Geekbrains

**Разработка веб приложения для записи заметок.**

**«Лучшая книга заметок»**

Программа:

Разработчик. Веб-разработка на Java

Нестеров Артем,

Логинова Кристина

Санкт-Петербург,

Казань

2024

**Оглавление**

Введение.......................................................................................................................3

Глава 1. Основы разработки веб-приложений..........................................................4

* 1. Основные компоненты веб-приложения.............................................................4
  2. Шаги разработки веб-приложения……………………………………………...5

1.3 Архитектура веб-приложений..............................................................................6

1.4 Основные компоненты Spring MVC....................................................................7

Глава 2. Разработка веб-приложения для заметок....................................................8

2.1.1 Порядок проведения работ.................................................................................8

2.1.2 Определение требований к приложению..........................................................8

2.2 Проектирование архитектуры веб-приложения................................................10

2.3 Компоненты приложения....................................................................................10

2.3.1 Model..................................................................................................................10

2.3.2 View....................................................................................................................13

2.3.3 Controller…........................................................................................................14

2.3.4 Repository...........................................................................................................16

2.3.5 Service….............................................................................................................17

2.4 Security..................................................................................................................19

Глава 3. Разворачивание Docker-контейнера...........................................................21

3.1 Основные шаги разворачивания Docker-контейнеров......................................21

Глава 4. Работа приложения в браузере……...........................................................23

4.1 Регистрация и вход в систему.............................................................................23

4.2 Главная страница……………..............................................................................24

Заключение.................................................................................................................25

Список литературы…................................................................................................26

**Введение**

В современном мире информация и знания играют ключевую роль в жизни каждого человека. Ежедневно мы сталкиваемся с огромным объемом информации, которую необходимо запомнить, систематизировать и хранить. Одним из наиболее эффективных способов управления этой информацией являются заметки. Заметки помогают фиксировать важные мысли, идеи, планы и задачи, облегчая их последующее использование.

**Цель проектной работы** – разработать веб-приложение для создания, хранения и управления заметками, которое будет предоставлять пользователям удобный и интуитивно понятный интерфейс, а также обеспечивать надежное хранение данных.

Основная функциональность приложения включает в себя:

1. **Создание и редактирование заметок**. Пользователь может легко создавать новые заметки, редактировать существующие записи и сохранять важную информацию. Интерфейс приложения интуитивно понятен и удобен для использования.
2. **Поиск**. В приложении реализована функция поиска, позволяющая быстро находить заметки по тегам.
3. **Безопасность и конфиденциальность**. Для защиты личных данных пользователей приложение обеспечивает возможность установки пароля и шифрования данных. Это гарантирует, что заметки останутся конфиденциальными и защищенными от несанкционированного доступа.
4. **Облачное хранение**. Все данные сохраняются в облаке, что обеспечивает доступ к заметкам с любого устройства в любое время. Это позволяет пользователям иметь под рукой все необходимые записи, независимо от местоположения.

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

* 1. **Основные компоненты веб-приложения**

Создание веб-приложений — это процесс разработки программного обеспечения, функционирующего на веб-серверах и доступного через интернет-браузеры. Этот процесс охватывает несколько ключевых аспектов и технологий. Рассмотрим основные концепции и этапы, которые необходимо учитывать при создании веб-приложений.

Основные компоненты веб-приложения включают:

1. **Клиентская сторона (Frontend)**:
   * **HTML**: Язык для структурирования веб-страниц.
   * **CSS**: Язык стилей для оформления и размещения элементов на веб-странице.
   * **JavaScript**: Скриптовый язык, добавляющий интерактивность и динамические элементы на веб-странице.
   * **Фреймворки и библиотеки**: Такие как React, Angular, Vue.js и другие, которые упрощают создание сложных пользовательских интерфейсов.
2. **Серверная сторона (Backend)**:
   * **Языки программирования**: Java, Python, PHP, Node.js и другие, используемые для разработки серверной логики.
   * **Фреймворки**: Spring, Django, Laravel, Express.js и другие, ускоряющие разработку серверной части.
   * **Серверы**: Apache, Nginx, Node.js и другие, обеспечивающие обработку клиентских запросов и отправку ответов.
3. **База данных**:
   * **СУБД**: MySQL, PostgreSQL, MongoDB, SQLite и другие, предназначенные для хранения данных.
   * **ORM (Object-Relational Mapping)**: Hibernate, Sequelize и другие, упрощающие работу с базой данных посредством объектно-реляционного отображения.
4. **API (Application Programming Interface)**:
   * **REST**: Принцип создания веб-сервисов.
   * **GraphQL**: Альтернатива REST, предоставляющая более гибкие запросы данных.

Эти компоненты работают вместе, обеспечивая функциональность и взаимодействие между различными частями веб-приложения.

* 1. **Шаги разработки веб-приложения**

**Порядок проведения работ.** В ходе создания проекта были проведены следующие виды работ:

1. **Планирование проекта**:
   * Определение функциональных требований к приложению.
   * Проектирование архитектуры приложения и выбор используемых технологий.
2. **Настройка окружения разработки**:
   * Установка и настройка Java Development Kit (JDK), Apache Maven (для сборки проекта), и Spring Boot (для создания и запуска приложения).
3. **Создание проекта**:
   * Создан новый проект с использованием Spring Initializr.
   * Выбраны необходимые зависимости, такие как Spring Web, Spring Data JPA, Lombok, Spring Security, Thymeleaf, MySQL Driver для поддержки функциональности приложения.
4. **Разработка архитектуры приложения**. Приложение построено по архитектуре Model-View-Controller (MVC) и включает следующие основные компоненты:
   * **Модель данных (Model)** – представляет собой структуру данных и логику работы с данными.
   * **Контроллеры (Controller)** – управляют запросами от пользователей и возвращают соответствующие представления.
   * **Сервисы (Service)** – содержат бизнес-логику приложения и взаимодействуют с репозиториями.
   * **Репозитории (Repository)** – обеспечивают доступ к базе данных.
   * **Представления (View)** – отображают данные пользователям и принимают ввод от них.
5. **Разработка пользовательского интерфейса**:
   * Создание HTML-шаблонов с использованием Thymeleaf (или других шаблонизаторов) для отображения данных и взаимодействия с пользователем.
6. **Отладка и оптимизация**:
   * Отладка кода для исправления ошибок и улучшения производительности.
   * Оптимизация кода и конфигурации приложения для улучшения скорости работы и эффективности использования ресурсов.
7. **Документирование**:
   * Создание документации по проекту, включая описание архитектуры, инструкции по установке и запуску и другие справочные материалы.
   1. **Архитектура веб-приложений**

Архитектура веб-приложений может отличаться в зависимости от масштаба, целей и требований проекта. Вот некоторые из наиболее распространенных архитектурных подходов:

1. **Монолитная архитектура**:
   * Вся логика приложения объединена в одном блоке.
   * Простота разработки и развертывания, но сложности с масштабированием и изменениями.
2. **Многоуровневая архитектура**:
   * Разделение на три уровня: презентационный, бизнес-логики и данных.
   * Улучшенная масштабируемость и поддержка, но увеличенные затраты на разработку.
3. **Микросервисная архитектура**:
   * Приложение разбито на независимые сервисы.
   * Масштабируемость и гибкость, но высокая сложность управления.
4. **Клиент-серверная архитектура**:
   * Отдельные клиентская и серверная части, взаимодействующие через сеть.
   * Четкое разделение обязанностей, но возможные проблемы с производительностью.
5. **Serverless архитектура**:
   * Код запускается без управления серверами.
   * Автоматическое масштабирование и упрощенное управление, но зависимость от поставщика услуг.
6. **Архитектура на основе событий**:
   * Компоненты взаимодействуют через события.
   * Высокая асинхронность и масштабируемость, но сложность разработки и отладки.
7. **Прогрессивные веб-приложения (PWA)**:
   * Веб-приложения, обеспечивающие аналогичный пользовательский опыт, как и нативные мобильные приложения.
   * Улучшенный пользовательский опыт и возможность работы офлайн, но ограниченные возможности по сравнению с нативными приложениями.

Выбор архитектуры зависит от множества факторов, таких как требования к масштабируемости, производительности и управляемости.

* 1. **Основные компоненты Spring MVC**

Spring MVC (Model-View-Controller) — это компонент фреймворка Spring, который разделяет логику приложения на модель, представление и контроллер, что упрощает организацию кода и поддержку приложения.

Основные компоненты Spring MVC:

1. **Model (Модель)**:
   * Обрабатывает бизнес-логику и управление данными приложения.
   * Включает объекты данных, которые манипулируются в приложении.
2. **View (Представление)**:
   * Отвечает за отображение данных пользователю.
   * Может быть реализовано с помощью различных технологий, таких как JSP, Thymeleaf и FreeMarker.
3. **Controller (Контроллер)**:
   * Обрабатывает пользовательские запросы, взаимодействует с моделью и выбирает представление для отображения данных.
   * Аннотируется с помощью @Controller или @RestController.

Процесс работы Spring MVC:

* Пользователь отправляет HTTP-запрос на сервер.
* DispatcherServlet принимает запрос и направляет его в соответствующий контроллер.
* Контроллер обрабатывает запрос, взаимодействует с моделью и выбирает представление.
* Данные из модели передаются в представление, которое генерирует ответ для пользователя.

Spring MVC помогает разделить обязанности в приложении, что упрощает разработку, тестирование и сопровождение, улучшая модульность и повторное использование кода.

**Глава 2. Разработка веб-приложения для заметок**

**2.1.1 Порядок проведения работ**

Процесс разработки приложения включает следующие этапы:

1. Определение требований к приложению.
2. Проектирование архитектуры и компонентов приложения.
3. Подготовка инструментов и среды разработки.
4. Написание кода для реализации требований.
5. Отладка и оптимизация.

**2.1.2 Определение требований к приложению**

Для разработки приложения «Лучшая книга заметок» были определены следующие требования:

**Функциональные требования:**

1. Пользователи должны иметь возможность создавать заметки и управлять ими.
2. Приложение должно предоставлять возможность поиска заметок по тегам.
3. Пользователи должны иметь возможность устанавливать пароли для защиты своих данных.
4. Возможность просмотра таблицы всех заметок с основной информацией о них (дата создания, теги, содержание).
5. Возможность просмотра профилей пользователей приложения.

**Нефункциональные требования:**

1. Удобство использования: Интерфейс должен быть интуитивно понятным и привлекательным для пользователя.
2. Безопасность: Данные пользователей должны храниться и передаваться в зашифрованном виде. Аутентификация и авторизация пользователей.
3. Производительность: Приложение должно обеспечивать быстрый доступ к данным и отзывчивый интерфейс.
4. Масштабируемость: Возможность расширения функциональности и увеличения числа пользователей без значительного ухудшения производительности.
5. Интерфейс: Простой, дружелюбный и понятный интерфейс.

**Технологии:**

Для реализации приложения «Лучшая книга заметок» используются следующие технологии:

Язык программирования: Java

Фреймворк: Spring

База данных: MySQL

Шаблонизатор: Thymeleaf

Обеспечение безопасности: Spring Security

Среда разработки: IntelliJ IDEA

В этом разделе мы рассмотрели порядок работы над приложением и определили ключевые требования. Эти шаги обеспечат успешное выполнение проекта и создание приложения, отвечающего всем функциональным и нефункциональным требованиям.

**2.2 Проектирование архитектуры веб-приложения**

Для данного веб-приложения выбрана монолитная архитектура, которая идеально подходит для небольших и средних проектов. Используя паттерн MVC, приложение разделяется на несколько ключевых компонентов:

1. **Model** – включает классы Role, User, и Note, которые представляют собой основные сущности приложения.
2. **View** – представление данных осуществляется с помощью шаблонизатора Thymeleaf, который позволяет интегрировать данные из контроллеров в HTML-шаблоны.
3. **Controller** – в приложении реализованы два контроллера: NoteController и UserController, которые управляют обработкой запросов и взаимодействием с пользователем.
4. **Repository** – используются три репозитория: RoleRepository, UserRepository и NoteRepository, обеспечивающие доступ к данным и их управление.
5. **Service** – бизнес-логику приложения реализуют классы NoteService и UserService, которые обрабатывают данные и выполняют необходимые операции.

**2.3 Компоненты приложения**

**2.3.1 Model**:

Класс Note (рис. 1) содержит информацию о заметке, такую как идентификатор заметки, заголовок, содержание, дату создания, теги и пользователь, которому принадлежит заметка. Это позволяет организовать и управлять заметками пользователей.

**Поля класса Note**:

* id (Long): уникальный идентификатор заметки.
* title (String): заголовок заметки.
* content (String): содержание заметки.
* createdDate (LocalDateTime): дата и время создания заметки.
* tags (Set<String>): набор тегов, присвоенных заметке для облегчения поиска.
* user (User): пользователь, создавший заметку.

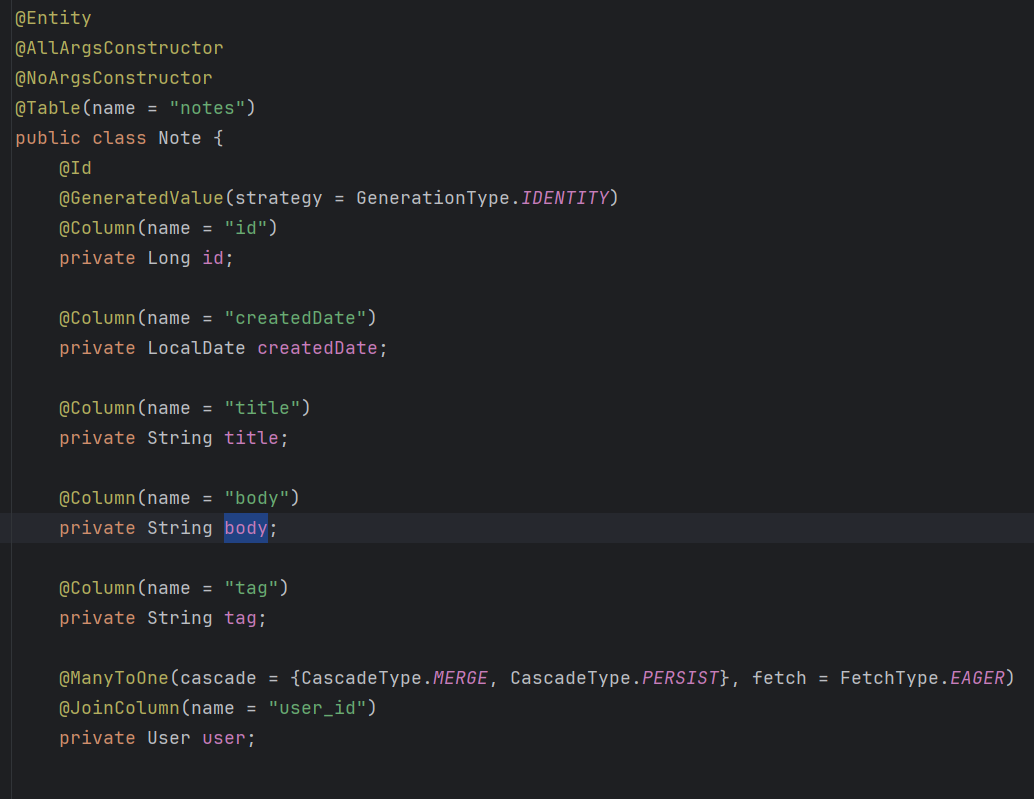


Рисунок 1

Класс User (рис. 2) содержит информацию о пользователе, такую как идентификатор пользователя, имя, электронную почту, пароль и роли. Это позволяет управлять учетными записями пользователей и их правами доступа.

**Поля класса User**:

* id (Long): уникальный идентификатор пользователя.
* username (String): имя пользователя.
* email (String): электронная почта пользователя.
* password (String): зашифрованный пароль пользователя.
* roles (Set<Role>): набор ролей, присвоенных пользователю.

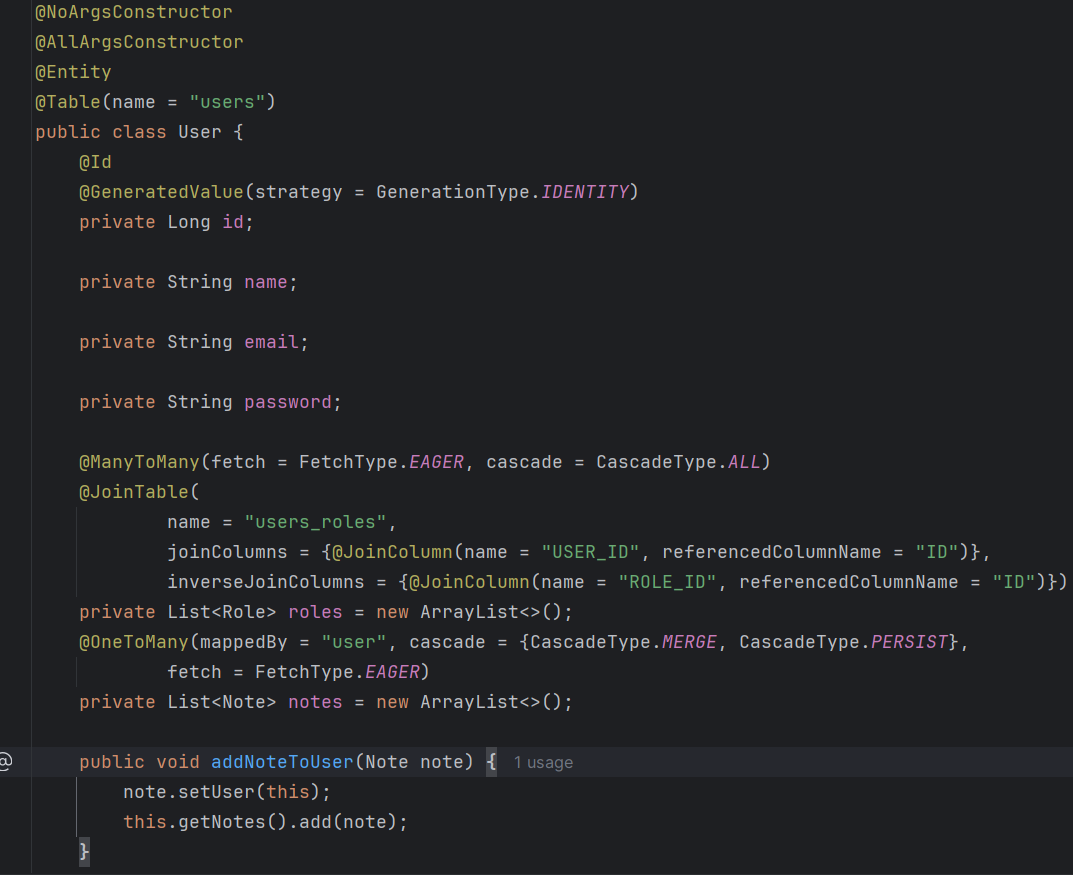


Рисунок 2

Класс Role (рис. 3) содержит информацию о роли пользователя, такую как идентификатор роли и название роли. Это позволяет управлять правами доступа пользователей.

**Поля класса Role**:

* id (Long): уникальный идентификатор роли.
* name (String): название роли

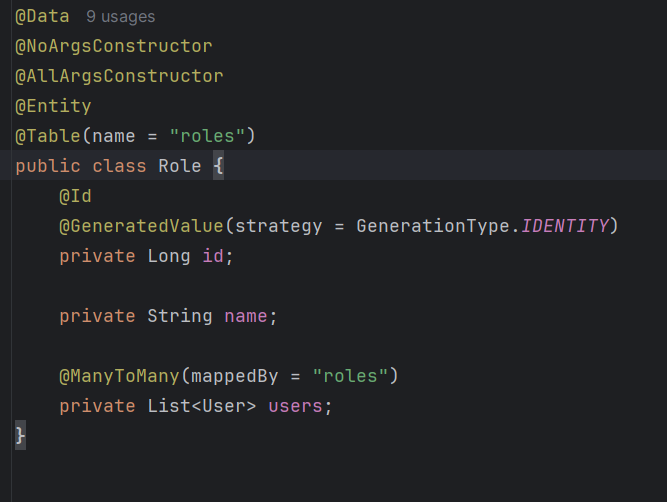


Рисунок 3

**2.3.2 View**:

Использование Thymeleaf для создания динамических страниц, на которых отображаются данные из контроллеров.

**2.3.3 Controller**:

Класс MyController (рис. 4) управляет запросами, связанными с созданием, редактированием и удалением заметок.

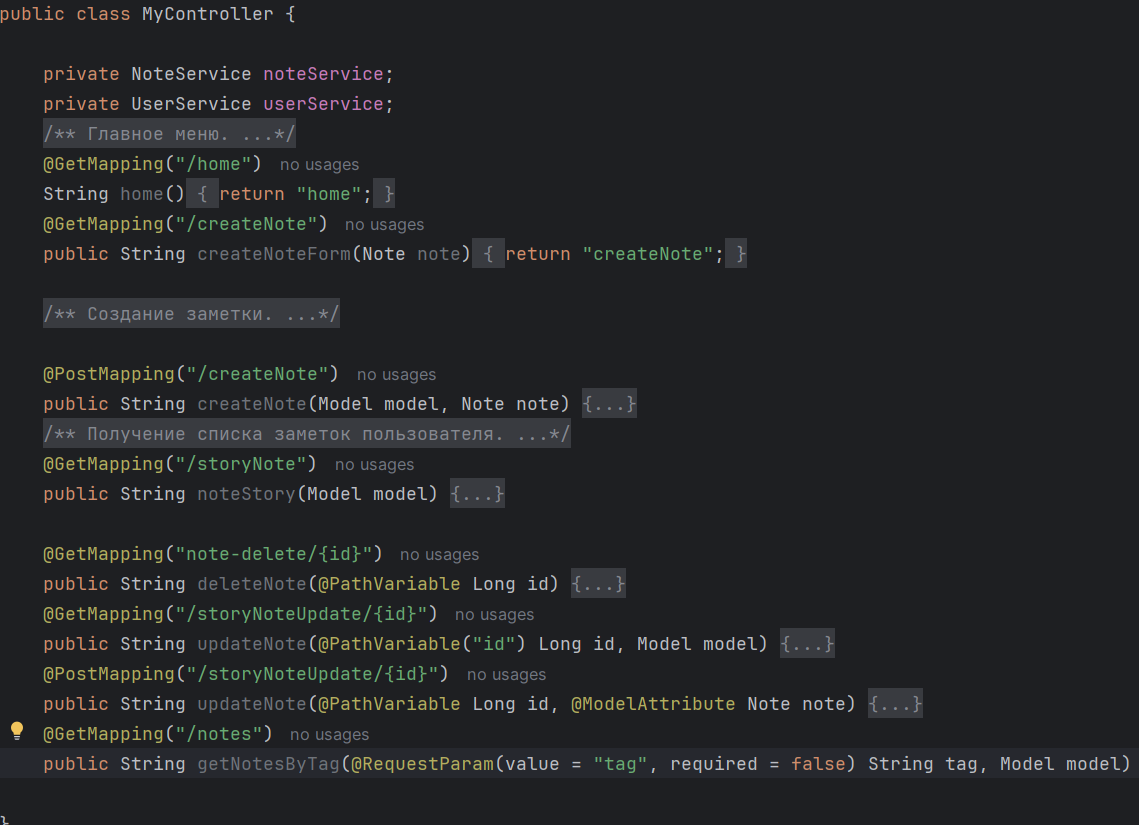


Рисунок 4

Класс UserController (рис. 5) обрабатывает запросы, связанные с управлением пользователями их регистрацией и аутентификацией.

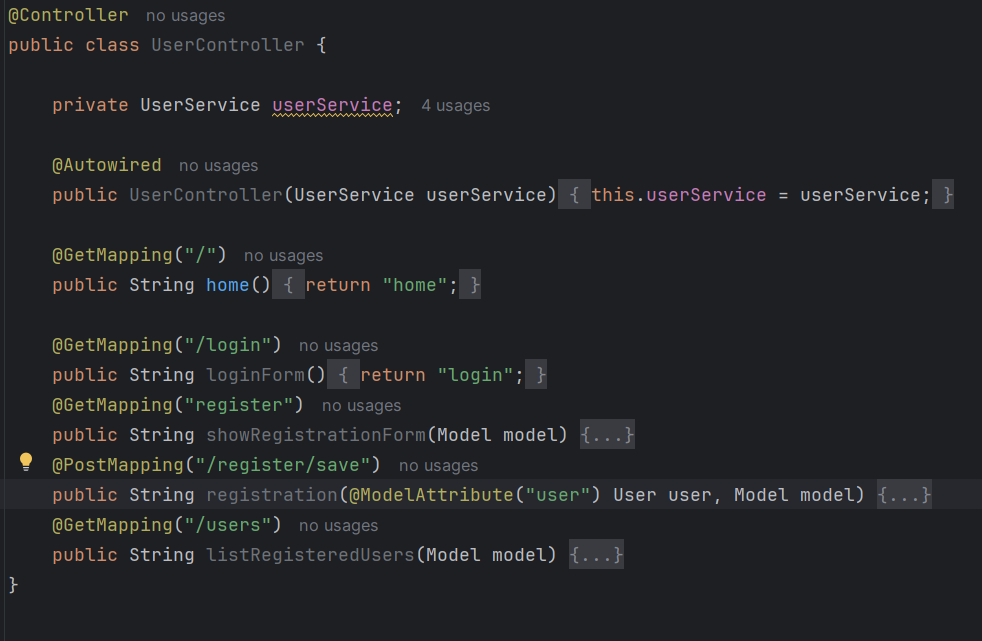


Рисунок 5

**2.3.4 Repository**:

Класс RoleRepository (рис. 6) обеспечивает доступ к данным о ролях пользователей.

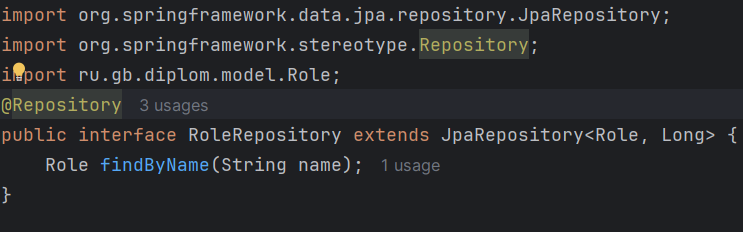


Рисунок 6

Класс UserRepository (рис. 7) отвечает за доступ и управление данными о пользователях.

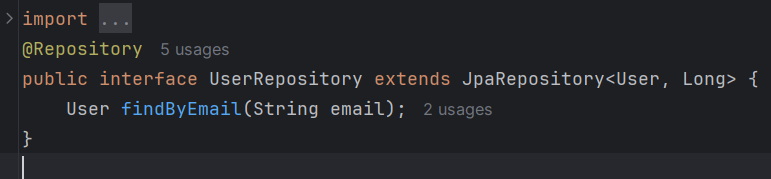


Рисунок 7

Класс NoteRepository (рис. 8) отвечает за доступ и управление данными заметок.

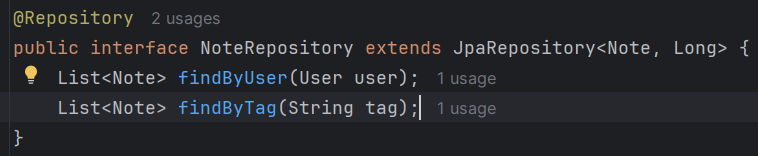


Рисунок 8

**2.3.5 Service:**

Класс NoteService (рис. 9) содержит бизнес-логику для работы с заметками.



Рисунок 9

Класс UserService (рис. 10) реализует бизнес-логику, связанную с пользователями.

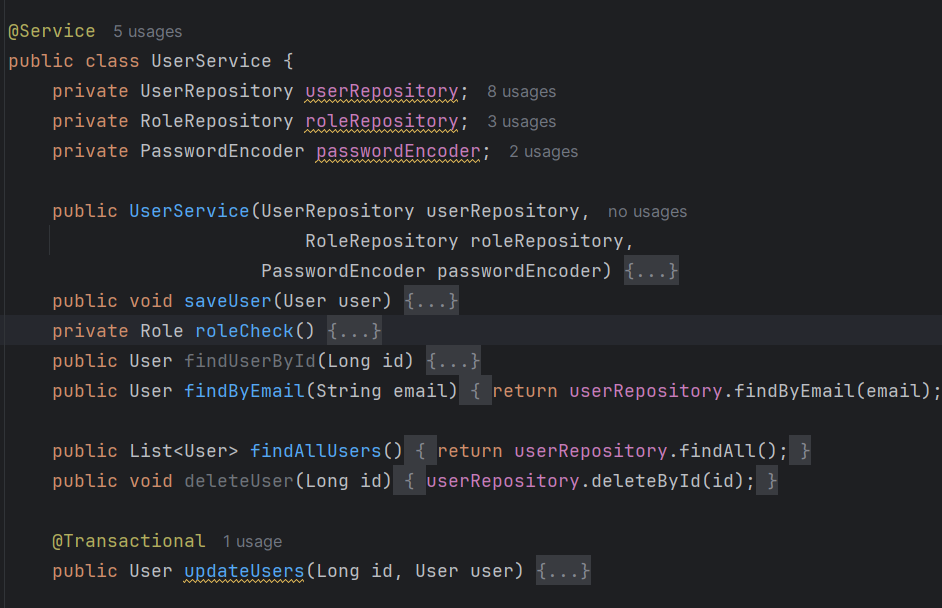


Рисунок 10

**2.4 Security**

Для обеспечения безопасности в приложении используется Spring Security.

Класс CustomUserDetailsService (рис. 11) предназначен для интеграции с Spring Security и обеспечивает аутентификацию пользователей на основе данных, хранящихся в базе данных. Этот класс загружает пользователя по его имени (логину), преобразует его роли в формат, который понимает Spring Security, и возвращает объект UserDetails, который используется для проверки аутентификационных данных и авторизации пользователя

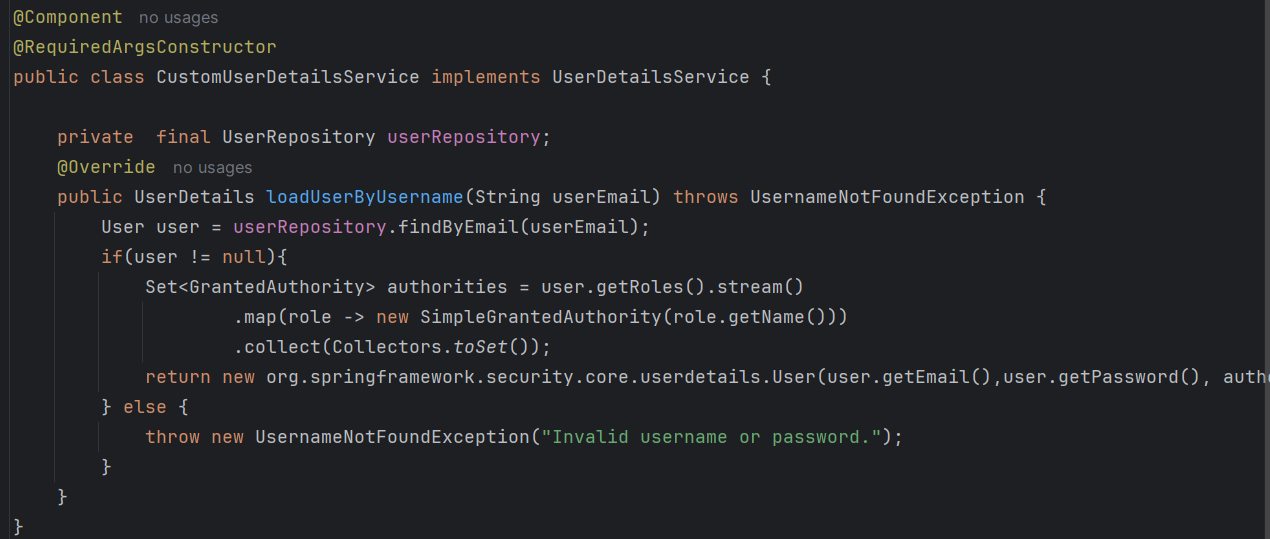


Рисунок 11

Класс CustomUserDetailsService реализует интерфейс UserDetailsService из Spring Security. Он загружает пользователя по его логину (email), преобразует его роли в формат, понятный Spring Security, и возвращает объект UserDetails, который используется для проверки аутентификации и авторизации (рис. 12)

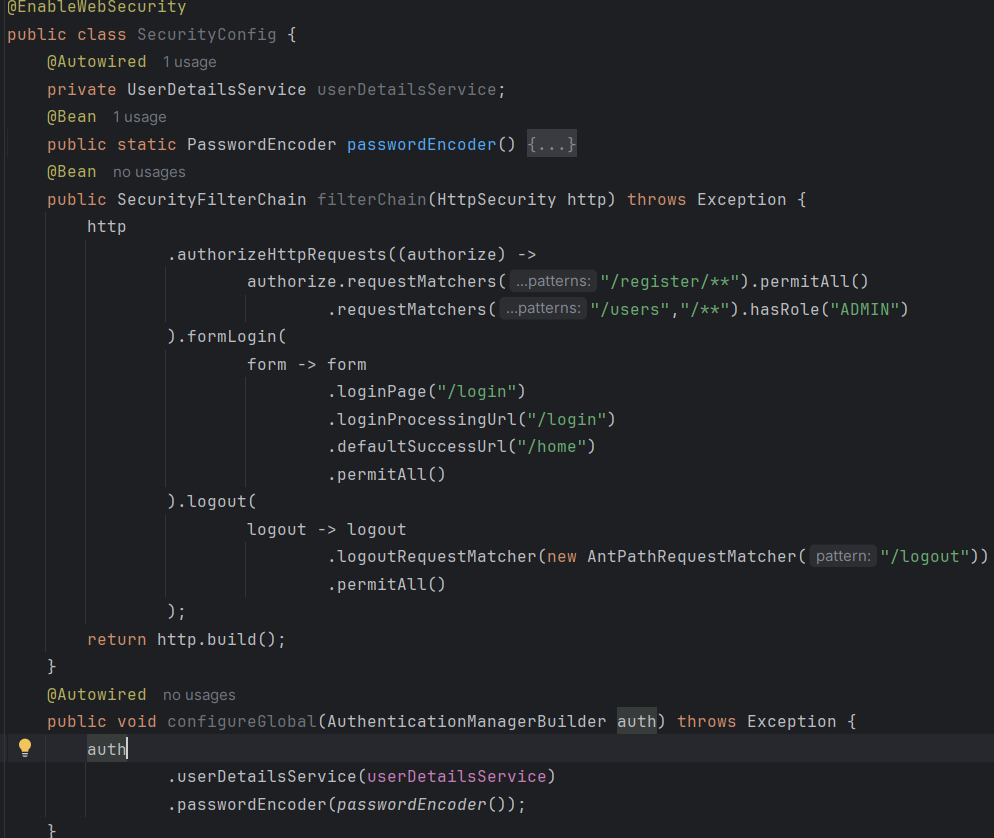


Рисунок 12

В этом разделе было рассмотрено проектирование архитектуры веб-приложения "Лучшая книга заметок". Выбор монолитной архитектуры и использование паттерна MVC позволяет создать легко управляемое и расширяемое приложение, которое отвечает всем требованиям и обеспечивает удобство работы как для разработчиков, так и для пользователей.

**Глава 3. Разворачивание Docker-контейнера**

Разворачивание Docker-контейнеров включает процесс запуска и управления контейнерами на целевом хосте. Для этого используется Docker-демон, который управляет жизненным циклом контейнеров, такими как создание, запуск, остановка и удаление.

**3.1 Основные шаги разворачивания Docker-контейнеров:**

1. **Установка Docker**:
   * Сначала необходимо установить Docker на целевом хосте. Инструкции по установке для различных операционных систем можно найти на официальном сайте Docker.
2. **Создание Docker-образа**:
   * Docker-образ представляет собой шаблон, на основе которого создается контейнер. Образ содержит все необходимые зависимости и конфигурации приложения. Образ можно создать с помощью Dockerfile, который описывает шаги по сборке образа.
3. **Создание контейнера**:
   * После создания образа необходимо создать контейнер на его основе. Это делается с помощью команды docker run. Например, для запуска контейнера на основе созданного образа:

bash

Копировать код

docker run --name diplom-mysql -e MYSQL\_ROOT\_PASSWORD=admin -e MYSQL\_DATABASE=diplom\_db -d -p 3307:3306 mysql

Команда docker run используется для создания и запуска контейнера на основе Docker-образа. Давайте разберем каждый параметр данной команды:

* --name diplom-mysql: Устанавливает имя контейнера как diplom-mysql. Это удобно для идентификации контейнера при его управлении.
* -e MYSQL\_ROOT\_PASSWORD=admin: Устанавливает переменную окружения MYSQL\_ROOT\_PASSWORD со значением admin для задания пароля для пользователя root в базе данных MySQL.
* -e MYSQL\_DATABASE=diplom\_db: Устанавливает переменную окружения MYSQL\_DATABASE со значением diplom\_db для создания базы данных с именем diplom\_db в MySQL.
* -d: Запускает контейнер в фоновом режиме (detached mode).
* -p 3307:3306: Пробрасывает порт 3307 из контейнера на хостовую машину. Первое число (3307) - порт на хостовой машине, второе число (3306) - порт в контейнере, на котором работает MySQL.
* mysql: Имя Docker-образа, на основе которого будет создан контейнер. В данном случае, это официальный образ MySQL из Docker Hub.

После выполнения этой команды будет создан и запущен контейнер с именем diplom-mysql, в котором будет работать MySQL с базой данных diplom\_db и паролем admin, пробрасывая порт 3307 для доступа к базе данных извне контейнера.

**Глава 4. Работа приложения в браузере**

**4.1 Регистрация и вход в систему**

Для отображения данных приложения был использован HTML и разработаны соответствующие файлы. Для страницы авторизации (Рис.13) – «login.html».

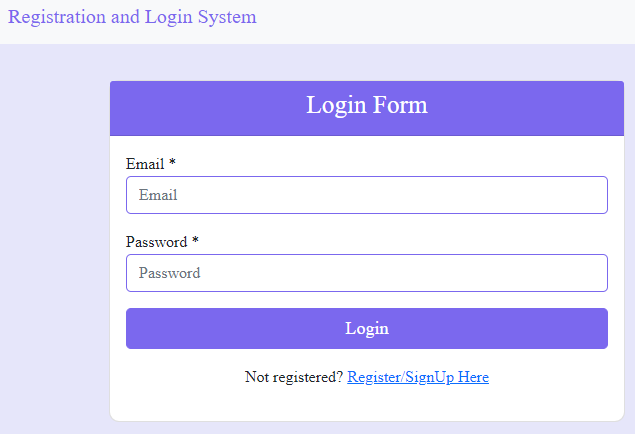


Рисунок 13

Для страницы регистрации (Рис. 14) – «register.html».

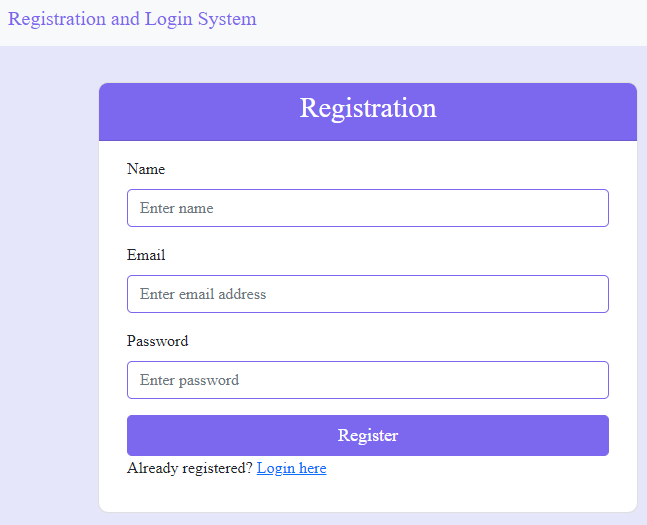
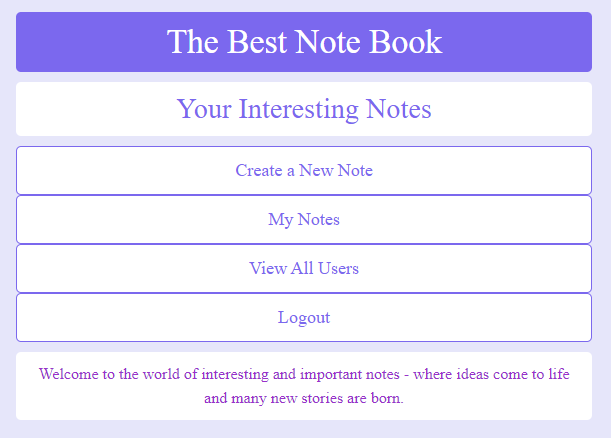


Рисунок 14

**4.2 Главная страница**

Для главной страницы (Рис. 15) – «home.html»

****

**Заключение**

В рамках данного дипломного проекта было разработано веб-приложение "Лучшая книга заметок", предназначенное для создания, хранения и управления заметками пользователей. Проект был реализован с использованием современных технологий веб-разработки, таких как Java Spring Framework, Thymeleaf, Bootstrap и MySQL.

В процессе разработки были достигнуты следующие цели:

* Создание удобного и интуитивно понятного интерфейса для пользователей приложения.
* Реализация функционала добавления, редактирования, поиска и удаления заметок.
* Обеспечение безопасности данных пользователей с использованием механизмов аутентификации и авторизации Spring Security.
* Использование принципов SOLID и паттернов проектирования для обеспечения гибкости и расширяемости кода.

Дальнейшее развитие приложения может включать в себя добавление нового функционала, такого как возможность совместной работы над заметками, интеграция с календарем для планирования задач, а также мобильное приложение для удобного доступа к заметкам из любой точки.

В целом, дипломный проект "Лучшая книга заметок" позволил не только показать навыки разработки веб-приложений с использованием современных технологий, но и продемонстрировать понимание принципов безопасности и удобства использования для конечного пользователя.

**Список литературы**

* 1. Лауренциу Спилкэ. Spring быстро. Санкт-Петербург: Питер, 2023.
  2. Евгений Борисов. Spring-потрошитель, часть 1 [Видео]. URL: https://www.youtube.com/watch?v=BmBr5diz8WA.
  3. Евгений Борисов. Spring-потрошитель, часть 1 [Видео]. URL: https://www.youtube.com/watch?v=cou\_qomYLNU.
  4. Евгений Борисов. Spring-построитель [Видео]. URL: https://www.youtube.com/watch?v=rd6wxPzXQvo.
  5. Крейг Уоллс. Spring в действии. Москва: ДМК Пресс, 2022.
  6. Alishev. Spring Framework [Видео]. URL: <https://www.youtube.com/watch?v=5ePo08sqcpk&list=PLAma_mKffTOR5o0WNHnY0mTjKxnCgSXrZ>
  7. Herbert Schildt. Java 12-е издание «Полное руководство», 2022