Мой гардероб" помогает пользователям систематизировать свои вещи, обеспечивая быстрый доступ к необходимым предметам одежды, что в свою очередь экономит время и усилия.

Целью проекта является разработка и внедрение веб-приложения, которое позволит пользователям создавать виртуальные шкафы, добавлять и организовывать вещи, а также эффективно искать их по различным параметрам. Для достижения этой цели были поставлены следующие задачи:

1. Узнать основы WEB-разработки приложение и его особенностей. Изучить архитектуру и технологии для разработки приложений.
2. Исследование существующих решений и потребностей пользователей. Важно понять, какие функции и возможности востребованы у пользователей подобных приложений, а также выявить недостатки текущих решений, чтобы создать продукт, превосходящий их по функционалу и удобству использования.
3. Проектирование и разработка интерфейса приложения. На этом этапе необходимо создать прототипы и макеты интерфейса, которые обеспечат максимально удобное и интуитивно понятное взаимодействие пользователя с приложением.
4. Реализация функционала авторизации и регистрации пользователей. Безопасность и конфиденциальность данных пользователей являются приоритетными задачами, поэтому важно обеспечить надежную систему аутентификации.
5. Создание модулей для добавления и организации гардероба и вещей. Пользователи должны иметь возможность легко добавлять вещи в свои виртуальные шкафы, указывать их параметры (тип, цвет, сезон и т.д.), а также организовывать их по различным категориям.
6. Внедрение системы фильтрации и поиска. Эффективный поиск и фильтрация вещей по различным параметрам позволит пользователям быстро находить нужные предметы одежды.

Проект "Мой гардероб" решает проблему беспорядка в гардеробе и трудностей с поиском нужных вещей. Приложение поможет пользователям систематизировать свои вещи и быстро находить нужные предметы одежды, что в конечном итоге способствует повышению эффективности и организации в повседневной жизни.

Дипломный проект выполняется по специализации "Разработчик". В рамках этой специализации были изучены и применены современные технологии и инструменты, необходимые для создания веб-приложений. Имеется опыт разработки веб-приложений с использованием таких технологий, как JavaScript, HTML, CSS, Java. Обучение на семинарах по разработке WEB-приложений позволило приобрести навыки в работе с базами данных и API, а также в организации процессов разработки и тестирования программного обеспечения.

Для реализации проекта были использованы следующие инструменты:

1. Языки программирования: Java, JavaScript
2. База данных: H2
3. Вёрстка: CSS, HTML
4. Система контроля версий: Git

В ходе разработки веб-приложения "Мой гардероб" мною были выполнены все роли: разработчика, дизайнера и тестировщика. В качестве разработчика я занимался созданием функционала и интеграцией различных компонентов. Этот подход позволил мне контролировать весь процесс разработки и обеспечивать высокое качество конечного продукта.

Таким образом, проект "Мой гардероб" представляет собой современное и удобное решение для управления личными вещами, которое помогает пользователям систематизировать свой гардероб и эффективно находить нужные предметы одежды. Введение данного веб-приложения в повседневную жизнь пользователей значительно упростит процесс организации и поиска вещей, что в свою очередь повысит их качество жизни и удовлетворенность.

# **Глава 1. Основы веб-разработки приложений**

## 1.1 Что такое веб-приложение и его особенности

Веб-приложение — это программа, которая работает на сервере и предоставляется пользователям через интернет с помощью веб-браузера. В отличие от традиционного программного обеспечения, которое необходимо устанавливать на локальный компьютер, веб-приложения доступны с любого устройства, подключенного к интернету. Это делает их особенно удобными и универсальными для различных целей, включая управление данными, электронную коммерцию, социальные сети, системы управления контентом и многое другое.

Основные характеристики и особенности веб-приложений включают:

1. Кроссплатформенность:

Универсальный доступ: Веб-приложения могут быть использованы на любой операционной системе, поддерживающей веб-браузеры, будь то Windows, macOS, Linux, iOS или Android.

Поддержка различных устройств: поскольку веб-приложения работают в браузере, они могут быть запущены на десктопах, ноутбуках, планшетах и смартфонах, обеспечивая пользователям возможность доступа к приложению в любое время и в любом месте.

1. Централизованное управление:

Обновления и поддержка: Веб-приложения обновляются на сервере, что позволяет мгновенно развертывать новые функции и исправления ошибок для всех пользователей без необходимости индивидуального обновления на каждом устройстве.

Администрирование и контроль: Администраторам проще управлять пользователями, данными и доступами из одного централизованного места, что упрощает процессы мониторинга и обеспечения безопасности.

1. Интерактивность и функциональность:

Динамический контент: Современные веб-приложения способны отображать динамический контент, который изменяется в зависимости от действий пользователя. Это делает взаимодействие с приложением более интерактивным и персонализированным.

Поддержка сложных операций: Веб-приложения могут выполнять сложные операции, такие как обработка данных, взаимодействие с базами данных, выполнение бизнес-логики и интеграция с внешними сервисами.

1. Масштабируемость и производительность:

Облачные технологии: Веб-приложения могут быть развернуты в облаке, что позволяет легко масштабировать ресурсы в зависимости от нагрузки и потребностей пользователей. Это особенно важно для приложений с большим количеством пользователей или высокой интенсивностью использования.

Кэширование и оптимизация: Использование методов кэширования и оптимизации на сервере и клиенте позволяет повысить производительность веб-приложений, уменьшая время загрузки страниц и улучшая общий пользовательский опыт.

1. Безопасность:

Защита данных: Веб-приложения могут использовать различные методы защиты данных, такие как SSL/TLS для шифрования передачи данных, аутентификация и авторизация пользователей, защита от атак (например, SQL-инъекций и XSS).

Регулярные обновления безопасности: Централизованное управление позволяет оперативно реагировать на угрозы безопасности и развертывать обновления, защищающие пользователей и их данные.

1. Экономичность:

Снижение затрат на разработку и развертывание: Веб-приложения не требуют сложной установки и настройки на каждом клиентском устройстве, что уменьшает затраты на поддержку и техническую помощь.

Доступ к широкому рынку: поскольку веб-приложения доступны через интернет, они могут привлечь пользователей со всего мира без необходимости создавать отдельные версии для различных платформ.

Таким образом, веб-приложения играют ключевую роль в современном цифровом мире, предоставляя пользователям доступ к различным сервисам и функциональности через интернет. Понимание их особенностей и принципов разработки является основой для создания успешных и востребованных продуктов.

## 1.2 Архитектура веб-приложений

Архитектура веб-приложений представляет собой сложную структуру, состоящую из нескольких уровней и компонентов, которые взаимодействуют друг с другом для обеспечения функциональности и удовлетворения потребностей пользователей. Эта многоуровневая архитектура играет ключевую роль в обеспечении высокой производительности, масштабируемости, безопасности и удобства использования приложения.

Основные уровни архитектуры веб-приложений можно разделить на несколько крупных категорий, каждая из которых выполняет определенные функции и задачи. Ключевые уровни включают клиентский уровень (Frontend), серверный уровень (Backend), базу данных (Database) и API (Application Programming Interface).

1. Клиентский уровень (Frontend) представляет интерфейс, с которым взаимодействует пользователь. Он отвечает за отображение информации, взаимодействие с пользователем и предоставление удобного интерфейса для работы с приложением. Этот уровень часто включает в себя HTML для структурирования контента, CSS для стилизации элементов и JavaScript для добавления интерактивности:

**HTML (HyperText Markup Language):** Основной язык разметки для создания веб-страниц. Он определяет структуру и содержание веб-страницы.

**CSS (Cascading Style Sheets):** Стиль, который управляет визуальным представлением HTML-элементов на странице. CSS позволяет разработчикам отделять содержание от дизайна, что облегчает поддержание и изменение стиля веб-страницы.

**JavaScript:** Скриптовый язык, который добавляет интерактивность и динамическое поведение веб-страницам. JavaScript позволяет выполнять задачи, такие как проверка данных на клиенте, взаимодействие с пользователем без перезагрузки страницы (AJAX), манипуляция DOM и многое другое.

Клиентский уровень играет ключевую роль в создании приятного пользовательского опыта и обеспечении удобства использования приложения.

2. Серверный уровень (Backend) отвечает за обработку бизнес-логики приложения и взаимодействие с базой данных. Здесь выполняются операции, связанные с обработкой запросов от клиентского уровня, а также управлением данными и отправкой информации обратно пользователю. Для написания серверного кода часто используются различные серверные языки программирования, такие как Java, Python, PHP и Node.js, а также фреймворки, облегчающие разработку и управление серверным кодом:

**Серверные языки программирования:** Различные языки программирования используются для написания серверного кода. Наиболее популярные из них включают Java, Python, PHP, Ruby, Node.js и ASP.NET. Выбор языка часто зависит от требований проекта, навыков команды и других факторов.

**Фреймворки:** Серверные фреймворки, такие как Spring (Java), Django (Python), Express (Node.js) и Laravel (PHP), облегчают разработку и управление серверным кодом. Они предоставляют структуру для построения веб-приложений, управления маршрутизацией, обработкой запросов и работы с базами данных.

**Управление сессиями и аутентификацией:** Серверный уровень также отвечает за управление сессиями пользователей и их аутентификацию. Это включает в себя хранение состояния пользователя, проверку учетных данных и защиту доступа к ресурсам приложения.

Серверный уровень играет важную роль в обеспечении безопасности, масштабируемости и надежности веб-приложения.

3. База данных (Database) является хранилищем информации, необходимой для работы приложения. Здесь хранятся данные, с которыми работает приложение, включая информацию о пользователях, продуктах, транзакциях и многое другое. Различные типы баз данных могут использоваться для хранения структурированных и неструктурированных данных, а также для обеспечения высокой производительности и отказоустойчивости.

**Реляционные базы данных:** Такие базы данных, как MySQL, PostgreSQL, Oracle и SQL Server, используют таблицы для хранения данных. Они поддерживают язык SQL (Structured Query Language) для управления данными и выполнения запросов. Реляционные базы данных хорошо подходят для структурированных данных и сложных запросов.

**Нереляционные базы данных (NoSQL):** Эти базы данных, такие как MongoDB, Cassandra, CouchDB и Redis, предлагают гибкость в хранении данных, которые не подходят для табличной модели. NoSQL базы данных часто используются для хранения больших объемов данных, неструктурированных данных или данных, которые часто изменяются.

База данных является важным компонентом веб-приложения, который обеспечивает постоянное хранение и доступ к данным пользователя.

4. API (Application Programming Interface) представляет собой интерфейс, позволяющий различным компонентам приложения взаимодействовать друг с другом. Он может быть внутренним, для взаимодействия между компонентами приложения, или внешним, для интеграции с другими системами и сервисами. Основные технологии API включают RESTful API и GraphQL, которые обеспечивают эффективное взаимодействие между клиентским и серверным уровнями приложения.

**RESTful API**: Стиль архитектуры API, который использует стандартные HTTP методы (GET, POST, PUT, DELETE) для взаимодействия с ресурсами. RESTful API широко используется из-за своей простоты и гибкости.

**GraphQL:** Альтернатива REST, предоставляющая клиентам возможность запрашивать только необходимые данные. GraphQL позволяет сократить количество запросов и увеличить производительность.

5. Дополнительные компоненты архитектуры веб-приложений играют важную роль в обеспечении эффективности, масштабируемости и надежности приложений. Некоторые из них включают:

**Кэширование:** Этот компонент позволяет улучшить производительность и снизить нагрузку на сервер путем временного хранения часто используемых данных. Кэширование может быть реализовано на уровне сервера с использованием инструментов, таких как Redis или Memcached, а также на стороне клиента, в браузерном кэше. Это позволяет ускорить доступ к данным и уменьшить время отклика приложения.

**Микросервисы:** это архитектурный подход, при котором приложение разбивается на небольшие и независимые сервисы, каждый из которых выполняет конкретную функцию или задачу. Микросервисы облегчают масштабирование и управление сложными системами, позволяя командам разрабатывать, развертывать и масштабировать части приложения независимо друг от друга. Это повышает гибкость и обеспечивает лучшее управление кодом и ресурсами.

**Системы управления контейнерами:** Инструменты, такие как Docker и Kubernetes, предоставляют средства для управления и развертывания приложений в контейнерах. Контейнеры обеспечивают изоляцию и консистентность среды выполнения, позволяя разработчикам упаковывать приложение и его зависимости в единый исполняемый блок. Это упрощает процесс развертывания, обновления и масштабирования приложений, а также обеспечивает консистентность окружения выполнения на различных платформах и средах.

## 1.3 Технологии для разработки веб-приложений

Разработка веб-приложений включает в себя использование широкого спектра технологий, каждая из которых играет ключевую роль в создании функциональных и привлекательных веб-ресурсов. Вот более подробное рассмотрение основных технологий, применяемых в веб-разработке:

HTML (HyperText Markup Language): это основной язык разметки, который определяет структуру веб-страницы. HTML используется для создания различных элементов, таких как заголовки, параграфы, изображения, формы и ссылки. Благодаря HTML разработчики могут организовывать контент и определять его иерархию на веб-странице.

CSS (Cascading Style Sheets): CSS является языком стилей, который определяет внешний вид и форматирование веб-страницы. С помощью CSS можно задавать цвета, шрифты, размеры, расположение элементов и другие визуальные аспекты веб-дизайна. CSS позволяет создавать красивые и привлекательные пользовательские интерфейсы, а также обеспечивает их согласованный стиль и внешний вид на различных устройствах и браузерах.

JavaScript: это скриптовый язык программирования, который добавляет интерактивность и динамическое поведение на веб-страницах. JavaScript позволяет создавать анимации, обрабатывать пользовательские события, взаимодействовать с элементами DOM (Document Object Model) и делать AJAX-запросы к серверу для обновления данных без перезагрузки страницы. Благодаря JavaScript веб-приложения становятся более функциональными, отзывчивыми и удобными для пользователей.

Фреймворки и библиотеки JavaScript: существует множество фреймворков и библиотек JavaScript, которые упрощают и ускоряют процесс разработки веб-приложений. Некоторые из самых популярных включают React, Angular, Vue.js, jQuery и Bootstrap. Фреймворки предоставляют готовые инструменты и архитектурные шаблоны для построения сложных пользовательских интерфейсов, в то время как библиотеки предлагают функциональные компоненты и утилиты для обработки данных и управления состоянием приложения.

Серверные языки программирования: для разработки серверной логики веб-приложений используются различные языки программирования, такие как Java, Python, PHP, Ruby и Node.js. Каждый из этих языков имеет свои преимущества и недостатки, а также широкий набор инструментов и фреймворков для работы. Например, Node.js позволяет создавать масштабируемые и высокопроизводительные веб-приложения с использованием асинхронного программирования, в то время как Django (Python) обеспечивает быструю разработку и обширный функционал для создания веб-приложений.

Базы данных: Хранение данных в веб-приложениях осуществляется с помощью различных систем управления базами данных (СУБД). Реляционные базы данных, такие как MySQL, PostgreSQL и Oracle, используются для хранения структурированных данных и обеспечивают поддержку транзакций, целостность данных и мощные средства запросов. Нереляционные базы данных, такие как MongoDB и Cassandra, предоставляют гибкое хранение неструктурированных данных и масштабируемость для работы с большими объемами информации.

Системы контроля версий: для эффективной работы над веб-проектами используются системы контроля версий, такие как Git. Они позволяют разработчикам отслеживать изменения в коде, управлять версиями проекта и совместно работать над кодовой базой. Системы контроля версий обеспечивают безопасное и надежное хранение истории изменений, а также упрощают процесс слияния изменений и управления конфликтами.

Инструменты для сборки и автоматизации: для упрощения процесса разработки и оптимизации рабочего процесса веб-разработчики часто используют инструменты для сборки и автоматизации задач. Такие инструменты, как Webpack, Gulp и Grunt, позволяют автоматизировать рутинные задачи, такие как компиляция и минификация кода, оптимизация изображений и тестирование приложения.

Контейнеризация и оркестрация: С ростом сложности и масштаба веб-приложений становится все более важным использовать контейнеризацию и оркестрацию. Контейнеризация позволяет упаковывать приложение и его зависимости в изолированные среды, называемые контейнерами, что обеспечивает консистентность окружения выполнения на различных платформах и средах. Один из самых популярных инструментов для контейнеризации - Docker, предоставляет простой и эффективный способ создания, развертывания и запуска контейнеров.

Оркестрация контейнеров, в свою очередь, обеспечивает управление и автоматизацию работы с контейнерами. Kubernetes (K8s) — это один из ведущих инструментов оркестрации, который позволяет разрабатывать, масштабировать и управлять приложениями, развернутыми в контейнерах, на основе сложных и гибких конфигураций.

Технологический стек для разработки веб-приложений постоянно развивается и расширяется, включая новые инструменты и технологии, направленные на улучшение производительности, безопасности и масштабируемости приложений. Основываясь на этом разнообразии инструментов и ресурсов, веб-разработчики могут создавать современные и инновационные веб-приложения, отвечающие требованиям современного цифрового мира.

## 1.4 Этапы разработки веб-приложений

Этапы разработки веб-приложений играют ключевую роль в организации и структурировании процесса создания приложения от начала до конца. Эти этапы не только обеспечивают логическую последовательность действий, но и способствуют систематизации работы, что упрощает управление проектом и повышает его эффективность.

Зачем нужны этапы разработки?

Структурировать процесс: Разделение проекта на отдельные этапы позволяет более организованно подходить к его реализации. Каждый этап имеет свои четко определенные цели и задачи, что упрощает управление процессом разработки.

Обеспечить качество: Каждый этап включает в себя определенные задачи и проверки, направленные на обеспечение качества разрабатываемого продукта. Это помогает выявлять и исправлять ошибки на ранних стадиях разработки, что способствует повышению качества и надежности приложения.

Соблюдать сроки: Планирование и последовательное выполнение этапов помогает контролировать ход работы и соблюдать установленные сроки. Это особенно важно для успешного завершения проекта вовремя и в рамках бюджета.

Основные этапы разработки веб-приложений:

Планирование: на этом этапе определяются основные цели и требования к приложению. Осуществляется анализ рынка, целевой аудитории, а также конкурентов. Происходит выработка стратегии разработки, определение функциональных и нефункциональных требований, а также создание плана проекта и распределение ресурсов.

Проектирование: В этой фазе определяется архитектура приложения, разрабатывается пользовательский интерфейс и внутренняя логика работы приложения. Создаются макеты (wireframes) и прототипы, которые визуализируют структуру и функционал приложения. Также происходит выбор технологий и инструментов для реализации задуманного.

Разработка: на этом этапе программисты начинают писать код приложения, используя выбранные технологии и инструменты. Разработка включает создание клиентской и серверной частей приложения, интеграцию с базой данных, а также тестирование кода на соответствие требованиям и корректную работу.

Тестирование: после завершения разработки приложение проходит через этап тестирования, включающий юнит-тестирование, интеграционное тестирование и системное тестирование. Целью тестирования является выявление ошибок и дефектов в работе приложения, а также проверка его соответствия требованиям и ожиданиям пользователей.

Внедрение: на этом этапе приложение развертывается на сервере и становится доступным для пользователей. Происходит настройка окружения выполнения, установка необходимых зависимостей и запуск приложения в рабочем режиме. Также важной частью внедрения является обучение пользователей и поддержка при первоначальном запуске.

Сопровождение: после внедрения приложение требует постоянного сопровождения и поддержки. Этот этап включает в себя регулярное обновление приложения, исправление ошибок и добавление нового функционала в соответствии с потребностями пользователей и изменениями в рыночной среде.

# **Глава 2. Анализ существующих решений и потребностей пользователей**

## 2.1 Обзор существующих приложений для управления гардеробом

На рынке существует несколько приложений, разработанных специально для управления гардеробом, предлагая пользователям различные инструменты для эффективной организации и управления своей одеждой. Анализ таких приложений позволяет выявить их сильные и слабые стороны, а также определить возможности для улучшения и разработки нового продукта.

**Stylebook** предлагает широкий набор инструментов для упорядочивания гардероба. Одной из его ключевых функций является возможность фотографировать или загружать изображения своей одежды, после чего пользователи могут указать различные характеристики каждого предмета, такие как цвет, размер, сезон и даже цену покупки. Это помогает пользователям лучше организовать свою одежду и быстро находить нужные предметы. Приложение позволяет составлять луки, оптимизируя процесс выбора нарядов на каждый день или особое мероприятие. Кроме того, Stylebook включает функции создания списков покупок и составления чемодана для поездок, что делает его полноценным инструментом для организации гардероба и планирования образов.

Однако у Stylebook есть и недостатки. Оно не поддерживает русский язык, что может быть неудобно для русскоязычных пользователей. Кроме того, приложение не имеет пробной бесплатной версии, что может отпугнуть потенциальных пользователей от его покупки.

Сильные стороны:

* Многофункциональность.
* Возможность детальной информации о каждом предмете.
* Инструменты для составления луков и планирования поездок.

Слабые стороны:

* Нет поддержки русского языка.
* Отсутствие бесплатной версии для тестирования.

**Getwardrobe** предлагает множество функций для управления гардеробом, включая возможность загрузки фотографий одежды из магазинов или сделанных самостоятельно. Пользователи могут указать дополнительную информацию о каждом предмете, такую как бренд, размер, стоимость и дата покупки. Одним из преимуществ приложения является возможность синхронизации данных между различными устройствами, что позволяет пользоваться Getwardrobe на всех своих гаджетах. Кроме того, пользователи могут делиться доступом к своему гардеробу с другими членами семьи или стилистами, что делает это приложение удобным инструментом для коллективного управления одеждой.

Тем не менее, бесплатная версия Getwardrobe имеет ограничения на количество предметов одежды, и иногда фотографии загружаются долго, что может затруднить работу с приложением.

Сильные стороны:

* Поддержка русского языка.
* Возможность ведения семейного гардероба и доступа для стилиста.
* Синхронизация на разных устройствах.

Слабые стороны:

* Ограничения бесплатной версии.
* Иногда медленная загрузка фотографий.

**Сombyne** представляет собой виртуальную примерочную, где пользователи могут создавать коллажи и мудборды изображений одежды. Интерфейс приложения простой и интуитивно понятный, что делает его легко доступным для всех пользователей. База данных включает в себя огромное количество вещей разных брендов, позволяя пользователям выбирать изображения для создания своих уникальных нарядов. Приложение также предоставляет ссылки на магазины, что облегчает процесс поиска и приобретения новой одежды. Однако у Сombyne есть некоторые недостатки: оно не поддерживает русский язык, а при загрузке фотографий своей одежды фон нужно удалять вручную.

Сильные стороны:

* Удобный редактор коллажей.
* Большая база данных брендов.
* Бесплатное использование.

Слабые стороны:

* Нет поддержки русского языка.
* Необходимость ручного удаления фона с фотографий.

**Lookscope** представляет собой минималистичное приложение для управления гардеробом, которое предлагает пользователю простой и интуитивно понятный интерфейс. С его помощью можно фотографировать одежду и создавать сочетания из разных предметов. Одним из основных преимуществ Lookscope является автоматическое удаление фона с фотографий, что упрощает процесс создания коллажей. Несмотря на то, что у приложения меньше функций по сравнению с другими аналогами, его минималистичный дизайн и простота использования делают его привлекательным для пользователей, предпочитающих простые решения.

Сильные стороны:

* Минималистичный интерфейс.
* Автоматическое удаление фона с фотографий.
* Бесплатное использование.

Слабые стороны:

* Мало функций по сравнению с аналогами.
* Нет поддержки русского языка.

**Closet Love** — это яркое и удобное приложение для управления гардеробом, которое предлагает пользователям разбивать вещи по функциям, например, для работы, занятий спортом или выходных. Приложение также позволяет создавать удачные сочетания внутри каждой категории, что помогает пользователям оптимально использовать свой гардероб. В платной версии доступны дополнительные функции, такие как автоматическое составление луков и создание списка одежды для поездок. Хотя бесплатная версия имеет некоторые ограничения, основные функции приложения вполне достаточно для эффективного управления гардеробом.

Сильные стороны:

* Удобное деление вещей на категории.
* Автоматическое составление луков.
* Яркий и интуитивно понятный интерфейс.

Слабые стороны:

* Ограничения бесплатной версии.
* Нет поддержки русского языка.

Таким образом, каждое из рассмотренных приложений обладает уникальными особенностями и преимуществами, которые могут удовлетворить различные потребности пользователей в управлении их гардеробом. Выбор подходящего приложения зависит от индивидуальных предпочтений и конкретных задач, которые пользователи хотят решить. Важно учитывать как функциональные возможности, так и потенциальные ограничения каждого приложения при выборе инструмента для управления гардеробом

## 2.2 Анализ потребностей пользователей

В данном разделе проводится анализ потребностей целевой аудитории, которая заинтересована в использовании приложения для управления гардеробом. Путем опросов, исследований и анализа рынка выявляются основные потребности и проблемы пользователей, которые могут быть решены с помощью создания нового приложения. Этот анализ является основой для определения функциональных требований к приложению.

Потребности пользователей в управлении гардеробом можно разделить на несколько ключевых категорий, каждая из которых отражает конкретные задачи и ожидания от приложения. Во-первых, пользователи хотят иметь возможность легко и удобно организовать свой гардероб. Для этого необходимо предоставить инструменты для добавления и категоризации одежды, с возможностью указания различных параметров, таких как цвет, размер, сезон и бренд. Возможность загружать фотографии одежды как из магазинов, так и самостоятельно, с автоматическим удалением фона, значительно облегчит процесс добавления новых предметов в гардероб.

Во-вторых, пользователи заинтересованы в функционале, который поможет им создавать и планировать наряды. Это включает в себя возможность составления луков, распределения их по категориям (например, для работы, отдыха или особых случаев) и сохранения любимых сочетаний. Планирование нарядов на неделю с помощью календаря или создание списков одежды для поездок также являются важными функциями, которые пользователи считают полезными для экономии времени и упрощения процесса выбора одежды.

Третьей важной потребностью является интеграция с социальными сетями и возможность делиться своим гардеробом с друзьями или стилистами. Пользователи хотят получать советы и вдохновение от других людей, а также возможность получать доступ к своему гардеробу с разных устройств и синхронизировать данные между ними. Это особенно актуально для семей, где нужно управлять гардеробами нескольких членов семьи, или для тех, кто пользуется услугами стилистов.

Четвертая категория потребностей связана с функциональностью, направленной на улучшение пользовательского опыта. Это включает в себя наличие статистики использования одежды, что помогает понять, какие вещи носятся чаще всего, и какие, возможно, не используются вообще. Функции напоминания о необходимости стирки или ремонте одежды также востребованы среди пользователей, так как помогают поддерживать гардероб в хорошем состоянии.

В ходе анализа также были выявлены проблемы, с которыми сталкиваются пользователи текущих решений. Среди них можно выделить отсутствие поддержки русского языка в большинстве популярных приложений, что затрудняет их использование для русскоязычных пользователей. Другой значимой проблемой является отсутствие пробных версий или ограниченные функциональные возможности бесплатных версий приложений, что ограничивает доступ к полному функционалу и не позволяет пользователям оценить все возможности приложения перед его покупкой.

Анализ потребностей пользователей показывает, что для создания успешного приложения для управления гардеробом необходимо учитывать несколько ключевых аспектов: удобство организации гардероба, функционал для создания и планирования нарядов, возможность интеграции с социальными сетями и другими устройствами, а также улучшение общего пользовательского опыта за счет дополнительных функций и устранения существующих проблем. Эти выводы станут основой для определения функциональных требований к разрабатываемому приложению и помогут создать продукт, максимально соответствующий ожиданиям и потребностям целевой аудитории.

## 2.3 Определение ключевых функциональных требований

На основе обзора существующих решений для управления гардеробом и анализа потребностей пользователей были выявлены ключевые функциональные требования, которые необходимо включить в разрабатываемое приложение. Эти требования учитывают потребности целевой аудитории и цели проекта, обеспечивая создание удобного, функционального и эффективного инструмента для управления гардеробом. В данном разделе формулируются основные функции, которые должно выполнять приложение.

1. Организация гардероба

Пользователи хотят иметь возможность легко и удобно добавлять одежду в приложение. Для этого необходимо реализовать:

* Добавление одежды: Возможность загружать фотографии одежды из магазинов или делать снимки самостоятельно с автоматическим удалением фона.
* Категоризация: Разделение одежды по категориям (цвет, размер, сезон, бренд и т.д.) для упрощения поиска и управления.
* Детализация: Возможность указания подробной информации о каждом предмете (состав ткани, дата покупки, цена и т.д.).

2. Создание и планирование нарядов

Для упрощения выбора одежды и экономии времени пользователям необходимо:

* Создание луков: Возможность составлять наряды из различных предметов гардероба и сохранять их в отдельные папки по категориям (работа, отдых, особые случаи).
* Календарь: Планирование нарядов на неделю или месяц вперед с помощью встроенного календаря.
* Списки для поездок: Создание списков одежды для поездок с возможностью отмечать сочетания и понимать, сколько уникальных образов получится.

3. Интеграция и социальные функции

Для повышения взаимодействия и получения советов пользователи ожидают:

* Синхронизация: Поддержка синхронизации данных между разными устройствами, включая возможность управления гардеробами нескольких членов семьи.
* Доступ для стилистов: Возможность предоставить доступ к гардеробу профессиональным стилистам для дистанционного упорядочения и консультирования.
* Социальные сети: Интеграция с социальными сетями для деления нарядами и получения обратной связи от друзей и подписчиков.

4. Улучшение пользовательского опыта

Для обеспечения максимального удобства и эффективности работы с приложением необходимо:

* Статистика использования: Предоставление данных о том, как часто носится каждая вещь, для оптимизации гардероба.
* Напоминания: Функции напоминаний о необходимости стирки, химчистки или ремонта одежды.
* Автоматическое составление луков: Функция автоматического подбора нарядов на основе имеющейся одежды ("Удиви меня").

5. Многоязычная поддержка и доступность

Для удовлетворения потребностей пользователей из разных регионов необходимо:

* Поддержка русского языка: Включение русского языка в интерфейс приложения.
* Доступность: Разработка интерфейса, доступного для людей с ограниченными возможностями, включая поддержку экранных читалок и других вспомогательных технологий.

6. Модели монетизации

Для обеспечения финансовой устойчивости проекта и предоставления пользователям гибких возможностей использования:

* Бесплатная версия: Ограниченная функциональность с возможностью загрузки до 100 предметов одежды.
* Премиум-версия: Расширенные возможности, включая синхронизацию между устройствами, доступ для стилистов, расширенные категории и подкатегории, а также функции автоматического составления луков и создания списков для поездок.

Эти ключевые функциональные требования обеспечат создание приложения, максимально соответствующего ожиданиям и потребностям целевой аудитории, что позволит предложить пользователям эффективный инструмент для управления своим гардеробом.

# **Глава 3. Проектирование и разработка веб-приложения 'Мой гардероб'**

## 3.1 Разработка требований к функционалу

Разработка функциональных требований — это важный этап в процессе создания веб-приложения. В разных компаниях существуют различные подходы к написанию функциональных требований, но я решил попробовать подход от компании Retail Rocket к написанию функциональных требований. Функциональные требования представляют собой детализированное описание того, что должно быть реализовано в рамках проекта. Они служат постановкой задачи для разработчиков и обеспечивают понимание между продакт-менеджерами и командой разработки.

Функциональные требования — это постановка задачи разработчику, которая определяет, что именно должно быть сделано. Все, что не указано в требованиях, выполняется на усмотрение разработчика, что может привести к расхождению с представлениями продакт-менеджера об ожидаемом результате. Поэтому функциональные требования должны быть максимально полными и содержать ответы на все возможные вопросы по задаче. Они играют ключевую роль в управлении ожиданиями и предотвращении недопонимания в процессе разработки.

Состав функциональных требований:

1. User Story — истории пользователей, которые описывают, чего ожидает конечный пользователь от системы. Это помогает команде разработки понять, для кого они создают продукт и как он будет использоваться.

2. Use Cases — сценарии использования, которые показывают, как фича будет применяться на практике. Они описывают пошагово, какие действия выполняет пользователь и какую реакцию системы он получает в ответ.

3. Wireframes — визуализация идеи. Вайрфреймы представляют собой наброски интерфейсов, которые помогают продемонстрировать, как будет выглядеть и работать приложение. Они позволяют наглядно представить расположение элементов и взаимодействие с ними.

Применение функциональных требований к проекту "Мой гардероб"

Для разрабатываемого мною веб-приложения "Мой гардероб" функциональные требования будут включать следующие аспекты:

User Story:

* Как пользователь, я хочу добавить одежду в свой виртуальный гардероб, чтобы иметь возможность управлять своим реальным гардеробом.
* Как пользователь, я хочу создавать и сохранять наряды, чтобы иметь готовые решения для разных случаев.
* Как пользователь, я хочу планировать наряды на неделю вперед, чтобы экономить время на сборы по утрам.
* Как пользователь, я хочу синхронизировать гардероб между разными устройствами, чтобы иметь доступ к нему в любом месте.

Use Cases:

* Добавление одежды: Пользователь фотографирует одежду или загружает изображение из интернета, система автоматически удаляет фон, пользователь добавляет информацию о предмете (цвет, размер, сезон и т.д.).
* Создание нарядов: Пользователь выбирает несколько предметов одежды и создает наряд, который можно сохранить и классифицировать по категориям (работа, отдых, особые случаи).
* Планирование нарядов: Пользователь открывает календарь, выбирает день и создает наряд для этого дня. Система отправляет напоминания и предоставляет доступ к списку запланированных нарядов.
* Синхронизация данных: Пользователь регистрируется и входит в систему на различных устройствах, данные автоматически синхронизируются, обеспечивая доступ к актуальному гардеробу.

Wireframes:

* Главная страница: Отображение последних добавленных предметов одежды, кнопки для добавления нового предмета, навигация по категориям.
* Страница добавления одежды: Поле для загрузки изображения, форма для ввода информации о предмете.
* Календарь нарядов: Календарь с отмеченными днями, кнопка для создания нового наряда, список запланированных нарядов.
* Страница создания наряда: Список доступной одежды, область для перетаскивания предметов и создания наряда, кнопка для сохранения.

Применения четко определенных функциональных требований, обеспечивают следующие преимущества:

1. Согласованность и понимание: Все участники проекта понимают, что именно нужно реализовать, и работают в одном направлении.
2. Снижение рисков: Полные и детализированные требования помогают избежать недоразумений и уменьшить количество доработок.
3. Управление ожиданиями: Функциональные требования помогают продакт-менеджерам и заинтересованным сторонам точно понимать, какие функции и в каком виде будут реализованы.
4. Повышение качества: Тщательная проработка требований способствует созданию более качественного продукта, удовлетворяющего потребности пользователей.

Таким образом, функциональные требования являются основой для успешной разработки веб-приложения "Мой гардероб". Они обеспечивают четкое представление о конечном результате, что способствует эффективной работе разработки.

## 3.2 Проектирование интерфейса

Проектирование пользовательского интерфейса (UI) является важным этапом в создании веб-приложения "Мой гардероб". Этот процесс направлен на создание интуитивно понятного, простого и привлекательного внешнего вида приложения, обеспечивая удобство использования для всех пользователей. В данном разделе описываются основные этапы проектирования UI, которые включают анализ пользовательского опыта, создание прототипов, разработку визуального дизайна и тестирование.

Первым шагом в проектировании интерфейса стало исследование целевой аудитории для определения их потребностей и предпочтений. Были проведены опросы и интервью с потенциальными пользователями, чтобы понять, как они взаимодействуют с подобными приложениями, какие функции для них наиболее важны и какие проблемы они хотели бы решить с помощью "Моего гардероба".

На основе собранных данных были созданы пользовательские сценарии и карты путешествий пользователей (user journey maps). Эти инструменты помогли визуализировать основные взаимодействия пользователей с приложением, выявить ключевые моменты и потенциальные проблемы, которые нужно учитывать при разработке интерфейса.

После анализа пользовательского опыта началась работа над созданием прототипов интерфейса. Прототипы представляют собой черновые макеты экранов приложения, которые включают основные элементы интерфейса, такие как кнопки, формы и меню. В отличие от использования инструментов, таких как Figma, здесь прототипы сразу воплощались в коде, что позволило быстро тестировать и вносить изменения.

Прототипы были протестированы с участием пользователей для получения обратной связи. Это тестирование позволило выявить, насколько интуитивно понятен и удобен интерфейс, и что нужно улучшить. Обратная связь пользователей была критически важной для внесения изменений и улучшений на ранних этапах разработки.

Разработка визуального дизайна включала определение цветовой схемы, шрифтов и стилей, которые будут использоваться в приложении. Основной целью было создание визуально привлекательного и в то же время функционального дизайна, который бы соответствовал ожиданиям пользователей и поддерживал общий стиль приложения.

Окончательные макеты экранов и элементов интерфейса включали детализированные версии всех страниц приложения, таких как главная страница, страницы добавления одежды. Каждый элемент был тщательно проработан, чтобы обеспечить удобство использования и эстетическую привлекательность.

Последним этапом проектирования интерфейса стало тестирование готового интерфейса с участием пользователей. Это тестирование помогло выявить оставшиеся проблемы и недостатки, которые могли быть упущены на предыдущих этапах. Пользователи оценивали удобство использования, визуальное восприятие и функциональность интерфейса.

На основе полученной обратной связи были внесены необходимые исправления и улучшения. Этот итеративный процесс тестирования и доработки продолжался до тех пор, пока интерфейс не стал максимально удобным и функциональным.

Важной особенностью интерфейса "Моего гардероба" является его интуитивность и удобство использования. На всех страницах приложения предусмотрена шапка с логотипом и необходимыми кнопками для каждой конкретной страницы (см. приложение 1). Шапка всегда зафиксирована сверху, при прокрутке страницы, она уменьшается в размере и становится прозрачнее для удобства пользования (см. приложение 2). Это обеспечивает легкую навигацию и доступ к основным функциям приложения.

Например, на главной странице пользователи могут быстро авторизоваться или зарегистрировать и перейти в личный кабинет (см. приложение 3), либо же получить контакты для связи. Визуальные элементы, такие как кнопки и иконки, оформлены так, чтобы они были легко узнаваемы и удобны в использовании. В личном кабинете есть также кнопки, для быстрого добавления шкафа, отображение текущих шкафов, и также редактирование их в модальном окне.

Таким образом, процесс проектирования интерфейса веб-приложения "Мой гардероб" включал тщательный анализ пользовательского опыта, создание и тестирование прототипов, разработку визуального дизайна и итеративное тестирование готового интерфейса. Все эти этапы были направлены на создание удобного, интуитивно понятного и привлекательного интерфейса, который удовлетворяет потребности пользователей и обеспечивает удобство использования.

## 3.3 Архитектура приложения

Архитектура веб-приложения "Мой гардероб" обеспечивает высокую производительность, масштабируемость, безопасность и удобство использования. Основные компоненты архитектуры включают клиентский уровень, серверный уровень, базу данных и API. В данном разделе описываются используемые технологии и подходы для реализации каждого из этих компонентов.

Клиентский интерфейс приложения был разработан с использованием HTML, CSS и JavaScript. Эти технологии обеспечивают создание динамичных и интерактивных пользовательских интерфейсов, которые удобны и интуитивны в использовании. Для обеспечения адаптивности интерфейса на различных устройствах и экранах были применены современные CSS-техники и медиазапросы.

Серверная часть приложения была реализована на Java с использованием фреймворка Spring Boot, который обеспечивает мощные возможности для разработки веб-приложений. Spring Boot использовался для создания контроллеров и REST-контроллеров, которые обрабатывают HTTP-запросы и управляют бизнес-логикой приложения.

Бизнес-логика была структурирована в виде сервисов, что обеспечивает модульность и упрощает поддержку кода. Например, сервисы для управления аккаунтами, гардеробами, предметами и файлами были разделены на отдельные компоненты, такие как AccountService, WardrobeService, ItemService, FileStorageService. Это позволяет легко масштабировать и модифицировать приложение по мере необходимости.

Для хранения данных временно использовалась встроенная база данных H2. Это решение обеспечивало быструю разработку и тестирование, а также простоту миграции на другие реляционные базы данных, такие как MySQL или PostgreSQL, в будущем. Доступ к данным осуществлялся с использованием JPA (Java Persistence API), что упрощало управление данными и взаимодействие с базой данных.

Использование H2 в начальных стадиях разработки позволило сосредоточиться на создании функциональности и бизнес-логики, не отвлекаясь на настройку сложных внешних баз данных. В дальнейшем, при необходимости, будет возможен переход на более производительные и масштабируемые решения.

Для взаимодействия между клиентским и серверным уровнями приложения был разработан RESTful API, обеспечивающий четкое разделение между клиентом и сервером, что упрощает разработку и тестирование приложения. RESTful API включает несколько основных компонентов, обеспечивающих функциональность приложения:

AccountService: Этот сервис отвечает за управление учетными записями пользователей, включая создание, обновление и удаление учетных записей. Взаимодействие с AccountService осуществляется через AccountController, который обрабатывает соответствующие запросы и вызывает методы сервиса для выполнения операций с учетными записями.

WardrobeService: Сервис для управления гардеробами пользователей, позволяющий создавать, редактировать и удалять гардеробы. WardrobeController предоставляет API для взаимодействия с этим сервисом, позволяя клиентам выполнять необходимые операции с гардеробами.

ItemService: Этот сервис отвечает за управление отдельными предметами одежды в гардеробе. Через ItemController клиенты могут добавлять, обновлять и удалять элементы гардероба, а также получать информацию о конкретных предметах одежды.

FileStorageService: Сервис для управления загрузкой и хранением файлов, таких как фотографии одежды. ImageController предоставляет API для получения фотографии, для дальнейшей обработки.

AuthController: Отдельный контроллер, который отвечает за аутентификацию и регистрацию пользователей. Этот контроллер использует AccountService для выполнения операций по аутентификации, включая вход в систему, регистрацию новых пользователей и управление сессиями. AuthController обеспечивает безопасный доступ к сервисам приложения, гарантируя, что только авторизованные пользователи могут выполнять определенные действия.

RESTful API обеспечивает безопасность и аутентификацию пользователей, гарантируя, что доступ к данным и функциональности приложения осуществляется только авторизованными пользователями. Этот подход также позволяет легко интегрировать дополнительные функции и сервисы в приложение, обеспечивая гибкость и масштабируемость архитектуры. Это также упрощает добавление новых клиентских приложений, таких как мобильные версии или интеграции с другими сервисами.

Для улучшения производительности и снижения нагрузки на сервер в будущем планируется реализация кэширования. Это позволит значительно ускорить обработку повторяющихся запросов и снизить время отклика приложения.

Использование микросервисной архитектуры также рассматривается как перспективный подход для дальнейшего развития приложения. Разделение на независимые компоненты облегчит масштабирование и управление приложением, улучшая его устойчивость и производительность.

Таким образом, архитектура веб-приложения "Мой гардероб" была спроектирована с учетом современных технологий и лучших практик, обеспечивая высокую производительность, масштабируемость и удобство использования для всех пользователей.

# **Глава 4. Реализация функционала приложения**

## 4.1 Создание и управление шкафами

В веб-приложении «Мой гардероб» пользователи имеют возможность создавать, просматривать, обновлять и удалять свои гардеробы. Я разработал механизм управления шкафами, который предоставляет пользователям простой и интуитивно понятный интерфейс для добавления, организации и управления своими коллекциями одежды.

Для реализации, были использованы следующие технологии:

Для бэкенд-части приложения я использовал Java с применением фреймворка Spring Boot, обеспечивая высокую производительность и надежность.

Также создал RESTful API для обмена данными между клиентской и серверной частями приложения, что позволяет пользователям выполнять различные операции с шкафами через HTTP-запросы.

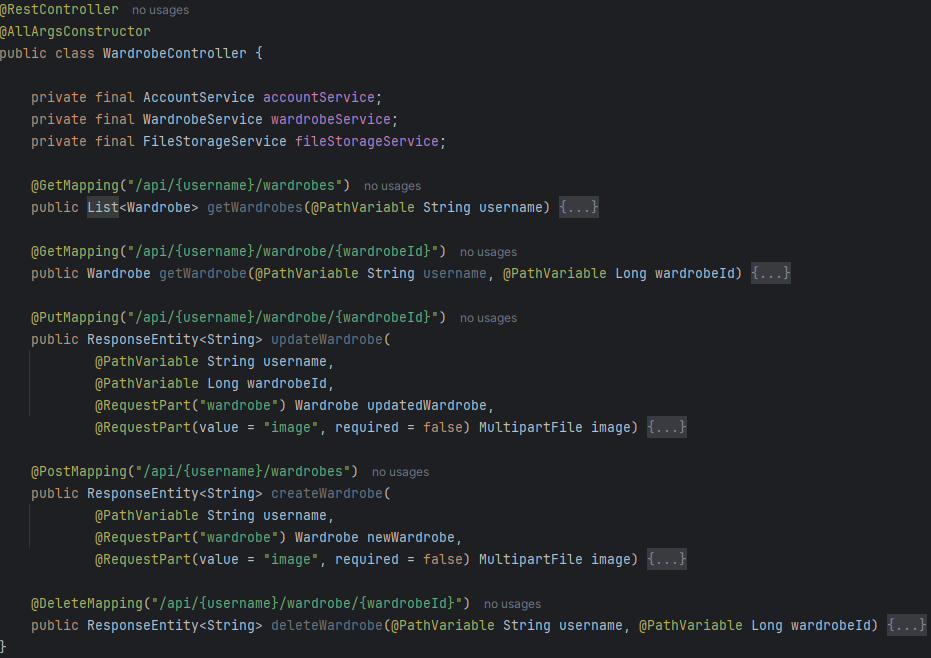
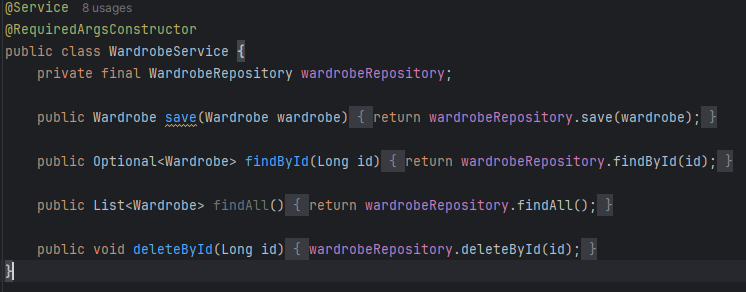
Для хранения изображений шкафов был использован FileStorageService, который обеспечивает удобное и эффективное хранение файлов и предоставляет URL-адреса для доступа к ним.

Для работы с базой данных была использована технология Hibernate, обеспечивающая удобную работу с объектно-реляционной моделью данных.

Взаимодействие:

Сервис WardrobeService: Я разработал сервис для управления шкафами, который включает в себя методы для создания, чтения, обновления и удаления шкафов. Этот сервис обеспечивает логику бизнес-процессов, связанных с управлением шкафами.

Контроллер WardrobeController: для обработки HTTP-запросов, связанных с операциями над шкафами, я создал контроллер, который взаимодействует с сервисом WardrobeService, обеспечивая выполнение соответствующих операций. Приведенный ниже код позволяет осуществлять операции по созданию, получению, обновлению и удалению шкафов:



Frontend UI: для предоставления пользовательского интерфейса я использовал HTML, CSS и JavaScript, обеспечивая простоту и удобство использования приложения.

Этот механизм управления шкафами предоставляет пользователям возможность эффективно организовывать свою одежду, сохраняя при этом простоту и удобство использования приложения.

## 4.2 Добавление и организация предметов

Для эффективного управления коллекцией одежды в приложении "Мой гардероб" разработан механизм, позволяющий пользователям создавать, просматривать, обновлять и удалять предметы. Этот функционал включает в себя использование ItemService (см. рис. 1) и ItemController (см. рис. 2), которые обеспечивают бизнес-логику и обработку HTTP-запросов, соответственно.

Как и в предыдущих разделах, для реализации приложения использовался Java с использованием фреймворка Spring Boot, RESTful API, Hibernate.

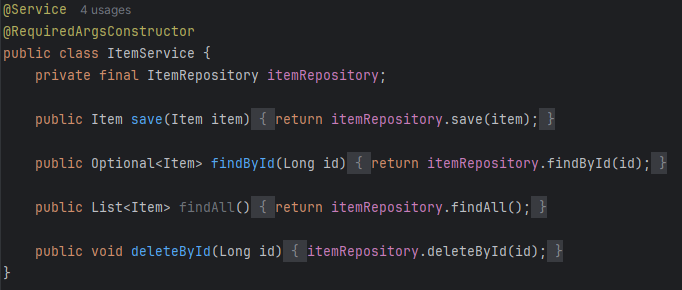


Рисунок 1

ItemService представляет собой сервисный класс, который обеспечивает основные операции с предметами одежды. В нем реализованы методы для сохранения, поиска, получения всех предметов и удаления предмета по его идентификатору. Сервис использует ItemRepository для взаимодействия с базой данных, где хранится информация о предметах.

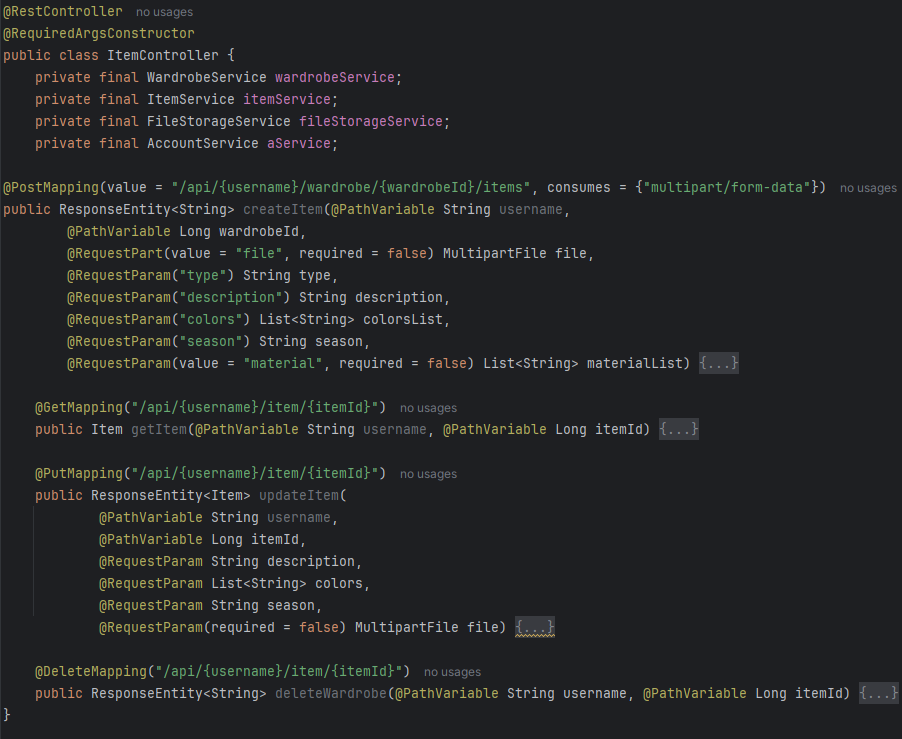


Рисунок 2

ItemController является REST контроллером, который обрабатывает HTTP-запросы, связанные с предметами одежды. Он содержит методы для создания, получения, обновления и удаления предметов в шкафах, что позволяет пользователям эффективно управлять своей коллекцией одежды через интерфейс приложения. Контроллер взаимодействует с сервисами ItemService, WardrobeService, FileStorageService и AccountService, обеспечивая выполнение бизнес-логики и сохранение данных.

Взаимодействие:

Создание предметов: Я разработал методы контроллера для приема данных о новом предмете, включая изображение, тип, описание, цвета, сезон и материалы, и сохранения этих данных в базе данных.

Получение предметов: Реализован метод для получения информации о предмете по его идентификатору.

Обновление предметов: Мы предоставили возможность обновлять данные предмета, включая описание, цвета, сезон и изображение, при необходимости.

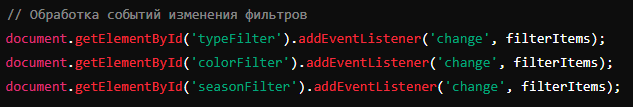
Удаление предметов: Реализована функция удаления предмета из шкафа по его идентификатору.

Этот функционал обеспечивает пользователям удобное управление и организацию их одежды в веб-приложении "Мой гардероб", обеспечивая при этом высокую производительность и надежность операций.

## 4.3 Система фильтрации и поиска

Для обеспечения удобства пользователей в поиске и организации своей одежды в приложении "Мой гардероб", разработана эффективная система фильтрации и поиска предметов. Этот функционал позволяет пользователям быстро находить нужную одежду, используя различные критерии, такие как тип предмета, цвет и сезон.

Фильтрация и поиск предметов осуществляется с использованием JavaScript на стороне клиента. Для обработки событий изменения фильтров, используются методы addEventListener, которые прослушивают изменения элементов веб-страницы. Это позволяет реагировать на изменения значений фильтров и автоматически запускать функцию фильтрации при каждом изменении. Ниже представлен код, демонстрирующий этот процесс:



В этом фрагменте кода для каждого фильтра (типа, цвета и сезона) устанавливается обработчик событий, который вызывает функцию filterItems при изменении значения фильтра.

Кроме того, в коде присутствует обработчик события сброса фильтров, который сбрасывает значения фильтров к их исходным значениям. Ниже представлен соответствующий фрагмент кода:



Этот код устанавливает обработчик события на кнопку сброса фильтров и определяет функцию, которая будет вызываться при щелчке на этой кнопке. Внутри этой функции устанавливаются исходные значения фильтров для возвращения к начальному состоянию.

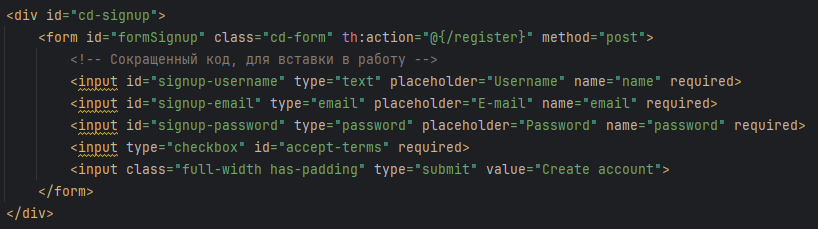
## 4.4 Реализация авторизации и регистрации пользователей

Авторизация и регистрация пользователей представляют собой ключевые компоненты веб-приложения "Мой гардероб", обеспечивая доступ к его функционалу и персонализацию. На текущем этапе разработки авторизация и регистрация реализованы без управления сессиями, и пользовательские данные сохраняются в LocalStorage браузера для удобства использования. В этом разделе описаны основные шаги по реализации данных функций в текущей версии приложения.

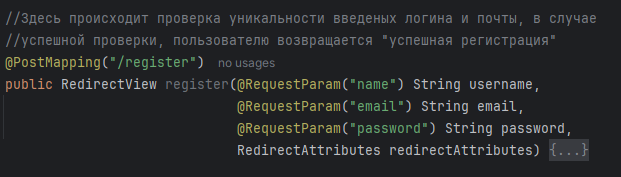
Процесс регистрации новых пользователей включает следующие шаги:

На главной странице предоставляется кнопка войти, в которой пользователь, выбирает регистрация и заполняет поля с необходимой информацией, такой как логин, адрес электронной почты и пароль (см. приложение 3).

Регистрация новых пользователей происходит через HTML форму, которая отправляет данные на сервер для обработки. Ниже приведена сокращенная разметка HTML:



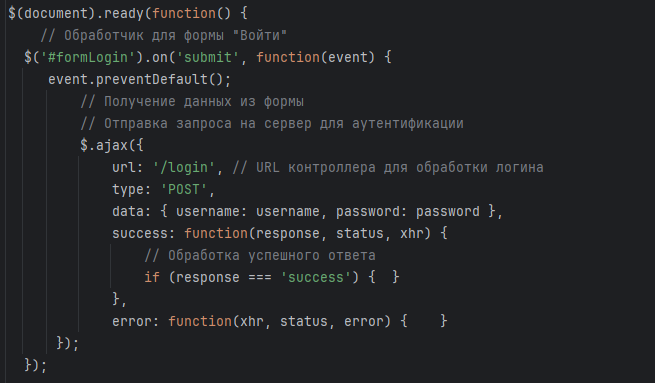
Далее данные передаются в AuthController, который обрабатывает их, и в случае обнаружения конфликтов с уже существующими данными выводится соответствующее сообщение об ошибке (см. приложение 4).

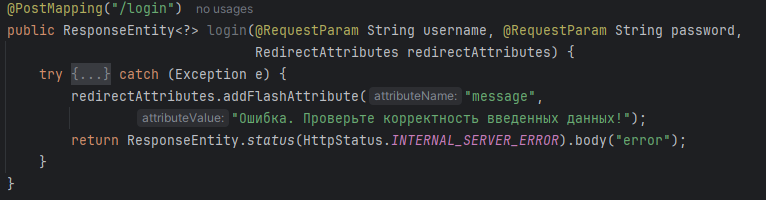


Процесс авторизации пользователей включает следующие шаги:

На главной странице после нажатия кнопки «Войти» также доступна форма для входа, в которой пользователь вводит свои данные: логин и пароль (см. приложение 5).

После отправки формы происходит проверка введенных данных на сервере с помощью AJAX-запроса к AuthController.





В случае успешной аутентификации пользователь перенаправляется на главную страницу приложения.

Процесс восстановления пароля включает следующие шаги:

Для восстановления пароля также предусмотрена форма на главной странице в кнопке «Войти», в которой пользователь вводит адрес своей электронной почты (см. приложение 6).

При отправке формы адрес электронной почты проверяется на сервере с помощью AJAX-запроса к AuthController. В будущем, после реализация данной функции в случае успешного совпадения адреса электронной почты генерируется ссылка для сброса пароля, которая отправляется на указанный адрес.

# ЗАКЛЮЧЕНИЕ

В ходе разработки проекта "Мой гардероб" были получены значительные теоретические и практические результаты. Теоретическая база исследования позволила глубже понять современные технологии и архитектуру веб-приложений, а практическая часть обеспечила создание функционального и удобного инструмента для управления гардеробом.

Оценка проведённого исследования и его результаты:

Проект "Мой гардероб" успешно реализован и продемонстрировал свою актуальность и эффективность. В ходе разработки были достигнуты следующие результаты:

Теоретическая часть:

* Исследованы основы веб-разработки, включая архитектуру веб-приложений и современные технологии, такие как JavaScript, HTML, CSS и Java.
* Проведен анализ существующих решений для управления гардеробом, выявлены их недостатки и определены ключевые функциональные требования для нового приложения.
* Изучены потребности пользователей, что позволило создать приложение, удовлетворяющее их ожидания и предпочтения.

Практическая часть:

* Спроектирован и разработан интерфейс приложения, обеспечивающий удобное и интуитивно понятное взаимодействие пользователя с системой.
* Реализованы функционалы авторизации и регистрации пользователей, что обеспечивает безопасность и конфиденциальность данных.
* Созданы модули для добавления и организации вещей, позволяющие пользователям эффективно управлять своими гардеробами.
* Внедрена система фильтрации и поиска, что значительно упрощает процесс нахождения нужных предметов одежды.

Практическая значимость работы:

Проект "Мой гардероб" обладает высокой практической значимостью. Приложение предоставляет пользователям инструмент для систематизации и управления своими вещами, что помогает экономить время и усилия. Введение данного веб-приложения в повседневную жизнь пользователей способствует повышению их организованности и эффективности.

Для дальнейшего совершенствования приложения и расширения его функционала рекомендуется:

1. Интеграция с облачными сервисами: это позволит пользователям синхронизировать данные между различными устройствами и обеспечит доступ к гардеробу в любой точке мира.

2. Разработка мобильного приложения: это расширит аудиторию пользователей и обеспечит удобство использования приложения на мобильных устройствах.

3. Внедрение машинного обучения: с помощью алгоритмов машинного обучения можно предложить пользователям рекомендации по подбору одежды и создания образов.

4. Расширение функционала:

4.1 Добавление новых функций, таких как планирование нарядов и отслеживание состояния вещей, сделает приложение еще более полезным и привлекательным для пользователей.

4.2 Добавление функции, для удаления заднего фона у загружаемых изображений и конвертации их в PNG.

4.3 Добавление функции конструктора нарядов, которая позволит пользователям комбинировать различные типы одежды и аксессуаров. Выбранные элементы будут отображаться на виртуальном манекене, что даст возможность предварительно увидеть, как будет выглядеть весь комплект.

Цели проекта достигнуты, задачи выполнены. В результате проведенного исследования и разработки веб-приложения "Мой гардероб" была доказана гипотеза о необходимости и полезности инструмента для управления личными вещами. Приложение позволяет пользователям систематизировать гардероб и быстро находить нужные предметы одежды, что улучшает качество их жизни.

Для дальнейшего совершенствования приложения "Мой гардероб" предлагается:

Расширение базы данных: Включение дополнительных параметров для вещей, таких как материал, бренд, стоимость и т.д.

Улучшение пользовательского интерфейса: Проведение регулярных тестирований и сбор обратной связи от пользователей для постоянного улучшения интерфейса.

Поддержка международных пользователей: Добавление мультиязычности и локализации для привлечения пользователей из разных стран.

В заключение, проект "Мой гардероб" является значимым вкладом в область веб-разработки и управления личными вещами, демонстрируя высокую практическую ценность и потенциал для дальнейшего развития и улучшения.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Barnes, D. Практикум по программированию на JavaScript. Москва: НОУ «Интуит», 2016.

Бретт, М. PHP и MySQL. Исчерпывающее руководство. Санкт-Петербург: Питер, 2012.

Веру, Л. Секреты CSS. Идеальные решения ежедневных задач. Санкт-Петербург: Питер, 2016.

Голдстайн, А., Лазарис, Л., & Уэйл, Э. HTML5 и CSS3 для всех. Москва: Вильямс, 2012.

Дамашке, Г. PHP и MySQL. Москва: НТ Пресс, 2012.

Дэкетт, Дж. HTML и CSS. Разработка и создание веб-сайтов. Москва: Эксмо, 2014.

Flanagan, D. JavaScript. Подробное руководство. Вильямс, 2012.

Бауэрс, Д., & Лейн, Л. HTML и CSS. Разработка и дизайн веб-сайтов. Санкт-Петербург: Питер, 2017.

Блох, Дж. Effective Java. Addison-Wesley, 2018.

Smith, J. Design Patterns in Modern Web Development. Journal of Web Engineering, 17(4), 345-362 (2018).

Johnson, M. User Experience Design for Web Applications. International Journal of Human-Computer Interaction, 35(2), 156-170 (2019).

Smirnov, Н. Правила создания красивого сайта. [Электронный ресурс] - Режим доступа: http://seoklub.ru/kak\_sdelat\_krasivyi\_sait.html (Дата обращения: 18.11.16).

Mozilla Developer Network (MDN). JavaScript Guide. [Электронный ресурс] - Режим доступа: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (Дата обращения: 10.06.2024).

Spring Framework. Spring Documentation. [Электронный ресурс] - Режим доступа: https://spring.io/projects/spring-framework#learn (Дата обращения: 10.06.2024).

Bootstrap. Bootstrap Documentation. [Электронный ресурс] - Режим доступа: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (Дата обращения: 10.06.2024).

World Wide Web Consortium (W3C). HTML5 Specification. [Электронный ресурс] - Режим доступа: https://www.w3.org/TR/html5/ (Дата обращения: 10.06.2024).

ECMAScript. ECMAScript 2018 Language Specification. [Электронный ресурс] - Режим доступа: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ (Дата обращения: 10.06.2024).

Proceedings of the International Conference on Web Engineering. [Электронный ресурс] - Режим доступа: https://icwe2020.webengineering.org/ (Дата обращения: 10.06.2024).

H2 Database Engine. H2 Documentation. [Электронный ресурс] - Режим доступа: http://www.h2database.com/html/main.html (Дата обращения: 10.06.2024).

Git. Git Documentation. [Электронный ресурс] - Режим доступа: https://git-scm.com/doc (Дата обращения: 10.06.2024).

Habr. Как писать функциональные требования. [электронный ресурс] - Режим доступа: https://habr.com/ru/companies/retailrocket/articles/431572/ (Дата обращения: 10.06.2024).

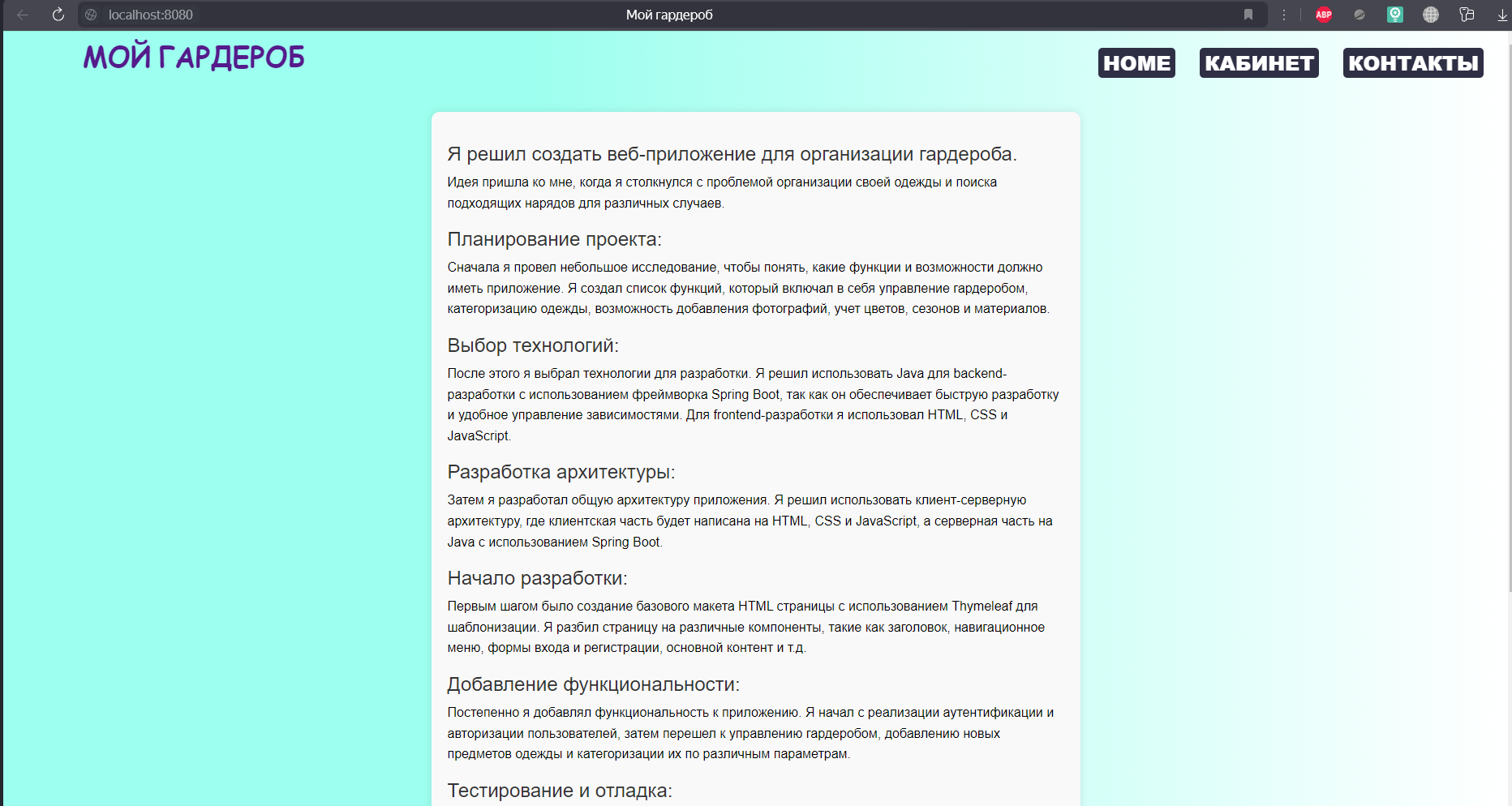
Habr. Основы разработки веб-приложений. [электронный ресурс] - Режим доступа: https://habr.com/ru/companies/piter/articles/241591/ (Дата обращения: 10.06.2024).

Habr. Модальное окно на чистом CSS и JS. [электронный ресурс] - Режим доступа: https://habr.com/ru/articles/716142/ (Дата обращения: 10.06.2024).

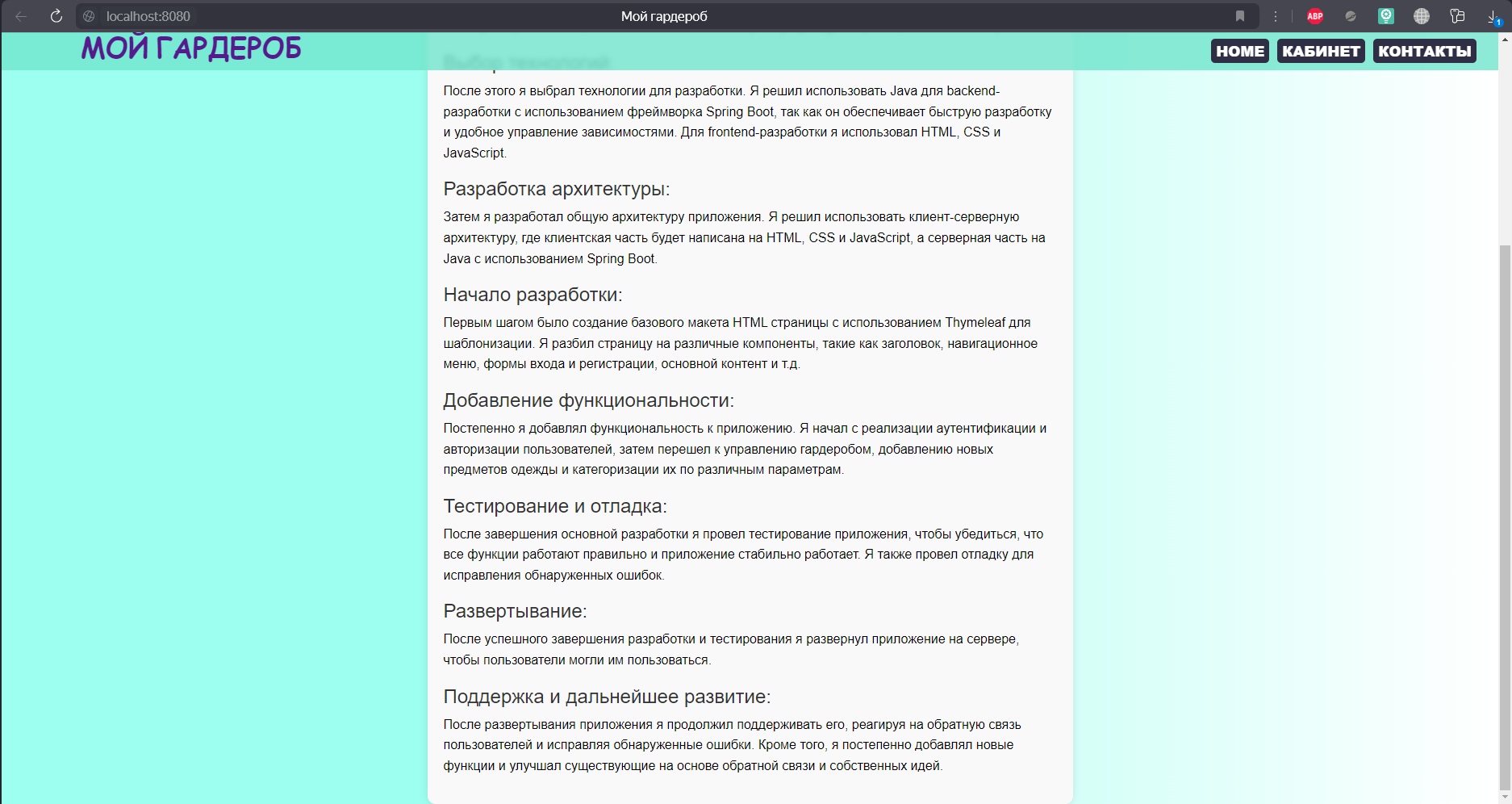
Habr. Внешние данные конфигурации в Spring. [электронный ресурс] - Режим доступа: https://habr.com/ru/companies/otus/articles/576910/ (Дата обращения: 10.06.2024).

Habr. Односторонние и двусторонние отношения в Hibernate. [электронный ресурс] - Режим доступа: https://habr.com/ru/articles/542328/ (Дата обращения: 10.06.2024).

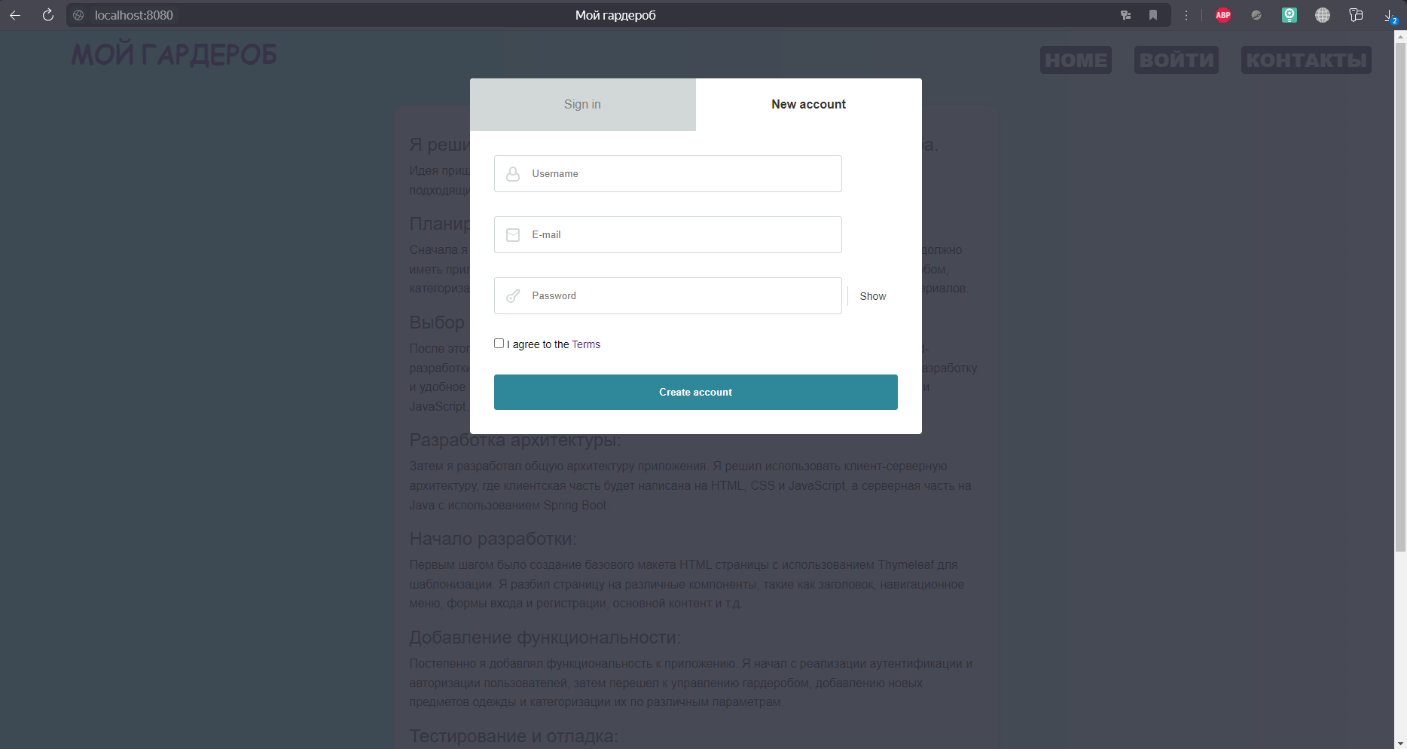
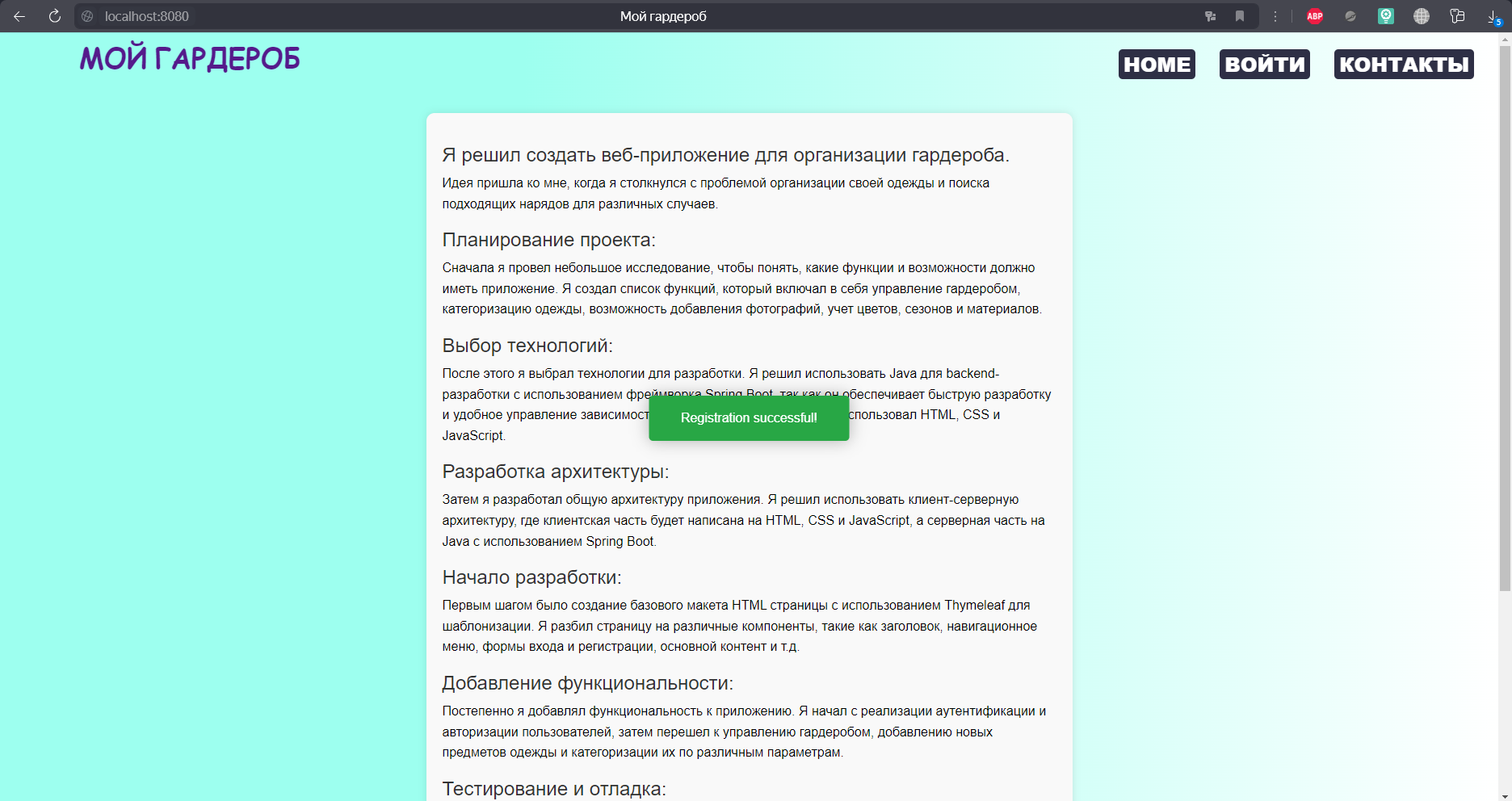
Приложение 1.



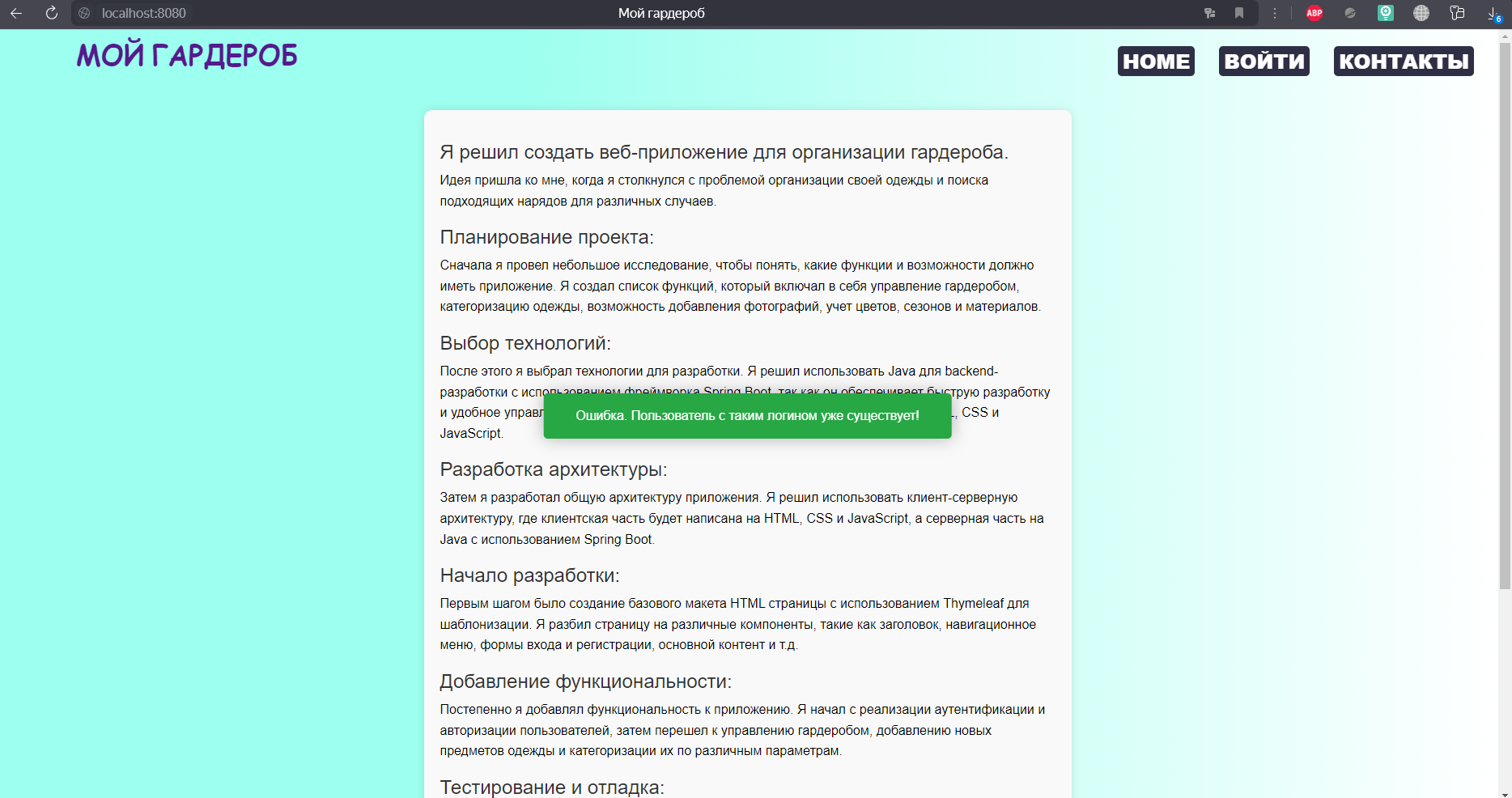
Приложение 2.

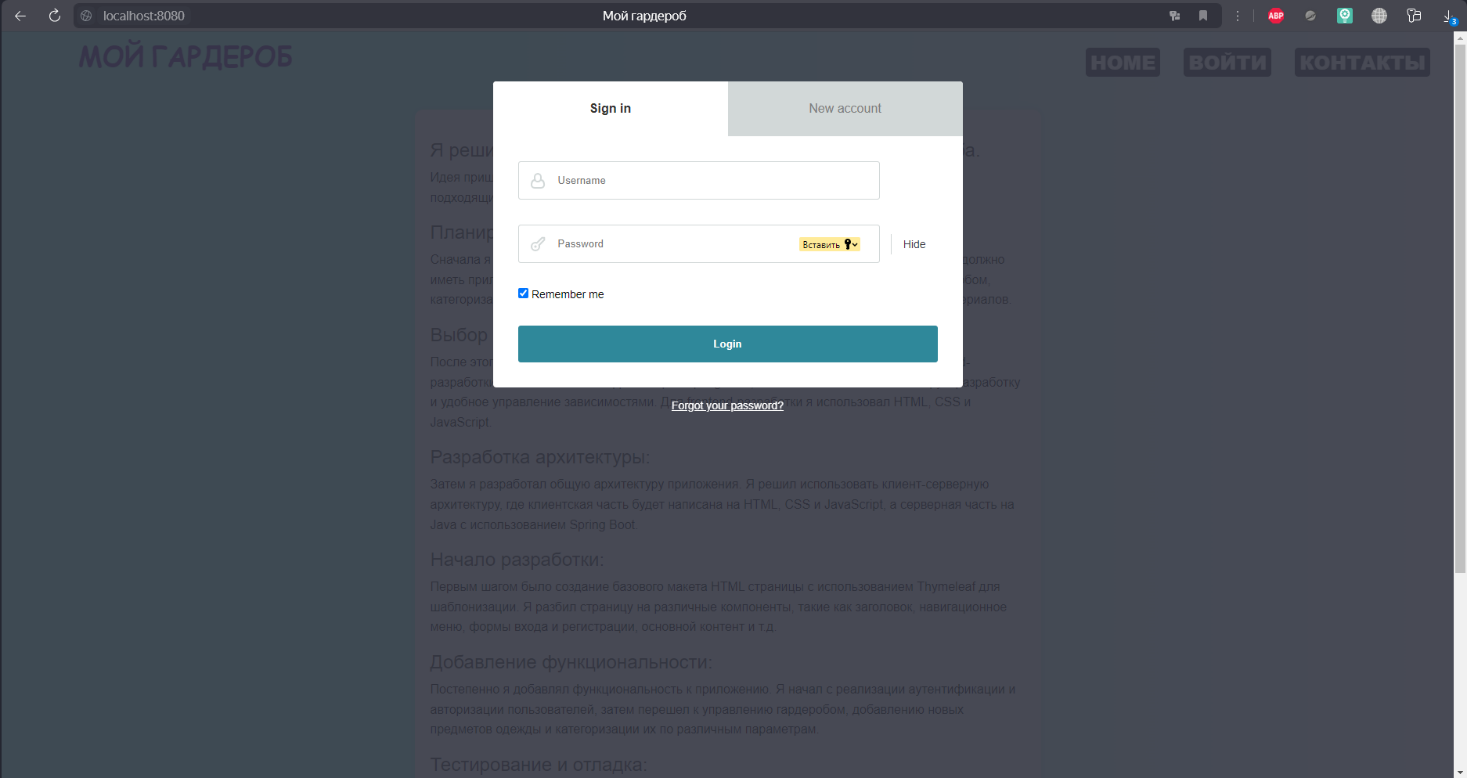


Приложение 3.

Приложение 4.



Приложение 5.

Приложение 6.

