



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Московский техникум космического приборостроения

Отделение _____ ИСиП _____

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

по профессиональному модулю ПМ.02 Осуществление интеграции программных модулей

Код, Специальность _____ 09.02.07 _____ Администратор баз данных _____

Место прохождения практики _____ АО «ЦНИИАГ» _____
(полное название организации)

Выполнил студент _____ Пичугина Софья Павловна _____
(фамилия, имя, отчество)

Курс _____ 3 _____ Группа _____ ТБД - 61 _____

Подпись студента _____

Оценка _____

Дата приема отчета _____ 2025 г.

Руководитель практики от техникума _____
(подпись) _____ (фамилия,
имя, отчество)

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	2
1 Постановка задачи.....	3
2 Описание предметной области.....	4
3 Моделирование проектируемой системы.....	7
4 Разработка функциональных требований.....	9
5 Обоснование выбора средств реализации.....	11
6 Структура программы.....	12
7 Схема данных.....	13
8 Схемы алгоритмов основной программы и подпрограмм.....	15
8.1 Схема алгоритма основной программы.....	15
8.2 Схема алгоритма подпрограммы авторизации пользователя.....	16
8.3 Схема алгоритма подпрограммы выдачи документа менеджеру.....	18
9 Тестирование и отладка программы.....	19
ЗАКЛЮЧЕНИЕ.....	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	23
ПРИЛОЖЕНИЕ А.....	24
(обязательное).....	24
ПРИЛОЖЕНИЕ Б.....	37
(обязательное).....	37

ВВЕДЕНИЕ

В современном мире, характеризующемся стремительной автоматизацией бизнес-процессов и возрастающими объемами электронной документации, вопросы эффективного и безопасного управления доступом к корпоративным информационным ресурсам выходят на первый план. Надежная защита конфиденциальных данных, обеспечение их целостности и доступности для авторизованных пользователей являются критически важными задачами для любой организации. Системы контроля доступа выступают ключевым инструментом в решении этих задач, определяя, кто, к каким ресурсам и при каких условиях имеет право доступа.

Актуальность разработки графического приложения для управления доступом к документам определяется несколькими ключевыми причинами. Во-первых, централизация управления правами необходима, так как ручное назначение прав через разные системы или файловые менеджеры неэффективно, часто приводит к ошибкам и плохо работает при увеличении числа документов и пользователей. Во-вторых, такое приложение повышает безопасность: наглядное отображение прав и простые инструменты для их изменения снижают риск ошибочного или несанкционированного доступа к важным данным. В-третьих, графический интерфейс (GUI) существенно упрощает и ускоряет работу администраторов по сравнению с использованием командной строки или редактированием сложных конфигурационных файлов. Наконец, приложение обеспечивает прозрачность за счет возможности легко просматривать текущие права доступа к документам, а в перспективе – и историю их изменений, что важно для контроля и аудита.

Основной целью производственной практики стала разработка графического приложения, предназначенного для централизованного и эффективного управления правами доступа пользователей и групп к документам. Данное приложение должно предоставлять администраторам удобный и наглядный инструмент на основе графического интерфейса (GUI), позволяющий упростить и ускорить процессы назначения, изменения и контроля прав доступа, а также обеспечить большую прозрачность и безопасность при работе с корпоративными документами. Приложение должно взаимодействовать с системой хранения документов для получения актуальной информации и применения изменений прав.

1 Постановка задачи

Для достижения поставленной цели – разработки графического приложения для централизованного управления доступом к документам – в ходе производственной практики были решены следующие задачи:

1. Проанализировать требования и существующие процессы: Изучить, как в компании в настоящее время управляют правами доступа к документам, и определить конкретные потребности и ожидания от нового приложения.

2. Спроектировать структуру приложения и базу данных: Определить, как будет организована информация о пользователях, группах, документах и их правах, и спроектировать схему базы данных для хранения этих данных.

3. Разработать пользовательский интерфейс (GUI): Создать макеты и реализовать основные экраны приложения, обеспечивающие:

- просмотр иерархии документов и папок;
- управление списками пользователей и групп;
- назначение, изменение и отзыв прав доступа (например, чтение, запись, удаление) для выбранных документов, папок, пользователей или групп;
- просмотр текущих назначенных прав для конкретного документа, папки или пользователя;

4. Реализовать взаимодействие с системой хранения документов: Обеспечить связь приложения с тем местом, где физически хранятся документы (например, с базой данных, файловым сервером или системой электронного документооборота), чтобы получать данные о документах и применять изменения прав доступа.

5. Протестировать функциональность приложения: Проверить работу реализованных функций (добавление пользователей, назначение прав, отображение структуры и т.д.) на соответствие требованиям и выявить возможные ошибки.

2 Описание предметной области

В рамках производственной практики была разработана система автоматизации документооборота для АО "ЦНИИАГ" — ведущего российского предприятия в области разработки систем управления для авиационной и космической техники. Особое внимание уделялось вопросам информационной безопасности и разграничения доступа к документам, что особенно актуально для организации, работающей с секретными материалами в условиях современных технологических вызовов.

АО "ЦНИИАГ" занимается созданием сложных технических систем для авиации, космоса и оборонного комплекса. Деятельность института включает разработку автоматизированных систем управления для самолетов и вертолетов, проектирование компонентов ракетной техники, создание бортовых компьютеров и систем наведения. Все эти направления работы требуют строгого контроля за документацией, которая содержит технические решения и ноу-хау предприятия.

АО "ЦНИИАГ" занимает уникальное положение в российской промышленности, осуществляя полный цикл разработки сложных систем управления — от фундаментальных научных исследований до серийного производства. Основные направления деятельности института требуют особого подхода к работе с документацией:

Авиационные системы включают разработку бортовых компьютеров, систем автоматического управления полетом, комплексных решений для беспилотных летательных аппаратов. Каждый такой проект сопровождается созданием тысяч страниц технической документации, содержащей уникальные инженерные решения.

Космические технологии — это прежде всего системы управления ракетными комплексами и космическими аппаратами, где требования к точности и надежности исключительно высоки. Документация по таким проектам имеет особый статус и подлежит строгому учету.

Оборонные заказы предполагают работу с материалами, содержащими сведения, составляющие государственную тайну. Здесь важна не только защита от внешних угроз, но и тщательный контроль внутреннего доступа.

Необходимость создания специализированной системы документооборота продиктована особенностями работы с материалами различной степени секретности. В организации существует четкая иерархия доступа, где каждый сотрудник может работать

только с документами, соответствующими его уровню допуска. При этом традиционные системы электронного документооборота не обеспечивают необходимой гибкости в управлении правами доступа и не соответствуют требованиям информационной безопасности, предъявляемым к оборонным предприятиям.

Разработанное программное решение построено на работе с БД, что обеспечивает надежное хранение данных и гибкость в управлении доступом. Централизованное хранение документов в защищенной базе данных решает проблемы целостности информации и позволяет вести детальный аудит всех действий с документами. Особое внимание уделено производительности системы — оптимизированные запросы и продуманная индексация обеспечивают быстрый доступ даже к большому массиву документов.

Система реализует трехуровневую модель доступа, где права пользователей строго соответствуют их должностным обязанностям. Наблюдатели (viewer) могут только просматривать документы минимального уровня секретности. Менеджеры (manager) имеют расширенные права, включая возможность изменять уровень доступа к документам, тем самым регулируя их доступность для других сотрудников. Администраторы (admin) обладают полным контролем над системой — от управления пользователями до работы с документами любого уровня секретности.

Ключевой особенностью системы является глубокая интеграция механизмов безопасности. Все пароли хранятся в хешированном виде, критически важные данные шифруются, а каждое действие в системе фиксируется в журнале событий. Это позволяет не только предотвратить утечку информации, но и быстро выявлять попытки несанкционированного доступа.

Внедрение данной системы документооборота позволит АО "ЦНИИАГ" существенно повысить эффективность работы с конфиденциальными материалами. Автоматизация процессов управления документами сократит время на поиск и согласование информации, минимизирует человеческий фактор при контроле доступа и обеспечит соответствие строгим требованиям по защите государственной тайны. В перспективе система может быть интегрирована с другими корпоративными решениями и системами электронного документооборота государственных органов.

Разработанная система управления документооборотом представляет собой не просто программный продукт, а комплексное организационно-техническое решение, учитывающее все особенности работы предприятия с конфиденциальной информацией. Внедрение системы позволит:

- 1) Повысить эффективность работы с документами на 30-40%.

- 2) Снизить риски утечки информации.
- 3) Обеспечить соответствие требованиям регуляторов.
- 4) Создать единое информационное пространство предприятия.

3 Моделирование проектируемой системы

Проектирование – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Обычно они пытаются удержать всё в голове или, в лучшем случае, записать некоторые важные сведения на листе бумаги. Как результат, у них нет чёткого плана дальнейших действий, и проект может быть отложен в долгий ящик.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобратся в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег.

На рисунках 3.1-3.2 представлен вариант организации автоматизации процессов.

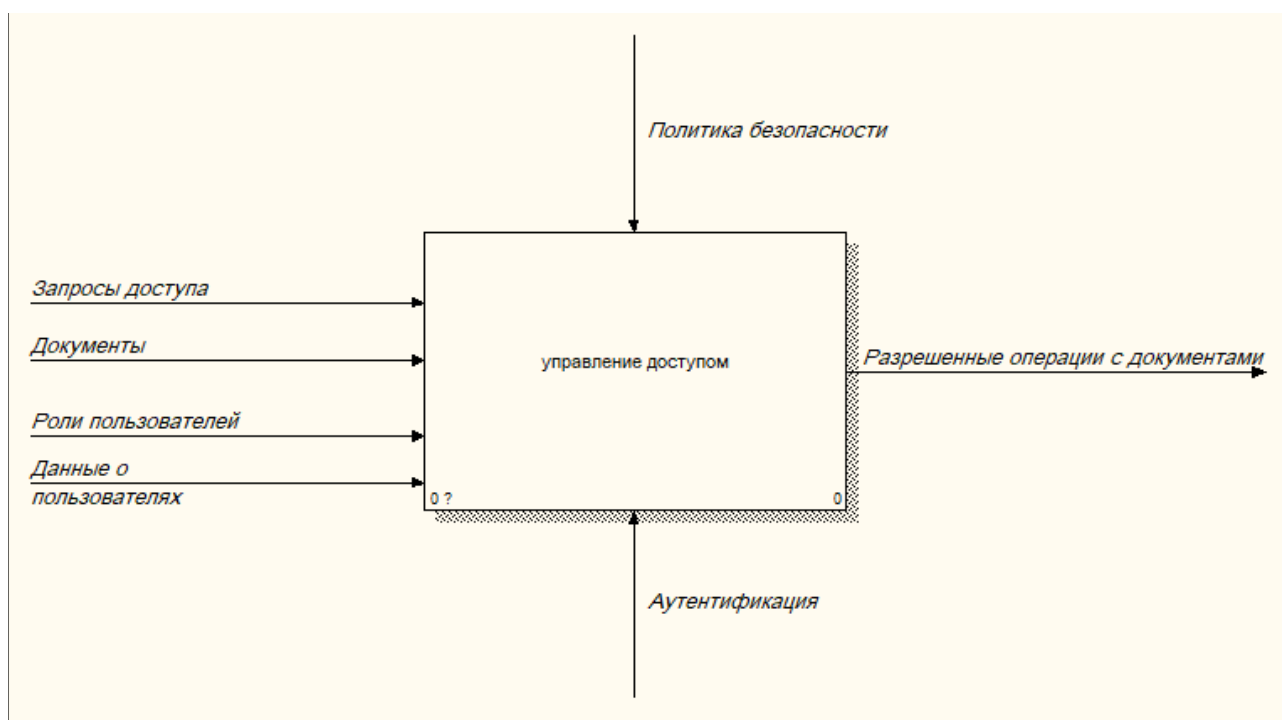


Рисунок 3.1 — контекстная диаграмма

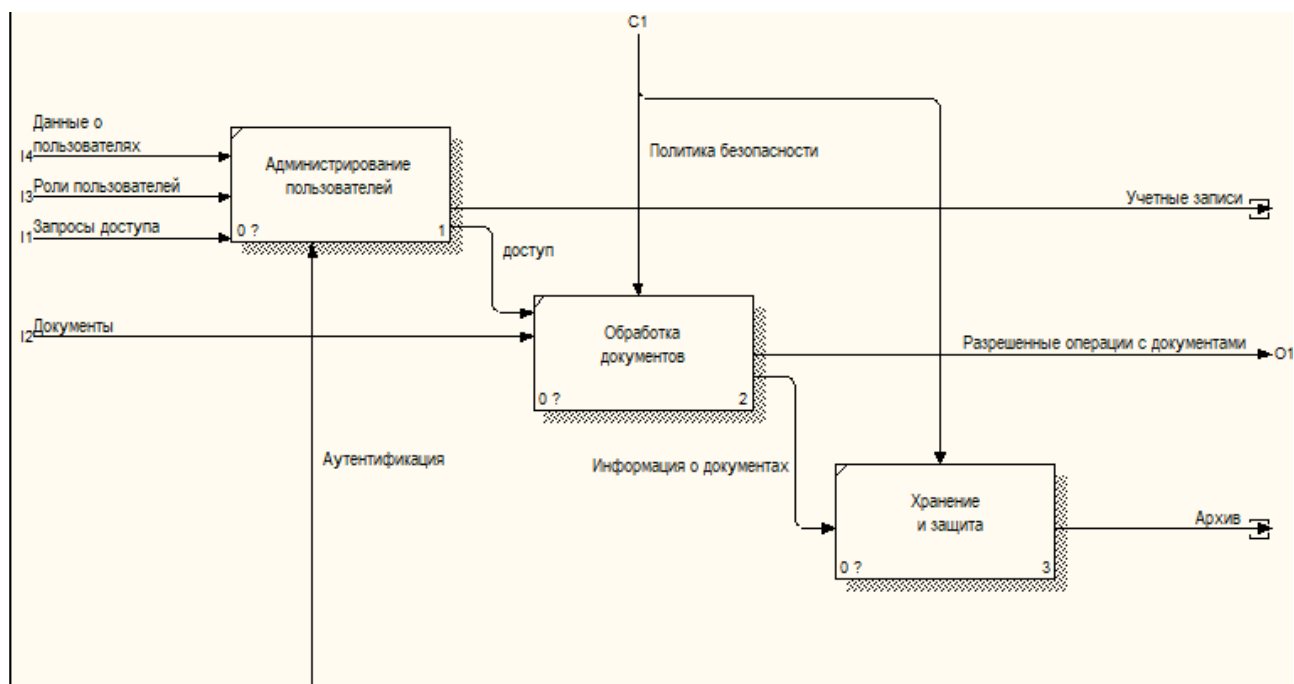


Рисунок 3.2 — Диаграмма декомпозиции

Помимо этого, необходимо построить диаграмму вариантов использования. Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. Построение данной диаграммы позволит четко определить необходимый функции и иметь наглядное представление о работе программного обеспечения. Диаграмма вариантов использования изображена на рисунке 3.3.

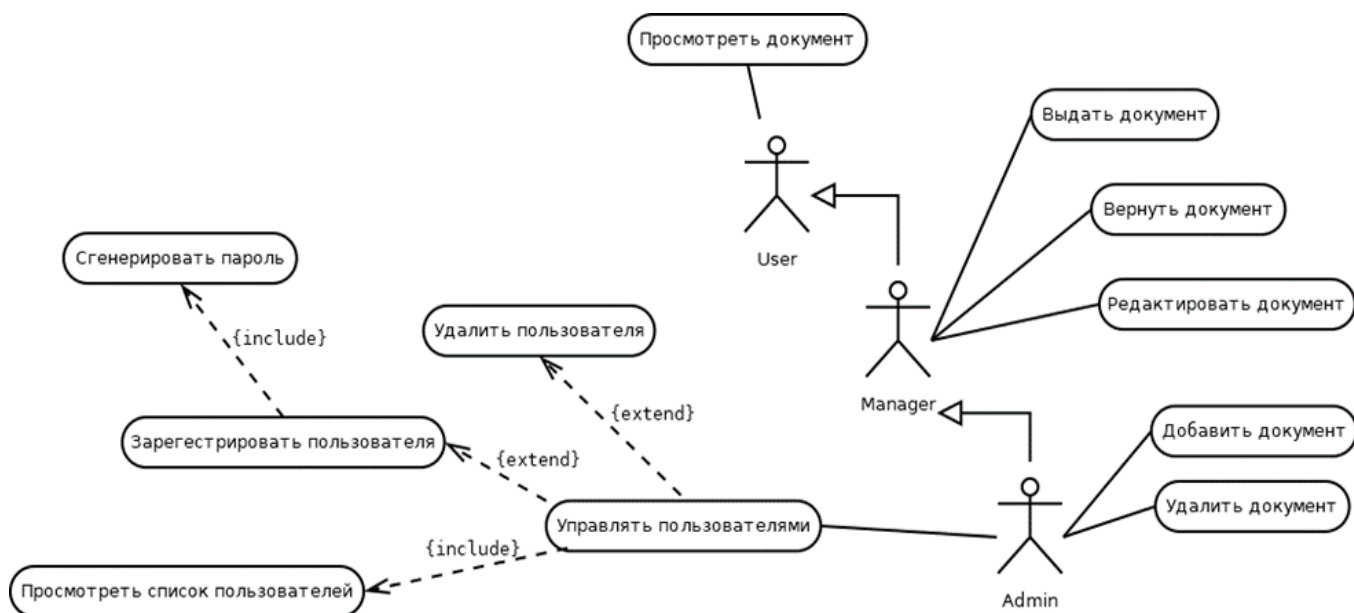


Рисунок 3.3 — Диаграмма вариантов использования

4 Разработка функциональных требований

Назначение программы.

Разрабатываемая программа будет хранить информацию о сотрудниках, документах и уровнях доступа. Будет производить добавление и выдачу прав сотрудникам для доступа к существующим документам.

Программа будет использоваться администратором для управления документами и сотрудниками, с целью автоматизации процессов доступа к документам.

Сотрудники в свою очередь могут использовать программу для быстрого доступа ко всем существующим документам с соответствующим уровнем доступа, для быстрого поиска существующего документа (по названию, по категории, по дате).

Разрабатываемое программное обеспечение обязано будет иметь следующий функционал:

- 1) Надежное хранение данных.
- 2) Пользовательский интерфейс для работы с документами.
- 3) Удобная панель администратора.
- 4) Авторизация пользователей.
- 5) Разграничение уровней доступа сотрудников.
- 6) Проверка вводимых данных.

Требования к надежности.

Для обеспечения надежности, средства ввода данных в систему должны обеспечивать контроль правильности данных по их типу. При изменении и удалении данных, средства программы должны запрашивать подтверждение правильности выданных команд. Надежность технических средств обеспечивается использованием сертифицированных средств вычислительной техники, их комплектующих и средств передачи данных. Прикладные программы должны иметь защиту от некорректных действий пользователей. Должно производиться регулярное обновление системы. Должно быть осуществлено разграничение прав доступа по пользователям.

Требования к составу и параметрам технических средств.

Требования к составу и параметрам технических средств включают в себя:

- 1) Персональный компьютер.
- 2) Процессор с тактовой частотой не менее 2 ГГц.
- 3) Оперативную память объемом не менее 1 Гигабайт.
- 4) MySQL Server 8.0.31.

- 5) Операционная система Windows 10 и больше.
- 6) Жесткий диск емкостью не менее 1 ТБ.
- 7) Любой браузер.
- 8) Текстовые редакторы для работы с файлами.

Требования к информационной и программной совместимости.

Для эксплуатации программного обеспечения необходимо наличие следующих компонентов: Операционная система семейства Microsoft®Windows®, установленные и сконфигурированные программные продукты PostgreSQL. Это обеспечит информационную и программную совместимость.

Требования к программной документации. Программная документация должна содержать:

- 1) Техническое задание.
- 2) Диаграммы, визуализирующие назначение программы.
- 3) Программа и методика испытаний.

5 Обоснование выбора средств реализации

PostgreSQL была выбрана в качестве основной системы управления базами данных благодаря своей надежности, масштабируемости и расширенной функциональности. Это реляционная СУБД с открытым исходным кодом, которая поддерживает сложные запросы, транзакции и целостность данных.

Ключевые преимущества PostgreSQL:

- масштабируемость и производительность – PostgreSQL эффективно работает с большими объемами данных, поддерживает индексы, партиционирование и оптимизированные запросы;
- кроссплатформенность – PostgreSQL работает на Linux, Windows, macOS и других ОС, что делает ее универсальным решением;
- расширяемость – поддерживает хранимые процедуры, триггеры, пользовательские функции на нескольких языках (PL/pgSQL, Python, JavaScript и др.);

Для взаимодействия с PostgreSQL была выбрана библиотека SQLAlchemy, которая имеет ряд преимуществ:

- абстракция от СУБД – можно легко перейти с PostgreSQL на MySQL, SQLite или другую СУБД без переписывания кода;
- гибкость – поддерживает как высокоуровневые ORM-запросы, так и "сырой" SQL;
- безопасность – предотвращает SQL-инъекции за счет параметризованных запросов;
- удобное управление транзакциями – автоматический commit/rollback в зависимости от успешности операций;

Python был выбран благодаря своей простоте, читаемости и мощной экосистеме.

Основные преимущества Python

- простота и скорость разработки – лаконичный синтаксис позволяет быстро писать код с минимальными накладными расходами;
- широкая поддержка веб-разработки – фреймворки Django, Flask, FastAPI позволяют создавать как небольшие API, так и сложные веб-приложения;
- кроссплатформенность – Python работает на Windows, Linux, macOS, что важно для развертывания в разных средах;

6 Структура программы

На рисунке 6.1 представлена структура программы для управления доступом к документам.

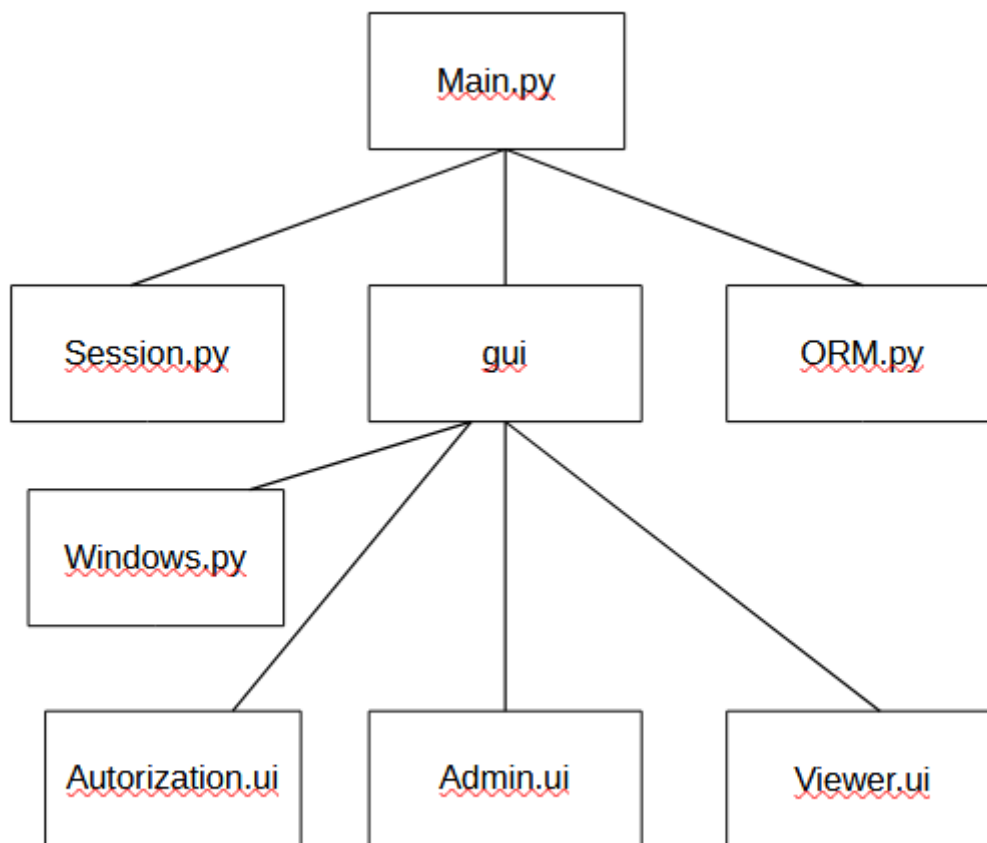


Рисунок 6.1 — структура программы

7 Схема данных

Для создания БД необходимо описать назначение таблиц, их количество и структуру.

1) Таблица roles (роли) хранит информацию о существующих в системе ролях и их идентификационный номер.

2) Таблица users (пользователи) хранит информацию о пользователях системы, в том числе: уникальный идентификационный номер, ФИО пользователя, адрес электронной почты, который будет являться логином пользователя, пароль и id роли, выданной пользователю администратором.

3) Таблица documents (документы) хранит информацию о документах. В таблице содержатся: уникальный идентификационный номер, название, категория, дата создания документа, уровень секретности, путь к файлу и id создателя документа.

На рисунке 7.1 представлена диаграмма IDEF1X описывающая структуру таблиц БД.

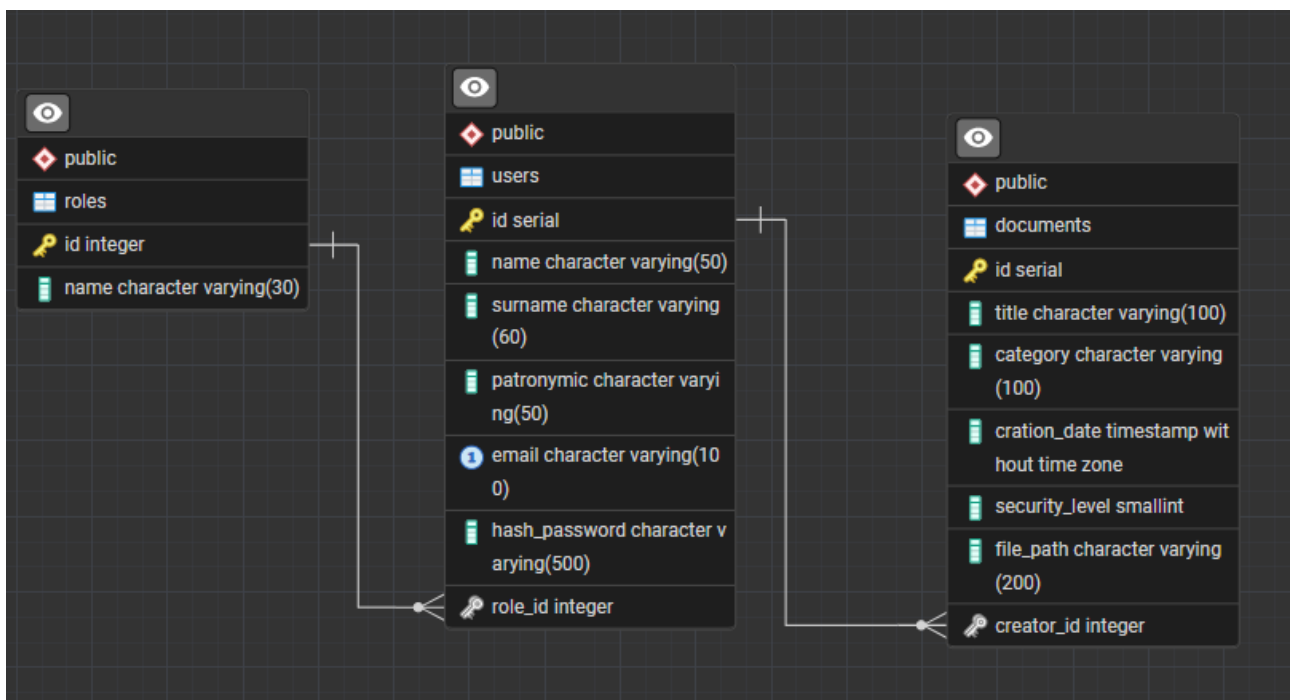


Рисунок 7.1 — диаграмма IDEF1X

В таблицах 7.2 — 7.4 представлены атрибуты сущностей.

Таблица 7.2 — Сущность «Роли»

Идентификатор	Описание	Тип данных	Размер
id	Уникальный идентификатор роли, PRIMARY KEY	SERIAL	
name	Наименование роли, NOT NULL	CHARACTER VARYING	30

Таблица 7.3 — Сущность «Пользователи»

Идентификатор	Описание	Тип данных	Размер
id	Уникальный идентификатор пользователя, PRIMARY KEY	SERIAL	
name	Имя пользователя, NOT NULL	CHARACTER VARYING	50
surname	Фамилия пользователя, NOT NULL	CHARACTER VARYING	60
patronymic	Отчество пользователя	CHARACTER VARYING	50
email	Адрес электронной почты, логин, UNIQUE NOT NULL	CHARACTER VARYING	100
hash_password	Захэшированный пароль пользователя, NOT NULL	CHARACTER VARYING	500
role_id	Идентификатор роли, NOT NULL FOREIGN KEY	INTEGER	

Таблица 7.4 — Сущность «Документы»

Идентификатор	Описание	Тип данных	Размер
id	Уникальный идентификатор документа, PRIMARY KEY	SERIAL	
title	Наименование документа, NOT NULL	CHARACTER VARYING	100
category	Категория документа, NOT NULL	CHARACTER VARYING	100
creation_date	Дата создания документа, NOT NULL	TIMESTAMP WITHOUT TIMEZONE	
security_level	Уровень защиты, NOT NULL	SMALLINT	
file_path	Путь к файлу, NOT NULL	CHARACTER VARYING	100
creator_id	Идентификатор создателя, NOT NULL FOREIGN KEY	INTEGER	

8 Схемы алгоритмов основной программы и подпрограмм

8.1 Схема алгоритма основной программы



Рисунок 8.1 — Схема алгоритма основной программы

8.2 Схема алгоритма подпрограммы авторизации пользователя

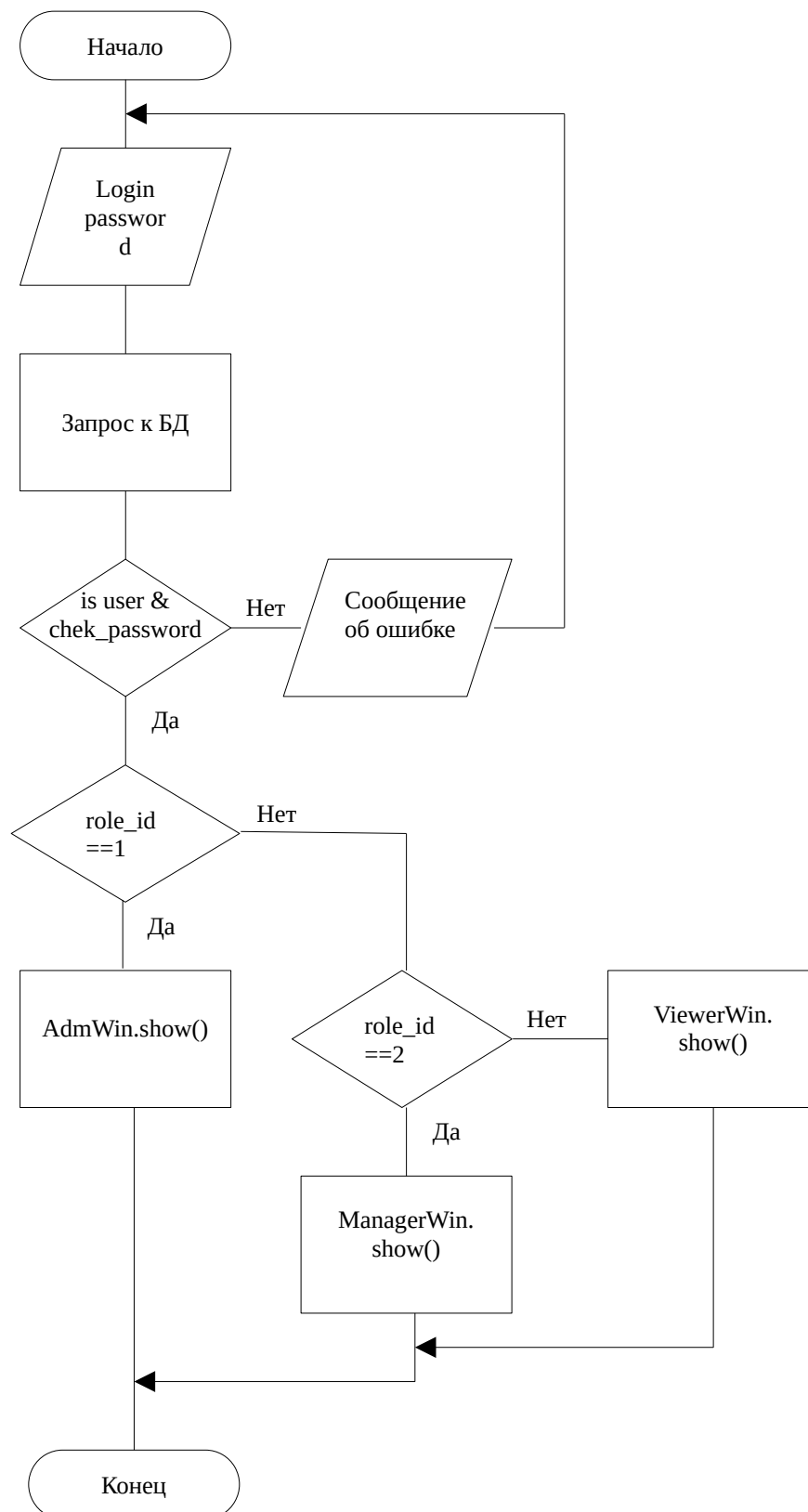


Рисунок 8.2 — Схема подпрограммы login

В таблице 8.3 представлено описание переменных, используемых в подпрограмме login.

Таблица 8.3 — Описание переменных

Идентификатор	Описание	Тип данных
login	Логин пользователя, электронная почта	str
password	Пароль пользователя	str
user	Объект класса UsersORM, данные о пользователе из БД	Class object <UsersORM>
role_id	Идентификатор роли конкретного пользователя, взятый из БД	int

8.3 Схема алгоритма подпрограммы выдачи документа менеджеру

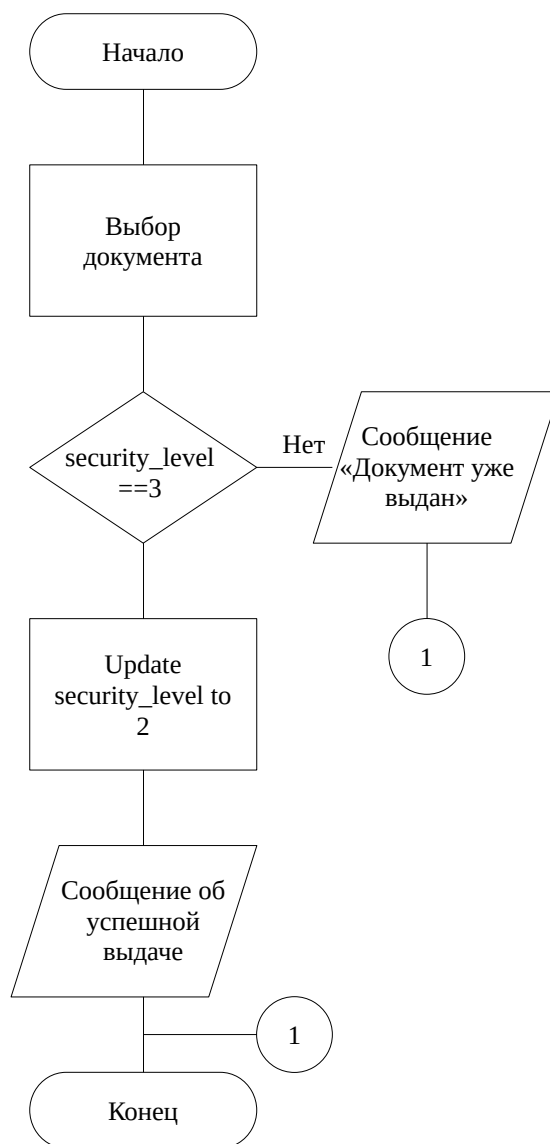


Рисунок 8.4 — схема подпрограммы issue_to_manager

В таблице 8.5 представлено описание переменных подпрограммы issue_to_manager

Таблица 8.5 — Описание переменных

Идентификатор	Описание	Тип данных
security_level	Уровень защиты документа, значение берется из БД	int

9 Тестирование и отладка программы

Проведем тестирование работы формы авторизации, регистрации новых пользователей и добавления документа в систему. Для этого напишем позитивные и негативные тест кейсы.

В таблицах 9.1-9.3 представлены позитивные тест кейсы

Таблица 9.1 — Позитивный тест кейс авторизации

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Авторизация»	<ul style="list-style-type: none">– Форма открыта– Все поля по умолчанию пусты– Кнопка «Войти» активна	Выполнено
2. Заполняем поля формы Логин = user@example.com Пароль = qwety123	<ul style="list-style-type: none">– Поля заполнены	Выполнено
3. Нажимаем «Войти»	<ul style="list-style-type: none">– Данные передаются в базы данных– Переход на главную форму в авторизованном режиме	Выполнено

Таблица 9.2 — Позитивный тест кейс регистрации

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Admin.users»	<ul style="list-style-type: none">– Форма открыта– Все поля по умолчанию пусты– Кнопка «Добавить» активна– ComboBox активен	Выполнено
2. Заполняем поля формы Имя = Andrey Фамилия = Ivanov Отчество = Egorovich email = andreyy@mail.ru role = 3-viewer	<ul style="list-style-type: none">– Поля заполнены	Выполнено
3. Нажимаем «Добавить»	<ul style="list-style-type: none">– Сообщение об успешном добавлении пользователя– Данные передаются в БД– Новый пользователь отображается в таблице просмотра пользователей	Выполнено

Таблица 9.3 — Позитивный тест кейс добавления документа

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Admin.documents»	<ul style="list-style-type: none"> – Форма открыта – Таблица документов содержит информацию – Кнопка «Добавить документ» активна 	Выполнено
2. Нажимаем кнопку «Добавить документ»	<ul style="list-style-type: none"> – Кнопка неактивна – Открыта форма добавления документа – Поля пустые 	Выполнено
3. Заполняем поля	<ul style="list-style-type: none"> – Поля заполнены – Кнопка «Выбрать файл» активна 	Выполнено
4. Нажимаем кнопку «Выбрать файл»	<ul style="list-style-type: none"> – Кнопка неактивна – Открыто меню выбора файлов – Можно выбрать любой файл 	Выполнено
5. Нажимаем кнопку «Выбрать»	<ul style="list-style-type: none"> – Кнопка неактивна – Поле «Путь к файлу» заполнено» – Путь к файлу верный 	Выполнено
6. Нажимаем кнопку «Добавить документ»	<ul style="list-style-type: none"> – Форма добавления документа закрыта – Сообщение об успешном добавлении – Новый документ отображается в таблице для просмотра документов 	Выполнено

В таблицах 9.4-9.6 представлены негативные тест кейсы.

Таблица 9.4 — Негативный тест кейс авторизации

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Авторизация»	<ul style="list-style-type: none"> – Форма открыта – Все поля по умолчанию пусты – Кнопка «Войти» активна 	Выполнено
3. Заполняем поля формы Логин = user@example.com Пароль = qwety	<ul style="list-style-type: none"> – Поля заполнены 	Выполнено
3. Нажимаем «Войти»	<ul style="list-style-type: none"> – Сообщение «Неверный логин или пароль» 	Выполнено

Таблица 9.5 — Негативный тест кейс регистрации

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Admin.users»	<ul style="list-style-type: none"> – Форма открыта – Все поля по умолчанию пусты – Кнопка «Добавить» активна – ComboBox активен 	Выполнено
2. Заполняем поля формы <ul style="list-style-type: none"> – Имя = Andrey – Фамилия = Ivanov – Отчество = Egorovich – email = andreyy@mail.ru – role = 3-viewer 	<ul style="list-style-type: none"> – Поля заполнены 	Выполнено
3. Нажимаем «Добавить»	<ul style="list-style-type: none"> – Сообщение «Данный пользователь уже есть в системе» – Данные не переданы в БД – В таблицу не добавлен новый пользователь 	Выполнено

Таблица 9.6 — Негативный тест кейс добавления документа

Действие	Ожидаемый результат	Результат теста
7. Открываем форму «Admin.documents»	<ul style="list-style-type: none"> – Форма открыта – Таблица документов содержит информацию – Кнопка «Добавить документ» активна 	Выполнено
8. Нажимаем кнопку «Добавить документ»	<ul style="list-style-type: none"> – Кнопка неактивна – Открыта форма добавления документа – Поля пустые 	Выполнено
9. Заполняем поля	<ul style="list-style-type: none"> – Поля заполнены – Кнопка «Выбрать файл» активна 	Выполнено
10. Нажимаем кнопку «Добавить документ»	<ul style="list-style-type: none"> – Сообщение об ошибке «файл не выбран» 	Выполнено

ЗАКЛЮЧЕНИЕ

В ходе производственной практики была успешно разработана система управления доступом к документам для АО «ЦНИИАГ», которая решает ключевые задачи централизованного контроля и защиты конфиденциальной информации. В ходе реализации проекта были выполнены следующие задачи:

1. Анализ требований и существующих процессов
Были изучены текущие механизмы управления доступом к документам, выявлены недостатки ручного управления правами и определены функциональные требования к новой системе. Это позволило создать решение, полностью соответствующее специфике работы предприятия с документами различного уровня секретности.

2. Проектирование структуры приложения и базы данных
Разработана логическая модель данных, включающая таблицы пользователей, документов, ролей и прав доступа. Использование реляционной СУБД PostgreSQL обеспечило надежное хранение информации, целостность данных и высокую производительность при работе с большими объемами документов.

3. Разработка пользовательского интерфейса (GUI)
Создан интуитивно понятный интерфейс на базе PyQt5, предоставляющий пользователям удобные инструменты работы.

4. Интеграция с системой хранения документов
Реализована безопасная связь между интерфейсом приложения и физическим хранилищем документов. Система поддерживает работу как с базой данных (метаданные документов), так и с файловым сервером (непосредственное хранение документов), обеспечивая согласованность данных и применение изменений прав в реальном времени.

5. Тестирование функциональности.

Итоговый результат – это надежная, масштабируемая система, которая:

- автоматизирует процессы управления доступом, сокращая время на рутинные операции;
- обеспечивает строгое соблюдение политик информационной безопасности;
- позволяет гибко управлять правами пользователей в соответствии с их должностными обязанностями;
- обеспечивает прозрачность и контроль всех действий с документами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Учебная и научная литература

- 1) Алан Болье «Изучаем SQL» – Диалектика, 2021 г.
- 2) Уолтер Шилдс «SQL: быстрое погружение» – Питер, 2022 г.
- 3) Энтони Молинаро, Роберт де Грааф «SQL. Сборник рецептов» – БХВ- Петербург, 2021 г.

Интернет-источники

- 1) Форум программистов Stack Over Flow – URL:
<https://ru.stackoverflow.com/>
- 2) Форум программистов и сисадминов Киберфорум –
URL: <https://www.cyberforum.ru/>
- 3) Статья «Нормализация отношений. Шесть нормальных форм» –
URL: <https://habr.com/ru/post/254773/>
- 4) Форум программистов Habr – URL: <https://habr.com/ru/all/>

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программы

Код программы

Файл main.py

```
#Программа: информационная система для управления доступом к документам предприятия
#Среда разработки: PyCharmCE 2022
#Производственная практика ПП.02 Осуществление интеграции программных модулей
#Язык программирования: Python
#Разработала: Пичугина Софья Павловна ТБД-61
#Дата 05.06.2025-18.06.2025
import sys
from PyQt5.QtCore import QFile, QIODevice, QTextStream
from PyQt5.QtWidgets import QApplication, QLineEdit, QMessageBox, QFileDialog,
QTableWidgetItem, QDialog, QVBoxLayout, \
    QTextBrowser, QPushButton
from sqlalchemy import or_, extract, update
from gui.windows import AutorizationWindow, AdminWindow, ManagerWindow, ViewerWindow,
DocumentForm, DocumentViewer
from session import session
import bcrypt
from orm import UsersORM, DocumentsORM
import random
import string
import datetime
import os

class Control():
    def __init__(self, w):
        self.AutorizationWin = w
        self.AdmWin = AdminWindow()
        self.ManagerWin = ManagerWindow()
        self.ViewerWin = ViewerWindow()
        self.DocsWin = DocumentForm()
        self.DocsViewer = DocumentViewer()
        self.AutorizationWin.lineEdit_2.setEchoMode(QLineEdit.Password)
        self.AutorizationWin.pushButton.clicked.connect(self.login)
        self.AdmWin.pushButton_10.clicked.connect(self.add_user)
        self.AdmWin.pushButton_8.clicked.connect(self.delete_user)
        self.AdmWin.pushButton_2.clicked.connect(self.show_add_docs)
        self.DocsWin.browse_button.clicked.connect(self.browse_file)
        self.DocsWin.add_button.clicked.connect(self.add_document)
        self.AdmWin.pushButton_5.clicked.connect(self.delete_document)
        self.AdmWin.pushButton.clicked.connect(self.search_documents)
        self.AdmWin.pushButton_6.clicked.connect(self.issue_to_manager)
        self.AdmWin.pushButton_7.clicked.connect(self.return_document)
        self.AdmWin.tableWidget_2.cellDoubleClicked.connect(self.admin_open_file)
        self.ManagerWin.pushButton.clicked.connect(self.search_documents)
        self.ManagerWin.tableWidget.cellDoubleClicked.connect(self.manager_open_file)
```

```

self.ViewerWin.tableWidget.cellDoubleClicked.connect(self.viewer_open_file)
self.ManagerWin.pushButton_3.clicked.connect(self.issue_to_viewer)
self.ManagerWin.pushButton_4.clicked.connect(self.return_from_viewer)

def login(self):
    email = self.AutorizationWin.lineEdit.text() # Поле для email
    password = self.AutorizationWin.lineEdit_2.text() # Поле для пароля
    user = session.query(UsersORM).filter_by(email=email).first()
    if user and self.check_password(user.hash_password, password):
        self.current_user_id = user.id
        self.current_user_role = user.role_id
        print(user.id)
        if user.role_id == 1: #роль 1 = админ
            self.AdmWin.show()
            self.show_users()
            self.show_documents()

        elif user.role_id == 2: # Менеджер
            self.ManagerWin.show()
            self.show_documents_manager()
        elif user.role_id == 3:
            self.ViewerWin.show()
            self.show_documents_viwer()
        self.AutorizationWin.close()
    else:
        self.AutorizationWin.label_error.setText("Неверный email или пароль")

def add_user(self):
    try:
        password = self.generate_password()
        new_user = UsersORM(name=self.AdmWin.lineEdit_2.text(),
                             surname=self.AdmWin.lineEdit_3.text(),
                             patronymic=self.AdmWin.lineEdit_4.text(),
                             email=self.AdmWin.lineEdit_5.text(),
                             role_id=int((self.AdmWin.comboBox.currentText())[0]),
                             hash_password=self.hash_password(password))

        session.add(new_user)
        session.commit()
        self.show_users()
    except Exception as e:
        session.rollback()
        QMessageBox.critical(None, "Ошибка", f"Не удалось добавить пользователя: {e}")

def find_user_in_table(self):
    selected_row = self.AdmWin.tableWidget.currentRow()
    user_id = int(self.AdmWin.tableWidget.item(selected_row, 0).text())
    return user_id

```

```

def delete_user(self):
    user_id = self.find_user_in_table()
    confirm = QMessageBox.question(
        None,
        "Подтверждение",
        f"Вы уверены, что хотите удалить пользователя с ID {user_id}?",
        QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No,
    )
    if confirm == QMessageBox.StandardButton.Yes:
        user = session.get(UsersORM, user_id)
        session.delete(user)
        session.commit()
        self.show_users()

def show_users(self):
    users = (session.query(UsersORM.id, UsersORM.name, UsersORM.surname,
UsersORM.patronymic, UsersORM.email, UsersORM.role_id).all())
    self.AdmWin.tableWidget.setRowCount(len(users))
    row = 0
    for user in users:
        col = 0
        for item in user:
            self.cellinfo = QTableWidgetItem(str(item))
            self.AdmWin.tableWidget.setItem(row, col, self.cellinfo)
            col += 1
        row += 1
    self.AdmWin.tableWidget.resizeColumnsToContents()

def generate_password(self):
    letters = (random.choices(string.ascii_letters, k=random.randint(7, 8)))
    digits = (random.choices(string.digits, k=random.randint(2, 3)))
    # Объединяем буквы и цифры, перемешиваем
    password = list(letters + digits)
    random.shuffle(password)
    password = str("".join(password))
    msgBox = QMessageBox()
    msgBox.setWindowTitle("registration succeed!")
    msgBox.setText(f"сгенерирован пароль: {password}")
    msgBox.exec_()
    return password

def check_password(self, hash_password: str, password):
    return bcrypt.checkpw(
        password.encode('utf-8'),
        hash_password.encode('utf-8'))

def hash_password(self, password: str) -> str:
    """Хэширует пароль с солью (salt) и возвращает строку в формате bcrypt."""
    salt = bcrypt.gensalt() # Генерация соли

```

```

hashed = bcrypt.hashpw(password.encode('utf-8'), salt)
return hashed.decode('utf-8') # Преобразуем bytes в строку

def show_add_docs(self):
    self.DocsWin.show()

def browse_file(self):
    file_path, _ = QFileDialog.getOpenFileName(
        None, 'Выберите файл документа', "",
        'Все файлы (*);;Документы (*.docx *.xlsx *.pdf *.txt)'
    )
    if file_path:
        self.DocsWin.file_input.setText(file_path)

def add_document(self):
    # Получаем данные из формы
    title = self.DocsWin.title_input.text().strip()
    category = self.DocsWin.category_input.currentText().strip()
    security_level = self.DocsWin.security_input.value()
    file_path = self.DocsWin.file_input.text().strip()
    print(title, category, security_level, file_path, self.current_user_id)
    # Валидация данных
    if not title:
        QMessageBox.warning(None, 'Ошибка', 'Введите название документа')
        return
    if not file_path:
        QMessageBox.warning(None, 'Ошибка', 'Выберите файл документа')
        return
    try:
        new_document = DocumentsORM(title=title,
                                     category=category,
                                     security_level=security_level,
                                     file_path=file_path,
                                     cration_date= datetime.datetime.now(),
                                     creator_id=self.current_user_id)
        session.add(new_document)
        session.commit()
        QMessageBox.information(None, 'Успех', 'Документ успешно добавлен')
    except Exception as db_error:
        session.rollback()
        QMessageBox.critical(self.DocsWin, 'Ошибка БД', f'Ошибка при сохранении:
{str(db_error)}')
        self.show_documents()
        self.DocsWin.hide()

def find_document_in_admin_table(self):
    selected_row = self.AdmWin.tableWidget_2.currentRow()
    document_id = int(self.AdmWin.tableWidget_2.item(selected_row, 0).text())
    return document_id

```

```

def find_document_in_manager_table(self):
    selected_row = self.ManagerWin.tableWidget.currentRow()
    document_id = int(self.ManagerWin.tableWidget.item(selected_row, 0).text())
    return document_id

def find_document_in_viewer_table(self):
    selected_row = self.ViewerWin.tableWidget.currentRow()
    document_id = int(self.ViewerWin.tableWidget.item(selected_row, 0).text())
    print(document_id)
    return document_id

def delete_document(self):
    document_id = self.find_document_in_admin_table()
    confirm = QMessageBox.question(
        None,
        "Подтверждение",
        f"Вы уверены, что хотите удалить документ с ID {document_id}?",
        QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No,
    )
    if confirm == QMessageBox.StandardButton.Yes:
        document = session.get(DocumentsORM, document_id)
        session.delete(document)
        session.commit()
        self.show_documents()

def show_documents(self):
    documents = (session.query(DocumentsORM.id, DocumentsORM.title,
DocumentsORM.category, DocumentsORM.cration_date, DocumentsORM.security_level,
DocumentsORM.file_path).all())
    self.AdmWin.tableWidget_2.setRowCount(len(documents))
    row = 0
    for document in documents:
        col = 0
        for item in document:
            cellinfo = QTableWidgetItem(str(item))
            self.AdmWin.tableWidget_2.setItem(row, col, cellinfo)
            col += 1
        row += 1
    self.AdmWin.tableWidget_2.resizeColumnsToContents()

def show_documents_manager(self):
    documents = (session.query(DocumentsORM.id, DocumentsORM.title,
DocumentsORM.category, DocumentsORM.cration_date, DocumentsORM.security_level,
DocumentsORM.file_path).filter(DocumentsORM.security_level <= 2).all())
    self.ManagerWin.tableWidget.setRowCount(len(documents))
    row = 0
    for document in documents:
        col = 0
        for item in document:
            cellinfo = QTableWidgetItem(str(item))

```

```

        self.ManagerWin.tableWidget.setItem(row, col, cellinfo)
        col += 1
    row += 1
    self.ManagerWin.tableWidget.resizeColumnsToContents()

def show_documents_viwer(self):
    documents = (session.query(DocumentsORM.id, DocumentsORM.title,
DocumentsORM.category, DocumentsORM.cration_date,
DocumentsORM.security_level).filter(DocumentsORM.security_level == 1).all())
    self.ViewerWin.tableWidget.setRowCount(len(documents))
    print(documents)
    row = 0
    for document in documents:
        col = 0
        for item in document:
            cellinfo = QTableWidgetItem(str(item))
            self.ViewerWin.tableWidget.setItem(row, col, cellinfo)
            col += 1
        row += 1
    self.ViewerWin.tableWidget.resizeColumnsToContents()

def search_documents(self):
    if self.current_user_role == 1:
        search_term = self.AdmWin.lineEdit.text()
        self.AdmWin.tableWidget_2.setRowCount(0)
    elif self.current_user_role == 2:
        search_term = self.ManagerWin.lineEdit.text()
        self.ManagerWin.tableWidget.setRowCount(0)
    else:
        search_term = self.ViewerWin.lineEdit.text()
        self.ViewerWin.tableWidget.setRowCount(0)
    try:
        query = session.query(DocumentsORM)

        # Проверяем, является ли поисковый запрос годом (4 цифры)
        year_search = None
        if search_term.isdigit() and len(search_term) == 4:
            try:
                year_search = int(search_term)
                if 1900 <= year_search <= datetime.datetime.now().year:
                    # Фильтр по году из даты создания
                    query = query.filter(
                        extract('year', DocumentsORM.cration_date) == year_search
                    )
            except ValueError:
                pass

        if not year_search:
            # Поиск по названию или категории (регистронезависимый)
            search_pattern = f"%{search_term}%"

```

```

        query = query.filter(
            or_(
                DocumentsORM.title.ilike(search_pattern),
                DocumentsORM.category.ilike(search_pattern)
            )
        )

    if self.current_user_role == 1:
        documents = query.order_by(DocumentsORM.cration_date.desc()).all()
        if not documents:
            QMessageBox.information(None, "Результаты", "Документы не найдены")
            return
        self.AdmWin.tableWidget_2.setRowCount(len(documents))
        for row, doc in enumerate(documents):
            self.AdmWin.tableWidget_2.setItem(row, 0, QTableWidgetItem(str(doc.id)))
            self.AdmWin.tableWidget_2.setItem(row, 1, QTableWidgetItem(doc.title))
            self.AdmWin.tableWidget_2.setItem(row, 2, QTableWidgetItem(doc.category))
            self.AdmWin.tableWidget_2.setItem(row, 3,
                QTableWidgetItem(doc.cration_date.strftime("%d.%m.%Y")))
            self.AdmWin.tableWidget_2.setItem(row, 4,
                QTableWidgetItem(str(doc.security_level)))
            self.AdmWin.tableWidget_2.setItem(row, 5, QTableWidgetItem(doc.file_path))
            # Настраиваем отображение таблицы
            self.AdmWin.tableWidget_2.resizeColumnsToContents()

    if self.current_user_role == 2:
        documents =
        query.order_by(DocumentsORM.cration_date.desc()).filter(DocumentsORM.security_level <=
            2).all()
        if not documents:
            QMessageBox.information(None, "Результаты", "Документы не найдены")
            return
        self.ManagerWin.tableWidget.setRowCount(len(documents))
        for row, doc in enumerate(documents):
            self.ManagerWin.tableWidget.setItem(row, 0, QTableWidgetItem(str(doc.id)))
            self.ManagerWin.tableWidget.setItem(row, 1, QTableWidgetItem(doc.title))
            self.ManagerWin.tableWidget.setItem(row, 2, QTableWidgetItem(doc.category))
            self.ManagerWin.tableWidget.setItem(row, 3,
                QTableWidgetItem(doc.cration_date.strftime("%d.%m.%Y")))
            self.ManagerWin.tableWidget.setItem(row, 4,
                QTableWidgetItem(str(doc.security_level)))
            self.ManagerWin.tableWidget.setItem(row, 5, QTableWidgetItem(doc.file_path))
            # Оптимизация отображения
            self.ManagerWin.tableWidget_2.resizeColumnsToContents()
        else:
            documents =
            query.order_by(DocumentsORM.cration_date.desc()).filter(DocumentsORM.security_level ==
                1).all()
            if not documents:
                QMessageBox.information(None, "Результаты", "Документы не найдены")

```



```

self.ViewerWin.tableWidget.setRowCount(len(documents)-1)
for row, doc in enumerate(documents):
    self.ViewerWin.tableWidget.setItem(row, 0, QTableWidgetItem(str(doc.id)))
    self.ViewerWin.tableWidget.setItem(row, 1, QTableWidgetItem(doc.title))
    self.ViewerWin.tableWidget.setItem(row, 2, QTableWidgetItem(doc.category))
    self.ViewerWin.tableWidget.setItem(row, 3,
QTableWidgetItem(doc.cration_date.strftime("%d.%m.%Y")))
    self.ViewerWin.tableWidget.setItem(row, 4,
QTableWidgetItem(str(doc.security_level)))
    if not documents:
        QMessageBox.information(None, "Результаты", "Документы не найдены")
    return

except Exception as e:
    print(f"Ошибка при поиске документов: {str(e)}")
    return []

def issue_to_manager(self):
    try:
        # Получаем текущий документ
        document_id = self.find_document_in_admin_table()
        document = session.get(DocumentsORM, document_id)
        if not document:
            QMessageBox.warning(None, "Ошибка", "Документ не найден")
            return False
        # Проверяем текущий уровень защиты
        if document.security_level == 3:
            # Обновляем уровень защиты
            session.execute(
                update(DocumentsORM)
                .where(DocumentsORM.id == document_id)
                .values(security_level=2)
            )
            session.commit()
            QMessageBox.information(None, "Успех",
                "Документ выдан менеджеру (уровень защиты изменен с 3 на 2)")
            self.show_documents()
            return True
        else:
            QMessageBox.information(None, "Информация",
                f"Документ уже имеет уровень защиты {document.security_level} -
изменения не требуются")
            return False
    except Exception as e:
        session.rollback()
        QMessageBox.critical(None, "Ошибка",
            f"Не удалось выдать документ менеджеру:\n{str(e)}")
        return False

```

```

def return_document(self):
    document_id = self.find_document_in_admin_table()
    try:
        # Получаем документ
        document = session.get(DocumentsORM, document_id)
        if not document:
            QMessageBox.warning(None, "Ошибка", "Документ не найден")
            return False
        # Проверяем текущий уровень защиты
        if document.security_level in (1, 2):
            old_level = document.security_level
            # Обновляем уровень защиты на 3
            stmt = (
                update(DocumentsORM)
                .where(DocumentsORM.id == document_id)
                .values(security_level=3)
            )
            session.execute(stmt)
            session.commit()
            QMessageBox.information(None, "Успех", "Документ возвращен (уровень защиты изменен)")
            self.show_documents()
            return True
        else:
            QMessageBox.information(None, "Информация", "Документ уже имеет уровень защиты 3 - изменения не требуются")
            return False
    except Exception as e:
        session.rollback()
        QMessageBox.critical(None, "Ошибка", f"Не удалось вернуть документ:\n{str(e)}")
        return False

def issue_to_viewer(self):
    """
    Менеджер выдает документ вьюверу (устанавливает уровень защиты 1)
    """
    try:
        # Получаем ID выбранного документа
        document_id = self.find_document_in_manager_table()
        if not document_id:
            QMessageBox.warning(None, "Ошибка", "Документ не выбран")
            return False
        # Получаем документ из БД
        document = session.get(DocumentsORM, document_id)
        if not document:
            QMessageBox.warning(None, "Ошибка", "Документ не найден в базе данных")
            return False
        # Проверяем текущий уровень защиты
        if document.security_level == 2:
            # Обновляем уровень защиты на 1

```

```

stmt = (
    update(DocumentsORM)
    .where(DocumentsORM.id == document_id)
    .values(security_level=1)
)
session.execute(stmt)
session.commit()

        QMessageBox.information(None, "Успех", "Документ выдан вьюверу (уровень
защиты изменен с 2 на 1)")
        self.show_documents_manager() # Обновляем таблицу документов
        return True

    elif document.security_level == 1:
        QMessageBox.information(None, "Информация", "Документ уже выдан вьюверу
(уровень защиты 1)")
        return False
    else:
        QMessageBox.warning(None, "Ошибка", "Нельзя выдать документ с уровнем защиты
3 вьюверу")
        return False
    except Exception as e:
        session.rollback()
        QMessageBox.critical(None, "Ошибка", f"Не удалось выдать документ вьюверу:\n{str(e)}")
        return False

def return_from_viewer(self):
    """
    Менеджер забирает документ у вьювера (устанавливает уровень защиты 2)
    """
    try:
        # Получаем ID выбранного документа
        document_id = self.find_document_in_manager_table()
        if not document_id:
            QMessageBox.warning(None, "Ошибка", "Документ не выбран")
            return False
        # Получаем документ из БД
        document = session.get(DocumentsORM, document_id)
        if not document:
            QMessageBox.warning(None, "Ошибка", "Документ не найден в базе данных")
            return False
        # Проверяем текущий уровень защиты
        if document.security_level == 1:
            # Обновляем уровень защиты на 2
            stmt = (
                update(DocumentsORM)
                .where(DocumentsORM.id == document_id)
                .values(security_level=2)
            )

```

```

        session.execute(stmt)
        session.commit()

        QMessageBox.information(
            None, "Успех", "Документ возвращен (уровень защиты изменен с 1 на 2)")
        self.show_documents_manager() # Обновляем таблицу документов
        return True

    elif document.security_level == 2:
        QMessageBox.information(None, "Информация", "Документ уже имеет уровень
защиты 2")
        return False
    else:
        return False

except Exception as e:
    session.rollback()
    QMessageBox.critical(None, "Ошибка", f"Не удалось вернуть документ:\n{str(e)}")
    return False

def admin_open_file(self):
    file = session.get(DocumentsORM, self.find_document_in_admin_table())
    file_path = file.file_path
    try:
        os.startfile(file_path)
    except Exception as e:
        QMessageBox.critical(None, "Ошибка", f"Не удалось открыть файл: {str(e)}")

def manager_open_file(self):
    file = session.get(DocumentsORM, self.find_document_in_manager_table())
    file_path = file.file_path
    try:
        os.startfile(file_path)
    except Exception as e:
        QMessageBox.critical(None, "Ошибка", f"Не удалось открыть файл: {str(e)}")

def viewer_open_file(self):
    document_id = self.find_document_in_viewer_table()
    if self.DocsViewer.load_document(document_id):
        self.DocsViewer.show()
    else:
        QMessageBox.warning(None, "Ошибка", "Не удалось загрузить документ")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    w = AutorizationWindow()
    w.show()
    c = Control(w)
    sys.exit(app.exec_())

```

Файл orm.py

```
import datetime

from sqlalchemy import create_engine, Column, Integer, VARCHAR, TIMESTAMP,
ForeignKey, func
from sqlalchemy.orm import declarative_base, mapped_column, Mapped

engine = create_engine('postgresql+psycopg2://postgres:password@localhost:5432/
docsystem')
engine.connect()
print(engine)

Base = declarative_base()

class RolesORM(Base):
    __tablename__ = 'roles'
    id: Mapped[int] = mapped_column(autoincrement=True, primary_key=True)
    name: Mapped[str] = mapped_column(unique=True)

class UsersORM(Base):
    __tablename__ = "users"
    id: Mapped[int] = mapped_column(autoincrement=True, primary_key=True)
    name: Mapped[str]
    surname: Mapped[str]
    patronymic: Mapped[str | None]
    email: Mapped[str] = mapped_column(unique=True)
    hash_password: Mapped[str]
    role_id: Mapped[int] = mapped_column(ForeignKey("roles.id", ondelete="CASCADE"))

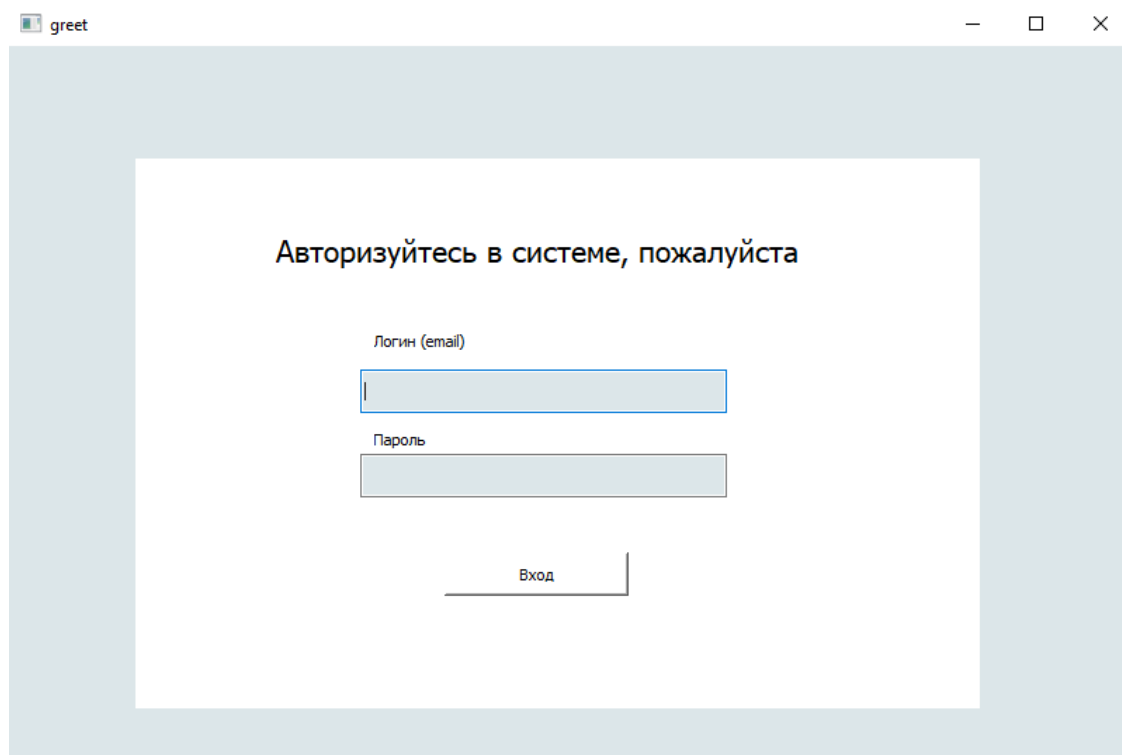
class DocumentsORM(Base):
    __tablename__ = "documents"
    id: Mapped[int] = mapped_column(autoincrement=True, primary_key=True)
    title: Mapped[str]
    category: Mapped[str]
    cration_date: Mapped[datetime.datetime]
    security_level: Mapped[int]
    file_path: Mapped[str]
    creator_id: Mapped[int] = mapped_column(ForeignKey("users.id"))
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Результат выполнения программы

На рисунке Б.1 представлена форма авторизации.



greet

Авторизуйтесь в системе, пожалуйста

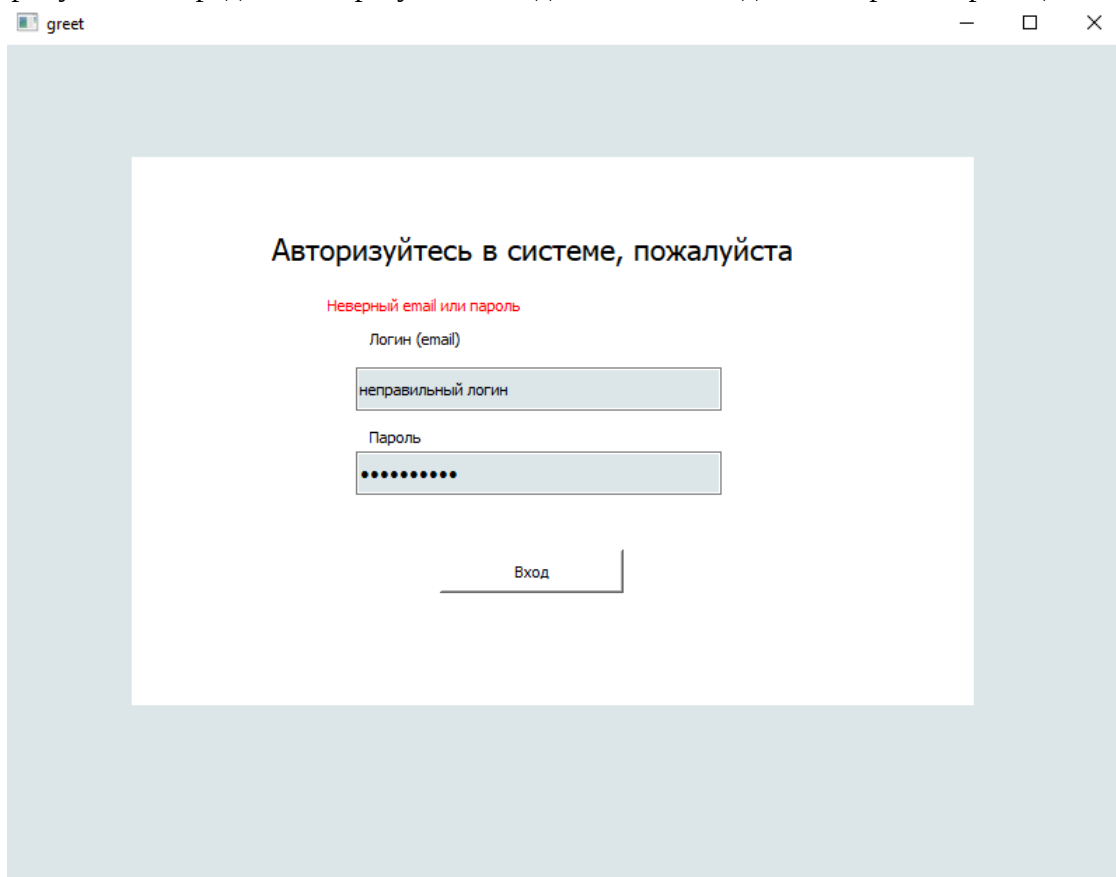
Логин (email)

Пароль

Вход

Рисунок Б.1 — Форма авторизации

На рисунке Б.2 представлен результат ввода ошибочных данных при авторизации.



greet

Авторизуйтесь в системе, пожалуйста

Неверный email или пароль

Логин (email)

неправильный логин

Пароль

Вход

Рисунок Б.2 — Проверка данных при авторизации

На рисунке Б.3 представлено главное окно для пользователя роли viewer

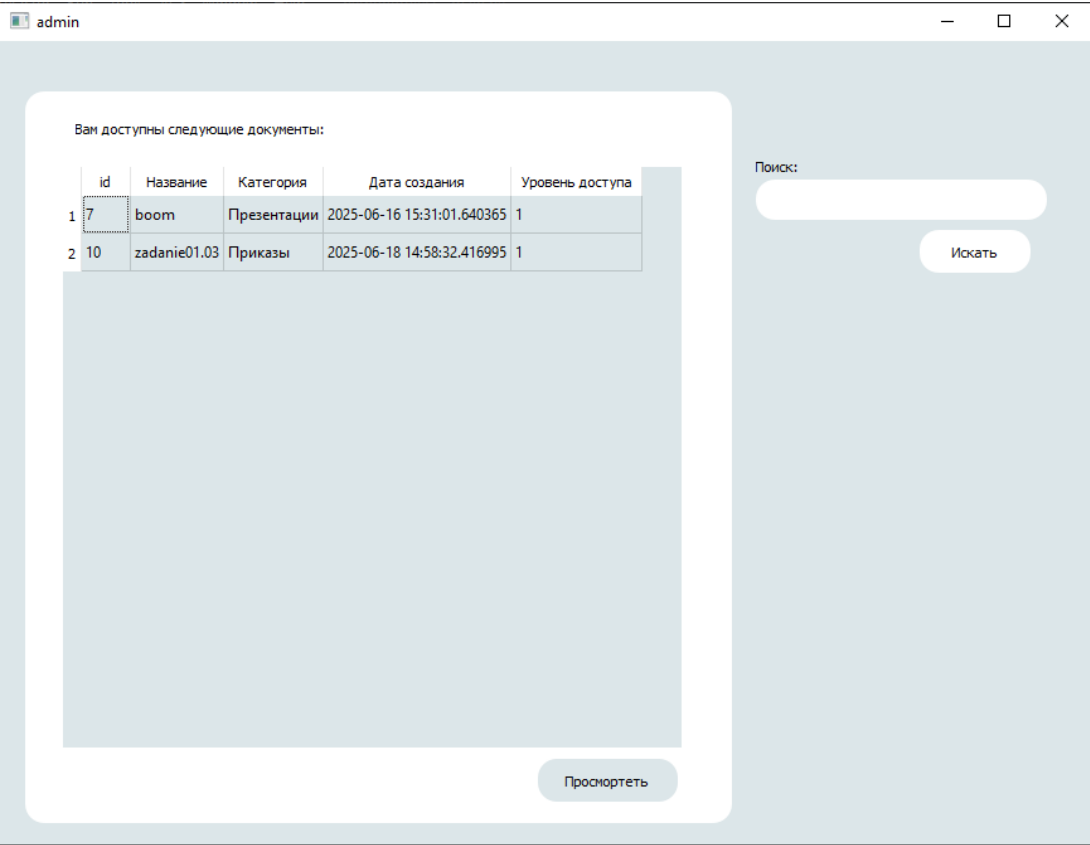


Рисунок Б.3 — главное окно пользователя

На рисунке Б.4 представлена форма для просмотра документа для Viewer

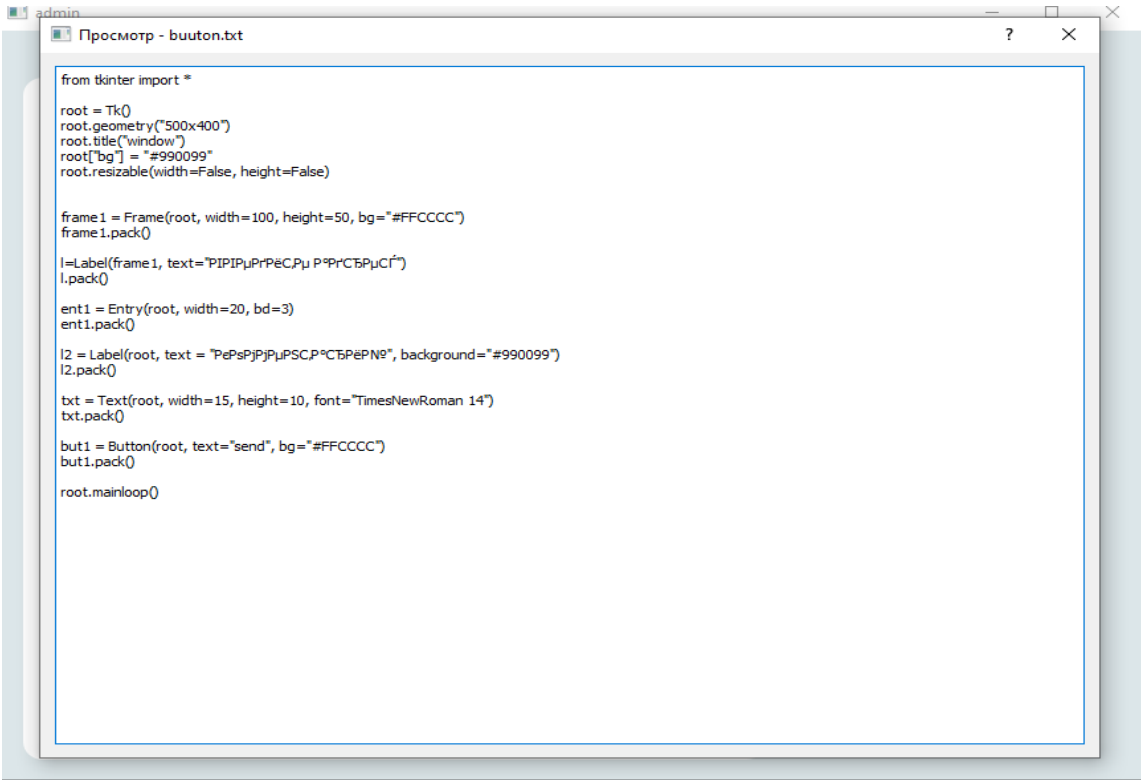


Рисунок Б.4 — форма просмотра документа

На рисунке Б.5 представлена панель администратора, вкладка «Документы»

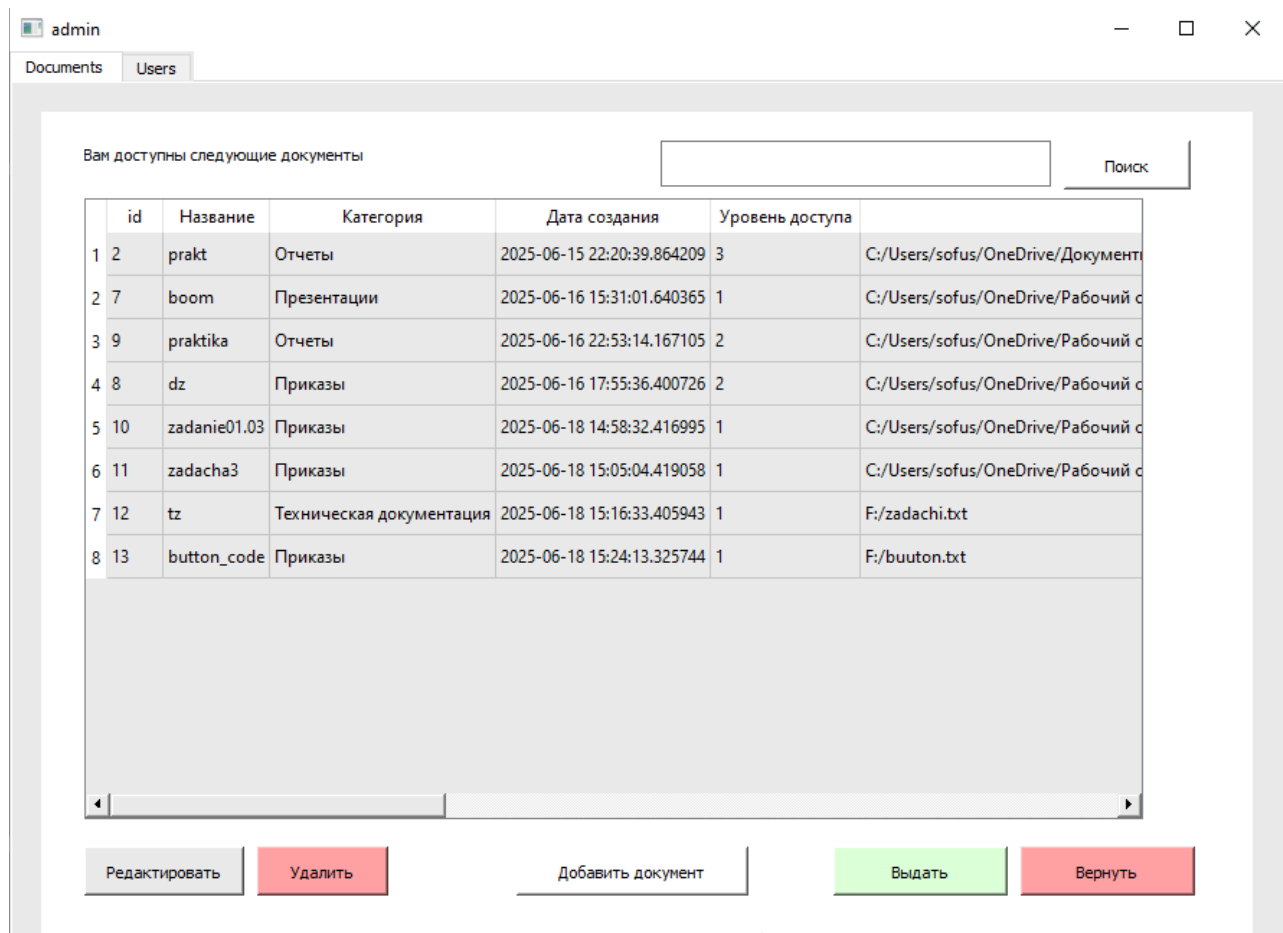


Рисунок Б.5 — админ панель

На рисунке Б.6 представлена форма добавления документа, выводющаяся при нажатии кнопки «Добавить документ».

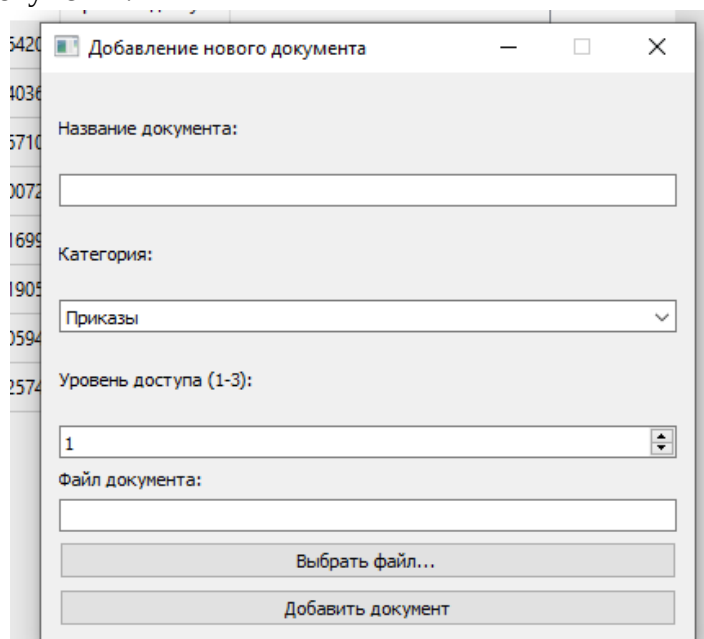


Рисунок Б.6 — Добавление документа

На рисунках Б.7-Б.9 представлен результат поиска документа по названию, категории и году.

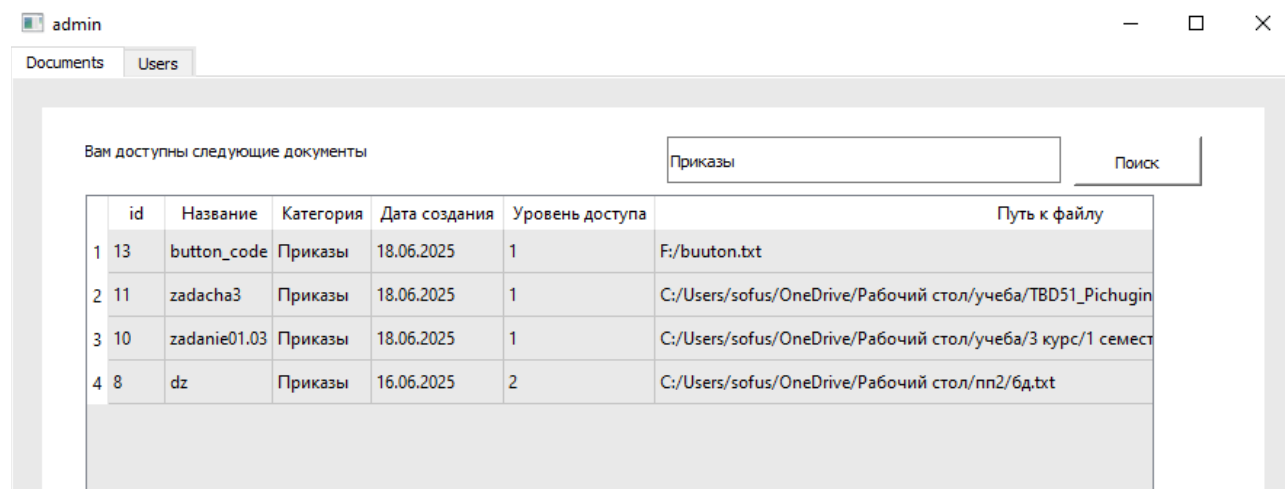


Рисунок Б.7 — Поиск документа по категории

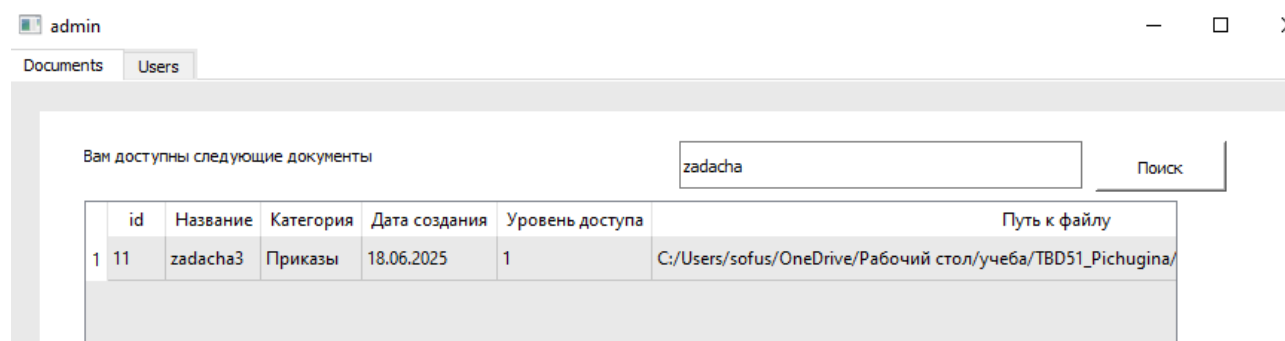


Рисунок Б.8 — Поиск документа по названию

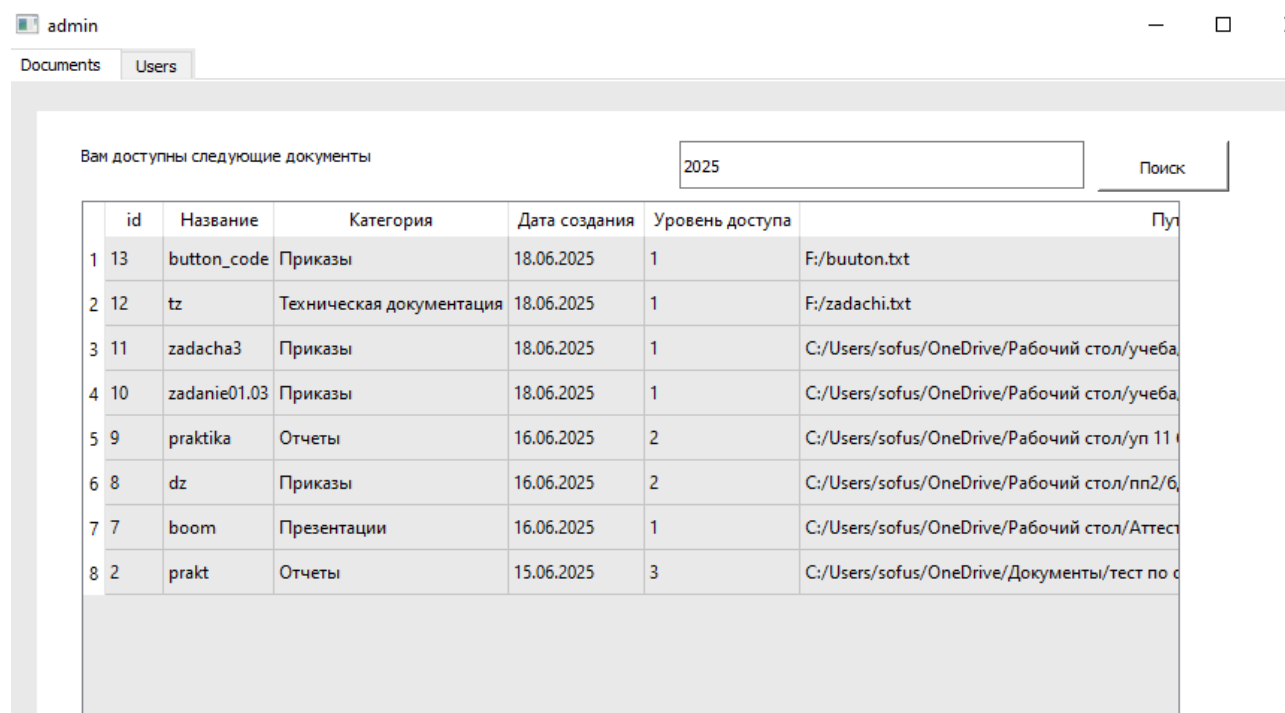


Рисунок Б.9 — Поиск документа по дате

На рисунке Б.10 представлена панель администратора, вкладка «Пользователи».

The screenshot shows a web application window titled 'admin'. It has two tabs: 'Documents' and 'Users', with 'Users' being the active tab. The main content area is divided into two sections. On the left, titled 'Список пользователей' (List of users), there is a table with 7 rows and 6 columns: 'id', 'Имя' (Name), 'Фамилия' (Surname), 'Отчество' (Patronymic), 'Почта' (Email), and 'Уровень доступа' (Access level). The first row is highlighted. Below the table are two buttons: 'Удалить' (Delete) and 'Редактировать' (Edit). On the right, titled 'Добавление пользователя' (Add user), there is a form with input fields for 'Имя', 'Фамилия', 'Отчество', and 'Email'. Below these is a dropdown menu for 'Роль' (Role) with '1-admin' selected, and a green 'Добавить' (Add) button.

	id	Имя	Фамилия	Отчество	Почта	Уровень доступа
1	1	sofya	pichugina	pavlovna	user@example.com	1
2	7	ivan	ivanov	petrovich	vanek@mail.ru	3
3	11	Erop	Краснов	Иванович	egorchik@mail.ru	3
4	15	viktor	korneplod	telega	less@mail.ru	3
5	16	Анатолий	Мясник	Мяснович	myaso@mail.ru	2
6	19	Соня	Пичугина	Павловна	sonka@mail.ru	3
7	20	Анна	Семенова	Валерьевна	ann@mail.ru	2

Рисунок Б.10 — панель управления пользователями

На рисунке Б.11 представлено добавление пользователя.

This is a close-up of the 'Добавление пользователя' (Add user) form. It contains the following fields and values: 'Имя' (Name) is 'Vadim', 'Фамилия' (Surname) is 'Rybak', 'Отчество' (Patronymic) is 'Rybov', and 'Email' is 'fishing@mail.ru'. The 'Роль' (Role) dropdown menu is open, showing three options: '1-admin', '2-manager' (which is highlighted in blue), and '3-viewer'. A green 'Добавить' (Add) button is located to the right of the dropdown.

Рисунок Б.11 — Добавление пользователя

На рисунке Б.12 представлено сообщение об успешной регистрации и генерация пароля для пользователя.

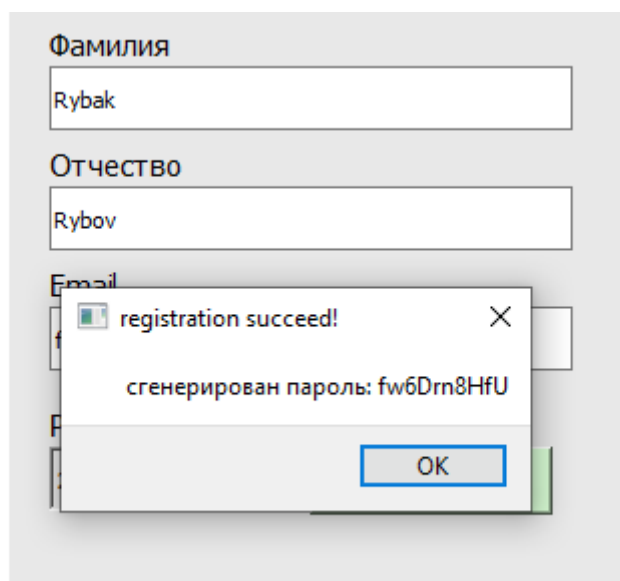


Рисунок Б.12 — Успешная регистрация

На рисунке Б.13 представлено главное окно для пользователя с ролью Manager.

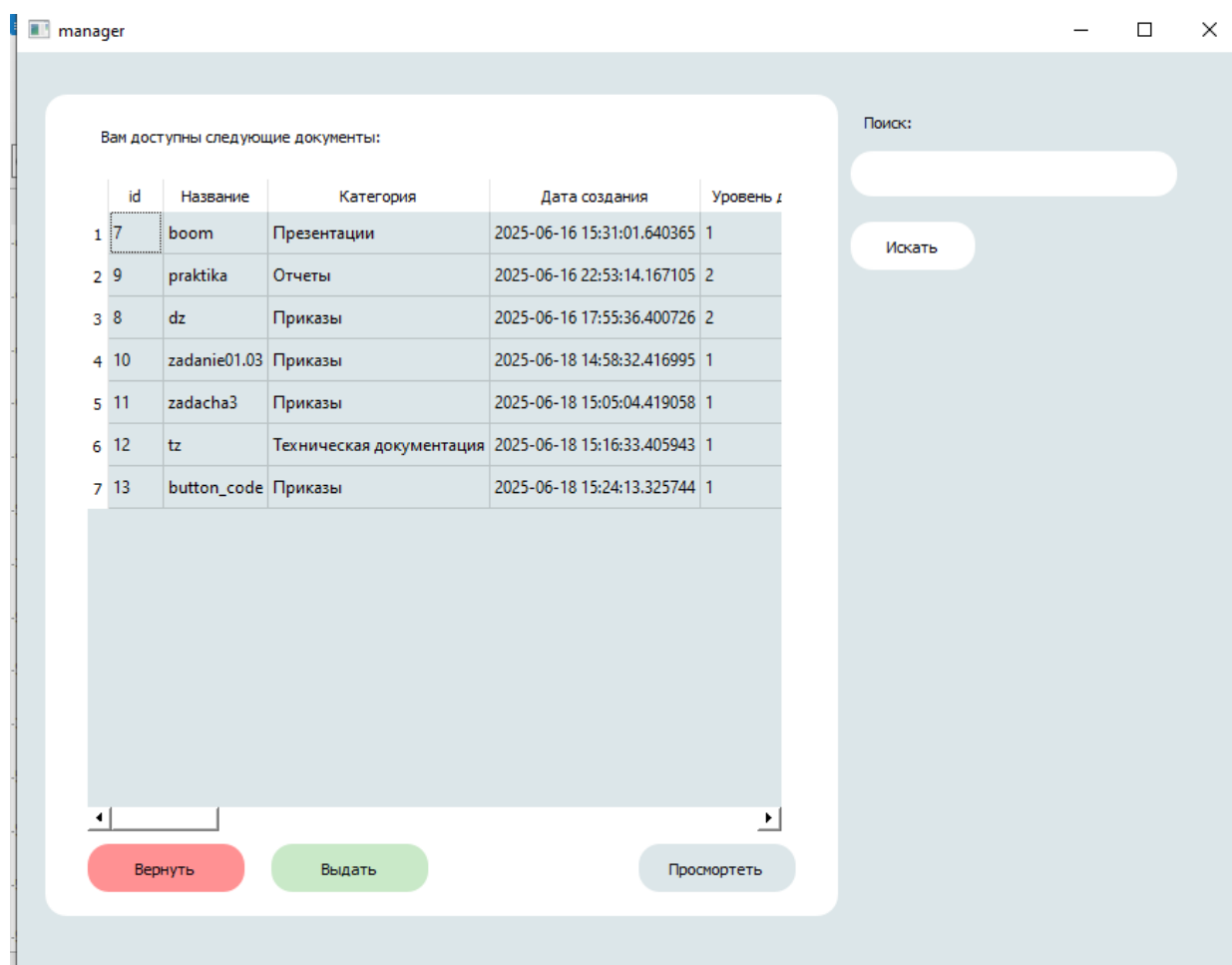


Рисунок Б.13 — Главное окно менеджера

На рисунке Б.14 представлена функция выдачи документа пользователю Viewer.

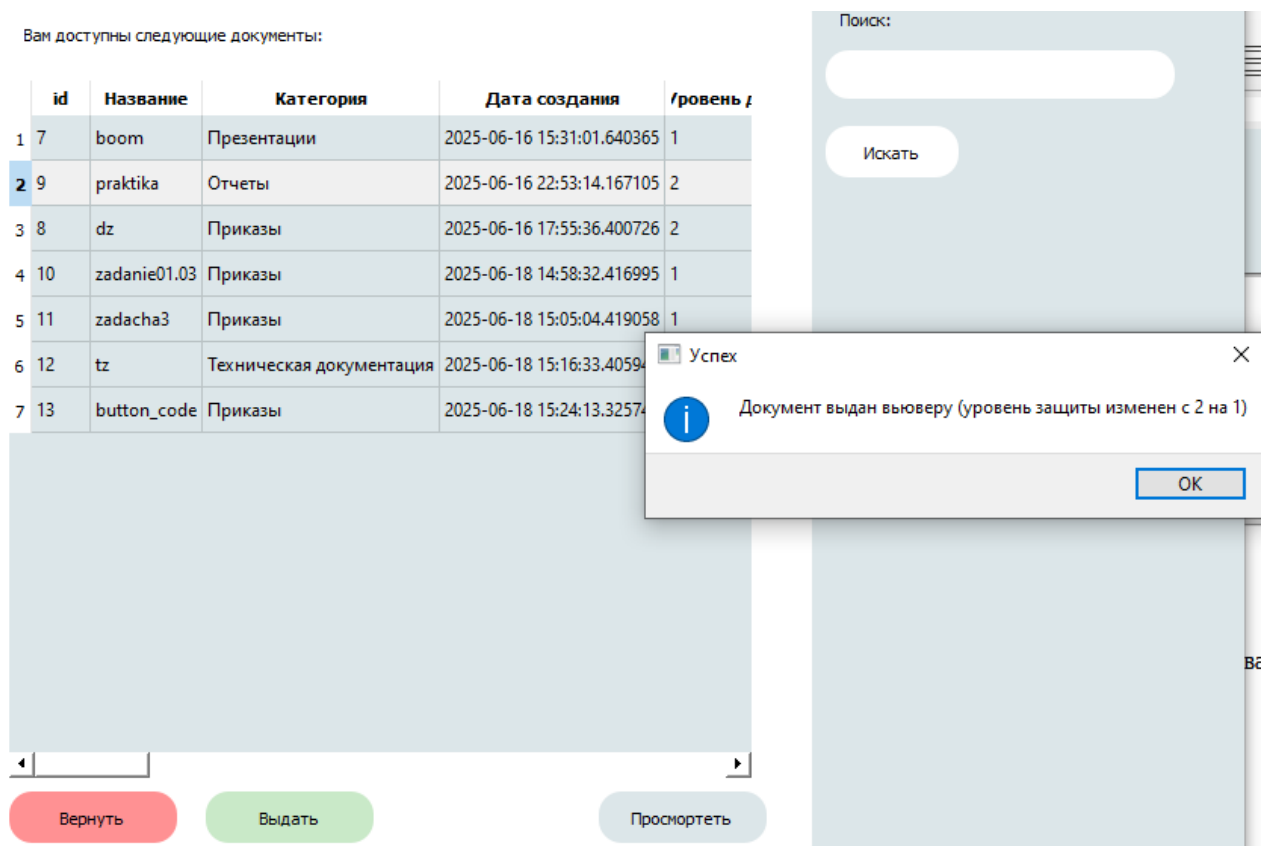


Рисунок Б.14 — Выдача документа

На рисунке Б.15 прествален результат возврата документа.

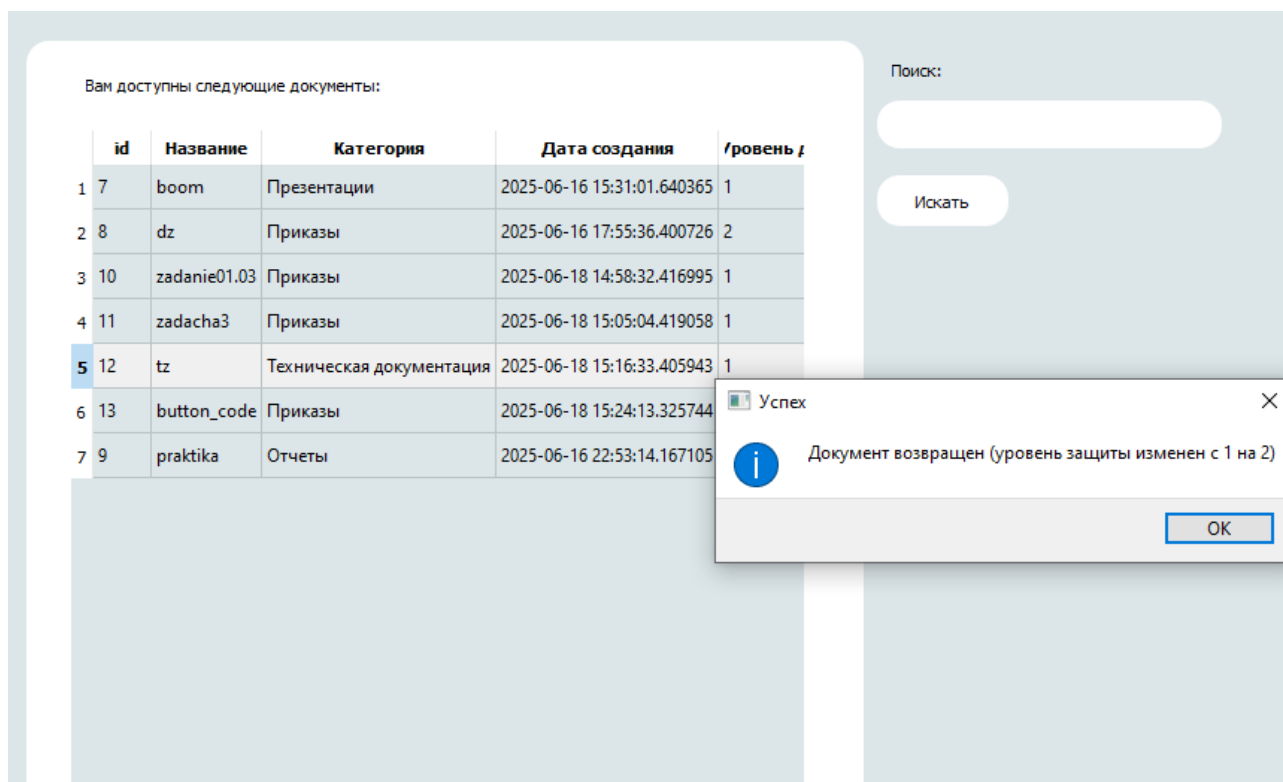


Рисунок Б.15 — Возврат документа

На рисунках Б.16-Б.17 представлены изменения документов в результате выдачи и возврата.

6	9	praktika	Отчеты	2025-06-16 22:53:14.167105	1
---	---	----------	--------	----------------------------	---

Рисунок Б.16 — уровень защиты документа praktika изменен на 1

7	12	tz	Техническая документация	2025-06-18 15:16:33.405943	2
---	----	----	--------------------------	----------------------------	---

Рисунок Б.17 — уровень защиты документа tz изменен на 2