Оглавление

[ВВЕДЕНИЕ 12](#_Toc170216352)

[**ГЛАВА 1. АНАЛИЗ И РАЗРАБОТКА ТРЕБОВАНИЙ** 15](#_Toc170216353)

[**1.3 Цель и назначение** 18](#_Toc170216354)

[**1.4. Функциональные требования** 20](#_Toc170216355)

[**1.5.Не функциональные требования** 22](#_Toc170216356)

[**1.6.Обоснование выбора инструментов разработки** 22](#_Toc170216357)

[**2.1.** **Модель бизнес-процесса** 26](#_Toc170216358)

[**2.2.** **Концептуальная модель** 32](#_Toc170216359)

[**2.3.** **Диаграмма деятельности (Activity diagram)** 39](#_Toc170216360)

[**14.Диаграмма база данных** 42](#_Toc170216361)

[**3.1.Требования к программному и аппаратному обеспечению** 47](#_Toc170216362)

[**3.2. Руководство программиста** 47](#_Toc170216363)

[**3.2.1. Руководство по Back-End части** 47](#_Toc170216364)

[**3.2.2. Руководство по Front-End части** 48](#_Toc170216365)

[**3.2.3. Карта сайта Web-сервиса для загрузки документов** 49](#_Toc170216366)

[**3.2.4. Библиотеки и зависимости** 50](#_Toc170216367)

[Рисунок 17. *Схема работы шаблона проектирования MVC* 51](#_Toc170216368)

[**3.2.5. Состав исходных файлов программы** 53](#_Toc170216369)

[**3.3. Руководство пользователя** 65](#_Toc170216370)

[**3.3.1. Руководство для клиентов веб-сервиса регистрации и учета пациентов клиники** 65](#_Toc170216371)

[**ЗАКЛЮЧЕНИЕ** 72](#_Toc170216372)

[**ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ** 74](#_Toc170216373)

[**ГЛОССАРИЙ** 75](#_Toc170216374)

ВВЕДЕНИЕ

Данная пояснительная записка представляет результаты разработки веб-приложения для хранения научных отчетов. Целью работы было создание универсальной платформы, позволяющей эффективно управлять различными аспектами работы с научными отчетами.

В современном научном сообществе активно используются цифровые технологии для обмена и хранения данных. Традиционные методы, такие как бумажные архивы и локальные файловые системы, становятся все менее эффективными в условиях роста объема информации и необходимости быстрого доступа к данным. Веб-приложения предоставляют централизованное хранение данных, упрощенный доступ и обмен информацией между исследователями и специалистами. Создание специализированного веб-приложения для хранения научных отчетов позволяет решить множество задач, связанных с безопасностью, удобством использования и функциональностью.

Целью данной работы было создание многофункционального веб-приложения, которое объединяет в себе следующие возможности:Надежное хранение научных отчетов с защитой данных от несанкционированного доступа.Удобный и интуитивно понятный интерфейс для управления отчетами, включая их добавление, редактирование и удаление.Возможность быстрого поиска и фильтрации отчетов по различным критериям.Поддержка различных форматов документов, таких как текстовые файлы, PDF и изображения. Интеграция с существующими информационными системами для обеспечения совместимости и обмена данными.

Задачи разработки

Для достижения поставленной цели были решены следующие задачи:

Анализ предметной области и определение требований:

Проведен анализ потребностей пользователей. Изучены существующие решения и выделены их преимущества и недостатки. Определены функциональные и нефункциональные требования к разрабатываемому приложению.

Проектирование системы: Разработана концептуальная модель системы, включающая основные модули и их взаимодействие. Созданы диаграммы UML, иллюстрирующие структуру и поведение системы. Спроектирована архитектура приложения с учетом требований к безопасности и производительности.

Разработка пользовательского интерфейса: Разработан удобный и интуитивно понятный интерфейс для управления научными отчетами. Обеспечена поддержка адаптивного дизайна для корректного отображения на различных устройствах.

Реализация функциональности: Разработаны модули для управления пользователями, научными отчетами и уведомлениями. Обеспечена поддержка различных форматов документов. Реализована функциональность многоязыковой поддержки.

Документирование:

Подготовлены руководства пользователя и разработчика. Описаны рекомендации по установке, настройке и использованию системы. Основные преимущества разработанного веб-приложения

Централизованное хранение данных: Обеспечивает легкий доступ и управление научными отчетами из любой точки мира с использованием интернет-соединения.

Интуитивно понятный интерфейс: Упрощает процесс добавления, редактирования и удаления отчетов, что делает работу с системой более эффективной.

Высокий уровень безопасности: Применяются современные методы шифрования и аутентификации, чтобы защитить данные от несанкционированного доступа.

**Проблемы, решаемые данным веб-приложением**

Упрощение доступа к научным данным: Позволяет исследователям легко находить и использовать нужные научные отчеты. Улучшение организации и управления документами: Обеспечивает удобное хранение и управление научными отчетами. Снижение рисков потери данных: Центральное хранение и регулярное резервное копирование данных обеспечивают их сохранность.

**Ожидаемые результаты:**

* Повышение эффективности работы исследователей:
* Улучшение качества хранения данных:
* Расширение возможностей для международного сотрудничества:

В дальнейшем планируется расширение функциональности веб-приложения, включающее интеграцию с новыми сервисами и улучшение пользовательского опыта. Приложение будет развиваться с учетом обратной связи от пользователей и изменения потребностей рынка.

**ГЛАВА 1. АНАЛИЗ И РАЗРАБОТКА ТРЕБОВАНИЙ**

* 1. **Обзор и анализ существующих систем учета и хранения научных отчетов**

Идея создания веб-приложения для хранения научных отчетов возникла из необходимости иметь централизованную и удобную систему для управления научной информацией. В современном научном сообществе активно используются цифровые технологии для обмена и хранения данных, однако многие существующие решения имеют недостатки в области удобства использования, безопасности и функциональности. Веб-приложение должно предоставить пользователям возможность эффективно управлять научными отчетами, обеспечивая при этом высокую степень защиты данных и удобство доступа.

* + 1. **Google Scholar**

Google Scholar – это бесплатная поисковая система, разработанная компанией Google, которая индексирует полный текст научных публикаций всех форматов и дисциплин. Она предоставляет исследователям, студентам и академическим работникам широкий доступ к научным статьям, отчетам, тезисам, книгам и другим публикациям. Google Scholar стал незаменимым инструментом для поиска научной информации благодаря своей обширной базе данных и мощным инструментам поиска.

**Плюсы Google Scholar:**

* Широкий охват: Google Scholar индексирует миллионы научных публикаций из различных источников, включая журналы, университетские репозитории и профессиональные общества. Это позволяет пользователям найти релевантную информацию практически по любой теме.
* Удобство использования: Интерфейс Google Scholar прост и интуитивно понятен, что делает его доступным для пользователей с разным уровнем технической подготовки.
* Бесплатный доступ: Платформа бесплатна для использования, что делает её доступной для всех, включая студентов и исследователей с ограниченными финансовыми ресурсами.
* Интеграция с Google аккаунтом: Пользователи могут сохранять статьи в своей библиотеке Google Scholar, следить за цитированием своих работ и получать уведомления о новых публикациях в интересующих их областях.
* Поисковые инструменты: Google Scholar предлагает мощные инструменты поиска, включая возможность поиска по автору, публикации и ключевым словам. Также доступна функция расширенного поиска для более точных результатов.

**Минусы Google Scholar:**

* Отсутствие возможности загрузки и хранения собственных отчетов: Google Scholar не предоставляет пользователям возможности загружать и хранить свои собственные научные отчеты. Пользователи могут только искать и читать уже опубликованные работы.
* Ограниченная функциональность управления документами: Платформа не обеспечивает инструментов для редактирования и управления собственными документами. Это ограничивает её использование для тех, кто хочет активно работать со своими отчетами и исследовательскими данными.
* Качество и надежность источников: Поскольку Google Scholar индексирует широкий спектр источников, качество и надежность найденных публикаций могут варьироваться. Некоторые работы могут быть не прошедшими рецензирование или опубликованными в малоизвестных журналах.
* Ограниченные возможности сортировки и фильтрации: Хотя Google Scholar предлагает основные инструменты поиска, возможности сортировки и фильтрации результатов ограничены по сравнению с некоторыми специализированными академическими базами данных.
* Отсутствие поддержки нескольких языков интерфейса: Хотя Google Scholar поддерживает поиск на различных языках, интерфейс доступен только на нескольких основных языках, что может ограничивать доступность для пользователей из разных регионов.
  1. **Обоснование необходимости разработки**

Традиционные методы хранения научных отчетов, такие как бумажные архивы и локальные файловые системы, становятся менее эффективными. Они требуют много времени на поиск и организацию документов, подвержены рискам повреждения или утраты данных и не обеспечивают удобный доступ для удаленной работы. Веб-приложение, напротив, предлагает централизованное хранение данных с надежной защитой, что минимизирует риски потери информации.В условиях удаленной работы и международного сотрудничества особенно важно иметь возможность доступа к научным отчетам из любого места и в любое время. Веб-приложение с многоязыковой поддержкой позволяет использовать систему на различных языках, что делает её удобной для пользователей из разных стран.

**Основные преимущества разработки включают:**

* Экономия времени и ресурсов:
* Удобство использования:
* Доступность и гибкость:
* Безопасность данных
* Совместная работа

Таким образом, разработка веб-приложения для хранения научных отчетов отвечает актуальным потребностям научного сообщества, улучшая управление научной информацией и повышая эффективность работы исследователе

## **1.3 Цель и назначение**

Основная цель разработки веб-приложения для хранения научных отчетов заключается в создании универсальной платформы, позволяющей эффективно управлять различными аспектами работы с научными отчетами, обеспечивая надежное хранение данных, удобный доступ к ним и поддержку многоязычной среды.

Для достижения этой цели можно выделить две группы подцелей:

**1.Улучшение производственной и управленческой функции исследовательских подразделений или научных учреждений:**

**Повышение оперативности и точности доступа к научным отчетам**:

* + Обеспечить быстрый и легкий доступ к научным отчетам из любой точки мира, используя интернет-соединение.

**Упрощение процессов загрузки, редактирования, удаления и поиска научных отчетов**:

* + Создать интуитивно понятный интерфейс, который позволяет пользователям легко выполнять все необходимые операции с отчетами.

**Обеспечение удобства использования для исследователей и научных сотрудников**:

* + Разработать адаптивный дизайн, который будет корректно отображаться на различных устройствах, включая ПК, планшеты и смартфоны.

**Снижение административной нагрузки на сотрудников**:

* + Автоматизировать управление научными отчетами, освобождая сотрудников от ручной работы с документами.

**2.Улучшение качества хранения, передачи и обработки информации**:

1. **Обеспечение надежного и безопасного хранения данных**:
   * Внедрить современные методы шифрования данных при передаче и хранении, чтобы защитить данные от несанкционированного доступа.
2. **Улучшение доступности данных за счет многоязыковой поддержки и возможности удаленного доступа**:
   * Реализовать многоязыковую поддержку, чтобы пользователи из разных стран могли легко работать с приложением на родном языке.
3. **Оптимизация процессов передачи и обмена научной информацией между пользователями**:
   * Обеспечить эффективные алгоритмы поиска и фильтрации отчетов, а также возможности для совместной работы и обмена данными.
4. **Повышение скорости обработки и поиска научных отчетов**:
   * Разработать централизованное хранилище данных и оптимизированные алгоритмы поиска, которые обеспечат быструю обработку запросов пользователей.

**1.3.1.Назначение**

Назначение веб-приложения заключается в автоматизации и оптимизации процессов управления научными отчетами, обеспечении надежного и безопасного хранения данных, а также предоставлении удобного доступа к научной информации для исследователей и научных сотрудников. Основные задачи, которые будут решены при внедрении веб-приложения, включают:

**1.Централизованное хранение научных отчетов**:

* Обеспечение единой платформы для хранения всех научных отчетов, что упрощает их поиск, управление и доступ.
* Снижение риска утраты данных за счет централизованного хранения и регулярного резервного копирования.

**2.** **Управление пользователями**:

* Поддержка регистрации и аутентификации пользователей, что обеспечивает контролируемый доступ к системе.
* Назначение ролей и прав доступа, что позволяет разграничивать возможности пользователей в зависимости от их обязанностей и потребностей.

**3. Обработка и управление документами**

* Предоставление функциональности для загрузки, редактирования, удаления и поиска научных отчетов, что упрощает работу с документами.
* Поддержка различных форматов документов, таких как текстовые файлы, PDF и т.д.

**4.Безопасность данных**

* Применение современных методов шифрования данных при передаче и хранении, что защищает данные от несанкционированного доступа.

## **1.4. Функциональные требования**

**1.4.1. Функциональные требования:**

**Аутентификация и авторизация:**

* Реализовать механизм аутентификации пользователей.
* Обеспечить многоуровневую систему авторизации с различными уровнями доступа (администратор, исследователь).

**Управление пользователями:**

* Возможность добавления, редактирования и удаления учетных записей пользователей.
* Предоставить возможность сброса пароля с использованием безопасного механизма восстановления.

**Учет и хранение научных отчетов:**

* Добавление новых научных отчетов с указанием названия, автора, даты создания и содержания.
* Редактирование и обновление существующих научных отчетов.
* Удаление научных отчетов с подтверждением действия.
* Поиск и фильтрация данных:
* Возможность поиска научных отчетов по ключевым словам, авторам, дате создания и другим параметрам.
* Реализация фильтрации данных для более удобного отображения информации.
* Система безопасности:

**Интерфейс пользователя:**

* Поддержка многопользовательского режима:
* Поисковая система для документов, книг и диссертаций и т.д.
* Функции категории для разных направлений
* Справочная информация
* Настройки для администратора
* Создание интуитивно понятного и привлекательного пользовательского интерфейса.
* Просмотр списка документов с последующим скачаиванием
* Предоставление механизма экспорта и импорта данных для облегчения переноса информации между системами.
* Функция для загрузки документов в сервер описанием автора, жанра и года выпуска
* Профиль для исследователей и админстрации

## **1.5.Не функциональные требования**

Сервер должен удовлетворять следующим критериям:

* операционные система Windows: Microsoft Windows 10/8/7.

Требования к компьютеру:

* IntelPentium III/800 МГц или выше (или аналогичный);
* минимум 4 Гб ОЗУ;
* 1 Тб свободного места на сервере;

Характеристики используемой системы:

* операционная система Microsoft Windows 10 x64;
* Intel Core i7;
* 16 Гб ОЗУ;

## **1.6.Обоснование выбора инструментов разработки**

В разработке сервиса будут применены современные технологии с использованием шаблонов проектирования. Так же после разработки проект будет размешен на удаленном сервере.

Полный список технологий, которые будут использованы при разработке проекта по ВКР:

1. **Язык программирования Java:**

Для разработки backend части информационной системы будет использоваться язык программирования Java.

Java – один из самых популярных и востребованных языков программирования в мире. Он используется в веб-разработке, разработке мобильных приложений, научных вычислениях и многих других областях. Несмотря на свою долгую историю, Java продолжает развиваться и оставаться актуальным инструментом для разработчиков. Java выбран из-за своей популярности, мощности и кроссплатформенности

1. **Фреймворк Spring:**

Spring является самым популярным инструментом при разработке промышленных приложений на Java – от больших монолитных приложений до микросервисов. Сразу стоит отметить, что под термином Spring мы понимаем уже не столько Spring Framework, сколько экосистему и Spring Boot – всё то, что позволяет нам "легко и непринужденно" использовать самые распространенные библиотеки из мира Java, зачастую даже не разбираясь в их настройках (всё настроено уже за нас).

Spring Boot настолько упрощает разработку, что достаточно часто его изучают уже с минимальными знаниями Java, воспринимая как «светлую магию» аннотации, которые «обучают» методы обычных классов обрабатывать HTTP-запросы и т.д.

В том числе, этому способствует большое количество курсов (как онлайн, так и оффлайн), где за пару десятков часов (или даже всего за пару часов) нам «отдадут» контент, содержащий все ключевые и популярные слова из вакансий: Spring, Spring Boot, REST, Hibernate, JPA, Security, JWT, OAuth2, Microservices, Kafka, Docker, Kubernetes.

Для построения backend части вашей системы будет использоваться фреймворк Spring.

* Spring обеспечивает удобную работу с зависимостями, управление транзакциями, безопасностью и другими аспектами веб-приложений.
* Spring Boot позволяет создавать автономные, самостоятельно настраиваемые приложения.

**Плюсы**

* Разработчики, переходящие с проекта на проект, используют одну и ту же экосистему.
* Новые разработчики, приходящие в команду из других компаний, уже знают Spring\*.
* Много обучающих материалов (по большей части, начального уровня).
* Существует большое количество модулей (Spring Boot, Spring MVC, Spring JPA и т.д.)

Приведённые выше пункты не нуждаются в расшифровке — достаточно посмотреть:

* Резюме большинства Java-разработчиков.
* Большинство вакансий Java-разработчиков.
* Состав учебных курсов по Java.

**Минусы**

* Отсутствие контроля. Spring Boot создает множество неиспользуемых зависимостей, что приводит к большому размеру файла развертывания.
* Сложный и трудоемкий процесс преобразования устаревшего или существующего проекта Spring в приложения Spring Boot.
* Не подходит для масштабных проектов. По мнению многих разработчиков, несмотря на отсутствие проблем при работе с микросервисами, Spring Boot не подходит для создания монолитных приложений.

Его целиком (или, по крайней мере, его части) «пихают» и тогда, когда он решает какие-то проблемы, и тогда, когда их только создаёт — кейс со Spring Security о котором пойдет речь ниже. Некоторых разработчиков впору называть не Java-разработчиками, а Spring Boot разработчиками\* – они мыслят только абстракциями Spring'а и в состоянии использовать только те инструменты, которые интегрированы со Spring'ом.

1. **Фреймворк Hibernate:**

Ни одна программа не может обойтись без подключения к базе данных. Для хранения данных используют различные базы — Oracle, MS SQL Server, MySQL, Postgres. Для подключения к базам данных в Java придумали стандарт JDBC. Он позволяет работать по единым правилам с различными базами данных одинаковыми методами. Для этого нужно лишь установить драйвер для определенной базы данных (например, для баз данных Oracle или Postgres) по стандарту JDBC.

*Hibernate* — это фреймворк, который придумали для того, чтобы облегчить жизнь программистам. Он устраняет повторяющийся код и скрывает код, необходимый для управления ресурсами, уменьшает количество ошибок. Работая с Hibernate, разработчик может сосредоточиться на бизнес-логике приложения. По сути, Hibernate при связи приложения с базой данных выступает адаптером.

Для работы с базой данных проекта будет использоваться фреймворк Hibernate.

Hibernate предоставляет возможности Object-Relational Mapping (ORM), что позволяет работать с объектами Java, а не с SQL-запросами напрямую. Он облегчает взаимодействие с базой данных и упрощает разработку и поддержку кода.

**1.6.1.Технологии:**

**Язык программирования:** Java.

**Фреймворк**: Spring (Spring MVC, Spring Data, Spring Security).

**База данных:Субд** Mysql

**Фронтенд**: HTML, CSS, JavaScript.

**Модули и функциональность**:

**Модуль аутентификации и авторизации:** Реализация безопасности с использованием Spring Security для обеспечения доступа только авторизованным пользователям.

**Модуль базы данных**: Интеграция с субд Mysql для хранения и управления научными отчетами.

**Модуль пользовательского интерфейса:** Разработка удобного и интуитивно понятного интерфейса для взаимодействия с приложением.

**Функциональные характеристики:**

* Добавление, редактирование и удаление научных отчетов.
* Аутентификация и авторизация пользователей с различными уровнями доступа.
* Отображение списка отчетов с возможностью фильтрации и поиска.

**Глава 2. Конструкторские работы**

* 1. **Модель бизнес-процесса**

IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность. Процесс моделирования системы в IDEF0 начинается с создания контекстной диаграммы — диаграммы наиболее абстрактного уровня описания системы в целом, содержащей определение субъекта моделирования, цели и точки зрения на модель.

Вход (Input) — материал или информация, которые используются или преобразуются работой для получения результата (выхода). Допускается, что работа может не иметь ни одной стрелки входа. Каждый тип стрелок подходит к определенной стороне прямоугольника, изображающего работу, или выходит из нее. Стрелка входа рисуется как входящая в левую грань работы. Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить информация о том, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то, скорее всего, это вход, если нет — управление.

Управление (Control) — правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелка управления рисуется как входящая в верхнюю грань работы. Управление влияет на работу, но не преобразуется работой. Если цель работы — изменить процедуру или стратегию, то такая процедура или стратегия будет для работы входом. В случае возникновения неопределенности в статусе стрелки (управление или вход) рекомендуется рисовать стрелку управления.

Выход (Output) — материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Стрелка выхода рисуется как исходящая из правой грани работы.

Механизм (Mechanism) — ресурсы, которые выполняют работу, например персонал предприятия, станки, устройства и т. д. Стрелка механизма ри8 суется как входящая в нижнюю грань работы. По усмотрению аналитика стрелки механизма могут не изображаться в модели.

Эта контекстная диаграмма IDEF0 и ее детализированные элементы предоставляют полное представление о системе управления научными отчетами. Она показывает, как данные пользователей и отчеты обрабатываются системой, какие результаты производятся, какие правила управляют процессами и какие ресурсы используются.

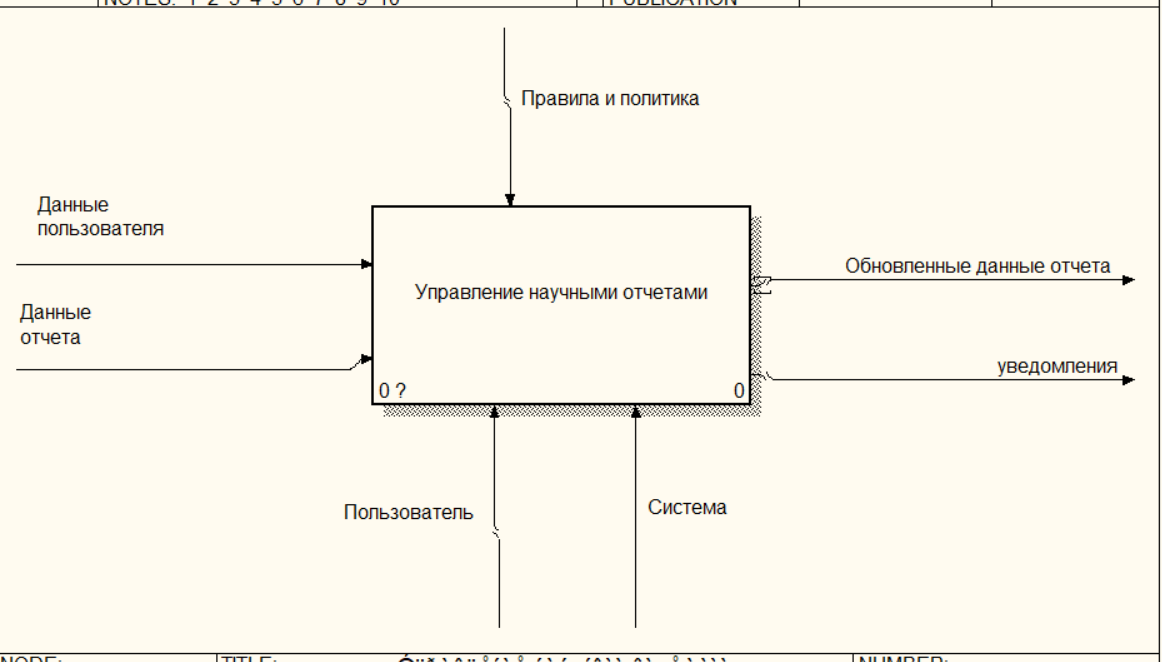


Рисунок 1.. Бизнес-процесс «Управление научными отчетами» (Модель AS-IS).

Таблица 1. Описание контекстной диаграммы верхнего уровня.

|  |  |  |  |
| --- | --- | --- | --- |
| № | **Наименование** | **Тип** | **Описание** |
| 1 | Данные пользователя | Вход | Имя, фамилия, электронная почта |
| 2 | Данные отчета | Вход | |  | | --- | |  | | Название отчета, авторы, дата создания, содержание. | | |
| 3 | Правила и политика | Управление | Правила и политика, определяющие условия использования системы и обработки данных. |
| 4 | Пользователь | Механизм | Исследователь или администратор, взаимодействующий с системой. |
| 5 | Система | Управление | Внешние системные вызовы или взаимодействия, необходимые для работы системы. |
| 6 | Обновленные данные отчета | Выход | Обновленная информация о научных отчетах после выполнения операций. |
| 7 | Уведомления | Выход | Сообщения для пользователей о выполненных операциях, изменениях или ошибках. |
| 8 | Администратор | Механизм | Администратор, который имеет права на управление всей системой. |

После описания контекстной диаграммы проводится функциональная декомпозиция – система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема, при необходимости, разбивается на более мелкие и так далее до достижения нужной степени подробности.

Контекстная диаграмма была разбита на 4 блока (рисунок 3):

1. Зарегистрироваться в системе.
2. Добавление нового научного отчета.
3. Редактирование отчета.
4. Уведомления.

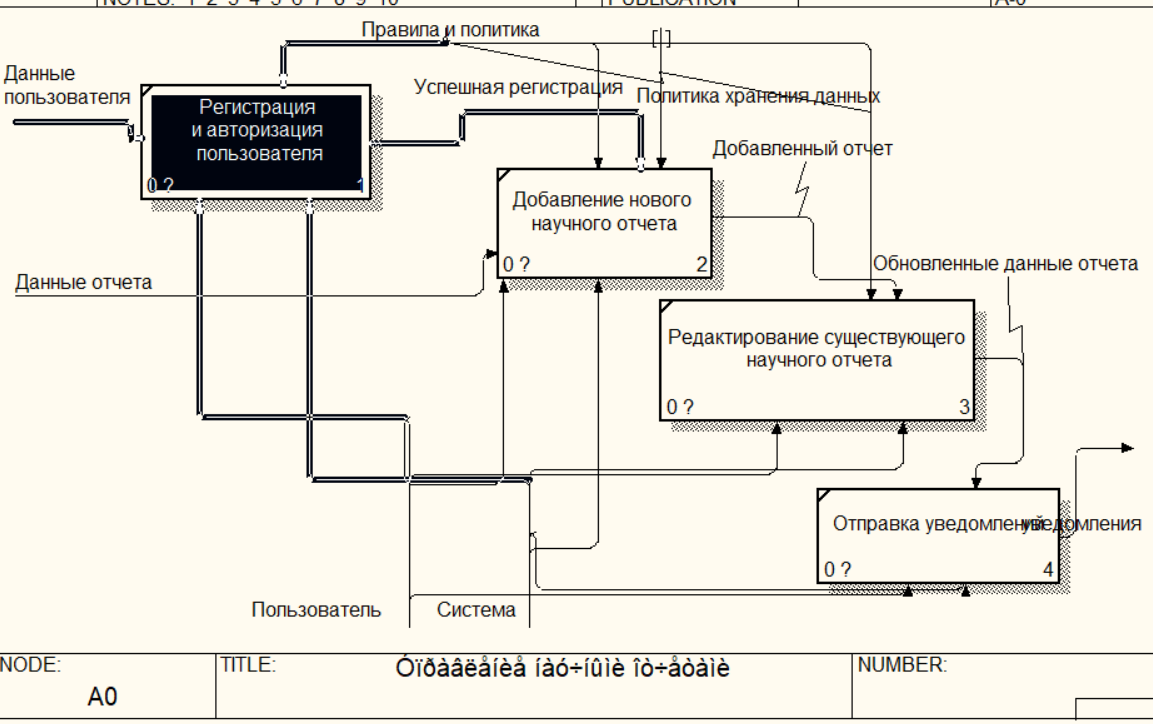


Рисунок 2. Декомпозиция диаграммы верхнего уровня.

Таблица 2. Описание контекстной диаграммы второго уровня.

|  |  |  |  |
| --- | --- | --- | --- |
| № | **Наименование** | **Тип** | **Описание** |
| 1 | Данные пользователя | Вход | Имя, фамилия, электронная почта, |
| 2 | Данные отчета | Вход | Название отчета, авторы, дата создания, содержание. |
| 3 | Правила и политика | Управление | Это набор инструкций и стандартов, которыми система руководствуется при выполнении своих функций. Они могут включать:  Политики безопасности (например, требования к паролям, меры для защиты данных пользователя).  Политики хранения данных (например, форматы и максимальные размеры файлов).  Правила редактирования (например, кто имеет право редактировать отчеты и как эти изменения должны быть отслежены). |
| 4 | Логин пароль | Вход | Логин и пароль предназначается для входа в систему |
| 5 | Пользователь: | Механизмы | Люди, которые взаимодействуют с системой. Пользователи могут регистрироваться в системе, входить в нее, добавлять новые отчеты, редактировать существующие и получать уведомления. |
| 6 | |  | | --- | |  |  |  | | --- | | Система | | Управление | Программное обеспечение и технические средства, которые обеспечивают функционирование системы. Система обрабатывает данные, управляет доступом пользователей, сохраняет отчеты и отправляет уведомления. |
| 7 | |  | | --- | |  | | Обновленные данные отчета | | | Выход | Это данные, которые система производит после обработки входных данных. Они могут включать новые научные отчеты, добавленные пользователем, или изменения в существующих отчетах. |
| 8 | Уведомления | Выход | Сообщения, которые система отправляет пользователям. Эти уведомления могут включать подтверждения успешной регистрации, добавления или редактирования отчета, а также сообщения об ошибках или других событиях. |

* 1. **Концептуальная модель**

Для того чтобы проанализировать и специфицировать общие требования к функциональному поведению проектируемой системы была разработана концептуальная модель. Данная модель описывает список всех пользователей системы, а также цели, которые они преследуют при её использовании.Каждый вариант использования определяет деятельность, которая должна быть выполнена проектируемой системой при взаимодействии ее с соответствующим актером. Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.

На Рисунок ‎3 представлена диаграмма Use Case, которая демонстрирует взаимодействие пользователя и администратора с системой.



Рисунок 3. Диаграмма вариантов использования

**Таблица 3: Регистрация**

|  |  |
| --- | --- |
| **Идентификатор и название:** | **UC-1 Регистрация** |
| Основное действующее лицо: | Пользователь |
| Описание: | Пользователь вводит свои данные для регистрации. |
| Триггер: | Пользователь открывает веб-приложение. |
| Предварительные условия: | PRE-1. Пользователь не зарегистрирован в системе. |
| Выходные условия: | POST-1. Пользователь успешно зарегистрирован и получает подтверждение. |
| Нормальное направление развития варианта использования: | 1. Пользователь открывает страницу регистрации.  2. Отображается форма для заполнения данных регистрации.  3. Пользователь вводит свои данные и отправляет форму.  4. Система проверяет данные и создаёт новую учетную запись.  5. Пользователь получает уведомление об успешной регистрации. |
| Исключения: | E1. Если пользователь ввёл некорректные данные  1.1. Система отображает сообщение об ошибке. |

Таблица 4: **Авторизация**

|  |  |
| --- | --- |
| **Идентификатор и название:** | UC-2 Авторизация |
| Основное действующее лицо: | Пользователь |
| Описание: | Пользователь вводит свои учетные данные для входа. |
| Триггер: | Пользователь открывает веб-приложение. |
| Предварительные условия: | PRE-1. Пользователь зарегистрирован в системе. |
| Выходные условия: | POST-1. Пользователь успешно авторизован и получает доступ к системе. |
| Нормальное направление развития варианта использования: | 1. Пользователь открывает страницу авторизации.  2. Отображается форма для ввода учетных данных.  3. Пользователь вводит свои данные и отправляет форму.  4. Система проверяет данные и предоставляет доступ.  5. Пользователь получает доступ к системе. |
| Исключения: | E1. Если пользователь ввёл некорректные данные  1.1. Система отображает сообщение об ошибке. |

**Таблица 5: Добавление документа**

|  |  |
| --- | --- |
| **Идентификатор и название:** | **UC-3 Добавление документа** |
| Основное действующее лицо: | Пользователь |
| Описание: | Пользователь вводит данные нового документа. |
| Триггер: | Пользователь открывает веб-приложение. |
| Предварительные условия: | PRE-1. Пользователь авторизован в системе. |
| Выходные условия: | POST-1. Документ успешно добавлен и пользователь получает подтверждение. |
| Нормальное направление развития варианта использования: | 1. Пользователь открывает страницу добавления документа.  2. Отображается форма для заполнения данных документа.  3. Пользователь вводит данные и отправляет форму.  4. Система проверяет данные и сохраняет новый документ.  5. Пользователь получает уведомление об успешном добавлении документа. |
| Исключения: | E1. Если пользователь ввёл некорректные данные 1.1. Система отображает сообщение об ошибке. |

**Таблица 6: поиск**

|  |  |
| --- | --- |
| **Идентификатор и название:** | UC-4 Поиск |
| Основное действующее лицо: | Пользователь, Администратор |
| Описание: | Пользователь вводит запрос для поиска документов. |
| Триггер: | Пользователь или администратор открывает веб-приложение. |
| Предварительные условия: | PRE-1. Пользователь или администратор авторизован в системе. |
| Выходные условия: | POST-1. Результаты поиска успешно отображены. |
| Нормальное направление развития варианта использования: | 1. Пользователь или администратор открывает страницу поиска.  2. Отображается форма для ввода поискового запроса.  3. Пользователь или администратор вводит запрос и отправляет форму.  4. Система выполняет поиск и отображает результаты. |
| Исключения: | E1. Если поиск не дал результатов  1.1. Система отображает сообщение о том, что ничего не найдено. |

**Таблица 7: обновить**

|  |  |
| --- | --- |
| **Идентификатор и название:** | UC-5 Обновить |
| Основное действующее лицо: | Администратор |
| Описание: | Администратор редактирует существующий документ. |
| Триггер: | Администратор открывает веб-приложение. |
| Предварительные условия: | PRE-1. Администратор авторизован в системе. |
| Выходные условия: | POST-1. Документ успешно обновлен и администратор получает подтверждение. |
| Нормальное направление развития варианта использования: | 1. Администратор открывает страницу редактирования документа.  2. Отображается форма с текущими данными документа.  3. Администратор вносит изменения и отправляет форму.  4. Система проверяет данные и сохраняет изменения.  5. Администратор получает уведомление об успешном обновлении документа. |
| Исключения: | E1. Если администратор ввёл некорректные данные  1.1. Система отображает сообщение об ошибке. |

**Таблица 8: удалить**

|  |  |
| --- | --- |
| **Идентификатор и название:** | UC-6 Удалить |
| Основное действующее лицо: | Администратор |
| Описание: | Администратор удаляет существующий документ. |
| Триггер: | Администратор открывает веб-приложение. |
| Предварительные условия: | PRE-1. Администратор авторизован в системе. |
| Выходные условия: | POST-1. Документ успешно удален и администратор получает подтверждение. |
| Нормальное направление развития варианта использования: | 1. Администратор открывает страницу удаления документа.  2. Администратор выбирает документ для удаления.  3. Администратор подтверждает удаление.  4. Система удаляет документ из базы данных.  5. Администратор получает уведомление об успешном удалении документа. |
| Исключения: | E1. Если документ не найден  1.1. Система отображает сообщение об ошибке. |

Таблица 9: Добавить пользователя

|  |  |
| --- | --- |
| **Идентификатор и название:** | UC-7 Добавить нового пользователя |
| Основное действующее лицо: | Администратор |
| Описание: | Администратор вводит данные нового пользователя. |
| Триггер: | Администратор открывает веб-приложение. |
| Предварительные условия: | PRE-1. Администратор авторизован в системе. |
| Выходные условия: | POST-1. Новый пользователь успешно добавлен и администратор получает подтверждение. |
| Нормальное направление развития варианта использования: | 1. Администратор открывает страницу добавления пользователя. <br> 2. Отображается форма для ввода данных нового пользователя. <br> 3. Администратор вводит данные и отправляет форму. <br> 4. Система проверяет данные и создаёт новую учетную запись. <br> 5. Администратор получает уведомление об успешном добавлении пользователя. |
| Исключения: | E1. Если администратор ввёл некорректные данные <br> 1.1. Система отображает сообщение об ошибке. |

* 1. **Диаграмма деятельности (Activity diagram)**

Диаграммы деятельности (Activity diagram), называемые также диаграммами активности или диаграммами видов деятельности, были введены в язык UML сравнительно недавно. Диаграмма деятельности — это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения

**Авторизация**

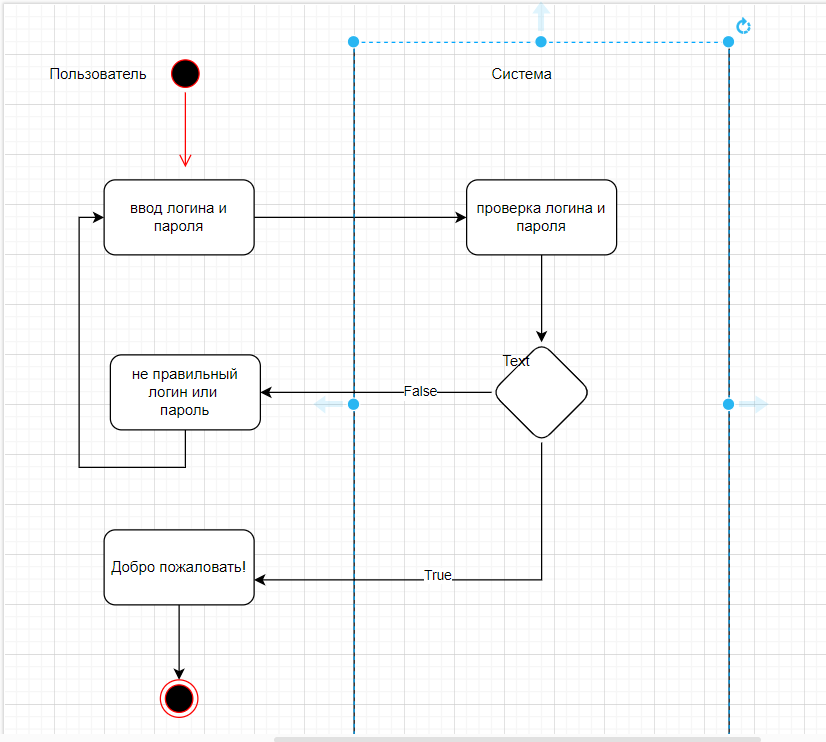


Рисунок 4. Авторизация

**Загрузка документа**

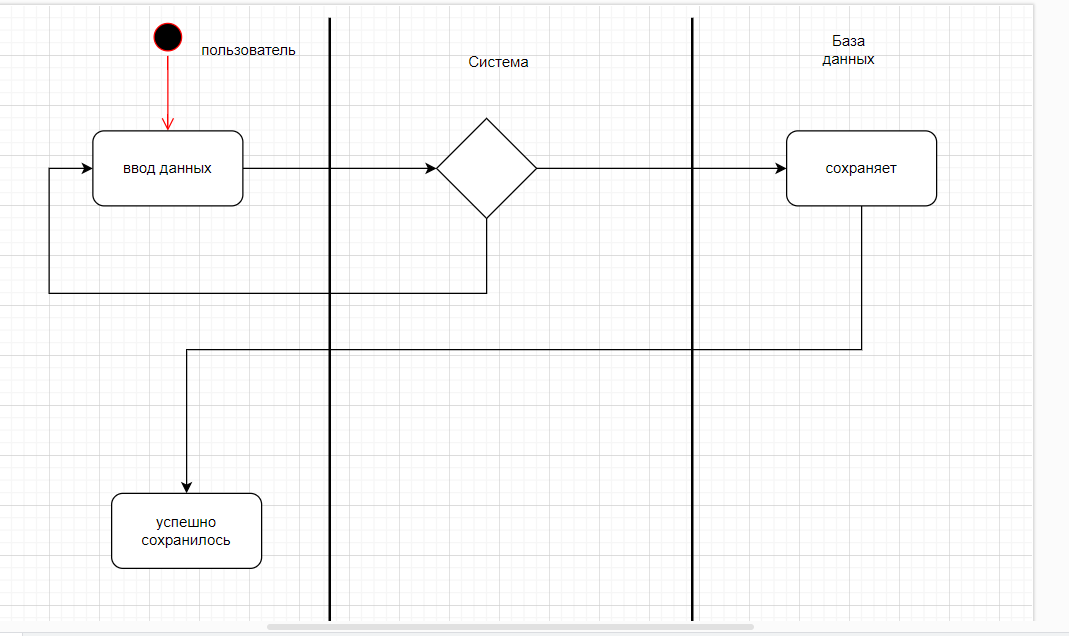


Рисунок 5.Загрузка документа

**Поиск научных документов**

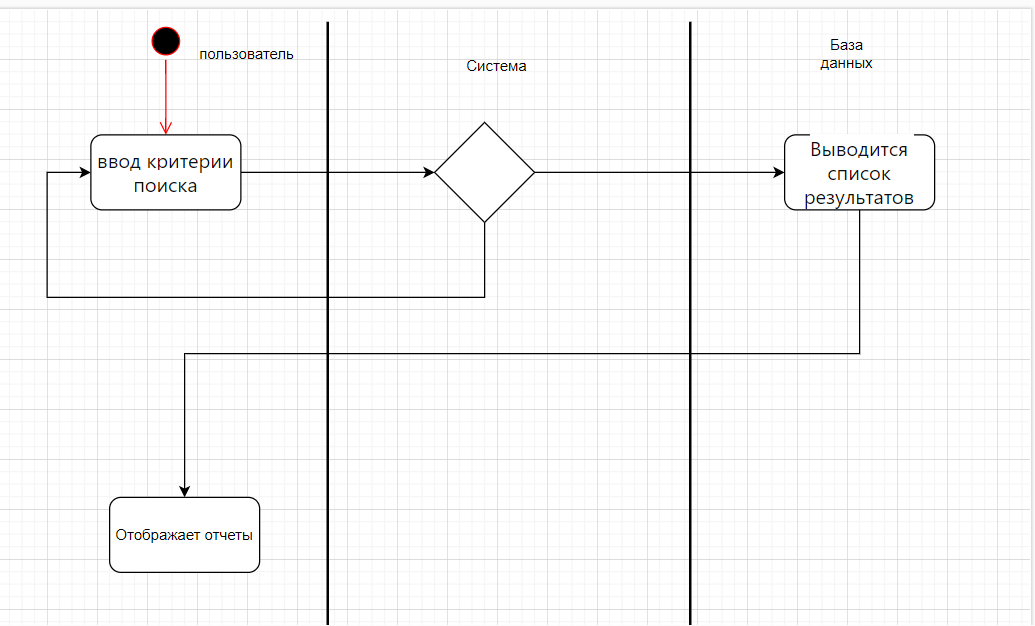


Рисунок 6. Поиск научных документов

## **14.Диаграмма база данных**

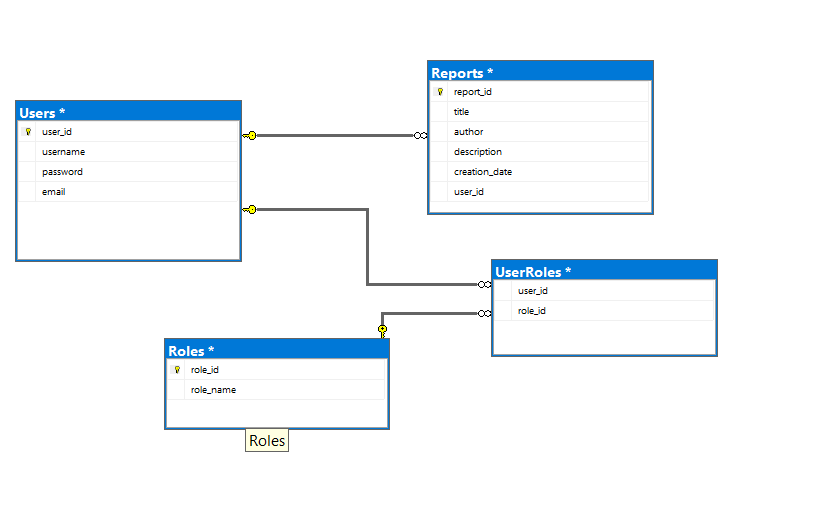


Рисунок 7. Диаграмма база данных

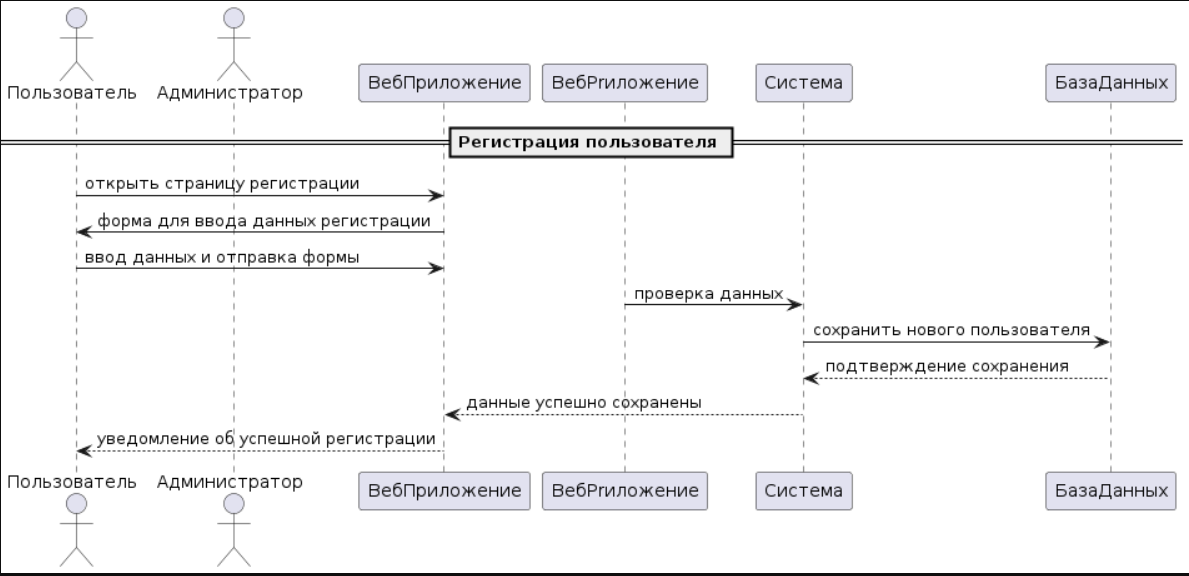
Диаграмма последовательности: Регистрация  
  


Рисунок 8. Диаграмма последовательности: Регистрация

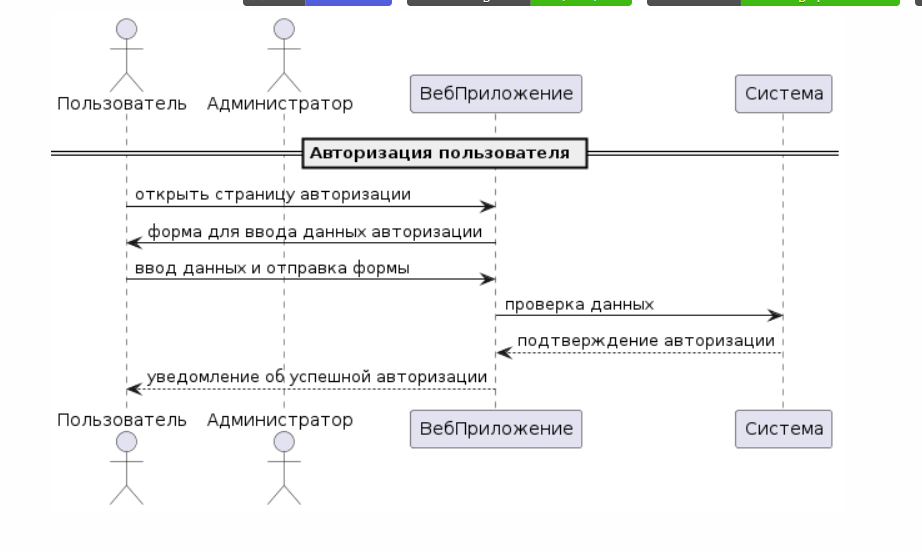
Диаграмма последовательности : авторизация  
  


Рисунок 9. Диаграмма последовательности: авторизация

Диаграмма последовательности: добавление документа

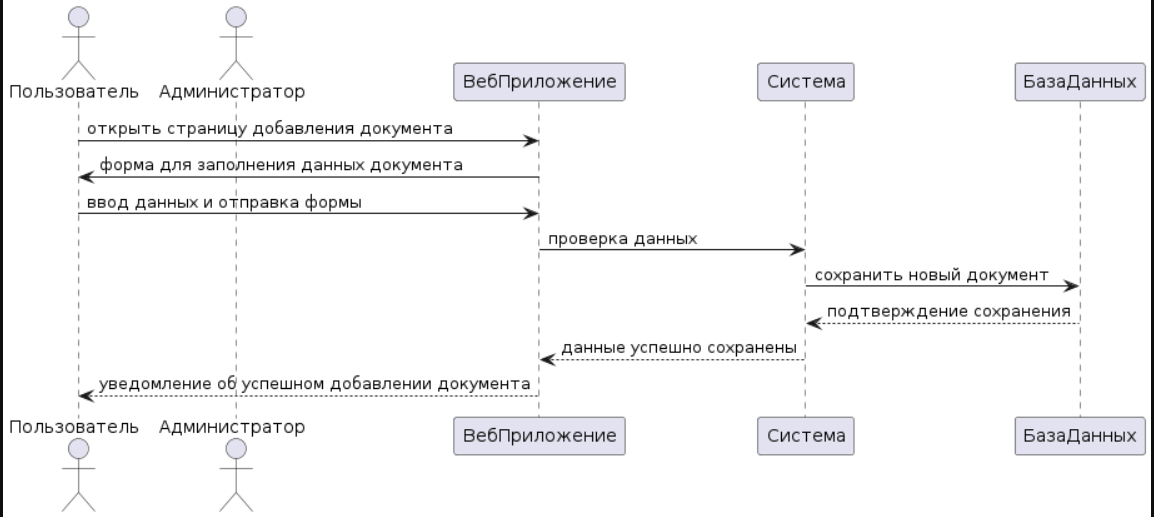


Рисунок 10. Диаграмма последовательности: Добавление документа

**Диаграмма последовательности : редактирование документов**

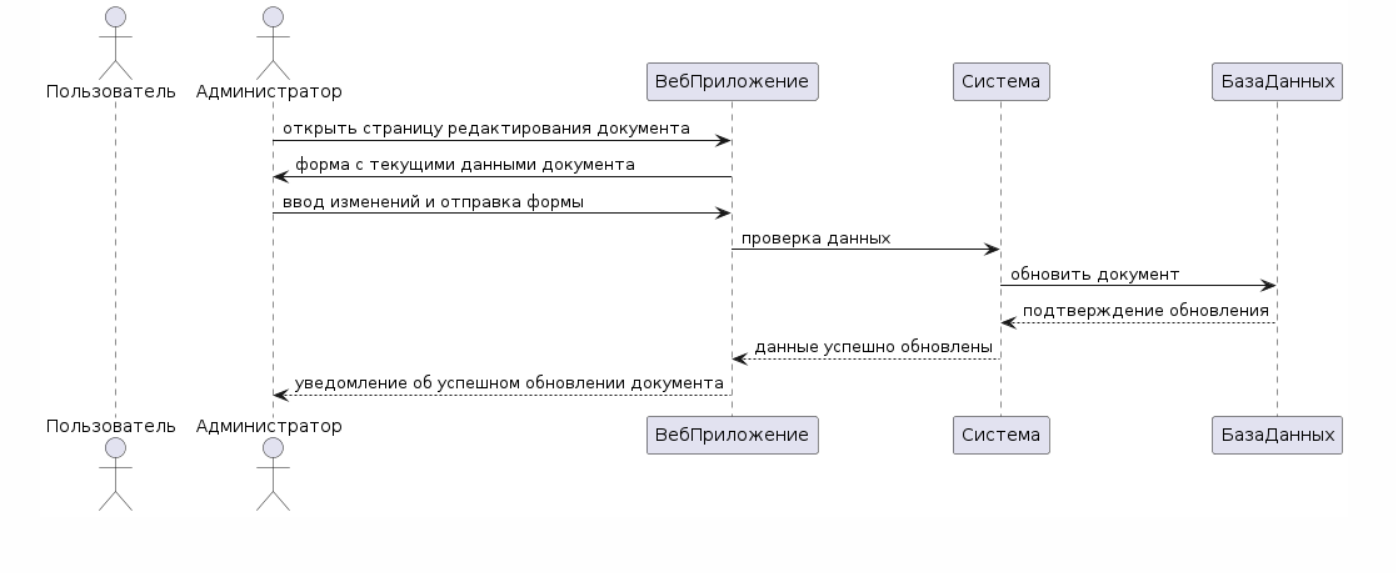


Рисунок 11. Диаграмма последовательности: Редактирование документа

**Диаграмма последовательности: удаление документа**

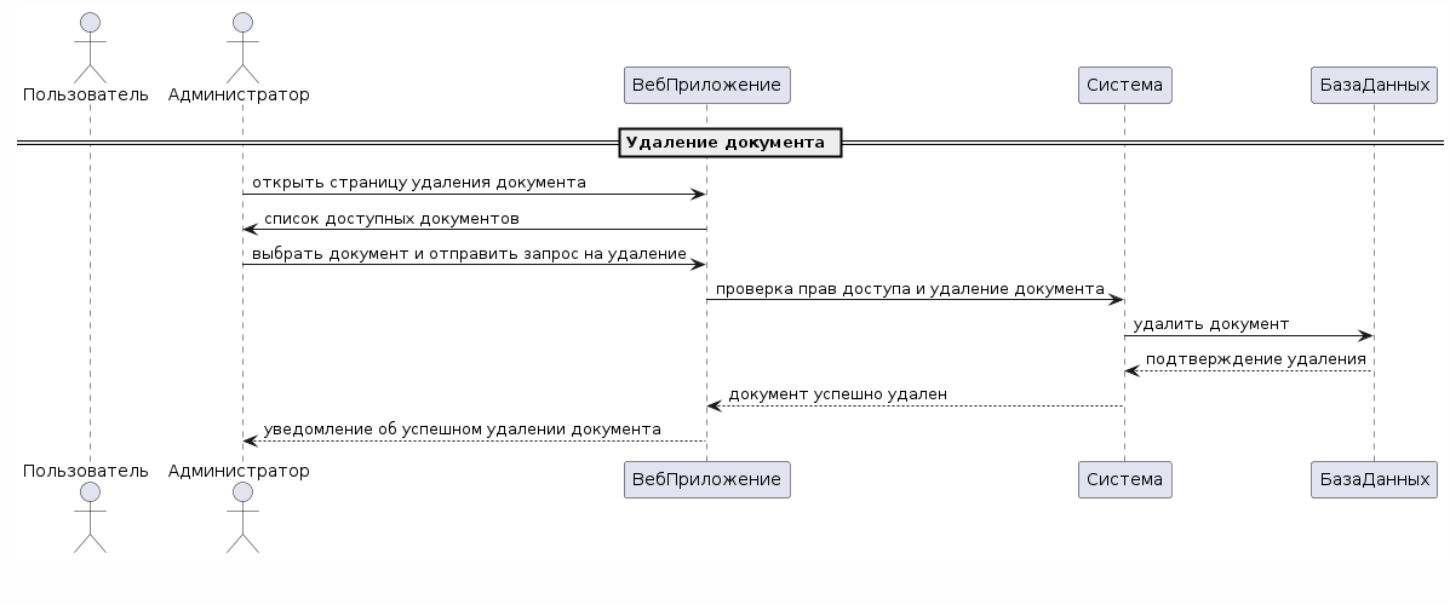


Рисунок 12. Диаграмма последовательности: Регистрация

**Диаграмма классов**

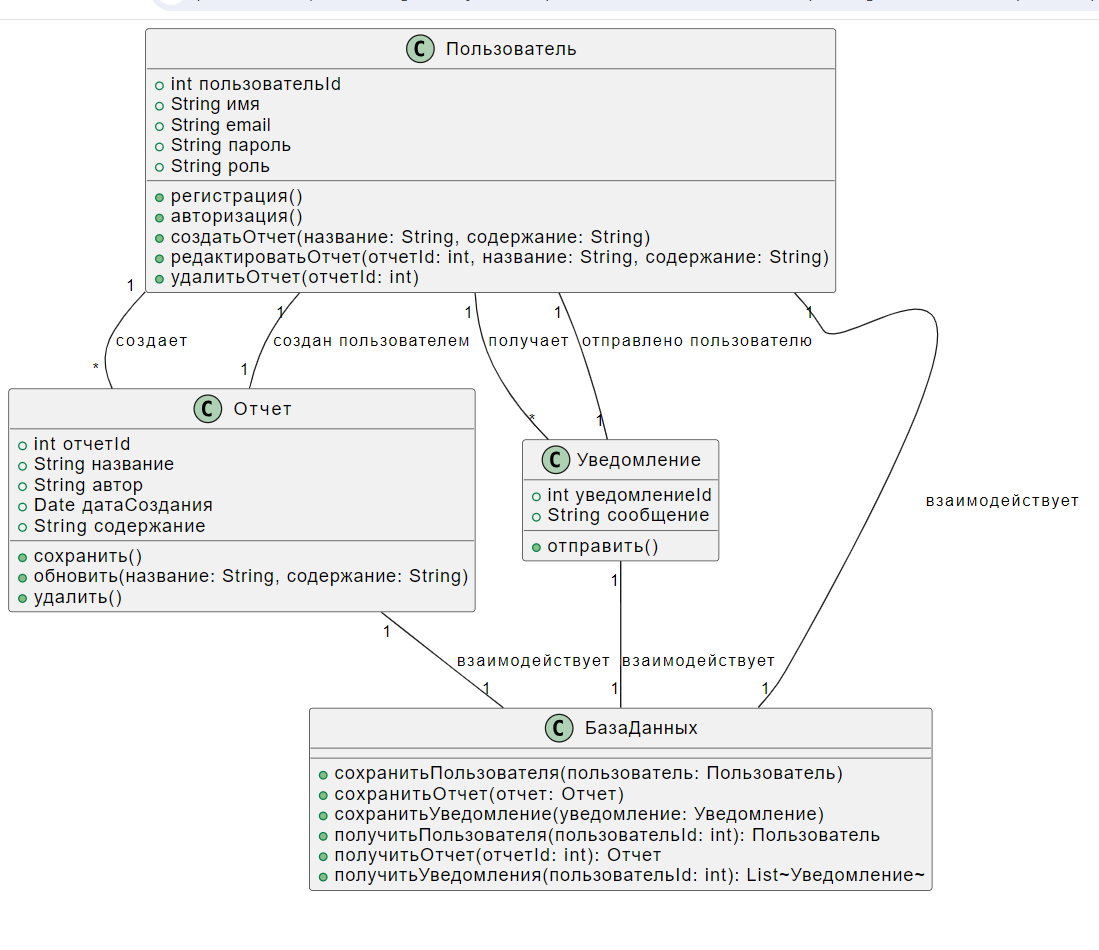


Рисунок 13. Диаграмма классов

Диаграмма компонентов



**Рисунок 14. Диаграмма компонентов**

**Диаграмма развертывания**



**Рисунок 15. Диаграмма развертывания**

**ГЛАВА 3. РАЗРАБОТКА ДОКУМЕНТАЦИИ**

**3.1.Требования к программному и аппаратному обеспечению**

В состав минимальных технических средств сервера для запуска серверной и клиентской части системы и запуска базы данных необходимо следующее оборудование:

* ОС: Linux, Windows;
* Процессор AMD EPYC 7000 series processor with an all core turbo clock speed of up to 2.5
* Оперативная память: минимум 8GB;
* Дисковое пространство: минимум 100GB

**3.2. Руководство программиста**

Данное руководство описывает структуру проекта, паттерны проектирования, которые были использованы при разработке системы, а также содержит нюансы разработки.

### **3.2.1. Руководство по Back-End части**

Для реализации Back-end части был использован язык программирования Java и Framework Spring. Разработка базы данных велась с помощью СУБД PostgreSQL. В качестве среды разработки использовалась IntelliJ IDEA.

Проект придерживается чистой архитектуры с использованием MVC (Model View Controller) паттерна.

MVC расшифровывается как «модель-представление-контроллер» (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

**Компоненты MVC:**

* Модель — Этот компонент отвечает за данные и бизнес-логику приложения. Модель управляет данными приложения, логикой и правилами взаимодействия с ними. Например, в приложении для хранения научных отчетов модель будет содержать классы, представляющие отчеты и пользователей, а также методы для взаимодействия с базой данных.

**Пример**: Класс Report, который включает свойства отчета и методы для сохранения, обновления и удаления отчетов.

* Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования. **Пример**: HTML-страницы или шаблоны, которые отображают списки отчетов, формы для добавления новых отчетов и т.д
* Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

### **3.2.2. Руководство по Front-End части**

Общие сведения:

* Тип web-приложения: Single-Page Application (SPA).
* Предназначение: Автоматизация процессов в организации клиники. Инструменты: Создание Web-сервиса по регистрации и учета пациентов клиники
* Основной инструмент на момент разработки приложения — JavaScript
* UI-инструмент: Material-UI (5.11.0).
* Архитектура Feature-Sliced Design (FSD).

В методологии FSD используются три уровня абстракции:

1) *Слои*. Этот уровень определяет масштаб ответственности слоя, а также уровень опасности изменений. Чем выше расположен слой, тем выше уровень его ответственности и знаний о других слоях. Чем ниже расположен слой, тем он более абстрактный и больше используется в верхних слоях, а значит больше опасности вносить в него изменения.

2) *Слайсы*. Они отображают конкретную функциональность бизнес-логики. Методология почти не влияет на этот уровень, многое зависит от конкретного проекта. По факту это папки, которые группируют файлы, реализующие тот или иной модуль.

3) *Сегменты*. Распределяются по назначению модуля в коде и реализации. По методологии каждый сегмент отвечает за свою часть технической реализации модуля:

api/ — работа с API.

Lib/ — различные утилитарные функции и вспомогательные библиотеки.

Config/ — конфигурация модуля.

Model/ — бизнес-логика: actions, effects, reducers и так далее.

Ui/ — отвечает за отображение.

### **3.2.3. Карта сайта Web-сервиса для загрузки документов**

Карта сайта позволяет эффективно сканировать и индексировать содержание сайта поисковым системам. На карте сайте отображены все существующие страницы на сайте. На рисунке 16 представлена карта сайта загрузки документов.

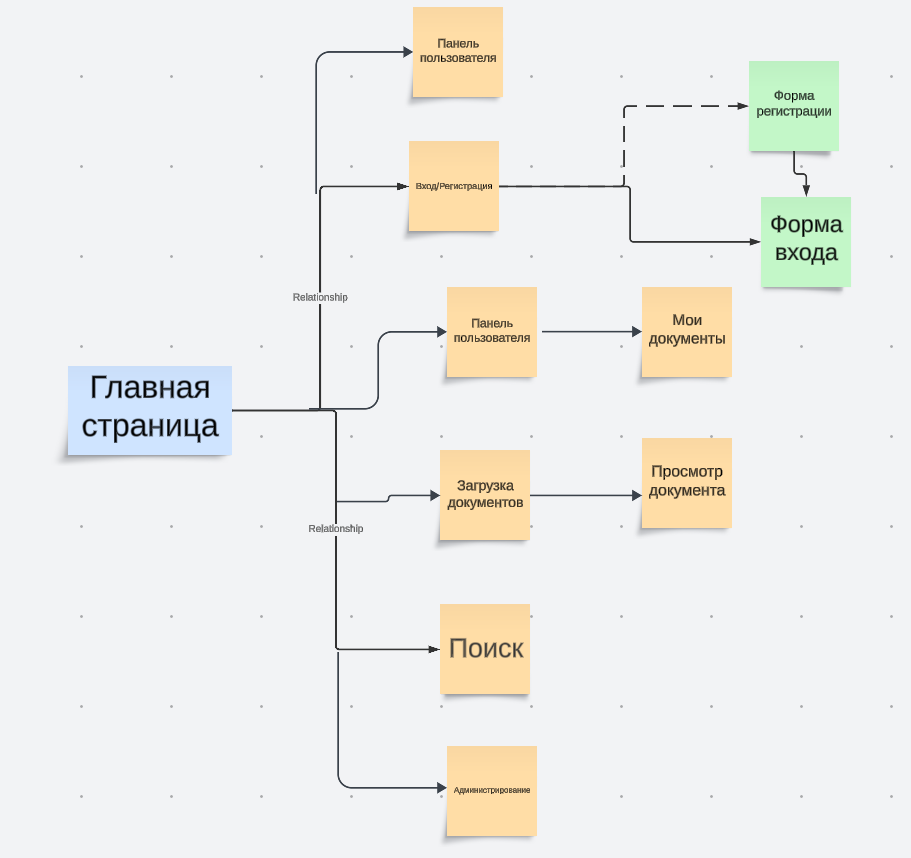


Рисунок 16.Карта сайта загрузка документов

### **3.2.4. Библиотеки и зависимости**

В проекте используется платформа Spring Framework, которая обеспечивает широкие возможности для создания современных Java-приложений. Средство сборки проекта — Maven, что позволяет управлять зависимостями и конфигурацией проекта эффективно и удобно.

Для установки этих зависимостей необходимо создать файл «pom.xml», с помощью которого погрузятся необходимые зависимости. Содержимое файла «pom.xml» показано на рисунке 18.

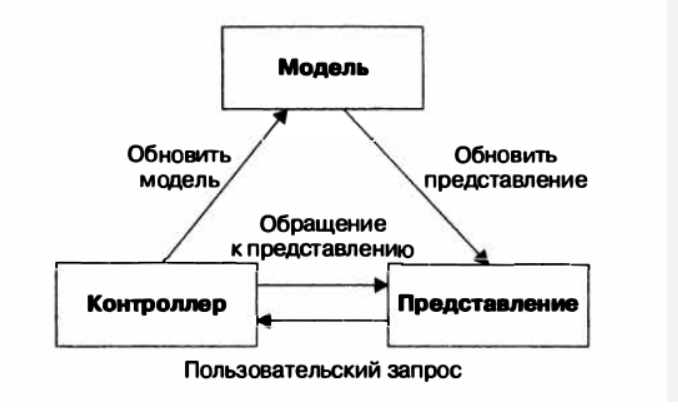
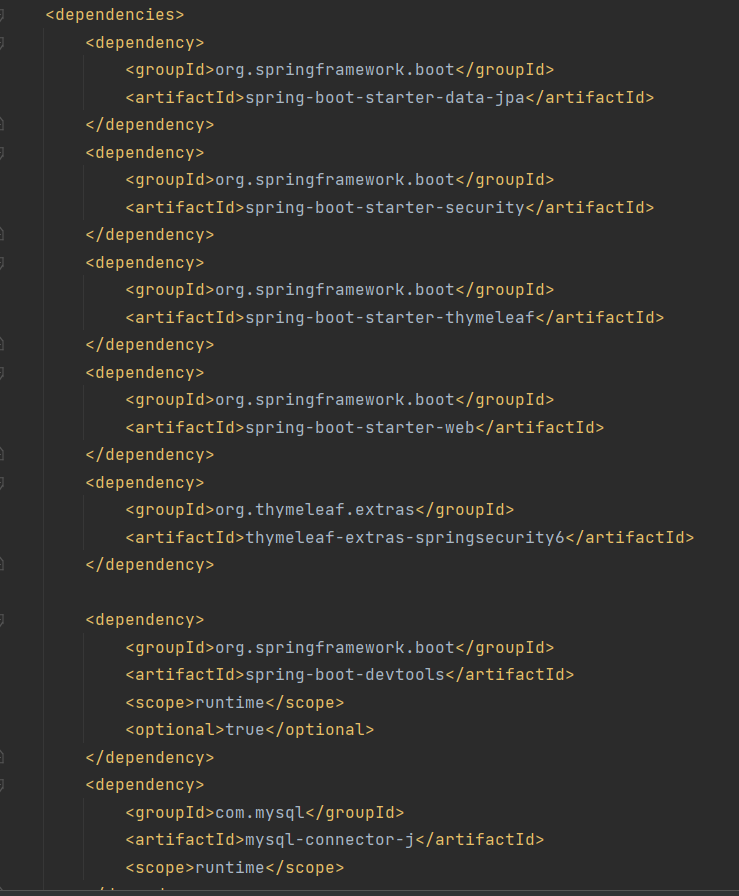


Рисунок 17. *Схема работы шаблона проектирования MVC*



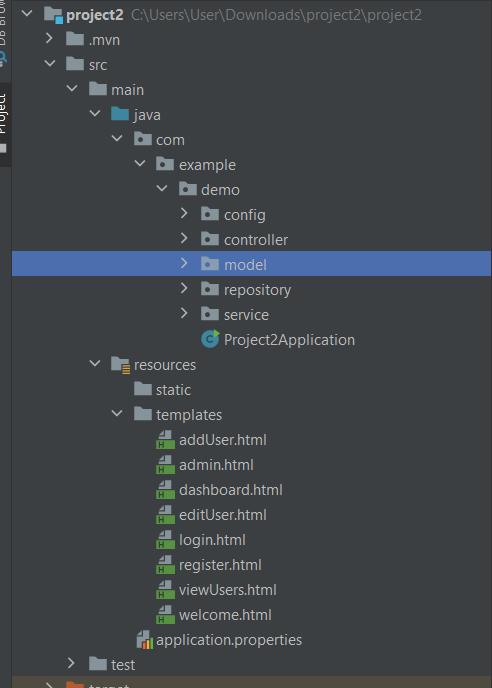
*Рисунок 18. Зависимости необходимые для работы проекта*



*Рисунок 19. Конфигурация проекта*

### **3.2.5. Состав исходных файлов программы**

Структура файлов проекта показано на рисунках 18-28. Главная директория проекта содержит следующие каталоги и файлы:

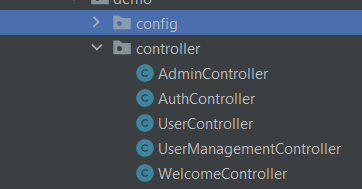


*Рисунок 20. Список папок главной директории проекта*

Таблица 10. Описание папок главной директории проекта

|  |  |
| --- | --- |
| Название папки | Описание |
| config | Этот каталог содержит все конфигурационные классы приложения. Он используется для централизации и управления настройками, связанными с безопасностью, конфигурациями баз данных, и другими важными параметрами, которые определяют поведение приложения. |
| controllers | В этом каталоге размещаются классы контроллеров, которые обрабатывают запросы от пользователей, управляют потоком данных между моделями и представлениями, и возвращают подходящие ответы. Контроллеры являются ключевой частью паттерна проектирования MVC (Model-View-Controller) |
| models | Содержит классы, которые описывают структуру данных приложения и могут включать методы для валидации данных. Эти модели представляют сущности, с которыми оперирует приложение. |
| repository | В этом каталоге находятся репозитории, которые предоставляют более высокоуровневый интерфейс для запросов к базе данных. Репозитории помогают организовать доступ к данным, используя паттерн Repository для облегчения тестирования и повышения модульности кода. |
| service | Каталог для сервисных классов, которые содержат бизнес-логику приложения. Сервисы обрабатывают сложные задачи и бизнес-операции, вызывая DAO или репозитории для доступа к данным и передавая результаты обратно в контроллеры. |
| resource | Этот каталог включает в себя необходимые ресурсы, такие как файлы свойств, XML-конфигурации, скрипты для базы данных и другие файлы, которые используются для конфигурации и выполнения приложения. |
| webapp | Содержит элементы веб-интерфейса приложения, включая JSP-файлы, HTML-страницы, клиентские скрипты (JavaScript), стили (CSS) и изображения. Это каталог для всех веб-ресурсов, которые напрямую предоставляются пользователям. |

Контроллеры отвечают за маршрутизацию в проекте. На рисунке 20 изображены классы, которые отвечают за маршрутизацию в проекте



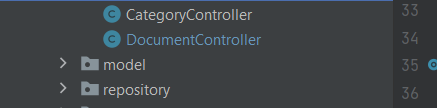
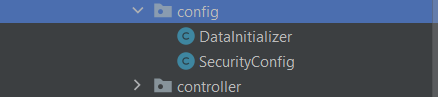


Рисунок 21. Список классов папки controller

Таблица 11. Описание содержание папки controllers

|  |  |
| --- | --- |
| Название | Описание |
| AdminController | Этот контроллер отвечает за административные функции в приложении. Он позволяет администраторам управлять учетными записями пользователей, назначать роли и права доступа, а также выполнять другие задачи, требующие административных привилегий. |
| AuthController | Этот контроллер занимается аутентификацией и авторизацией пользователей. Он обрабатывает запросы на вход в систему (логин), выход из системы (логаут), а также другие операции, связанные с проверкой подлинности пользователей. |
| UserController | Этот контроллер предоставляет функциональность для управления данными пользователя. Он позволяет пользователям просматривать и редактировать свои профили, а также выполнять другие действия, связанные с их учетной записью |
| UserManagementController | Этот контроллер используется для управления учетными записями пользователей. Он позволяет администраторам создавать новые учетные записи, обновлять информацию о существующих пользователях, а также удалять учетные записи по мере необходимости. |
| WelcomeController | Этот контроллер обрабатывает запросы на главную страницу приложения и предоставляет общую информацию о сайте. Он может отображать приветственное сообщение, новости и другую важную информацию для пользователей. |
| DocumentController | Этот контроллер предоставляет функциональность для управления документами. Он позволяет пользователям загружать новые документы, редактировать существующие, удалять ненужные документы и выполнять поиск по различным критериям. Контроллер обеспечивает основные операции CRUD (Create, Read, Update, Delete) для документов. |
| CategoryController | то контроллер, который отвечает за управление категориями документов в веб-приложении. Он обеспечивает основные операции CRUD (Create, Read, Update, Delete) для категорий и взаимодействует с клиентами через REST API. |

В папке config содержатся все конфигурации проекта



*Рисунок 22. Список классов папки config*

Таблица 12. Описание содержание папки config

|  |  |
| --- | --- |
| Название | Описание |
| DataInitializer | Этот конфигурационный класс, который инициализирует базу данных начальными данными. Он проверяет наличие определенных пользователей в базе данных и создает их, если они отсутствуют. **Аннотация @Configuration**: Указывает, что этот класс является конфигурационным и должен быть обработан контейнером Spring для создания бинов |
| SecurityConfig | этот конфигурационный класс, который отвечает за настройку безопасности приложения с использованием Spring Security. Он включает в себя настройку аутентификации, авторизации, шифрования паролей и обработку успешной аутентификации.  **Аннотация @Configuration**: Указывает, что этот класс является конфигурационным.   **Аннотация @EnableWebSecurity**: Включает поддержку веб-безопасности в приложении. |

**DataInitializer** и **SecurityConfig** работают вместе для обеспечения инициализации данных и настройки безопасности в веб-приложении. Вот как они взаимодействуют:

DataInitializer

**Инициализация данных при запуске приложения**:

1. Когда приложение запускается, Spring создает объекты всех конфигурационных классов, включая DataInitializer.
2. DataInitializer содержит метод init(), который выполняется сразу после создания объекта этого класса.
3. Метод init() проверяет, есть ли в базе данных пользователь с именем "user". Если такого пользователя нет, он создает нового пользователя с этим именем и сохраняет его в базе данных. Пароль пользователя шифруется перед сохранением для обеспечения безопасности.

SecurityConfig

**Настройка безопасности**:

1. Spring также создает объект класса SecurityConfig при запуске приложения.
2. Класс SecurityConfig отвечает за настройку безопасности в приложении, используя Spring Security.
3. Он определяет, какие URL-адреса доступны всем пользователям, а какие требуют аутентификации (входа в систему).
4. Также он настраивает форму входа и выхода из системы, указывая, какие страницы показывать при входе и выходе.
5. SecurityConfig включает метод для шифрования паролей, чтобы пароли сохранялись в базе данных в зашифрованном виде.
6. Он также настраивает загрузку данных пользователя из базы данных при входе в систему.

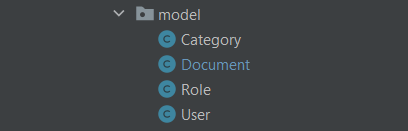
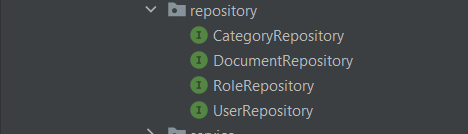


Рисунок 23. Список классов папки model

Таблица 13. Описание содержание папки models

|  |  |
| --- | --- |
| **Название класса** | **Описание** |
| Category | |  | | --- | |  |  |  | | --- | | Этот класс представляет категории для организации документов. Он включает атрибуты, такие как уникальный идентификатор категории (id), название категории (name) и набор документов, связанных с этой категорией (documents). | |
| Document | Этот класс представляет научные отчеты или документы. Он включает атрибуты, такие как уникальный идентификатор документа (id), название документа (documentName), путь к файлу (path), автор документа (author), описание (description), категория документа (category) и дата создания (creationDate). |
| Role | |  | | --- | |  | | Модель Role описывает роль пользователя в системе, например, пользователь или администратор. Атрибуты включают уникальный идентификатор роли (id), название роли (name) и набор пользователей, связанных с этой ролью (users). | | |
| User | Модель User представляет пользователя системы. Атрибуты включают уникальный идентификатор пользователя (id), имя пользователя (username), пароль (password), флаг активности пользователя (enabled), электронную почту (email) и набор ролей, назначенных пользователю (roles). |

В контексте разработки приложения, особенно при использовании фреймворков, таких как Spring, репозиторий (repository) — это шаблон проектирования, который используется для инкапсуляции логики доступа к данным или источникам данных. Репозиторий обеспечивает абстракцию для доступа к данным, позволяя остальной части приложения взаимодействовать с этими данными как с коллекцией объектов. На рисунке 25 показаны классы, которые выполняют этот функционал.

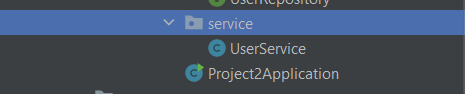


*Рисунок 24. Список классов папки repository*

Таблица 14. Описание содержание папки repository

|  |  |
| --- | --- |
| Название класса | Описание |
| CategoryRepository | Репозиторий для работы с объектами Category, обеспечивающий основные операции CRUD (создание, чтение, обновление, удаление). Наследуются от JpaRepository, включают методы для создания, чтения, обновления и удаления категорий.   * save(Category category): Сохраняет или обновляет категорию. * findById(Long id): Находит категорию по идентификатору. * findAll(): Возвращает список всех категорий. * deleteById(Long id): Удаляет категорию по идентификатору. |
| DocumentRepository | Репозиторий для работы с объектами Document, обеспечивающий основные операции CRUD и дополнительные методы для поиска документов по ключевым словам и категориям. Наследуются от JpaRepository, включают методы для создания, чтения, обновления и удаления документов.   * save(Document document): Сохраняет или обновляет документ. * findById(Integer id): Находит документ по идентификатору. * findAll(): Возвращает список всех документов. * deleteById(Integer id): Удаляет документ по идентификатору |
| RoleRepository | Репозиторий для работы с объектами Role, обеспечивающий основные операции CRUD и метод для поиска ролей по имени. |
| UserRepository | Репозиторий для работы с объектами User, обеспечивающий основные операции CRUD и метод для поиска пользователей по имени пользователя. |

Service — это классы или компоненты, которые содержат бизнес-логику приложения. Они служат связующим звеном между уровнем представления (например, веб-контроллеры) и уровнем доступа к данным (например, репозитории). На рисунке 24 показаны все сервисы, которые есть в приложении.



*Рисунок 25. Список классов папки service*

Таблица 15. Описание содержание папки service

|  |  |
| --- | --- |
| Название класса | Описание |
| **UserService** | **UserService** – это сервисный класс, который предоставляет методы для управления пользователями. Этот класс взаимодействует с UserRepository для выполнения операций над данными пользователей и использует PasswordEncoder для шифрования паролей. |

На рисунке 25 изображён раздел проекта с ресурсами. Ресурсы хранят формы, стили, изображения и т. д. Этот раздел отвечает за визуальную часть сервиса.

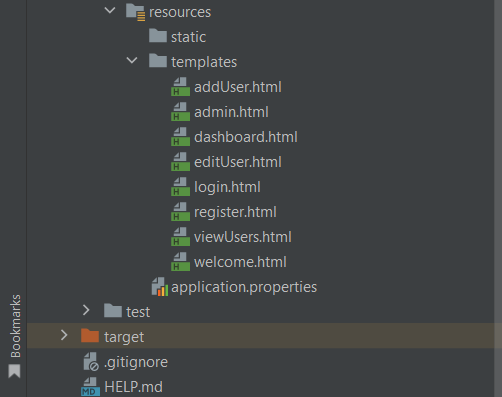


Рисунок 26. Список ресурсов

## **3.3. Руководство пользователя**

### **3.3.1. Руководство для клиентов веб-сервиса регистрации и учета пациентов клиники**

При переходе на веб-сервис отображается главная страница. Страница изображена на рисунке 27.

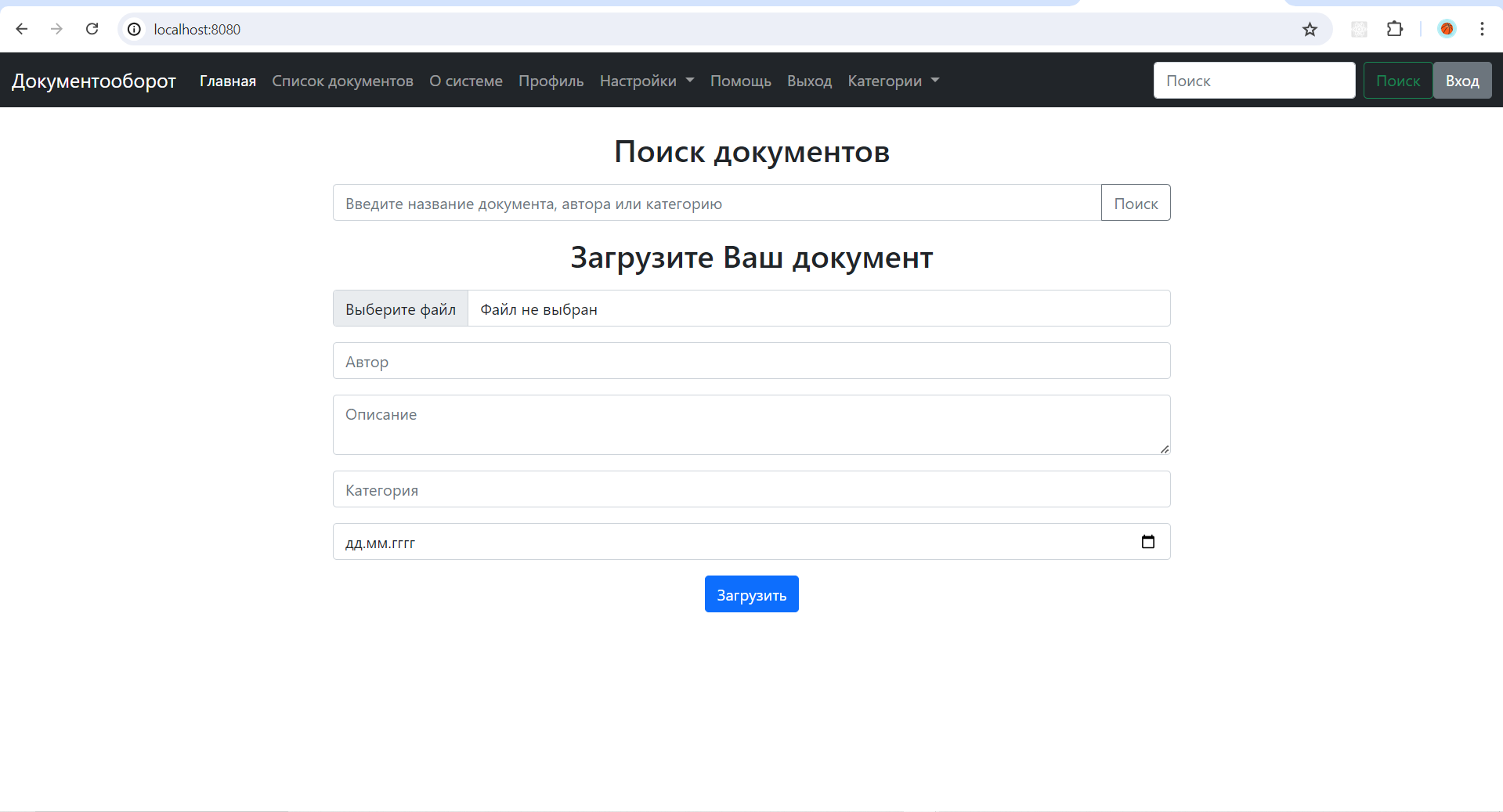


Рисунок 27. Главная страница

Для того , чтобы загрузить документ нужно нажать на кнопку «вход»



Рисунок 28.Вход

После нажатия на кнопки вход, отображается регистрация. Пользователь, который еще не имеет аккаунта, может заполнить форму регистрации, введя имя пользователя, пароль и адрес электронной почты, а затем нажать кнопку "Register" для создания нового аккаунта.

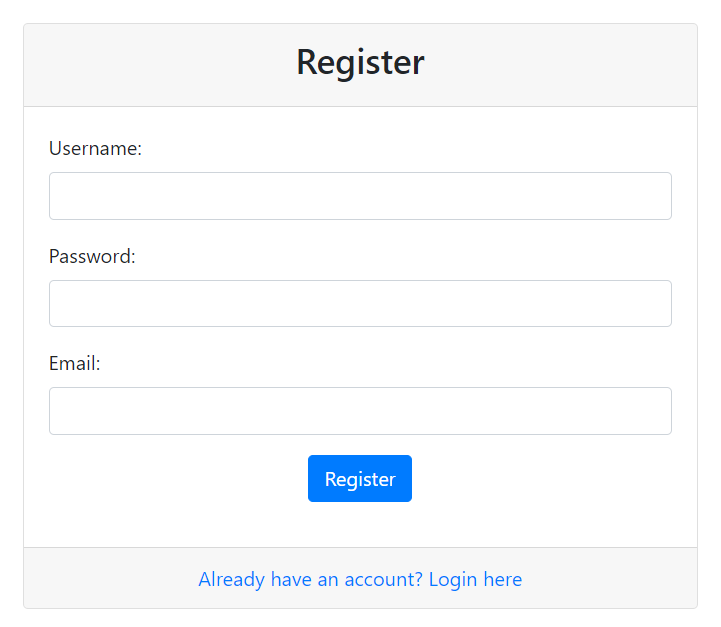


Рисунок 29.Регистрация

Пользователь, который уже имеет аккаунт, может заполнить форму входа, введя свое имя пользователя и пароль, а затем нажать кнопку "Login" для входа в систему.

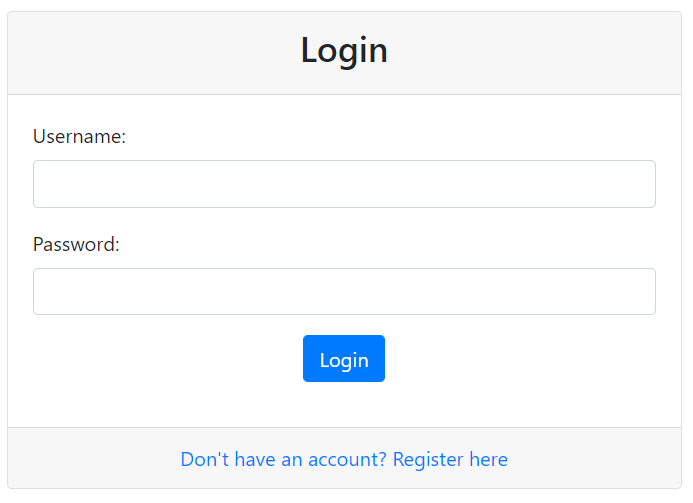


Рисунок 30.Аутенфикация

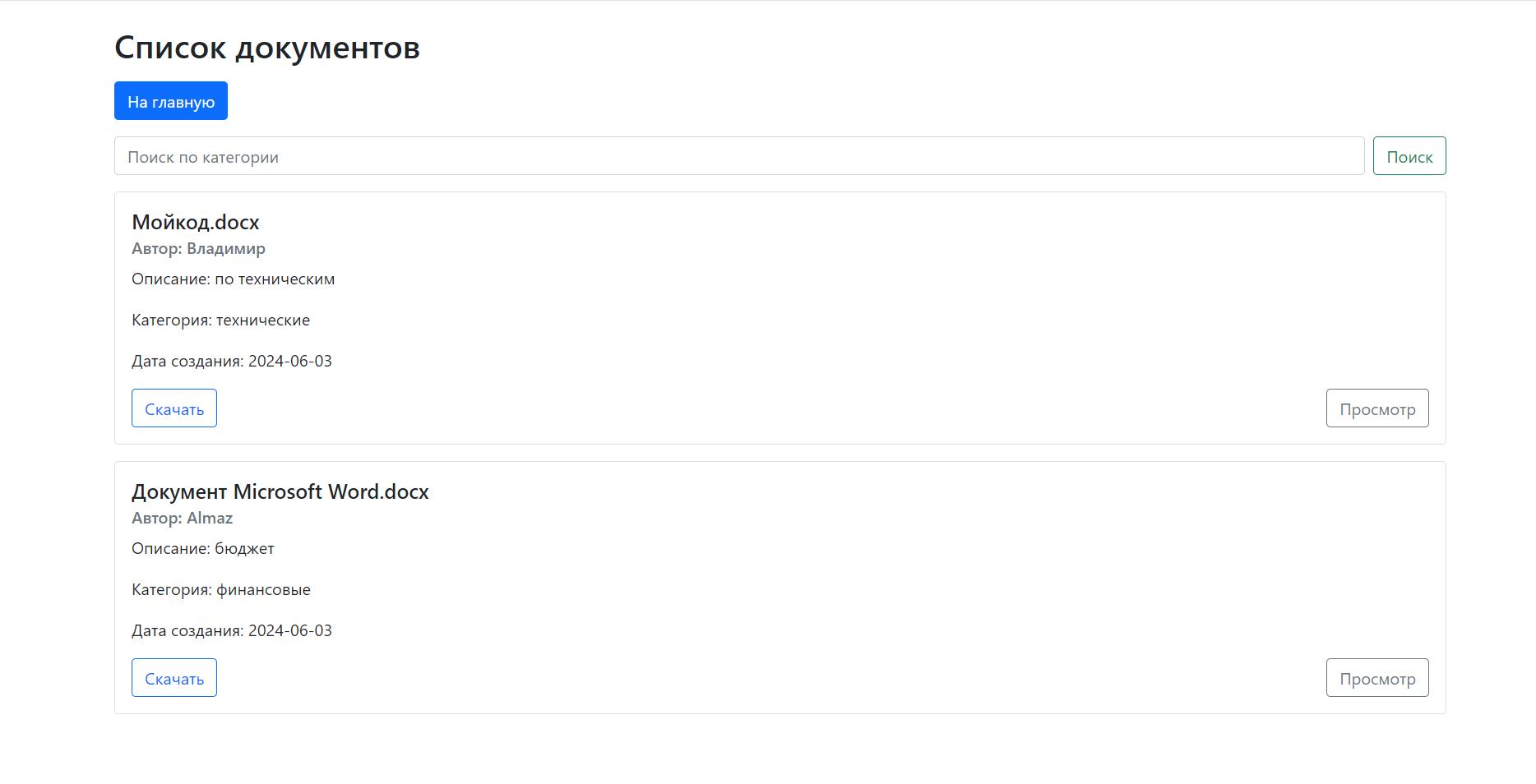


Рисунок 31. Список документов

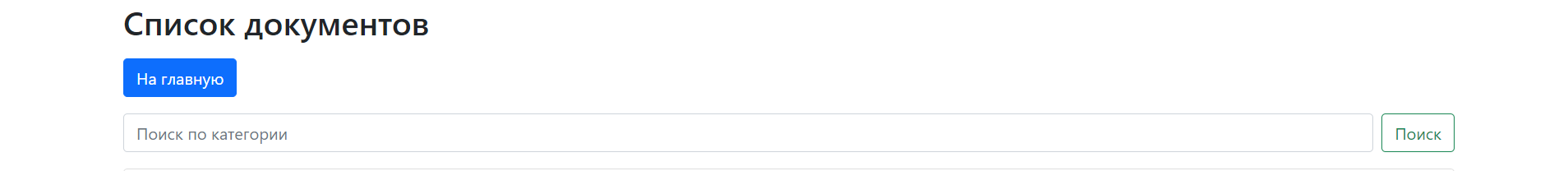


Рисунок 32. Поиск

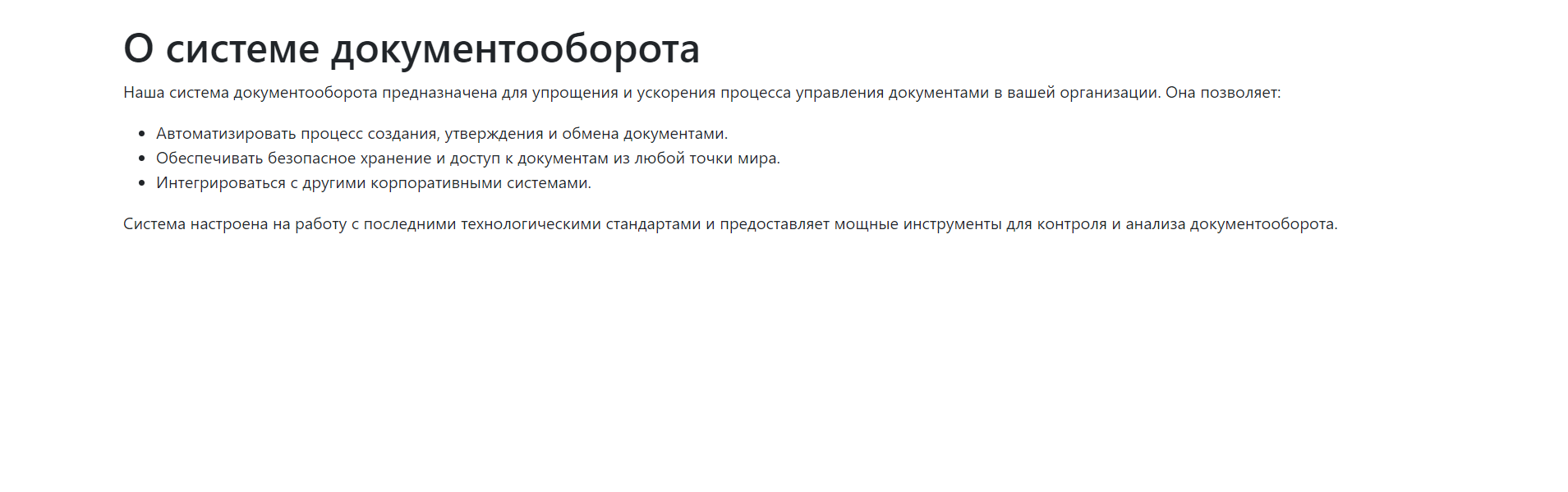


Рисунок 33. Отображение информации о системе

После всех процессов пользователь может добавлять документов.

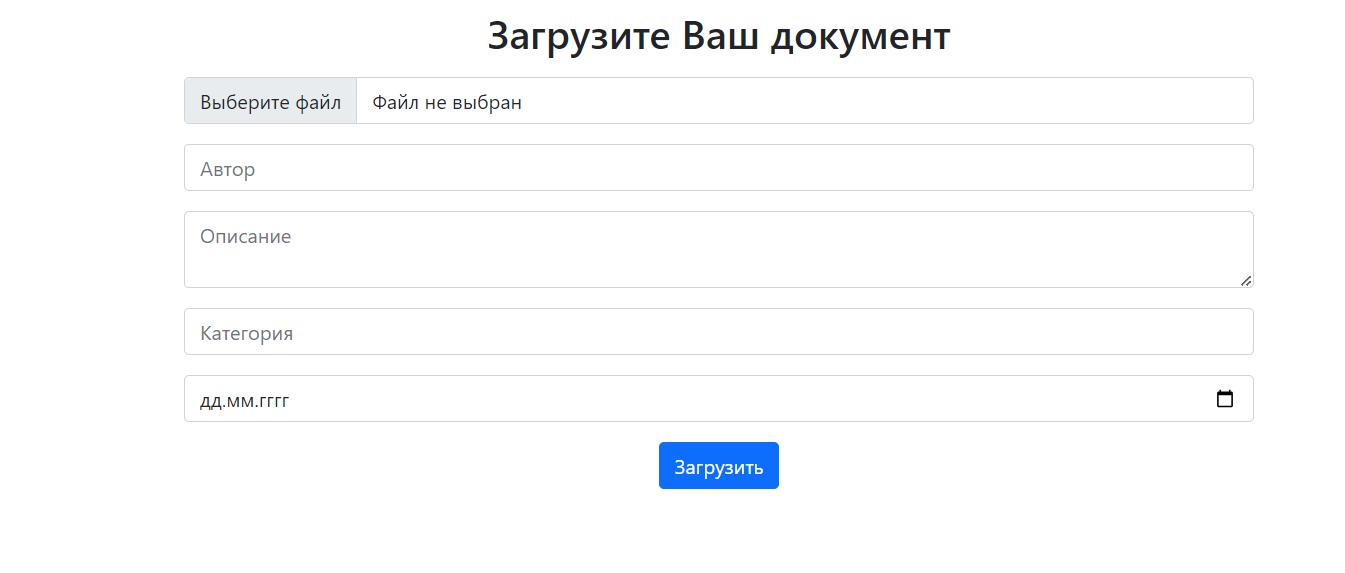


Рисунок 34.Загрузка документов

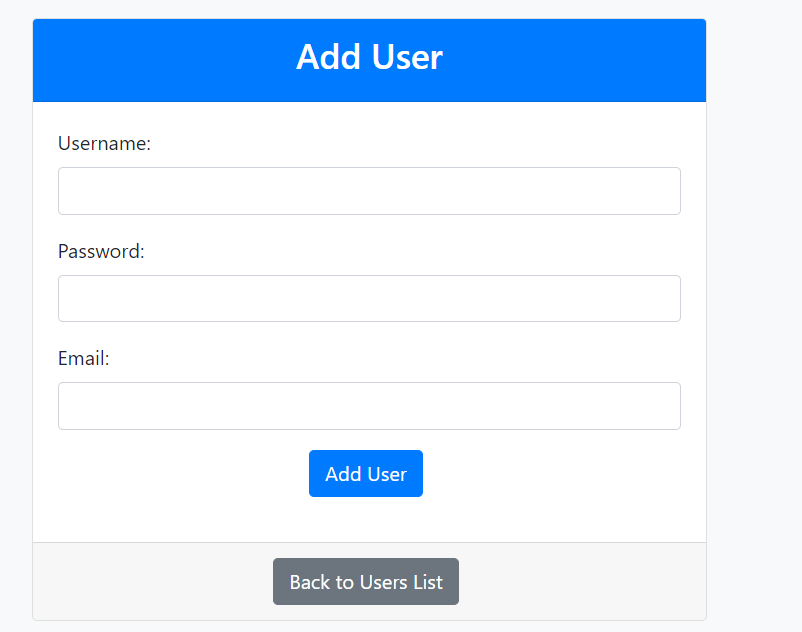


Рисунок 35. Добавление нового пользователя

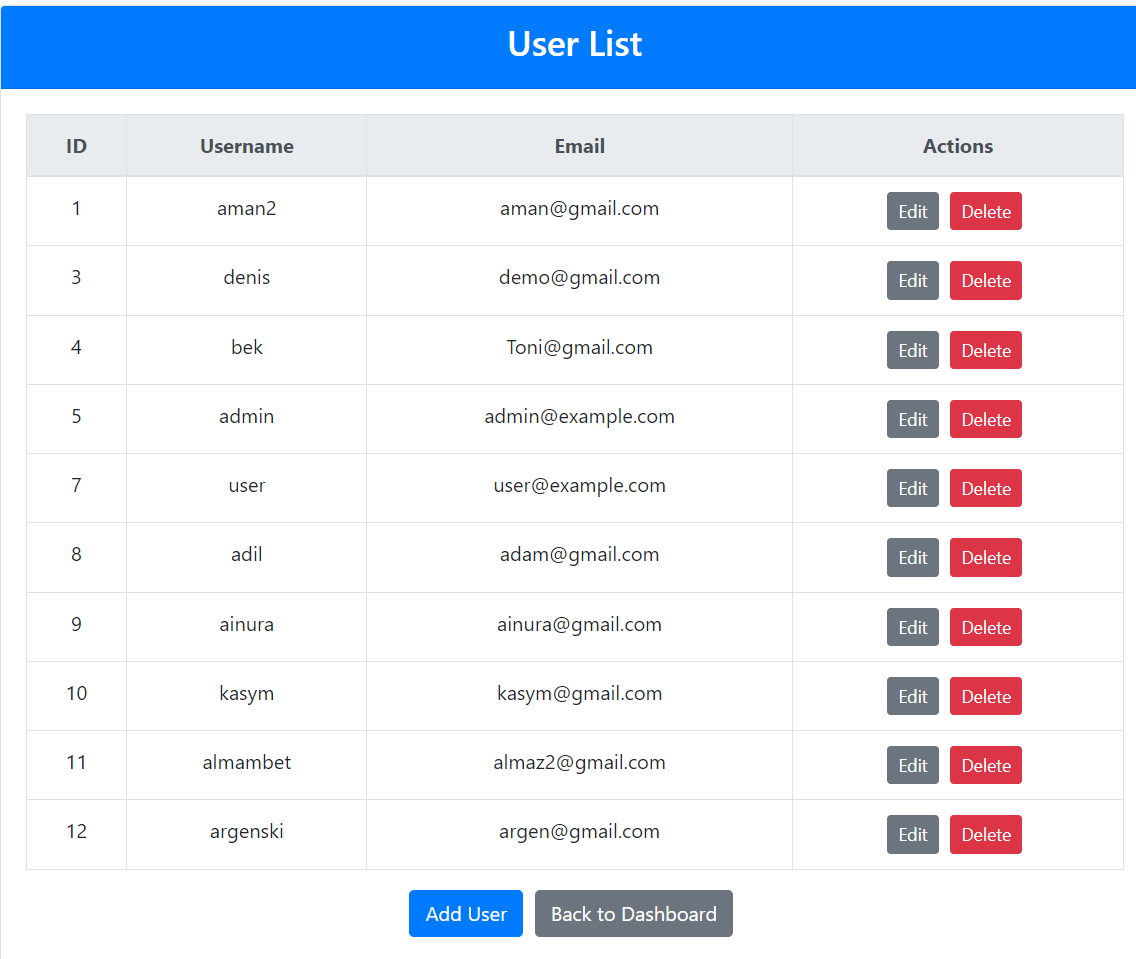


Рисунок 36. Редактирования, удаления пользователя

# **ЗАКЛЮЧЕНИЕ**

При выполнении выпускной квалификационной работы были достигнуты следующие результаты:

* Проведён детальный анализ предметной области;
* Обоснована актуальность работы и необходимость разработки;
* Исследованы необходимые бизнес-процессы;
* Определены задачи и цели разработки;
* Проведён обзор и анализ существующих систем;
* Разработаны функциональные и нефункциональные требования к системе;
* Определены требования к аппаратным средствам;
* Спроектирована карта сайта;
* Описана работа системы с помощью диаграмм деятельности;
* Разработана структура базы данных;
* Обоснован выбор средств реализации системы, тип и архитектура системы;
* Разработан пользовательский интерфейс;
* Разработаны руководство программиста и руководство пользователя.

Веб-сервис по учету и регистрации научных отчетов помогает значительно упростить работу пользователей, тем самым ускоряя доступ к необходимым документам и повышая эффективность работы. Проект успешно решает задачу автоматизации процессов добавления, редактирования, удаления и поиска научных отчетов, а также управления пользователями и отправки уведомлений.

В процессе выполнения работы были изучены и применены современные технологии веб-разработки, включая Spring Framework для создания серверной части приложения и Thymeleaf для разработки пользовательского интерфейса. Также использовалась система управления базами данных PostgreSQL для хранения и обработки данных.

В дальнейшем планируется развитие системы путем интеграции дополнительных модулей, таких как системы автоматического анализа научных данных, интеграция с другими информационными системами и внедрение инструментов анализа данных для получения статистической и аналитической информации. Таким образом, выполненная работа позволила создать функциональное веб-приложение для хранения и управления научными отчетами, которое отвечает всем современным требованиям и стандартам в области веб-разработки и информационных систем.

# **ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ**

1. Автоматизированная система управления https://ru.wikipedia.org/wiki/Автоматизированная\_система\_управления
2. Монахов, В.В. Язык программирования Java и среда NetBeans. 3-е изд., пер. / В.В. Монахов. — СПб.: BHV, 2017. — 704 c.
3. П. Наутон, Г. Шилдт. Java 2. Наиболее полное руководство в подлиннике. – СПб.: БХВ-Петербург, 2000. – 1072 с.: ил.
4. Рамбо Дж., Якобсон А., Буч Г. UML: специальный справочник. – СПб.: Питер, 2002. – 656 c.
5. Давыдов, С. IntelliJ IDEA. Профессиональное программирование на Java / С. Давыдов. - СПб.: BHV, 2005. - 800 c.
6. Герман, О.В Программирование на Java и C# для студентов / О. В. Герман. - СПб.: BHV, 2005. - 512 c.
7. Машнин, Т.С. Web-сервисы Java. Профессиональное программирование / Т.С. Машнин. - СПб.: BHV, 2012. - 560 c.
8. <https://docs.oracle.com/en/java/javase/22/> - Документация Java
9. https://www.uml.org/ - Документация по UML
10. https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16 – Документация по SQL
11. https://ru.wikipedia.org/wiki/REST - Описание архитектурного стиля REST

# **ГЛОССАРИЙ**

Таблица 4. Глоссарий

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| Application Programming Interface | Описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы |
| HTTP | Протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных |
| Фреймворк | Программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. |
| HTML | стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора |
| CSS | CSS — формальный язык декодирования и описания внешнего вида документа, написанного с использованием языка разметки. |
| Placeholder | Текст-заполнитель — это текст, который имеет некоторые характеристики реального письменного текста, но является случайным набором слов или сгенерирован иным образом. |
| Back-End | Направление веб-разработки, которое отвечает за все, что происходит с сервером, данными и общей логикой веб-сайта. |
| Front-End | Презентационная часть информационной или программной системы, её пользовательский интерфейс и связанные с ним компоненты |
| CRUD | CRUD — это аббревиатура, которая представляет четыре основные операции, используемые в приложениях для работы с данными: Создание (Create), Чтение (Read), Обновление (Update) и Удаление (Delete). Эти операции описывают стандартные методы взаимодействия с базами данных или любыми системами управления данными. |
| UML | UML (Unified Modeling Language) — это стандартизированный язык моделирования, используемый в области разработки программного обеспечения для визуализации, спецификации, конструирования и документирования систем. UML предоставляет стандартный способ описания структуры системы, включая её элементы, их взаимосвязи, динамику поведения и многое другое. |
| DAO | DAO (Data Access Object) — это шаблон проектирования, используемый в программировании для абстрагирования и инкапсуляции всех механизмов доступа к данным из остального приложения. Цель DAO — отделить слой бизнес-логики приложения от слоя доступа к данным. Это позволяет менять источники данных или их структуру без значительных изменений в бизнес-логике. |