Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ярославский государственный технический университет» Кафедра «Информационные системы и технологии»

Курсовой проект защищен

с оценкой \_\_\_\_\_\_\_

Преподаватель

\_\_\_\_\_\_\_\_\_\_ Ю. Н. Шулева

«\_\_\_» \_\_\_\_\_\_\_\_\_2025

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «Приложение для электронного документооборота DocChat»**

Пояснительная записка к курсовому проекту по дисциплине «Системы управления базами данных»

ЯГТУ 09.03.02-000 КП

Нормоконтролер Проект выполнил

Преподаватель студент группы Цис-26

\_\_\_\_\_\_\_\_\_ Ю. Н. Шулева В.И.Коробов

«\_\_\_» \_\_\_\_\_\_\_\_2025 «\_\_\_» \_\_\_\_\_\_\_\_2025

2025

Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ярославский государственный технический университет»

Кафедра **«**Информационные системы и технологии» \_

**ЗАДАНИЕ индивидуально по курсовому проектированию**

Студенту Коробову Вадиму Ивановичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ институт «Цифровых систем» .курс 2 .группа Цис-26 .

**I. Тема проекта и исходные данные**

Разработка информационной системы по теме «Разработка информационной системы «Приложение для электронного документооборота DocChat»

СУБД: PostgreSQL\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Среда разработки клиентского приложения: PyCharm\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**I. Представить следующие материалы:**

1. **текстовые**

Пояснительная записка к курсовому проекту, включающая: системный анализ предметной области, концептуальную схему, логическую структуру базы данных, описание приложения, руководство пользователя и техническое задание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Графические**

Иллюстративные материалы представляются студентом на защиту в виде компьютерной презентации в формате MS Power Point .

**III. Рекомендуемая литература и материал:**

1. Ткачев О.А. Создание и манипулирование базами данных средствами СУБД Microsoft SQL

Server 2008 [Электронный pecypc]: Учебное пособие/ Ткачев О.А. – Электрон. текстовые данные. – М.: Московский городской педагогический университет, 2013. – 152 с. – Режим доступа: http://www.bibliocomlectator.ru/book/?id=26613 .

1. Чубукова И.А. Data Mining [Электронный pecypc]/ Чубукова И.А. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 470 с. –

Режим доступа: http://www.bib1iocomp1ectator.ru/book/?id–

56315.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Дата выдачи задания**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Срок сдачи законченного проекта**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Отметка о явке на консультацию:**

1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 4)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 6)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель проекта \_\_\_\_\_\_Ю. Н. Шулева зав. кафедрой\_\_\_\_\_\_ С. Ю. Бойков**

**Задание принял к исполнению «**\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ **Студент\_\_\_\_\_\_\_\_\_**

РЕФЕРАТ

84 с., 46 рис., 14 табл., 5 источников, 5 прил.

PostgreSQL, СУБД, Python, Pycharm,Web-приложение

Задание:

Разработать безопасную информационную систему для хранения и управления документами с разграничением прав доступа. Система предназначена для пользователей, работающих с конфиденциальными документами. Регистрация и вход пользователей осуществляются через email и одноразовый пароль (OTP). После входа пользователи могут загружать документы, управлять версиями файлов, предоставлять доступ другим участникам или группам, а также отслеживать историю операций.

Разработать: модуль аутентификации с OTP, хранение и шифрование документов, управление версиями файлов, модуль передачи и расшаривания документов, панель администратора для управления пользователями и группами, а также систему журналирования действий.

Объектом исследования является автоматизированная система управления электронными документами с обеспечением безопасности и контроля доступа.

Актуальность разработки. В эпоху цифровизации и роста объёма конфиденциальной информации необходимо использовать надёжные средства защиты данных. Проект DocChat позволяет организациям централизованно и безопасно управлять своими документами, исключая несанкционированный доступ и повышая прозрачность процессов.

Цель работы:  
а) Автоматизация хранения, обмена и контроля версий документов;  
б) Обеспечение безопасного доступа к данным через систему ролей и прав;  
в) Обеспечение удобства и прозрачности при совместной работе над документами.

Методы исследования и используемые средства. В системе реализовано шифрование документов (AES-256), разграничение прав доступа на основе ролей, двухфакторная аутентификация, логирование всех действий. Хранилище реализовано на базе MinIO с версионностью объектов, веб-интерфейс построен на Django.

Полученные результаты. Реализована веб-система DocChat, обеспечивающая: безопасную регистрацию, загрузку и скачивание документов, управление версиями, предоставление доступа другим пользователям или группам, а также логирование всех операций. Все функции доступны через интуитивно понятный интерфейс.

Эффективность. Повышается уровень безопасности и контролируемости при работе с документами, уменьшаются риски утечки и потери данных, упрощается документооборот в организации.

Область применения. Корпоративные системы документооборота, юридические отделы, образовательные учреждения, органы госуправления, работающие с конфиденциальной информацией.

Основные эксплуатационные характеристики. Система обеспечивает надёжность, гибкость в настройке прав, совместную работу, контроль версий и защищённое хранение документов.

В результате всего процесса создаётся безопасная и функциональная система управления документами, способная заменить устаревшие и уязвимые решения в организациях, где требуется строгий контроль над электронным документооборотом.

Содержание

[Введение 5](#_Toc114091)

[1.1. Системный анализ предметной области 6](#_Toc114092)

[1.2. Назначение проектируемой системы 9](#_Toc114093)

[1.3. Цели создания проектируемой системы 10](#_Toc114094)

[1.4. Задачи, решаемые с помощью проектируемой системы 10](#_Toc114095)

[2. Проектная часть 10](#_Toc114096)

[2.3. Проектирование базы данных 17](#_Toc114104)

[2.3.1. Концептуальное проектирование 17](#_Toc114105)

[2.3.2. Логическое проектирование 17](#_Toc114106)

[2.3.3. Физическое проектирование 24](#_Toc114107)

[2.3.4. Роли и права доступа к данным 36](#_Toc114108)

[2.4. Проектирование интерфейса клиентской части 37](#_Toc114109)

[3. Технологическая часть 47](#_Toc114110)

[3.1. Выбор программных средств 47](#_Toc114111)

[3.1.1. Выбор СУБД 47](#_Toc114112)

[3.1.2. Выбор средств разработки 48](#_Toc114113)

[3.2. Описание работы с программой 48](#_Toc114114)

[Заключение 50](#_Toc114115)

[Список использованных источников и литературы 51](#_Toc114116)

[Приложение А 52](#_Toc114117)

[Приложение Б 62](#_Toc114118)

[Приложение В 68](#_Toc114119)

[Приложение Г 69](#_Toc114120)

# 

# Введение

Сегодня информационные системы — это неотъемлемая часть цифрового мира, позволяющая обрабатывать, хранить и защищать информацию в удобной форме. Особенно востребованы такие системы в организациях, где ежедневно обрабатываются конфиденциальные документы, и важно обеспечить как быстрый доступ к ним, так и высокий уровень безопасности.

Каждый день создаётся и передаётся огромное количество документов, многие из которых содержат чувствительные данные. Их необходимо не только сохранить, но и защитить от несанкционированного доступа, предоставить возможность контролировать версии и точно отслеживать, кто, когда и что с ними делал. Специализированные информационные системы позволяют решать эти задачи, минимизируя риск потери или искажения данных. Электронный документооборот избавляет организации от громоздкой бумажной работы, ускоряет процессы и делает их прозрачными и управляемыми.

Таким образом, актуальность разработки безопасной системы управления документами очевидна. Внедрение такой системы позволяет организациям контролировать доступ к корпоративным документам, отслеживать изменения, обеспечивать шифрование и вести журнал всех операций. Это особенно важно в условиях, когда информационная безопасность и соответствие внутренним регламентам являются приоритетами для компаний и учреждений.

1. Аналитическая часть

## 1.1. Системный анализ предметной области

База данных создается с целью обеспечения защищённого электронного документооборота внутри организации.

База данных должна содержать:

а) Данные о пользователях  
б) Данные о ролях и правах  
в) Данные о группах пользователей  
г) Данные о документах  
д) Данные об историях версий документов  
е) Данные о передаче документов  
ж) Данные о сертификатах и подписях  
з) Данные об аудит-логах (журнал действий)

В соответствии с предметной областью система строится с учётом следующих особенностей:

а) Пользователь может зарегистрироваться с подтверждением корпоративной почты через одноразовый код (OTP);  
б) Пользователь может загружать, скачивать и удалять свои документы;  
в) Пользователь может передавать документы другим пользователям и группам;  
г) Поддерживается версионность документов — каждая новая версия сохраняется отдельно;  
д) Роль и права пользователей определяются заранее и могут быть изменены только администратором;  
е) Система обеспечивает детальный аудит всех операций;  
ж) Документы хранятся в защищённом хранилище MinIO с применением шифрования AES-256;  
з) Передача и скачивание документов возможны только при наличии соответствующих прав;  
и) Поддерживается расширение до цифровой подписи и проверки целостности.

Выделим базовые сущности этой предметной области:

1. Пользователь (CustomUser)  
   1.1) id пользователя  
   1.2) email  
   1.3) имя пользователя  
   1.4) пароль  
   1.5) роль  
   1.6) ФИО  
   1.7) должность  
   1.8) OTP и время создания OTP  
   1.9) OAuth-данные (опционально)  
   1.10) права (can\_manage\_documents, can\_sign\_documents и т.д.)
2. Роль (Role)  
   2.1) id роли  
   2.2) название  
   2.3) описание  
   2.4) уровень доступа
3. Группа пользователей (UserGroup)  
   3.1) id группы  
   3.2) название  
   3.3) участники  
   3.4) руководитель
4. Документ (Document)  
   4.1) id документа  
   4.2) имя файла  
   4.3) исходное имя  
   4.4) тип контента  
   4.5) владелец  
   4.6) дата загрузки  
   4.7) статус  
   4.8) список пользователей с доступом  
   4.9) список групп с доступом  
   4.10) флаг шифрования
5. История версий документа (DocumentVersionHistory)  
   5.1) id версии  
   5.2) id документа  
   5.3) version\_id  
   5.4) URL версии  
   5.5) размер  
   5.6) etag  
   5.7) комментарий  
   5.8) дата загрузки
6. История передачи (DocumentTransferHistory)  
   6.1) id истории  
   6.2) id документа  
   6.3) отправитель  
   6.4) получатель (пользователь или группа)  
   6.5) дата  
   6.6) комментарий  
   6.7) тип действия  
   6.8) версия
7. Аудит (AuditLog)  
   7.1) id лога  
   7.2) пользователь  
   7.3) действие (login, upload, delete и др.)  
   7.4) дата  
   7.5) описание действия
8. Цифровой сертификат (DigitalCertificate)  
   8.1) id сертификата  
   8.2) пользователь  
   8.3) серийный номер  
   8.4) PEM-сертификат  
   8.5) pkcs12-файл  
   8.6) дата выдачи и окончания  
   8.7) статус (активен/отозван)  
   8.8) зашифрованный приватный ключ (опционально)
9. Журнал подписей PDF (PdfSignatureLog)  
   9.1) id подписи  
   9.2) документ  
   9.3) сертификат  
   9.4) дата  
   9.5) хэш  
   9.6) данные подписи  
   9.7) комментарий

Функции:

1) Функции пользователя с ролью “User”:  
1.1) Регистрация с подтверждением OTP  
1.2) Загрузка, просмотр, удаление собственных документов  
1.3) Получение расшаренных документов  
1.4) Загрузка новых версий файлов  
1.5) Отправка документа другим пользователям/группам  
1.6) Просмотр истории документа

2) Функции пользователя с ролью “Manager”:  
2.1) Все функции “User”  
2.2) Расширенное управление группами  
2.3) Возможность делегировать права другим пользователям  
2.4) Доступ к аналитике (при наличии флага can\_view\_statistics)

3) Функции пользователя с ролью “Admin”:  
3.1) Все функции “Manager”  
3.2) Управление пользователями, их правами и ролями  
3.3) Генерация цифровых сертификатов  
3.4) Просмотр и редактирование групп  
3.5) Доступ ко всем журналам (AuditLog)

Выводы. Система должна обеспечивать:

а) Защищённую регистрацию и вход с двухфакторной проверкой  
б) Гибкую систему ролей и прав пользователей  
в) Защищённую загрузку, шифрование и скачивание документов  
г) Поддержку версионности документов  
д) Возможность передачи документов между пользователями и группами  
е) Контроль доступа к документам  
ж) Аудит всех действий пользователей  
з) Возможность масштабирования и подключения новых модулей (подпись, блокчейн и др.)

Бизнес-правила:

1. Все пользователи регистрируются по корпоративной почте.
2. OTP действителен в течение 10 минут и обязателен при входе.
3. Документ может быть расшарен только владельцем.
4. Пользователь не может удалить чужой документ.
5. Документ перед загрузкой шифруется, а при скачивании — расшифровывается.
6. Каждая версия документа фиксируется и может быть отдельно скачана.
7. Только администратор может изменять роли и права пользователей.
8. Доступ к аналитике и управлению группами предоставляется только при наличии соответствующих флагов.
9. Все действия пользователей логируются в журнале.

## 1.2. Назначение проектируемой системы

Проектируемая система предназначена для автоматизации процессов защищённого электронного документооборота внутри организаций и предприятий, работающих в условиях повышенных требований к безопасности.

Система обеспечивает сбор, шифрование, хранение, обработку и передачу конфиденциальных документов между пользователями с различными уровнями доступа. Также она позволяет отслеживать историю версий, контролировать доступ, вести журнал действий и защищать документы от несанкционированного доступа.

Техническое задание по проектируемой информационной системе представлено в Приложении А.

## 1.3. Цели создания проектируемой системы

Целью создания системы является построение безопасной платформы для взаимодействия сотрудников и подразделений при передаче и согласовании документов.

## 1.4. Задачи, решаемые с помощью проектируемой системы

С помощью проектируемой системы DocChat решаются следующие задачи:

1. Хранение, защита и обработка данных пользователей с различными уровнями доступа;
2. Безопасная регистрация и аутентификация пользователей с использованием одноразовых кодов (OTP);
3. Загрузка, шифрование, хранение и скачивание документов с возможностью управления доступом;
4. Реализация механизма передачи документов между пользователями и группами с ведением истории;
5. Ведение истории версий документов с возможностью добавления новых и просмотра предыдущих;
6. Управление пользователями, их ролями и правами доступа через административный интерфейс;
7. Ведение аудита всех пользовательских действий для обеспечения прозрачности и контроля;
8. Подготовка к интеграции модулей цифровой подписи и проверки целостности документов.

# 2. Проектная часть

## 2.3. Проектирование базы данных

### 2.3.1. Концептуальное проектирование

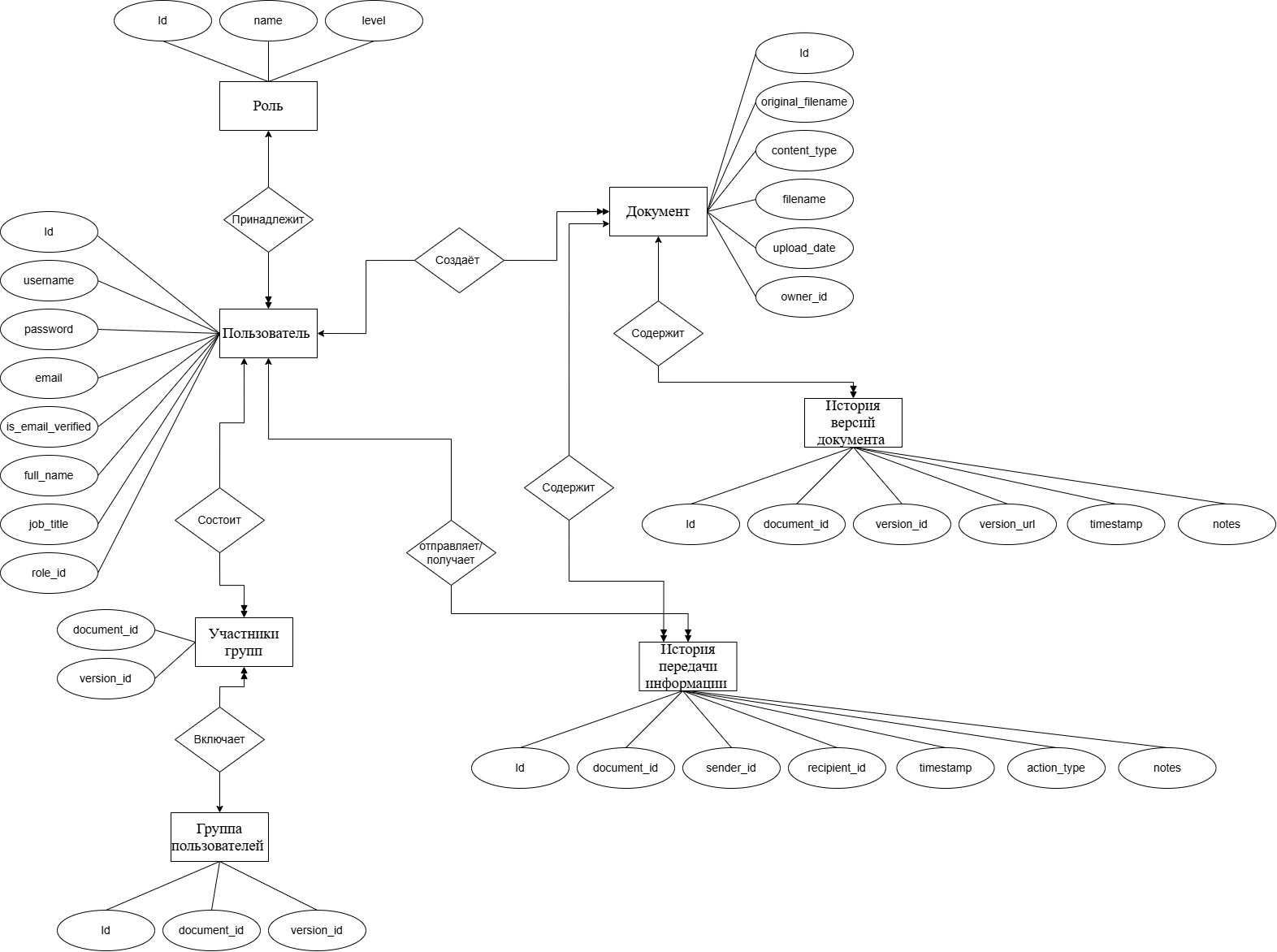


Рисунок 7 – Концептуальная модель

Объекты в моей программе:

1.Role – Роль пользователя

Атрибуты:

* id: PK
* name: название роли (например, "Admin", "User")
* level: уровень доступа

2. CustomUser – Пользователь

Атрибуты:

* id: PK
* username, password, email
* is\_email\_verified
* full\_name, job\_title
* role\_id: FK → Role

Связи:

* M:1 с Role
* 1:M с Document
* 1:M с DocumentTransferHistory (как отправитель или получатель)

3. Document – Документ

Атрибуты:

* id: PK
* filename, original\_filename
* content\_type
* upload\_date
* owner\_id: FK → CustomUser

Связи:

* M:1 с CustomUser (владелец)
* 1:M с DocumentVersionHistory, DocumentTransferHistory

4. DocumentVersionHistory – Версии документа

Атрибуты:

* id: PK
* document\_id: FK → Document
* version\_id: строка версии (например, v1.1)
* version\_url, timestamp
* notes

Связи:

* M:1 с Document

5. DocumentTransferHistory – История передачи

Атрибуты:

* id: PK
* document\_id: FK → Document
* sender\_id: FK → CustomUser
* recipient\_id: FK → CustomUser
* timestamp, action\_type, notes

Связи:

* M:1 с Document
* M:1 с CustomUser (отправитель и получатель)

5. UserGroup – Группа пользователей

Атрибуты:

* id: PK
* name: название группы
* leader\_id: FK → CustomUser (необязательный руководитель группы)

Тип связи: Many-to-Many

* Пользователь может быть участником нескольких групп
* Группа может включать многих пользователей

Для реализации создаётся промежуточная таблица usergroup\_members

### 2.3.2. Логическое проектирование

Логическое проектирование – это описание и построение схем связей между элементами данных безотносительно к их содержанию и среде хранения.

Цель логического проектирования:

1. выборка наиболее эффективной структуры данных,
2. обеспечения быстрого доступа к данным;
3. исключения дублирования данных, обеспечения целостности данных таким образом, чтобы при изменении одних объектов автоматически происходило соответствующее изменение связанных с ними объектов.

До приведения в нормальную форму.

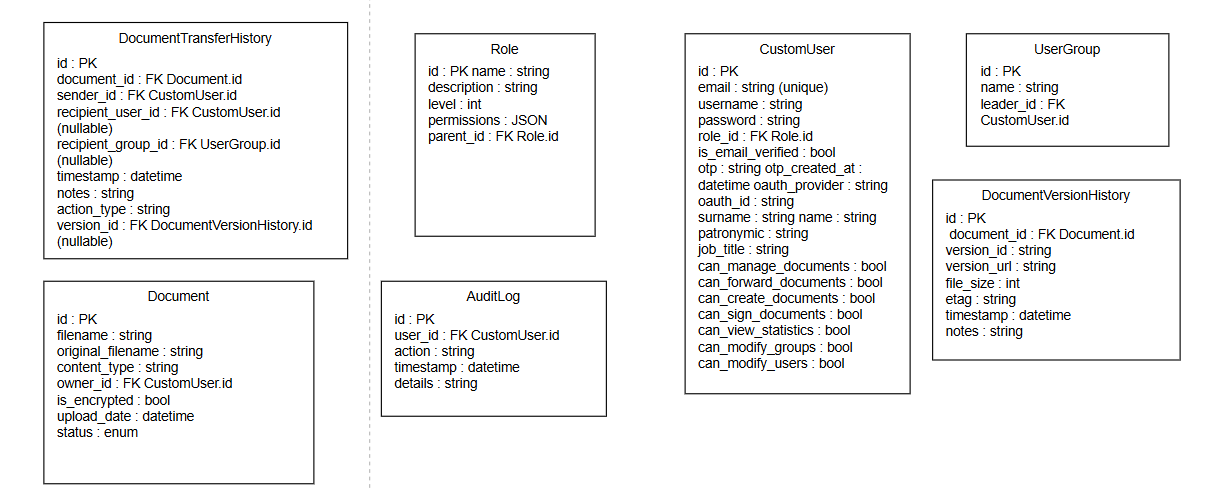


Рисунок 8 – Таблицы в неатомарном виде

Атрибуты на рисунке 8 не во всех таблицах удовлетворяют требованиям 1НФ.

Формат представления в 1 НФ:

Требование:

* Устранение повторяющихся групп данных.
* Все поля содержат атомарные (неделимые) значения.
* Каждая ячейка таблицы — одно значение.

Отличие:

* Можно допустить избыточность, нарушения функциональных зависимостей.

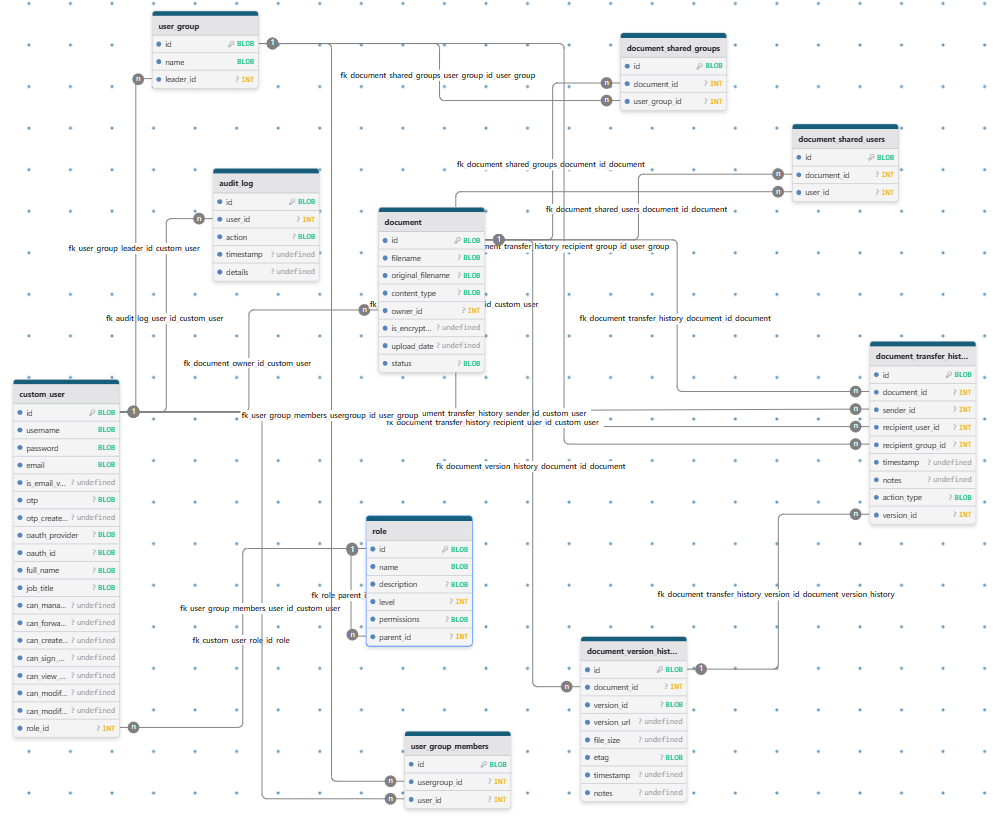


Рисунок 9 – Таблицы, приведенные в 1NF

Теперь схема на рисунке 9 отвечает требованиям 1NF.

Для преобразования во 2 НФ:

Требования:

* Удовлетворяет 1NF.
* Все неключевые атрибуты зависят от всего первичного ключа, а не от его части.
* Исключение частичной избыточности.

Изменения по сравнению с 1NF:

* Вынесены многозначные зависимости, например:
  + Права пользователя в отдельную таблицу user\_permissions.
  + Статусы документов в справочник document\_status.

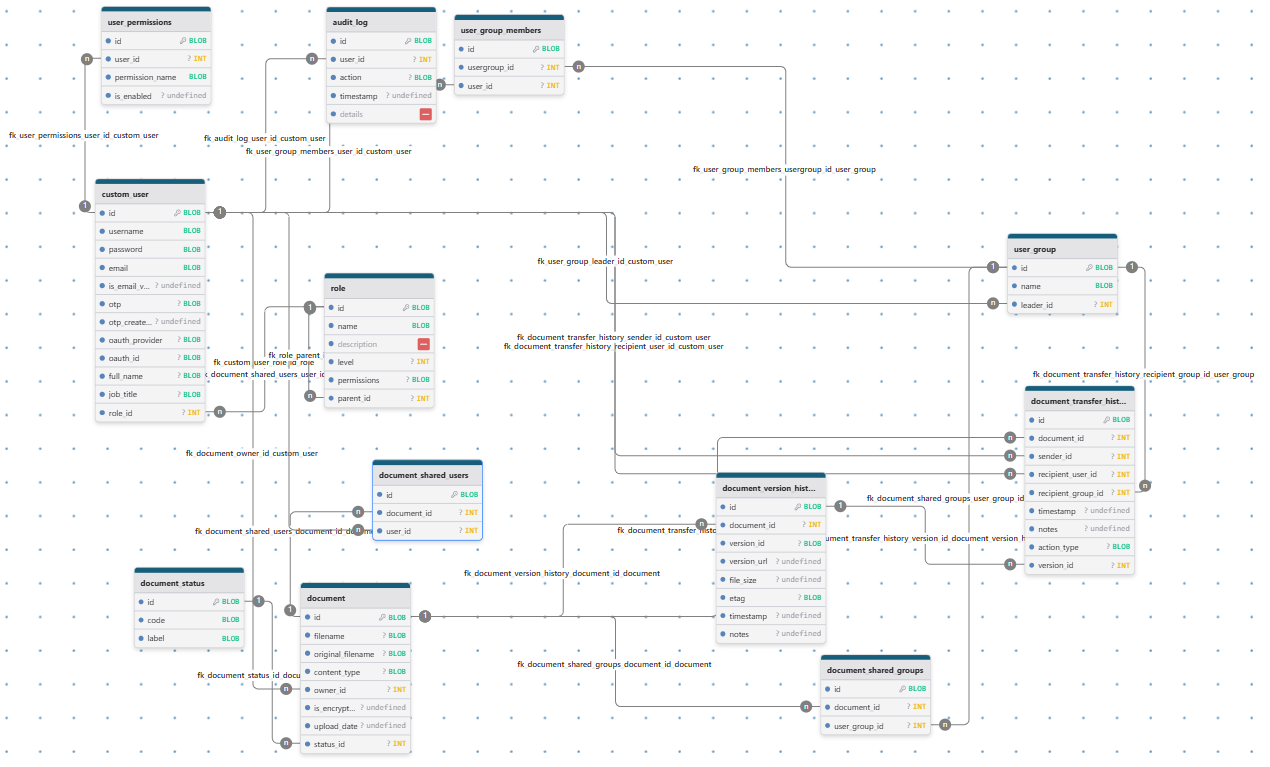


Рисунок 10 – Таблицы, приведенные в 2NF

Наша схема, изображенная на рисунке 10, приведена ко 2NF.

Для 3NF :

Требования:

* Удовлетворяет 2NF.
* Все неключевые атрибуты не зависят от других неключевых атрибутов (нет транзитивных зависимостей).

Изменения:

* Вынесение описаний ролей и метаданных в отдельные таблицы.
* Вынос permissions из JSON в отдельную нормализованную таблицу role\_permission.

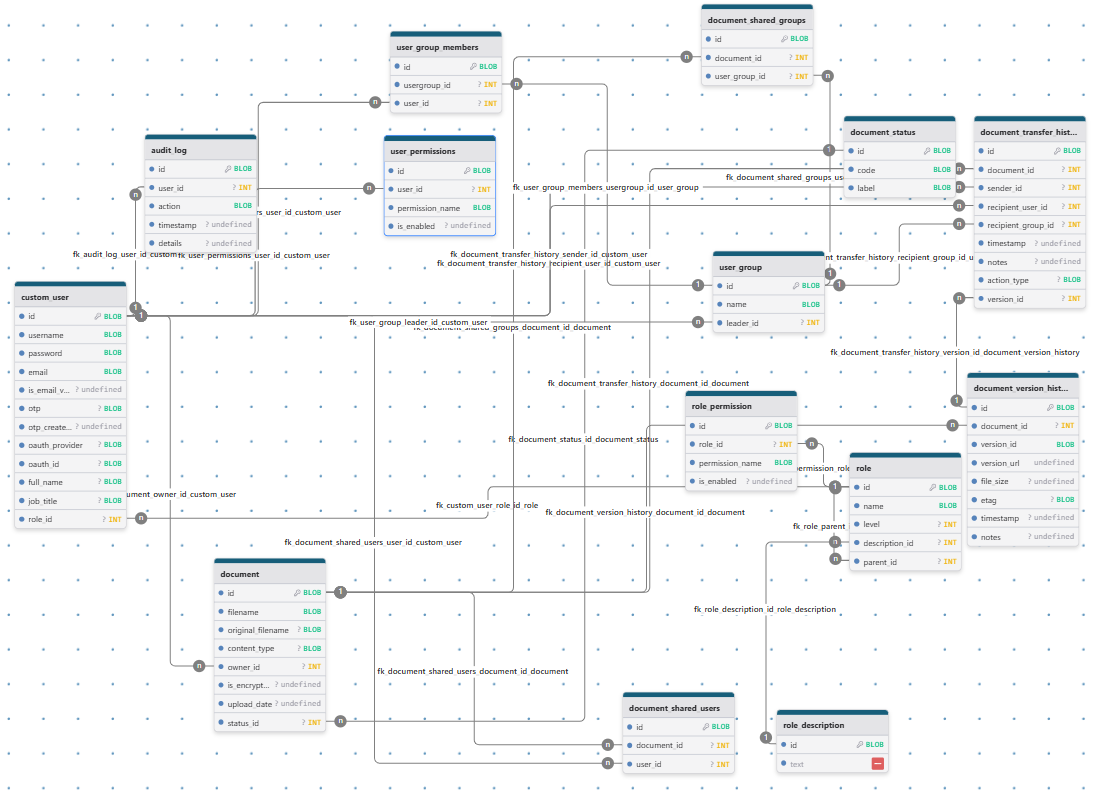


Рисунок 11 – Таблицы, приведенные в 3NF Теперь БД на рисунке 11 является нормализованной.

### 2.3.3. Физическое проектирование

Сделать описание каждого столбца каждой таблицы с помощью указанного ниже способа

Таблица 1 – Role (Роли пользователей)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | INT IDENTITY | PRIMARY KEY | Уникальный идентификатор роли. Автоинкрементируемое значение. |
| name | VARCHAR(100) | NOT NULL | Название роли (например, «Администратор», «Оператор»). |
| description | TEXT | NULL | Дополнительное описание роли, может содержать произвольный текст. |
| level | INT | NOT NULL DEFAULT 0 | Уровень иерархии роли: чем выше число, тем выше привилегии. |
| permissions | JSONB | NULL | JSON-объект, содержащий права доступа, специфичные для роли. |
| parent\_id | INT | FOREIGN KEY → Role(id) | Ссылка на родительскую роль, поддерживает наследование прав. |

Описание таблицы:

Хранит сведения о ролях, которые задают набор прав пользователей. Используется как основа системы разграничения доступа. Возможность иерархии ролей позволяет строить древовидную модель управления правами.

Таблица 2 – CustomUser (Пользователи системы)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | UUID | PRIMARY KEY | Уникальный идентификатор пользователя. Используется для внешних связей. |
| email | VARCHAR(255) | NOT NULL, UNIQUE | Адрес электронной почты пользователя. Используется для входа и уведомлений. |
| username | VARCHAR(150) | NOT NULL, UNIQUE | Уникальное имя пользователя. |
| password | VARCHAR(255) | NOT NULL | Хэшированный пароль. |
| role\_id | INT | FOREIGN KEY → Role(id) | Назначенная пользователю роль. |
| is\_email\_verified | BOOLEAN | NOT NULL DEFAULT FALSE | Указывает, подтверждён ли адрес электронной почты. |
| otp | VARCHAR(10) | NULL | Одноразовый код подтверждения (например, для 2FA). |
| otp\_created\_at | TIMESTAMP | NULL | Время создания OTP-кода. |
| oauth\_provider | VARCHAR(50) | NULL | Название внешнего провайдера (Google, GitHub и т. д.). |
| oauth\_id | VARCHAR(255) | NULL | Уникальный ID пользователя во внешней OAuth-системе. |
| surname | VARCHAR(100) | NULL | Фамилия пользователя. |
| name | VARCHAR(100) | NULL | Имя пользователя. |
| patronymic | VARCHAR(100) | NULL | Отчество пользователя. |
| job\_title | VARCHAR(100) | NULL | Должность пользователя в организации. |
| can\_manage\_documents | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли пользователь управлять документами. |
| can\_forward\_documents | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли пересылать документы. |
| can\_create\_documents | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли создавать новые документы. |
| can\_sign\_documents | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли подписывать документы. |
| can\_view\_statistics | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли просматривать статистику и отчёты. |
| can\_modify\_groups | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли управлять пользовательскими группами. |
| can\_modify\_users | BOOLEAN | NOT NULL DEFAULT FALSE | Может ли управлять другими пользователями. |

Описание таблицы:

Содержит полную информацию о пользователях системы, включая регистрационные данные, роль, OAuth-связь и набор флагов прав. Таблица поддерживает стандартную аутентификацию, двухфакторную аутентификацию и OAuth.

Таблица 3 – UserGroup (Группы пользователей)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | SERIAL | PRIMARY KEY | Уникальный идентификатор группы. |
| name | VARCHAR(100) | NOT NULL, UNIQUE | Название группы (например, «Юристы», «Секретари»). |
| leader\_id | UUID | FOREIGN KEY → CustomUser(id) | Идентификатор пользователя, руководителя данной группы. |

Описание таблицы:

Позволяет объединять пользователей в группы для совместной работы с документами. Каждая группа имеет лидера, ответственного за действия и контроль внутри группы.

Таблица 4 – Document (Документы)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | UUID | PRIMARY KEY | Уникальный идентификатор документа. |
| filename | VARCHAR(255) | NOT NULL | Имя файла при хранении на сервере. |
| original\_filename | VARCHAR(255) | NOT NULL | Имя файла, заданное пользователем при загрузке. |
| content\_type | VARCHAR(100) | NOT NULL | MIME-тип содержимого документа. |
| owner\_id | UUID | FOREIGN KEY → CustomUser(id) | Владелец (создатель) документа. |
| is\_encrypted | BOOLEAN | NOT NULL DEFAULT FALSE | Признак того, что файл зашифрован. |
| upload\_date | TIMESTAMP | NOT NULL DEFAULT CURRENT\_TIMESTAMP | Дата и время загрузки документа. |
| status | VARCHAR(50) | NOT NULL | Текущий статус (например, «черновик», «подписан», «удалён»). |

Описание таблицы:

Описывает загруженные документы, включая технические характеристики, владельца, тип и статус. Поддерживается шифрование на уровне поля.

Таблица 5 – DocumentTransferHistory (История передачи документов)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | SERIAL | PRIMARY KEY | Уникальный идентификатор записи. |
| document\_id | UUID | FOREIGN KEY → Document(id) | Идентификатор документа. |
| sender\_id | UUID | FOREIGN KEY → CustomUser(id) | Пользователь, отправивший документ. |
| recipient\_user\_id | UUID | FOREIGN KEY → CustomUser(id), NULLABLE | Получатель-другой пользователь. |
| recipient\_group\_id | INT | FOREIGN KEY → UserGroup(id), NULLABLE | Получатель-группа. |
| timestamp | TIMESTAMP | NOT NULL DEFAULT CURRENT\_TIMESTAMP | Время совершения действия. |
| notes | TEXT | NULL | Дополнительные комментарии к передаче. |
| action\_type | VARCHAR(50) | NOT NULL | Тип действия (например, «передача», «возврат», «отзыв»). |
| version\_id | UUID | FOREIGN KEY → DocumentVersionHistory(id) | Версия документа, связанная с этим действием. |

Описание таблицы:

Фиксирует передачу документа между пользователями и группами. Отражает все действия по маршрутизации, включая указание конкретной версии документа.

Таблица 6 – DocumentVersionHistory (История версий документов)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | UUID | PRIMARY KEY | Уникальный идентификатор записи версии. |
| document\_id | UUID | FOREIGN KEY → Document(id) | Документ, к которому относится версия. |
| version\_id | VARCHAR(100) | NOT NULL | Обозначение версии (например, «v1.0», «v2.1»). |
| version\_url | VARCHAR(255) | NOT NULL | URL-адрес сохранённой версии файла. |
| file\_size | INT | NOT NULL | Размер файла в байтах. |
| etag | VARCHAR(100) | NULL | Контрольная сумма или хеш для проверки целостности. |
| timestamp | TIMESTAMP | NOT NULL DEFAULT CURRENT\_TIMESTAMP | Время создания данной версии. |
| notes | TEXT | NULL | Дополнительная информация или примечания. |

Описание таблицы:

Сохраняет информацию о каждой версии документа. Каждая версия имеет собственный идентификатор и URL, позволяющий восстановить её независимо от текущей версии.

Таблица 7 – AuditLog (Журнал аудита)

| Имя поля | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| id | SERIAL | PRIMARY KEY | Уникальный идентификатор записи аудита. |
| user\_id | UUID | FOREIGN KEY → CustomUser(id) | Пользователь, совершивший действие. |
| action | VARCHAR(255) | NOT NULL | Тип действия (например, «удаление документа»). |
| timestamp | TIMESTAMP | NOT NULL DEFAULT CURRENT\_TIMESTAMP | Время события. |
| details | TEXT | NULL | Дополнительная информация (путь файла, IP, параметры). |

Описание таблицы:

Представляет собой системный лог, фиксирующий действия пользователей. Полезна для обеспечения безопасности, анализа и внутреннего контроля.

### 2.3.4. Роли и права доступа к данным

В системе DocChat доступ к данным пользователей и документам регулируется с помощью ролевой модели, групп пользователей, флагов полномочий и истории действий. Ниже описано, какие пользователи что могут делать и как это контролируется.

Основные механизмы разграничения доступа

| Механизм | Назначение |
| --- | --- |
| Роль (Role) | Общая категория полномочий (например, "Администратор", "Юрист", "Секретарь"). |
| Пользовательские флаги | Гибкая настройка конкретных действий внутри роли. |
| Группа (UserGroup) | Совместный доступ к документам и управлению ими. |
| История передачи (DocumentTransferHistory) | Контролирует, кто может видеть/редактировать конкретный документ. |
| Версии и аудит | Обеспечивают прозрачность действий и контроль версий. |

Примеры типов пользователей и их доступ

1. Администратор

| Доступ | Как регулируется |
| --- | --- |
| Управление всеми пользователями и ролями | Флаг can\_modify\_users = TRUE, роль уровня 10+ |
| Управление группами пользователей | can\_modify\_groups = TRUE |
| Просмотр и удаление всех документов | Роль, флаг доступа + отсутствие ограничений по владельцу |
| Просмотр статистики и логов | can\_view\_statistics = TRUE |
| Подпись и отправка документов | can\_sign\_documents = TRUE |

2. Руководитель группы (например, начальник отдела)

| Доступ | Как регулируется |
| --- | --- |
| Просмотр и отправка документов своей группы | Через recipient\_group\_id и принадлежность к группе |
| Просмотр членов своей группы | Через UserGroup и leader\_id |
| Контроль версий и истории | На основании прав и участия в DocumentTransferHistory |
| Подпись документов | can\_sign\_documents = TRUE |

3. Секретарь / помощник

| Доступ | Как регулируется |
| --- | --- |
| Загрузка новых документов | can\_create\_documents = TRUE |
| Пересылка документов | can\_forward\_documents = TRUE |
| Просмотр истории переданных документов | через DocumentTransferHistory |
| Без права подписи | can\_sign\_documents = FALSE |

4. Юрист / исполнитель

| Доступ | Как регулируется |
| --- | --- |
| Получение и подписание документов | Документ передан ему или его группе; can\_sign\_documents = TRUE |
| Создание версий документа | Через интерфейс редактирования, при наличии документа |
| Просмотр истории документа | Доступен по Document.id из DocumentVersionHistory |

5. Гость / внешний пользователь

| Доступ | Как регулируется |
| --- | --- |
| Только просмотр полученного документа | Передан напрямую (recipient\_user\_id) |
| Нельзя пересылать, удалять, изменять | Все флаги FALSE |

Контроль доступа к документам

Доступ к конкретному документу даётся на основе:

1. Владельца: Document.owner\_id == user.id
2. Истории передачи: если пользователь или его группа есть в DocumentTransferHistory
3. Флагов доступа: может ли он создавать, пересылать, редактировать, подписывать
4. Версии: если пользователь создавал или получал версию в DocumentVersionHistory

Контроль версий и аудита

* Любое действие фиксируется в таблице AuditLog — кто, что сделал, когда, над каким документом.
* Каждое изменение документа — это новая запись в DocumentVersionHistory, и пользователь может видеть только те версии, к которым у него был доступ.

## 2.4. Проектирование интерфейса клиентской части

Для любых пользователей:

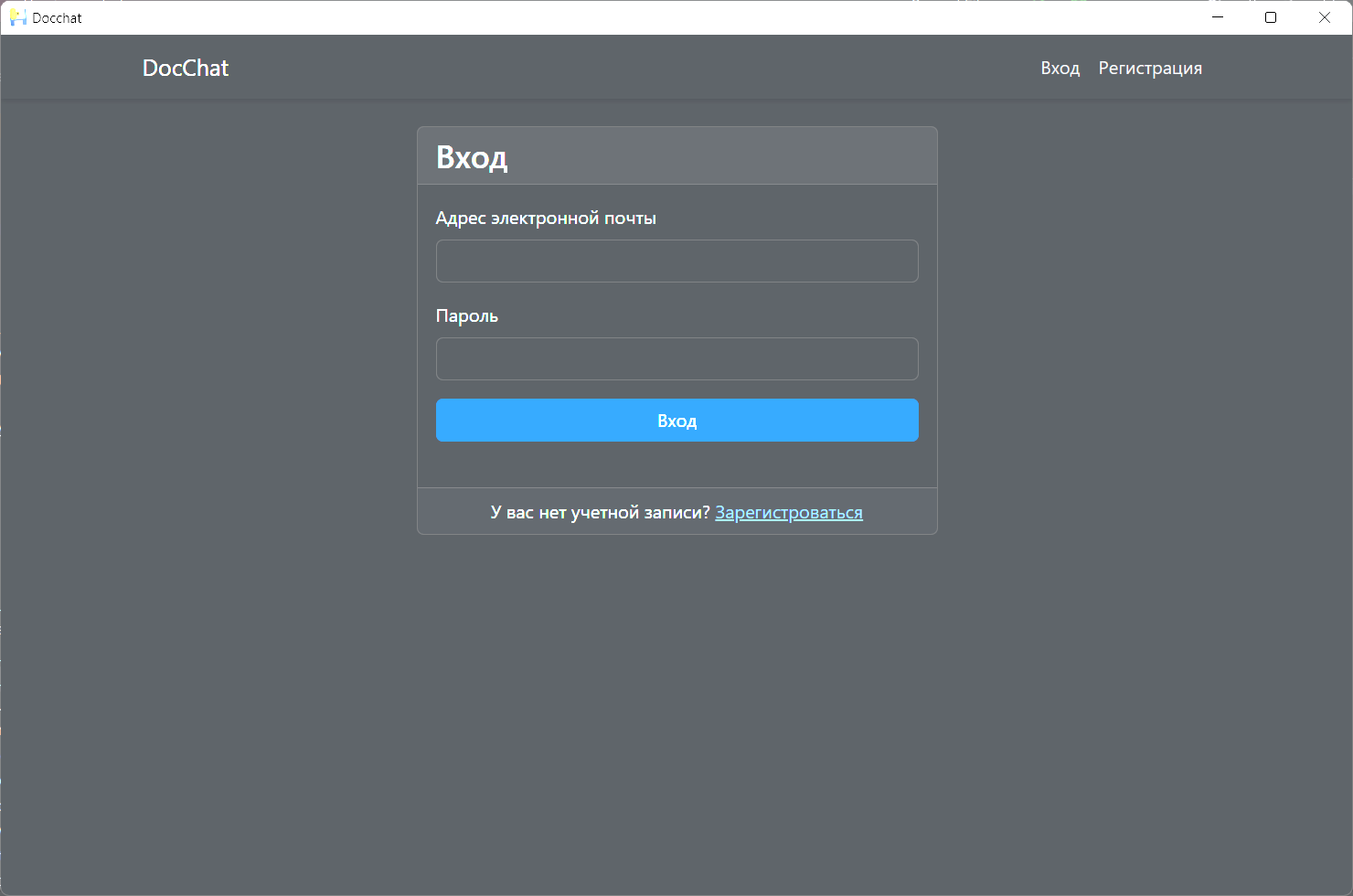


Рис. 12 Страница входа

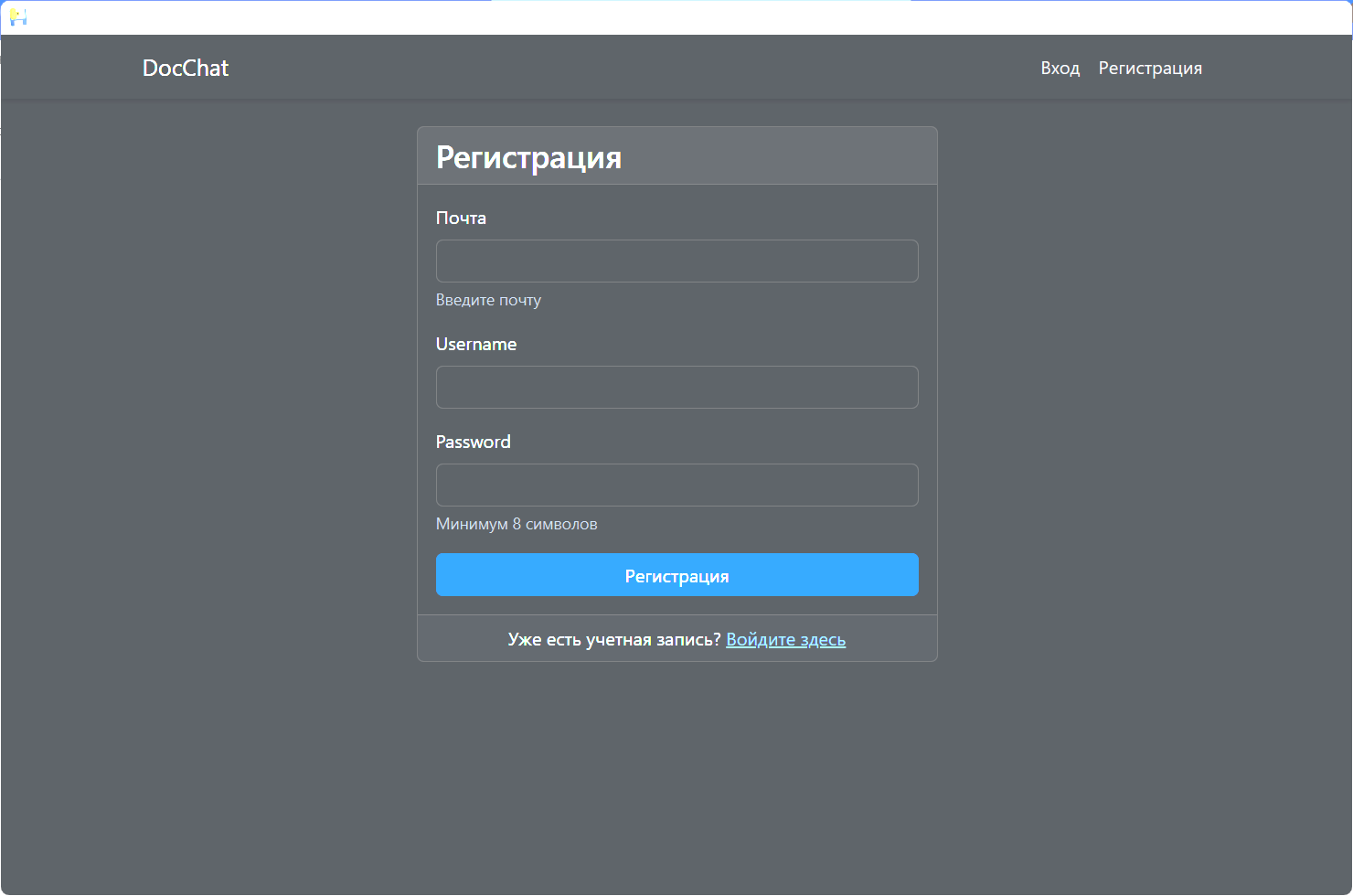


Рис. 13 Страница регистрации

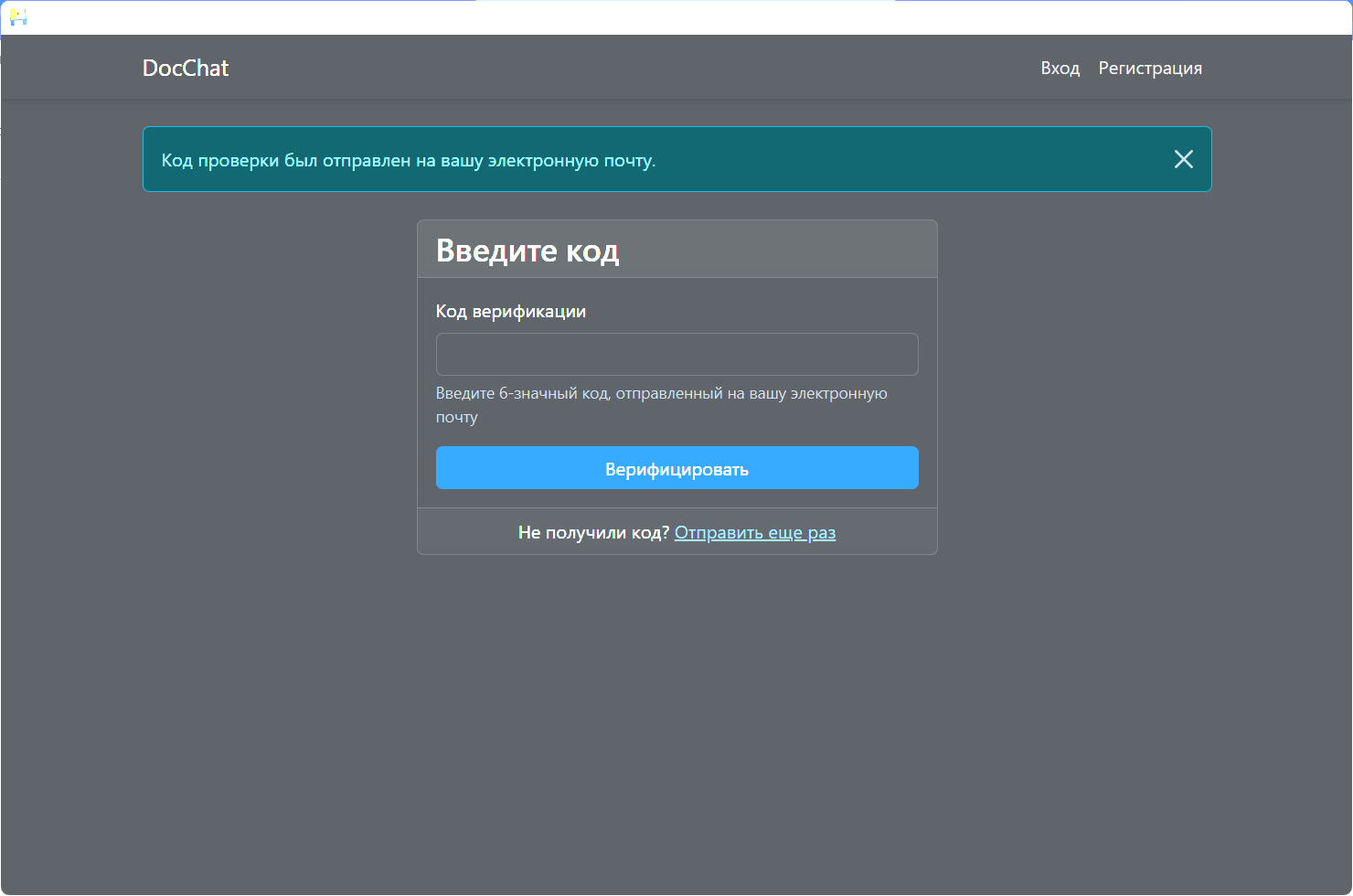


Рис. 14 Верификация по OTP

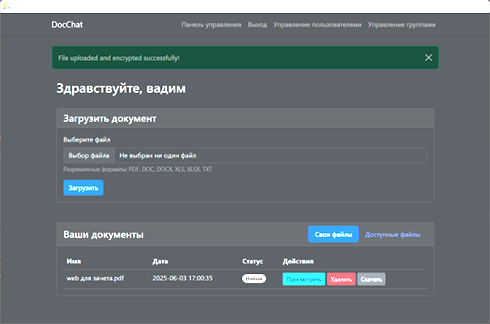


Рис. 15 Главная страница

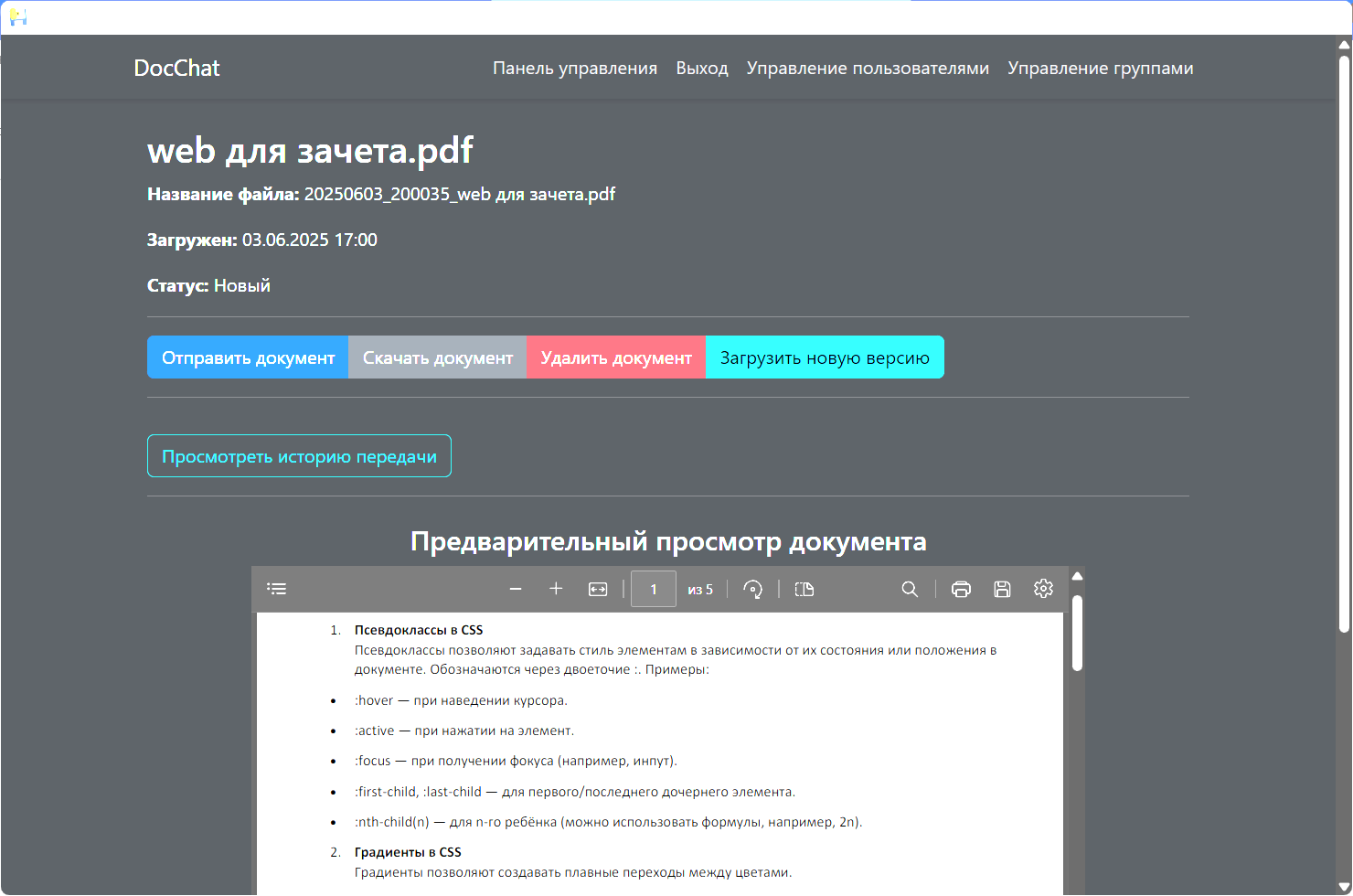


Рис. 16 Страница документа

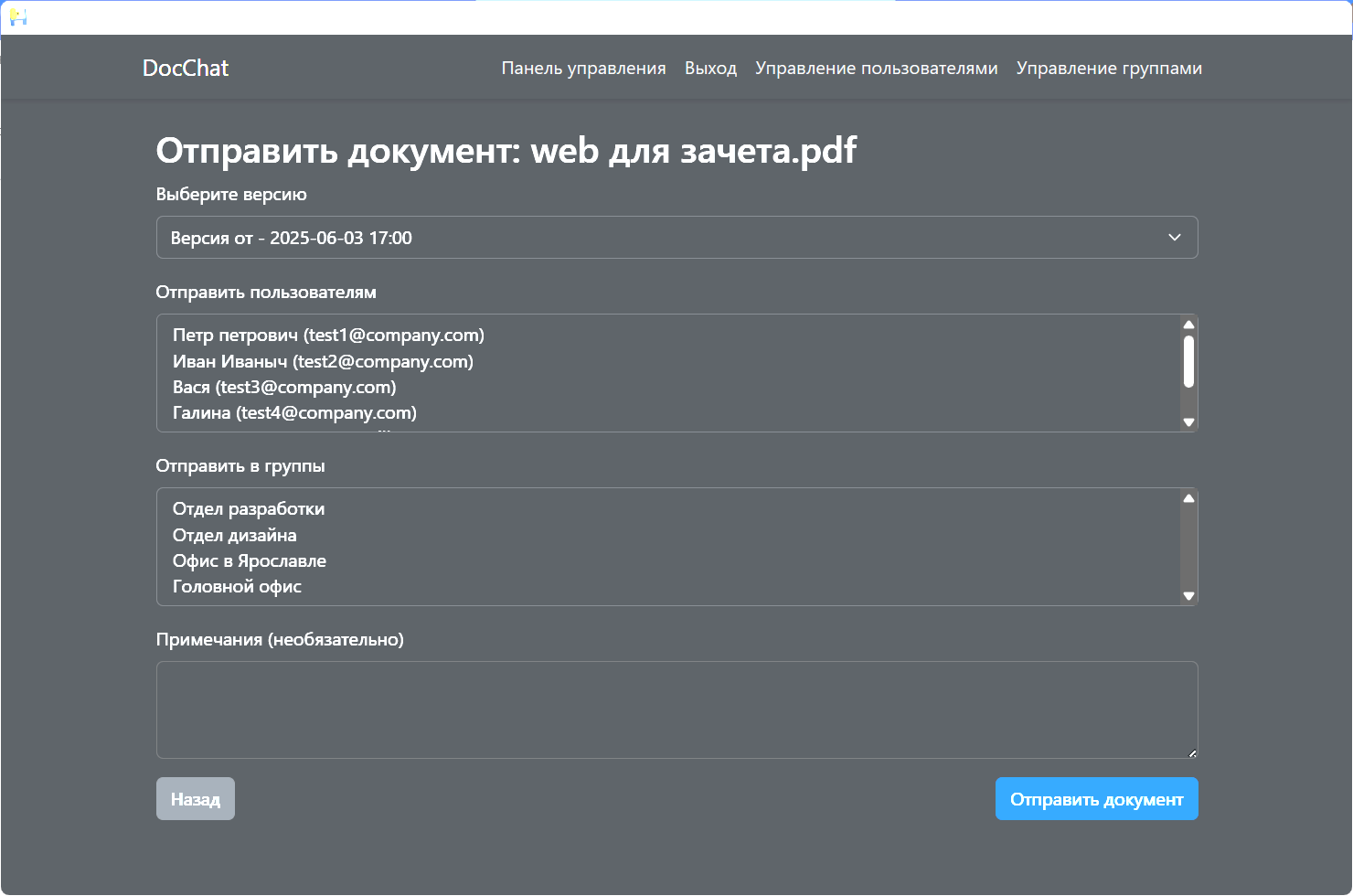


Рис. 17 Страница отправки документа

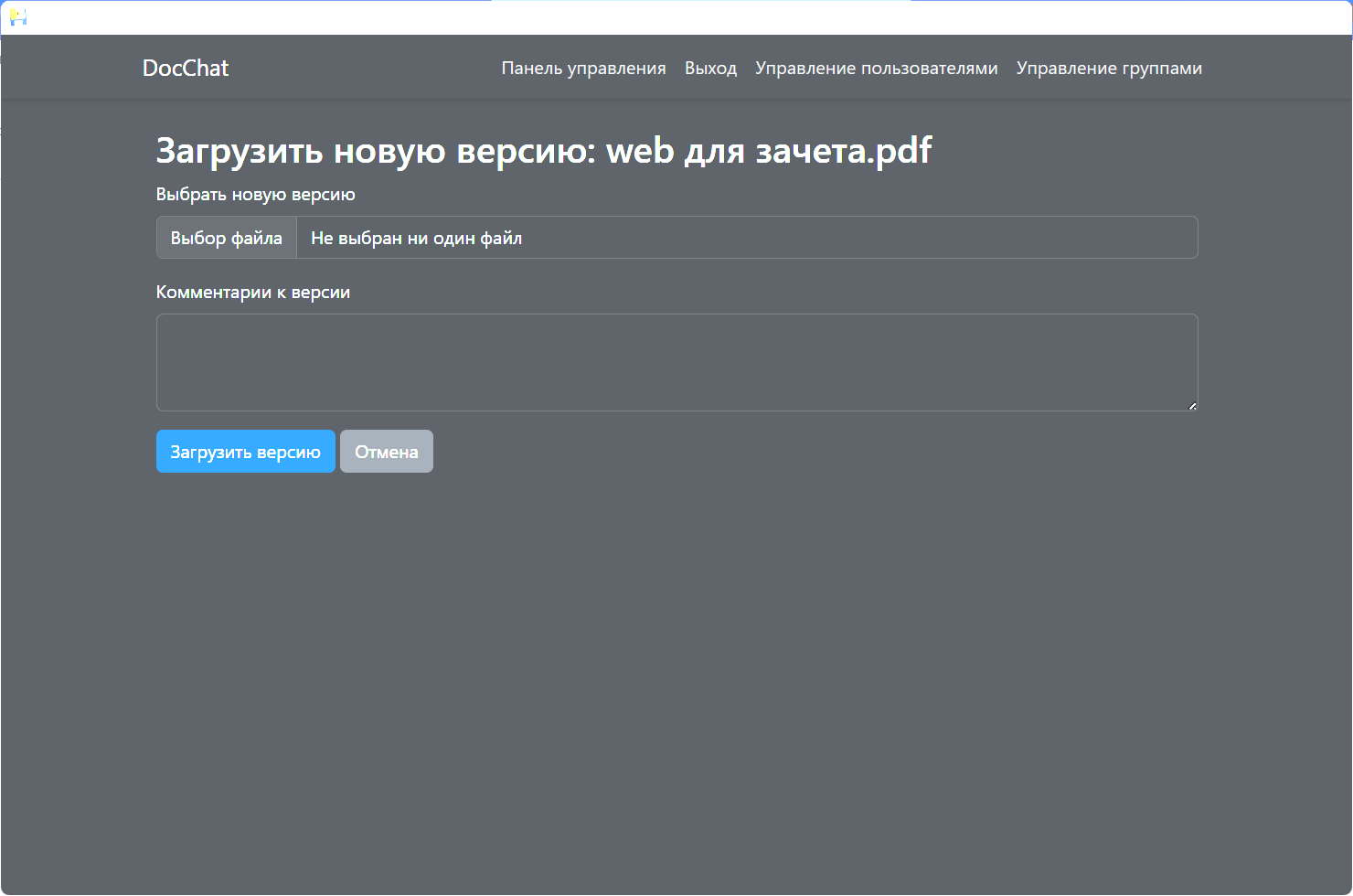


Рис. 18 Страница загрузки новой версии документа

От «Администратора»:

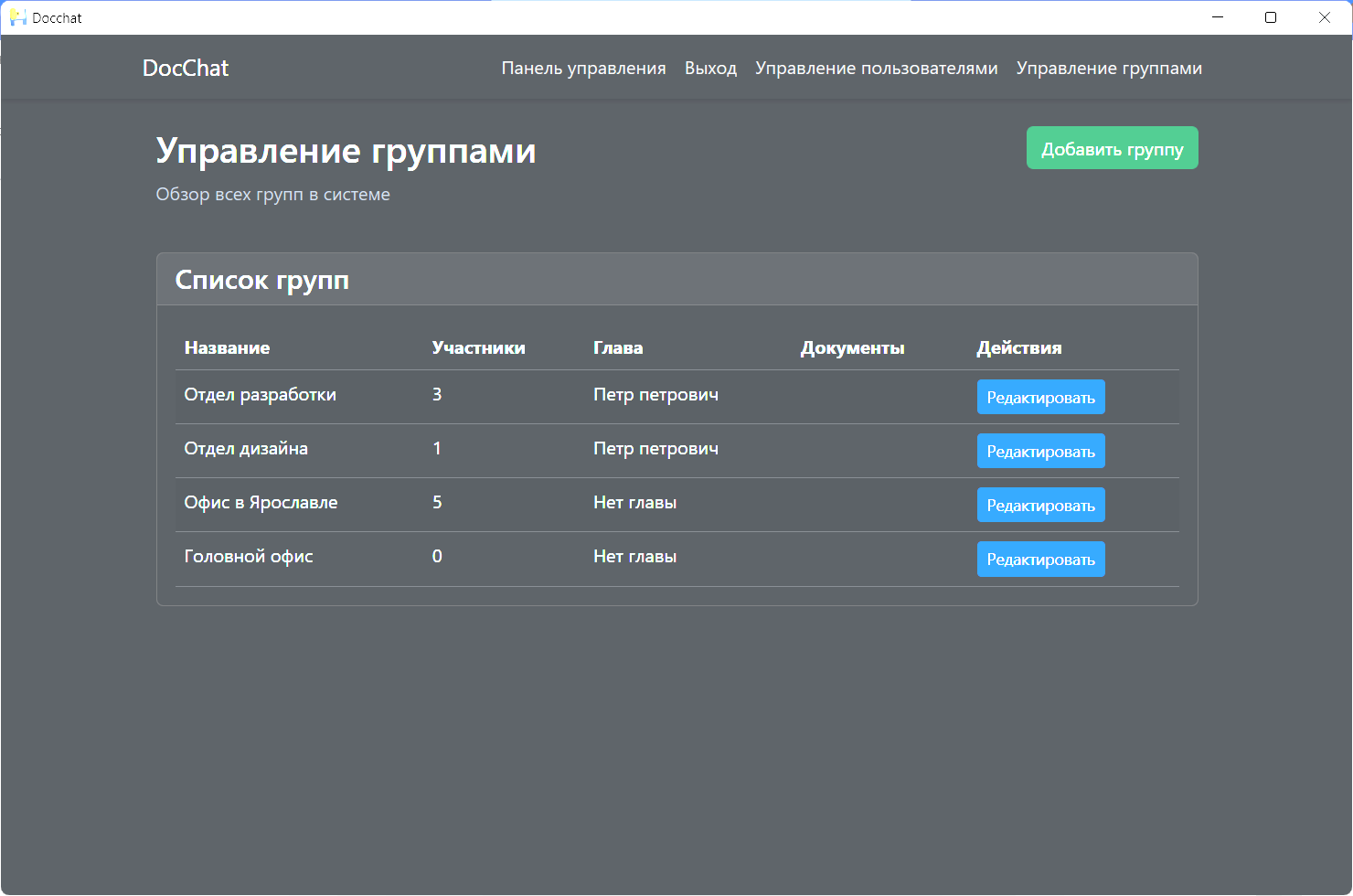


Рис. 19 Страница управления группами

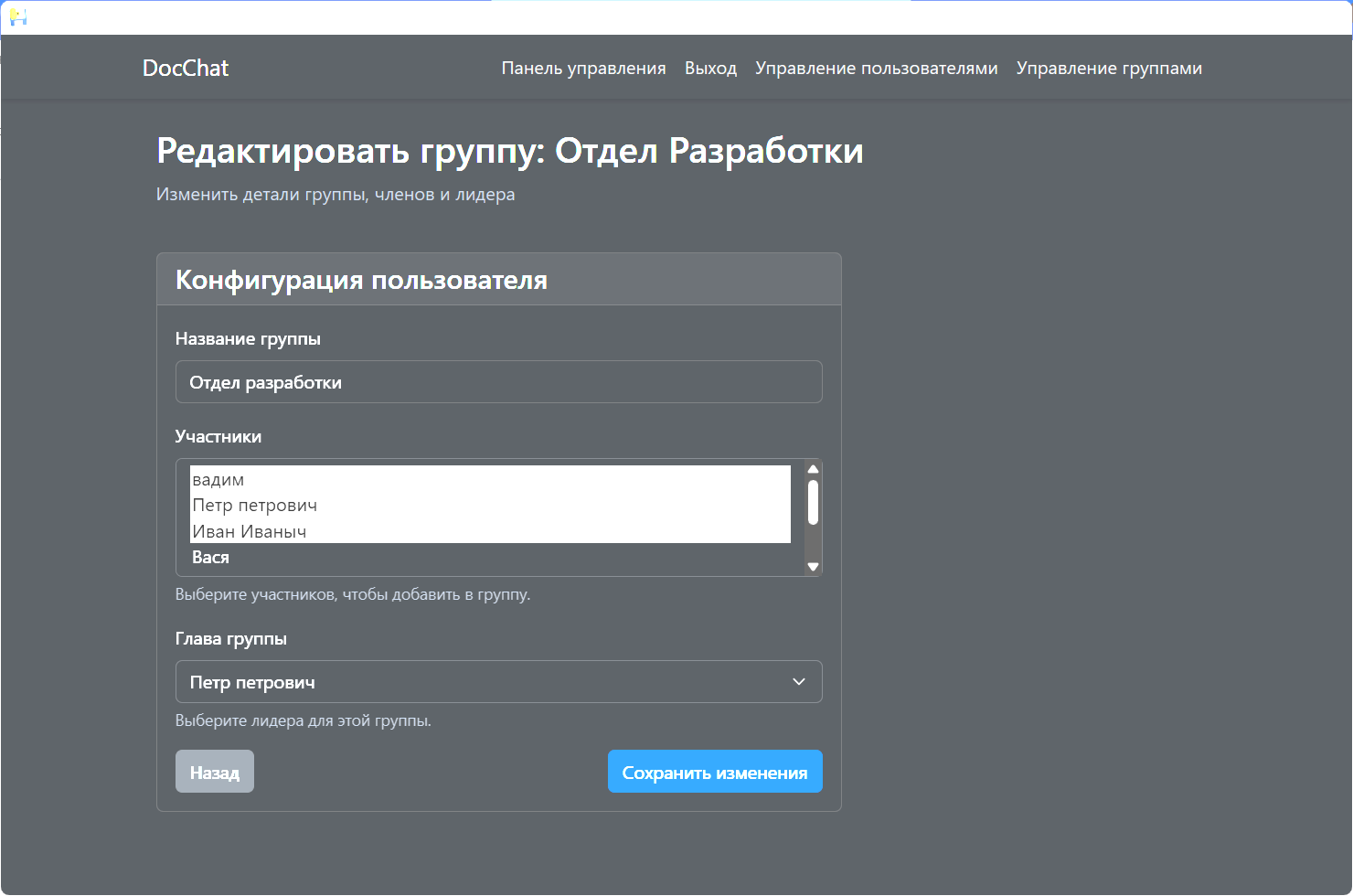


Рис. 20 Страница управления группой

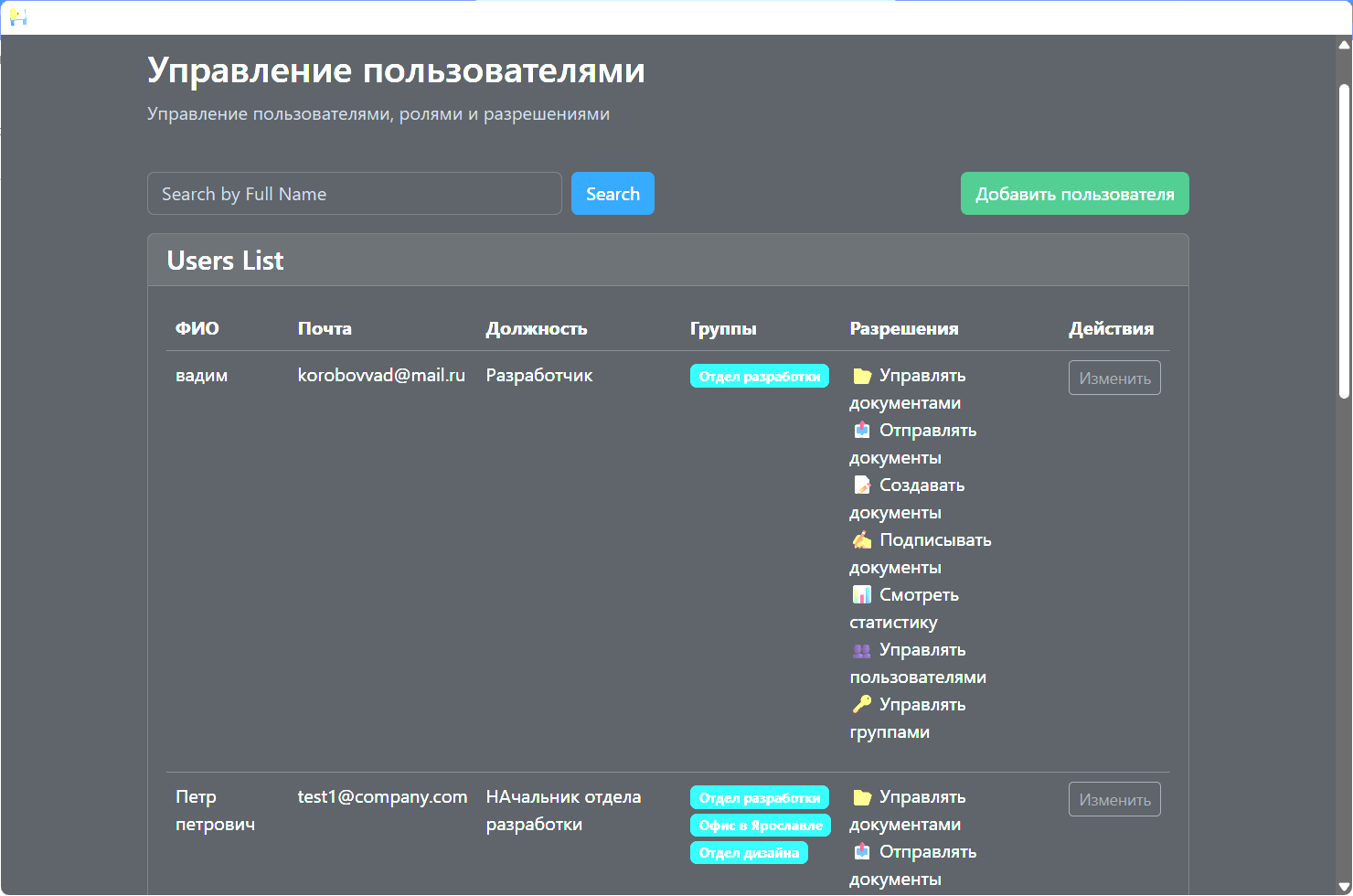


Рис. 21 Страница управления пользователями

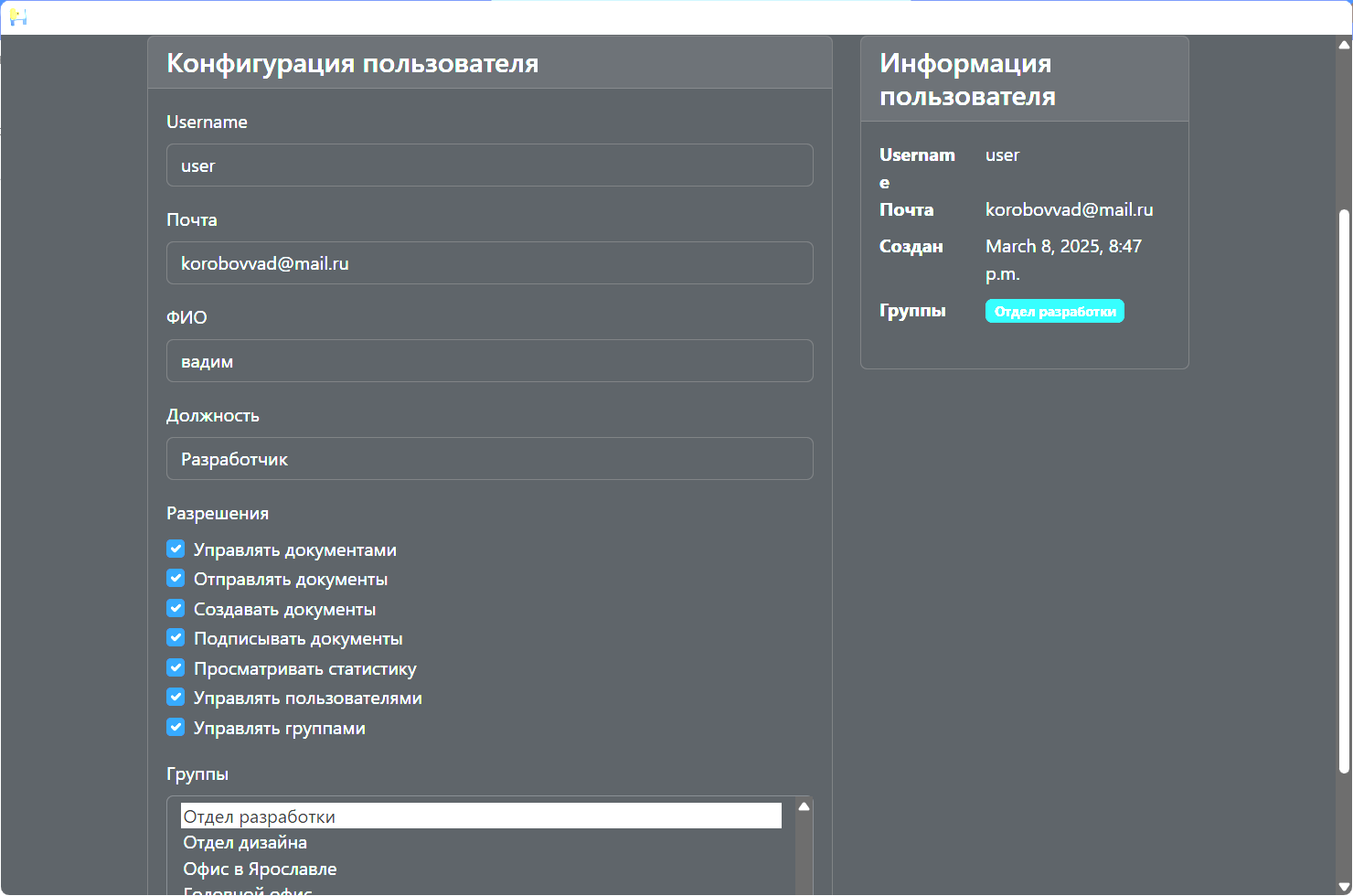


Рис. 22 Страница изменения/создания пользователя

# 3. Технологическая часть

## 3.1. Выбор программных средств

### 3.1.1. Выбор СУБД

### PostgreSQL — это мощная объектно-реляционная система управления базами данных с открытым исходным кодом. PostgreSQL поддерживает расширенный SQL-стандарт, транзакции, внешние ключи, триггеры, представления, хранимые процедуры и пользовательские типы данных. Благодаря своей надёжности, масштабируемости и расширяемости PostgreSQL широко используется как для малых приложений, так и для крупных корпоративных решений.

### PostgreSQL обеспечивает высокую производительность, гибкость в настройке и поддержку множества расширений (например, PostGIS для работы с геоданными или pgcrypto для криптографии).

### Для взаимодействия с СУБД PostgreSQL используются разнообразные клиентские инструменты.

### 3.1.2. Выбор средств разработки

PyCharm — это интегрированная среда разработки (IDE), ориентированная на язык программирования Python. Разработана компанией JetBrains. Идеально подходит для создания веб-приложений, автоматизации, научных расчётов и, в частности, для взаимодействия с базами данных.

PyCharm обеспечивает удобную разработку благодаря следующим возможностям:

Возможности PyCharm:

а) Поддержка Python-проектов — включая виртуальные окружения, управление зависимостями и интерпретаторами;

б) Автодополнение и рефакторинг кода — интеллектуальные подсказки, автоматическое завершение кода, переименование переменных, перемещение функций и т.п.;

в) Продвинутая навигация по проекту — переход к определению, поиск по символам, структурный анализ кода;

г) Индикация ошибок в реальном времени — IDE анализирует код и выделяет ошибки и предупреждения ещё до запуска;

д) Встроенная отладка и профилирование — удобный интерфейс отладки, пошаговое выполнение, просмотр значений переменных;

е) Интеграция с системами контроля версий — поддержка Git, GitHub, SVN, Mercurial и других, включая просмотр истории, слияние веток, сравнение изменений;

ж) Работа с базами данных — встроенный инструмент Database Explorer позволяет подключаться к PostgreSQL, выполнять запросы, просматривать структуру таблиц, редактировать данные;

з) Поддержка плагинов — расширение функциональности через JetBrains Plugin Marketplace;

и) Работа с фреймворками — поддержка Django, Flask, FastAPI и других популярных библиотек.

## 3.2. Описание работы с программой

Пользователь с ролью «Администратор»:

1. Авторизация по логину и паролю для входа в систему.
2. Администратор обладает максимальными правами, ему доступны все функции системы.
3. Администратор может создавать, просматривать, редактировать и удалять пользователей, группы и документы.
4. Он может управлять правами пользователей, назначать роли и изменять их.
5. Для управления документами администратор может загружать новые документы, пересылать их другим пользователям и группам, создавать новые версии и подписывать документы.
6. Администратор имеет доступ к статистике и журналу аудита, может просматривать действия всех пользователей в системе.
7. Для выхода из приложения выбирает пункт меню «Выход».

Пользователь с ролью «Менеджер»:

1. Авторизация по логину и паролю для входа в систему.
2. Менеджер имеет ограниченные права по сравнению с администратором, но может управлять документами в своем департаменте или группе.
3. Он может создавать, просматривать и редактировать документы, пересылать их другим пользователям и группам.
4. Менеджер может просматривать статистику, но не имеет прав на изменение пользователей и ролей.
5. Менеджер может создавать и управлять группами пользователей, если у него есть соответствующие права.
6. Для выхода из приложения выбирает пункт меню «Выход».

Пользователь с ролью «Пользователь» (обычный пользователь):

1. Авторизация по логину и паролю для входа в систему.
2. Пользователь может создавать документы, загружать их и просматривать свои документы.
3. Может пересылать документы другим пользователям и группам, если имеет соответствующее право.
4. Пользователь может подписывать документы, если у него есть разрешение.
5. Ограничен в возможностях по управлению пользователями и группами.
6. Для выхода из приложения выбирает пункт меню «Выход».

Дополнительные пояснения по правам (зависит от полей пользователя):

* can\_manage\_documents — право загружать, изменять и удалять собственные документы.
* can\_forward\_documents — право пересылать документы другим пользователям или группам.
* can\_create\_documents — право создавать новые документы.
* can\_sign\_documents — право подписывать документы (например, электронно).
* can\_view\_statistics — доступ к просмотру статистики по документам и действиям пользователей.
* can\_modify\_groups — возможность создавать и изменять группы пользователей.
* can\_modify\_users — возможность изменять пользователей и их права.

# Заключение

При разработке проекта учитывались все особенности информационной системы.

В результате курсового проекта были решены следующие задачи:

1. Проанализирована предметная область;
2. Спроектирована схема базы данных;
3. Создана и заполнена БД;
4. Осуществлен удобный доступ к данным базы данных;
5. Разработано web-приложение для работы пользователей;
6. Создана возможность создания отчета, в котором будут храниться данные вакансии, соискателя, компании, з/п, комиссионные и дата.

Итогом курсовой работы стала система, позволяющая реализовать электронный документооборот.

# Список использованных источников и литературы

1. Ткачев О.А. Создание и манипулирование базами данных средствами СУБД Microsoft SQL Server 2008 [Электронный pecypc]: Учебное пособие/ Ткачев О.А. – Электрон. текстовые данные. – М.: Московский городской педагогический университет, 2013. – 152 с. – Режим доступа: <http://www.bibliocomlectator.ru/book/?id=26613>
2. Чубукова И.А. Data Mining [Электронный pecypc]/ Чубукова И.А. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий

(ИНТУИТ), 2016. – 470 с. – Режим доступа: <http://www.bib1iocomp1ectator.ru/book/?id–56315>

1. [Электронный ресурс] – Режим доступа. -<https://habr.com/ru/post/326614/>
2. [Электронный ресурс] – Режим доступа. - [https://tproger.ru/articles/pishem-javaveb-prilozhenie-na-sovremennom-steke-s-nulja-do-mikroservisnoj-arhitektury-chast-](https://tproger.ru/articles/pishem-java-veb-prilozhenie-na-sovremennom-steke-s-nulja-do-mikroservisnoj-arhitektury-chast-1/)

[1/](https://tproger.ru/articles/pishem-java-veb-prilozhenie-na-sovremennom-steke-s-nulja-do-mikroservisnoj-arhitektury-chast-1/)

Приложение Б

Представление базы данных через ORM Django Python:

class Role(models.Model):  
 name = models.CharField(max\_length=50, unique=True)  
 description = models.CharField(max\_length=255, blank=True, null=True)  
 level = models.IntegerField(default=0)  
 permissions = models.JSONField(default=dict, blank=True, null=True)  
 parent = models.ForeignKey(  
 'self',  
 on\_delete=models.SET\_NULL,  
 null=True,  
 blank=True,  
 related\_name='children'  
 )  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 @classmethod  
 def init\_roles(cls):  
 roles = {  
 'admin': {'level': 3, 'description': 'Administrator with full access'},  
 'manager': {'level': 2, 'description': 'Manager with department access'},  
 'user': {'level': 1, 'description': 'Regular user with basic access'}  
 }  
 for role\_name, data in roles.items():  
 if not cls.objects.filter(name=role\_name).exists():  
 cls.objects.create(  
 name=role\_name,  
 level=data['level'],  
 description=data['description'],  
 permissions={}  
 )  
  
class CustomUser(AbstractUser):  
 email = models.EmailField(unique=True)  
 role = models.ForeignKey(  
 Role,  
 on\_delete=models.SET\_NULL,  
 null=True,  
 blank=True,  
 related\_name='users'  
 )  
 is\_email\_verified = models.BooleanField(default=False)  
 otp = models.CharField(max\_length=6, blank=True, null=True)  
 otp\_created\_at = models.DateTimeField(blank=True, null=True)  
 oauth\_provider = models.CharField(max\_length=20, blank=True, null=True)  
 oauth\_id = models.CharField(max\_length=100, blank=True, null=True)  
  
 # Дополнительные данные пользователя  
 full\_name = models.CharField("ФИО", max\_length=255, blank=True, null=True)  
 job\_title = models.CharField("Должность", max\_length=100, blank=True, null=True)  
  
 # Права пользователя (каждое можно включать/выключать)  
 can\_manage\_documents = models.BooleanField(default=True) # загрузка/выгрузка/удаление своих документов  
 can\_forward\_documents = models.BooleanField(default=False) # отправка/перенаправление документов  
 can\_create\_documents = models.BooleanField(default=True) # создание документов  
 can\_sign\_documents = models.BooleanField(default=False) # подпись документов  
 can\_view\_statistics = models.BooleanField(default=False) # доступ к статистической информации  
 can\_modify\_groups = models.BooleanField(default=False) # доступ к изменению групп  
 can\_modify\_users = models.BooleanField(default=False) # выдача прав  
  
 def generate\_otp(self):  
 self.otp = ''.join(secrets.choice('0123456789') for \_ in range(6))  
 self.otp\_created\_at = timezone.now()  
 self.save(update\_fields=['otp', 'otp\_created\_at'])  
 return self.otp  
  
 def verify\_otp(self, otp):  
 if not self.otp or not self.otp\_created\_at:  
 return False  
 time\_diff = timezone.now() - self.otp\_created\_at  
 if time\_diff.total\_seconds() > 600:  
 return False  
 return self.otp == otp  
  
 @classmethod  
 def get\_or\_create\_oauth\_user(cls, email, username, provider, provider\_id):  
 user, created = cls.objects.get\_or\_create(email=email, defaults={  
 'username': username,  
 'oauth\_provider': provider,  
 'oauth\_id': provider\_id,  
 'is\_email\_verified': True,  
 })  
 return user  
  
class UserGroup(models.Model):  
 name = models.CharField(max\_length=100, unique=True)  
 members = models.ManyToManyField(CustomUser, related\_name='custom\_groups', blank=True)  
 leader = models.ForeignKey(  
 CustomUser, on\_delete=models.SET\_NULL,  
 null=True, blank=True, related\_name='leading\_groups'  
 )  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Document(models.Model):  
 STATUS\_CHOICES = [  
 ('new', 'Новый'),  
 ('in\_progress', 'В работе'),  
 ('approved', 'Подтвержден'),  
 ('rejected', 'Отказ'),  
 ]  
  
 filename = models.CharField(max\_length=255)  
 original\_filename = models.CharField(max\_length=255)  
 content\_type = models.CharField(max\_length=50)  
 owner = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, related\_name='owned\_documents')  
 is\_encrypted = models.BooleanField(default=False)  
 upload\_date = models.DateTimeField(auto\_now\_add=True)  
 status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='new')  
  
 # Параметры отправки/общего доступа:  
 shared\_users = models.ManyToManyField(CustomUser, related\_name='shared\_documents', blank=True)  
 shared\_groups = models.ManyToManyField(UserGroup, related\_name='shared\_documents', blank=True)  
  
 def \_\_str\_\_(self):  
 return self.original\_filename  
  
class DocumentTransferHistory(models.Model):  
 ACTION\_TYPES = [  
 ('transfer', 'Transfer'),  
 ('version\_upload', 'Version Upload'),  
 ]  
 document = models.ForeignKey(Document, on\_delete=models.CASCADE, related\_name='transfer\_history')  
 sender = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, related\_name='sent\_document\_transfers')  
 recipient\_user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, null=True, blank=True, related\_name='received\_document\_transfers')  
 recipient\_group = models.ForeignKey(UserGroup, on\_delete=models.CASCADE, null=True, blank=True, related\_name='received\_document\_transfers')  
 timestamp = models.DateTimeField(auto\_now\_add=True)  
 notes = models.TextField(blank=True, null=True)  
 action\_type = models.CharField(  
 max\_length=20,  
 choices=ACTION\_TYPES,  
 default='transfer'  
 )  
 version = models.ForeignKey(  
 'DocumentVersionHistory',  
 on\_delete=models.SET\_NULL,  
 null=True,  
 blank=True,  
 related\_name='transfer\_histories'  
 )  
 def \_\_str\_\_(self):  
 return f"Transfer of {self.document} from {self.sender} at {self.timestamp}"  
  
class AuditLog(models.Model):  
 user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, related\_name='audit\_logs')  
 action = models.CharField(max\_length=50)  
 timestamp = models.DateTimeField(auto\_now\_add=True)  
 details = models.TextField()  
  
 def \_\_str\_\_(self):  
 return f"{self.user} - {self.action} at {self.timestamp}"  
  
  
class DocumentVersionHistory(models.Model):  
 document = models.ForeignKey(  
 Document,  
 on\_delete=models.CASCADE,  
 related\_name='version\_history'  
 )  
 version\_id = models.CharField(max\_length=255)  
 version\_url = models.TextField(null=True)  
 file\_size = models.BigIntegerField(null=True, blank=True)  
 etag = models.CharField(max\_length=100, blank=True, null=True)  
 timestamp = models.DateTimeField(default=timezone.now)  
 notes = models.TextField(blank=True, null=True)  
  
 def \_\_str\_\_(self):  
 return f"Version {self.version\_id} of {self.document.original\_filename} at {self.timestamp}"