Geekbrains

**Разработка веб приложения «Менеджер задач» на JavaSpring.**

Разработчик

Программист

Веб-разработка на Java

Дятчин С.А.

Москва

2025

**Дипломный проект**

**«Разработка веб приложения «Менеджер задач» на JavaSpring»**

**Содержание**

|  |  |
| --- | --- |
| Введение | 4 |
| Глава 1. Предпроектная стадия | 6 |
| 1.1.Разработка java приложений с Spring Framework | 6 |
| 1.2. Шаги создания проекта Spring Framework | 7 |
| 1.3. Основные сложности при разработке на Spring Framework | 9 |
| 1.4. Итоги выбора технологии Spring Framework | 10 |
| 1.5. Архитектура приложения | 11 |
| 1.6. Apache Maven | 12 |
| 1.7. База данных | 13 |
| 1.8. View | 13 |
| Глава 2. Проектирование и разработка | 15 |
| 2.1. Порядок проведения работ | 15 |
| 2.2. определение требований к приложению | 15 |
| 2.3. Выбор стека технологий и архитектуры приложения | 16 |
| 2.4. Настройка окружения | 17 |
| 2.5. Создание проекта и выбор зависимостей | 17 |
| 2.6. Разработка модели данных | 18 |
| 2.7. Разработка сервисного слоя | 24 |
| 2.8. Внедрение безопасности | 31 |
| 2.9 Разработка контролеров | 32 |
| 2.10 Разработка пользовательского интерфейса | 37 |
| Заключение | 59 |
| Список использованной литературы | 60 |

**Введение**

В современном мире, где количество дел и обязанностей постоянно растет, менеджеры задач становятся незаменимыми помощниками. Эти инструменты позволяют не только структурировать работу, но и обеспечивают контроль над выполнением задач, что особенно важно в условиях многозадачности. Менеджеры задач могут быть использованы в различных сферах: от личных дел и учебы до сложных профессиональных проектов. Они помогают пользователям не только планировать свои действия, но и отслеживать прогресс, что способствует достижению поставленных целей. В условиях, когда время является одним из самых ценных ресурсов, умение эффективно его использовать становится критически важным. Менеджеры задач предоставляют все необходимые инструменты для этого.

**Основные функции и возможности проекта:**

**Создание и управление задачами** Менеджер задач позволяет создавать задачи, устанавливать сроки выполнения и назначать ответственных. Это помогает структурировать работу и не забывать о важных делах.

**Приоритизация задач по срокам.** С помощью менеджера задач можно расставлять приоритеты, что позволяет сосредоточиться на наиболее важных и срочных делах. Это особенно полезно в условиях ограниченного времени. Приоритизация задач помогает избежать ситуаций, когда менее важные дела занимают все время, оставляя критически важные задачи без внимания. Например, в рабочей среде это может помочь сосредоточиться на проектах с ближайшими дедлайнами.

**Совместная работа.** Менеджер задач поддерживает функции совместной работы. Совместная работа в менеджерах задач позволяет распределять обязанности, отслеживать прогресс каждого участника и обеспечивать прозрачность выполнения задач. Например, в проектной команде можно назначить ответственных за различные этапы работы и следить за их выполнением в реальном времени.

**Преимущества использования менеджеров задач:**

Использование менеджеров задач имеет множество преимуществ, которые делают их незаменимыми инструментами в повседневной жизни и работе.

**Повышение продуктивности.** Менеджеры задач помогают организовать работу и сосредоточиться на важных делах, что повышает продуктивность. Они позволяют избежать хаоса и неразберихи в делах. Организованная работа способствует более эффективному использованию времени и ресурсов, что в конечном итоге приводит к лучшим результатам.

**Экономия времени.** С помощью менеджеров задач можно эффективно планировать свое время и избегать ненужных задержек. Это позволяет быстрее и качественнее выполнять задачи. Экономия времени достигается за счет автоматизации рутинных процессов и более четкого планирования. Например, в бизнесе это может помочь сократить время на выполнение проектов и увеличить прибыль.

**Улучшение коммуникации.** Менеджеры задач с функциями совместной работы способствуют улучшению коммуникации внутри команды. Это помогает избежать недопонимания и ошибок в работе. Улучшение коммуникации достигается за счет прозрачности выполнения задач и возможности обмена информацией в реальном времени. Например, в проектной команде это может помочь избежать дублирования усилий и улучшить координацию действий.

В данном проекте я участвовал в роли Java-разработчика. Также на проекте я был задействован как дизайнер и тестировщик.

**Глава 1. Предпроектная проработка**

* 1. **Разработка java приложений с Spring Framework**

Spring Framework является одним из самых популярных и мощных фреймворков для разработки веб-приложений на языке Java. Он предоставляет разработчикам множество инструментов и решений, упрощающих процесс создания приложений. В данной статье мы рассмотрим основные принципы и преимущества разработки приложений с использованием Spring Framework.

Spring Framework работает на принципе инверсии управления (IoC), где объекты создаются, настраиваются и связываются между собой контейнером Spring. Это позволяет создавать слабо связанные компоненты приложения, что способствует повышению его гибкости и поддерживаемости. Фреймворк также предлагает ряд расширений, таких как Spring MVC, Spring Data и Spring Security, которые упрощают разработку различных слоев веб-приложений.

Spring Framework Spring Framework - это популярный фреймворк для разработки Java-приложений. Он предоставляет множество функциональных возможностей, таких как управление зависимостями, инверсия управления, аутентификация и авторизация, обработка HTTP-запросов и многое другое. Spring Framework позволяет разработчикам создавать масштабируемые и надежные приложения, сокращая количество необходимого кода и повышая производительность разработки.

Spring Boot Spring Boot - это инструмент на основе Spring Framework, который упрощает процесс разработки приложений на Java. Он предлагает автоматическую конфигурацию и управление зависимостями, что позволяет разработчикам сосредоточиться на написании бизнес-логики приложения, а не на настройке среды разработки. Spring Boot также включает в себя встроенный сервер приложений, что позволяет легко запускать и развертывать приложения без дополнительных настроек.

Для разработки java приложений с использованием Spring Framework необходимо выполнить несколько шагов.

* 1. **Шаги создания проекта Spring Framework**

**1.2.1. Создание проекта**

Первым шагом является создание нового проекта с использованием среды разработки, такой как IntelliJ IDEA или Eclipse. Необходимо выбрать тип проекта "Maven" или "Gradle" и добавить зависимости для Spring Framework и других модулей, которые вы планируете использовать.

**1.2.2. Конфигурация Spring контейнера**

Spring Framework основан на конфигурации через XML, аннотации или Java-код. Конфигурация определяет, какие компоненты создаются и как они взаимодействуют между собой. Вы можете указать путь к конфигурационному файлу в файле web.xml или использовать аннотации,такие как @Configuration и @ComponentScan, для определения конфигурации в Java-коде.

**1.2.3. Создание бинов**

Бины являются основными строительными блоками Spring Framework. Они представляют собой объекты, которые создаются и настраиваются контейнером Spring. Вы можете определить бины в конфигурационном файле XML или с помощью аннотаций, таких как @Component или @Service. Бины могут иметь зависимости друг от друга, которые автоматически разрешаются контейнером Spring.**1.2.4. Инжектирование зависимостей**

Инъекция зависимостей - это механизм, позволяющий автоматически связывать компоненты друг с другом. Spring Framework предоставляет несколько способов инъекции зависимостей, включая конструктор, сеттеры и аннотацию @Autowired. Это позволяет уменьшить связность между компонентами и повысить их переиспользуемость и тестируемость.

**1.2.5. Создание контроллеров**

Spring MVC (Model-View-Controller) является частью Spring Framework, предназначенной для разработки веб-приложений. Контроллеры обрабатывают входящие HTTP-запросы и возвращают соответствующие HTTP-ответы. Вы можете использовать аннотации, такие как @Controller и @RequestMapping, для определения контроллеров и их обработчиков запросов.

**1.2.6. Создание представлений**

Представления являются частью Spring MVC и отвечают за отображение данных, возвращаемых контроллерами, на веб-страницах. Представления могут быть созданы с использованием шаблонов JSP, Thymeleaf, Velocity или других поддерживаемых технологий. Вы можете связать модель данных с представлением с помощью аннотации @ModelAttribute или объекта ModelAndView.

**1.2.7. Настройка базы данных**

Spring Framework предлагает модуль Spring Data, который упрощает взаимодействие с базами данных. Вы можете настроить доступ к базе данных, используя аннотации, такие как @Entity и @Repository, а также определить репозитории для выполнения операций с данными. Spring поддерживает различные базы данных, такие как MySQL, PostgreSQL или MongoDB.

**1.2.8. Обработка исключений и аутентификация**

Spring Framework предоставляет инструменты для обработки исключений и авторизации пользователей. Вы можете определить обработчики исключений с помощью аннотации @ExceptionHandler, чтобы обрабатывать исключения, возникающие в процессе выполнения приложения. Кроме того, вы можете использовать Spring Security для настройки аутентификации и авторизации пользователей в веб-приложении.

**1.2.9. Тестирование приложения**

Spring Framework поддерживает модуль Spring Test, который предоставляет инструменты для написания и запуска автоматических тестов. Вы можете использовать аннотацию @RunWith(SpringRunner.class) для запуска тестов в контексте Spring. Spring также предлагает множество мок-объектов, инструментов для упрощения разработки и тестирования.

**1.2.10. Развертывание приложения**

Последний шаг - развертывание приложения на сервере. Вы можете использовать контейнер сервлетов, такой как Tomcat или Jetty, или облачные платформы, такие как Heroku или AWS, для развертывания приложения.Spring Framework предлагает различные инструменты для упрощения и автоматизации развертывания веб-приложений.

**1.3 Основные сложности при разработке на Spring Framework**

**1.3.1 Сложность конфигурации**

Одной из основных проблем при разработке java приложений с использованием Spring Framework является сложность конфигурации. Spring Framework предлагает широкий набор модулей и функциональность, которые требуют правильной настройки и интеграции. Некорректная конфигурация может привести к ошибкам во время выполнения или неправильной работе приложения. Кроме того, из-за большого количества возможностей Spring Framework может быть сложно определить, какие именно модули и функциональность использовать в конкретном проекте.

**1.3.2. Сложность обучения**

Еще одной проблемой при разработке java приложений с Spring Framework является сложность обучения. Spring Framework имеет большой объем документации и множество различных модулей, что требует времени и усилий для овладения этими знаниями. Кроме того, множество функциональных возможностей Spring Framework может вызывать путаницу у новичков, их может смутить необходимость изучения концепций и практик, которые не являются стандартными для разработки java приложений.

**1.3.3. Сложность внедрения зависимостей**

Еще одной проблемой при разработке java приложений с Spring Framework является сложность внедрения зависимостей. Spring Framework предлагает мощный механизм внедрения зависимостей, который позволяет упростить управление зависимостями в приложении. Однако, для использования этого механизма требуется настроить правильные метаданные и установить правильные связи между компонентами, что может быть сложно для начинающих разработчиков. Кроме того, неправильное внедрение зависимостей может привести к ошибкам во время выполнения или неправильной работе приложения.

**1.4 Итоги выбора технологии Spring Framework**

Spring Framework является мощным и гибким инструментом для разработки веб-приложений на языке Java. Он позволяет создавать высококачественные и масштабируемые приложения с использованием принципов инверсии управления и внедрения зависимостей. Благодаря широкому набору модулей и инструментов, Spring Framework упрощает иускоряет процесс разработки и обеспечивает высокую производительность и надежность веб-приложений.

**1.5 Архитектура приложения**

**Архитектура MVC (Model-View-Controller)** — это схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер.

****

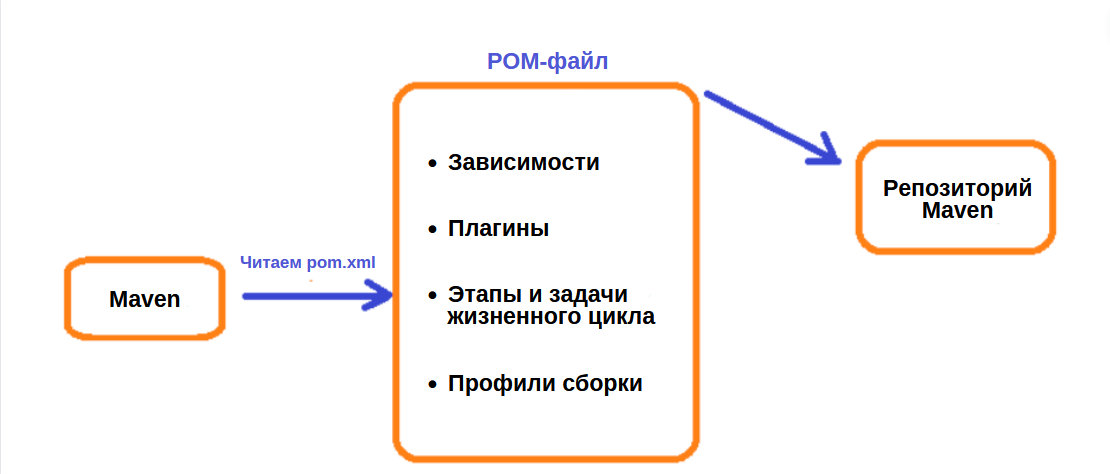
**Модель (Model)** предоставляет данные и реагирует на команды контроллера, изменяя своё состояние. Например, в системе бронирования модель может обрабатывать пользовательские данные, детали бронирования и расчёты цен.

**Представление (View)** отвечает за отображение данных модели пользователю, реагируя на изменения модели. Например, на свёрстанной странице сайта или в приложении на телефоне.

**Контроллер (Controller)** интерпретирует действия пользователя, оповещая модель о необходимости изменений. Например, если пользователь нажимает кнопку «Удалить заказ», то контроллер отмечает этот заказ в модели удалённым.

Этот шаблон обычно используется при разработке программного обеспечения для создания, организованного и простого в обслуживании кода.

**1.6. Apache Maven**

Maven – это система сборки, и его задача аналогична Ant – такой же выдающейся системы сборки. Это инструмент управления проектами по разработке программного обеспечения, который привнес в нашу жизнь новую концепцию объектной модели проекта (POM).

 Maven упрощает и стандартизирует процесс сборки проекта. Он позволяет работать совместно над одной задачей и над отдельными задачами, а также с легкостью решать вопросы, связанные с компиляцией, распространением и документацией. Maven увеличивает шансы на повторное использование, а также учитывает большое количество задач, связанных со сборкой проекта. Он оказывает содействие на различных этапах, например, при добавлении jar-файлов в библиотеки проекта, составлении отчетов, выполнении тестов Junits, создании jar, war, ear-файлов в рамках проекта и прочего.

**1.7. База данных**

**PostgreSQL** — это объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. 1 Она разработана для сохранения, организации и обработки структурированной информации.

PostgreSQL поддерживает большую часть стандарта SQL и предлагает множество современных функций, например сложные запросы, внешние ключи, триггеры, изменяемые представления, транзакционную целостность и многоверсионность.

Система поддерживается во всех основных операционных системах, включая Windows, Linux, macOS, FreeBSD и OpenBSD, и обрабатывает широкий спектр рабочих нагрузок — от отдельных машин до хранилищ данных, озёр данных или веб-сервисов со многими одновременными пользователями.

Благодаря свободной лицензии PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных.

**1.8. View**

**1.8.1Thymeleaf**

Thymeleaf - современный серверный механизм Java-шаблонов для веб- и автономных сред. Позволяет в реальном времени заменять определённые конструкции в коде данными с сервера. С помощью Thymeleaf можно один раз написать шаблон и переиспользовать его несколько раз. Он работает с шестью видами шаблонов: HTML, XML, CSS, JavaScript, RAW и обычным текстом.

Шаблоны Thymeleaf отвечают за отображение данных и управление динамическим контентом. С их помощью разработчики могут встраивать выражения и директивы прямо в HTML-код, облегчая процесс создания динамических страниц. Thymeleaf удобно применять в связке с фреймворком Spring.

**1.8.2. Bootstrap**

Bootstrap - это открытый и бесплатный HTML-, CSS- и JS-фреймворк, который используют веб-разработчики для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений. Включает в себя CSS- и HTML-шаблоны оформления для веб-форм, меток, типографики, кнопок, блоков навигации и других компонентов веб-интерфейса. Основные преимущества Bootstrap:

* Bootstrap позволяет создавать адаптивные сайты. Дизайн сайта будет корректно отображаться на экранах устройств разных размеров вне зависимости от их диагонали.
* Сайты, сделанные с использованием Bootstrap, будут одинаково отображаться во всех современных браузерах.
* Понятный код
* Bootstrap легко использовать в разработке, в нем легко разобраться.
* Bootstrap позволяет писать качественный и понятный код, который легко поймет другой разработчик. Это значительно упрощает разработку в команде.
* Элементы Bootstrap смотрятся гармонично между собой и позволяют создавать страницы и сайты в едином стиле.

**2. Проектирование и разработка**

**2.1 Порядок проведения работ**

В ходе разработки приложения были проведены следующие работы:

* + Планирование
  + Определение функциональных требований.
  + Выбор архитектуры приложения.
  + Выбор используемых технологий.
  + Настройка окружения
  + Создание проекта и выбор зависимостей
  + Разработка модели данных
  + Разработка сервисного слоя
  + Разработка контролеров
  + Внедрение безопасности
  + Разработка пользовательского интерфейса
  + Отладка тестирование и исправление ошибок
  + Документирование

**2.2. Определение требований к приложению**

Для приложения "менеджер задач" были определены следующие бизнес требования:

* + Доступ пользователей только после авторизации.
  + Создание задач.
  + Назначение задач на исполнение другим пользователям.
  + Редактирование и удаление задач с учетом авторства задачи.
  + Комментирование задачи исполнителем изменение статуса задачи.
  + Отображение списка всех задач.
  + Отображение карточек задач на главные станицы с ближайшим сроком исполнения.
  + Переход в карточку задания по клику.
  + Аутентификация и авторизация пользователей.

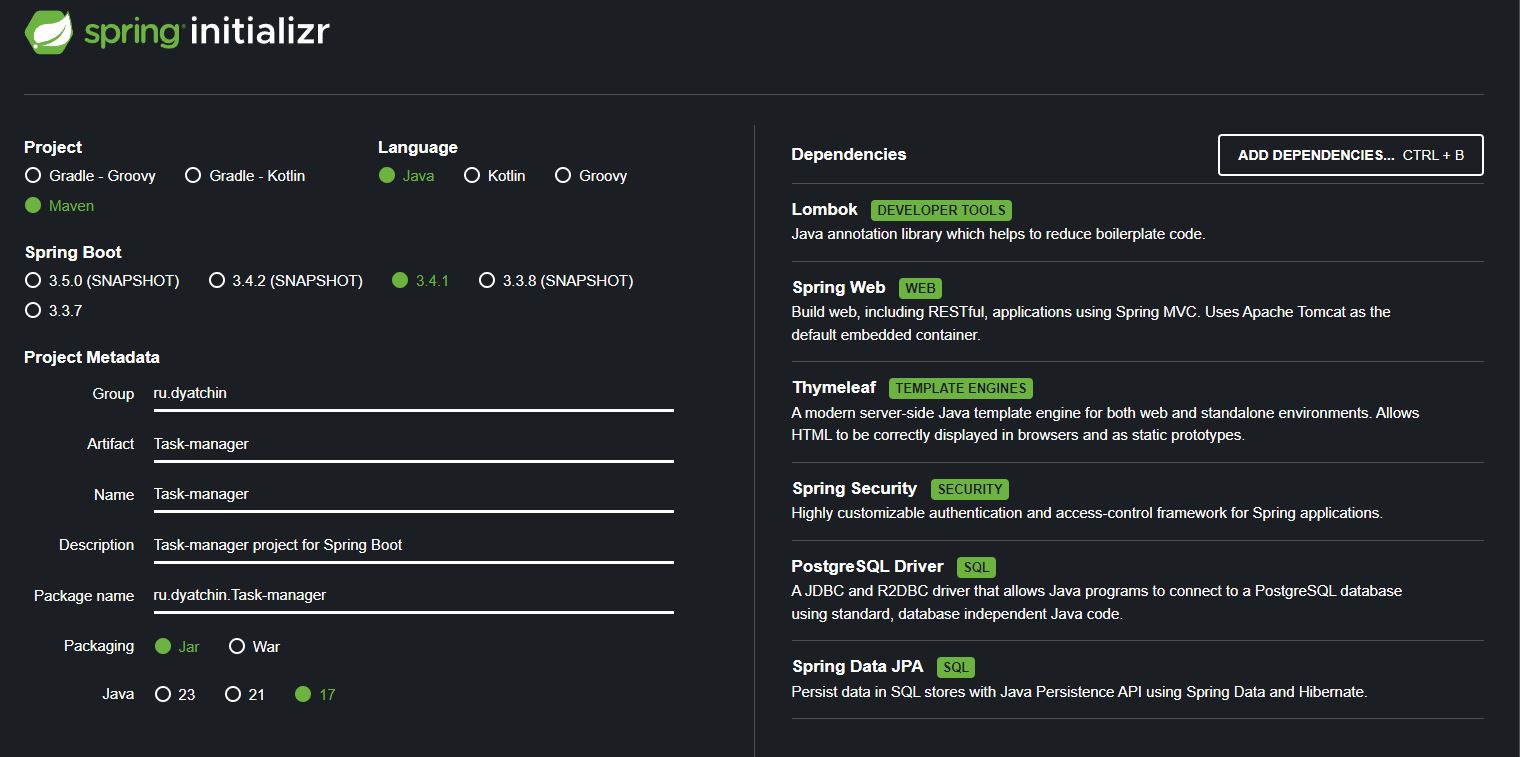
**2.3. Выбор стека технологий и архитектуры приложения**

* **Язык программирования**: Java для разработки серверной части приложения так как этот язык хорошо подходит для создания масштабируемых и надежных веб-приложений.
* **Фреймворк для веб-разработки**: SpringBoot с модулями Spring MVC и SpringSecurity для быстрой и удобной разработки серверной части приложения. SpringSecurity обеспечит безопасность приложения, а Spring MVC позволит создать удобный интерфейс.
* **База данных**: PostgreSQL для хранения данных. Обе системы хорошо масштабируются и обеспечивают надежное хранение данных.
* **Фронтенд:** HTML для создания интерфейса приложения. Для упрощения разработки интерфейса можно использовать фреймворк Bootstrap и Шаблонизатор Thymeleaf.
* **Аутентификация и авторизация**: SpringSecurity используется для обеспечения безопасности приложения, включая аутентификацию пользователей и управление доступом к ресурсам.
* **Среда разработки**: IntelliJ IDEA для разработки серверной части приложения.
* **Архитектура** приложения в рамках JavaSpring основывается на паттерне проектирования MVC (Model-View-Controller), который позволяет разделить приложение на три основных компонента: модель (Model), представление (View) и контроллер (Controller). Наше приложение состоит из данных составных частей:

**2.4. Настройка окружения**

* Установка и настройка JavaDevelopmentKit (JDK).
* Установка и настройка IntelliJ IDEA — это интегрированная среда разработки (IDE) для Java, от компании JetBrains, ApacheMaven (для сборки проекта), и SpringBoot (для создания и запуска приложения) в комплекте.

**2.5 Создание проекта и выбор зависимостей**

Проект создан при помощи SpringInitializr:

Версия java: 17

Spring Boot 3.4.1

Добавлены следующие зависимости:

* Lombook
* SpringWeb
* Thymeleaf
* SpringSecurity
* PostgresSQL Driver
* SpringDataJPA

**2.6 Разработка модели данных**

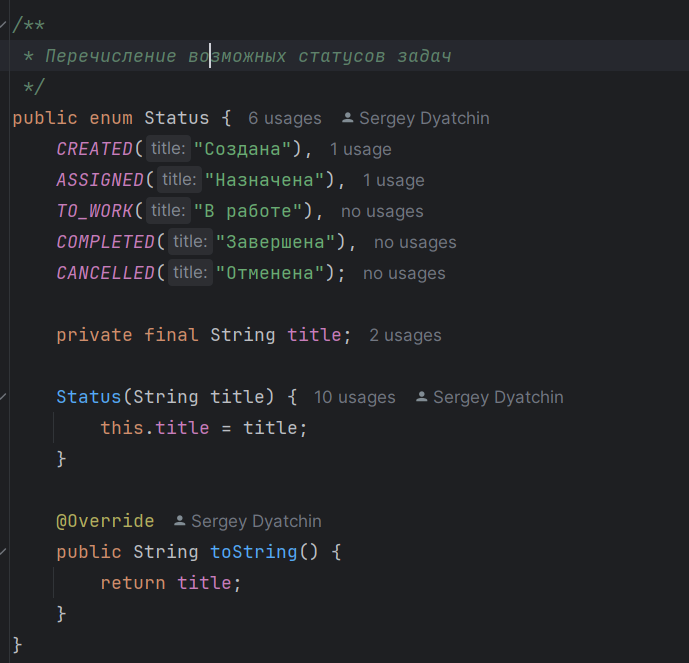
Первым разработанным классом стал Task.



Класс Task содержал все необходимы поля для хранения информации о задаче.

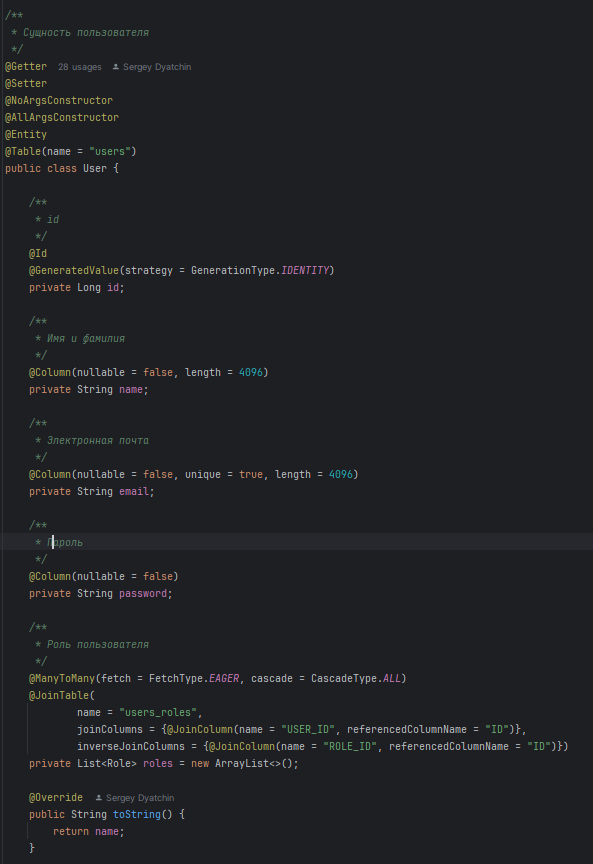
* Id
* Дата создания
* Дата дедлайна
* Название
* Содержание
* Комментарий
* Статус задачи (Enum)
* Автор (связь многие к одному с User)
* Исполнитель (связь многие к одному с User)

Класс Status является перечислением и определяет возможные статусы задачи.



Классc User содержит поля, описывающие пользователя.

* id
* Имя
* Email
* Пароль (в необратимом виде)
* Роль (связь многие к многим с Role реализуемое через промежуточную таблицу)



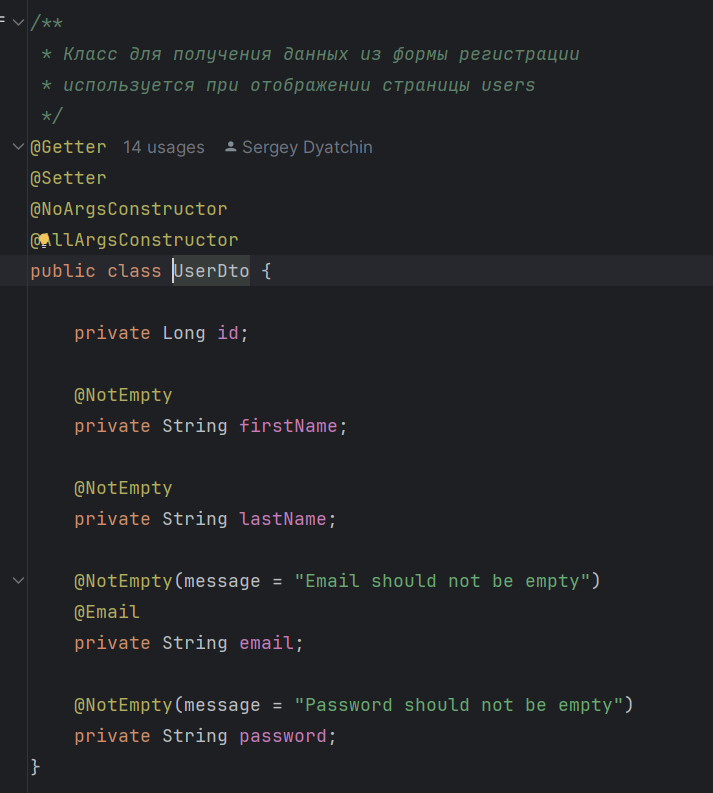
Класс Role описывает роль пользователя и содержит следующие поля

* Id
* Имя
* Список пользователей, которые эту роль имеют

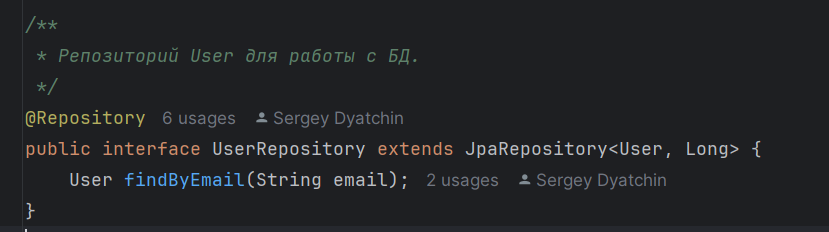


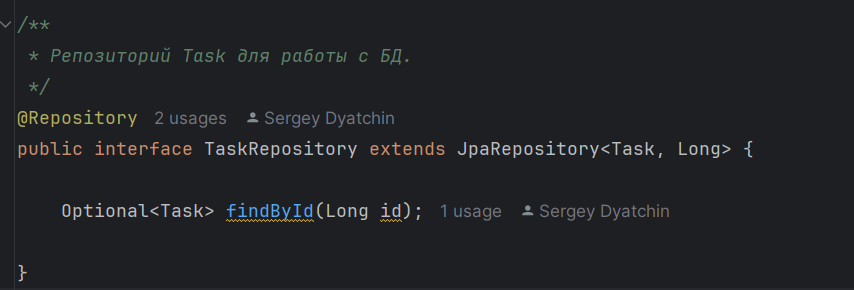
Класс UserDto необходим как сущность для получения данных из формы регистрации и дальнейшего сохранения в базу данных в виде класса User. Он содержит все теже формы что регистрационная форма на сайте:

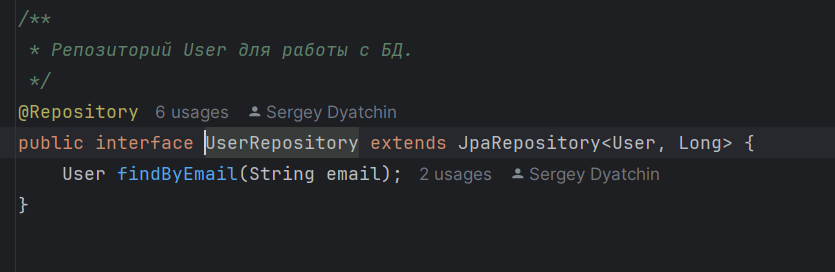
* Id
* Имя
* Фамилия
* Email
* Пароль



Для работы с базой данных были созданы интерфейсы депозитариев для сущностей Task, User, Role.

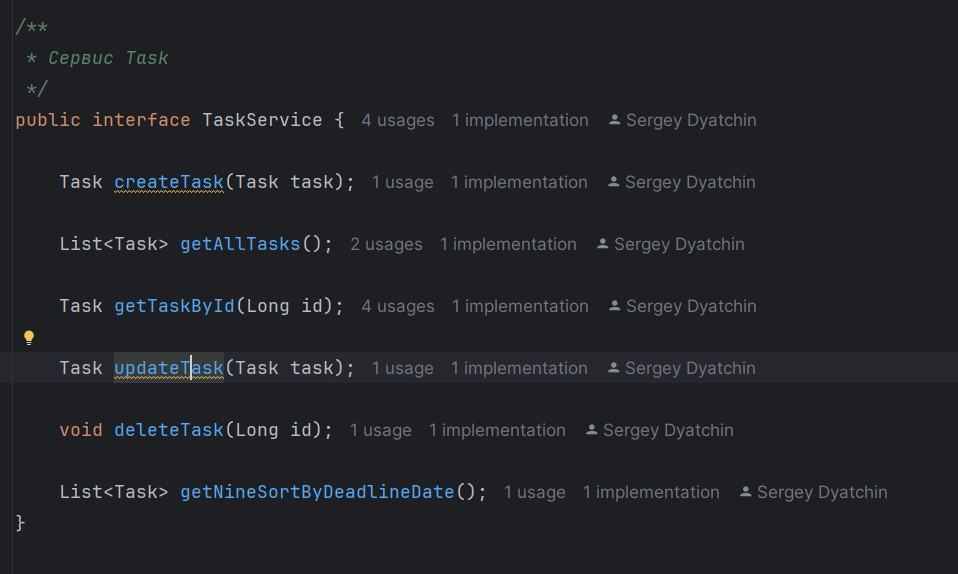


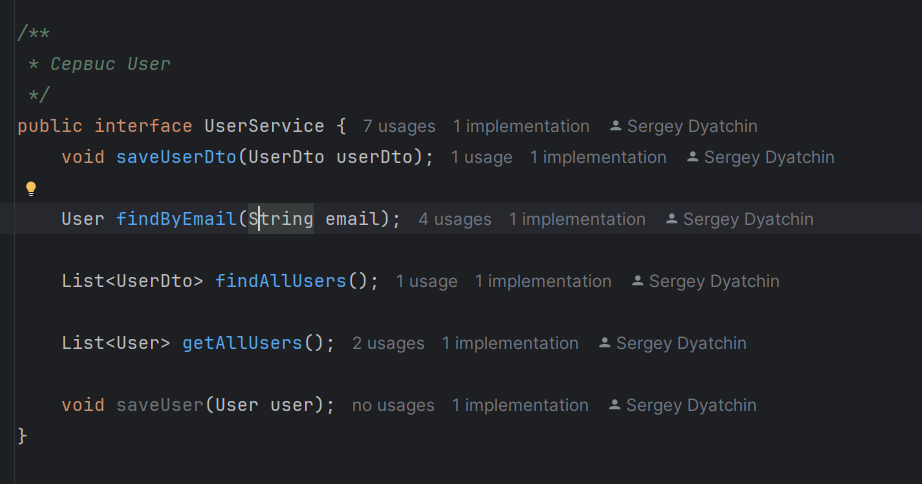




**2.7 Разработка сервисного слоя**

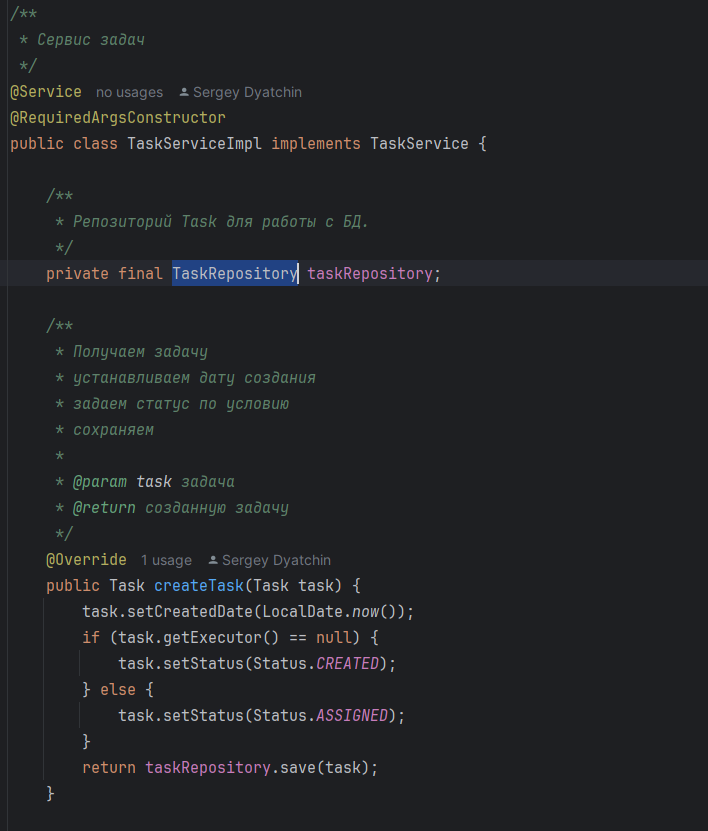
Разработка сервисов началась с создание интерфейсов и декларирование необходимых методов по манипуляциям с сущностями. Конкретная реализация методов была произведена позже в классах, имплементирующих данные методы.

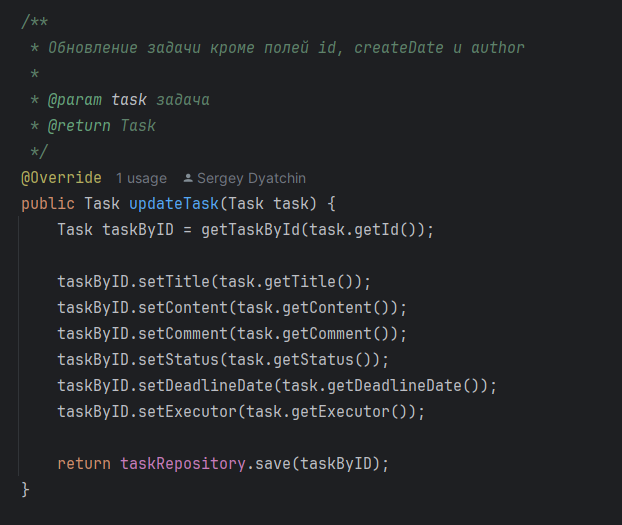
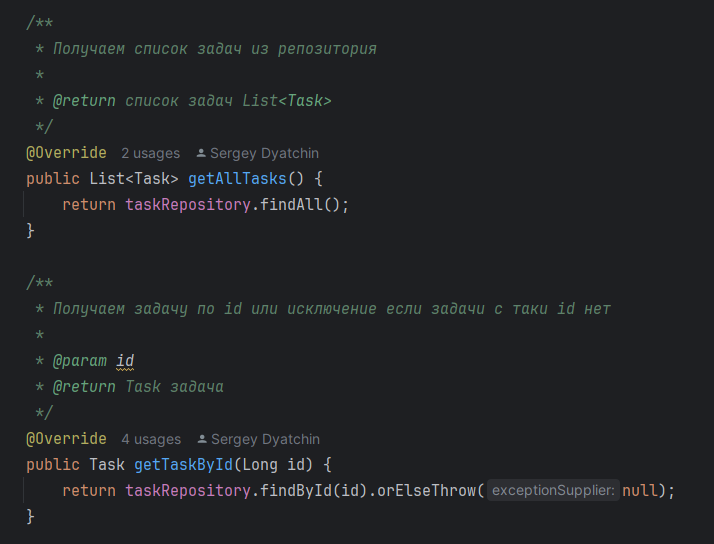


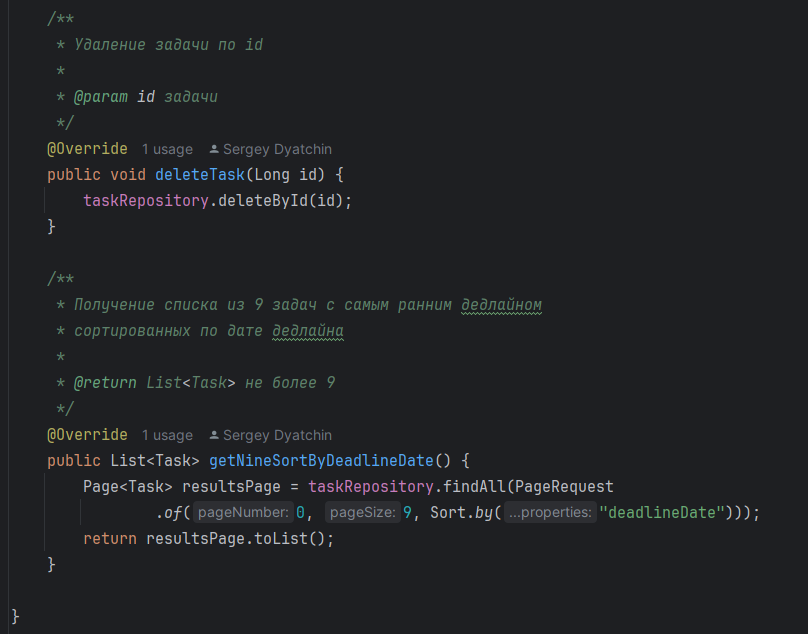


Класс TaskServiceImpl имплементирующий интерфейс TaskService реализовал указанные обязательный методы. Кроме того для проведения CRUD операций в него был инжектирован TaskRepository.

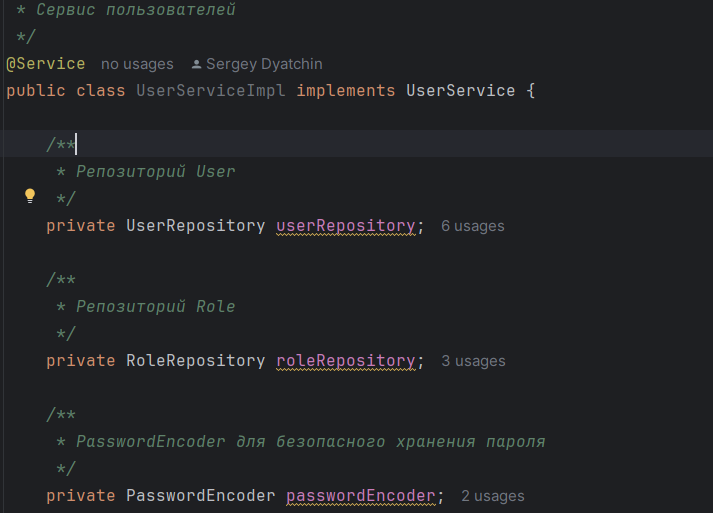
Подробное описание работы каждого метода можно увидеть в комментариях на картинках.



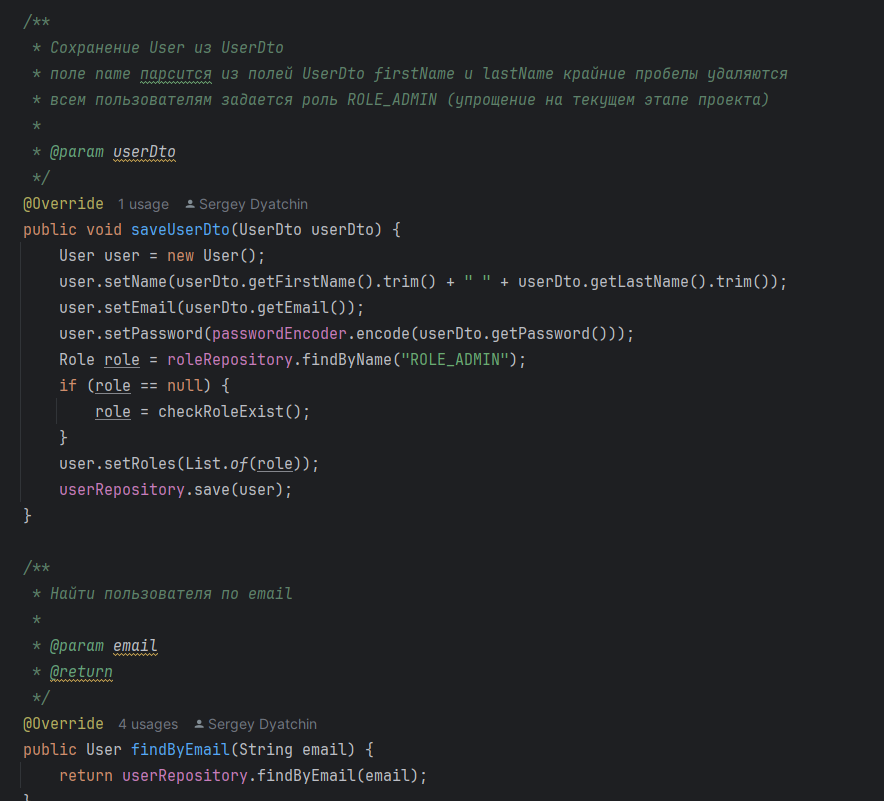


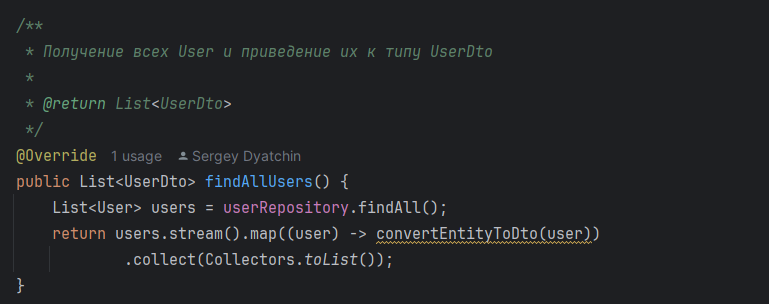


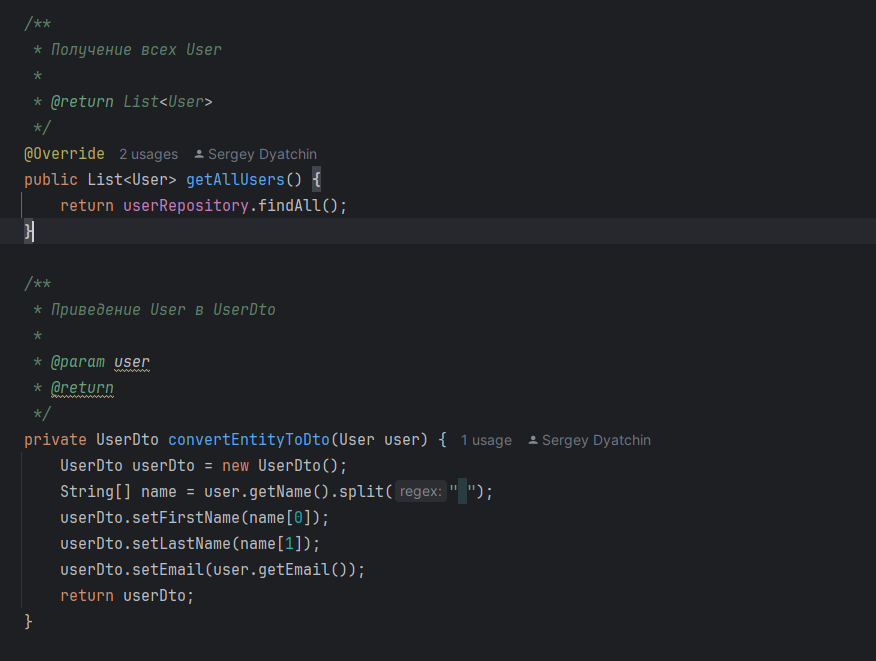
Класс UserServiceImpl имплементирующий интерфейс TaskService реализовал указанные обязательный методы. Кроме того, для проведения CRUD операций в него был инжектирован UserRepository, RoleRepository, а для безопасного хранения пароля PasswordEncoder.

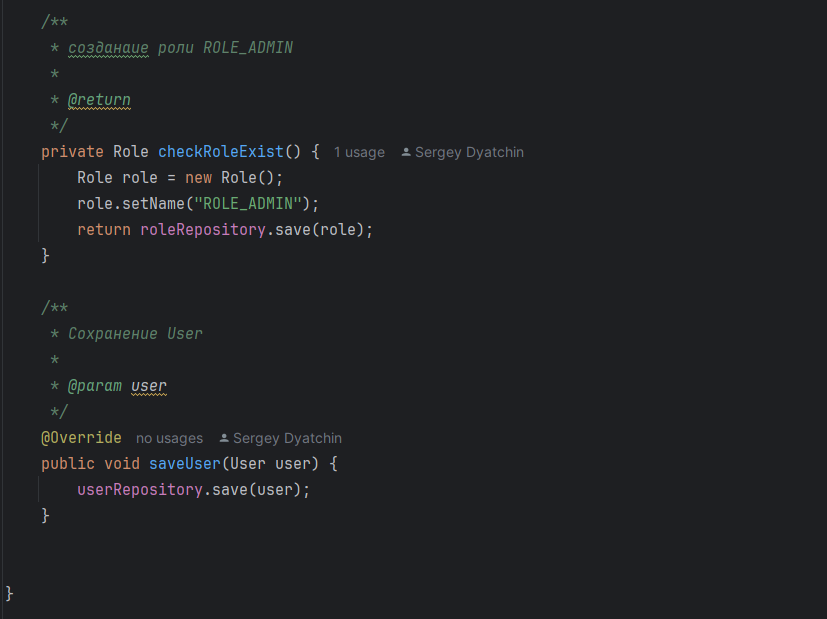


Подробное описание работы каждого метода можно увидеть в комментариях на картинках.



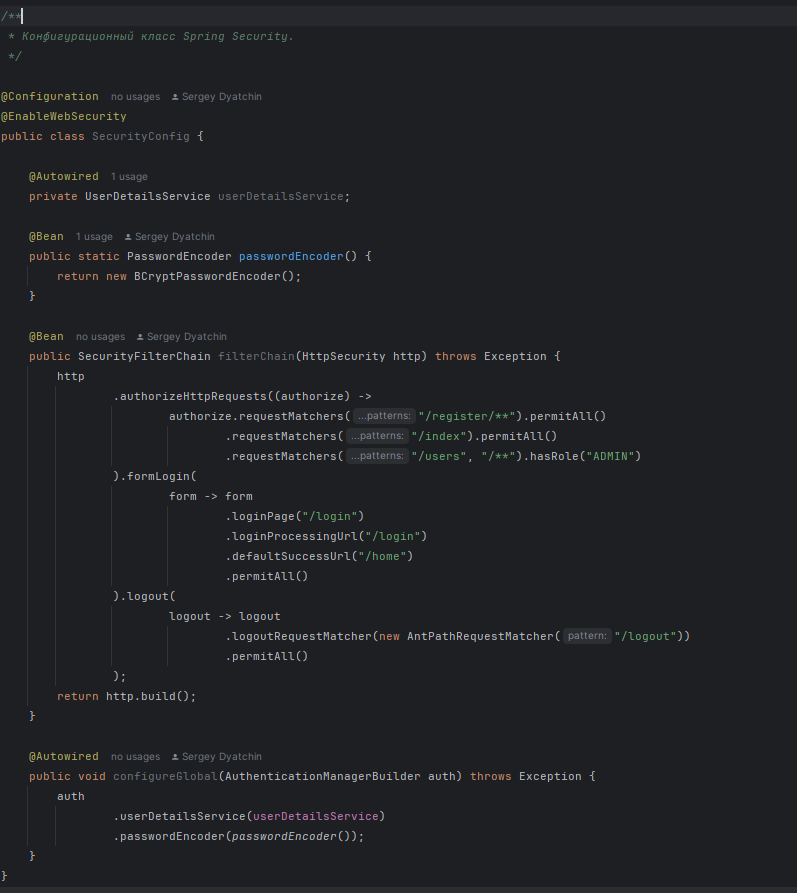




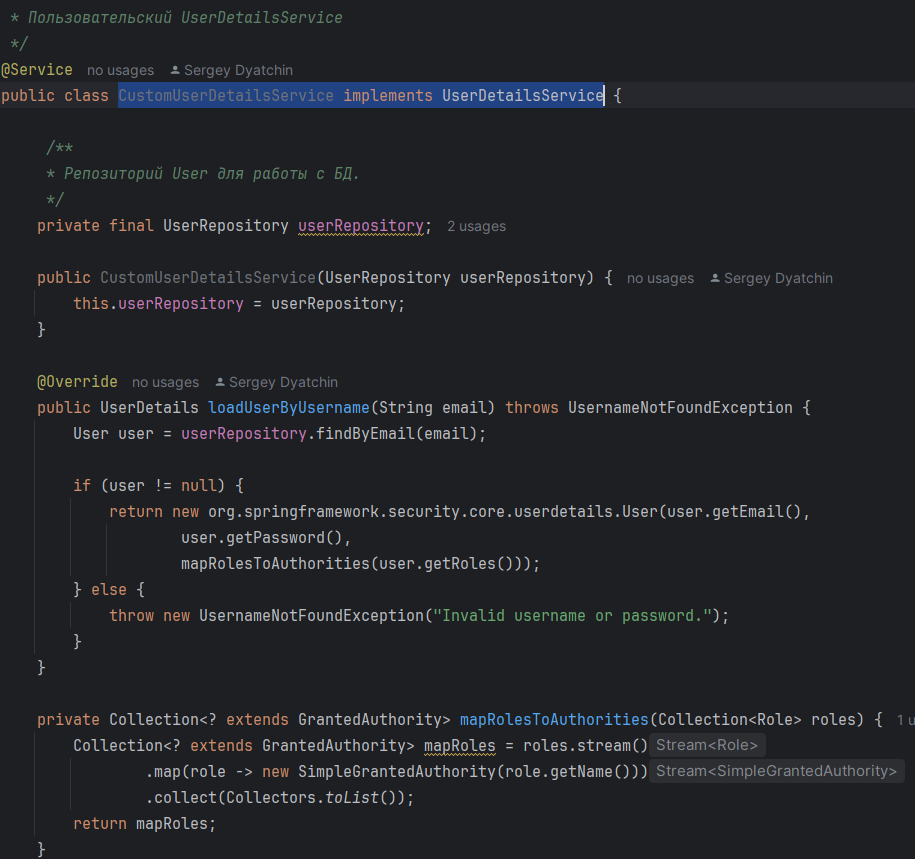


**2.8 Внедрение безопасности**

Внедрение Spring Security начинается с создание Bean SecurityConfig.

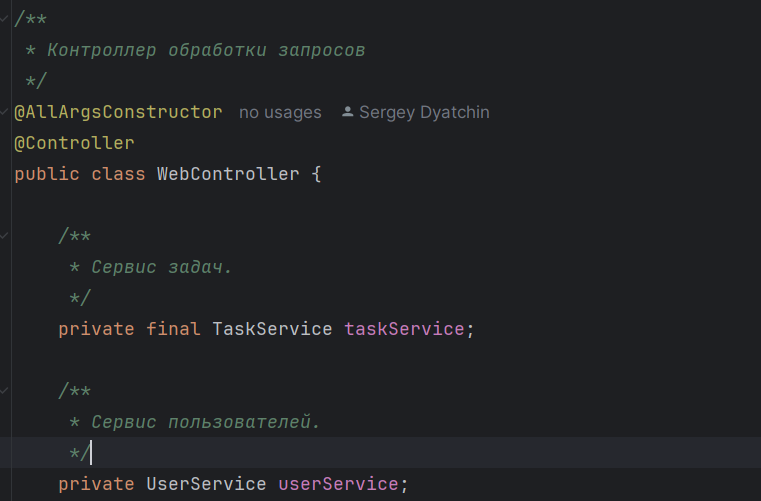


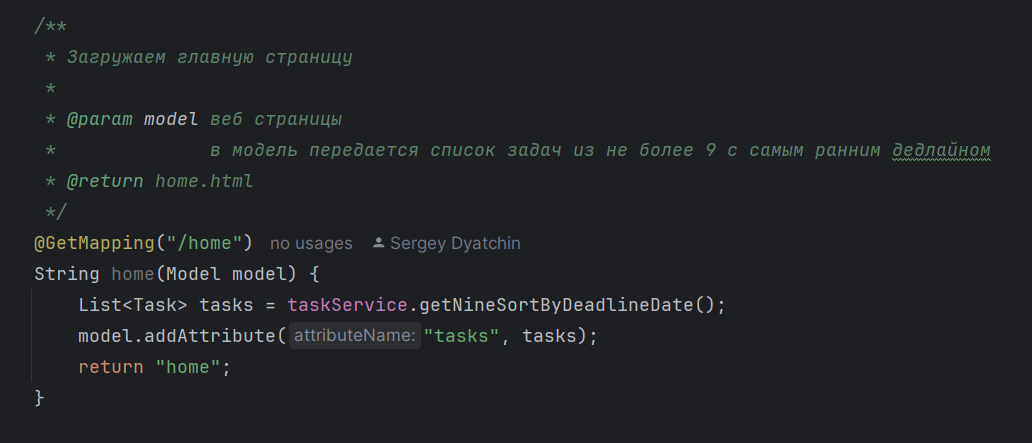
Отдельно стоит отметить Класс CustomUserDetailsService имплементирующий UserDetailsService, Класс отвечает за авторизацию Spring Security.

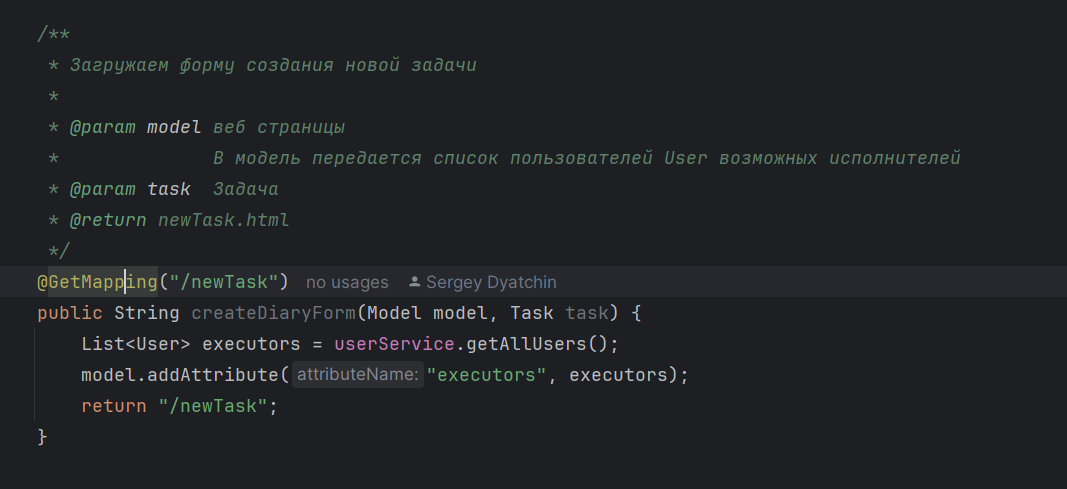


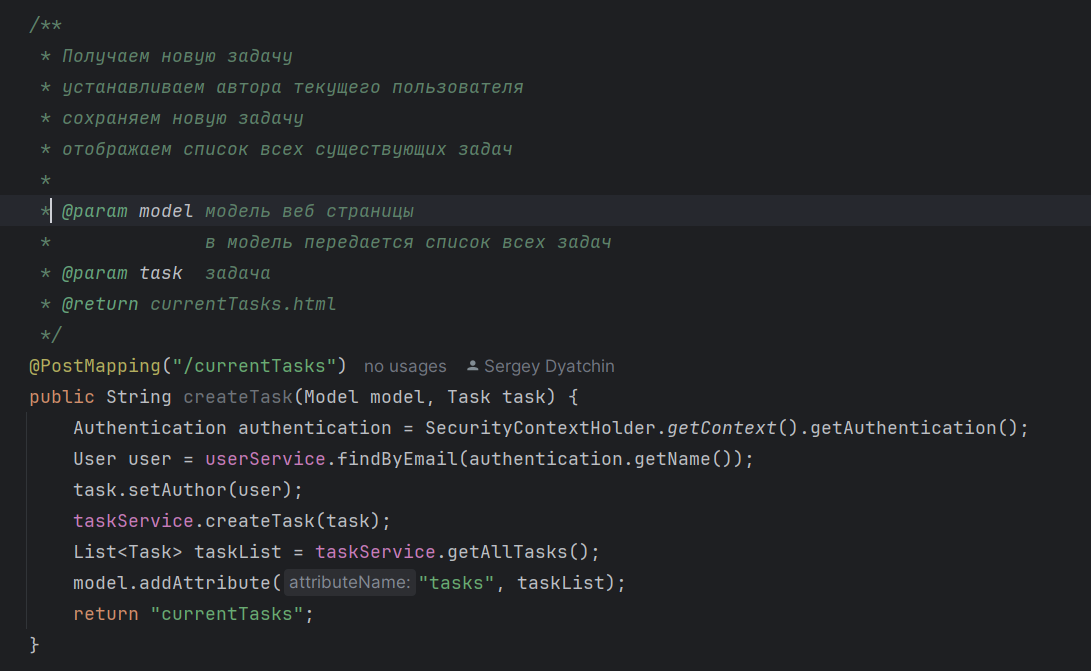
**2.9 Разработка контролеров**

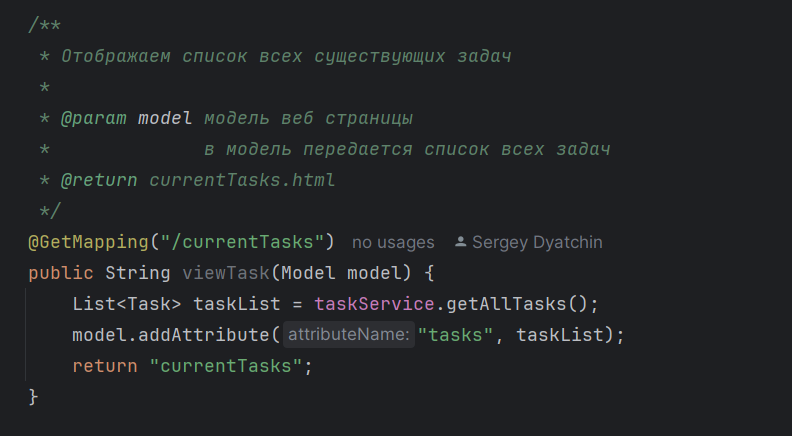
Сердцем программы является класс WebController именно он обрабатывает запросы клиента. Для работы в него был инжектирован TaskService и UserServise. Подробное описание работы каждого метода можно увидеть в комментариях на картинках.

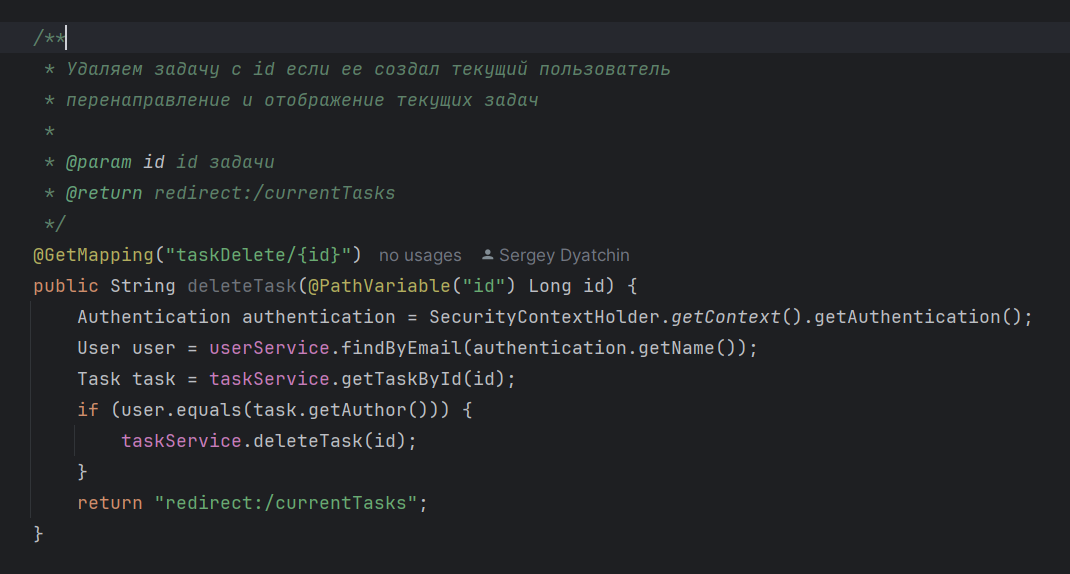


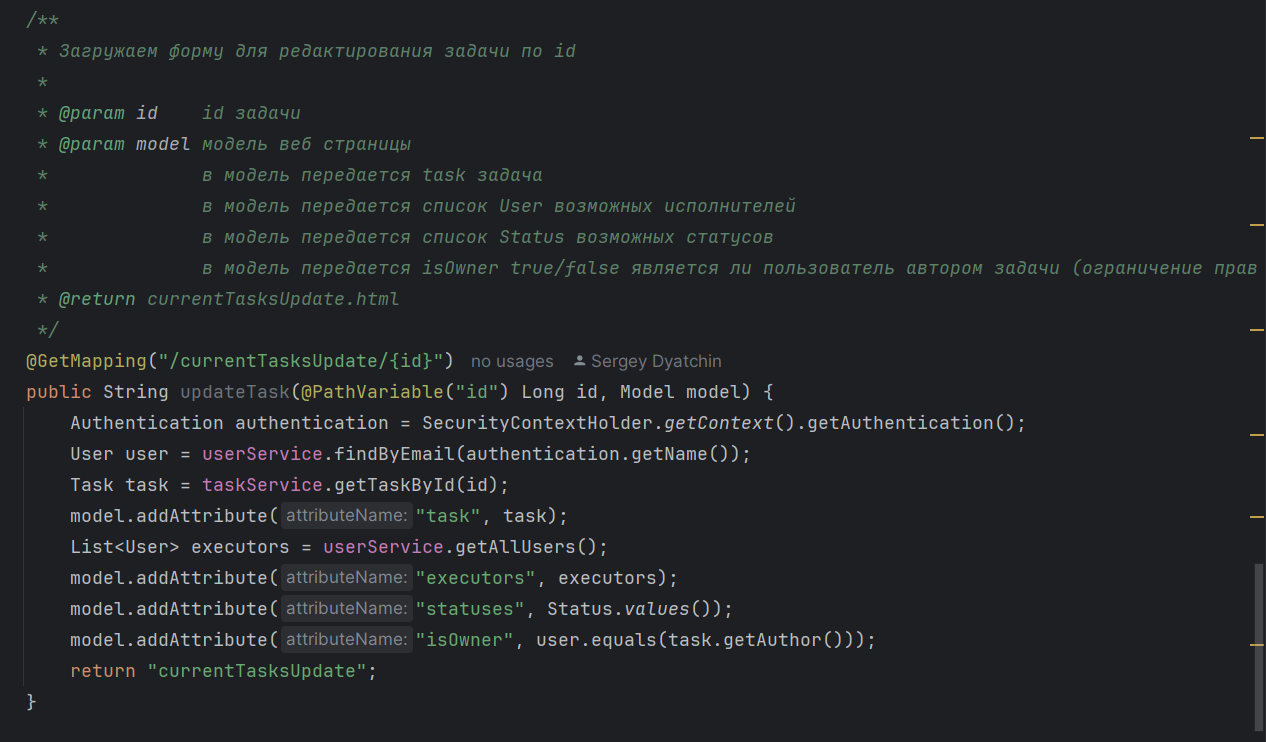


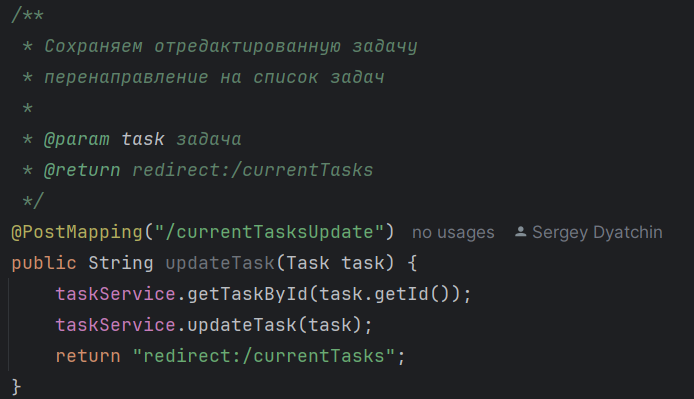




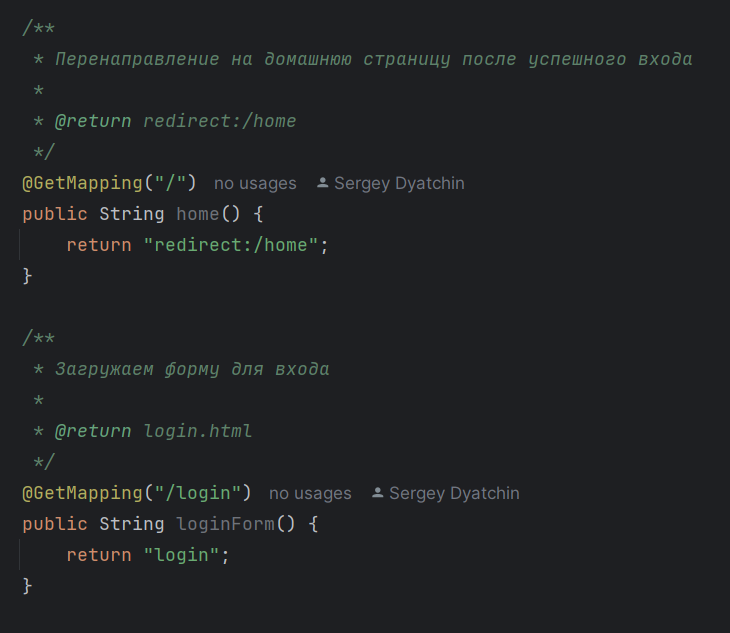




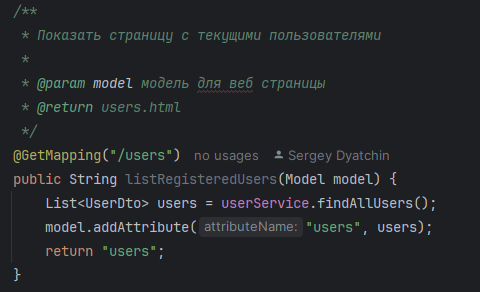




Класс UserController обрабатывает запросы, касающиеся пользователей авторизации регистрации и безопасности. Подробное описание работы каждого метода можно увидеть в комментариях на картинках.



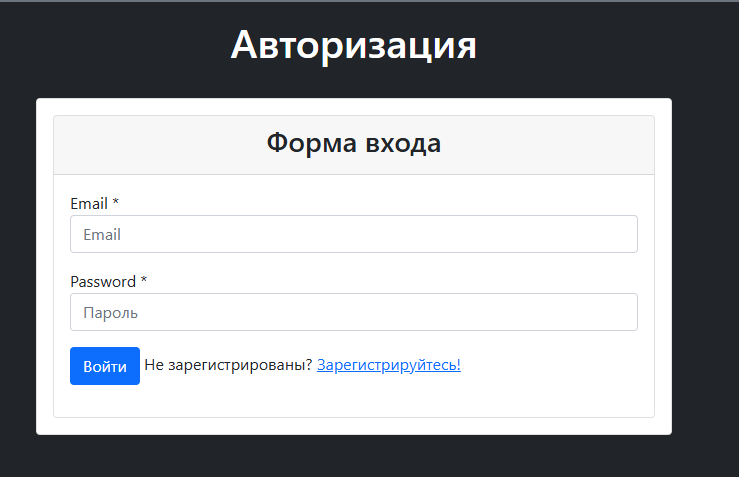




* 1. **Разработка пользовательского интерфейса**

При разработке пользовательского интерфейса применялся фреймворк Bootstrap и Шаблонизатор Thymeleaf.

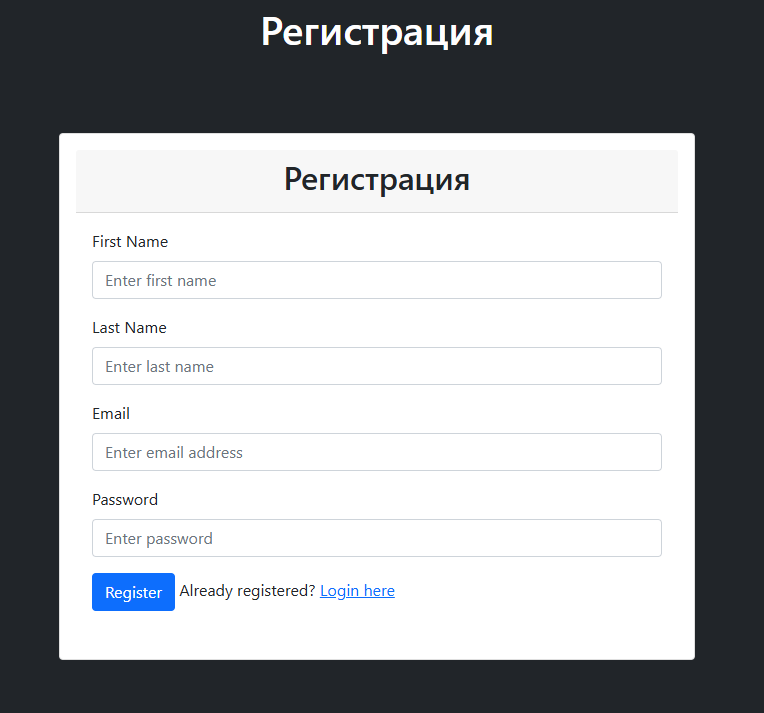
* + 1. **Страница для входа**

**Внешний вид**

**Код**

<!DOCTYPE html>  
<html lang="en"  
 xmlns:th="http://www.thymeleaf.org"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Registration and Login System</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
</head>  
  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="/">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" th:href="@{/register}">Регистрация</a>  
 </li>  
 </ul>  
</nav>  
<h1 class="text-white text-center">Авторизация</h1>  
<br/>  
<div class="container">  
 <div class="row">  
 <div class="col-md-6 offset-md-3">  
 <div class="card bg-white p-3">  
  
 <div th:if="${param.error}">  
 <div class="alert alert-danger">Invalid Email and Password.</div>  
 </div>  
 <div th:if="${param.logout}">  
 <div class="alert alert-success">You have been logged out.</div>  
 </div>  
 <div class="card">  
 <div class="card-header">  
 <h3 class="text-center">Форма входа</h3>  
 </div>  
 <div class="card-body">  
 <form  
 method="post"  
 th:action="@{/login}"  
 class="form-horizontal"  
 role="form"  
 >  
 <div class="mb-3">  
 <label for="username" class="control-label">Email \*</label>  
 <input type="text"  
 id="username"  
 name="username"  
 class="form-control"  
 placeholder="Email"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label for="password" class="control-label">Password \*</label>  
 <input type="password"  
 id="password"  
 name="password"  
 class="form-control"  
 placeholder="Пароль"  
 />  
 </div>  
  
 <div class="mb-3">  
 <button type="submit" class="btn btn-primary">Войти</button>  
 <span> Не зарегистрированы? <a th:href="@{/register}"> Зарегистрируйтесь!</a></span>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
</body>  
</html>

* + 1. **Страница регистрации**

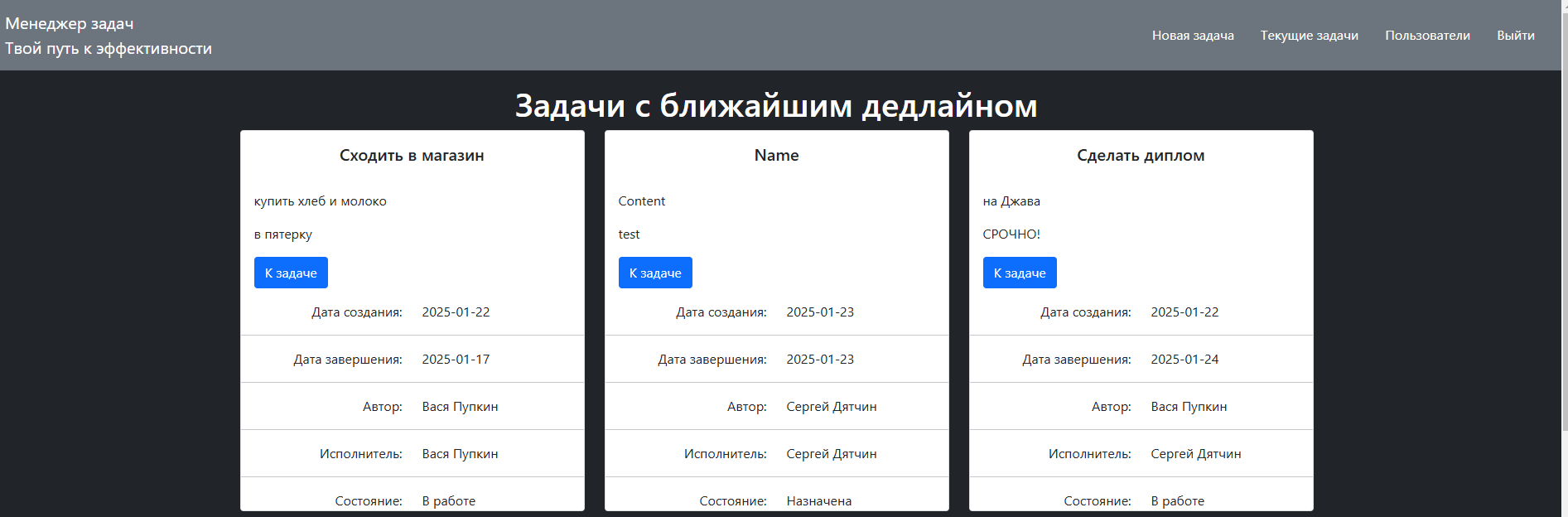
**Внешний вид**

**Код**

<!DOCTYPE html>  
<html lang="en"  
 xmlns:th="http://www.thymeleaf.org"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Registration and Login System</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
</head>  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="/">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" th:href="@{/login}">Вход</a>  
 </li>  
 </ul>  
</nav>  
<h1 class="text-white text-center">Регистрация</h1>  
<br/>  
<br/>  
<br/>  
<div class="container">  
 <div class="row">  
 <div class="col-md-6 offset-md-3">  
 <div class="card bg-white p-3">  
 <div th:if="${param.success}">  
 <div class="alert alert-info">You've successfully registered  
 to our app!  
 </div>  
 </div>  
 <div class="card-header">  
 <h2 class="text-center">Регистрация</h2>  
 </div>  
 <div class="card-body">  
 <form method="post" role="form" th:action="@{/register/save}" th:object="${user}">  
  
 <div class="form-group mb-3">  
 <label class="form-label">First Name</label>  
 <input class="form-control"  
 id="firstName"  
 name="firstName"  
 placeholder="Enter first name"  
 th:field="\*{firstName}"  
 type="text"  
 />  
 <p th:errors="\*{firstName}" class="text-danger"  
 th:if="${#fields.hasErrors('firstName')}">  
 </p>  
 </div>  
  
 <div class="form-group mb-3">  
 <label class="form-label">Last Name</label>  
 <input class="form-control"  
 id="lastName"  
 name="lastName"  
 placeholder="Enter last name"  
 th:field="\*{lastName}"  
 type="text"  
 />  
 <p th:errors="\*{lastName}" class="text-danger"  
 th:if="${#fields.hasErrors('lastName')}">  
 </p>  
 </div>  
 <div class="form-group mb-3">  
 <label class="form-label">Email</label>  
 <input class="form-control"  
 id="email"  
 name="email"  
 placeholder="Enter email address"  
 th:field="\*{email}"  
 type="email"  
 />  
 <p th:errors="\*{email}" class="text-danger"  
 th:if="${#fields.hasErrors('email')}">  
 </p>  
 </div>  
 <div class="form-group mb-3">  
 <label class="form-label">Password</label>  
 <input class="form-control"  
 id="password"  
 name="password"  
 placeholder="Enter password"  
 th:field="\*{password}"  
 type="password"  
 />  
 <p th:errors="\*{password}" class="text-danger"  
 th:if="${#fields.hasErrors('password')}">  
 </p>  
 </div>  
  
 <div class="form-group mb-3">  
 <button class="btn btn-primary" type="submit">Register</button>  
 <span>Already registered? <a href="/" th:href="@{/login}">Login  
 here</a></span>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
</body>  
</html>

* + 1. **Главная страница**

**Внешний вид**

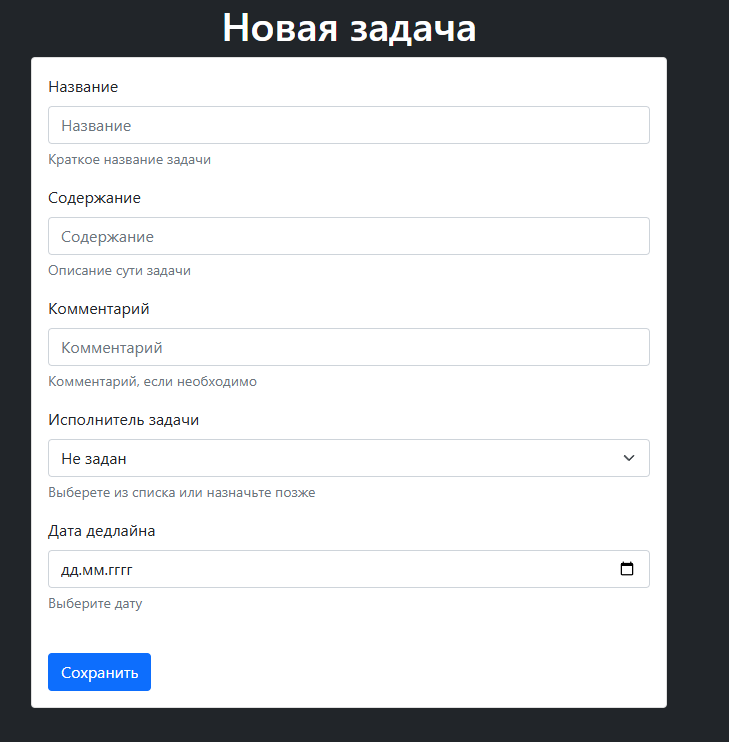


**Код**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Менеджер задач</title>  
 <style>  
 h1 {  
 margin-bottom: 20px;  
 }  
 </style>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
</head>  
  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="#">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/newTask">Новая задача</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/currentTasks">Текущие задачи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/users">Пользователи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/logout">Выйти</a>  
 </li>  
 </ul>  
</nav>  
  
  
<h1 class="text-white text-center">Задачи с ближайшим дедлайном</h1>  
  
<div class="container">  
 <div class="row">  
 <div class="col-sm-4 mb-3" th:each="task : ${tasks}">  
 <div class="card h-100">  
 <div class="card-body">  
 <h5 class="card-title text-center" th:text="${task.title}"></h5>  
 <br>  
 <p class="card-text" th:text="${task.content}"></p>  
 <p class="card-text" th:text="${task.comment}"></p>  
 <a th:href="@{currentTasksUpdate/{id}(id=${task.id})}" class="btn btn-primary stretched-link"  
 data-bs-toggle="tooltip" data-bs-title="Редактировать задачу">К задаче</a>  
 </div>  
 <div class="container">  
 <div class="row">  
 <div class="col-6 text-end">  
 Дата создания:  
 </div>  
 <div class="col-6" th:text="${task.createdDate}"></div>  
 </div>  
 </div>  
 <hr>  
 <div class="container">  
 <div class="row">  
 <div class="col-6 text-end">  
 Дата завершения:  
 </div>  
 <div class="col-6" th:text="${task.deadlineDate}"></div>  
 </div>  
 </div>  
 <hr>  
 <div class="container">  
 <div class="row">  
 <div class="col-6 text-end">  
 Автор:  
 </div>  
 <div class="col-6" th:text="${task.author}"></div>  
 </div>  
 </div>  
 <hr>  
 <div class="container">  
 <div class="row">  
 <div class="col-6 text-end">  
 Исполнитель:  
 </div>  
 <div class="col-6" th:text="${task.executor}"></div>  
 </div>  
 </div>  
 <hr>  
 <div class="container">  
 <div class="row">  
 <div class="col-6 text-end">  
 Состояние:  
 </div>  
 <div class="col-6" th:text="${task.status}"></div>  
 </div>  
 </div>  
  
 <!--div class="card-body">  
 <a href="#" class="card-link">Ссылка карточки</a>  
 <a href="#" class="card-link">Другая ссылка</a>  
 </div-->  
 </div>  
 </div>  
 </div>  
</div>  
  
</body>  
</html>

* + 1. **Создать задачу**

**Внешний вид**

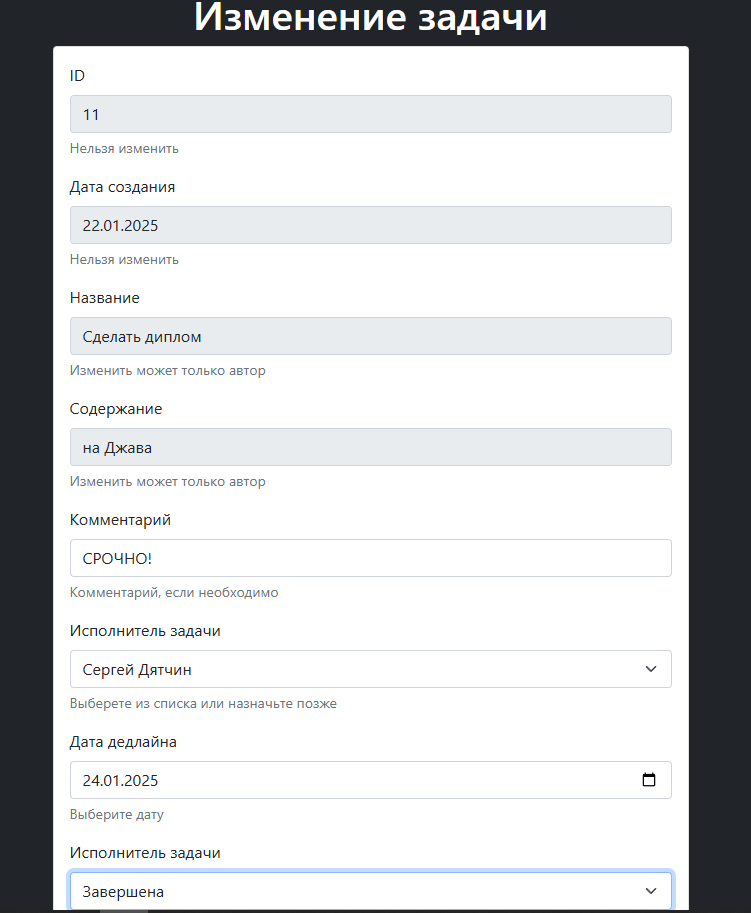


**Код**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Добавление задачи</title>  
 <style>  
 h1 {  
 margin-bottom: 20px;  
 }  
 </style>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
</head>  
  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="/">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/newTask">Новая задача</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/currentTasks">Текущие задачи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/users">Пользователи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/logout">Выйти</a>  
 </li>  
 </ul>  
</nav>  
<h1 class="text-white text-center">Новая задача</h1>  
  
<div class="container">  
 <div class="row justify-content-center">  
 <div class="col-12 col-md-8 col-lg-6">  
 <div class="card bg-white p-3">  
 <form th:action=@{/currentTasks} th:object="${task}" method="post">  
 <div class="mb-3">  
 <label for="title" class="form-label">Название</label>  
 <input type="text" th:field="\*{title}" class="form-control" id="title"  
 aria-describedby="titleHelp" placeholder="Название">  
 <div id="titleHelp" class="form-text">Краткое название задачи</div>  
 </div>  
 <div class="mb-3">  
 <label for="content" class="form-label">Содержание</label>  
 <input type="text" th:field="\*{content}" class="form-control" id="content"  
 aria-describedby="contentHelp" placeholder="Содержание">  
 <div id="contentHelp" class="form-text">Описание сути задачи</div>  
 </div>  
 <div class="mb-3">  
 <label for="comment" class="form-label">Комментарий</label>  
 <input type="text" th:field="\*{comment}" class="form-control" id="comment"  
 aria-describedby="commentHelp" placeholder="Комментарий">  
 <div id="commentHelp" class="form-text">Комментарий, если необходимо</div>  
 </div>  
 <div class="mb-3">  
 <label for="executor" class="form-label">Исполнитель задачи</label>  
 <select class="form-select" id="executor" th:field="\*{executor}">  
 <option value="" selected>Не задан</option>  
 <option th:each="e: ${executors}"  
 th:value="${e.id}"  
 th:text="${e.name}"></option>  
 </select>  
 <div id="executorHelp" class="form-text">Выберете из списка или назначьте позже</div>  
 </div>  
 <div class="mb-3">  
 <label for="deadlineDate" class="form-label">Дата дедлайна</label>  
 <input type="date" th:field="\*{deadlineDate}" class="form-control" id="deadlineDate"  
 aria-describedby="deadlineDateHelp" placeholder="Дата дедлайна">  
 <div id="deadlineDateHelp" class="form-text">Выберите дату</div>  
 </div>  
  
 <br/>  
 <button type="submit" class="btn btn-primary">Сохранить</button>  
 </form>  
 </div>  
 </div>  
 </div>  
</div>  
</body>  
</html>

* + 1. **Создать задачу**

**Внешний вид**



**Код**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">  
<meta charset="UTF-8">  
<head>  
 <title>Изменение задачи</title>  
 <style>  
  
 h1 {  
 margin-bottom: 20px;  
 }  
  
 </style>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
</head>  
  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="/">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/newTask">Новая задача</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/currentTasks">Текущие задачи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/users">Пользователи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/logout">Выйти</a>  
 </li>  
 </ul>  
</nav>  
  
<h1 class="text-white text-center">Изменение задачи</h1>  
<div class="container">  
 <div class="row justify-content-center">  
 <div class="col-12 col-md-8 col-lg-6">  
 <div class="card bg-white p-3">  
 <form th:action=@{/currentTasksUpdate} th:object="${task}" method="post">  
 <div class="mb-3">  
 <label for="id" class="form-label">ID</label>  
 <input type="text" th:field="\*{id}" class="form-control" id="id"  
 aria-describedby="idHelp" readonly>  
 <div id="idHelp" class="form-text">Нельзя изменить</div>  
 </div>  
 <div class="mb-3">  
 <label for="createdDate" class="form-label">Дата создания</label>  
 <input type="date" th:field="\*{createdDate}" class="form-control" id="createdDate"  
 aria-describedby="createdDateHelp" readonly>  
 <div id="createdDateHelp" class="form-text">Нельзя изменить</div>  
 </div>  
 <div class="mb-3">  
 <label for="title" class="form-label">Название</label>  
 <input th:if="${isOwner}" type="text" th:field="\*{title}" class="form-control" id="title"  
 aria-describedby="titleHelp">  
 <input th:if="!${isOwner}" type="text" readonly th:field="\*{title}" class="form-control" id="title"  
 aria-describedby="titleHelp">  
 <div th:if="!${isOwner}" id="titleHelp" class="form-text">Изменить может только автор</div>  
 </div>  
 <div class="mb-3">  
 <label for="content" class="form-label">Содержание</label>  
 <input th:if="${isOwner}" type="text" th:field="\*{content}" class="form-control" id="content"  
 aria-describedby="contentHelp">  
 <input th:if="!${isOwner}" type="text" readonly th:field="\*{content}" class="form-control" id="content"  
 aria-describedby="contentHelp">  
 <div th:if="!${isOwner}" id="contentHelp" class="form-text">Изменить может только автор</div>  
 </div>  
 <div class="mb-3">  
 <label for="comment" class="form-label">Комментарий</label>  
 <input type="text" th:field="\*{comment}" class="form-control" id="comment"  
 aria-describedby="commentHelp" placeholder="Комментарий">  
 <div id="commentHelp" class="form-text">Комментарий, если необходимо</div>  
 </div>  
 <div class="mb-3">  
 <label for="executor" class="form-label">Исполнитель задачи</label>  
 <select class="form-select" id="executor" th:field="\*{executor}">  
 <option value="">Не задан</option>  
 <option th:each="e: ${executors}"  
 th:value="${e.id}"  
 th:text="${e.name}"></option>  
 </select>  
 <div id="executorHelp" class="form-text">Выберете из списка или назначьте позже</div>  
 </div>  
 <div class="mb-3">  
 <label for="deadlineDate" class="form-label">Дата дедлайна</label>  
 <input type="date" th:field="\*{deadlineDate}" class="form-control" id="deadlineDate"  
 aria-describedby="deadlineDateHelp" placeholder="Дата дедлайна">  
 <div id="deadlineDateHelp" class="form-text">Выберите дату</div>  
 </div>  
 <div class="mb-3">  
 <label for="status" class="form-label">Исполнитель задачи</label>  
 <select class="form-select" id="status" th:field="\*{status}">  
 <option th:each="e: ${statuses}"  
 th:value="${e.name}"  
 th:text="${e}"></option>  
 </select>  
 <div id="statusHelp" class="form-text">Установите новый статус задачи</div>  
 </div>  
 <br/>  
 <button type="submit" class="btn btn-primary">Сохранить</button>  
 </form>  
 </div>  
 </div>  
 </div>  
</div>  
</body>  
  
</html>

* + 1. **Текущие задачи**

**Внешний вид**



**Код**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Текущие задачи</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
 <style>  
  
 h1 {  
 margin-bottom: 20px;  
 }  
  
 .tg {  
 border-collapse: collapse;  
 border-spacing: 0;  
 border-color: #ccc;  
 }  
  
 .tg td {  
 font-family: Arial, sans-serif;  
 font-size: 14px;  
 padding: 10px 5px;  
 border-style: solid;  
 border-width: 1px;  
 overflow: hidden;  
 word-break: normal;  
 border-color: #ccc;  
 color: #333;  
 background-color: #fff;  
 }  
  
 .tg th {  
 font-family: Arial, sans-serif;  
 font-size: 14px;  
 font-weight: normal;  
 padding: 10px 5px;  
 border-style: solid;  
 border-width: 1px;  
 overflow: hidden;  
 word-break: normal;  
 border-color: #ccc;  
 color: #333;  
 background-color: #f0f0f0;  
 }  
 </style>  
  
</head>  
  
<body class="bg-dark">  
<nav id="navbar-example2" class="navbar bg-body-tertiary px-3 mb-3 bg-secondary">  
 <a class="navbar-brand text-white" href="/">Менеджер задач <br>Твой путь к эффективности</a>  
 <ul class="nav nav-pills">  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/newTask">Новая задача</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/currentTasks">Текущие задачи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/users">Пользователи</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-white" href="/logout">Выйти</a>  
 </li>  
 </ul>  
</nav>  
<h1 class="text-white text-center">Текущие задачи</h1>  
<div class="container">  
 <table class="tg">  
 <thead>  
 <tr>  
 <th width="80">Дата создания</th>  
 <th width="80">Дата дедлайна</th>  
 <th width="160">Название</th>  
 <th width="160">Содержание</th>  
 <th width="160">Комментарий</th>  
 <th width="160">Автор задачи</th>  
 <th width="160">Исполнитель задачи</th>  
 <th width="160">Статус</th>  
 <th width="60">Редактировать</th>  
 <th width="60">Удалить</th>  
  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="task : ${tasks}">  
  
 <td th:text="${task.createdDate}"></td>  
 <td th:text="${task.deadlineDate}"></td>  
 <td th:text="${task.title}"></td>  
 <td th:text="${task.content}"></td>  
 <td th:text="${task.comment}"></td>  
 <td th:text="${task.author}"></td>  
 <td th:text="${task.executor}"></td>  
 <td th:text="${task.status}"></td>  
 <td><a th:href="@{currentTasksUpdate/{id}(id=${task.id})}">Редактировать</a></td>  
 <td><a th:href="@{taskDelete/{id}(id=${task.id})}">Удалить</a></td>  
  
 </tr>  
 </tbody>  
 </table>  
</div>  
  
</body>  
</html>

**Заключение**

В рамках данного дипломного проекта было разработано веб-приложение «Менеджер задач», предназначенное для сохранения и отслеживания задач. Проект был реализован с использованием современных технологий веб-разработки, таких как Java Spring Framework, Thymeleaf, Bootstrap и PostgreSQL.

В процессе разработки были достигнуты следующие цели:

* Создание удобного и интуитивно понятного интерфейса для пользователей приложения.
* Реализация функционала добавления, редактирования и удаления задач.
* Обеспечение авторизации 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:

* Доработка ролей пользователей управления пользователя удаление редактирование.
* Создание групп пользователей с разграничением видимости задач по группам.
* Интеграция с API telegram для уведомления пользователей через bota о новых задачах, изменении в задачах.
* Просмотр задач через телеграмм.
* Разделение приложения на микросервисы.
* Покрытие приложения тестами.

**Список литературы**

1. Учебный материал Гикбрейнс.
2. **Java Полное руководство Герберт Шилдт (12-е издание** 2022 год **)**
3. **Spring в действии,** Крейг Уоллс (6-е издание, 2022 год).
4. Статьи и видеоматериалы из сети интернет.