

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ОБЛАЧНОЕ ХРАНИЛИЩЕ С
УПРАВЛЕНИЕМ ДОСТУПОМ
И СИСТЕМОЙ ПОИСКА

Студент 26.05.26 Хоссейни Нежад Сеййед Али Сеййед Мортеза
(подпись, дата)

Руководитель 26.05.26 Юрова Ольга Викторовна
(подпись, дата)

Нормоконтролёр 26.05.26 Бычков Андрей Станиславович
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

02 июня 2026 г. № 6)

Секретарь ГЭК

(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 07 » 04 20 26 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Хоссейни Нежад Сеййед Али Сеййед Мортеза гр. 22ВВП1 факультета вычислительной техники

2. Тема работы: Облачное хранилище с управлением доступом и системой поиска

Тема утверждена приказом ПГУ № 347/0 от "07" апреля 2026 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

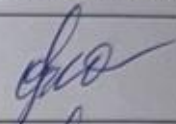
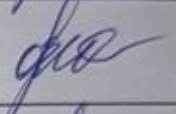
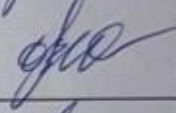
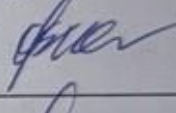
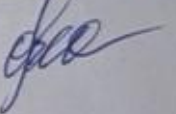
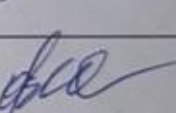
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать веб-приложение «Облачное хранилище с управлением доступом и системой поиска» с функциями управления доступом и системой поиска. Приложение должно позволять пользователям выполнять загрузку, скачивание, перемещение, переименование и удаление собственных файлов, а также предоставлять доступ к ним другим пользователям по почте или публичной ссылке. Приложение должно обеспечивать регистрацию и авторизацию пользователей, редактирование информации о профиле, а также осуществлять поиск файлов по имени, типу и содержанию.

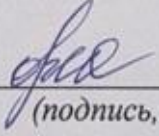
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги облачных хранилищ (Google Диск, Dropbox, Яндекс.Диск), выявить их достоинства и недостатки в части управления доступом и поисковых возможностей. Также необходимо изучить технологии для разработки веб-приложения, такие как Spring Framework (Spring Boot, Spring MVC, Spring Data JPA, Spring Security, Spring AI), PostgreSQL, а также принципы построения REST API и методы реализации полнотекстового поиска.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	07.04.2026- 11.04.2026	
2.	Анализ требований	10	12.04.2026- 13.04.2026	
3.	Выбор технологий, применяемых при проектировании и разработке	50	14.04.2026- 20.04.2026	
4.	Описание алгоритмов	40	21.04.2026- 24.04.2026	
5.	Разработка облачного хранилища с управлением доступом и системой поиска	150	25.04.2026- 18.05.2026	
6.	Оформление пояснительной записки	24	19.05.2026- 21.05.2026	

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы  07.04.26 Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент  07.04.26 Хоссейни Нежад С.А.С.М.
(подпись, дата)

Работу к защите допустить
Декан факультета


28.05.2026

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 203 листов, 47 рисунков, 41 таблицу, 20 источников и 3 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, МИКРОСЕРВИСЫ, СИСТЕМА ПОИСКА, УПРАВЛЕНИЕ ДОСТУПОМ

Цель работы – разработать веб-приложения облачного хранилища с управлением доступом и системой поиска.

Объектами разработки являются:

Серверная часть для выполнения регистрации, аутентификации, поиска и управления правами доступа;

Клиентская часть для взаимодействия с пользователем.

В результате проделанной работы была проанализирована предметная область, выбраны необходимые инструменты и технологии для разработки, спроектирована архитектура, реализовано и протестировано веб-приложение «Облачное хранилище с управлением доступом и системой поиска».

					ПГУ 1.090301.24.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Облачное хранилище с управлением доступом и системой поиска Пояснительная записка	Лит.	Лист	Листов
Разраб.		Хоссейн Нежад		26.05.16				
Провер.		Юрова О.В.		26.05.16			2	108
Н. Контр.		Бычков А. С.		26.05.16		ФВТ, каф. ВТ гр. 22ВВП1		
Утверд.		Митрохин М.А.		26.05.16				

Содержание

Перечень сокращений и обозначений	6
Введение.....	8
1 Анализ предметной области.....	9
1.1 Анализ актуальности темы	9
1.2 Анализ аналогов.....	9
1.2.1 Облачное хранилище «Google Диск»	10
1.2.2 Облачное хранилище «Dropbox».....	12
1.2.3 Облачное хранилище «Яндекс.Диск»	13
1.2.4 Вывод.....	14
1.3 Постановка задачи	14
2 Выбор технологий и средств разработки приложения	16
2.1 Java.....	16
2.2 Spring Framework	17
2.2.1 Spring Boot.....	17
2.2.2 Spring MVC	17
2.2.3 Spring Data JPA.....	18
2.2.4 Spring Security	18
2.2.5 Spring AI.....	18
2.3 PostgreSQL.....	18
2.4 RabbitMQ	19
2.5 JavaScript.....	19
2.6 React.....	19
2.7 IntelliJ IDEA.....	19
2.8 Visual Studio Code.....	20
2.9 Apache Maven.....	20
2.10 Docker.....	20
2.11 Ollama.....	20
2.12 Apache Tomcat.....	21
2.13 Nginx	21
3 Архитектура веб-приложения	22

3.1	Общая архитектура системы.....	22
3.2	Слоистая архитектура серверных сервисов	24
3.3	Основной сервис (backend).....	25
3.4	Сервис поиска (search-service)	26
3.5	Сервис чата с документами (chat-service).....	27
3.6	Сервис отправки электронной почты (email-service).....	28
3.7	Сервис переранжирования (reranker-service)	29
3.8	Взаимодействие компонентов.....	29
4	Реализация приложения	31
4.1	Описание работы приложения.....	31
4.2	Структура базы данных	35
4.3	Функционал приложения.....	43
4.3.1	Функционал неавторизованного пользователя	43
4.3.2	Функционал авторизованного пользователя	45
4.4	Описание основных компонентов.....	47
4.4.1	Основной сервис (backend).....	47
4.4.2	Реализация управления доступом	51
4.4.3	Реализация системы поиска	55
4.4.5	Сервис чата с документами (chat-service)	61
4.4.7	Сервис переранжирования (reranker-service).....	65
4.4.8	Клиентское приложение (frontend).....	67
4.5	Основные алгоритмы взаимодействия клиента и сервера	70
4.5.1	Алгоритм аутентификации и авторизации запросов.....	70
4.5.2	Алгоритм загрузки и индексации файла	71
4.5.3	Алгоритм гибридного поиска файлов	72
4.5.4	Алгоритм AI-чата.....	73
4.5.5	Алгоритм управления правами доступа.....	73
4.5.6	Алгоритм создания и использования публичных ссылок	74
4.5.7	Алгоритм работы корзины	75
5	Тестирование приложения	76
6	Безопасность жизнедеятельности.....	103
6.1	Требования к помещению и рабочему месту.....	103

6.2 Требования к освещению.....	104
6.3 Требования к режиму труда и отдыха.....	104
6.4 Рекомендации по использованию программного обеспечения	105
Заключение.....	106
Список использованных источников	107
Приложение А – UML-диаграммы.....	109
Приложение Б – Листинг программы	117
Приложение В – Презентация.....	194

Перечень сокращений и обозначений

В настоящей работе используются следующие сокращения и обозначения

БД – база данных

ПО – программное обеспечение

СУБД – система управления базой данных

API (от англ. Application Programming Interface) – программный интерфейс приложения, содержащий набор правил и инструкция для обмена данными

AI (от англ. Artificial Intelligence) – искусственный интеллект

AMQP (от англ. Advanced Message Queuing Protocol) – протокол прикладного уровня, используемый для асинхронного обмена данными

CSS (от англ. Cascading Style Sheets) – язык стилей, определяющий отображение элементы HTML

DTO (от англ. Data Transfer Object) – паттерн проектирования, используемый для облегчения обмена данными

HTTP (от англ. Hypertext Transfer Protocol) – протокол прикладного уровня, обеспечивающий передачу данных в интернете без шифрования TLS

HTTPS (от англ. HyperText Transfer Protocol Secure) – Протокол защищенного передачи данных, использующий шифрование TLS

HTML (от англ. Hyper Text Markup Language) – язык гипертекстовой разметки

JPA (от англ. Java Persistence API) – спецификация, предоставляющая возможность сохранения Java - объекты в базе данных

JSON (от англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на синтаксисе JavaScript.

JWT (от англ. JSON Web Token) – открытый стандарт формирования токенов доступа

MVC (от англ. Model-View-Controller) – архитектурный паттерн, разделяющий приложение на три компонента: модель, представление и контроллер

REST (от англ. Representational State Transfer) – архитектурный стиль взаимодействия, определяющий принцип построения API

					<i>ПГУ 1.090301.24.001 ПЗ</i>	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

SQL (англ. Structured Query Language) – язык структурированных запросов

TLS (от англ. Transport Layer Security) – криптографический протокол, обеспечивающий защищенную передачу данных в интернете

XML (от англ. eXtensible Markup Language) – расширяемый язык разметки для хранения и передачи данных

					ПГУ 1.090301.24.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире каждый день создается огромное количество данных. В данных условиях локальное хранение информации становится не эффективным, так как оно не обеспечивает нужного уровня масштабируемости и ее доступности. Решением данной проблемы и является облачное хранение данных, позволяющее распоряжаться нужными данными с любого места через сеть интернета.

Однако простое размещение файлов недостаточно, на первый план также выходят вопросы управления правами доступа и системы поиска, по которой можно будет найти файлы не только по названию, но и по его смыслу.

Управление доступом разграничивает права доступа к файлам, что очень важно не только компаниям, но и обычным пользователям, желающим поделиться своими данными. Система поиска поможет пользователям находить быстро файлы по имени, тексту и его смыслу, что очень сильно сократит время на поиск самого файла.

Таким образом, разработка облачного хранилища с управлением доступом и системой поиска является актуальной темой. Веб-приложение позволит повысить удобство, безопасность и скорость работы с файлами.

1 Анализ предметной области

В данном разделе рассматривается анализ актуальности темы, анализ аналогов и составлена постановка задачи на основе проведенных анализов предметной области.

1.1 Анализ актуальности темы

В современном мире облачные хранилища стали популярны, так как предоставляют доступ к данным с любого места, где есть интернет, а также позволяют экономить локальное дисковое пространство.

Вместе с тем есть ряд проблем в использовании облачных хранилищ, среди которых важное место занимает разграничение прав доступа к данным. Не каждый файл должен доступен одинаково всем пользователям. Решением данной проблемы будет управление доступом над объектами, над которыми пользователь сам будет решать с какими правами он хочет поделиться с другим пользователем.

Не менее важной проблемой является долгий поиск нужного файла при их большом количестве. Решением данной проблемы является система поиска, при которой пользователь сможет легко искать файлы по названию, тексту, смыслу и другим параметрам файла.

Таким образом, разработка облачного хранилища с управлением доступом и системой поиска является актуальной темой, решающая востребованные проблемы при работе с информацией.

1.2 Анализ аналогов

На рынке существует огромное количество облачных хранилищ данных с самым различным функционалом. Чтобы выявить сильные и слабые стороны существующих решений был проведен анализ трех облачных хранилищ:

- облачное хранилище «Google Диск»;
- облачное хранилище «Dropbox»;
- облачное хранилище «Яндекс.Диск».

1.2.1 Облачное хранилище «Google Диск»

Интерфейс облачного хранилища «Google Диск» представлен на рисунке 1, 2 и 3.

Веб-приложение предоставляет следующие функции:

- система поиска, позволяющая искать файлы по названию, содержимому, типу и дате изменения;
- присутствует управление доступом, позволяющая разграничить права на файл;
- бесплатное хранилище объемом 15 ГБ.

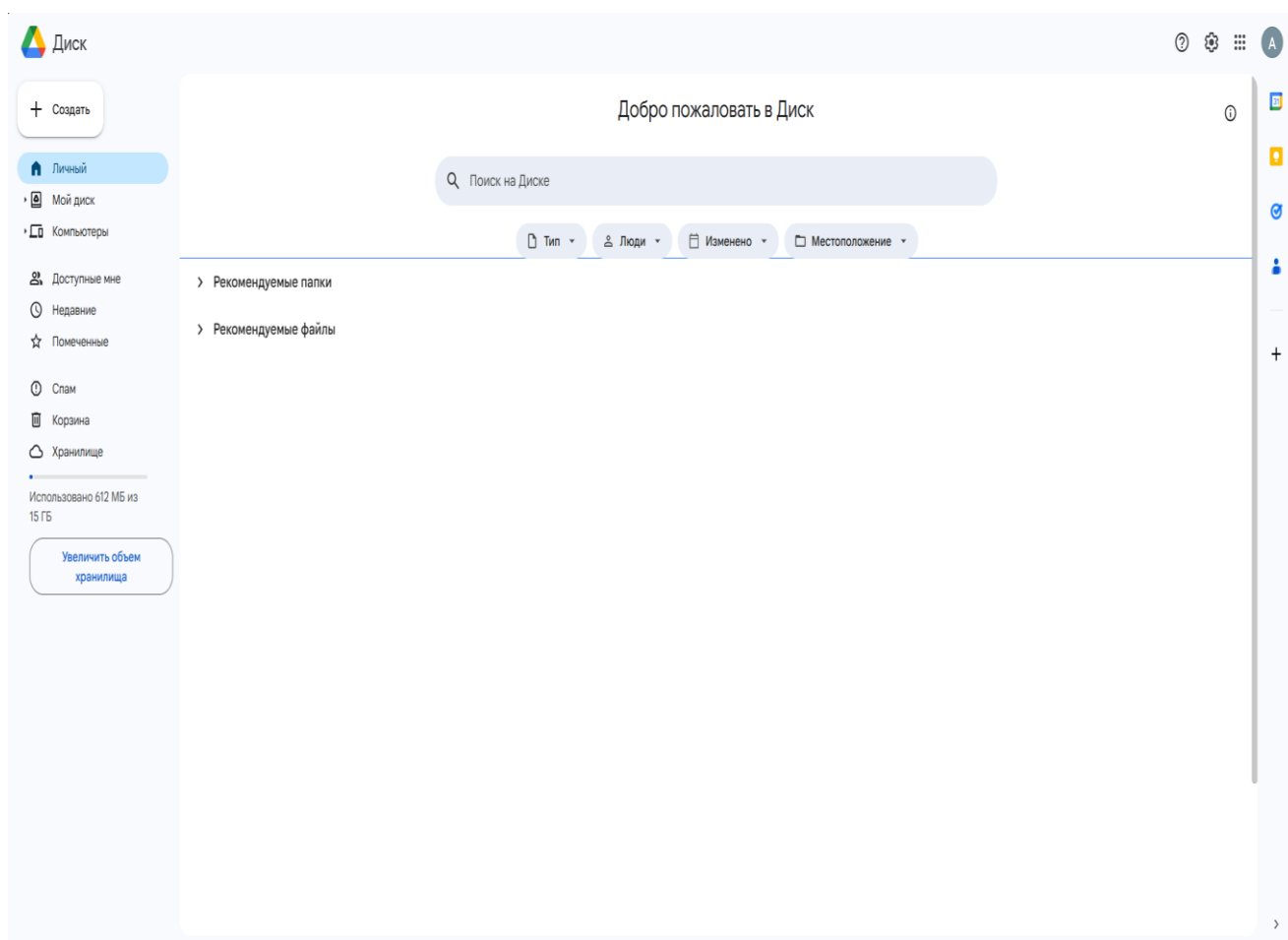


Рисунок 1 – Главная страница сервиса Google Диск

Доступ – Варианты заданий к лабораторным работам



Добавьте пользователей, группы, рабочие области и м...

Пользователи, имеющие доступ



Али Хоссейни (you)
alihuseyni523@gmail.com

Владелец

Общий доступ



Доступ ограничен. ▼

Открывать контент по этой ссылке могут только пользователи, имеющие доступ

🔗 Копировать ссылку

Готово

Рисунок 2 – Интерфейс настройки прав доступа к файлу

Расширенный поиск

✕

Тип

Любое

Владелец

Все в интернете

Содержит слова

Введите слова, найденные в файле

Название объекта

Введите часть названия файла

Местоположение

Везде

☐ Корзина
☐ Помеченные
☐ Зашифровано

Дата изменения

В любое время

Одобрения и электронные подписи

☐ Ожидают моего одобрения
☐ Мои запросы, ожидающие одобрения

Кому предоставлен доступ

Введите имя или адрес электронной почты...

Напоминания

-

Подробнее...

Восстановить

Поиск

Рисунок 3 – Интерфейс системы поиска

1.2.2 Облачное хранилище «Dropbox»

Интерфейс «Dropbox» представлен на рисунке 4.

«Dropbox» является одним из первых облачных хранилищ и до сих пор сохраняет свою популярность.

Ключевые функции «Dropbox»:

- возможность создавать общие папки;
- возможность защищать ссылки паролем в платной версии;
- версионирование файлов;

— поиск по названиям файлов и полнотекстовый поиск.

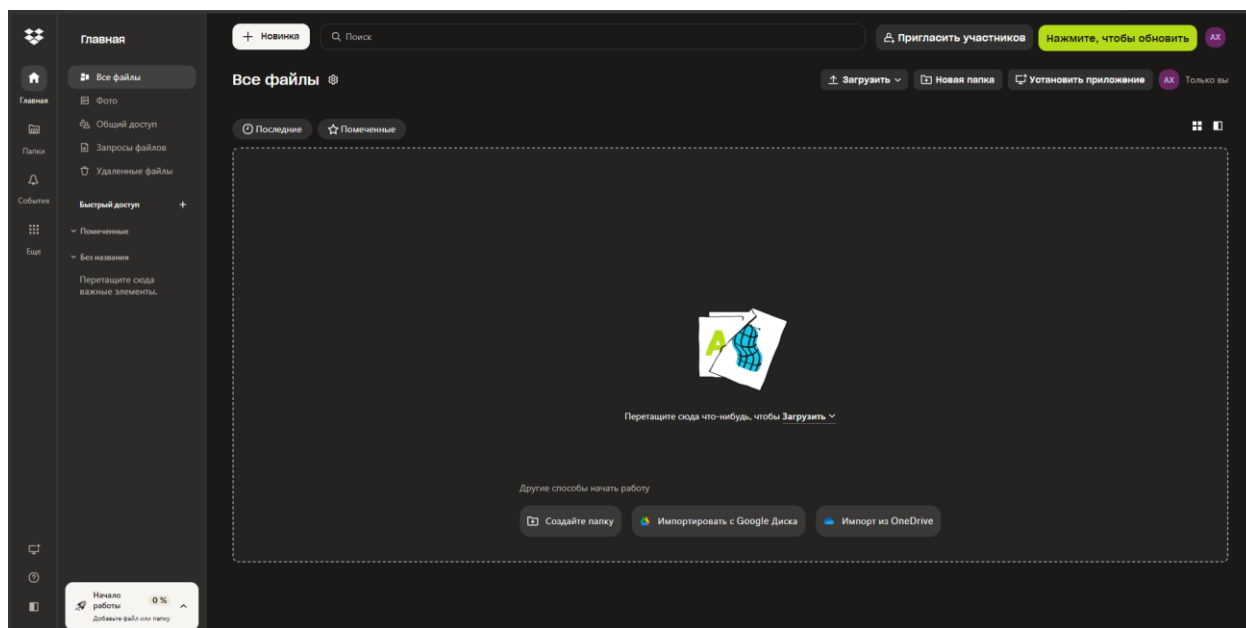


Рисунок 4 – Главная страница веб-версии Dropbox

1.2.3 Облачное хранилище «Яндекс.Диск»

Интерфейс сайта «Яндекс Диска» представлен на рисунке 5.

Одной из главной отличительной чертой этого хранилища является популярность в русскоязычном сегменте и интеграции с сервисами Яндекса.

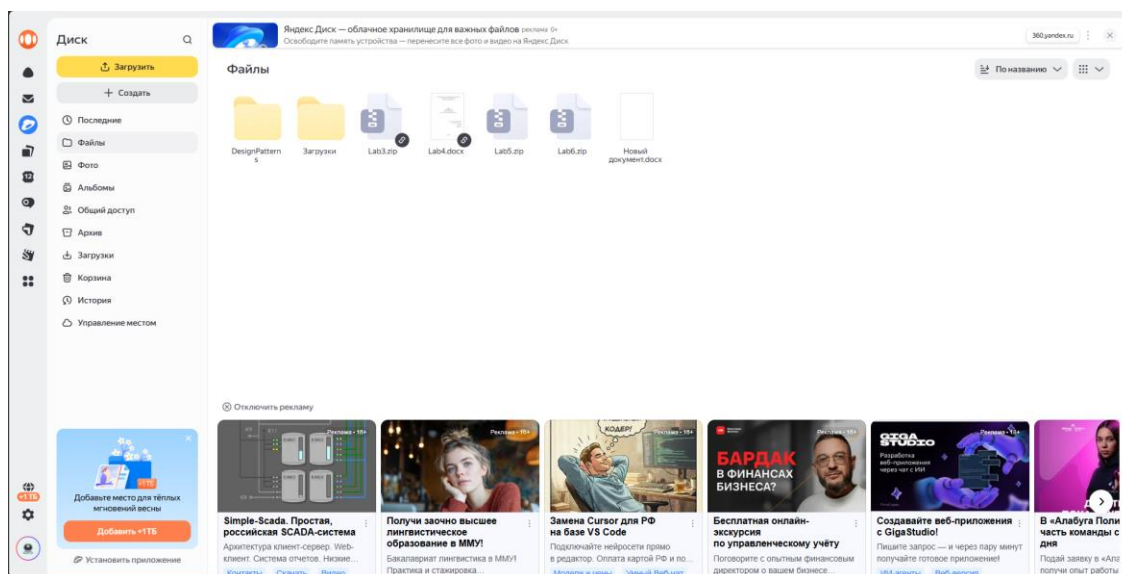


Рисунок 5 – Главная страница сервиса

Основные возможности «Яндекс.Диска»:

- бесплатный объём хранилища до 5 ГБ;
- управление доступом: возможность создавать публичные ссылки для просмотра или скачивания;
- поиск по именам файлов.

1.2.4 Вывод

Результаты анализа, демонстрирующие отрицательные и положительные возможности облачных хранилищ, представлены в таблице 1.

На основе проведённого анализа было принято решение о разработке собственного облачного хранилища с управлением правами доступа и системой поиска.

Таблица 1 – Достоинства и недостатки облачных хранилищ

Функционал	Google Диск	Dropbox	Яндекс.Диск
Бесплатный объём хранилища	15 ГБ	2 ГБ	5 ГБ
Предоставление доступа по ссылке	Да	Да	Да
Полнотекстовый поиск	Да	Нет	Нет
Гибкое назначение разных прав разным пользователям	Да	Да	Нет
Загрузка и скачивание файлов	Да	Да	Да

1.3 Постановка задачи

Разрабатываемое облачное хранилище с управлением доступом и системой поиска должно представлять собой веб-приложение, обеспечивающее следующий функционал:

- регистрация и авторизация пользователей;
- удаление учётной записи;
- выход из учётной записи;

- управление профилем пользователя;
- восстановление пароля;
- загрузка, скачивание, удаление, перемещение и переименование файлов;
- удаление, перемещение и переименование папок;
- сортировка объектов хранилища;
- наличие корзины и возможность восстановления объектов из неё;
- отображение индикатора занятого дискового пространства;
- предоставление доступа к файлам и папкам другим пользователям с указанием уровня прав (Читатель, Автор, Редактор, Владелец);
 - создание публичных ссылок к объектам с указанием уровня прав (Читатель, Автор, Редактор);
 - система поиска: поиск по имени, типу, дате файла, полнотекстовый и семантический поиск;
 - AI-чат по файлу.

Минимальные системные требования:

- браузер: Microsoft Edge, Mozilla Firefox, Opera;
- минимальная скорость интернет-соединения: 2-3 Мбит/сек;
- минимальный объем оперативной памяти: 1 ГБ.

2 Выбор технологий и средств разработки приложения

В данном разделе рассматривается и описывается выбор технологий и средств разработки облачного хранилища с управлением доступом и системой поиска. Выбор технологий был основан на постановке задачи из 1 раздела и необходимости масштабируемости, безопасности и надежности веб-приложения.

Сформированный в результате анализа стек технологий включает следующие технологии и средства разработки:

- язык программирования серверной части: Java;
- фреймворк серверной части Spring Framework, включая Spring Boot для конфигурации приложения, Spring MVC для реализации REST API, Spring Data JPA для взаимодействия с базой данных, Spring Security для аутентификации и Spring AI для интеграции с языковой моделью;
- система управления базами данных: PostgreSQL 16;
- брокер сообщений: RabbitMQ;
- язык программирования клиентской части: JavaScript;
- библиотека пользовательского интерфейса: React 18;
- среда разработки серверной части: IntelliJ IDEA, Visual Studio Code;
- среда разработки клиентской части: Visual Studio Code;
- система сборки и управления зависимостями: Apache Maven;
- среда контейнеризации: Docker;
- LLM-сервер: Ollama;
- веб-сервер: Apache Tomcat и Nginx.

2.1 Java

Java является одним из самых популярных языков программирования. Он используется в веб-приложениях, мобильной разработке, в финансовых приложениях, в играх, в облачных и распределенных системах. Причины

популярности в кроссплатформенности, масштабируемости, богатой экосистеме, стабильности и долгосрочной поддержки.

Java является объектно ориентированным языком программирования. Это позволяет логично разделять систему на отдельные сущности. Такой подход отлично подходит для разработки облачного хранилища с управлением доступом и системой поиска.

2.2 Spring Framework

Spring Framework является самым популярным фреймворком для языка программирования Java. С его помощью можно легко создавать слабосвязанные и легко тестируемые компоненты за счет его принципа внедрения зависимостей. В Spring существует огромное количество модулей, предоставляющая готовые решения для большинства задач веб-разработки, такие как: Spring MVC, Spring Data JPA, Spring Security, Spring AI [1].

2.2.1 Spring Boot

Spring Boot – это расширение фреймворка Spring Framework, которое позволяет автоматически подключить нужные библиотеки, настроить проект и быстро его запустить с минимальным набором настроек.

Одним из главных преимуществ Spring Boot является наличие встроенного веб-сервера Apache Tomcat по умолчанию, что позволяет запускать веб-приложение без необходимости установки внешнего сервера [2].

2.2.2 Spring MVC

Spring MVC – модуль Spring Framework для создания веб-приложений под паттерн MVC. Модуль предоставляет поддержку архитектурного стиля REST: контроллеры `@RestController`, принимающие HTTP-запросы, маппинг URL-шаблонов через аннотации `@GetMapping`, `@PostMapping` и другие, извлечение данных из тела запроса через аннотацию `@RequestBody` и формирование ответов через аннотацию `@ResponseBody`. Также модуль предоставляет глобальные обработчики исключений через `@ControllerAdvice` [1].

2.2.3 Spring Data JPA

Spring Data JPA – это модуль для упрощения взаимодействия с базой данных через Java Persistence API (JPA). За счет объявления интерфейса унаследованного от JpaRepository модуль самостоятельно реализует его во время выполнения. Модуль избавляет от написания шаблонного кода и обеспечивает удобный способ реализации слоя доступа к данным и управлением транзакциями [3].

2.2.4 Spring Security

Spring Security - это модуль для обеспечения безопасности. В основные функции модуля входит проверка аутентификации и проверку доступа к ресурсам. Spring Security поддерживает множество различных механизмов аутентификации, такие например как JWT и OAuth2 [4].

2.2.5 Spring AI

Spring Security – это модуль для интеграции языковых моделей в приложения. Он предоставляет абстракции для упрощения работы с языковыми моделями. В модуле присутствует spring-ai-model, которая определяет абстракцию модели, реализуемая поставщиками [5].

2.3 PostgreSQL

PostgreSQL – это бесплатная объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. Одной из ключевой особенностью является ACID-совместимость, что гарантирует надежность транзакций по 4 критериям: атомарность, согласованность, изолированность, устойчивость [6].

Также большим преимуществом PostgreSQL является наличие типов данных tsvector, tsquery и GIN-индексы, которые позволяют выполнять полнотекстовый поиск.

2.4 RabbitMQ

RabbitMQ – это брокер сообщений с открытым исходным кодом. Используется для асинхронного обмена сообщениями и гарантии надежной доставки сообщений между компонентами системы. RabbitMQ использует протокол AMQP и позволяет вынести операции в фоновый режим [7].

2.5 JavaScript

JavaScript является самым популярным языком программирования клиентской части веб-приложений. Он поддерживается всеми современными браузерами. Является мультипарадигменным, с динамической типизацией, интерпретируемым языком программирования, поддерживает императивный, функциональный и объектно-ориентированные стили. Позволяет динамически менять HTML и CSS на веб-странице и обмениваться данными с сервером.

2.6 React

React – это декларативная JavaScript библиотека для реализации пользовательского интерфейса. Архитектура состоит в использовании компонентного подхода, при котором пользовательский интерфейс делится на независимые и многократно используемые элементы [8].

Одним из ключевых особенностей является виртуальный DOM, при котором создается кэш-структура в памяти и React оптимально обновляет только те элементы, которые реально изменились, из-за вычитывания разницы между предыдущей и текущей версией.

2.7 IntelliJ IDEA

IntelliJ IDEA является одной из самых популярных сред разработки для Java. Она изначально и создавалась только для этого языка программирования, но со временем стала поддерживать и другие языки. IDE поддерживает подсветку синтаксиса, автодополнения и анализ ошибок.

2.8 Visual Studio Code

Visual Studio Code – это один из самых популярных в веб-разработке текстовый редактор для языка программирования JavaScript. Функционал включает подсветку синтаксиса, автодополнения, интеграция с Git, подсветку ошибок и предложение их исправления, встроенный терминал.

2.9 Apache Maven

Apache Maven – это фреймворк для автоматизации сборки проектов на основе информации содержащейся в файле pom.xml. Главным преимуществом является централизованное управления зависимостями через pom.xml, с его помощью Maven автоматически скачивает нужные библиотеки, прописанные в файле из удаленных репозиториях [9].

2.10 Docker

Docker – это платформа для контейнеризации приложений, позволяет упаковывать приложение со всеми его зависимостями в изолированный контейнер. Основная идея - работа приложений независимо от их окружения и операционной системы [10].

Для описания всех сервисов и их сетевых настроек существует Docker Compose. Он позволяет описывать все эти данные в единственном YAML-файле.

2.11 Ollama

Ollama – бесплатная программная платформа для выполнения больших языковых моделей [11]. Ollama позволяет скачивать языковые модели из репозиториях, запускать их в интерактивном режиме.

Ключевые преимущества:

- запросы не передаются на внешние сервисы;
- после загрузки модели можно использовать ее без доступа к интернету;
- поддерживаются разные типы модели.

2.12 Apache Tomcat

Apache Tomcat является встроенным по умолчанию веб-сервером в Spring Boot. Именно через веб-сервер обрабатываются запросы от пользователей. Tomcat создаёт среду для работы Java-сервлетов и JSP-страниц и является кроссплатформенным [2].

2.13 Nginx

Nginx – веб-сервер с открытым исходным кодом. Через него обрабатываются запросы от пользователей. Одной из ключевой функцией является обратный прокси сервер, за счет чего происходит перенаправления запросов на серверы с балансировкой нагрузки и кэшированием [12].

					ПГУ 1.090301.24.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

3 Архитектура веб-приложения

Данный раздел рассматривает общую архитектуру разработанного облачного хранилища с управлением доступом и системой поиска, каждый отдельно взятый микросервис, а также взаимодействие выбранных технологий внутри продукта. Архитектурным подходом выбрана микросервисная архитектура на уровне системы и слоистая архитектура внутри каждого серверного сервиса.

3.1 Общая архитектура системы

Архитектурным подходом на уровне системы является микросервисная архитектура. Микросервисная архитектура в отличие от монолитной подразумевает разделение системы на множество независимых сервисов. Такой подход позволяет сделать систему более гибкой, масштабируемой и устойчивой к сбоям.

Система включает следующие компоненты:

- backend - сервис, отвечающий за регистрацию, авторизацию, управление файлами, папками, корзиной, правами доступа, публичными ссылками и профилем пользователя;
- search-service – сервис, отвечающий за индексацию и поиск;
- chat-service – сервис, отвечающий за чат с документами на основе RAG технологии;
- email-service – сервис, отвечающий за асинхронной отправки писем подтверждения электронной почты и восстановления пароля;
- reranker-service – сервис, отвечающий за переранжирования результатов поиска;
- frontend – клиентская часть приложения.

Приложение включает следующие инфраструктурные компоненты:

- база данных PostgreSQL 16 с расширениями pgvector, pg_trgm, uuid-osspl и unaccent;

- платформу Ollama для запуска локальных языковых моделей bge-m3 и qwen2.5:3b;
- брокер сообщений RabbitMQ;
- обратный прокси-сервер Nginx;

Все компоненты разворачиваются через Docker Compose.

Архитектура приложения представлена на рисунке 6.

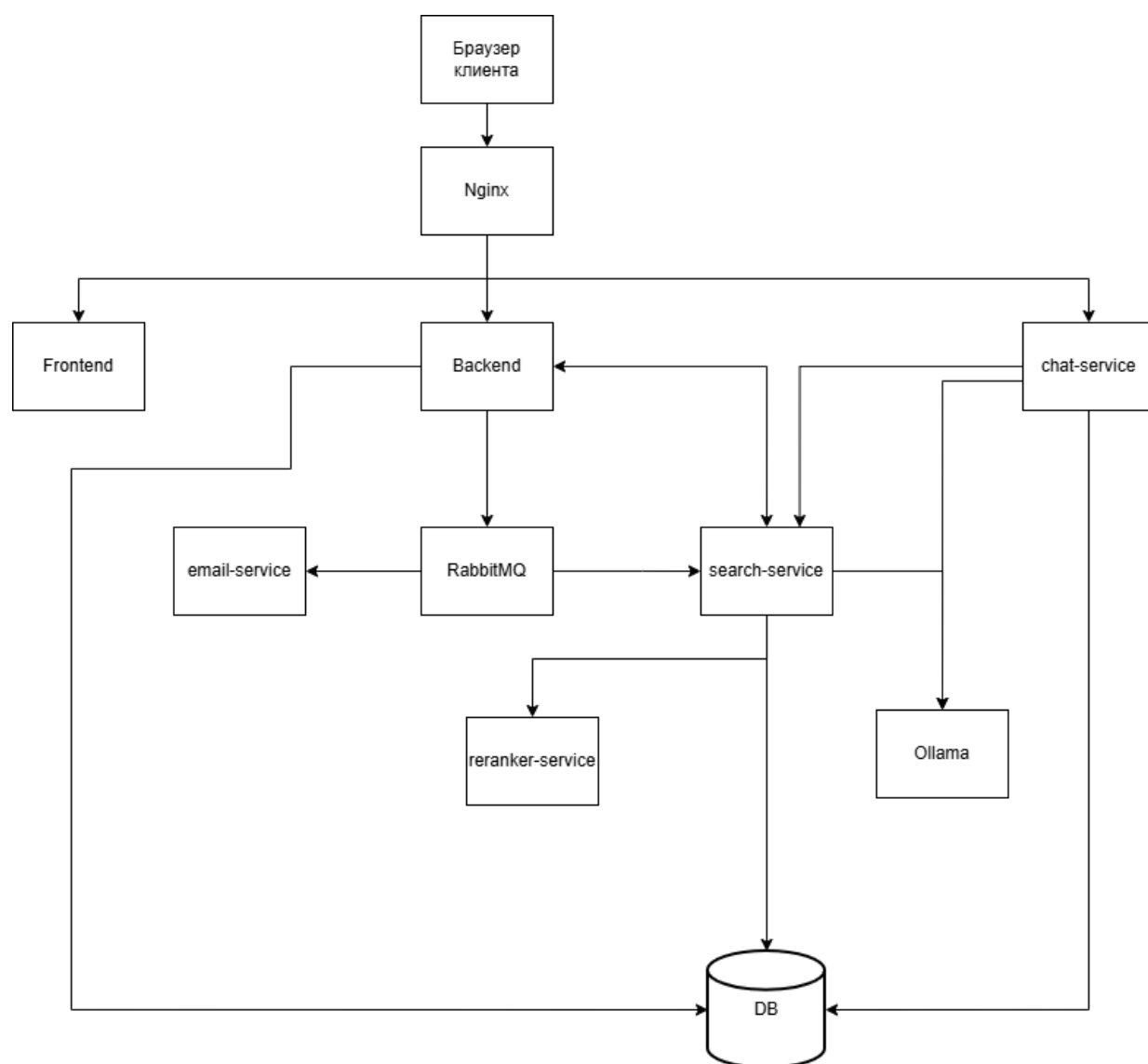


Рисунок 6 – Архитектура приложения

3.2 Слоистая архитектура серверных сервисов

Каждый микросервис, написанный на Java реализует слоистую архитектуру. Данный подход разделяет код на несколько слоев, каждый из которых имеет свою зону ответственности и взаимодействует только со смежными слоями.

В проекте выделяются следующие слои:

- слой представления. Самый верхний слой, принимающий HTTP-запросы. Контроллеры не содержат бизнес логику, их основной задачей является маршрутизация запросов в сервисный слой и формирование ответа. Контроллеры помечаются аннотацией `@RestController`;
- слой бизнес-логики. Вся логика приложения сосредоточена в слое сервисов. Сервисы помечаются аннотацией `@Service`. Этот слой получает данные от контроллеров и взаимодействует с репозиториями, другими сервисами, публикует события в RabbitMQ;
- слой доступа к данным. Слой репозитория для взаимодействия с базой данных. Интерфейсы репозитория наследуются от `JpaRepository`, за счет чего у репозитория автоматически появляются стандартные CRUD операции. Этот слой используется сервисным слоем для записи данных в базу данных;
- слой сущностей и объектов передачи данных. JPA-сущности это классы, которые напрямую отображаются в базе данных. Помечаются аннотацией `@Entity`. DTO классы нужны для передачи данных между сервисами и слоями;
- слой безопасности. Этот слой пронизывает все остальные слои и реализован через Spring Security. `JwtAuthenticationFilter` перехватывает каждый входящий запрос и извлекает JWT токен, проверяет его и устанавливает помещает информацию о пользователе в `SecurityContextHolder`.

Таким образом, слоистая архитектура обеспечивает слабую связанность компонентов приложения, что сильно увеличивает масштабируемость системы.

3.3 Основной сервис (backend)

Сервис backend работает на порту 8080 и реализует следующие пользовательские сценарии: регистрация, авторизация, загрузка и скачивание файлов, управление правами доступа, профилем пользователя, корзиной и публичными ссылками.

Слоистая архитектура backend-сервиса:

- слой представления. Этот слой состоит из пяти контроллеров. AuthController обрабатывает эндпоинты регистрации, авторизации, выхода из учетной записи, подтверждения почты, сброса пароля и обновления токена. FileController обрабатывает эндпоинты операций над файлами. FileController обрабатывает эндпоинты с публичными ссылками. PublicFileController обрабатывает доступ к объектам по токену публичной ссылки. UserController реализует эндпоинты взаимодействия с профилем;

- слой бизнес-логики. Этот слой состоит из шести классов. FileService обрабатывает логику работы с файлами. AuthService реализует регистрацию, авторизацию и восстановление пароля. UserService обрабатывает взаимодействие с профилем пользователя. ShareLinkService для управления публичными ссылками. TrashCleanupService для удаления по расписанию объектов с корзины. SearchServiceClient обеспечивает вызов эндпоинтов search-service;

- слой доступа к данным. Этот слой состоит из шести репозиториях. StorageObjectRepository обеспечивает рекурсивное получение всех дочерних объектов, поиск по названию, дате, MIME-типу и полнотекстовый. AccessPermissionRepository нужен для хранения записей о правах доступа. RefreshTokenRepository обеспечивает управление токенами обновления сессий. ShareLinkRepository для работы с публичными ссылками. TrashItemRepository обеспечивает хранение данных корзины. UserRepository для поиска пользователей;

- слой сущностей и объектов передачи данных. Этот слой состоит из шести JPA-сущностей и из 27 DTO классов. JPA-сущности: User, StorageObject,

AccessPermission, RefreshToken, TrashItem, ShareLink. DTO-классы: ApiResponse, StorageObjectDto, UserDto, SemanticSearchResult, MessageDto, SearchIndexMessage, EmailMessage, AuthDto, RegisterRequest, LoginRequest, RefreshRequest, ForgotPasswordRequest, ResetPasswordRequest, VerifyEmailRequest, AuthResponse, FileOperationDto, CreateFolderRequest, RenameRequest, MoveRequest, CopyRequest, SearchRequest, GrantPermissionRequest, PermissionDto, ShareLinkRequest, ShareLinkDto, UpdateProfileRequest, ChangePasswordRequest;

- слой безопасности. Этот слой пронизывает все остальные слои и реализован через Spring Security. JwtAuthenticationFilter перехватывает каждый входящий запрос и извлекает JWT токен, проверяет его и устанавливает информацию о пользователе в SecurityContextHolder.

3.4 Сервис поиска (search-service)

Сервис поиска работает на порту 8081 и отвечает за индексацию файлов и выполнение поисковых запросов. Сервис взаимодействует с основным сервисом через RabbitMQ.

Слоистая архитектура search-service организована следующим образом:

- слой представления состоит из классов: SemanticSearchController, отвечающего за семантический поиск, гибридный поиск, переиндексацию файлов, получение релевантных текстовых фрагментов;

- слой бизнес-логики. Этот слой состоит из семи классов. HybridSearchService реализует гибридный поиск. SemanticSearchService нужен для семантического поиска. При гибридном поиске реранкинг не применяется. ChunkingService нужен для разбития текста на фрагменты. EmbeddingService реализует генерацию векторного представления текста. TextExtractorService используется для извлечения текста из файлов. RerankService необходим для переранжирования результатов, для этого отправляет запрос reranker-service. ReindexService нужен для нахождения файлов без эмбеддингов или без чанков;

- слой доступа к данным. Этот слой использует четыре класса.

EmbeddingJdbcRepository нужен для сохранения и обновления whole-document эмбедингов. ChunkEmbeddingJdbcRepository реализует сохранение, удаление и поиск векторных эмбедингов чанков в базе данных. SearchIndexRepository нужен для полнотекстового поиска. EmbeddingRepository требуется для загрузки векторных эмбедингов из базы данных;

- слой сущностей и объектов передачи данных. Этот слой состоит из двух JPA-сущности и двух DTO-классов. JPA-сущности: DocumentChunk, StorageObjectIndex. DTO-классы: SearchIndexMessage, SemanticSearchResult.

В сервисе реализован гибридный поиск: результаты полнотекстового и семантического поиска на основе эмбедингов объединяются алгоритмом RRF. После слияния результат переранжируется сервисом reranker-service.

3.5 Сервис чата с документами (chat-service)

Сервис чата работает на порту 8084. С его помощью пользователям можно задавать вопрос по содержимому файлу. Для этого используется архитектура RAG.

Слоистая архитектура chat-service организована следующим образом:

- слой представления. Этот слой состоит из двух контроллеров. FileChatController для истории переписки, SSE-стриминга ответа языковой модели и очистки истории чата. HealthController необходим для мониторинга доступности сервиса;

- слой бизнес-логики. Этот слой состоит из 4 классов. FileChatService реализует основной пайплайн RAG-чата. SearchServiceClient для запроса релевантных фрагментов у сервиса search-service. FileAccessChecker нужен для загрузки метаданных файла из таблицы storage_objects и проверки прав доступа. SearchServiceClient - выполняет запрос к search-service для получения контекста;

- слой доступа к данным. Этот слой состоит из ChatMessageRepository для сохранения, загрузки и удаления истории с чатом и FileAccessChecker для проверки доступа к файлу;

- слой сущностей и объектов передачи данных. Слой состоит из класса ChatDto с четырьмя вложенными классами: MessageResponse, ApiResponse<T>, FileInfo, PermissionInfo;

- слой безопасности. Этот слой пронизывает все остальные слои и реализован через Spring Security. JwtAuthenticationFilter перехватывает каждый входящий запрос и извлекает JWT токен, проверяет его и устанавливает помещает информацию о пользователе в SecurityContextHolder.

Nginx сконфигурирован с отключённой буферизацией прокси для поддержки SSE-стриминга.

3.6 Сервис отправки электронной почты (email-service)

Сервис электронной почты работает на порту 8083. Он подписан на очередь RabbitMQ и реализует отправку писем по запросам от основного сервиса.

Слоистая архитектура email-service организована следующим образом:

- слой представления реализован HealthController для мониторинга доступности сервиса. Точкой входа является очередь из RabbitMQ;

- слой бизнес-логики. Этот слой реализован классом EmailConsumer. Этот класс нужен для получения сообщения из RabbitMQ и отправки письма пользователю;

- слой сущностей и объектов передачи данных. Этот слой представлен единственным классом EmailMessage.

UML-диаграмма классов микросервиса email-service представлена на рисунке 7.

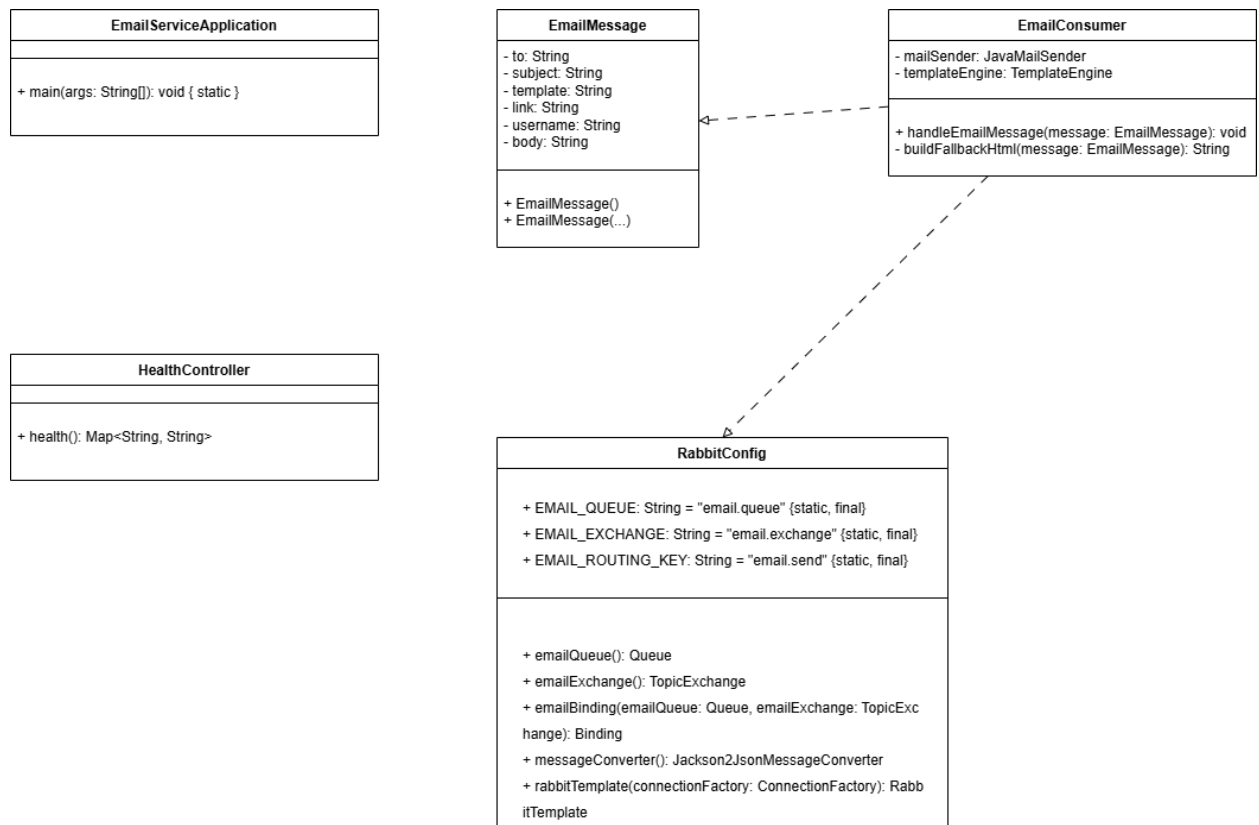


Рисунок 7 – UML-диаграмма классов

3.7 Сервис переранжирования (reranker-service)

Сервис переранжирования реализован на языке программирования Python и работает на порту 8082. В сервисе присутствует один эндпоинт /rerank. Эндпоинт принимает поисковый запрос и текстовые фрагменты, а возвращает переупорядоченный список с оценкой релевантности. В качестве cross-encoder модели используется BAAI/bge-reranker-v2-m3.

3.8 Взаимодействие компонентов

Взаимодействие между компонентами системы организовано следующим образом. Клиентский браузер направляет все запросы на обратный-прокси Nginx, который маршрутизирует их в chat-service при /api/files/{uuid}/chat/, а остальные в основной backend. Запросы к корню перенаправляются в frontend.

В проекте используется RabbitMQ, с помощью этого инструмента реализована асинхронная работа между сервисами. При загрузке файла основной

сервис публикует событие для индексации файла, а search-service извлекает это событие и индексирует файл. Также основной сервер публикует события на отправку писем, а уже email-service извлекает это событие и выполняет его

В проекте также используется синхронное взаимодействие между некоторыми сервисами. Chat-service для получения релевантного контекста обращается к search-service. При поисковых запросах backend сервис обращается к search-service. Search-service обращается к reranker-service для переранжирования результатов и к Ollama для создания эмбедингов.

Таким образом, выбранная микросервисная архитектура позволяет разделить ответственность между сервисами, увеличить надежность и масштабируемость веб-приложения, а слоистая архитектура обеспечивает слабую связанность кода.

					ПГУ 1.090301.24.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

4 Реализация приложения

В данном разделе рассматривается описание работы приложения, взаимодействия между клиентом и микросервисами, функционал приложения, реализация базы данных и основных компонентов.

4.1 Описание работы приложения

При взаимодействии с веб-приложением пользователем браузер отправляет запросы на обратный прокси-сервер Nginx. Nginx автоматически маршрутизирует запросы `/api/files/{uuid}/chat/` в `chat-service`, остальные запросы `/api/` перенаправляются в `backend`.

На рисунке 8 представлена UML-диаграмма развёртывания.

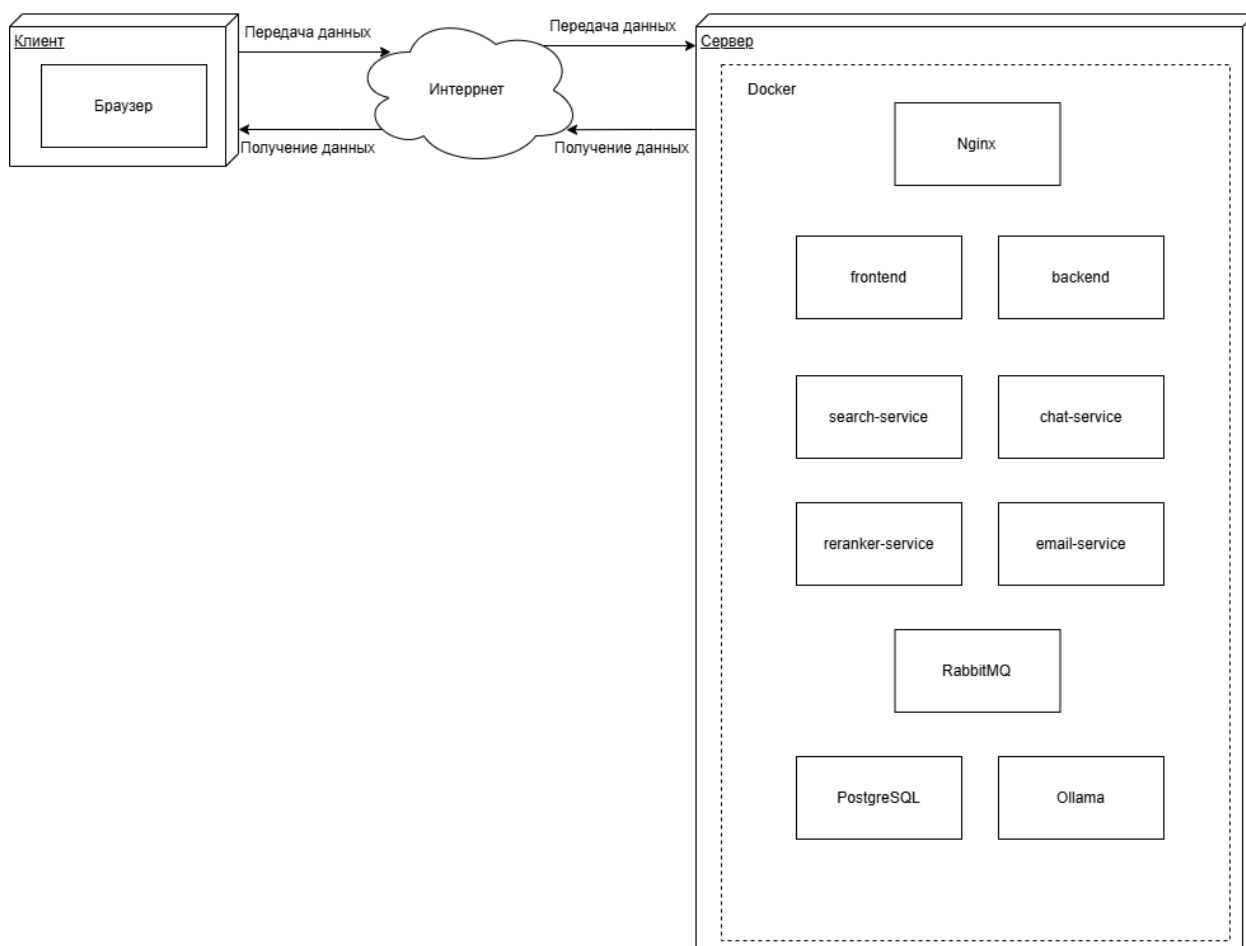


Рисунок 8 – UML-диаграмма развёртывания

При загрузке файла он сохраняется по пути/storage/{userId}/{filename} backend сервисом, дальше сервис отправляет в очередь RabbitMQ search.index.queue запрос на индексацию файла. search-service подписан на очередь индексации файла, забирает этот запрос и выполняет следующие действия: извлекает текст файла через Apache Tika, обновляет полнотекстовый вектор content_tsv, создает эмбединг всего текста, делит файл на чанки и векторизует каждый из них [13].

Чат по содержимому файлу с AI реализована через архитектуру RAG. При поступлении запроса к чату chat-service обращается к search-service за наиболее релевантными чанками файла для получения контекста. После этого chat-service создает промт с контекстом и отправляет запрос к Ollama для генерации ответа пользователю. В Ollama модель qwen2.5:3b генерирует ответ пользователю через SSE-стриминг.

Асинхронно через RabbitMQ выполняется не только индексация файла, но и запросы, связанные с электронной почтой. Для взаимодействия с почтой был создан отдельный микросервис email-service, который подписан на очередь RabbitMQ email.queue.

На рисунках 9-11 представлена UML-диаграмма последовательности функционирования приложения.

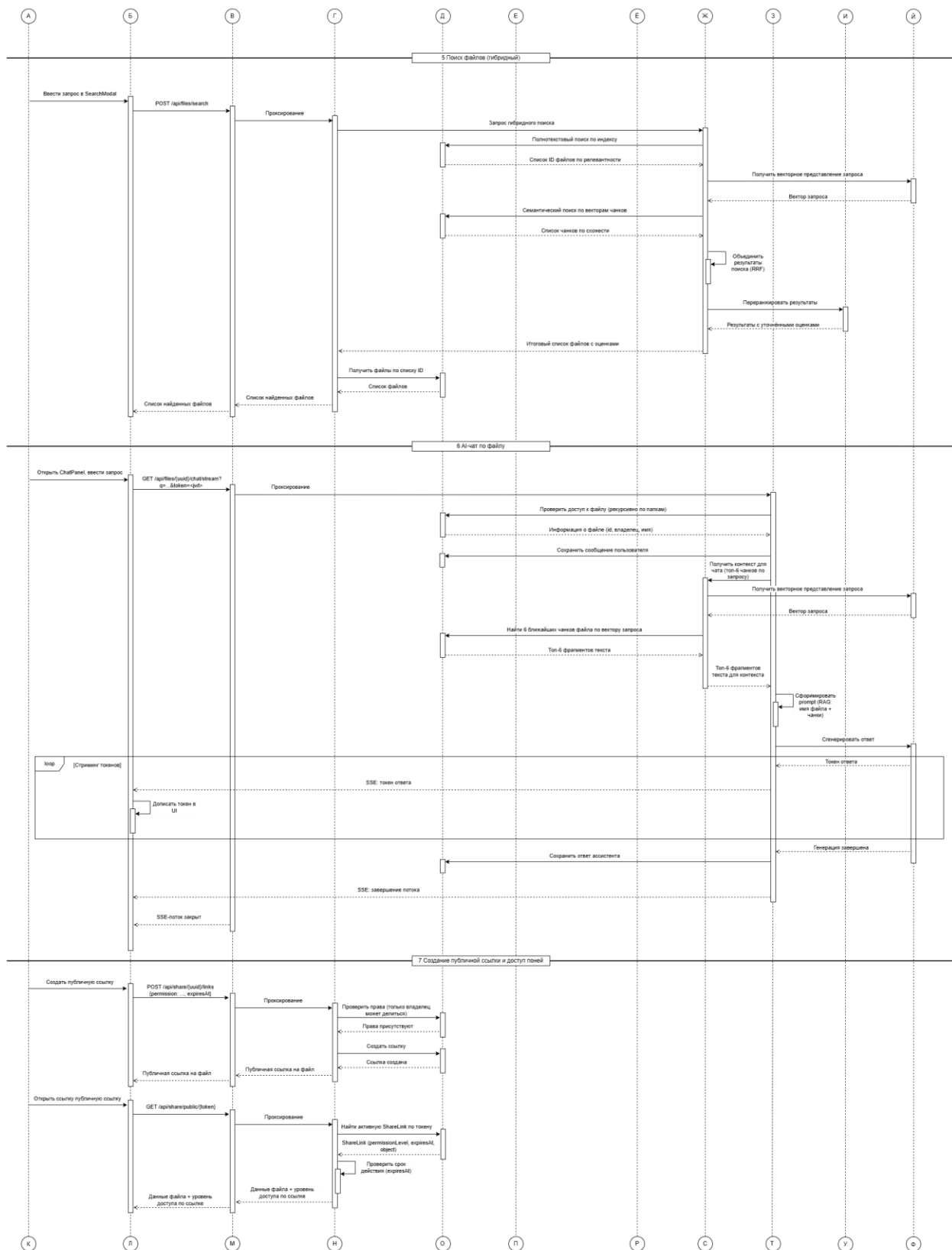


Рисунок 10 – UML-диаграмма последовательности функционирования приложения. Часть 2

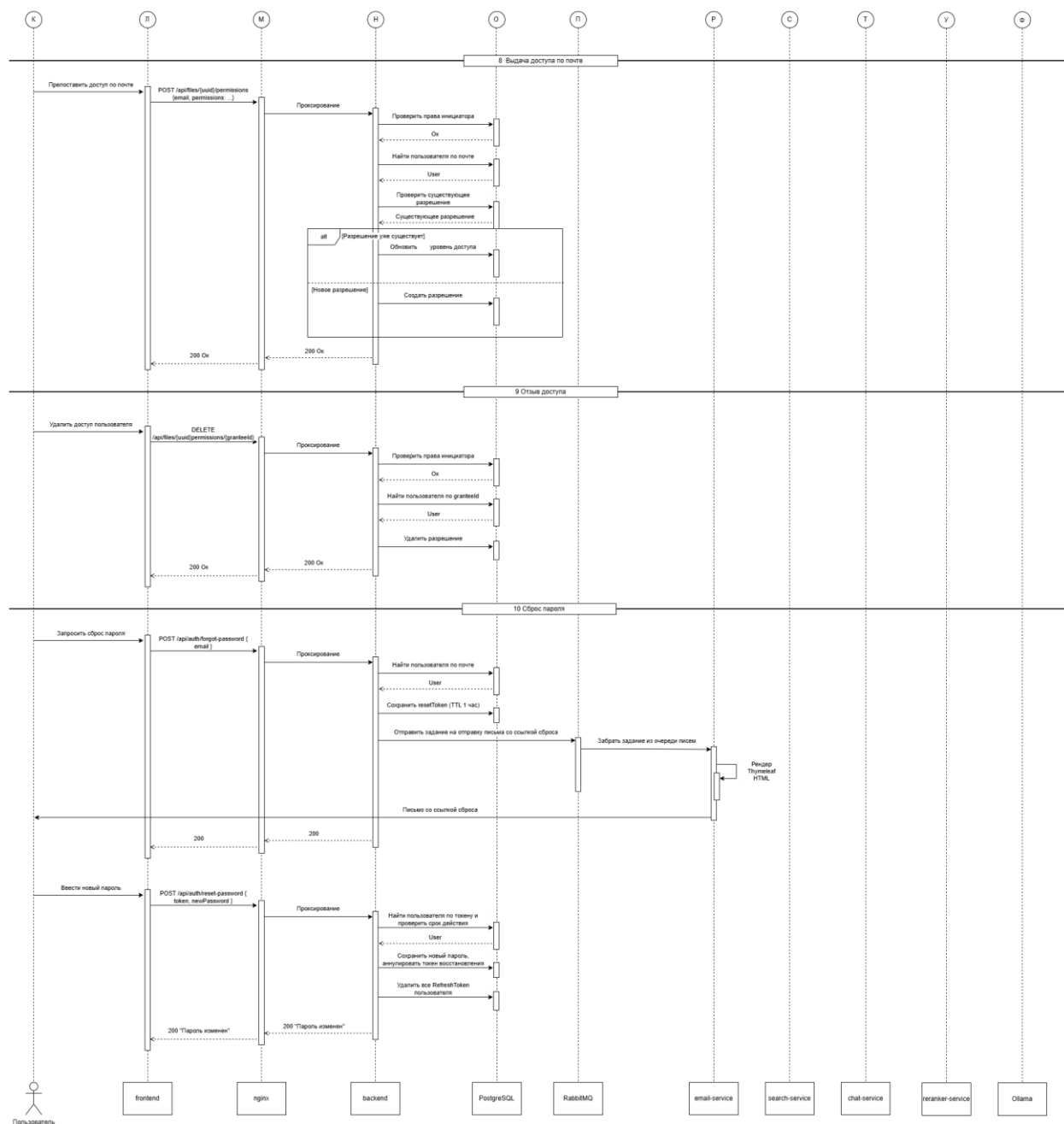


Рисунок 11 – UML-диаграмма последовательности функционирования приложения. Часть 3

4.2 Структура базы данных

В данном подразделе рассматриваются основные таблицы, атрибуты и связи в базе данных, реализованные для облачного хранилища с управлением доступом и системой поиска. А также необходимые расширения СУБД для реализации всего функционала.

В результате анализа всех необходимых требований было решено использование следующих таблиц в базе данных:

- таблица users с учётными данными, информации об использованном дисковом пространстве, токенами верификации и восстановления пароля (см. таблицу 2);
- таблица refresh_tokens для хранения refresh-токенов (см. таблицу 3);
- таблица storage_objects для хранения метаданных файлов и папок (см. таблицу 4);
- таблица access_permissions для хранения прав доступа к папкам и файлам с уровнями доступа READ, CONTRIBUTE, WRITE и OWNER (см. таблицу 5);
- таблица share_links для хранения публичных ссылок (см. таблицу 6);
- таблица trash_items для хранения элементов корзины (см. таблицу 7);
- таблица document_chunks для хранения текстовых фрагментов файлов и соответствующим им векторных эмбедингов (см. таблицу 8);
- таблица chat_messages для хранения истории переписки с AI-ассистентом (см. таблицу 9).

В качестве системы управления базами данных используется PostgreSQL со следующими расширениями: pg_trgm для поиска по триграммам, uuid-ossr для генерации UUID, unaccent для нормализации текста.

Таблица 2 – Атрибуты таблицы «users»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор пользователя	PK	1
email	VARCHAR(255)	Электронная почта		user@mail.ru
username	VARCHAR(100)	Имя пользователя (логин)		oleg_ivanov
password_hash	VARCHAR(255)	Хэш пароля		\$3b\$13\$...
first_name	VARCHAR(100)	Имя		Олег

Поле	Тип данных	Описание	Ключ	Пример
last_name	VARCHAR(100)	Фамилия		Иванов
avatar_url	VARCHAR(500)	Ссылка на аватар		/avatars/100.png
is_email_verified	BOOL	Флаг подтверждения почты		true
email_verification_token	VARCHAR(255)	Токен подтверждения почты		Abc623...
email_verification_expires	TIMESTAMP	Срок действия токена подтверждения		2026-05-21 15:00:54.716
password_reset_token	VARCHAR(255)	Токен сброса пароля		yyz207...
password_reset_expires	TIMESTAMP	Срок действия токена сброса		2026-05-21 16:00:54.716
storage_used	INT8	Используемое дисковое пространство (байт)		2048000
storage_limit	INT8	Лимит хранилища (байт)		10737418240
created_at	TIMESTAMP	Дата создания записи		2026-05-21 10:00:54.716
updated_at	TIMESTAMP	Дата последнего обновления		2026-05-21 11:00:54.716
is_deleted	BOOL	Флаг мягкого удаления		false

Таблица 3 – Атрибуты таблицы «refresh_tokens»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор токена	PK	1
user_id	INT8	Идентификатор пользователя	FK	5
token	VARCHAR(512)	Значение refresh-токена		eyJhbGci...
expires_at	TIMESTAMP	Срок действия токена		2026-05-21 10:00:54.716
created_at	TIMESTAMP	Дата создания		2026-04-21 10:00:54.716

Таблица 4 – Атрибуты таблицы «storage_objects»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор объекта	PK	1
uuid	UUID	Уникальный публичный идентификатор		550e5500-...
owner_id	INT8	Идентификатор владельца	FK	3
parent_id	INT8	Идентификатор родительской папки	FK	2
name	VARCHAR(500)	Имя файла или папки		document.pdf
object_type	VARCHAR(10)	Тип объекта		FILE
mime_type	VARCHAR(255)	MIME-тип		application/

Поле	Тип данных	Описание	Ключ	Пример
		файла		vnd.openxmlformats-officedocument.wordprocessingml.document
size	INT8	Размер файла в байтах		254800
storage_path	VARCHAR(1000)	Путь к физическому файлу на диске		users/5/doc.pdf
is_deleted	BOOL	Флаг мягкого удаления		false
deleted_at	TIMESTAMP	Дата удаления		2026-05-21 09:00:54.716
created_at	TIMESTAMP	Дата создания		2026-04-21 10:00:54.716
updated_at	TIMESTAMP	Дата последнего обновления		2026-04-22 12:00:54.716
content_tsv	TSVECTOR	Вектор полнотекстового поиска		-1':298,988 '-104':1413 '-4'...
embedding_updated_at	TIMESTAMP	Дата обновления векторного эмбединга		2026-04-02 08:00:54.716
embedding	_FLOAT8	Векторное представление файла		[-1.9630684852600098, ...]

Таблица 5 – Атрибуты таблицы «access_permissions»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор записи о	PK	1

Поле	Тип данных	Описание	Ключ	Пример
		доступе		
object_id	INT8	Идентификатор объекта хранилища	FK	10
grantee_id	INT8	Идентификатор пользователя, которому выдан доступ	FK	4
permission_level	VARCHAR(20)	Уровень доступа (READ / CONTRIBUTE / WRITE / OWNER)		READ
granted_by	INT8	Идентификатор пользователя, выдавшего доступ	FK	1
granted_at	TIMESTAMP	Дата и время выдачи доступа		2026-03-15 14:00:54.716

Таблица 6 – Атрибуты таблицы «share_links»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор ссылки	PK	1
token	VARCHAR(255)	Уникальный токен публичной ссылки		abc123xyz
object_id	INT8	Идентификатор объекта хранилища	FK	7
created_by	INT8	Идентификатор создавшего ссылку	FK	2
permission_level	VARCHAR(20)	Уровень доступа по ссылке		READ
expires_at	TIMESTAMP	Срок действия ссылки		2026-12-31

Поле	Тип данных	Описание	Ключ	Пример
				23:59:54.716
is_active	BOOL	Флаг активности ссылки		true
created_at	TIMESTAMP	Дата создания ссылки		2026-05-01 10:00:54.716

Таблица 7 – Атрибуты таблицы «trash_items»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор записи корзины	PK	1
object_id	INT8	Идентификатор удалённого объекта	FK	12
user_id	INT8	Идентификатор пользователя	FK	3
original_parent_id	INT8	Идентификатор исходной родительской папки		5
original_path	TEXT	Исходный путь файла в хранилище		/work/lab1.docx
deleted_at	TIMESTAMP	Дата перемещения в корзину		2026-04-10 11:00:54.716
auto_delete_at	TIMESTAMP	Дата автоматического удаления		2026-05-10 11:00:54.716

Таблица 8 – Атрибуты таблицы «document_chunks»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор	PK	1

Поле	Тип данных	Описание	Ключ	Пример
		фрагмента		
object_id	INT8	Идентификатор исходного файла	FK	10
owner_id	INT8	Идентификатор владельца файла	FK	3
chunk_index	INT	Порядковый номер фрагмента в документе		0
chunk_text	TEXT	Текстовое содержимое фрагмента		Lab1.docx Лабораторная работа...
embedding	PUBLIC.VECTOR	Векторный эмбединг фрагмента для поиска		[- 1.258682,0.38332322,...]

Таблица 9 – Атрибуты таблицы «chat_messages»

Поле	Тип данных	Описание	Ключ	Пример
id	BIGSERIAL	Идентификатор сообщения	PK	1
object_id	INT8	Идентификатор файла- контекста	FK	10
user_id	INT8	Идентификатор пользователя	FK	3
role	VARCHAR(20)	Роль отправителя		user
content	TEXT	Текст сообщения		Напиши ключевые пункты файла
created_at	TIMESTAMP	Дата и время сообщения		2026-05-01 15:00:54.716

На рисунке 12 представлена схема базы данных.

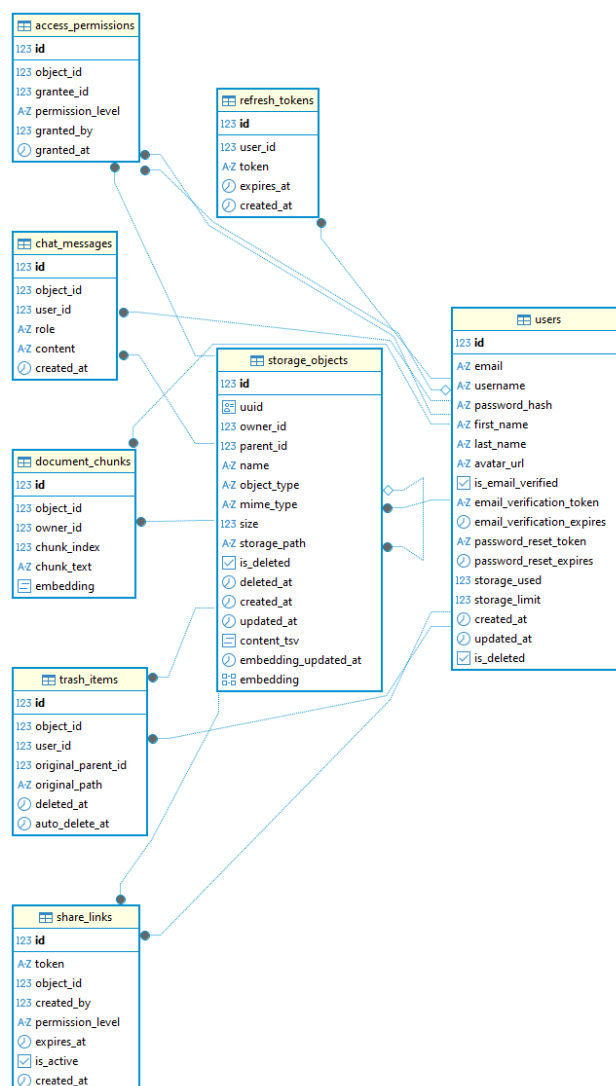


Рисунок 12 – Схема базы данных

4.3 Функционал приложения

Функционал приложения отличается в зависимости от того авторизован ли пользователь или нет.

4.3.1 Функционал неавторизованного пользователя

Функционал неавторизованного пользователя сильно ограничен и состоит из следующих возможностей:

- регистрация новой учетной записи;
- авторизация;
- сброс пароля по почте;

- просмотр объектов по публичной ссылке.
- скачивание файлов при наличии уровня доступа «ЧИТАТЕЛЬ» или выше по публичной ссылке;
- загрузка файлов в папку при наличии уровня доступа АВТОР или выше по публичной ссылке;
- переименование объектов внутри папки при наличии уровня доступа АВТОР или выше по публичной ссылке;
- перемещение объектов внутри папки при наличии уровня доступа РЕДАКТОР или выше по публичной ссылке;
- удаление объектов в папке при наличии уровня доступа РЕДАКТОР или выше по публичной ссылке.

На рисунке 13 представлена UML-диаграмма прецедентов неавторизованного пользователя.

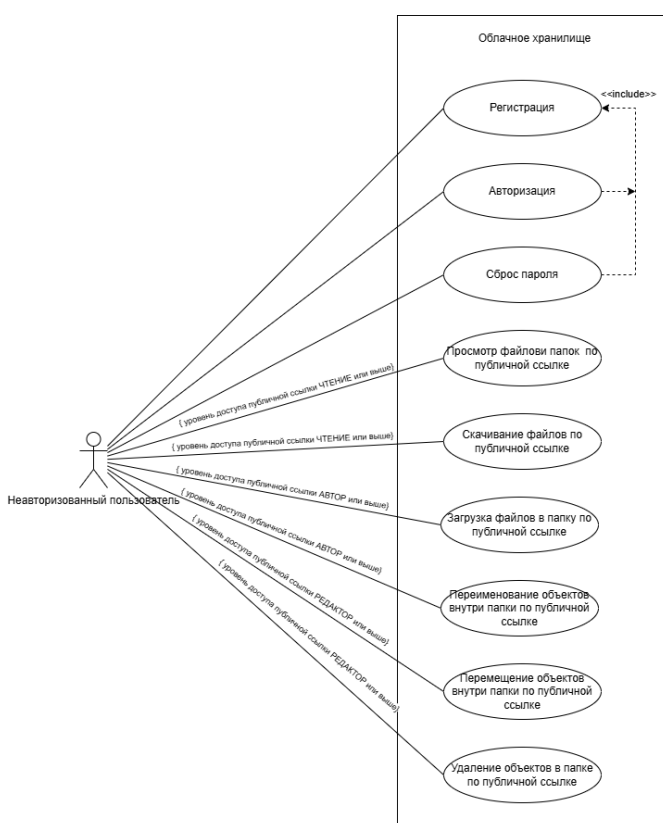


Рисунок 13 – UML-диаграмма прецедентов неавторизованного пользователя

4.3.2 Функционал авторизованного пользователя

Авторизованному пользователю доступен полный функционал приложения:

- смена пароля;
- выход из учетной записи;
- удаление учетной записи;
- редактирование личных данных;
- переименование файлов и папок;
- удаление файлов и папок;
- скачивание файлов и папок;
- загрузка файлов;
- перемещение файлов;
- просмотр содержимого корзины, восстановление удалённых объектов, их окончательное удаление и полная очистка корзины;
- управление доступом к файлам и папкам: выдача и отзыв прав другим пользователям с уровнями ЧИТАТЕЛЬ (просмотр и скачивание), АВТОР (загрузка файлов, создание папок, переименование), РЕДАКТОР (перемещение, удаление), ВЛАДЕЛЕЦ (все предыдущие права + управление правами доступа);
- создание и отзыв публичных ссылок на объекты; Уровни доступа: ЧИТАТЕЛЬ (просмотр и скачивание), АВТОР (загрузка файлов, создание папок, переименование), РЕДАКТОР (перемещение, удаление);
- система поиска: поиск по имени, полнотекстовый поиск, семантический поиск, а также гибридный режим;
- ведение AI-диалога с содержимым документа;
- задание вопросов AI-чату по файлу;
- просмотр истории чата с AI;
- очистка истории чата с AI;
- сортировка объектов.

На рисунке 14 представлена UML-диаграмма прецендентов авторизованного пользователя.

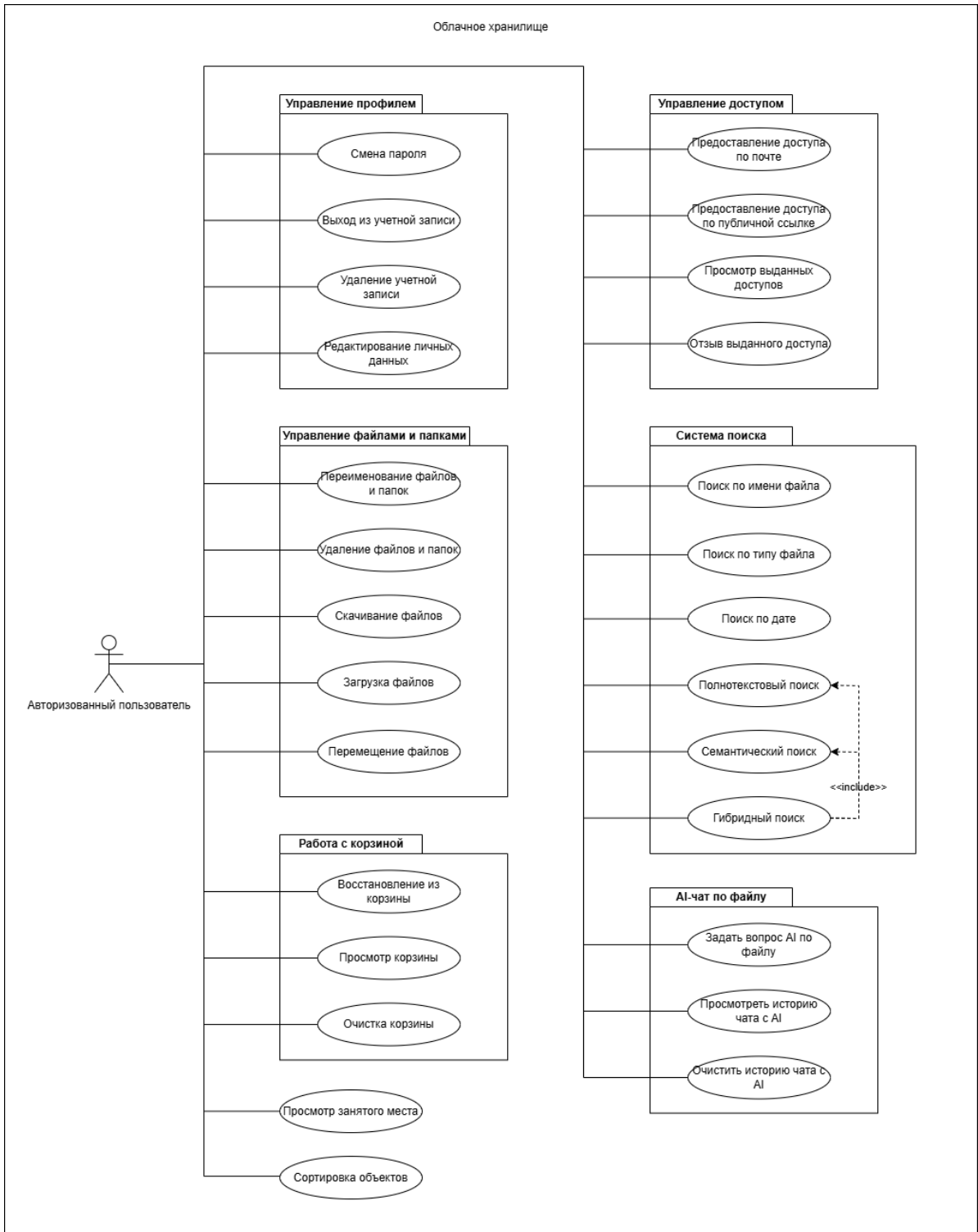


Рисунок 14 – UML-диаграмма прецендентов авторизованного пользователя

4.4 Описание основных компонентов

В данном подразделе рассматривается реализация основных компонентов системы и главных функций веб-приложения: управления доступом и системы поиска.

4.4.1 Основной сервис (backend)

Микросервис backend разработан на языке программирования Java и работает на порту 8080 и реализует следующие пользовательские сценарии: регистрация, авторизация, загрузка и скачивание файлов, управление правами доступа, профилем пользователя, корзиной и публичными ссылками.

Генерация и валидация JWT-токенов реализовано через компонент JwtUtil. Токен содержит почту, дату выдачи и истечения токена. Код этого компонента представлен в листинге 1.

Листинг 1. JwtUtil:

```
@Component
@Slf4j
public class JwtUtil {

    @Value("${app.jwt.secret}")
    private String secret;

    @Value("${app.jwt.expiration}")
    private Long expiration;

    private SecretKey getSigningKey() {
        byte[] keyBytes = Decoders.BASE64.decode(secret);
        return Keys.hmacShaKeyFor(keyBytes);
    }

    public String generateToken(UserDetails userDetails) {
        return generateToken(new HashMap<>(), userDetails);
    }

    public String generateToken(Map<String, Object> extraClaims,
        UserDetails userDetails) {
        return Jwts.builder()
            .claims(extraClaims)
            .subject(userDetails.getUsername())
            .issuedAt(new Date())
```

```

        .expiration(new Date(System.currentTimeMillis() +
expiration))
        .signWith(getSigningKey())
        .compact();
    }

    public boolean isValidToken(String token, UserDetails
userDetails) {
        try {
            final String username = extractUsername(token);
            return username.equals(userDetails.getUsername()) &&
!isTokenExpired(token);
        } catch (Exception e) {
            return false;
        }
    }

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T>
claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parser()
            .verifyWith(getSigningKey())
            .build()
            .parseSignedClaims(token)
            .getPayload();
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }
}

```

При каждом входящем запросе фильтр `JwtAuthenticationFilter` извлекает токен из заголовка `Authorization`, проверяет его и помещает данные пользователя в `SecurityContextHolder`. `JwtAuthenticationFilter` является фильтром класса `OncePerRequestFilter` из модуля `Spring Security`. Код этого фильтра представлен в листинге 2.

					ИГУ 1.090301.24.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг 2. JwtAuthenticationFilter:

```

@Component
@RequiredArgsConstructor
@Slf4j
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtUtil jwtUtil;
    private final UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(
        @NonNull HttpServletRequest request,
        @NonNull HttpServletResponse response,
        @NonNull FilterChain filterChain
    ) throws ServletException, IOException {

        final String authHeader =
request.getHeader("Authorization");

        final String queryToken = request.getParameter("token");

        String jwt = null;
        if (authHeader != null && authHeader.startsWith("Bearer "))
        {
            jwt = authHeader.substring(7);
        } else if (queryToken != null && !queryToken.isBlank()) {
            jwt = queryToken;
        }

        if (jwt == null) {
            filterChain.doFilter(request, response);
            return;
        }

        try {
            final String userEmail = jwtUtil.extractUsername(jwt);

            if (userEmail != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
                UserDetails userDetails =
userDetailsService.loadUserByUsername(userEmail);
                if (jwtUtil.isTokenValid(jwt, userDetails)) {
                    UsernamePasswordAuthenticationToken authToken =
                        new
UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
                    authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
                    SecurityContextHolder.getContext().setAuthentication(authToken);
                }
            }
        }
    }
}

```

```

        }
    }
    } catch (Exception e) {
        log.debug("JWT validation failed: {}", e.getMessage());
    }

    filterChain.doFilter(request, response);
}
}

```

Регистрация и авторизация пользователя реализована через сервис AuthService, который проверяет уникальность данных при регистрации, хеширует пароли и сохраняет пользователя. Код регистрации из AuthService приведен в листинге 3.

Листинг 2. register():

```

@Transactional
public String register(RegisterRequest request) {
    if (userRepository.existsByEmail(request.getEmail())) {
        throw new IllegalArgumentException("Email already
registered");
    }
    if (userRepository.existsByUsername(request.getUsername()))
{
        throw new IllegalArgumentException("Username already
taken");
    }

    String verificationToken = UUID.randomUUID().toString();

    User user = User.builder()
        .email(request.getEmail())
        .username(request.getUsername())

        .passwordHash(passwordEncoder.encode(request.getPassword()))
        .firstName(request.getFirstName())
        .lastName(request.getLastName())
        .isEmailVerified(false)
        .emailVerificationToken(verificationToken)
        .emailVerificationExpires(Instant.now().plus(24,
ChronoUnit.HOURS))
        .storageUsed(0L)
        .storageLimit(10737418240L)
        .isDeleted(false)
        .build();

    userRepository.save(user);
}

```

```

        String link = baseUrl + "/api/auth/verify-email?token=" +
verificationToken;
        rabbitTemplate.convertAndSend(
            RabbitMQConfig.EMAIL_EXCHANGE,
            RabbitMQConfig.EMAIL_ROUTING_KEY,
            new EmailMessage(user.getEmail(), "Подтверждение email",
                "email-verification", link, user.getUsername(),
null)
        );

        return "Registration successful. Please check your email to
verify your account.";
    }

```

4.4.2 Реализация управления доступом

Управление доступом реализовано в микросервисе backend. Существует четыре уровня прав: READ, CONTRIBUTE, WRITE и OWNER. Каждый уровень выше по привилегиям включает предыдущий. Сущность прав доступа и их иерархия представлена в листинге 4.

Листинг 4. AccessPermission:

```

@Entity
@Table(name = "access_permissions")
@Getter @Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class AccessPermission {

    public enum PermissionLevel {
        READ, CONTRIBUTE, WRITE, OWNER;

        public boolean includes(PermissionLevel other) {
            return this.ordinal() >= other.ordinal();
        }
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "object_id", nullable = false)
    private StorageObject object;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "grantee_id")

```

```

private User grantee;

@Enumerated(EnumType.STRING)
@Column(name = "permission_level", nullable = false)
private PermissionLevel permissionLevel;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "granted_by", nullable = false)
private User grantedBy;

@Column(name = "granted_at")
private Instant grantedAt = Instant.now();
}

```

Права на объект проверяются следующим образом: обходятся родительские папки вверх по иерархии для проверки ограничивающих прав, если владельцем корневой папки является сам пользователь, то возвращается уровень доступа OWNER, иначе проверяется прямой уровень доступа на объект и унаследованный. Код демонстрирующий определение уровня прав представлен в листинге 5.

Листинг 5. FileService.getEffectivePermission():

```

public PermissionLevel getEffectivePermission(User user,
StorageObject obj) {
    boolean isObjectOwner =
obj.getOwner().getId().equals(user.getId());

    PermissionLevel inheritedCap = null;
    StorageObject current = obj.getParent();
    int depth = 0;
    while (current != null && depth < 50) {
        if (!current.getOwner().getId().equals(user.getId())) {

            Optional<AccessPermission> perm =
permissionRepository.findByObjectAndGrantee(current, user);
            if (perm.isPresent()) {
                inheritedCap = perm.get().getPermissionLevel();
                break;
            }

        }

        current = current.getParent();
        depth++;
    }
}

```

```

        if (isObjectOwner) {
            return inheritedCap != null ? inheritedCap :
PermissionLevel.OWNER;
        }

        Optional<AccessPermission> directPerm =
permissionRepository.findByObjectAndGrantee(obj, user);
        if (directPerm.isPresent()) return
directPerm.get().getPermissionLevel();
        return inheritedCap;
    }

```

Выдача прав реализована через метод `FileService.grantPermission()`. Выдать права может только пользователь с уровнем прав OWNER. Код демонстрирующий выдачу прав представлен в листинге 6.

Листинг 6. `FileService.grantPermission()`:

```

@Transactional
    public void grantPermission(User granter, UUID objectId,
GrantPermissionRequest request) {
        StorageObject obj = getObjectOrThrow(objectId);
        checkPermission(granter, obj, PermissionLevel.OWNER);

        User grantee =
userRepository.findByEmail(request.getEmail())
                .orElseThrow(() -> new
ResourceNotFoundException("User not found: " + request.getEmail()));

        if (grantee.getId().equals(granter.getId())) {
            throw new IllegalArgumentException("Cannot grant
permission to yourself");
        }

        AccessPermission existing =
permissionRepository.findByObjectAndGrantee(obj,
grantee).orElse(null);
        if (existing != null) {
            existing.setPermissionLevel(request.getPermission());
            permissionRepository.save(existing);
        } else {
            AccessPermission perm = AccessPermission.builder()
                .object(obj)
                .grantee(grantee)
                .permissionLevel(request.getPermission())
                .grantedBy(granter)
                .build();
            permissionRepository.save(perm);
        }
    }

```

```
}
```

Отзыв прав реализован через метод `FileService.revokePermission()`. Отозвать права может только пользователь с уровнем прав OWNER. При отзыве прав удаляется запись из таблицы `access_permissions`. Код демонстрирующий отзыв прав представлен в листинге 7.

Листинг 7. `FileService.revokePermission()`:

```
@Transactional
public void revokePermission(User revoker, UUID objectUuid, Long
granteeId) {
    StorageObject obj = getObjectOrThrow(objectUuid);
    checkPermission(revoker, obj, PermissionLevel.OWNER);
    User grantee = userRepository.findById(granteeId)
        .orElseThrow(() -> new
ResourceNotFoundException("User not found"));
    permissionRepository.deleteByObjectAndGrantee(obj, grantee);
}
```

Публичные ссылки создаются через метод `createShareLink()`, который генерирует случайный UUID-токен, сохраняет уровень доступа и дату истечения ссылки. Для доступа по этой ссылке реализован метод `resolvePublicToken()`, который проверяет действительна ли ссылка. Эти методы реализованы в сервисе `ShareLinkService`. Их код представлен в листинге 8.

Листинг 8. `createShareLink()` и `resolvePublicToken()`:

```
@Transactional
public ShareLinkDto createShareLink(User user, UUID objectUuid,
ShareLinkRequest request) {
    StorageObject obj =
fileService.getObjectOrThrow(objectUuid);
    fileService.checkPermission(user, obj,
PermissionLevel.OWNER);

    PermissionLevel permission = request.getPermission() != null
? request.getPermission() : PermissionLevel.READ;
    if (permission == PermissionLevel.OWNER) {
        throw new IllegalArgumentException("Cannot create a
public share link with OWNER permission");
    }

    ShareLink link = ShareLink.builder()
        .token(UUID.randomUUID().toString())
```

```

        .object(obj)
        .createdBy(user)
        .permissionLevel(permission)
        .expiresAt(request.getExpiresAt())
        .isActive(true)
        .build();

    shareLinkRepository.save(link);
    return toDto(link);
}

public ShareLink resolvePublicToken(String token) {
    ShareLink link =
    shareLinkRepository.findByTokenAndIsActiveTrue(token)
        .orElseThrow(() -> new
ResourceNotFoundException("Share link not found or inactive"));
    if (link.getExpiresAt() != null &&
link.getExpiresAt().isBefore(Instant.now())) {
        throw new ResourceNotFoundException("Share link
expired");
    }
    return link;
}

```

4.4.3 Реализация системы поиска

Поддерживаются следующие режимы поиска: по названию, полнотекстовый, семантический и гибридный, совмещающий полнотекстовый и семантический поиск через алгоритм RRF. Запрос поиска маршрутизируется в backend сервис. Маршрутизация поиска представлена в листинге 9.

Листинг 9. Маршрутизация поиска в FileService.search():

```

if ("semantic".equals(type)) {
    if (request.getQuery() == null ||
request.getQuery().isBlank()) return List.of();

    final Long finalFolderId = folderId;
    if (isShared) {
        return semanticOrHybridShared(user,
request.getQuery(), true)
            .stream()
            .filter(dto -> finalFolderId == null ||
isInSubtreeByUuid(dto.getUuid(), finalFolderId))
            .toList();
    }

    return searchServiceClient.semanticSearch(user.getId(),
request.getQuery())

```

```

        .stream()
        .map(r -> r.getUuid() != null
            ?
storageObjectRepository.findByUuidAndIsDeletedFalse(r.getUuid()).orE
lse(null)
                : null)
        .filter(Objects::nonNull)
        .filter(o ->
o.getOwner().getId().equals(user.getId()))
        .filter(o -> finalFolderId == null ||
isInSubtree(o, finalFolderId))
        .map(o -> toDto(o, user))
        .toList();
    }

    if ("name".equals(type)) {
        String q = (request.getQuery() != null &&
!request.getQuery().isBlank())
            ? request.getQuery() : null;
        if (q == null) return List.of();
        List<StorageObject> nameResults;
        if (isShared && folderId != null) {
            nameResults =
storageObjectRepository.searchByNameSharedInFolder(folderId, q,
request.getMimeType(), dateFrom, dateTo);
        } else if (isShared) {
            nameResults =
storageObjectRepository.searchByNameShared(user.getId(), q,
request.getMimeType(), dateFrom, dateTo);
        } else if (folderId != null) {
            nameResults =
storageObjectRepository.searchByNameInFolder(user.getId(), folderId,
q, request.getMimeType(), dateFrom, dateTo);
        } else {
            nameResults =
storageObjectRepository.searchByName(user.getId(), q,
request.getMimeType(), dateFrom, dateTo);
        }
        return nameResults.stream()
            .sorted(buildComparator(request.getSortBy(),
request.getSortDir()))
            .map(o -> toDto(o, user))
            .toList();
    }

    if ("all".equals(type)) {
        if (request.getQuery() == null ||
request.getQuery().isBlank()) return List.of();

        final Long finalFolderId = folderId;
        if (isShared) {

```

```

        return semanticOrHybridShared(user,
request.getQuery(), false)
            .stream()
            .filter(dto -> finalFolderId == null ||
isInSubtreeByUuid(dto.getUuid(), finalFolderId))
            .toList();
    }

    return searchServiceClient.hybridSearch(user.getId(),
request.getQuery())
        .stream()
        .map(r -> r.getUuid() != null
            ?
storageObjectRepository.findByUuidAndIsDeletedFalse(r.getUuid()).orE
lse(null)
                : null)
        .filter(Objects::nonNull)
        .filter(o ->
o.getOwner().getId().equals(user.getId()))
        .filter(o -> finalFolderId == null ||
isInSubtree(o, finalFolderId))
        .map(o -> toDto(o, user))
        .toList();
    }

```

Поиск по имени и полнотекстовый выполняется полностью силами PostgreSQL. Поиск по имени выполняется с помощью ILIKE, а полнотекстовый с помощью поля content_tsv типа TSVECTOR. Код реализации этих видов поиска представлен в листинге 10.

Листинг 10. Полнотекстовый и поиск по имени:

```

@Query(value = """
    SELECT * FROM storage_objects
    WHERE owner_id = :ownerId
    AND is_deleted = false
    AND name ILIKE '%' || :query || '%'
    AND (CAST(:mimeType AS TEXT) IS NULL OR mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
    AND (CAST(:dateFrom AS TEXT) IS NULL OR updated_at >=
CAST(:dateFrom AS TIMESTAMP))
    AND (CAST(:dateTo AS TEXT) IS NULL OR updated_at <=
CAST(:dateTo AS TIMESTAMP))
    ORDER BY updated_at DESC
    LIMIT 100
    """, nativeQuery = true)
List<StorageObject> searchByName(
    @Param("ownerId") Long ownerId,
    @Param("query") String query,

```

```

        @Param("mimeType") String mimeType,
        @Param("dateFrom") String dateFrom,
        @Param("dateTo") String dateTo
    );

    @Query(value = """
        SELECT * FROM storage_objects
        WHERE owner_id = :ownerId
        AND is_deleted = false
        AND (
            (CAST(:query AS TEXT) IS NULL OR name ILIKE '%' ||
CAST(:query AS TEXT) || '%')
            OR (CAST(:query AS TEXT) IS NOT NULL AND CAST(:query AS
TEXT) <> ' '
                AND content_tsv @@ plainto_tsquery('russian',
CAST(:query AS TEXT)))
        )
        AND (CAST(:mimeType AS TEXT) IS NULL OR mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
        AND (CAST(:dateFrom AS TEXT) IS NULL OR updated_at >=
CAST(:dateFrom AS TIMESTAMP))
        AND (CAST(:dateTo AS TEXT) IS NULL OR updated_at <=
CAST(:dateTo AS TIMESTAMP))
        ORDER BY updated_at DESC
        LIMIT 100
    """, nativeQuery = true)
    List<StorageObject> advancedSearch(
        @Param("ownerId") Long ownerId,
        @Param("query") String query,
        @Param("mimeType") String mimeType,
        @Param("dateFrom") String dateFrom,
        @Param("dateTo") String dateTo
    );

```

При семантическом и гибридном поиске backend микросервис отправляет запрос на эти виды поиска в search-service. При семантическом поиске происходит первоначальный поиск релевантных файлов через векторизацию запроса и поиск ближайших соседей моделью bge-m3, дальше релевантный список отправляется на переранжирование в микросервис reranker-service с моделью bge-reranker-v2-m3. Код для оценки файла представлен в листинге 11.

Листинг 11. SemanticSearchService.searchByChunks():

```

private List<SemanticSearchResult> searchByChunks(Long ownerId,
String query,
                                                    float[]
queryEmbedding, List<ChunkRow> chunks) {

```

```

Map<Long, Double> maxScoreByObject = new HashMap<>();
Map<Long, String> bestChunkTextByObject = new HashMap<>();

for (ChunkRow chunk : chunks) {
    double score =
embeddingService.cosineSimilarity(queryEmbedding,
chunk.embedding());
    Double prev = maxScoreByObject.get(chunk.objectId());
    if (prev == null || score > prev) {
        maxScoreByObject.put(chunk.objectId(), score);

        bestChunkTextByObject.put(chunk.objectId(),
chunk.chunkText());
    }
}

log.debug("Scored {} chunks → {} distinct objects for owner
{}", chunks.size(),
        maxScoreByObject.size(), ownerId);

List<EmbeddingRow> allRows =
embeddingRepository.findEmbeddingsByOwner(ownerId);
Map<Long, EmbeddingRow> metaById = new HashMap<>();
for (EmbeddingRow row : allRows) metaById.put(row.id(),
row);

List<SemanticSearchResult> sorted =
maxScoreByObject.entrySet().stream()
    .map(e -> {
        EmbeddingRow meta = metaById.get(e.getKey());
        String name      = meta != null ? meta.name()
: "unknown";
        String mime      = meta != null ?
meta.mimeType() : null;
        UUID uuid        = meta != null ? meta.uuid()
: null;
        String chunkText =
bestChunkTextByObject.get(e.getKey());
        return new SemanticSearchResult(e.getKey(),
uuid, name, mime, e.getValue(), chunkText);
    })

.sorted(Comparator.comparingDouble(SemanticSearchResult::getScore).r
everse())

    .toList();

return applyFilters(sorted, query);
}

```

При гибридном поиске результаты полнотекстового и семантического поиска без переранжирования объединяются алгоритмом RRF, а потом общий список после RRF подается в микросервис переранжирования. Алгоритм RRF состоит в следующем: после полнотекстового и семантического поиска без переранжирования получаем 2 списка и формируем офенку для каждого списка и файла формулой $1/(k + \text{rank})$, где $k = 60$. Код гибридного поиска представлен в листинге 12.

Листинг 12. HybridSearchService.search():

```
public List<SemanticSearchResult> search(Long ownerId, String query)
{
    if (query == null || query.isBlank()) return List.of();

    List<Long> bm25Ids = bm25Search(ownerId, query);
    log.debug("BM25 returned {} results", bm25Ids.size());

    List<SemanticSearchResult> semanticResults =
        semanticSearchService.searchWithoutRerank(ownerId,
query);
    log.debug("Semantic returned {} results (no rerank)",
semanticResults.size());

    Map<Long, Double> rrfScores = new HashMap<>();

    for (int i = 0; i < bm25Ids.size(); i++) {
        long id = bm25Ids.get(i);
        rrfScores.merge(id, 1.0 / (RRF_K + i + 1), Double::sum);
    }

    for (int i = 0; i < semanticResults.size(); i++) {
        long id = semanticResults.get(i).getId();
        rrfScores.merge(id, 1.0 / (RRF_K + i + 1), Double::sum);
    }

    if (rrfScores.isEmpty()) return List.of();

    Map<Long, SemanticSearchResult> metaMap = new HashMap<>();
    for (SemanticSearchResult r : semanticResults)
metaMap.put(r.getId(), r);

    Set<Long> missingIds = new HashSet<>(rrfScores.keySet());
    missingIds.removeAll(metaMap.keySet());
    if (!missingIds.isEmpty()) {
        loadMetadata(missingIds).forEach(r ->
metaMap.put(r.getId(), r));
    }
}
```

```

    }

    List<SemanticSearchResult> merged =
rrfScores.entrySet().stream()
                .sorted(Map.Entry.<Long,
Double>comparingByValue().reversed())
                .limit(MAX_FINAL_RESULTS)
                .map(e -> {
                    SemanticSearchResult meta =
metaMap.get(e.getKey());
                    if (meta == null) return null;
                    return new SemanticSearchResult(
                        meta.getId(), meta.getUuid(),
                        meta.getName(), meta.getMimeType(),
                        e.getValue(),
                        meta.getChunkText());
                })
                .filter(Objects::nonNull)
                .toList();

    log.debug("Hybrid RRF returned {} merged results for query
'{}'", merged.size(), query);

    List<SemanticSearchResult> reranked =
rerankService.rerank(query, merged);
    log.debug("After rerank: {} results for query '{}'",
reranked.size(), query);
    return reranked;
}

```

4.4.5 Сервис чата с документами (chat-service)

Серви chat-service работает на порту 8084 и написан на языке программирования Java. В основе сервиса лежит RAG архитектура: извлечение релевантных материалов и генерация на ее основе с помощью LLM ответа пользователю.

При поступлении в chat-service вопроса пользователя микросервис обращается к search-service за наиболее релевантными чанками для контекста файла. После chat-service генерирует промт на основе вопроса и контекста и отправляет этот промт в Ollama для генерации ответа пользователю на основе модели qwen2.5:3b. Модель в свою очередь отвечает через SSE-стриминг и пользователь в реальном времени видит как AI-ассистент ему отвечает. Вопрос пользователя и ответ ассистента записываются в базу данных. Листинг 8

демонстрирует ключевой фрагмент метода. RAG-конвейер со стримингом ответа через SSE представлен в листинге 13.

Листинг 13. FileChatService:

```
@Service
@RequiredArgsConstructor
@Slf4j
public class FileChatService {

    private static final int RAG_TOP_K = 6;

    public static final long SSE_TIMEOUT_MS = 180_000L;

    private final FileAccessChecker    accessChecker;
    private final SearchServiceClient  searchClient;
    private final OllamaClient         ollamaClient;
    private final ChatMessageRepository messageRepo;

    public List<MessageResponse> getHistory(UUID fileUuid, Long
userId) {
        FileInfo file = accessChecker.getFileOrThrow(fileUuid,
userId);
        return messageRepo.findByObjectAndUser(file.getId(),
userId);
    }

    public void clearHistory(UUID fileUuid, Long userId) {
        FileInfo file = accessChecker.getFileOrThrow(fileUuid,
userId);
        int deleted =
messageRepo.deleteByObjectAndUser(file.getId(), userId);
        log.debug("Cleared {} chat messages for object={} user={}",
deleted, file.getId(), userId);
    }

    public SseEmitter streamChat(UUID fileUuid, Long userId, String
question) {
        SseEmitter emitter = new SseEmitter(SSE_TIMEOUT_MS);
        FileInfo file = accessChecker.getFileOrThrow(fileUuid,
userId);

        CompletableFuture.runAsync(() -> {
            try {

                messageRepo.save(file.getId(), userId, "user",
question);

                List<String> chunks = searchClient.getChatContext(
```

```

        file.getId(), file.getOwnerId(), question,
RAG_TOP_K);

        String context = chunks.isEmpty()
            ? "(контекст не найден - файл ещё не
проиндексирован или не содержит текста)"
            : String.join("\n\n---\n\n", chunks);

        String systemPrompt = ""
            Ты - умный ассистент для работы с
документами. \

            Тебе предоставлены фрагменты из файла «%s».
Правила:
1. Отвечай только на основе предоставленного
контекста.
2. Если ответа нет в контексте - честно
скажи об этом, \
не придумывай информацию.
3. Отвечай на том же языке, что и вопрос
пользователя.
4. Цитируй конкретные места из текста, если
это уместно.

            Контекст из файла:
            %s
            """.formatted(file.getName(), context);

        StringBuilder fullAnswer = new StringBuilder();

        ollamaClient.streamChat(systemPrompt, question,
token -> {
            fullAnswer.append(token);
            try {

                String sseToken = token.replace("\n",
"\n\n").replace(" ", "\\s");

                emitter.send(SseEmitter.event().name("token").data(sseToken));
            } catch (IOException e) {
                throw new RuntimeException("SSE send
failed", e);
            }
        });

        messageRepo.save(file.getId(), userId, "assistant",
fullAnswer.toString());

        emitter.send(SseEmitter.event().name("done").data("[DONE]"));
        emitter.complete();

    } catch (Exception e) {

```

```

        log.error("Chat stream error for file={} user={}:
{}", fileUuid, userId, e.getMessage(), e);
        try {
            emitter.send(SseEmitter.event()
                .name("error")
                .data("Ошибка генерации ответа: " +
e.getMessage()));
        } catch (IOException ignored) {}
        emitter.completeWithError(e);
    }
});

return emitter;
}
}

```

4.4.6 Сервис отправки электронной почты (email-service)

Микросервис email-service написан на языке программирования Java и работает на порту 8083. Сервис отвечает за отправку писем пользователям и подписан на очередь RabbitMQ email.queue. EmailConsumer получает сообщение EmailMessage и отправляет письмо через JavaMailSender. Код представлен в листинге 14.

Листинг 14. EmailConsumer:

```

@RabbitListener(queues = RabbitConfig.EMAIL_QUEUE)
public void handleEmailMessage(EmailMessage message) {
    try {
        Context ctx = new Context();
        ctx.setVariable("username", message.getUsername());
        ctx.setVariable("link", message.getLink());
        ctx.setVariable("body", message.getBody());

        String html;
        try {
            html = templateEngine.process(message.getTemplate(),
ctx);
        } catch (Exception e) {
            log.warn("Template '{}' not found, using fallback",
message.getTemplate());
            html = buildFallbackHtml(message);
        }

        MimeMessage mime = mailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(mime,
true, "UTF-8");
        helper.setFrom("noreply@cloudstorage.app");
    }
}

```

```

        helper.setTo(message.getTo());
        helper.setSubject(message.getSubject());
        helper.setText(html, true);
        mailSender.send(mime);

        log.info("Email sent to {}: {}", message.getTo(),
message.getSubject());
    } catch (Exception e) {
        log.error("Failed to send email to {}: {}",
message.getTo(), e.getMessage());
    }
}

```

4.4.7 Сервис переранжирования (reranker-service)

Микросервис переранжирования написан на языке программирования Python и работает на порту 8082. Сервис выполняет переранжирование на основе сопоставления вопроса пользователя и теста с помощью модели bge-reranker-v2-m3. В сервис переранжирования данные передаются через запрос от микросервиса search-service с релевантным списком файлов после ранжирования через векторы моделью bge-m3. С помощью этого микросервиса переранжирования достигается высокая точность при поиске. Код запроса и ответа из RerankService представлен в листинге 15.

Листинг 15. RerankService:

```

@Data
static class RerankRequest {
    private final String query;
    private final List<String> documents;
    private final Integer top_n;
}

@Data
static class RerankResponse {
    private List<RerankResult> results;
    private String model;
    private int ms;
}

@Data
static class RerankResult {
    private int index;
    private double score;
}

```

```

public List<SemanticSearchResult> rerank(String query,
                                         List<SemanticSearchResult>
candidates) {
    List<String> documents = candidates.stream()
        .map(this::resolveDocText)
        .collect(Collectors.toList());

    RerankRequest request = new RerankRequest(query, documents,
topN);
    RerankResponse response = restTemplate.postForObject(
        rerankerUrl + "/rerank", request, RerankResponse.class);

    return response.getResults().stream()
        .map(r -> {
            SemanticSearchResult orig =
candidates.get(r.getIndex());
            return new SemanticSearchResult(
                orig.getId(), orig.getUuid(), orig.getName(),
                orig.getMimeType(), r.getScore(),
orig.getChunkText());
        })
        .collect(Collectors.toList());
}

```

Код переранжирования представлен в 16 листинге.

Листинг 16. Методы rerank и _score_pairs:

```

@app.post("/rerank", response_model=RerankResponse)
def rerank(req: RerankRequest):
    if model is None or tokenizer is None:
        raise HTTPException(status_code=503, detail="Model not
loaded yet")

    if not req.documents:
        return RerankResponse(results=[], model=MODEL_NAME, ms=0)

    t0 = time.time()

    pairs = [[req.query, doc] for doc in req.documents]

    scores = _score_pairs(pairs)

    def sigmoid(x: float) -> float:
        return 1.0 / (1.0 + math.exp(-x))

    results = [
        RerankResult(index=i, score=sigmoid(s))
        for i, s in enumerate(scores)
    ]
    results.sort(key=lambda r: r.score, reverse=True)

```

```

    if req.top_n is not None:
        results = results[: req.top_n]

    elapsed_ms = int((time.time() - t0) * 1000)
    log.info(
        "Reranked %d docs for query %r → top score=%.4f [%dms]",
        len(req.documents), req.query[:60], results[0].score if
        results else 0, elapsed_ms,
    )

    return RerankResponse(results=results, model=MODEL_NAME,
ms=elapsed_ms)

def _score_pairs(pairs: list[list[str]]) -> list[float]:
    all_scores: list[float] = []

    for i in range(0, len(pairs), BATCH_SIZE):
        batch = pairs[i : i + BATCH_SIZE]

        inputs = tokenizer(
            batch,
            padding=True,
            truncation=True,
            max_length=MAX_LENGTH,
            return_tensors="pt",
        ).to(device)

        with torch.no_grad():
            logits = model(**inputs).logits

        if logits.dim() == 2:
            logits = logits.squeeze(-1)

        all_scores.extend(logits.cpu().tolist())

    return all_scores

```

4.4.8 Клиентское приложение (frontend)

Фронтенд веб-приложения написан на JavaScript и реализовано как одностраничное приложение SPA через React. Управление аутентификацией происходит через AuthContext: контекст проверяет наличие токена в localStorage и загружает профиль пользователя. С помощью функций login и logout управляются токены и состояния пользователя. Код реализации AuthContext представлен в листинге 17.

Листинг 17. AuthContext:

```
export function AuthProvider({ children }) {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  const loadUser = useCallback(async () => {
    try {
      const res = await userApi.getProfile();
      setUser(res.data.data);
    } catch {
      setUser(null);
    } finally {
      setLoading(false);
    }
  }, []);

  useEffect(() => {
    if (localStorage.getItem('accessToken')) loadUser();
    else setLoading(false);
  }, [loadUser]);

  const login = async (email, password) => {
    const res = await authApi.login({ email, password });
    const { accessToken, refreshToken, user: u } = res.data.data;
    localStorage.setItem('accessToken', accessToken);
    localStorage.setItem('refreshToken', refreshToken);
    setUser(u);
    return u;
  };

  const logout = async () => {
    try {
      const rt = localStorage.getItem('refreshToken');
      if (rt) await authApi.logout(rt);
    } finally {
      localStorage.clear();
      setUser(null);
    }
  };

  const refreshUser = async () => {
    await loadUser();
  };

  return (
    <AuthContext.Provider value={{ user, loading, login, logout,
refreshUser }}>
      {children}
    </AuthContext.Provider>
  );
}
```

```
}
```

Благодаря модулю `api.js` создается экземпляр `Axios` со ссылкой `/api`. Перехватчик запросов автоматически добавляет JWT-токен из `localStorage` в каждый запрос. Перехватчик ответа обрабатывает каждый ответ и при ответа 401 и наличии токена обновления выполняет обмен на `access-token` и повторяет запрос, а при отсутствии токена обновления или его истечения перенаправляет на страницу авторизации. Код этого модуля представлен в листинге 18.

Листинг 18. `api.js`:

```
const api = axios.create({
  baseURL: '/api',
  timeout: 60000,
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem('accessToken');
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});

api.interceptors.response.use(
  (res) => res,
  async (error) => {
    const original = error.config;
    if (error.response?.status === 401 && !original._retry) {
      original._retry = true;
      const refreshToken = localStorage.getItem('refreshToken');
      if (refreshToken) {
        try {
          const res = await axios.post('/api/auth/refresh', {
            refreshToken
          });
          const { accessToken } = res.data.data;
          localStorage.setItem('accessToken', accessToken);
          original.headers.Authorization = `Bearer ${accessToken}`;
          return api(original);
        } catch {
          localStorage.clear();
          window.location.href = '/login';
        }
      } else {
        window.location.href = '/login';
      }
    }
    return Promise.reject(error);
  }
);
```

);

Так как Axios не поддерживает SSE-стриминг, то для функций AI-чата было использовано EventSource. Код демонстрируется в листинге 19.

Листинг 19. chatApi

```
export const chatApi = {
  getHistory: (uuid) => api.get(`/files/${uuid}/chat/history`),
  clearHistory: (uuid) => api.delete(`/files/${uuid}/chat/history`),

  openStream: (uuid, question) => {
    const token = localStorage.getItem('accessToken');
    const params = new URLSearchParams({ q: question, token });
    return new
EventSource(`/api/files/${uuid}/chat/stream?${params}`);
  },
};
```

4.5 Основные алгоритмы взаимодействия клиента и сервера

В данном подразделе подробно описывается алгоритм взаимодействия между клиентской частью и микросервисами.

4.5.1 Алгоритм аутентификации и авторизации запросов

Алгоритм первичной аутентификации представлен в таблице 10.

Таблица 10 – Алгоритм первичной аутентификации

Участник	Описание шага
Клиент	Пользователь вводит email и пароль в LoginPage, вызывается функция login() из AuthContext
Клиент	Запрос /api/auth/login с почтой и паролем
backend	AuthenticationManager валидирует пароль через BCrypt; при неудаче возвращает 401
backend	JwtUtil генерирует access-токен и RefreshToken. Сохраняет refresh-токен в БД
backend	Возвращает клиенту accessToken, refreshToken, user с кодом 200
Клиент	AuthContext устанавливает user в состояние React и сохраняет токены в localStorage

Участник	Описание шага
Клиент	Пользователь перенаправляется на /files

Алгоритм автоматического обновления после истечения access-токена представлен в таблице 11.

Таблица 11 – Алгоритм автоматического обновления access-токена

Участник	Описание шага
Клиент	Axios-перехватчик добавляет токен к Authorization в запросе.
backend	JwtAuthenticationFilter обнаруживает истёкший токен.
backend	Возвращает 401 Unauthorized
Клиент	Response-перехватчик перехватывает 401, флаг _retry = false
Клиент	Отправляется запрос POST /api/auth/refresh с refreshToken
backend	RefreshTokenRepository находит токен и проверяет срок действия
backend	JwtUtil создает новый access-токен и возвращает accessToken, refreshToken и user
Клиент	Перехватчик сохраняет новый accessToken в localStorage
Клиент	Исходный запрос повторяется
backend	Запрос обрабатывается успешно

4.5.2 Алгоритм загрузки и индексации файла

Алгоритм загрузки и индексации файла представлен в таблице 12.

Таблица 12 – Алгоритм загрузки и индексации файла

Участник	Описание шага
Клиент	Пользователь загружает файл, вызывается fileApi.upload()
Клиент	Отправляется запрос POST /api/files/upload
backendя	FileService проверяет доступно ли пользователю свободное место для файла
backend	При parentUuid проверяется также право CONTRIBUTE на родительскую папку

Участник	Описание шага
backend	Файл записывается на диск в /storage/{userId}/{filename}
backend	Создаётся запись storage_objects в БД и обновляется storage_used пользователя
backend	Публикуется событие в RabbitMQ на индексацию файла
backend	Возвращает StorageObjectDto клиенту для отображения файла пользователю
search-service	IndexingConsumer получает сообщение из очереди search.index.queue
search-service	Получает сообщение из очереди RabbitMQ и с помощью Apache Tika извлекает текст. После извлечения текста обновляется content_tsv в базе данных
search-service	Генерируется эмбединг на файл через модель bge-m3 в Ollama
search-service	ChunkingService делит текст на чанки
search-service	Для каждого чанка создается эмбединг через модель bge-m3 в Ollama и сохраняется в базу данных

4.5.3 Алгоритм гибридного поиска файлов

Алгоритм гибридного поиска представлен в таблице 13.

Таблица 13 – Алгоритм гибридного поиска

Участник	Описание шага
Клиент	Пользователь вводит запрос
Клиент	Отправляется запрос POST /api/files/search с фильтрами в теле.
backend	FileService определяет тип поиска как гибридный и вызывает SearchServiceClient.hybridSearch()
backend	SearchServiceClient выполняет запрос GET /internal/search/hybrid?ownerId=...&q=...
search-service	HybridSearchService запускает параллельно полнотестовый поиск и семантический поиск без переранжирования
search-service	Выполнение полнотестового поиска
search-service	Выполнение семантического поиска через векторизацию вопроса
search-service	Объединение двух результатов поиска по RRF алгоритму

Участник	Описание шага
search-service	Релевантный список передаётся в RerankService.rerank()
reranker-service	bge-reranker-v2-m3 переранжирует результаты
search-service	Возвращает список после переранжирования
backend	FileService отправляет финальный список клиенту
Клиент	Отображение результатов поиска

4.5.4 Алгоритм AI-чата

Алгоритм AI-чата с файлом представлен в таблице 14.

Таблица 14 – Алгоритм AI-чата с документом

Участник	Описание шага
Клиент	Пользователь вводит вопрос в чат; Создаётся EventSource с /api/files/{uuid}/chat/stream?q=...&token=...
Клиент	Браузер устанавливает SSE-соединение
chat-service	JwtAuthenticationFilter извлекает JWT из запроса
chat-service	FileAccessChecker проверяет права
chat-service	Вопрос пользователя сохраняется в базу данных
chat-service	SearchServiceClient вызывает запрос контекста файла через GET /internal/search/chat/context?objectId=...&ownerId=...&q=...&topK=6
search-service	Отправляет релевантные чанки chat-service
chat-service	Создает системный промпт с полученным контекстом
chat-service	Открывает HTTP-соединение к Ollama /api/chat
Ollama	Модель qwen2.5:3b генерирует токены
chat-service	Каждый токен отправляется клиенту через SSE-поток
Клиент	Клиент получает текст
chat-service	Сохраняет ответ ассистента в базе данных

4.5.5 Алгоритм управления правами доступа

Алгоритм управления правами доступа представлен в таблице 15.

Таблица 15 – Алгоритм выдачи прав доступа другому пользователю

Участник	Описание шага
Клиент	Пользователь выдает права на объект
Клиент	Отправляется запрос POST /api/files/{uuid}/permissions
backend	Проверяется что пользователь выдающий права имеет уровень прав «ВЛАДЕЛЕЦ»
backend	UserRepository.findByEmail() находит получателя
backend	Если запись уже существует обновляются права, иначе создается новая запись с правами доступа
backend	Сохраняет запись в базу данных
backend	Возвращает ответ 200
Клиент	Отображение выданного права доступа

4.5.6 Алгоритм создания и использования публичных ссылок

Алгоритм создания и использования публичных ссылок приведен в таблице 16.

Таблица 16 – Алгоритм доступа по публичной ссылке

Участник	Описание шага
Владелец	Пользователь с правами «ВЛАДЕЛЕЦ» создает публичную ссылку, отправляется запрос POST /api/share/{uuid}/links
backend	ShareLinkService генерирует UUID-токен и сохраняет публичную ссылку
Получатель	Открывает публичную ссылку
Клиент	Отправляется запрос GET /api/share/public/{token}
backend	Находит ссылку и проверяет не истекла ли она.
backend	Возвращает метаданные объекта и permissionLevel ссылки
Клиент	Отображаются данные по ссылке

4.5.7 Алгоритм работы корзины

Алгоритм работы корзины представлено в таблице 17.

Таблица 17 – Алгоритм работы корзины

Участник	Описание шага
Клиент	Пользователь удаляет файл
Клиент	Отправляется запрос DELETE /api/files/{uuid}
backend	FileService.deleteToTrash() проверяет права доступа «РЕДАКТОР» или выше
backend	markDeletedRecursive(): рекурсивно помечает объект удаленным
backend	Уменьшается объем доступного пользователю дискового пространства
backend	Создаётся объект корзины
backend	Публикует запрос на очистку чанков и content_tsv
search-service	Очищает content_tsv и чанки из базы данных
Клиент	У пользователя пропадает объект из «Мои файлы» или «Общий доступ» и появляется в корзине
backend	TrashCleanupService ежедневно удаляет файлы с корзины, лежащие там дольше 30 дней

5 Тестирование приложения

Тестирование – это процесс проверки соответствия фактического и ожидаемого поведения приложения.

Одним из видов тестирования является smoke-тестирование. Smoke-тестирование – тестирование основных и наиболее критичных функций приложения.

Для проведения смоук тестирования были разработаны тест-кейсы, представленные в таблицах 5.1-5.24, охватывающие основной функционал приложения. Ниже приведены тест-кейсы и полученные результаты.

Таблица 5.1 – Тест-кейс № 1: Регистрация новой учетной записи

Параметр	Описание
Предусловие	Пользователь не зарегистрирован
Шаги выполнения	Перейти в форму регистрации Заполнить и отправить форму регистрации Подтвердить почту в письме
Ожидаемый результат	Отображение информации об успешной регистрации
Фактический результат	Пользователь успешно зарегистрировался

Фактический результат тест-кейса представлен на рисунке 15.

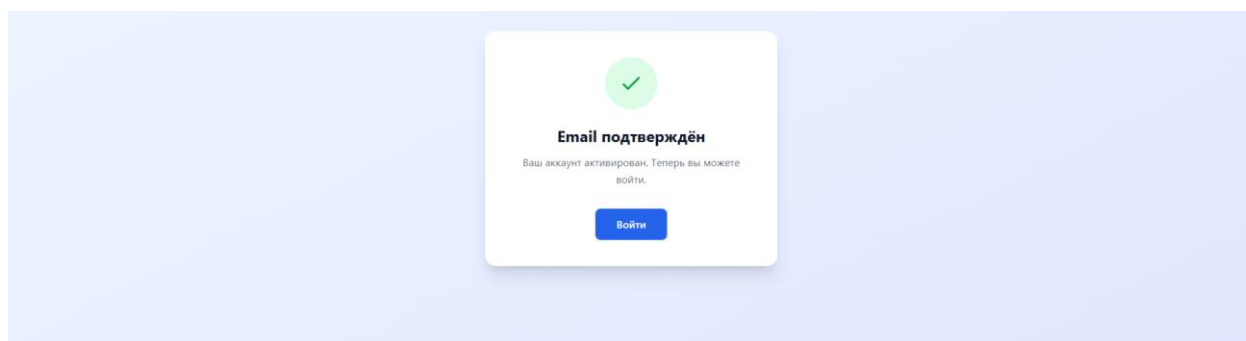


Рисунок 15 – Успешная регистрация аккаунта

Таблица 5.2 – Тест-кейс № 2: Авторизация учетной записи

Параметр	Описание
Предусловие	Пользователь зарегистрирован, но не авторизован
Шаги выполнения	Перейти в форму авторизации Ввести данные пользователя Нажать кнопку входа
Ожидаемый результат	Пользователь успешно вошел в систему
Фактический результат	Пользователь успешно вошел в систему

Фактический результат тест-кейса представлен на рисунке 16.

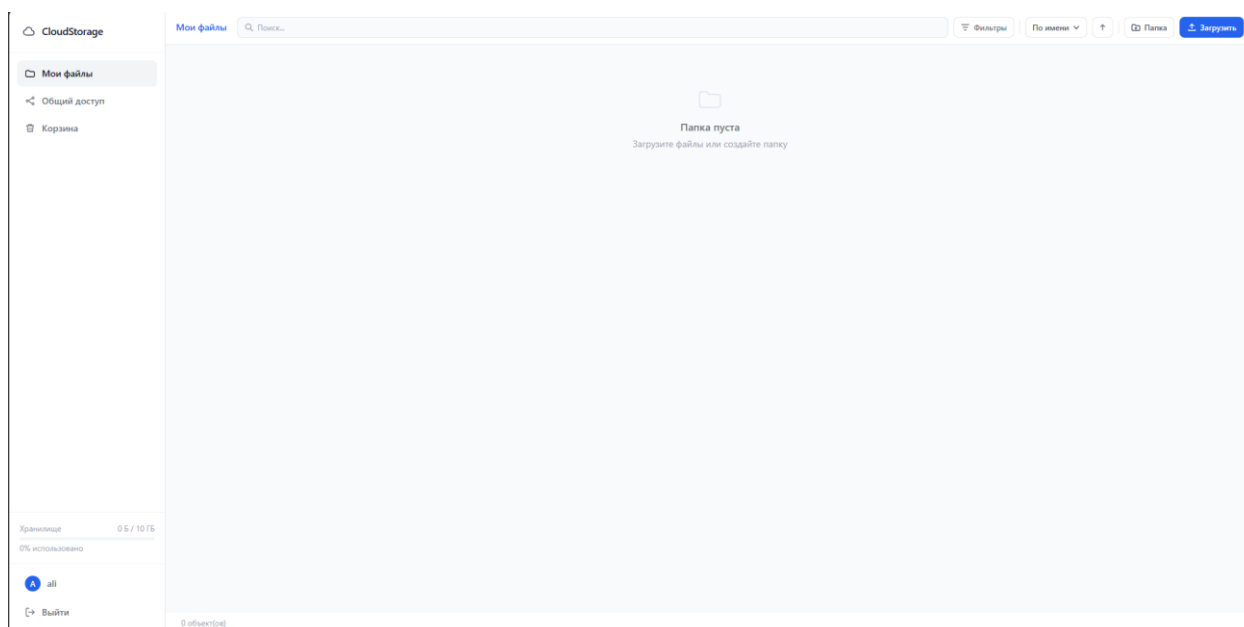


Рисунок 16 – Успешная авторизация аккаунта

Таблица 5.3 – Тест-кейс № 3: Восстановление пароля

Параметр	Описание
Предусловие	Пользователь авторизован
Шаги выполнения	Нажать на кнопку «Забыли пароль?» Ввести почту для восстановления пароля

Параметр	Описание
	Подтвердить сброс пароля в письме из почты Ввести новый пароль Выполнить вход по новому паролю
Ожидаемый результат	Появление уведомления об успешном изменении пароля. Пользователь успешно авторизовался под новым паролем
Фактический результат	Появление уведомления об успешном изменении пароля. Пользователь успешно авторизовался под новым паролем

Фактический результат тест-кейса представлен на рисунке 17 и 18.

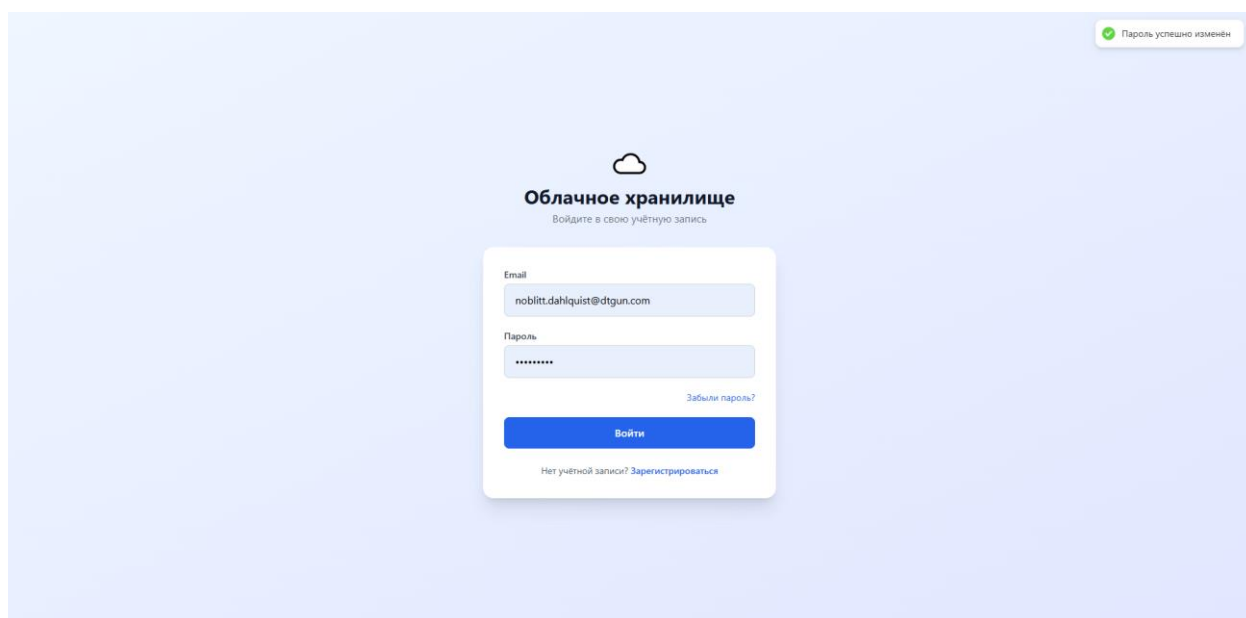


Рисунок 17 – Появление уведомления об успешном изменении пароля

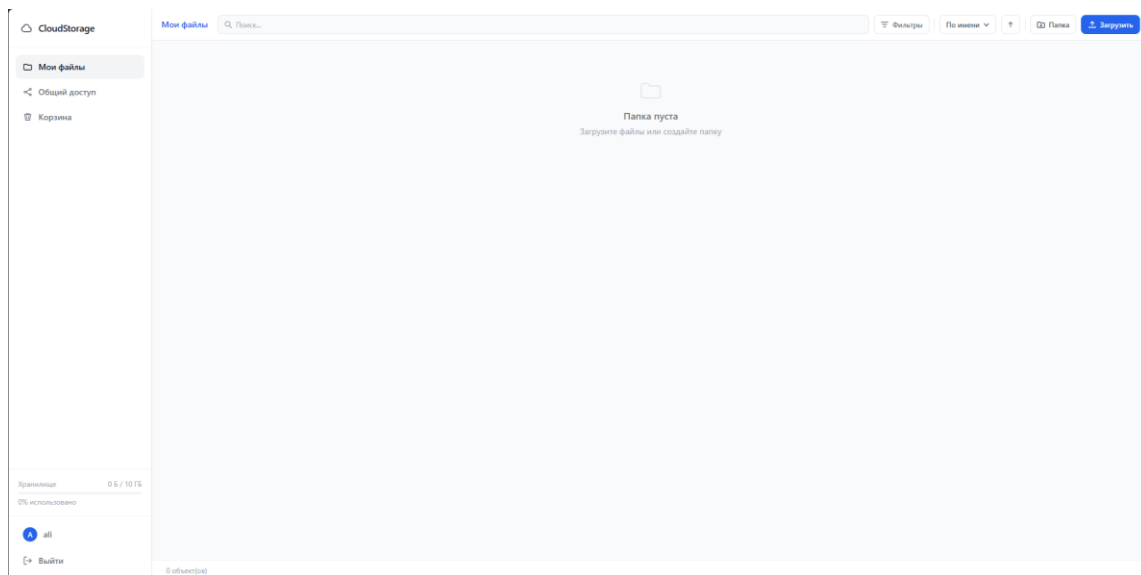


Рисунок 18 – Пользователь успешно авторизовался под новым паролем

Таблица 5.4 – Тест-кейс № 4: Смена пароля учетной записи

Параметр	Описание
Предусловие	Пользователь авторизован
Шаги выполнения	<p>Перейти в профиль</p> <p>Выбрать вкладку «Пароль»</p> <p>Ввести текущий, новый и подтверждение нового пароля</p> <p>Нажать кнопку «Изменить пароль»</p> <p>Выйти из учетной записи</p> <p>Войти в систему под новым паролем</p>
Ожидаемый результат	При нажатии на кнопку смены пароля появилось уведомление об успешном изменении пароля. Пользователь смог войти под новым паролем
Фактический результат	При нажатии на кнопку смены пароля появилось уведомление об успешном изменении пароля. Пользователь смог войти под новым паролем

Фактический результат тест-кейса представлен нарисунке 19 и 20.

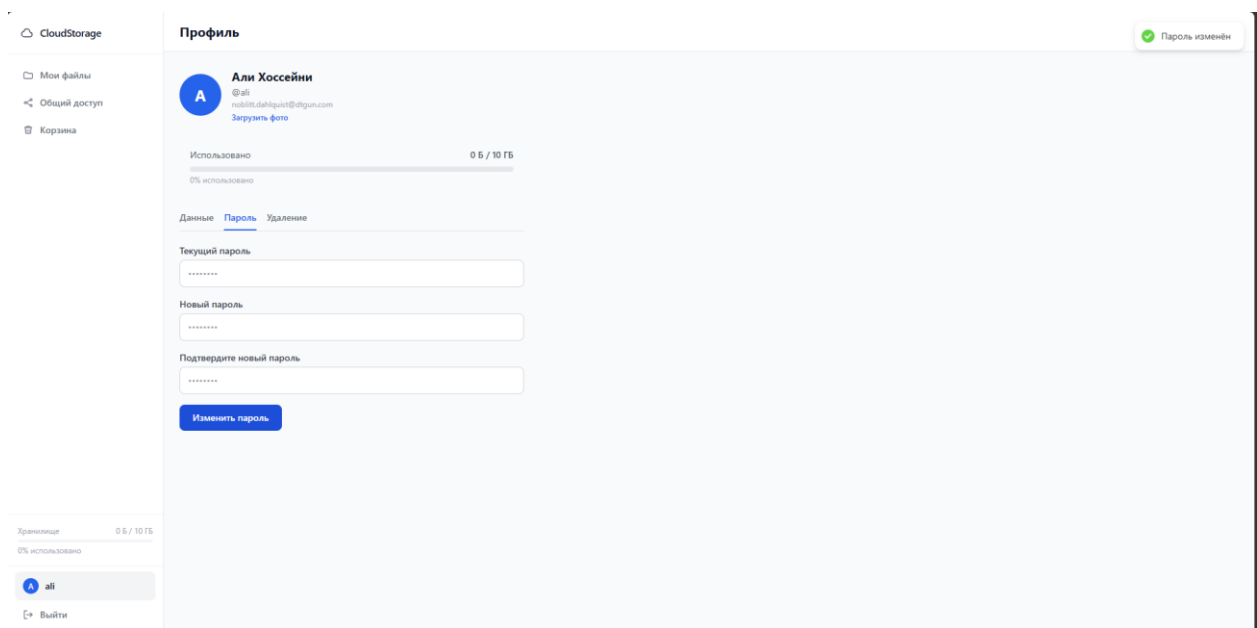


Рисунок 19 – Появление уведомления об успешном изменении пароля

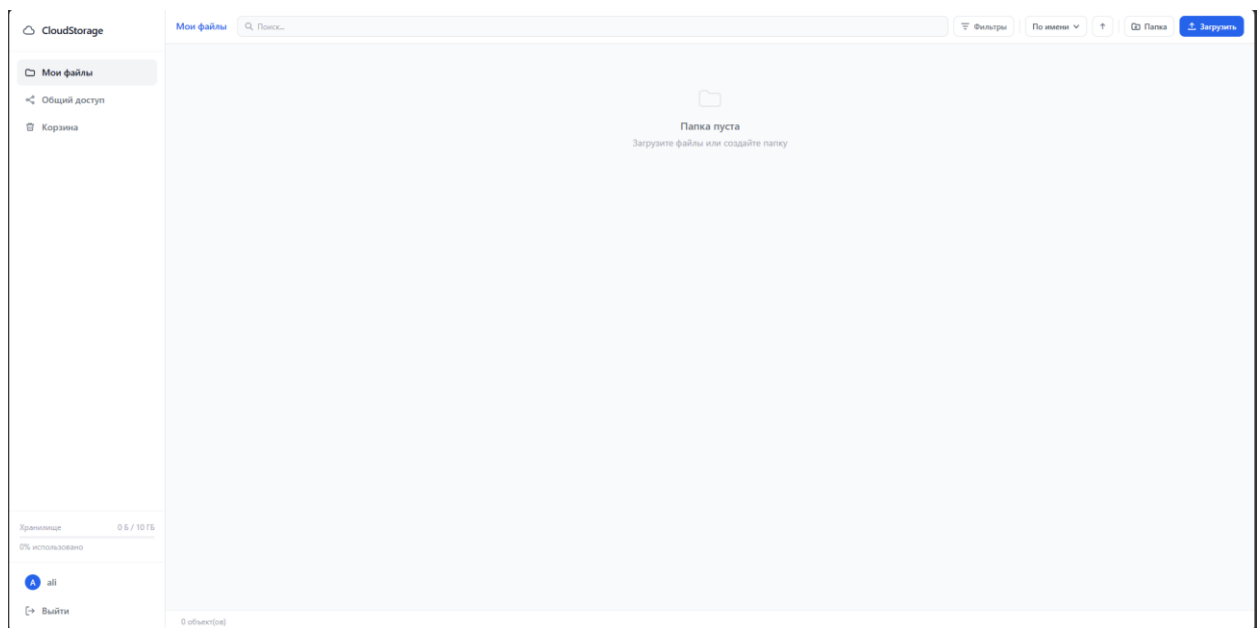


Рисунок 20 – Пользователь успешно авторизовался под новым паролем

Таблица 5.5 – Тест-кейс № 5: Загрузка файлов

Параметр	Описание
Предусловие	Пользователь авторизован Открыт раздел «Мои файлы»
Шаги выполнения	Нажать на кнопку загрузки файла Перетащить файл в зону загрузки файла
Ожидаемый результат	Файл успешно загружен
Фактический результат	Файл успешно загружен

Фактический результат тест-кейса представлен на рисунке 21.

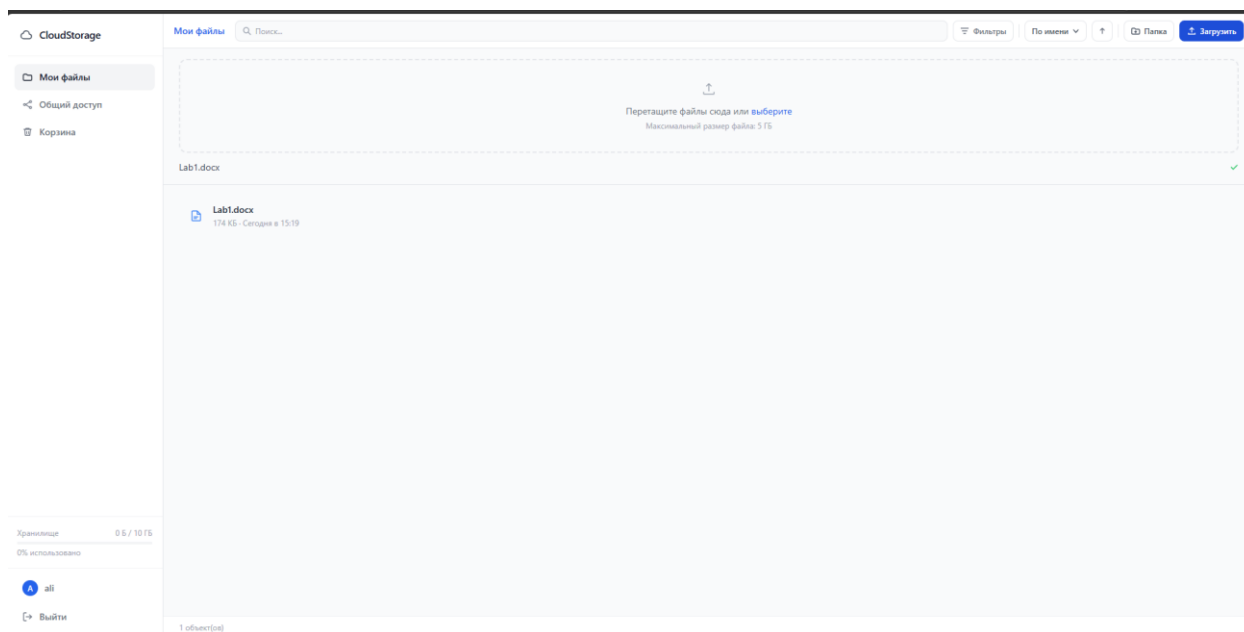


Рисунок 21 – Успешная загрузка файла

Таблица 5.6 – Тест-кейс № 6: Переименование файлов

Параметр	Описание
Предусловие	Пользователь авторизован Открыт раздел «Мои файлы» Файл загружен
Шаги выполнения	Выбрать файл

Параметр	Описание
	Вызвать панель инструментов Выбрать переименование Назвать файл «Успешно переименован»
Ожидаемый результат	Файл успешно переименовался
Фактический результат	Файл успешно переименовался

Фактический результат тест-кейса представлен нарисунке 22.

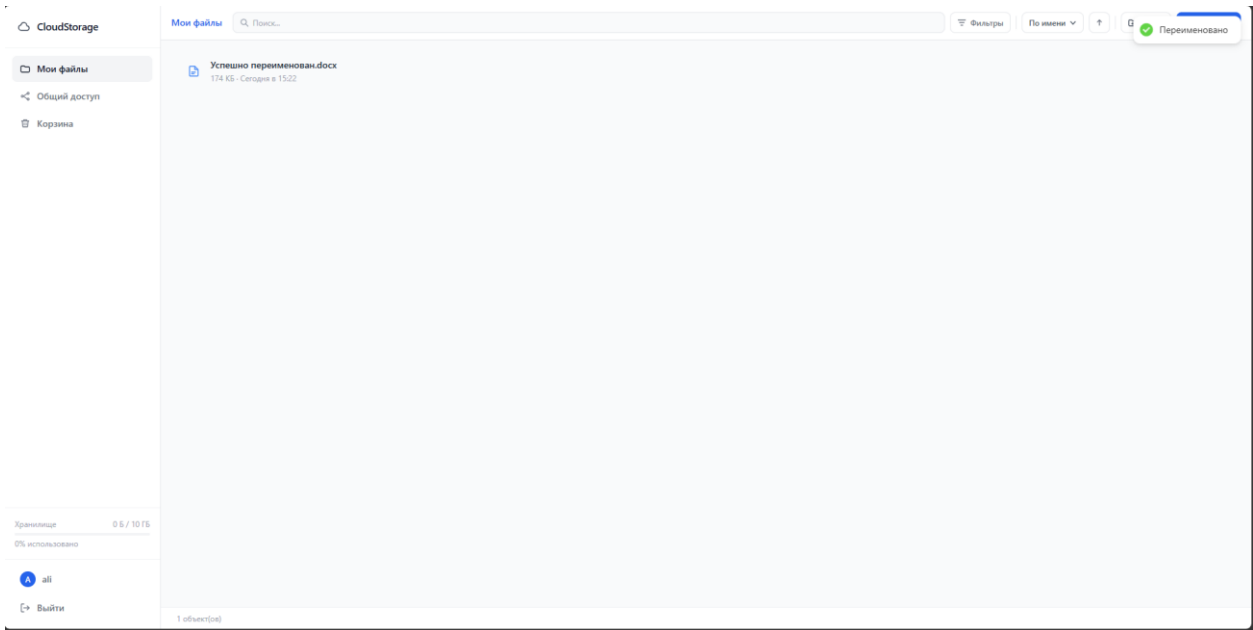


Рисунок 22 – Успешная переименовка файла

Таблица 5.7 – Тест-кейс № 7: Скачивание файлов

Параметр	Описание
Предусловие	Пользователь авторизован Открыт раздел «Мои файлы» Файл загружен
Шаги выполнения	Выбрать файл Вызвать панель инструментов Выбрать «Скачать»
Ожидаемый результат	Файл успешно установлен пользователю

Параметр	Описание
Фактический результат	Файл успешно установлен пользователю

Фактический результат тест-кейса представлен на рисунке 23.

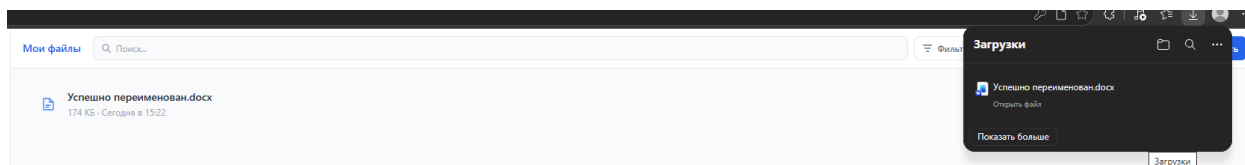


Рисунок 23 – Успешное скачивание файла

Таблица 5.8 – Тест-кейс № 8: Создание папок

Параметр	Описание
Предусловие	Пользователь авторизован Открыт раздел «Мои файлы»
Шаги выполнения	Нажать на кнопку «Папка» Ввести название папки Нажать «Ок»
Ожидаемый результат	Папка успешно создана
Фактический результат	Папка успешно создана

Фактический результат тест-кейса представлен на рисунке 24.

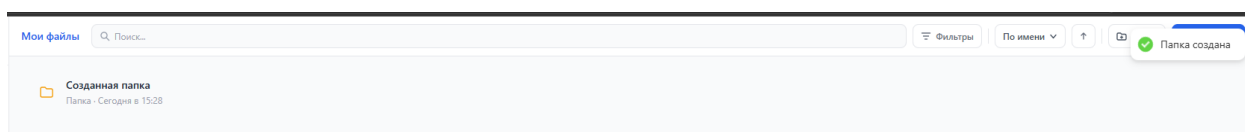


Рисунок 24 – Успешное создание папки

Таблица 5.9 – Тест-кейс № 9: Перемещение файлов

Параметр	Описание
Предусловие	Пользователь авторизован

Параметр	Описание
	Открыт раздел «Мои файлы» Файл загружен Создана папка
Шаги выполнения	Выбрать файл Вызвать панель инструментов Выбрать «Переместить» Выбрать перемещение в папку Открыть папку
Ожидаемый результат	Файл успешно перемещен в папку
Фактический результат	Файл успешно перемещен в папку

Фактический результат тест-кейса представлен на рисунке 25 и 26.

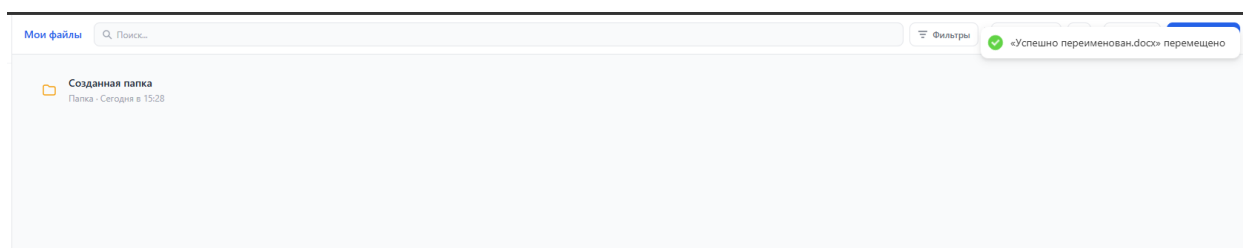


Рисунок 25 – Уведомление об успешном перемещении файла

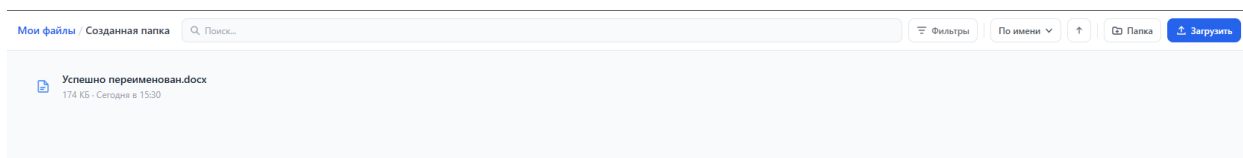


Рисунок 26 – Успешно перемещение файла

Таблица 5.10 – Тест-кейс № 10: Удаление файлов

Параметр	Описание
Предусловие	Пользователь авторизован Открыт раздел «Мои файлы» Файл загружен

Параметр	Описание
Шаги выполнения	Выбрать файл Вызвать панель инструментов Выбрать удаление
Ожидаемый результат	Файл успешно перемещен в корзину
Фактический результат	Файл успешно перемещен в корзину

Фактический результат тест-кейса представлен на рисунке 27.

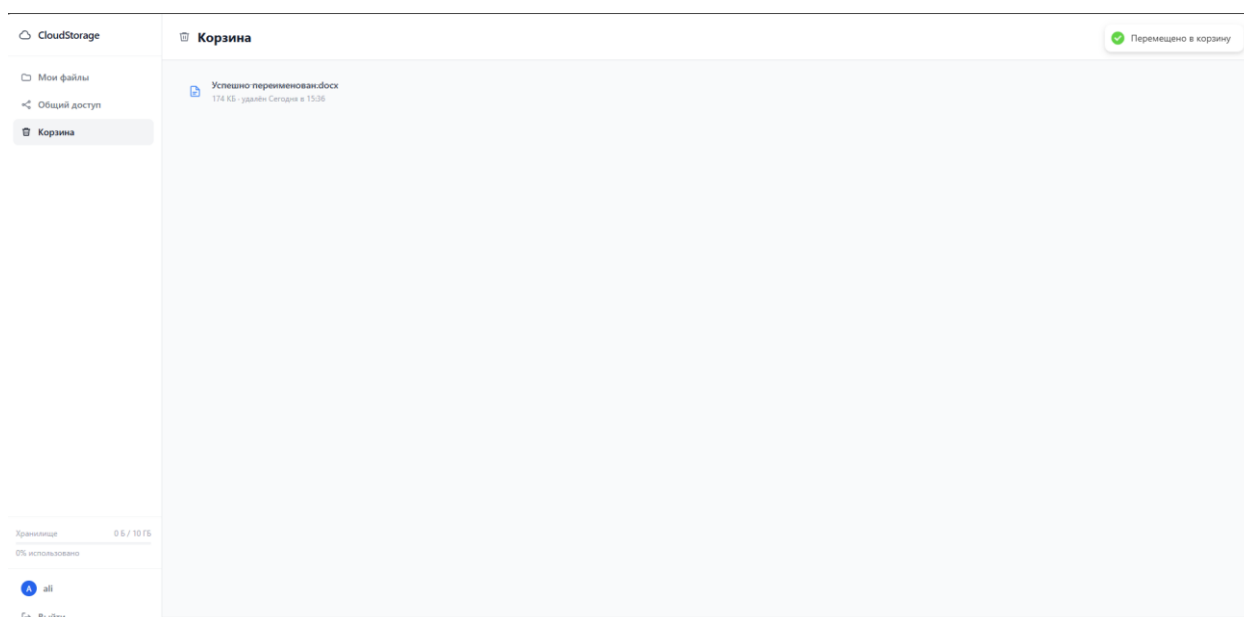


Рисунок 27 – Удаление файла

Таблица 5.11 – Тест-кейс № 11: Удаление файла из корзины

Параметр	Описание
Предусловие	Пользователь авторизован Пользователь удалил файл
Шаги выполнения	Перейти в раздел корзины Нажать на кнопку «Удалить» в требуемом для удаления файле Подтвердить удаление файла кнопкой «ок»
Ожидаемый результат	Файл успешно полностью удален

Параметр	Описание
Фактический результат	Файл успешно полностью удален

Фактический результат тест-кейса представлен нарисунке 28.

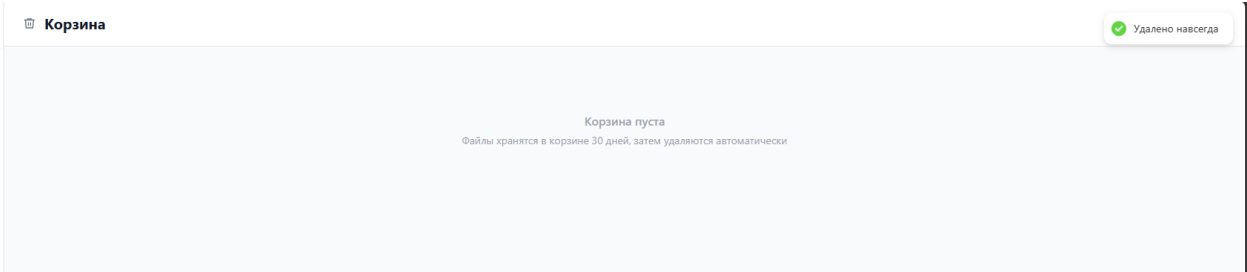


Рисунок 28 – Удаление файла из корзины

Таблица 5.12 – Тест-кейс № 12: Восстановление удаленных объектов из корзины

Параметр	Описание
Предусловие	Пользователь авторизован Пользователь удалил файл
Шаги выполнения	Перейти в раздел корзины Нажать на «Восстановить» на нужном файле
Ожидаемый результат	Файл успешно восстановлен
Фактический результат	Файл успешно восстановлен

Фактический результат тест-кейса представлен нарисунках 29-31.

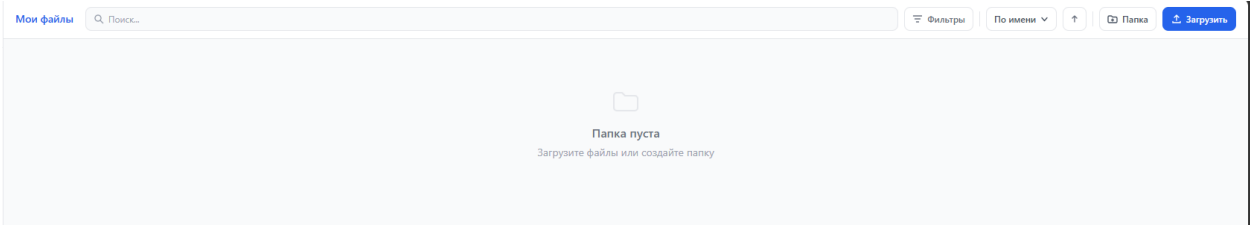


Рисунок 29 – Раздел «Мои файлы» до восстановления файла

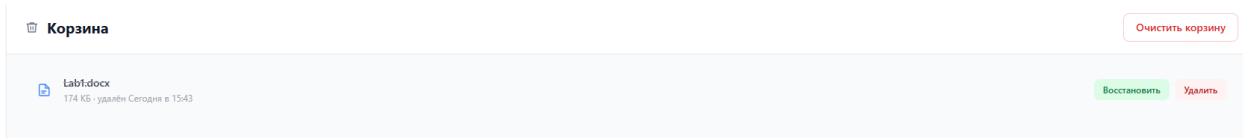


Рисунок 30 – Нажатие кнопки «Восстановить» в разделе корзины

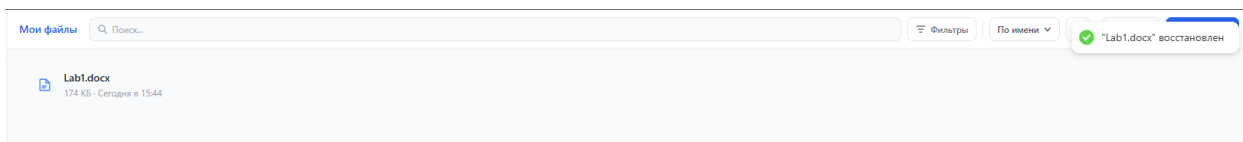


Рисунок 31 – Успешное восстановление файла

Таблица 5.13 – Тест-кейс № 13: AI-чат по содержимому файла

Параметр	Описание
Предусловие	Пользователь авторизован
Шаги выполнения	Загрузить файл 1 лабораторной работы расчета интегралла на языке программирования Java. Вызвать панель инструментов у файла Выбрать «Спросить AI» Задать вопрос «О чём этот документ?»
Ожидаемый результат	В чате отображается корректный ответ о содержимом файла
Фактический результат	В чате отображается корректный ответ о содержимом файла

Фактический результат тест-кейса представлен на рисунке 32.

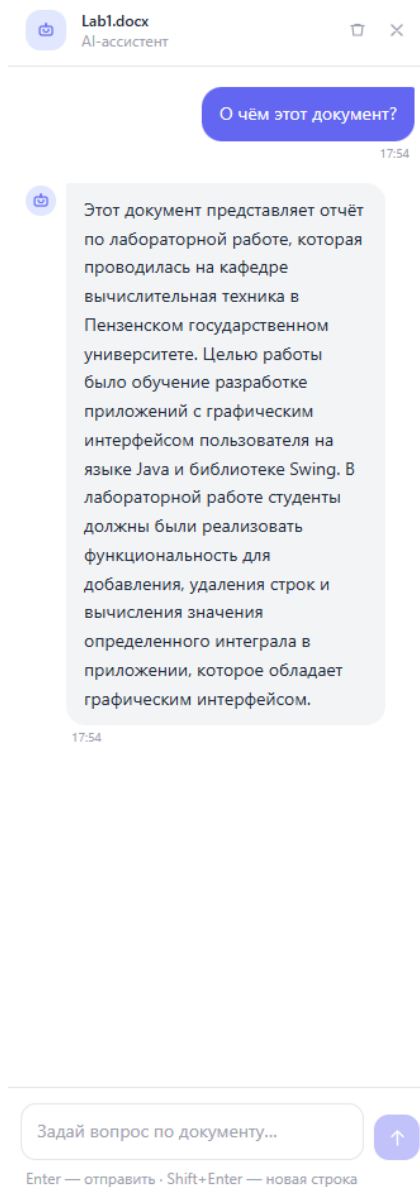


Рисунок 32 - Корректный ответ о содержимом файла

Таблица 5.14 – Тест-кейс № 14: Выдача доступа к файлу другому пользователю с уровнем прав «Читатеть»

Параметр	Описание
Предусловие	Пользователь 1 и 2 зарегистрированы и авторизованы.
Шаги выполнения	Загрузить пользователем 1 файл Нажать на кнопку «Поделиться»

Параметр	Описание
	Ввести в поле почту 2 пользователя Выбрать уровень прав «Читатель» Нажать кнопку «Дать доступ»
Ожидаемый результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI.
Фактический результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI

Фактический результат тест-кейса представлен нарисунке 33 и 34

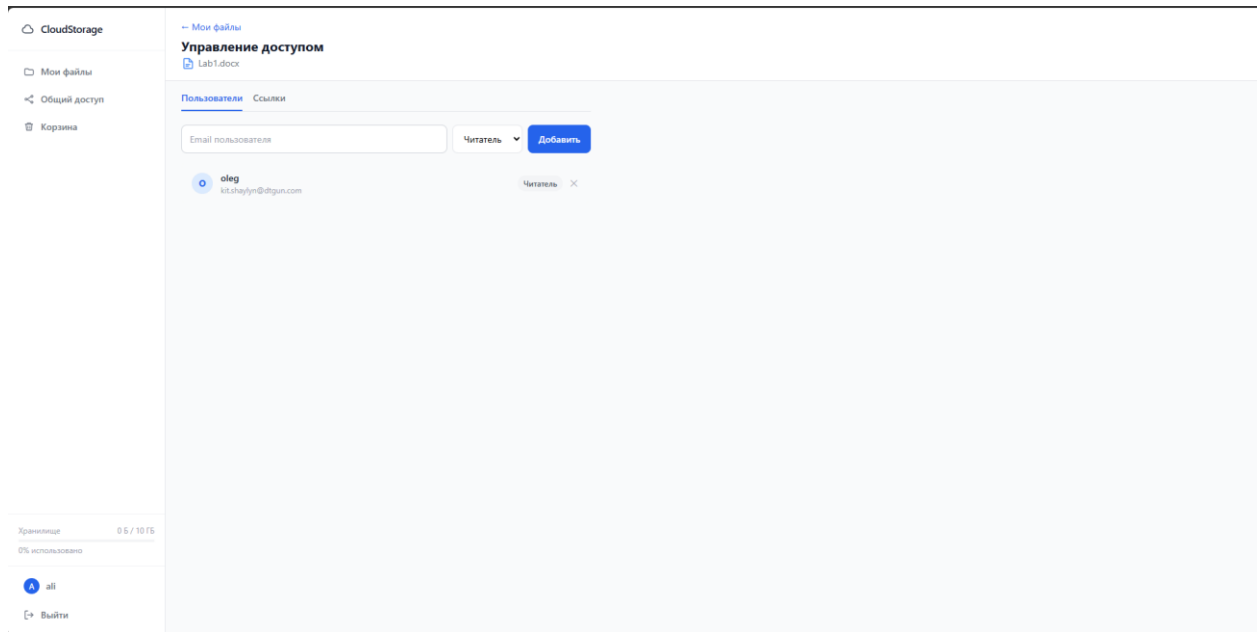


Рисунок 33 – Отображение выданного доступа в управлении доступом

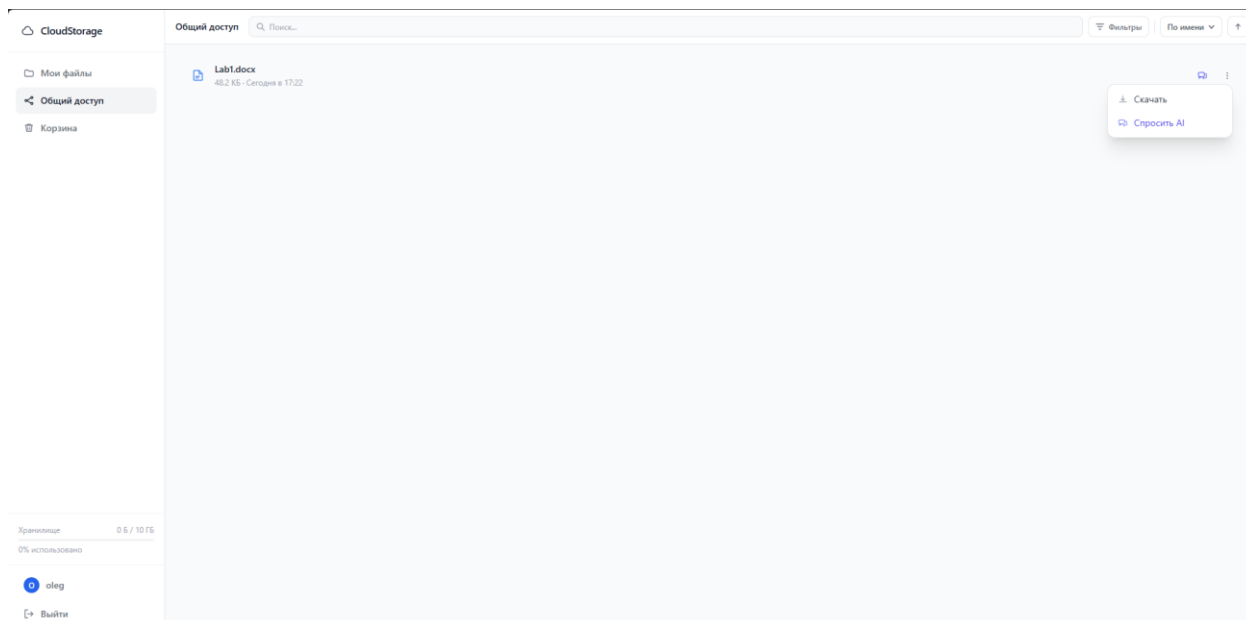


Рисунок 34 – Отображение файла у пользователя с функциями: скачать, спросить AI

Таблица 5.15 – Тест-кейс № 15: Выдача доступа к файлу другому пользователю с уровнем прав «Автор»

Параметр	Описание
Предусловие	Пользователь 1 и 2 зарегистрированы и авторизованы.
Шаги выполнения	Загрузить пользователем 1 файл Нажать на кнопку «Поделиться» Ввести в поле почту 2 пользователя Выбрать уровень прав «Автор» Нажать кнопку «Дать доступ»
Ожидаемый результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать
Фактический результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился

Параметр	Описание
	файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать

Фактический результат тест-кейса представлен нарисунке 35 и 36.

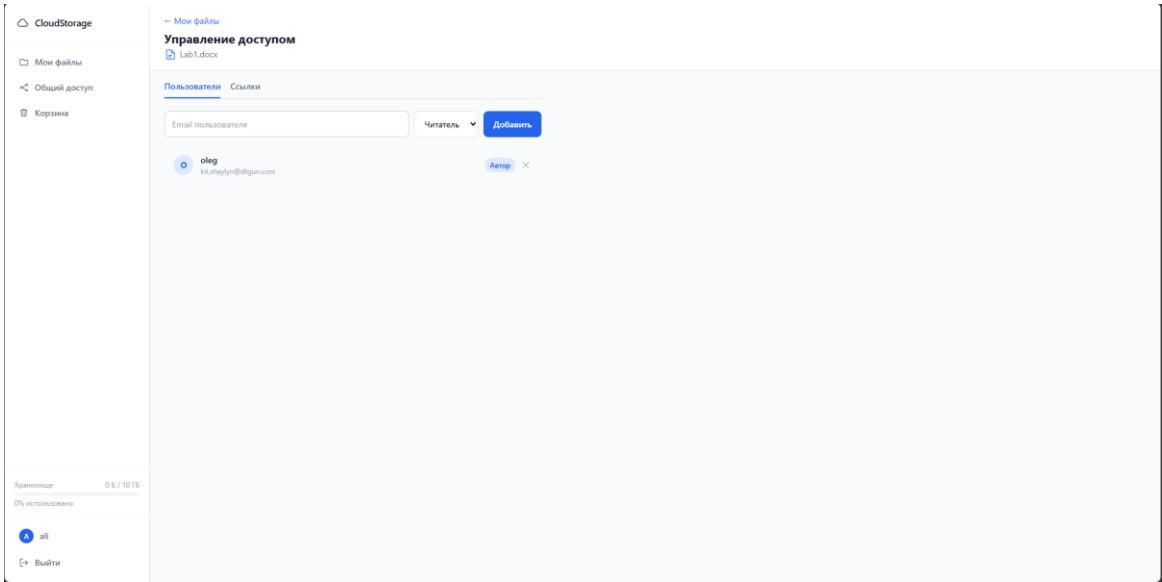


Рисунок 35 – Отображение выданного доступа в управлении доступом

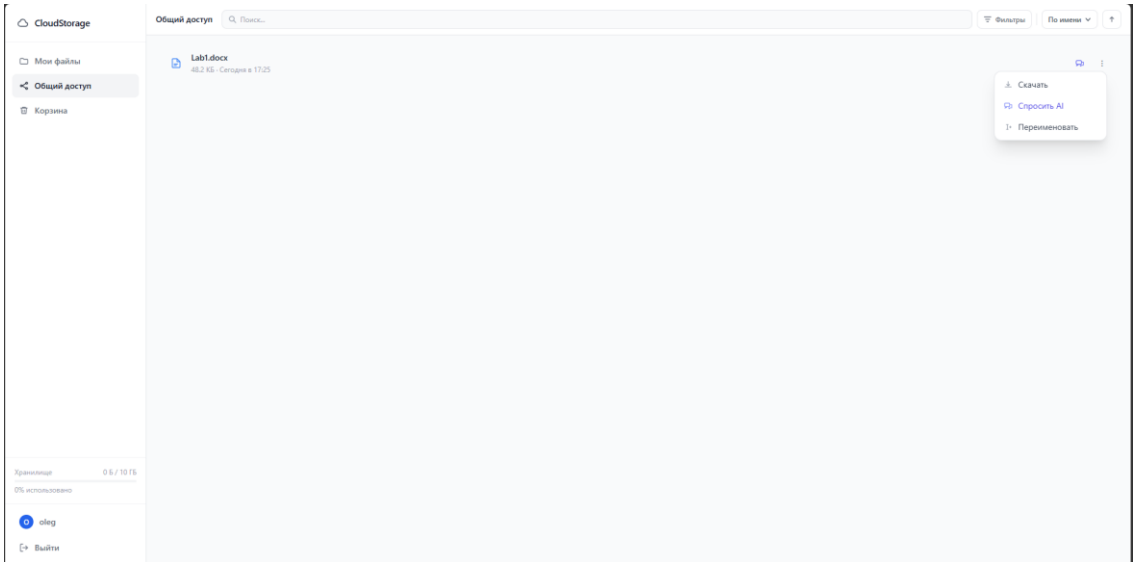


Рисунок 36 – Отображение файла у пользователя с функциями: скачать, спросить AI, переименовать

Таблица 5.16 – Тест-кейс № 16: Выдача доступа к файлу другому пользователю с уровнем прав «Редактор»

Параметр	Описание
Предусловие	Пользователь 1 и 2 зарегистрированы и авторизованы.
Шаги выполнения	Загрузить пользователем 1 файл Нажать на кнопку «Поделиться» Ввести в поле почту 2 пользователя Выбрать уровень прав «Редактор» Нажать кнопку «Дать доступ»
Ожидаемый результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать, переместить, в корзину
Фактический результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать, переместить, в корзину

Фактический результат тест-кейса представлен на рисунке 37 и 38.

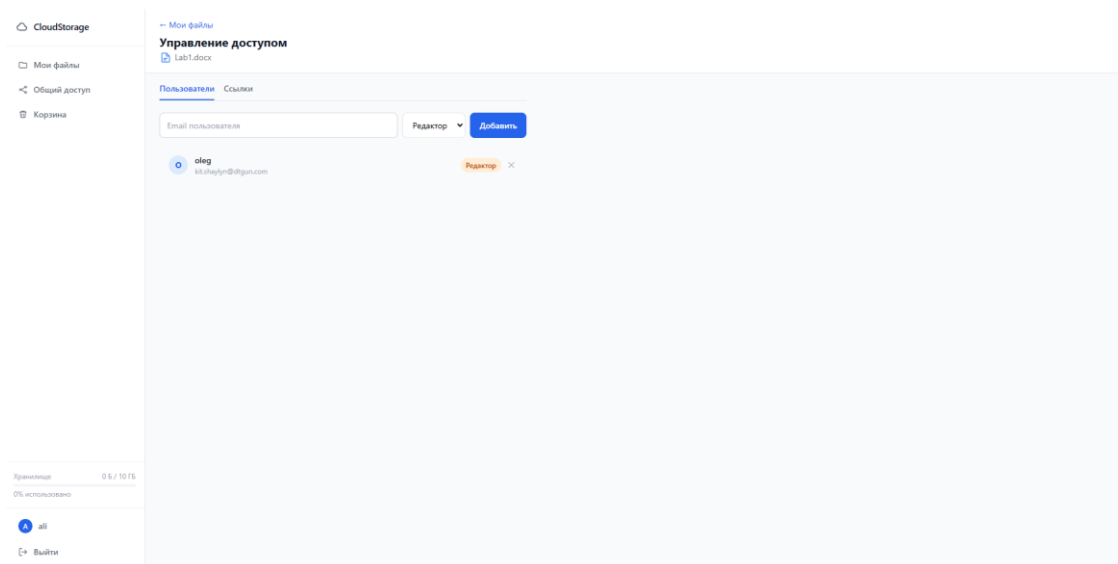


Рисунок 37 – Отображение выданного доступа в управлении доступом

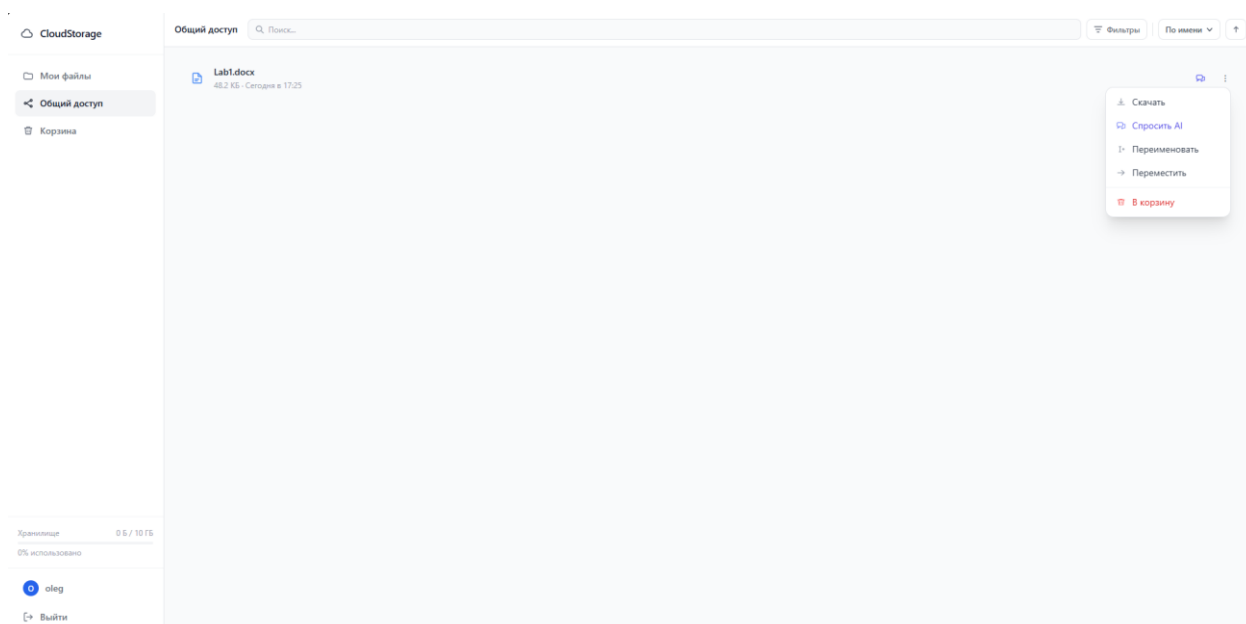


Рисунок 38 – Отображение файла у пользователя с функциями: скачать, спросить AI, переименовать, переместить, в корзину

Таблица 5.17 – Тест-кейс № 17: Выдача доступа к файлу другому пользователю с уровнем прав «Владелец»

Параметр	Описание
Предусловие	Пользователь 1 и 2 зарегистрированы и авторизованы.
Шаги выполнения	Загрузить пользователем 1 файл Нажать на кнопку «Поделиться» Ввести в поле почту 2 пользователя Выбрать уровень прав «Владелец» Нажать кнопку «Дать доступ»
Ожидаемый результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать, переместить, в корзину
Фактический результат	В «Управление доступом» отображается выданный доступ. У 2 пользователя появился файл в «Общий доступ» с доступными функциями: скачать, спросить AI, переименовать, переместить, в корзину поделиться, управление доступом

Фактический результат тест-кейса представлен на рисунке 39 и 40.

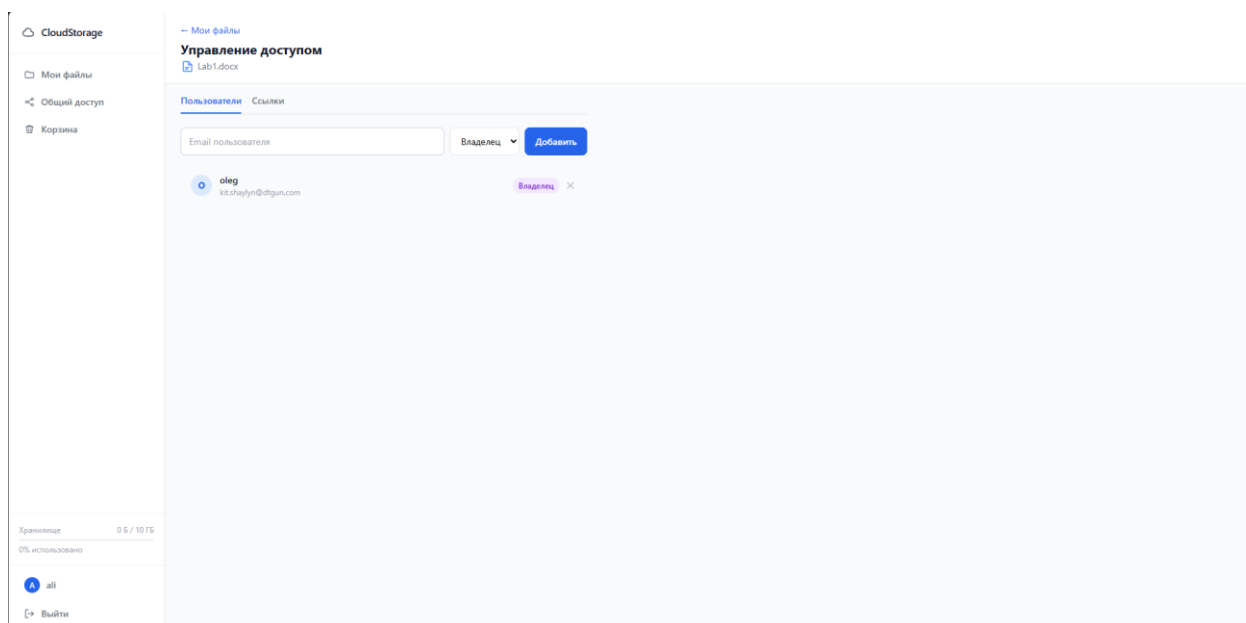


Рисунок 39 – Отображение выданного доступа в управлении доступом

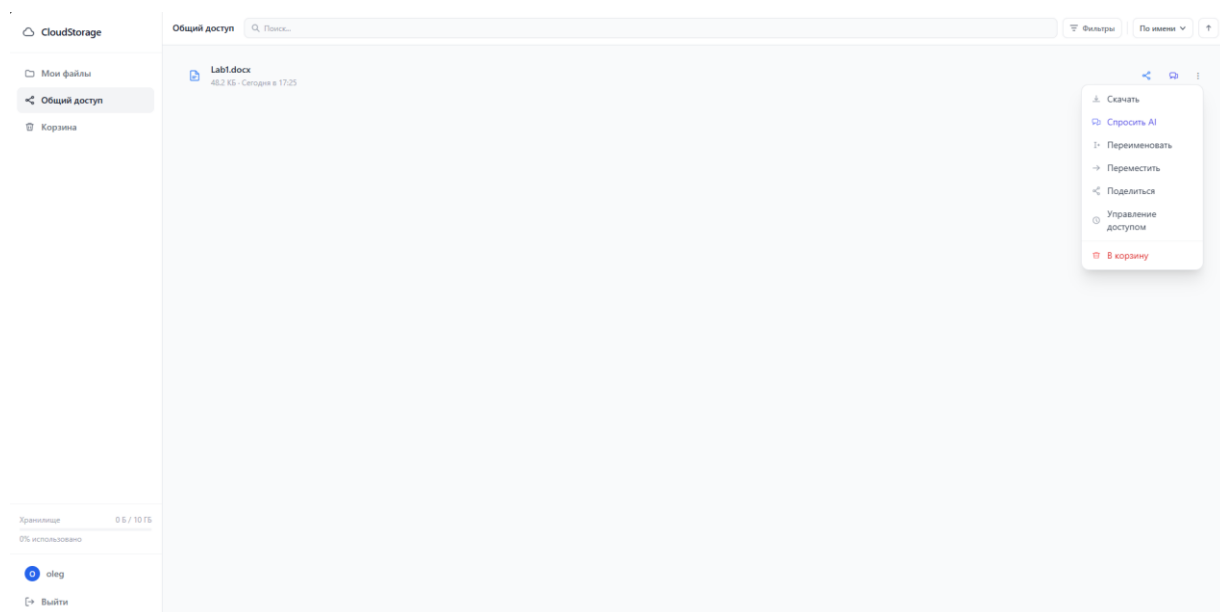


Рисунок 40 – Отображение файла у пользователя с функциями: скачать, спросить AI, переименовать, переместить, в корзину, поделиться, управление доступом

Таблица 5.18 – Тест-кейс № 18: Создание публичной ссылки на папку с уровнем прав «Читатель»

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	Создать пользователем папку и загрузить в нее файл. Нажать на кнопку «Поделиться» Переключить вкладку на «Ссылки» Выбрать уровень прав «Читатель» Нажать кнопку «создать ссылку» Нажать на кнопку «Скопировать ссылку»
Ожидаемый результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функция «Скачать»
Фактический результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функция «Скачать»

Фактический результат тест-кейса представлен на рисунке 41.

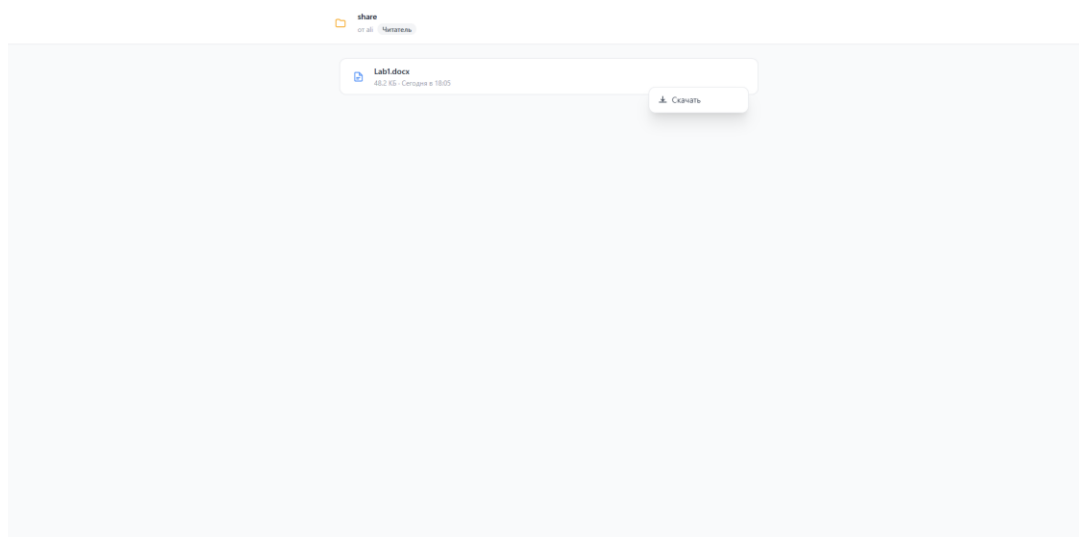


Рисунок 41 – Отображение функций при публичной ссылке с уровнем доступа «Читатель»

Таблица 5.19 – Тест-кейс № 19: Создание публичной ссылки на папку с уровнем прав «Автор»

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	Создать пользователем папку и загрузить в нее файл. Нажать на кнопку «Поделиться» Переключить вкладку на «Ссылки» Выбрать уровень прав «Читатель» Нажать кнопку «создать ссылку» Нажать на кнопку «Скопировать ссылку»
Ожидаемый результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функции: «Скачать», «Переименовать», «Загрузить», создание папки.
Фактический результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функции: «Скачать», «Переименовать», «Загрузить», создание папки.

Фактический результат тест-кейса представлен на рисунке 42.

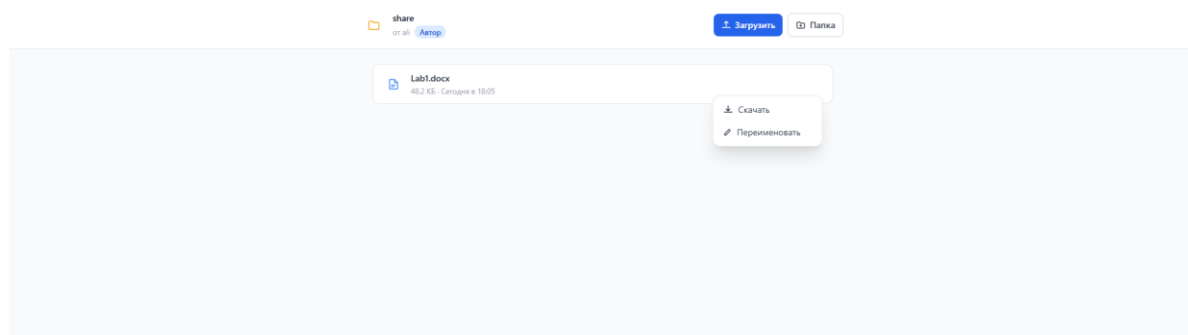


Рисунок 42 – Отображение функций при публичной ссылке с уровнем доступа «Автор»

Таблица 5.20 – Тест-кейс № 20: Создание публичной ссылки на папку с уровнем прав «Редактор»

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	Создать пользователем папку и загрузить в нее файл. Нажать на кнопку «Поделиться» Переключить вкладку на «Ссылки» Выбрать уровень прав «Редактор» Нажать кнопку «создать ссылку» Нажать на кнопку «Скопировать ссылку»
Ожидаемый результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функции по отношению к вложенным объектам: «Скачать», «Переименовать», «Загрузить», создание папки, «Удалить»
Фактический результат	При переходе по ссылке пользователю отображается содержимое папки. Доступна функции по отношению к вложенным объектам: «Скачать», «Переименовать», «Загрузить», создание папки, «Удалить»

Фактический результат тест-кейса представлен на рисунке 43.

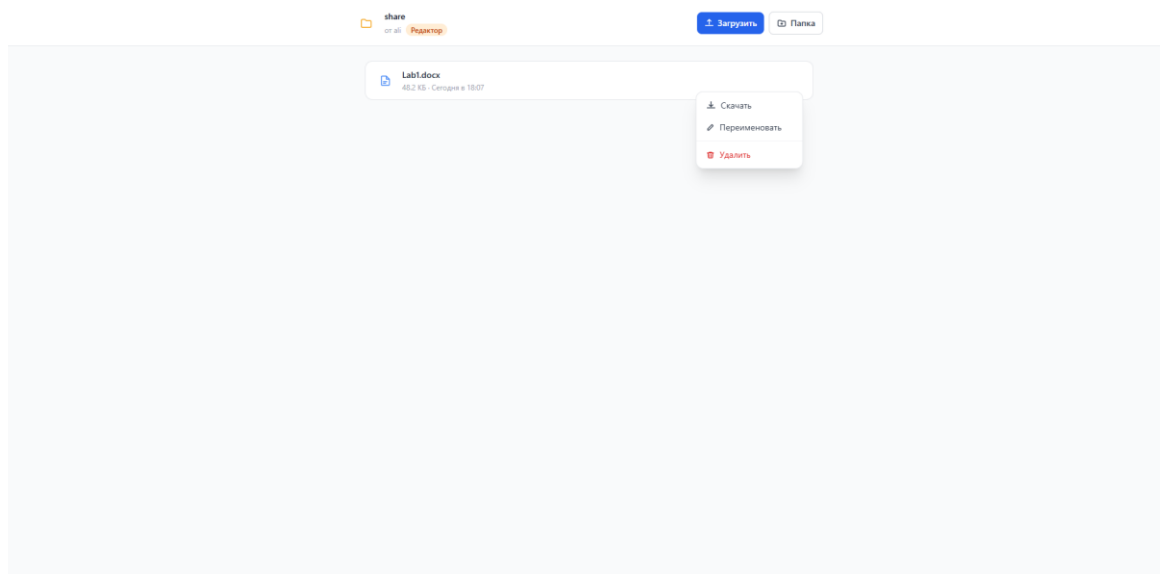


Рисунок 43 – Отображение функций при публичной ссылке с уровнем доступа «Редакторы»

Таблица 5.21 – Тест-кейс № 21: Поиск по названию

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	Загрузить 3 файла: Lab1, Lab2, Lab3 Выбрать в фильтрах режим поиска «По названию» Ввести в строку поиска «Lab2» Нажать клавишу «Enter»
Ожидаемый результат	В результате поиска отображается только файл с названием «Lab2»
Фактический результат	В результате поиска отображается только файл с названием «Lab2»

Фактический результат тест-кейса представлен на рисунке 44.

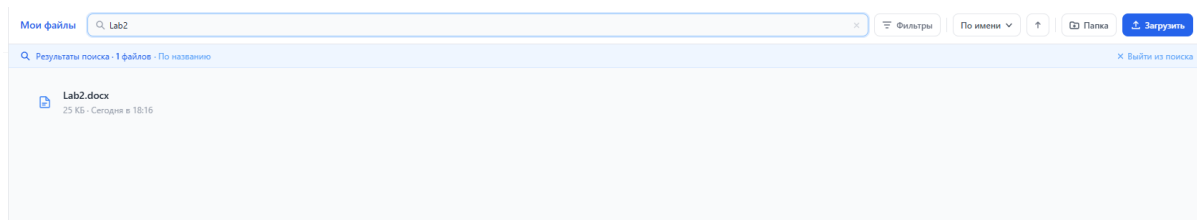


Рисунок 44 – Поиск по названию

Таблица 5.22 – Тест-кейс № 22: Полнотекстовый поиск

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	Загрузить 3 файла: Lab1, Lab2, Lab3 Выбрать в фильтрах режим поиска «Полнотекстовый» Ввести в строку поиска «по лабораторной работе №3» Нажать клавишу «Enter»
Ожидаемый результат	В результате на 1 месте отображается файл с названием «Lab3»
Фактический результат	В результате на 1 месте отображается файл с названием «Lab3»

Фактический результат тест-кейса представлен на рисунке 45.

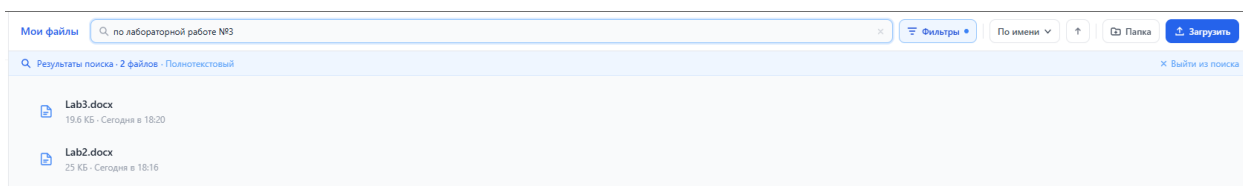


Рисунок 45 – Полнотекстовый поиск

Таблица 5.23 – Тест-кейс № 23: Семантический поиск

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	<p>Загрузить 3 файла: Lab1, Lab2, Lab3</p> <p>Выбрать в фильтрах режим поиска «Семантический»</p> <p>Ввести в строку поиска «3 лабораторная работа по Java»</p> <p>Нажать клавишу «Enter»</p>
Ожидаемый результат	В результате на 1 месте отображается файл с названием «Lab3»
Фактический результат	В результате на 1 месте отображается файл с названием «Lab3»

Фактический результат тест-кейса представлен на рисунке 46.

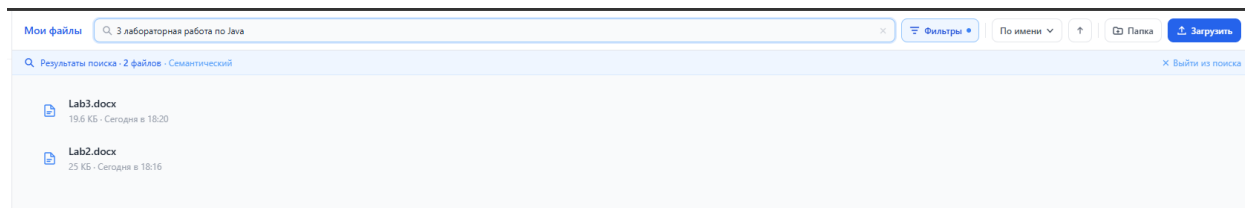


Рисунок 46 – Семантический поиск

Таблица 5.24 – Тест-кейс № 24: Гибридный поиск

Параметр	Описание
Предусловие	Пользователь зарегистрирован и авторизован.
Шаги выполнения	<p>Загрузить 3 файла: Lab1, Lab2, Lab3</p> <p>Выбрать в фильтрах режим поиска «Гибридный»</p> <p>Ввести в строку поиска «3 лабораторная работа по Java»</p> <p>Нажать клавишу «Enter»</p>

Параметр	Описание
Ожидаемый результат	В результате на 1 месте отображается файл с названием «Lab3»
Фактический результат	В результате на 1 месте отображается файл с названием «Lab3»

Фактический результат тест-кейса представлен на рисунке 47.

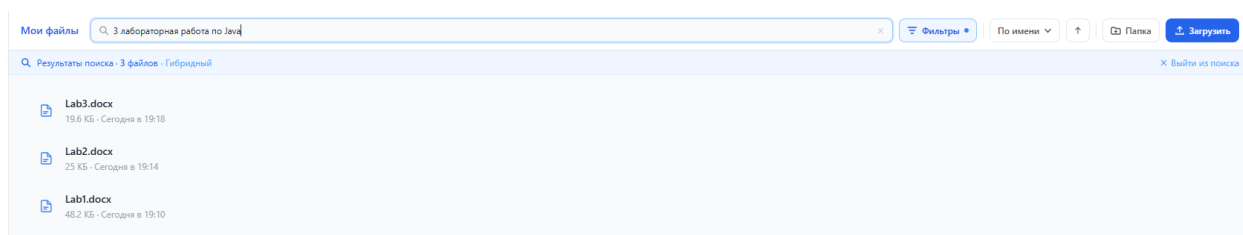


Рисунок 47 – Гибридный поиск

6 Безопасность жизнедеятельности

Безопасность жизнедеятельности является одним из наиболее важных аспектов при работе за компьютером. Несоблюдения мер безопасности при работе за компьютером может привести ко многим проблемам со здоровьем, в частности со зрением и ментальным здоровьем.

В данном разделе будут рассмотрены основные требования безопасности жизнедеятельности при работе за компьютером [14].

6.1 Требования к помещению и рабочему месту

Помещение обязательно должно иметь искусственное и естественное освещение. Должна быть хорошая вентиляция помещения, для предотвращения скапливания вредных веществ и поступления свежего воздуха. Площадь на одно рабочее место с жидкокристаллическими мониторами должна быть более 4,4 квадратных метров. Уровень шума в помещении не должен превышать 50 дБ. Допустимая температура помещения 20-24 °С, а влажность не более 75 %.

Для соблюдения норм по электромагнитному излучению от мониторов, клавиатуры, соединительных кабелей, необходимо защитное заземление.

Расположение рабочего стола должно быть таким, чтобы окна располагались к бокам монитора. На окнах должны присутствовать шторы от попадания прямых солнечных лучей.

Эргономические рекомендации рабочего места:

- расстояние от экрана до глаз должно быть не менее 500 мм;
- верхний край экрана должен находиться чуть ниже уровня глаз;
- экран должен располагаться прямо перед пользователем;
- рабочий стул должен быть подъемно-поворотным;
- клавиатура должна располагаться на расстоянии 10-30 см от края стола;
- угол между плечом и предплечьем должен составлять 90 градусов.

6.2 Требования к освещению

Очень важно соблюдать требования к освещению, так как при неправильном освещении появляется напряжение и усталость глаз, что влияет не только на здоровье, но и на производительность труда.

В помещении должен быть равномерный и входящий в нормы яркости свет. Этого можно добиться за счет использования одновременно и естественного и искусственного освещения. Искусственное освещение должно осуществляться системами равномерного освещения, такими как люстры, потолочные светильники, лампы накаливания и люминесцентными лампами. Нельзя использовать лампы с голубым свечением. В целях предотвращения слепящего эффекта и бликов на экранах мониторов необходимо чтобы люминесцентные лампы были оборудованы экранирующими решетками.

Рекомендации по освещению:

- для использования естественного освещения расположения рабочего стола должно быть рядом с окном;
- должна быть равномерная освещенность рабочего стола менее 501 лк;
- должна быть общая освещенность на уровне 300 лк;
- цветовая температура искусственного освещения в пределах 4000-6500 К.

6.3 Требования к режиму труда и отдыха

Для минимизации ухудшения здоровья, получения травм, усталости и ошибок существует режим труда и отдыха. Режим труда и отдыха – это порядок чередования работы и перерывов в ней для восстановления сил.

При работе за компьютером рекомендуется делать перерывы каждые 60-90 минут работы и на 10-15 минут выйти из помещения или сделать зарядку.

Нормы режима труда и отдыха по ТК РФ:

- Продолжительность рабочего времени в неделе должно быть 40 часов.

Исключения: несовершеннолетние, инвалиды, работающие на вредных

					ПГУ 1.090301.24.001 ПЗ	Лист
						104
Изм.	Лист	№ докум.	Подпись	Дата		

производствах;

- Еженедельный непрерывный отдых должен длиться 42 часа при пятидневной рабочей неделе;
- Перерывы на отдых и питание обязательны если рабочий день превышает 4 часа;
- Максимальное время сверхурочной работы не должно превышать 4 часов в течение 2 дней подряд и не более 120 часов в год. К сверхурочной работе не привлекаются беременные женщины и несовершеннолетние.

6.4 Рекомендации по использованию программного обеспечения

Для соблюдения безопасности работы с компьютером рекомендуется применять специализированное программное обеспечение. Например для снижения напряжения глаз следует использовать Night Shift на Mac и Night Light в Windows. Чтобы следить за временем труда и отдыха следует использовать такие программы как EyeLeo или Workrave. Рекомендуется также использовать программы уменьшения эмоционального напряжения, которые воспроизводят спокойную музыку или звуки природы. К таким программам относятся Noisli или A Soft Murmur.

Заключение

В ходе выполнения выпускной квалификационной работы была изучена актуальность облачных хранилищ, анализ аналогов, были выбраны технологии и среды разработки, спроектирована архитектура, реализовано веб-приложение «Облачное хранилище с управлением доступом и системой поиска», а также было проведено тестирование.

Во время разработки были получены знания построения микросервисной архитектуры с использованием брокера сообщений RabbitMQ. Углублены знания программирования серверной части на фреймворке Spring Boot с языком программирования Java и вспомогательного сервиса на Python. Закреплены навыки работы с фреймворком React и языком программирования JavaScript при разработке клиентской части. Получен опыт построения RAG-систем. Улучшены навыки работы с базой данных PostgreSQL с расширениями pgvector и pg_trgm. Получен опыт работы с Docker и Docker Compose для контейнеризации. Результаты тестирования показали, что реализованное веб-приложение «Облачное хранилище с управлением доступом и системой поиска» полностью соответствует всем поставленным требованиям.

В будущем можно улучшить веб-приложение добавив совместную работу над файлами, за счет которой пользователи смогут одновременно работать в режиме реального времени над одним документом. Также будет важно добавить версионирование файлов для восстановления предыдущих редакций документа. Кроме того, одним из перспективных улучшений будет добавление механизма увеличения доступного объема хранилища.

Полученный опыт в ходе разработки микросервисного веб-приложения «Облачное хранилища с управлением доступом и системой доступа» представляет огромную ценность для дальнейшей профессионально длительности в сфере разработки веб-приложений.

Список использованных источников

1. Spring Framework Documentation :: Spring Framework [Электронный ресурс]. — URL: <https://docs.spring.io/spring-framework/reference/index.html> — Дата обращения: 14.04.2026.
2. Spring Boot Reference Documentation [Электронный ресурс]. — URL: <https://docs.spring.io/spring-boot/docs/3.3.0-SNAPSHOT/reference/htmlsingle/> — Дата обращения: 14.04.2026.
3. Spring Data JPA :: Spring Data JPA [Электронный ресурс]. — URL: <https://docs.spring.io/spring-data/jpa/reference/> — Дата обращения: 14.04.2026.
4. Spring Security :: Spring Security [Электронный ресурс]. — URL: <https://docs.spring.io/spring-security/reference/index.html> — Дата обращения: 14.05.2026.
5. Introduction :: Spring AI Reference [Электронный ресурс]. — URL: <https://docs.spring.io/spring-ai/reference/> — Дата обращения: 14.04.2026.
6. PostgreSQL: Documentation: 16: PostgreSQL 16.14 Documentation [Электронный ресурс]. — URL: <https://www.postgresql.org/docs/16/index.html> — Дата обращения: 15.04.2026.
7. RabbitMQ Documentation | RabbitMQ [Электронный ресурс]. — URL: <https://www.rabbitmq.com/docs> — Дата обращения: 16.04.2026.
8. Quick Start – React [Электронный ресурс]. — URL: <https://react.dev/learn> — Дата обращения: 17.04.2026.
9. Maven Documentation – Maven [Электронный ресурс]. — URL: <https://maven.apache.org/guides/index.html> — Дата обращения: 14.04.2026.
10. Docker Compose | Docker Docs [Электронный ресурс]. — URL: <https://docs.docker.com/compose> — Дата обращения: 19.04.2026.
11. Ollama's documentation - Ollama [Электронный ресурс]. — URL: <https://docs.ollama.com/> — Дата обращения: 18.04.2026.
12. nginx: документация [Электронный ресурс]. — URL: <https://nginx.org/ru/docs/> — Дата обращения: 17.04.2026.

13. Apache Tika – Getting Started with Apache Tika [Электронный ресурс]. — URL: <https://tika.apache.org/2.9.1/gettingstarted.html> — Дата обращения: 16.04.2026.

14. Санитарно-гигиенические рекомендации для организации рабочего места пользователя персональным компьютером [Электронный ресурс]. — URL: <https://sakhgig.ru/main/info/664-sanitarno-gigienicheskie-rekomendacii-dlya-organizacii-rabochego-mesta-polzovatelya-personalnym-kompyuterom.html> — Дата обращения: 19.05.2026.

Приложение А

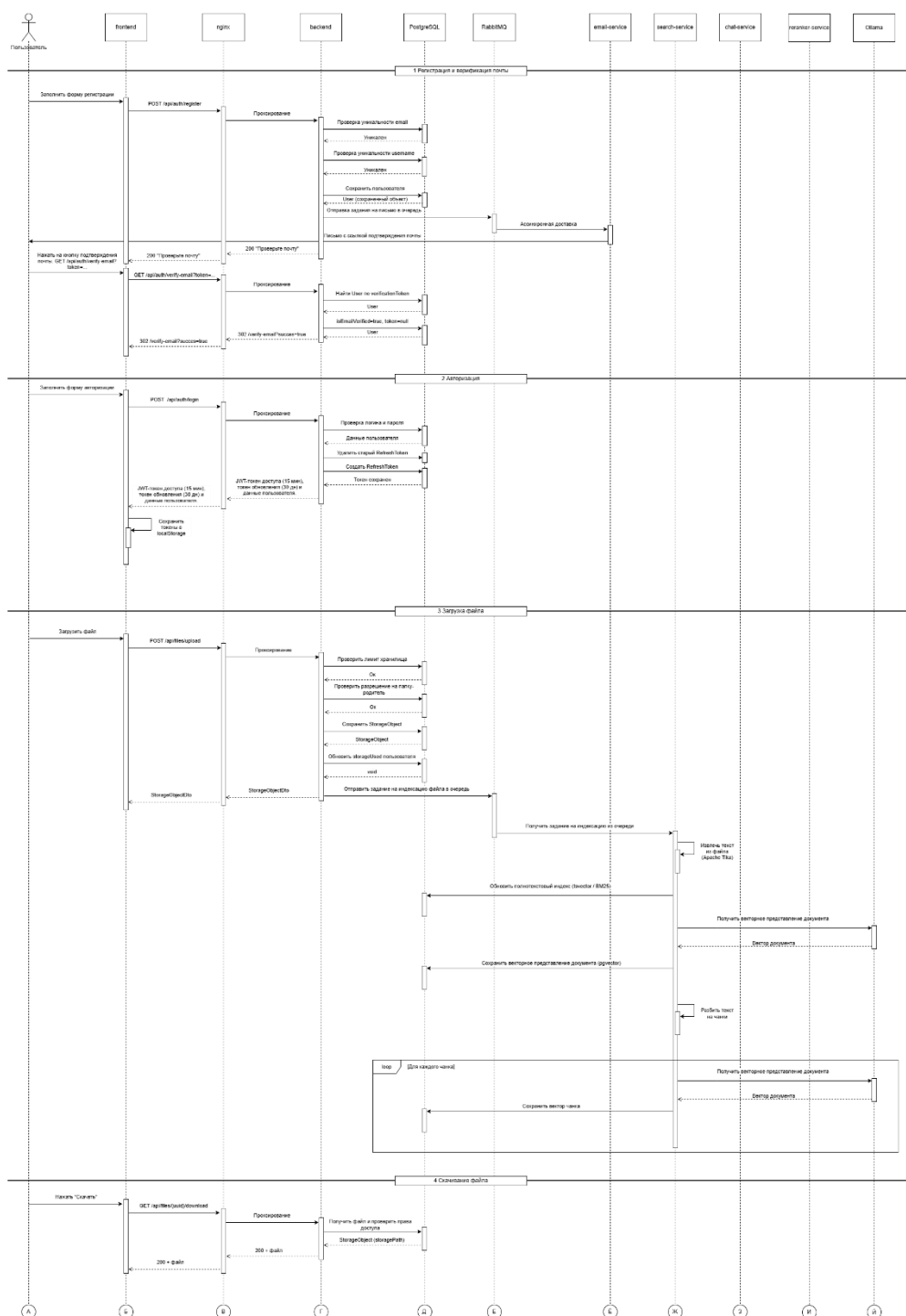
(обязательное)

Облачное хранилище с управлением доступом и системой поиска

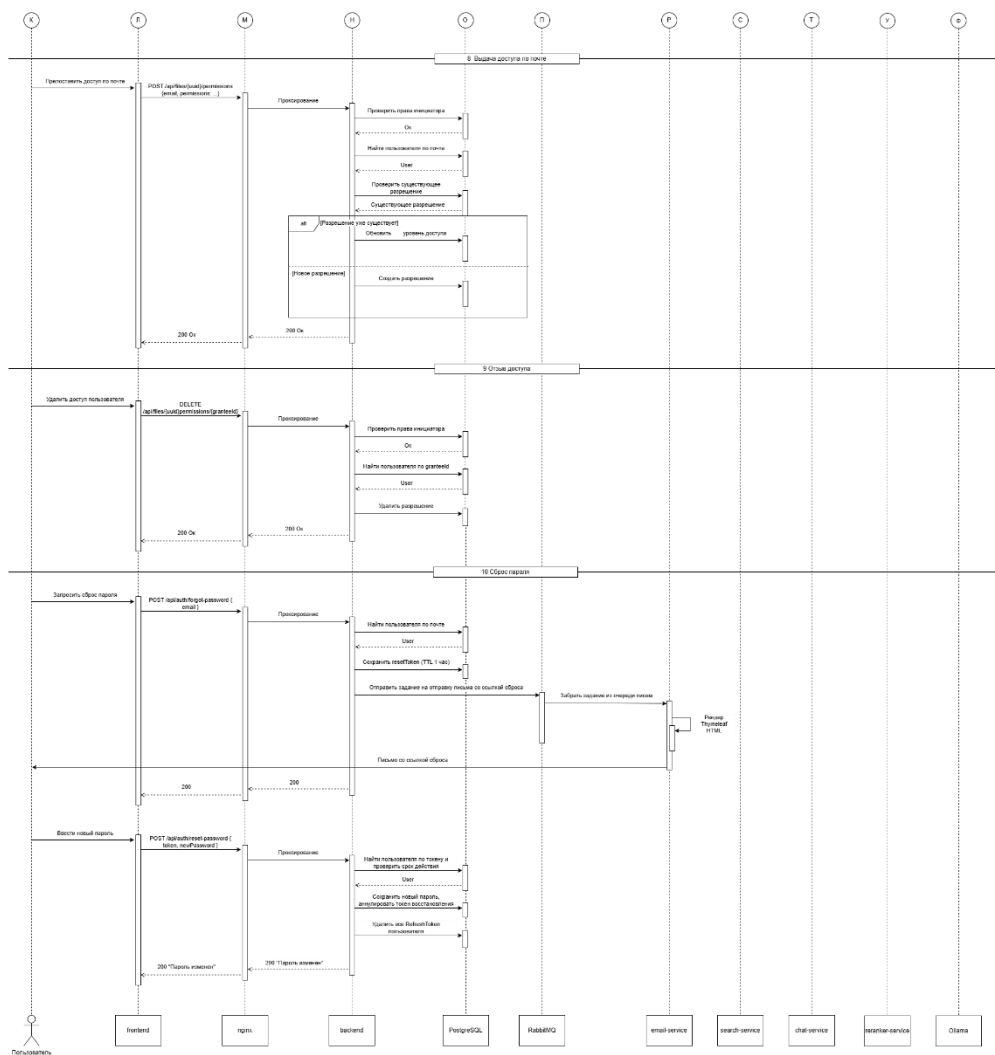
UML – диаграммы

					<i>ПГУ 1.090301.24.001 ПЗ</i>	<i>Лист</i>
						109
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

UML-диаграмма последовательностей

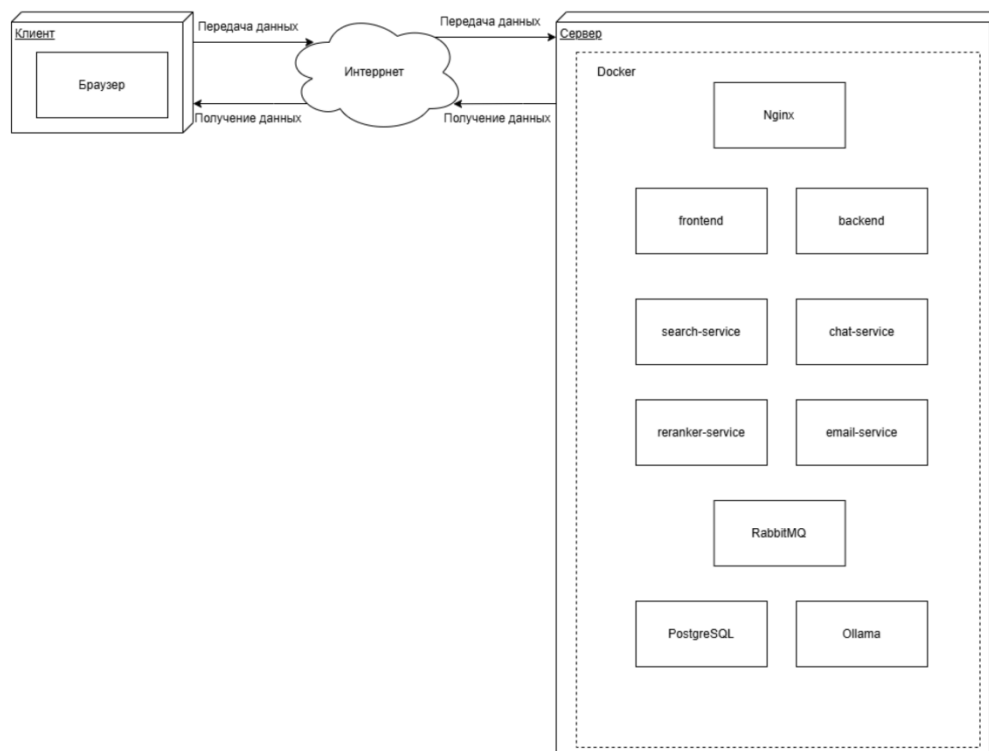


ПГУ 1.090301.24.002 Д14									
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.
Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.	Введ.



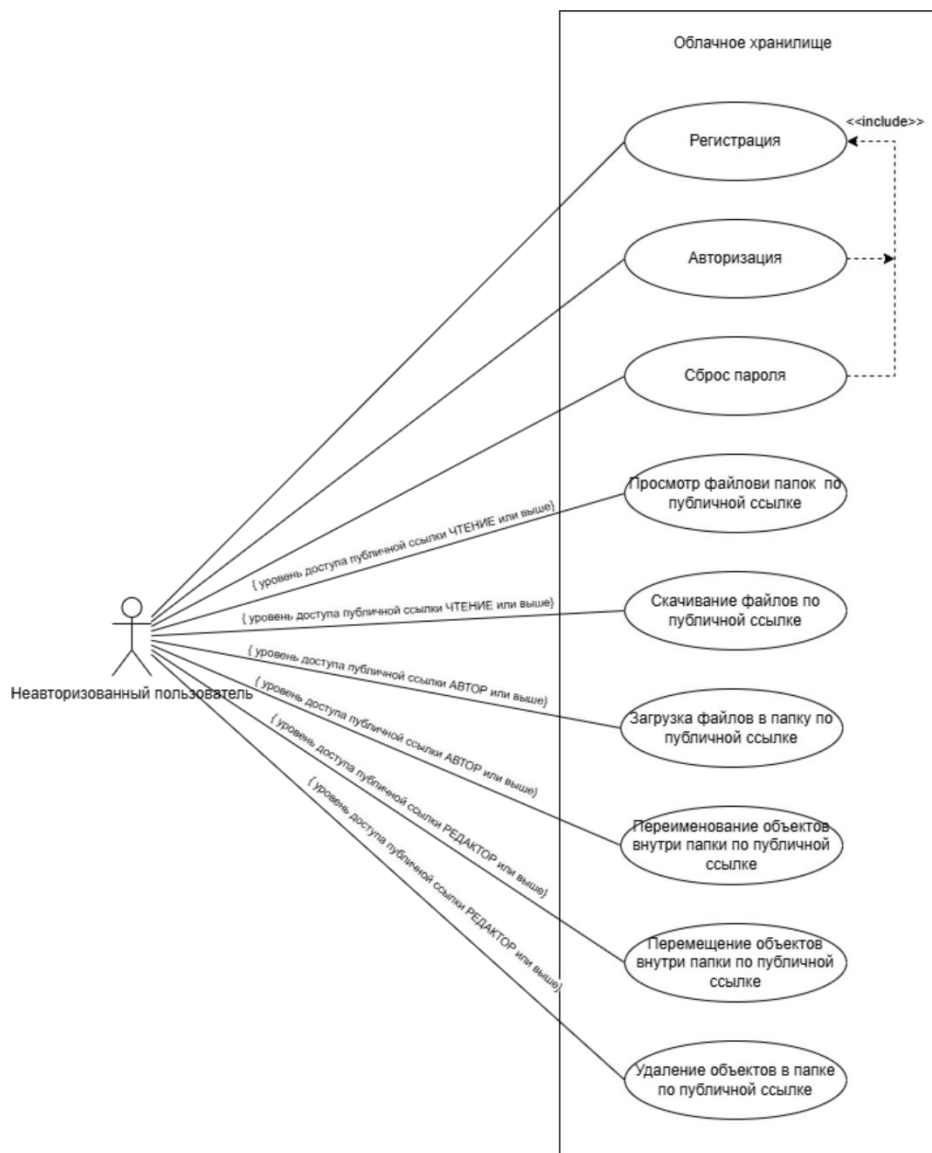
ПГУ 1.090301.24.002 Д14				
Имя	Фамилия	Пол	Дата	Подпись
Роль	Подпись	Дата	Подпись	Дата
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись
Имя	Фамилия	Пол	Дата	Подпись

UML-диаграмма развертывания



				ПГУ 1.090301.24.003 Д19			
Имя	Зачет	По зачету	Подпись	Дата	Обобщенное краткое описание с управленческими функциями и системой поиска UML-дисциплины разрабатываемые		
Григорьев	Семин	Семин			Вс	Мин	Макс
Павлов	Юрков	О.В.					
В.Иванов					Зачет 1	Зачет 2	
И.Иванов	Иванов	А.С.			ФВТ, стр. 22ВВ/ПТ		

UML-диаграмма прецедентов неавторизованного пользователя



ПГУ 1.090301.24.004 Д16				
Изм.	Доп.	Получ.	Получ.	Доп.
Фамилия	Имя	Фамилия	Имя	Фамилия
Т. номер	Объект	Объект	Объект	Объект
В. номер	В. номер	В. номер	В. номер	В. номер
Т. номер	Т. номер	Т. номер	Т. номер	Т. номер

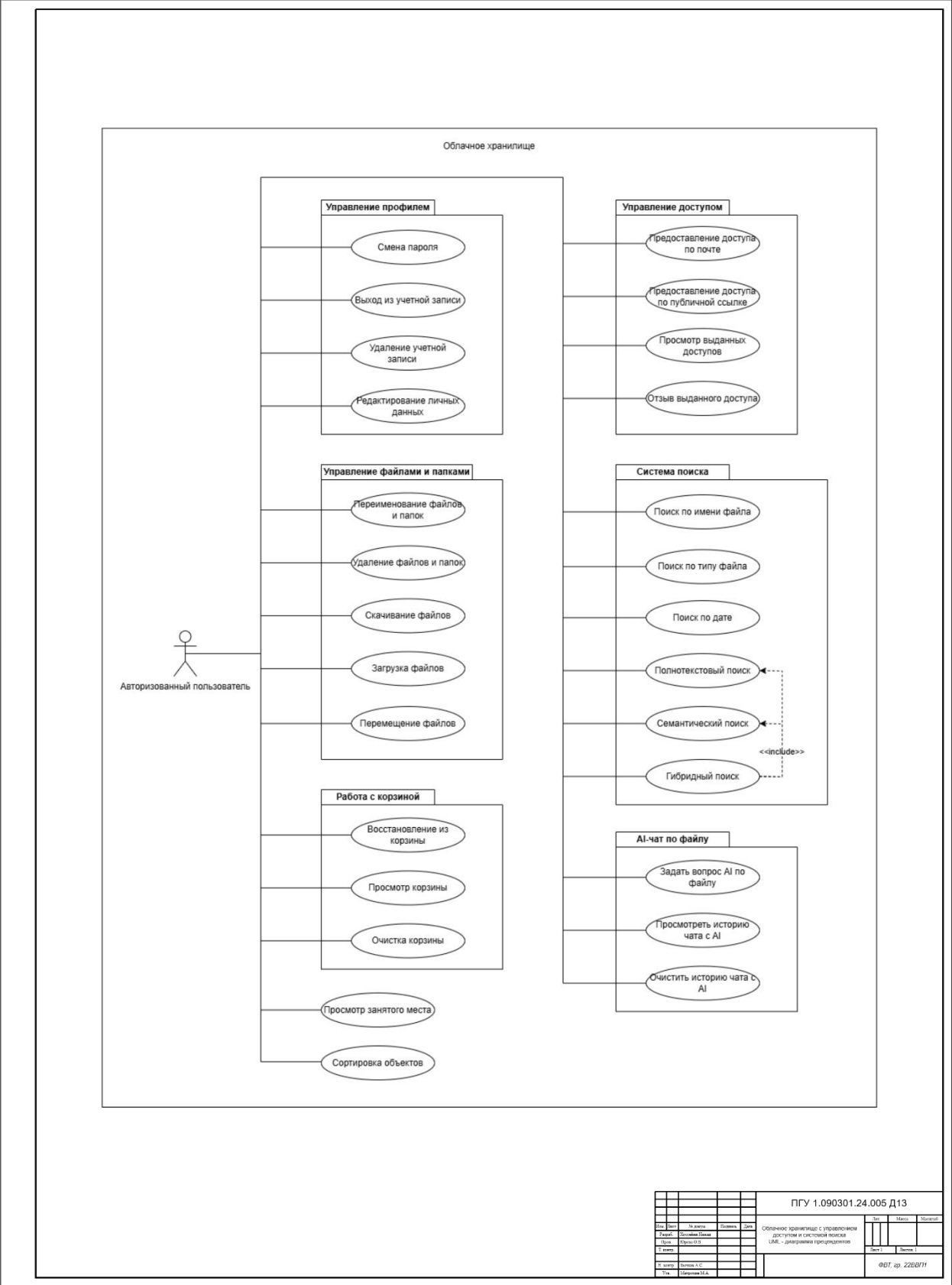
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.24.001 ПЗ

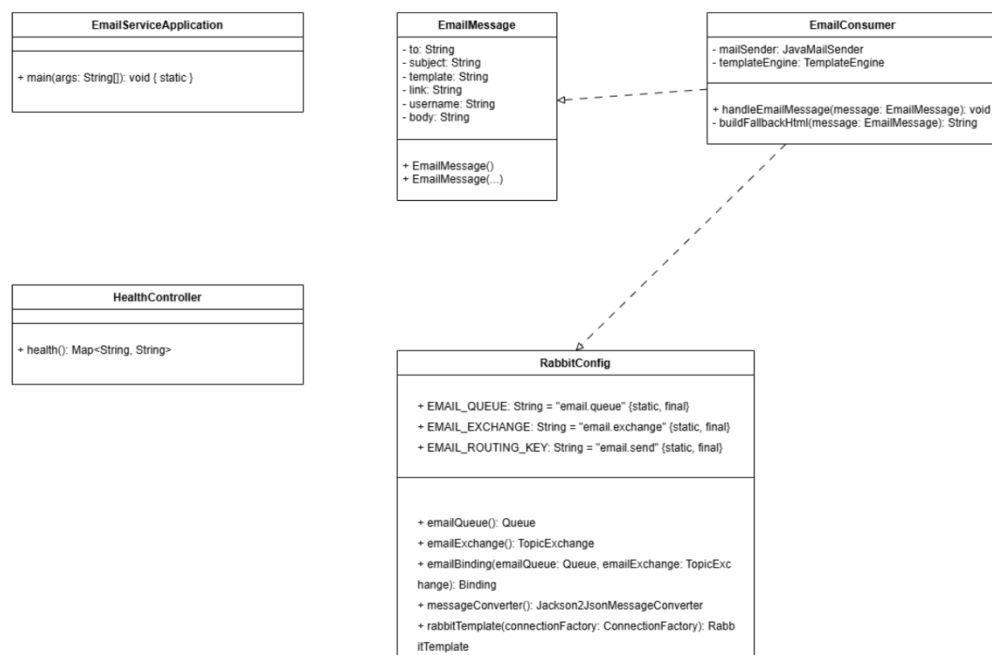
Лист

114

UML-диаграмма прецедентов авторизованного пользователя



UML-диаграмма классов микросервиса email-service



					ПГУ 1.090301.24.006 Д11		
Исполн.	Провер.	Утверд.	Дата	Подпись	Обязательная информация с указанием должности и подписью исполнителя UML-диаграммы классов		
Исполн.	Провер.	Утверд.	Дата	Подпись			
Исполн.	Провер.	Утверд.	Дата	Подпись	Лист 1		
Исполн.	Провер.	Утверд.	Дата	Подпись	ФБП, пр. 22/09/11		

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.24.001 ПЗ

Лист

116

Приложение Б
(обязательное)

Облачное хранилище с управлением доступом и системой поиска
Листинг программы

Листинг микросервиса backend

Файл AuthService.java

```
package com.cloudstorage.service.impl;

import com.cloudstorage.config.RabbitMQConfig;
import com.cloudstorage.dto.AuthDto.*;
import com.cloudstorage.dto.MessageDto.EmailMessage;
import com.cloudstorage.dto.UserDto;
import com.cloudstorage.entity.RefreshToken;
import com.cloudstorage.entity.User;
import com.cloudstorage.exception.ResourceNotFoundException;
import com.cloudstorage.repository.RefreshTokenRepository;
import com.cloudstorage.repository.UserRepository;
import com.cloudstorage.security.JwtUtil;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.UUID;

@Service
@RequiredArgsConstructor
@Slf4j
public class AuthService {

    private final UserRepository userRepository;
    private final RefreshTokenRepository refreshTokenRepository;
    private final PasswordEncoder passwordEncoder;
    private final JwtUtil jwtUtil;
    private final AuthenticationManager authenticationManager;
    private final UserDetailsService userDetailsService;
    private final RabbitTemplate rabbitTemplate;

    @Value("${app.base-url}")
    private String baseUrl;

    @Value("${app.jwt.refresh-expiration}")
    private Long refreshTokenExpiration;

    @Transactional
    public String register(RegisterRequest request) {
        if (userRepository.existsByEmail(request.getEmail())) {
```

```

        throw new IllegalArgumentException("Email already registered");
    }
    if (userRepository.existsByUsername(request.getUsername())) {
        throw new IllegalArgumentException("Username already taken");
    }

    String verificationToken = UUID.randomUUID().toString();

    User user = User.builder()
        .email(request.getEmail())
        .username(request.getUsername())

        .passwordHash(passwordEncoder.encode(request.getPassword()))
        .firstName(request.getFirstName())
        .lastName(request.getLastName())
        .isEmailVerified(false)
        .emailVerificationToken(verificationToken)
        .emailVerificationExpires(Instant.now().plus(24,
ChronoUnit.HOURS))
        .storageUsed(0L)
        .storageLimit(10737418240L)
        .isDeleted(false)
        .build();

    userRepository.save(user);

    String link = baseUrl + "/api/auth/verify-email?token=" +
verificationToken;
    rabbitTemplate.convertAndSend(
        RabbitMQConfig.EMAIL_EXCHANGE,
        RabbitMQConfig.EMAIL_ROUTING_KEY,
        new EmailMessage(user.getEmail(), "Подтверждение email",
            "email-verification", link, user.getUsername(), null)
    );

    return "Registration successful. Please check your email to verify
your account.";
}

@Transactional
public AuthResponse login(LoginRequest request) {
    try {
        authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(request.getEmail(),
request.getPassword())
        );
    } catch (Exception e) {
        throw new BadCredentialsException("Invalid email or password");
    }

    User user = userRepository.findByEmail(request.getEmail())
        .orElseThrow(() -> new ResourceNotFoundException("User not
found"));

    UserDetails userDetails =
userDetailsService.loadUserByUsername(user.getEmail());
    String accessToken = jwtUtil.generateToken(userDetails);

```

					<i>ПГУ 1.090301.24.001 ПЗ</i>	Лист
						119
Изм.	Лист	№ докум.	Подпись	Дата		

```

        String refreshToken = createRefreshToken(user);

        return new AuthResponse(accessToken, refreshToken, mapToDto(user));
    }

    @Transactional
    public AuthResponse refresh(RefreshRequest request) {
        RefreshToken token =
refreshTokenRepository.findByToken(request.getRefreshToken())
            .orElseThrow(() -> new IllegalArgumentException("Invalid
refresh token"));

        if (token.getExpiresAt().isBefore(Instant.now())) {
            refreshTokenRepository.delete(token);
            throw new IllegalArgumentException("Refresh token expired");
        }

        User user = token.getUser();
        UserDetails userDetails =
userService.loadUserByUsername(user.getEmail());
        String newAccessToken = jwtUtil.generateToken(userDetails);

        return new AuthResponse(newAccessToken, request.getRefreshToken(),
mapToDto(user));
    }

    @Transactional
    public void logout(String refreshToken) {
        refreshTokenRepository.deleteByToken(refreshToken);
    }

    @Transactional
    public void verifyEmail(String token) {
        User user = userRepository.findByEmailVerificationToken(token)
            .orElseThrow(() -> new IllegalArgumentException("Invalid
verification token"));

        if (user.getEmailVerificationExpires().isBefore(Instant.now())) {
            throw new IllegalArgumentException("Verification token
expired");
        }

        user.setIsEmailVerified(true);
        user.setEmailVerificationToken(null);
        user.setEmailVerificationExpires(null);
        userRepository.save(user);
    }

    @Transactional
    public void forgotPassword(String email) {
        userRepository.findByEmail(email).ifPresent(user -> {
            String resetToken = UUID.randomUUID().toString();
            user.setPasswordResetToken(resetToken);
            user.setPasswordResetExpires(Instant.now().plus(1,
ChronoUnit.HOURS));
            userRepository.save(user);
        });
    }

```

```

        String link = baseUrl + "/reset-password?token=" + resetToken;
        rabbitTemplate.convertAndSend(
            RabbitMQConfig.EMAIL_EXCHANGE,
            RabbitMQConfig.EMAIL_ROUTING_KEY,
            new EmailMessage(user.getEmail(), "Восстановление пароля",
                "password-reset", link, user.getUsername(), null)
        );
    });
}

@Transactional
public void resetPassword(ResetPasswordRequest request) {
    User user =
userRepository.findByPasswordResetToken(request.getToken())
                .orElseThrow(() -> new IllegalArgumentException("Invalid
reset token"));

    if (user.getPasswordResetExpires().isBefore(Instant.now())) {
        throw new IllegalArgumentException("Reset token expired");
    }

    user.setPasswordHash(passwordEncoder.encode(request.getNewPassword()));
    user.setPasswordResetToken(null);
    user.setPasswordResetExpires(null);
    userRepository.save(user);

    refreshTokenRepository.deleteByUser(user);
}

private String createRefreshToken(User user) {
    refreshTokenRepository.deleteByUser(user);
    RefreshToken token = RefreshToken.builder()
        .user(user)
        .token(UUID.randomUUID().toString())
        .expiresAt(Instant.now().plusSeconds(refreshExpiration /
1000))
        .build();
    return refreshTokenRepository.save(token).getToken();
}

public UserDto mapToDto(User user) {
    return UserDto.builder()
        .id(user.getId())
        .email(user.getEmail())
        .username(user.getUsername())
        .firstName(user.getFirstName())
        .lastName(user.getLastName())
        .avatarUrl(user.getAvatarUrl())
        .isEmailVerified(user.getIsEmailVerified())
        .storageUsed(user.getStorageUsed())
        .storageLimit(user.getStorageLimit())
        .createdAt(user.getCreatedAt())
        .build();
}
}

```

Файл UserService.java

```
package com.cloudstorage.service.impl;

import com.cloudstorage.dto.FileOperationDto.*;
import com.cloudstorage.dto.UserDto;
import com.cloudstorage.entity.User;
import com.cloudstorage.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.net.MalformedURLException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Set;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class UserService {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
    private final AuthService authService;

    @Value("${app.storage.path}")
    private String storagePath;

    private static final Set<String> ALLOWED_TYPES = Set.of(
        "image/jpeg", "image/png", "image/gif", "image/webp"
    );
    private static final long MAX_AVATAR_SIZE = 5 * 1024 * 1024;

    public UserDto getProfile(User user) {
        User fresh = userRepository.findById(user.getId()).orElseThrow();
        return authService.mapToDto(fresh);
    }

    @Transactional
    public UserDto updateProfile(User user, UpdateProfileRequest request) {
        User fresh = userRepository.findById(user.getId()).orElseThrow();

        if (request.getFirstName() != null)
            fresh.setFirstName(request.getFirstName());
        if (request.getLastName() != null)
            fresh.setLastName(request.getLastName());

        if (request.getUsername() != null &&
            !request.getUsername().equals(fresh.getUsername())) {

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		122

```

        if (userRepository.existsByUsername(request.getUsername())) {
            throw new IllegalArgumentException("Username already
taken");
        }
        fresh.setUsername(request.getUsername());
    }

    userRepository.save(fresh);
    return authService.mapToDto(fresh);
}

@Transactional
public void changePassword(User user, ChangePasswordRequest request) {
    User fresh = userRepository.findById(user.getId()).orElseThrow();

    if (!passwordEncoder.matches(request.getCurrentPassword(),
fresh.getPasswordHash())) {
        throw new IllegalArgumentException("Current password is
incorrect");
    }

    fresh.setPasswordHash(passwordEncoder.encode(request.getNewPassword()));
    userRepository.save(fresh);
}

@Transactional
public void deleteAccount(User user) {
    User fresh = userRepository.findById(user.getId()).orElseThrow();
    fresh.setIsDeleted(true);
    fresh.setEmail(fresh.getEmail() + "_deleted_" +
System.currentTimeMillis());
    userRepository.save(fresh);
}

@Transactional
public UserDto uploadAvatar(User user, MultipartFile file) {
    if (file.isEmpty()) throw new IllegalArgumentException("File is
empty");
    if (file.getSize() > MAX_AVATAR_SIZE) throw new
IllegalArgumentException("File too large (max 5 MB)");

    String contentType = file.getContentType();
    if (contentType == null || !ALLOWED_TYPES.contains(contentType)) {
        throw new IllegalArgumentException("Only JPEG, PNG, GIF and
WEBP images are allowed");
    }

    String ext = switch (contentType) {
        case "image/png" -> ".png";
        case "image/gif" -> ".gif";
        case "image/webp" -> ".webp";
        default -> ".jpg";
    };

    String filename = "avatar_" + user.getId() + "_" +
UUID.randomUUID() + ext;

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		123

```

        Path avatarsDir = Paths.get(storagePath, "avatars");

        try {
            Files.createDirectories(avatarsDir);
            Path dest = avatarsDir.resolve(filename);
            file.transferTo(dest.toFile());
        } catch (IOException e) {
            throw new RuntimeException("Failed to save avatar", e);
        }

        User fresh = userRepository.findById(user.getId()).orElseThrow();

        if (fresh.getAvatarUrl() != null) {
            try {
                String oldFilename =
fresh.getAvatarUrl().substring(fresh.getAvatarUrl().lastIndexOf('/') + 1);
                Files.deleteIfExists(avatarsDir.resolve(oldFilename));
            } catch (IOException ignored) {}
        }

        fresh.setAvatarUrl("/api/users/avatars/" + filename);
        userRepository.save(fresh);
        return authService.mapToDto(fresh);
    }

    @Transactional
    public UserDto deleteAvatar(User user) {
        User fresh = userRepository.findById(user.getId()).orElseThrow();

        if (fresh.getAvatarUrl() != null) {
            try {
                String oldFilename =
fresh.getAvatarUrl().substring(fresh.getAvatarUrl().lastIndexOf('/') + 1);
                Path avatarsDir = Paths.get(storagePath, "avatars");
                Files.deleteIfExists(avatarsDir.resolve(oldFilename));
            } catch (IOException ignored) {}
            fresh.setAvatarUrl(null);
            userRepository.save(fresh);
        }

        return authService.mapToDto(fresh);
    }

    public Resource getAvatarResource(String filename) {

        if (filename.contains("..") || filename.contains("/") ||
filename.contains("\\\\")) {
            throw new IllegalArgumentException("Invalid filename");
        }
        Path file = Paths.get(storagePath,
"avatars").resolve(filename).normalize();
        try {
            Resource resource = new UrlResource(file.toUri());
            if (!resource.exists() || !resource.isReadable()) {
                throw new RuntimeException("Avatar not found");
            }
        }
        return resource;
    }

```

```

        } catch (MalformedURLException e) {
            throw new RuntimeException("Avatar not found", e);
        }
    }
}

```

Файл FileService.java

```
package com.cloudstorage.service.impl;
```

```

import com.cloudstorage.config.RabbitMQConfig;
import com.cloudstorage.dto.FileOperationDto.*;
import com.cloudstorage.dto.MessageDto.SearchIndexMessage;
import com.cloudstorage.dto.StorageObjectDto;
import com.cloudstorage.entity.*;
import com.cloudstorage.entity.AccessPermission.PermissionLevel;
import com.cloudstorage.exception.AccessDeniedException;
import com.cloudstorage.exception.ResourceNotFoundException;
import com.cloudstorage.exception.StorageLimitException;
import com.cloudstorage.repository.*;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.transaction.support.TransactionSynchronization;
import org.springframework.transaction.support.TransactionSynchronizationManager;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.time.Instant;
import java.util.*;

```

```
@Service
```

```
@RequiredArgsConstructor
```

```
@Slf4j
```

```
public class FileService {
```

```

    private final StorageObjectRepository storageObjectRepository;
    private final AccessPermissionRepository permissionRepository;
    private final TrashItemRepository trashItemRepository;
    private final UserRepository userRepository;
    private final RabbitTemplate rabbitTemplate;
    private final com.cloudstorage.service.SearchServiceClient
searchServiceClient;

```

```
    @Value("${app.storage.path}")
```

```
    private String storagePath;
```

```
    @Transactional(readOnly = true)
```

```

    public List<StorageObjectDto> listFolder(User user, UUID parentUuid) {
        List<StorageObject> objects;
        if (parentUuid == null) {
            objects = storageObjectRepository.findRootByOwnerFetched(user);

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

```

    } else {
        StorageObject parent = getObjectOrThrow(parentUuid);
        checkPermission(user, parent, PermissionLevel.READ);

        objects = storageObjectRepository.findByParentFetched(parent);
    }
    return objects.stream().map(o -> toDto(o, user)).toList();
}

@Transactional(readOnly = true)
public List<StorageObjectDto> listSharedWithMe(User user) {
    return storageObjectRepository.findSharedWithUser(user)
        .stream().map(o -> toDto(o, user)).toList();
}

@Transactional
public StorageObjectDto createFolder(User user, CreateFolderRequest
request) {
    StorageObject parent = null;
    if (request.getParentUuid() != null) {
        parent = getObjectOrThrow(request.getParentUuid());
        checkPermission(user, parent, PermissionLevel.CONTRIBUTE);
    }
    validateName(request.getName());

    StorageObject folder = StorageObject.builder()
        .owner(user)
        .parent(parent)
        .name(request.getName())
        .objectType(StorageObject.ObjectType.FOLDER)
        .size(0L)
        .isDeleted(false)
        .build();

    storageObjectRepository.save(folder);
    return toDto(folder, user);
}

@Transactional
public StorageObjectDto uploadFile(User user, UUID parentUuid,
MultipartFile file) throws IOException {

    long fileSize = file.getSize();
    User freshUser =
userRepository.findById(user.getId()).orElseThrow();
    if (freshUser.getStorageUsed() + fileSize >
freshUser.getStorageLimit()) {
        throw new StorageLimitException("Storage limit exceeded");
    }

    StorageObject parent = null;
    if (parentUuid != null) {
        parent = getObjectOrThrow(parentUuid);
        checkPermission(user, parent, PermissionLevel.CONTRIBUTE);
    }

```

```

        String originalName = file.getOriginalFilename() != null ?
file.getOriginalFilename() : "unnamed";
        String relativePath = buildStoragePath(freshUser.getId(),
originalName);
        Path fullPath = Paths.get(storagePath, relativePath);
        Files.createDirectories(fullPath.getParent());

        try (InputStream in = file.getInputStream()) {
            Files.copy(in, fullPath, StandardCopyOption.REPLACE_EXISTING);
        }

        StorageObject obj = StorageObject.builder()
            .owner(freshUser)
            .parent(parent)
            .name(originalName)
            .objectType(StorageObject.ObjectType.FILE)
            .mimeType(file.getContentType())
            .size(fileSize)
            .storagePath(relativePath)
            .isDeleted(false)
            .build();

        storageObjectRepository.save(obj);

        freshUser.setStorageUsed(freshUser.getStorageUsed() + fileSize);
        userRepository.save(freshUser);

        final SearchIndexMessage indexMsg = new SearchIndexMessage(
            obj.getId(), "INDEX", relativePath, file.getContentType(),
originalName);
        TransactionSynchronizationManager.registerSynchronization(new
TransactionSynchronization() {
            @Override
            public void afterCommit() {
                try {
                    rabbitTemplate.convertAndSend(
                        RabbitMQConfig.SEARCH_INDEX_EXCHANGE,
                        RabbitMQConfig.SEARCH_INDEX_ROUTING_KEY,
                        indexMsg
                    );
                } catch (Exception e) {
                    log.warn("Could not send search index message: {}",
e.getMessage());
                }
            }
        });

        return toDto(obj, freshUser);
    }

    @Transactional(readOnly = true)
    public Path getFilePath(User user, UUID uuid) {
        StorageObject obj = getObjectOrThrow(uuid);
        checkPermission(user, obj, PermissionLevel.READ);
        if (obj.getObjectType() != StorageObject.ObjectType.FILE) {
            throw new IllegalArgumentException("Not a file");
        }
    }

```

```

        return Paths.get(storagePath, obj.getStoragePath());
    }

    @Transactional(readOnly = true)
    public StorageObject getObject(User user, UUID uuid) {
        StorageObject obj = getObjectOrThrow(uuid);
        checkPermission(user, obj, PermissionLevel.READ);
        return obj;
    }

    @Transactional
    public StorageObjectDto rename(User user, UUID uuid, RenameRequest
request) {
        StorageObject obj = getObjectOrThrow(uuid);
        checkPermission(user, obj, PermissionLevel.CONTRIBUTE);
        validateName(request.getNewName());
        obj.setName(request.getNewName());
        storageObjectRepository.save(obj);
        return toDto(obj, user);
    }

    @Transactional
    public StorageObjectDto move(User user, UUID uuid, MoveRequest request)
{
        StorageObject obj = getObjectOrThrow(uuid);
        checkPermission(user, obj, PermissionLevel.WRITE);

        StorageObject newParent = null;
        if (request.getTargetParentUuid() != null) {
            newParent = getObjectOrThrow(request.getTargetParentUuid());
            checkPermission(user, newParent, PermissionLevel.WRITE);
        }

        obj.setParent(newParent);
        storageObjectRepository.save(obj);
        return toDto(obj, user);
    }

    @Transactional
    public StorageObjectDto copy(User user, UUID uuid, CopyRequest request)
throws IOException {
        StorageObject original = getObjectOrThrow(uuid);
        checkPermission(user, original, PermissionLevel.READ);

        StorageObject targetParent = null;
        if (request.getTargetParentUuid() != null) {
            targetParent = getObjectOrThrow(request.getTargetParentUuid());
            checkPermission(user, targetParent,
PermissionLevel.CONTRIBUTE);
        }

        return copyRecursive(user, original, targetParent);
    }

    private StorageObjectDto copyRecursive(User user, StorageObject source,
StorageObject targetParent) throws IOException {
        String newName = source.getName();

```

```

        StorageObject copy = StorageObject.builder()
            .owner(user)
            .parent(targetParent)
            .name(newName)
            .objectType(source.getObjectType())
            .mimeType(source.getMimeType())
            .isDeleted(false)
            .build();

        if (source.getObjectType() == StorageObject.ObjectType.FILE &&
source.getStoragePath() != null) {
            User freshUser =
userRepository.findById(user.getId()).orElseThrow();
            if (freshUser.getStorageUsed() + source.getSize() >
freshUser.getStorageLimit()) {
                throw new StorageLimitException("Storage limit exceeded
during copy");
            }
            String newRelPath = buildStoragePath(user.getId(),
UUID.randomUUID() + "_" + source.getName());
            Path srcPath = Paths.get(storagePath, source.getStoragePath());
            Path dstPath = Paths.get(storagePath, newRelPath);
            Files.createDirectories(dstPath.getParent());
            Files.copy(srcPath, dstPath,
StandardCopyOption.REPLACE_EXISTING);
            copy.setStoragePath(newRelPath);
            copy.setSize(source.getSize());
        }

        storageObjectRepository.save(copy);

        if (source.getObjectType() == StorageObject.ObjectType.FOLDER) {
            List<StorageObject> children = storageObjectRepository
                .findByOwnerAndParentAndIsDeletedFalse(user, source);
            for (StorageObject child : children) {
                copyRecursive(user, child, copy);
            }
        }

        return toDto(copy, user);
    }

    @Transactional
    public void deleteToTrash(User user, UUID uuid) {
        StorageObject obj = getObjectOrThrow(uuid);
        checkPermission(user, obj, PermissionLevel.WRITE);

        Long originalParentId = obj.getParent() != null ?
obj.getParent().getId() : null;
        markDeletedRecursive(obj);

        TrashItem trash = TrashItem.builder()
            .object(obj)
            .user(user)
            .originalParentId(originalParentId)
            .build();
    }

```

```

        trashItemRepository.save(trash);
    }

    private void markDeletedRecursive(StorageObject obj) {
        obj.setIsDeleted(true);
        obj.setDeletedAt(Instant.now());
        storageObjectRepository.save(obj);

        if (obj.getObjectType() == StorageObject.ObjectType.FOLDER) {

            List<StorageObject> children =
storageObjectRepository.findAllDescendants(obj.getId());
            for (StorageObject child : children) {
                if (!child.getId().equals(obj.getId())) {
                    child.setIsDeleted(true);
                    child.setDeletedAt(Instant.now());
                    storageObjectRepository.save(child);
                }
            }
        }

        rabbitTemplate.convertAndSend(
            RabbitMQConfig.SEARCH_INDEX_EXCHANGE,
            RabbitMQConfig.SEARCH_INDEX_ROUTING_KEY,
            new SearchIndexMessage(obj.getId(), "DELETE", null, null, null)
        );
    }

    @Transactional(readOnly = true)
    public List<StorageObjectDto> listTrash(User user) {
        return trashItemRepository.findByUserOrderByDeletedAtDesc(user)
            .stream()
            .map(t -> toDto(t.getObject(), user))
            .toList();
    }

    @Transactional
    public StorageObjectDto restoreFromTrash(User user, UUID uuid) {
        StorageObject obj = getDeletedObjectOrThrow(uuid);
        TrashItem trashItem =
trashItemRepository.findById(obj.getId())
            .orElseThrow(() -> new ResourceNotFoundException("Trash
item not found"));

        obj.setIsDeleted(false);
        obj.setDeletedAt(null);

        if (trashItem.getOriginalParentId() != null) {

storageObjectRepository.findById(trashItem.getOriginalParentId()).ifPresent
OrElse(
            parent -> {
                if (!parent.getIsDeleted()) obj.setParent(parent);
                else obj.setParent(null);
            },
            () -> obj.setParent(null)
        );
    }

```

```

    }

    storageObjectRepository.save(obj);
    trashItemRepository.delete(trashItem);

    return toDto(obj, user);
}

@Transactional
public void emptyTrash(User user) {
    List<TrashItem> items =
trashItemRepository.findByUserOrderByDeletedAtDesc(user);
    for (TrashItem item : items) {
        StorageObject obj = item.getObject();
        trashItemRepository.delete(item);
        trashItemRepository.flush();
        permanentlyDelete(obj);
    }
}

@Transactional
public void permanentlyDeleteFromTrash(User user, UUID uuid) {
    StorageObject obj = getDeletedObjectOrThrow(uuid);
    TrashItem trashItem =
trashItemRepository.findById(obj.getId())
        .orElseThrow(() -> new ResourceNotFoundException("Trash
item not found"));

    trashItemRepository.delete(trashItem);
    trashItemRepository.flush();
    permanentlyDelete(obj);
}

private void permanentlyDelete(StorageObject obj) {
    if (obj.getStoragePath() != null) {
        try {
            Files.deleteIfExists(Paths.get(storagePath,
obj.getStoragePath()));
        } catch (IOException e) {
            log.warn("Could not delete file: {}",
obj.getStoragePath());
        }
    }
    storageObjectRepository.delete(obj);
}

@Transactional(readOnly = true)
public List<StorageObjectDto> search(User user, SearchRequest request)
{
    String type = request.getSearchType();
    boolean isShared = "shared".equals(request.getScope());

    Long folderId = null;
    if (request.getFolderUuid() != null) {
        StorageObject folder =
getObjectOrThrow(request.getFolderUuid());
        checkPermission(user, folder, PermissionLevel.READ);
    }
}

```

```

        folderId = folder.getId();
    }

    String dateFrom = request.getDateFrom() != null ?
request.getDateFrom().toString() : null;
    String dateTo = request.getDateTo() != null ?
request.getDateTo().toString() : null;

    if ("semantic".equals(type)) {
        if (request.getQuery() == null || request.getQuery().isBlank())
return List.of();

        final Long finalFolderId = folderId;
        if (isShared) {
            return semanticOrHybridShared(user, request.getQuery(),
true)

                .stream()
                .filter(dto -> finalFolderId == null ||
isInSubtreeByUuid(dto.getUuid(), finalFolderId))
                .toList();
        }

        return searchServiceClient.semanticSearch(user.getId(),
request.getQuery())

            .stream()
            .map(r -> r.getUuid() != null
?
storageObjectRepository.findByUuidAndIsDeletedFalse(r.getUuid()).orElse(nul
l)
: null)
            .filter(Objects::nonNull)
            .filter(o -> o.getOwner().getId().equals(user.getId()))
            .filter(o -> finalFolderId == null || isInSubtree(o,
finalFolderId))

            .map(o -> toDto(o, user))
            .toList();
    }

    if ("name".equals(type)) {
        String q = (request.getQuery() != null &&
!request.getQuery().isBlank())
? request.getQuery() : null;
        if (q == null) return List.of();
        List<StorageObject> nameResults;
        if (isShared && folderId != null) {
            nameResults =
storageObjectRepository.searchByNameSharedInFolder(folderId, q,
request.getMimeType(), dateFrom, dateTo);
        } else if (isShared) {
            nameResults =
storageObjectRepository.searchByNameShared(user.getId(), q,
request.getMimeType(), dateFrom, dateTo);
        } else if (folderId != null) {
            nameResults =
storageObjectRepository.searchByNameInFolder(user.getId(), folderId, q,
request.getMimeType(), dateFrom, dateTo);
        } else {

```

```

        nameResults =
storageObjectRepository.searchByName(user.getId(), q,
request.getMimeType(), dateFrom, dateTo);
    }
    return nameResults.stream()
        .sorted(buildComparator(request.getSortBy(),
request.getSortDir()))
        .map(o -> toDto(o, user))
        .toList();
    }

    if ("all".equals(type)) {
        if (request.getQuery() == null || request.getQuery().isBlank())
return List.of();

        final Long finalFolderId = folderId;
        if (isShared) {
            return semanticOrHybridShared(user, request.getQuery(),
false)
                .stream()
                .filter(dto -> finalFolderId == null ||
isInSubtreeByUuid(dto.getUuid(), finalFolderId))
                .toList();
        }

        return searchServiceClient.hybridSearch(user.getId(),
request.getQuery())
            .stream()
            .map(r -> r.getUuid() != null
                ?
storageObjectRepository.findByUuidAndIsDeletedFalse(r.getUuid()).orElse(nul
l)
                : null)
            .filter(Objects.nonNull)
            .filter(o -> o.getOwner().getId().equals(user.getId()))
            .filter(o -> finalFolderId == null || isInSubtree(o,
finalFolderId))
            .map(o -> toDto(o, user))
            .toList();
    }

    String q = (request.getQuery() != null &&
!request.getQuery().isBlank())
        ? request.getQuery() : null;
    List<StorageObject> results;
    if (isShared && folderId != null) {
        results =
storageObjectRepository.advancedSearchSharedInFolder(folderId, q,
request.getMimeType(), dateFrom, dateTo);
    } else if (isShared) {
        results =
storageObjectRepository.advancedSearchShared(user.getId(), q,
request.getMimeType(), dateFrom, dateTo);
    } else if (folderId != null) {
        results =
storageObjectRepository.advancedSearchInFolder(user.getId(), folderId, q,
request.getMimeType(), dateFrom, dateTo);
    }

```

```

        } else {
            results = storageObjectRepository.advancedSearch(user.getId(),
q, request.getMimeType(), dateFrom, dateTo);
        }

        return results.stream()
            .sorted(buildComparator(request.getSortBy(),
request.getSortDir()))
            .map(o -> toDto(o, user))
            .toList();
    }

    private boolean isInSubtree(StorageObject obj, Long folderId) {
        StorageObject current = obj;
        int depth = 0;
        while (current != null && depth < 50) {
            if (current.getId().equals(folderId)) return true;
            current = current.getParent();
            depth++;
        }
        return false;
    }

    private boolean isInSubtreeByUuid(UUID objUuid, Long folderId) {
        return storageObjectRepository.findByUuidAndIsDeletedFalse(objUuid)
            .map(o -> isInSubtree(o, folderId))
            .orElse(false);
    }

    private List<StorageObjectDto> semanticOrHybridShared(User user, String
query, boolean semantic) {

        List<StorageObject> accessible =
storageObjectRepository.findAllAccessibleForUser(user.getId());
        if (accessible.isEmpty()) return List.of();

        Set<UUID> accessibleUuids = accessible.stream()
            .map(StorageObject::getUuid)
            .collect(java.util.stream.Collectors.toSet());

        Set<Long> ownerIds = accessible.stream()
            .map(o -> o.getOwner().getId())
            .collect(java.util.stream.Collectors.toSet());

        return ownerIds.stream()
            .flatMap(ownerId -> {
                List<com.cloudstorage.dto.SemanticSearchResult> raw =
semantic
                    ? searchServiceClient.semanticSearch(ownerId,
query)
                    : searchServiceClient.hybridSearch(ownerId,
query);
                return raw.stream()
                    .filter(r -> r.getUuid() != null &&
accessibleUuids.contains(r.getUuid()));
            })
            .distinct()
    }

```

```

        .map(r ->
storageObjectRepository.findByUuidAndIsDeletedFalse(r.getUuid()).orElse(nul
l))
        .filter(Objects::nonNull)
        .map(o -> toDto(o, user))
        .toList();
    }

    private Comparator<StorageObject> buildComparator(String sortBy, String
sortDir) {
        Comparator<StorageObject> cmp = switch (sortBy != null ? sortBy :
"updatedAt") {
            case "name" -> Comparator.comparing(StorageObject::getName,
String.CASE_INSENSITIVE_ORDER);
            case "size" -> Comparator.comparing(o -> o.getSize() != null ?
o.getSize() : 0L);
            case "mimeType" -> Comparator.comparing(o -> o.getMimeType() !=
null ? o.getMimeType() : "");
            default -> Comparator.comparing(o -> o.getUpdatedAt() != null ?
o.getUpdatedAt() : Instant.EPOCH);
        };
        if ("asc".equalsIgnoreCase(sortDir)) return cmp;
        return cmp.reversed();
    }

    @Transactional
    public void grantPermission(User granter, UUID objectId,
GrantPermissionRequest request) {
        StorageObject obj = getObjectOrThrow(objectId);
        checkPermission(granter, obj, PermissionLevel.OWNER);

        User grantee = userRepository.findByEmail(request.getEmail())
            .orElseThrow(() -> new ResourceNotFoundException("User not
found: " + request.getEmail()));

        if (grantee.getId().equals(granter.getId())) {
            throw new IllegalArgumentException("Cannot grant permission to
yourself");
        }

        AccessPermission existing =
permissionRepository.findByObjectAndGrantee(obj, grantee).orElse(null);
        if (existing != null) {
            existing.setPermissionLevel(request.getPermission());
            permissionRepository.save(existing);
        } else {
            AccessPermission perm = AccessPermission.builder()
                .object(obj)
                .grantee(grantee)
                .permissionLevel(request.getPermission())
                .grantedBy(granter)
                .build();
            permissionRepository.save(perm);
        }
    }

    @Transactional

```

```

    public void revokePermission(User revoker, UUID objectId, Long
granteeId) {
        StorageObject obj = getObjectOrThrow(objectId);
        checkPermission(revoker, obj, PermissionLevel.OWNER);
        User grantee = userRepository.findById(granteeId)
            .orElseThrow(() -> new ResourceNotFoundException("User not
found"));
        permissionRepository.deleteByObjectAndGrantee(obj, grantee);
    }

    @Transactional(readonly = true)
    public List<PermissionDto> listPermissions(User user, UUID objectId)
    {
        StorageObject obj = getObjectOrThrow(objectId);
        checkPermission(user, obj, PermissionLevel.OWNER);
        return permissionRepository.findByObject(obj).stream().map(p -> {
            PermissionDto dto = new PermissionDto();
            dto.setId(p.getId());
            dto.setObjectId(obj.getId());
            dto.setObjectUuid(obj.getUuid());
            dto.setGranteeId(p.getGrantee().getId());
            dto.setGranteeEmail(p.getGrantee().getEmail());
            dto.setGranteeUsername(p.getGrantee().getUsername());
            dto.setGranteeAvatarUrl(p.getGrantee().getAvatarUrl());
            dto.setPermissionLevel(p.getPermissionLevel());
            dto.setGrantedAt(p.getGrantedAt());
            return dto;
        }).toList();
    }

    public StorageObject getObjectOrThrow(UUID uuid) {
        return storageObjectRepository.findByUuidAndIsDeletedFalse(uuid)
            .orElseThrow(() -> new ResourceNotFoundException("Object
not found: " + uuid));
    }

    public StorageObject getDeletedObjectOrThrow(UUID uuid) {
        return storageObjectRepository.findByUuid(uuid)
            .orElseThrow(() -> new ResourceNotFoundException("Object
not found: " + uuid));
    }

    public void checkPermission(User user, StorageObject obj,
PermissionLevel required) {

        PermissionLevel effective = getEffectivePermission(user, obj);
        if (effective == null) {

            throw new AccessDeniedException("Access denied");
        }
        if (!effective.includes(required)) {
            throw new AccessDeniedException("Insufficient permissions");
        }
    }

    public PermissionLevel getEffectivePermission(User user, StorageObject
obj) {

```

					ИПУ 1.090301.24.001 ИЗ	Лист
						136
Изм.	Лист	№ докум.	Подпись	Дата		

```

        boolean isObjectOwner =
obj.getOwner().getId().equals(user.getId());

        PermissionLevel inheritedCap = null;
        StorageObject current = obj.getParent();
        int depth = 0;
        while (current != null && depth < 50) {
            if (!current.getOwner().getId().equals(user.getId())) {

                Optional<AccessPermission> perm =
permissionRepository.findByObjectAndGrantee(current, user);
                if (perm.isPresent()) {
                    inheritedCap = perm.get().getPermissionLevel();
                    break;
                }

            }

            current = current.getParent();
            depth++;
        }

        if (isObjectOwner) {
            return inheritedCap != null ? inheritedCap :
PermissionLevel.OWNER;
        }

        Optional<AccessPermission> directPerm =
permissionRepository.findByObjectAndGrantee(obj, user);
        if (directPerm.isPresent()) return
directPerm.get().getPermissionLevel();
        return inheritedCap;
    }

    private void validateName(String name) {
        if (name == null || name.isBlank()) throw new
IllegalArgumentException("Name cannot be blank");
        if (name.contains("/") || name.contains("\\\\"))
            throw new IllegalArgumentException("Name cannot contain
slashes");
        if (name.length() > 255) throw new IllegalArgumentException("Name
too long");
    }

    private String buildStoragePath(Long userId, String filename) {
        return "users/" + userId + "/" + UUID.randomUUID() + "_" +
sanitizeFilename(filename);
    }

    private String sanitizeFilename(String name) {
        return name.replaceAll("[^a-zA-Z0-9._\\-]", "_");
    }

    public StorageObjectDto toDto(StorageObject obj, User currentUser) {
        PermissionLevel myPerm = getEffectivePermission(currentUser, obj);
        return StorageObjectDto.builder()
            .id(obj.getId())

```

					ИПУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		137

```

        .uuid(obj.getUuid())
        .name(obj.getName())
        .objectType(obj.getObjectType())
        .mimeType(obj.getMimeType())
        .size(obj.getSize())
        .parentId(obj.getParent() != null ? obj.getParent().getId()
: null)
        .parentUuid(obj.getParent() != null ?
obj.getParent().getUuid() : null)
        .ownerId(obj.getOwner().getId())
        .ownerUsername(obj.getOwner().getUsername())
        .createdAt(obj.getCreatedAt())
        .updatedAt(obj.getUpdatedAt())
        .myPermission(myPerm)

.isOwner(obj.getOwner().getId().equals(currentUser.getId()))
        .build();
    }
}

```

Файл ShareLinkService.java

```

package com.cloudstorage.service.impl;

import com.cloudstorage.dto.FileOperationDto.*;
import com.cloudstorage.entity.*;
import com.cloudstorage.entity.AccessPermission.PermissionLevel;
import com.cloudstorage.exception.ResourceNotFoundException;
import com.cloudstorage.repository.ShareLinkRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.Instant;
import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class ShareLinkService {

    private final ShareLinkRepository shareLinkRepository;
    private final FileService fileService;

    @Value("${app.base-url}")
    private String baseUrl;

    @Transactional
    public ShareLinkDto createShareLink(User user, UUID objectUuid,
ShareLinkRequest request) {
        StorageObject obj = fileService.getObjectOrThrow(objectUuid);
        fileService.checkPermission(user, obj, PermissionLevel.OWNER);

        PermissionLevel permission = request.getPermission() != null ?
request.getPermission() : PermissionLevel.READ;
        if (permission == PermissionLevel.OWNER) {
            throw new IllegalArgumentException("Cannot create a public

```

```

share link with OWNER permission");
    }

    ShareLink link = ShareLink.builder()
        .token(UUID.randomUUID().toString())
        .object(obj)
        .createdBy(user)
        .permissionLevel(permission)
        .expiresAt(request.getExpiresAt())
        .isActive(true)
        .build();

    shareLinkRepository.save(link);
    return toDto(link);
}

public List<ShareLinkDto> listShareLinks(User user, UUID objectUuid) {
    StorageObject obj = fileService.getObjectOrThrow(objectUuid);
    fileService.checkPermission(user, obj, PermissionLevel.OWNER);
    return
shareLinkRepository.findByObjectAndIsActiveTrue(obj).stream().map(this::toD
to).toList();
}

@Transactional
public void deactivateShareLink(User user, Long linkId) {
    ShareLink link = shareLinkRepository.findById(linkId)
        .orElseThrow(() -> new ResourceNotFoundException("Share
link not found"));
    fileService.checkPermission(user, link.getObject(),
PermissionLevel.OWNER);
    link.setIsActive(false);
    shareLinkRepository.save(link);
}

public ShareLink resolvePublicToken(String token) {
    ShareLink link =
shareLinkRepository.findByTokenAndIsActiveTrue(token)
        .orElseThrow(() -> new ResourceNotFoundException("Share
link not found or inactive"));
    if (link.getExpiresAt() != null &&
link.getExpiresAt().isBefore(Instant.now())) {
        throw new ResourceNotFoundException("Share link expired");
    }
    return link;
}

private ShareLinkDto toDto(ShareLink link) {
    ShareLinkDto dto = new ShareLinkDto();
    dto.setId(link.getId());
    dto.setToken(link.getToken());
    dto.setUrl(baseUrl + "/s/" + link.getToken());
    dto.setObjectUuid(link.getObject().getUuid());
    dto.setObjectName(link.getObject().getName());
    dto.setPermissionLevel(link.getPermissionLevel());
    dto.setExpiresAt(link.getExpiresAt());
    dto.setCreatedAt(link.getCreatedAt());
}

```

```

        dto.setIsActive(link.getIsActive());
        return dto;
    }
}

```

Файл TrashCleanupService.java

```

package com.cloudstorage.service.impl;

import com.cloudstorage.entity.TrashItem;
import com.cloudstorage.repository.TrashItemRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.Instant;
import java.util.List;

@Service
@RequiredArgsConstructor
@Slf4j
public class TrashCleanupService {

    private final TrashItemRepository trashItemRepository;
    private final FileService fileService;

    @Scheduled(cron = "0 0 3 * * *")
    @Transactional
    public void cleanupExpiredTrash() {
        List<TrashItem> expired =
trashItemRepository.findByAutoDeleteAtBefore(Instant.now());
        log.info("Auto-deleting {} expired trash items", expired.size());
        for (TrashItem item : expired) {
            try {
                fileService.permanentlyDeleteFromTrash(item.getUser(),
item.getObject().getUuid());
            } catch (Exception e) {
                log.warn("Failed to auto-delete trash item {}: {}",
item.getId(), e.getMessage());
            }
        }
    }
}

```

Файл FileController.java

```

package com.cloudstorage.controller;

import com.cloudstorage.dto.ApiResponse;
import com.cloudstorage.dto.FileOperationDto.*;
import com.cloudstorage.dto.StorageObjectDto;
import com.cloudstorage.entity.StorageObject;
import com.cloudstorage.entity.User;
import com.cloudstorage.service.impl.FileService;
import com.cloudstorage.util.CurrentUserUtil;
import lombok.RequiredArgsConstructor;
import org.springframework.core.io.FileSystemResource;
import org.springframework.core.io.Resource;

```

					ПГУ 1.090301.24.001 ПЗ	Лист
						140
Изм.	Лист	№ докум.	Подпись	Дата		

```

import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;
import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("/files")
@RequiredArgsConstructor
public class FileController {

    private final FileService fileService;
    private final CurrentUserUtil currentUserUtil;

    @GetMapping
    public ResponseEntity<ApiResponse<List<StorageObjectDto>>> listRoot() {
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.listFolder(user, null)));
    }

    @GetMapping("/folder/{uuid}")
    public ResponseEntity<ApiResponse<List<StorageObjectDto>>>
listFolder(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.listFolder(user, uuid)));
    }

    @GetMapping("/shared")
    public ResponseEntity<ApiResponse<List<StorageObjectDto>>> listShared()
{
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.listSharedWithMe(user)));
    }

    @GetMapping("/{uuid}")
    public ResponseEntity<ApiResponse<StorageObjectDto>>
getObject(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        StorageObject obj = fileService.getObject(user, uuid);
        return ResponseEntity.ok(ApiResponse.ok(fileService.toDto(obj,
user)));
    }

    @PostMapping("/folders")
    public ResponseEntity<ApiResponse<StorageObjectDto>>
createFolder(@RequestBody CreateFolderRequest request) {
        User user = currentUserUtil.getCurrentUser();

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		141

```

        return
        ResponseEntity.ok(ApiResponse.ok(fileService.createFolder(user, request)));
    }

    @PostMapping(value = "/upload", consumes =
MediaType.MULTIPART_FORM_DATA_VALUE)
    public ResponseEntity<ApiResponse<StorageObjectDto>> upload(
        @RequestParam("file") MultipartFile file,
        @RequestParam(required = false) UUID parentUuid) throws
IOException {
        User user = currentUserUtil.getCurrentUser();
        return
        ResponseEntity.ok(ApiResponse.ok(fileService.uploadFile(user, parentUuid,
file)));
    }

    @GetMapping("/{uuid}/download")
    public ResponseEntity<Resource> download(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        StorageObject obj = fileService.getObject(user, uuid);
        Path path = fileService.getFilePath(user, uuid);
        Resource resource = new FileSystemResource(path);

        String encodedName = URLEncoder.encode(obj.getName(),
StandardCharsets.UTF_8).replace("+", "%20");
        return ResponseEntity.ok()
            .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename*=UTF-8'" + encodedName)
            .contentType(obj.getMimeType() != null
                ? MediaType.parseMediaType(obj.getMimeType())
                : MediaType.APPLICATION_OCTET_STREAM)
            .body(resource);
    }

    @PatchMapping("/{uuid}/rename")
    public ResponseEntity<ApiResponse<StorageObjectDto>> rename(
        @PathVariable UUID uuid, @RequestBody RenameRequest request) {
        User user = currentUserUtil.getCurrentUser();
        return ResponseEntity.ok(ApiResponse.ok(fileService.rename(user,
uuid, request)));
    }

    @PostMapping("/{uuid}/move")
    public ResponseEntity<ApiResponse<StorageObjectDto>> move(
        @PathVariable UUID uuid, @RequestBody MoveRequest request) {
        User user = currentUserUtil.getCurrentUser();
        return ResponseEntity.ok(ApiResponse.ok(fileService.move(user,
uuid, request)));
    }

    @PostMapping("/{uuid}/copy")
    public ResponseEntity<ApiResponse<StorageObjectDto>> copy(
        @PathVariable UUID uuid, @RequestBody CopyRequest request)
    throws IOException {
        User user = currentUserUtil.getCurrentUser();
        return ResponseEntity.ok(ApiResponse.ok(fileService.copy(user,
uuid, request)));
    }

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		142

```

    }

    @DeleteMapping("/{uuid}")
    public ResponseEntity<ApiResponse<Void>> delete(@PathVariable UUID
uuid) {
        User user = currentUserUtil.getCurrentUser();
        fileService.deleteToTrash(user, uuid);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }

    @PostMapping("/search")
    public ResponseEntity<ApiResponse<List<StorageObjectDto>>>
search(@RequestBody SearchRequest request) {
        User user = currentUserUtil.getCurrentUser();
        return ResponseEntity.ok(ApiResponse.ok(fileService.search(user,
request)));
    }

    @GetMapping("/trash")
    public ResponseEntity<ApiResponse<List<StorageObjectDto>>> listTrash()
{
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.listTrash(user)));
    }

    @PostMapping("/trash/{uuid}/restore")
    public ResponseEntity<ApiResponse<StorageObjectDto>>
restore(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.restoreFromTrash(user,
uuid)));
    }

    @DeleteMapping("/trash/{uuid}")
    public ResponseEntity<ApiResponse<Void>> permanentDelete(@PathVariable
UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        fileService.permanentlyDeleteFromTrash(user, uuid);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }

    @DeleteMapping("/trash")
    public ResponseEntity<ApiResponse<Void>> emptyTrash() {
        User user = currentUserUtil.getCurrentUser();
        fileService.emptyTrash(user);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }

    @GetMapping("/{uuid}/permissions")
    public ResponseEntity<ApiResponse<List<PermissionDto>>>
listPermissions(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        return
ResponseEntity.ok(ApiResponse.ok(fileService.listPermissions(user, uuid)));
    }

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		143

```

    @PostMapping("/{uuid}/permissions")
    public ResponseEntity<ApiResponse<Void>> grantPermission(
        @PathVariable UUID uuid, @RequestBody GrantPermissionRequest
        request) {
        User user = currentUserUtil.getCurrentUser();
        fileService.grantPermission(user, uuid, request);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }

    @DeleteMapping("/{uuid}/permissions/{granteeId}")
    public ResponseEntity<ApiResponse<Void>> revokePermission(
        @PathVariable UUID uuid, @PathVariable Long granteeId) {
        User user = currentUserUtil.getCurrentUser();
        fileService.revokePermission(user, uuid, granteeId);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }
}

```

Файл AuthController.java

```

package com.cloudstorage.controller;

import com.cloudstorage.dto.ApiResponse;
import com.cloudstorage.dto.AuthDto.*;
import com.cloudstorage.service.impl.AuthService;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/auth")
@RequiredArgsConstructor
public class AuthController {

    private final AuthService authService;

    @Value("${app.frontend-url}")
    private String frontendUrl;

    @PostMapping("/register")
    public ResponseEntity<ApiResponse<String>> register(@Valid @RequestBody
    RegisterRequest request) {
        String msg = authService.register(request);
        return ResponseEntity.ok(ApiResponse.ok(msg));
    }

    @PostMapping("/login")
    public ResponseEntity<ApiResponse<AuthResponse>> login(@Valid
    @RequestBody LoginRequest request) {
        return
        ResponseEntity.ok(ApiResponse.ok(authService.login(request)));
    }
}

```

					ИПУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		144

```

        @PostMapping("/refresh")
        public ResponseEntity<ApiResponse<AuthResponse>> refresh(@Valid
@RequestBody RefreshRequest request) {
            return
ResponseEntity.ok(ApiResponse.ok(authService.refresh(request)));
        }

        @PostMapping("/logout")
        public ResponseEntity<ApiResponse<Void>> logout(@RequestBody
RefreshRequest request) {
            authService.logout(request.getRefreshToken());
            return ResponseEntity.ok(ApiResponse.ok(null));
        }

        @GetMapping("/verify-email")
        public ResponseEntity<Void> verifyEmail(@RequestParam String token) {
            HttpHeaders headers = new HttpHeaders();
            try {
                authService.verifyEmail(token);
                headers.add("Location", frontendUrl + "/verify-
email?success=true");
            } catch (Exception e) {
                String reason = e.getMessage() != null &&
e.getMessage().contains("expired")
                ? "expired" : "invalid";
                headers.add("Location", frontendUrl + "/verify-email?error=" +
reason);
            }
            return
ResponseEntity.status(HttpStatus.FOUND).headers(headers).build();
        }

        @PostMapping("/forgot-password")
        public ResponseEntity<ApiResponse<String>> forgotPassword(@Valid
@RequestBody ForgotPasswordRequest request) {
            authService.forgotPassword(request.getEmail());
            return ResponseEntity.ok(ApiResponse.ok("If an account with that
email exists, a reset link has been sent"));
        }

        @PostMapping("/reset-password")
        public ResponseEntity<ApiResponse<String>> resetPassword(@Valid
@RequestBody ResetPasswordRequest request) {
            authService.resetPassword(request);
            return ResponseEntity.ok(ApiResponse.ok("Password reset
successfully"));
        }
    }

```

Файл ShareController.java

```

package com.cloudstorage.controller;

import com.cloudstorage.dto.ApiResponse;
import com.cloudstorage.dto.FileOperationDto.*;
import com.cloudstorage.entity.User;
import com.cloudstorage.service.impl.ShareLinkService;
import com.cloudstorage.util.CurrentUserUtil;

```

					ИГУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		145

```

import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("/share")
@RequiredArgsConstructor
public class ShareController {

    private final ShareLinkService shareLinkService;
    private final CurrentUserUtil currentUserUtil;

    @PostMapping("/{uuid}/links")
    public ResponseEntity<ApiResponse<ShareLinkDto>> createLink(
        @PathVariable UUID uuid, @RequestBody ShareLinkRequest request)
    {
        User user = currentUserUtil.getCurrentUser();
        return
        ResponseEntity.ok(ApiResponse.ok(shareLinkService.createShareLink(user,
        uuid, request)));
    }

    @GetMapping("/{uuid}/links")
    public ResponseEntity<ApiResponse<List<ShareLinkDto>>>
    listLinks(@PathVariable UUID uuid) {
        User user = currentUserUtil.getCurrentUser();
        return
        ResponseEntity.ok(ApiResponse.ok(shareLinkService.listShareLinks(user,
        uuid)));
    }

    @DeleteMapping("/links/{linkId}")
    public ResponseEntity<ApiResponse<Void>> deactivateLink(@PathVariable
    Long linkId) {
        User user = currentUserUtil.getCurrentUser();
        shareLinkService.deactivateShareLink(user, linkId);
        return ResponseEntity.ok(ApiResponse.ok(null));
    }
}

```

Файл SecurityConfig.java

```

package com.cloudstorage.config;

import com.cloudstorage.security.JwtAuthenticationFilter;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.AuthenticationProvider;
import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.configuration

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		146

```

.AuthenticationConfiguration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import java.util.List;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthFilter;
    private final UserDetailsService userDetailsService;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception {
        http
            .cors(cors ->
cors.configurationSource(corsConfigurationSource()))
            .csrf(AbstractHttpConfigurer::disable)
            .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/auth/**").permitAll()
                .requestMatchers("/share/public/**").permitAll()
                .requestMatchers(HttpMethod.GET,
"/users/avatars/**").permitAll()
                .requestMatchers(HttpMethod.OPTIONS, "/*").permitAll()
                .anyRequest().authenticated()
            )
            .authenticationProvider(authenticationProvider())
            .addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    public CorsConfigurationSource corsConfigurationSource() {

```

					ИПУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		147

```

        CorsConfiguration config = new CorsConfiguration();
        config.setAllowedOriginPatterns(List.of("*"));
        config.setAllowedMethods(List.of("GET", "POST", "PUT", "PATCH",
"DELETE", "OPTIONS"));
        config.setAllowedHeaders(List.of("*"));
        config.setAllowCredentials(true);
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", config);
        return source;
    }

    @Bean
    public AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider provider = new
DaoAuthenticationProvider();
        provider.setUserDetailsService(userDetailsService);
        provider.setPasswordEncoder(passwordEncoder());
        return provider;
    }

    @Bean
    public AuthenticationManager
authenticationManager(AuthenticationConfiguration config) throws Exception
{
        return config.getAuthenticationManager();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}

```

Файл JwtUtil.java

```

package com.cloudstorage.security;

import io.jsonwebtoken.*;
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Component
@Slf4j
public class JwtUtil {

    @Value("${app.jwt.secret}")
    private String secret;
}

```

					ИПУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		148

```

@Value("${app.jwt.expiration}")
private Long expiration;

private SecretKey getSigningKey() {
    byte[] keyBytes = Decoders.BASE64.decode(secret);
    return Keys.hmacShaKeyFor(keyBytes);
}

public String generateToken(UserDetails userDetails) {
    return generateToken(new HashMap<>(), userDetails);
}

public String generateToken(Map<String, Object> extraClaims,
UserDetails userDetails) {
    return Jwts.builder()
        .claims(extraClaims)
        .subject(userDetails.getUsername())
        .issuedAt(new Date())
        .expiration(new Date(System.currentTimeMillis() +
expiration))
        .signWith(getSigningKey())
        .compact();
}

public boolean isTokenValid(String token, UserDetails userDetails) {
    try {
        final String username = extractUsername(token);
        return username.equals(userDetails.getUsername()) &&
!isTokenExpired(token);
    } catch (Exception e) {
        return false;
    }
}

public String extractUsername(String token) {
    return extractClaim(token, Claims::getSubject);
}

public Date extractExpiration(String token) {
    return extractClaim(token, Claims::getExpiration);
}

public <T> T extractClaim(String token, Function<Claims, T>
claimsResolver) {
    final Claims claims = extractAllClaims(token);
    return claimsResolver.apply(claims);
}

private Claims extractAllClaims(String token) {
    return Jwts.parser()
        .verifyWith(getSigningKey())
        .build()
        .parseSignedClaims(token)
        .getPayload();
}

```

```

        private boolean isTokenExpired(String token) {
            return extractExpiration(token).before(new Date());
        }
    }

```

Файл JwtAuthenticationFilter.java

```

package com.cloudstorage.security;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.lang.NonNull;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
@RequiredArgsConstructor
@Slf4j
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtUtil jwtUtil;
    private final UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(
        @NonNull HttpServletRequest request,
        @NonNull HttpServletResponse response,
        @NonNull FilterChain filterChain
    ) throws ServletException, IOException {

        final String authHeader = request.getHeader("Authorization");

        final String queryToken = request.getParameter("token");

        String jwt = null;
        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            jwt = authHeader.substring(7);
        } else if (queryToken != null && !queryToken.isBlank()) {
            jwt = queryToken;
        }

        if (jwt == null) {
            filterChain.doFilter(request, response);
        }
    }

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		150

```

        return;
    }

    try {
        final String userEmail = jwtUtil.extractUsername(jwt);

        if (userEmail != null &&
            SecurityContextHolder.getContext().getAuthentication() == null) {
            UserDetails userDetails =
                userDetailsService.loadUserByUsername(userEmail);
            if (jwtUtil.isTokenValid(jwt, userDetails)) {
                UsernamePasswordAuthenticationToken authToken =
                    new
                UsernamePasswordAuthenticationToken(userDetails, null,
                userDetails.getAuthorities());
                authToken.setDetails(new
                WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(authToken);
            }
        }
        catch (Exception e) {
            log.debug("JWT validation failed: {}", e.getMessage());
        }

        filterChain.doFilter(request, response);
    }
}

```

Файл AccessPermission.java

```

package com.cloudstorage.entity;

import jakarta.persistence.*;
import lombok.*;

import java.time.Instant;

@Entity
@Table(name = "access_permissions")
@Getter @Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class AccessPermission {

    public enum PermissionLevel {
        READ, CONTRIBUTE, WRITE, OWNER;

        public boolean includes(PermissionLevel other) {
            return this.ordinal() >= other.ordinal();
        }
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
}

```

```

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "object_id", nullable = false)
    private StorageObject object;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "grantee_id")
    private User grantee;

    @Enumerated(EnumType.STRING)
    @Column(name = "permission_level", nullable = false)
    private PermissionLevel permissionLevel;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "granted_by", nullable = false)
    private User grantedBy;

    @Column(name = "granted_at")
    private Instant grantedAt = Instant.now();
}

```

Файл ShareLink.java

```

package com.cloudstorage.entity;

import jakarta.persistence.*;
import lombok.*;

import java.time.Instant;

@Entity
@Table(name = "share_links")
@Getter @Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ShareLink {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String token;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "object_id", nullable = false)
    private StorageObject object;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "created_by", nullable = false)
    private User createdBy;

    @Enumerated(EnumType.STRING)
    @Column(name = "permission_level")
    private AccessPermission.PermissionLevel permissionLevel =
        AccessPermission.PermissionLevel.READ;
}

```

```

@Column(name = "expires_at")
private Instant expiresAt;

@Column(name = "is_active")
private Boolean isActive = true;

@Column(name = "created_at")
private Instant createdAt = Instant.now();
}

```

Файл StorageObjectRepository.java

```

package com.cloudstorage.repository;

import com.cloudstorage.entity.StorageObject;
import com.cloudstorage.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
public interface StorageObjectRepository extends
JpaRepository<StorageObject, Long> {

    Optional<StorageObject> findByUuidAndIsDeletedFalse(UUID uuid);

    Optional<StorageObject> findByUuid(UUID uuid);

    List<StorageObject> findByOwnerAndParentIsNullAndIsDeletedFalse(User
owner);

    List<StorageObject> findByOwnerAndParentAndIsDeletedFalse(User owner,
StorageObject parent);

    @Query("""
        SELECT DISTINCT so FROM StorageObject so
        JOIN FETCH so.owner
        LEFT JOIN FETCH so.parent
        WHERE so.isDeleted = false
            AND EXISTS (
                SELECT ap FROM AccessPermission ap
                WHERE ap.object = so AND ap.grantee = :user
            )
    """)
    List<StorageObject> findSharedWithUser(@Param("user") User user);

    @Query("""
        SELECT so FROM StorageObject so
        JOIN FETCH so.owner
        LEFT JOIN FETCH so.parent
        WHERE so.owner = :owner AND so.parent IS NULL AND so.isDeleted =
false
    """)

```

					ИГУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		153

```

List<StorageObject> findRootByOwnerFetched(@Param("owner") User owner);

@Query("""
    SELECT so FROM StorageObject so
    JOIN FETCH so.owner
    LEFT JOIN FETCH so.parent
    WHERE so.owner = :owner AND so.parent = :parent AND so.isDeleted =
false
    """)
List<StorageObject> findByOwnerAndParentFetched(@Param("owner") User
owner, @Param("parent") StorageObject parent);

@Query("""
    SELECT so FROM StorageObject so
    JOIN FETCH so.owner
    LEFT JOIN FETCH so.parent
    WHERE so.parent = :parent AND so.isDeleted = false
    """)
List<StorageObject> findByParentFetched(@Param("parent") StorageObject
parent);

@Query(value = """
    WITH RECURSIVE obj_with_ancestors AS (
        SELECT so.id AS obj_id, so.id AS ancestor_id, so.parent_id
        FROM storage_objects so
        WHERE so.is_deleted = false
        UNION ALL
        SELECT owa.obj_id, so.id AS ancestor_id, so.parent_id
        FROM obj_with_ancestors owa
        JOIN storage_objects so ON so.id = owa.parent_id
        WHERE so.is_deleted = false
    ),
    accessible_obj_ids AS (
        SELECT DISTINCT owa.obj_id
        FROM obj_with_ancestors owa
        JOIN access_permissions ap ON ap.object_id = owa.ancestor_id
        WHERE ap.grantee_id = :userId
    )
    SELECT so.* FROM storage_objects so
    JOIN accessible_obj_ids acc ON acc.obj_id = so.id
    WHERE so.is_deleted = false
        AND so.name ILIKE '%' || :query || '%'
        AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
        AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP))
        AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP))
    ORDER BY so.updated_at DESC
    LIMIT 100
    """, nativeQuery = true)
List<StorageObject> searchByNameShared(
    @Param("userId") Long userId,
    @Param("query") String query,
    @Param("mimeType") String mimeType,
    @Param("dateFrom") String dateFrom,
    @Param("dateTo") String dateTo

```

```

);

@Query(value = """
    WITH RECURSIVE obj_with_ancestors AS (
        SELECT so.id AS obj_id, so.id AS ancestor_id, so.parent_id
        FROM storage_objects so
        WHERE so.is_deleted = false
        UNION ALL
        SELECT owa.obj_id, so.id AS ancestor_id, so.parent_id
        FROM obj_with_ancestors owa
        JOIN storage_objects so ON so.id = owa.parent_id
        WHERE so.is_deleted = false
    ),
    accessible_obj_ids AS (
        SELECT DISTINCT owa.obj_id
        FROM obj_with_ancestors owa
        JOIN access_permissions ap ON ap.object_id = owa.ancestor_id
        WHERE ap.grantee_id = :userId
    )
    SELECT so.* FROM storage_objects so
    JOIN accessible_obj_ids acc ON acc.obj_id = so.id
    WHERE so.is_deleted = false
    AND (
        (CAST(:query AS TEXT) IS NULL OR so.name ILIKE '%' ||
CAST(:query AS TEXT) || '%')
        OR (CAST(:query AS TEXT) IS NOT NULL AND CAST(:query AS TEXT)
<> ' '
            AND so.content_tsv @@ plainto_tsquery('russian',
CAST(:query AS TEXT)))
    )
    AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
    AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP))
    AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP))
    ORDER BY so.updated_at DESC
    LIMIT 100
    """, nativeQuery = true)
List<StorageObject> advancedSearchShared(
    @Param("userId") Long userId,
    @Param("query") String query,
    @Param("mimeType") String mimeType,
    @Param("dateFrom") String dateFrom,
    @Param("dateTo") String dateTo
);

@Query(value = """
    SELECT * FROM storage_objects
    WHERE owner_id = :ownerId
    AND is_deleted = false
    AND name ILIKE '%' || :query || '%'
    AND (CAST(:mimeType AS TEXT) IS NULL OR mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
    AND (CAST(:dateFrom AS TEXT) IS NULL OR updated_at >=
CAST(:dateFrom AS TIMESTAMP))
    AND (CAST(:dateTo AS TEXT) IS NULL OR updated_at <= CAST(:dateTo

```

```

AS TIMESTAMP))
    ORDER BY updated_at DESC
    LIMIT 100
    "", nativeQuery = true)
List<StorageObject> searchByName(
    @Param("ownerId") Long ownerId,
    @Param("query") String query,
    @Param("mimeType") String mimeType,
    @Param("dateFrom") String dateFrom,
    @Param("dateTo") String dateTo
);

@Query(value = ""
    SELECT * FROM storage_objects
    WHERE owner_id = :ownerId
    AND is_deleted = false
    AND (
        (CAST(:query AS TEXT) IS NULL OR name ILIKE '%' || CAST(:query
AS TEXT) || '%')
        OR (CAST(:query AS TEXT) IS NOT NULL AND CAST(:query AS TEXT)
<> ''
            AND content_tsv @@ plainto_tsquery('russian', CAST(:query
AS TEXT)))
    )
    AND (CAST(:mimeType AS TEXT) IS NULL OR mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
    AND (CAST(:dateFrom AS TEXT) IS NULL OR updated_at >=
CAST(:dateFrom AS TIMESTAMP))
    AND (CAST(:dateTo AS TEXT) IS NULL OR updated_at <= CAST(:dateTo
AS TIMESTAMP))
    ORDER BY updated_at DESC
    LIMIT 100
    "", nativeQuery = true)
List<StorageObject> advancedSearch(
    @Param("ownerId") Long ownerId,
    @Param("query") String query,
    @Param("mimeType") String mimeType,
    @Param("dateFrom") String dateFrom,
    @Param("dateTo") String dateTo
);

@Query(value = ""
    WITH RECURSIVE children AS (
        SELECT id FROM storage_objects WHERE id = :parentId
        UNION ALL
        SELECT so.id FROM storage_objects so
        JOIN children c ON so.parent_id = c.id
    )
    SELECT * FROM storage_objects WHERE id IN (SELECT id FROM children)
AND is_deleted = false
    "", nativeQuery = true)
List<StorageObject> findAllDescendants(@Param("parentId") Long
parentId);

@Query(value = ""
    WITH RECURSIVE descendants AS (
        -- Base: objects directly shared with the user

```

```

        SELECT so.id, so.owner_id
        FROM storage_objects so
        JOIN access_permissions ap ON ap.object_id = so.id
        WHERE ap.grantee_id = :userId
            AND so.is_deleted = false
        UNION ALL
        -- Recursive: children of already-reachable objects
        SELECT so.id, so.owner_id
        FROM storage_objects so
        JOIN descendants d ON so.parent_id = d.id
        WHERE so.is_deleted = false
    )
    SELECT so.* FROM storage_objects so
    JOIN descendants d ON d.id = so.id
    WHERE so.is_deleted = false
    """ , nativeQuery = true)
    List<StorageObject> findAllAccessibleForUser(@Param("userId") Long
userId);

    @Query(value = """
        WITH RECURSIVE subtree AS (
            SELECT id FROM storage_objects WHERE id = :folderId AND
is_deleted = false
            UNION ALL
            SELECT so.id FROM storage_objects so
            JOIN subtree s ON so.parent_id = s.id
            WHERE so.is_deleted = false
        )
        SELECT so.* FROM storage_objects so
        JOIN subtree st ON st.id = so.id
        WHERE so.owner_id = :ownerId
            AND so.is_deleted = false
            AND so.name ILIKE '%' || :query || '%'
            AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
            AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP WITH TIME ZONE))
            AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP WITH TIME ZONE))
            ORDER BY so.updated_at DESC
            LIMIT 100
    """ , nativeQuery = true)
    List<StorageObject> searchByNameInFolder(
        @Param("ownerId") Long ownerId,
        @Param("folderId") Long folderId,
        @Param("query") String query,
        @Param("mimeType") String mimeType,
        @Param("dateFrom") String dateFrom,
        @Param("dateTo") String dateTo
    );

    @Query(value = """
        WITH RECURSIVE subtree AS (
            SELECT id FROM storage_objects WHERE id = :folderId AND
is_deleted = false
            UNION ALL
            SELECT so.id FROM storage_objects so

```

```

        JOIN subtree s ON so.parent_id = s.id
        WHERE so.is_deleted = false
    )
    SELECT so.* FROM storage_objects so
    JOIN subtree st ON st.id = so.id
    WHERE so.owner_id = :ownerId
        AND so.is_deleted = false
        AND (
            (CAST(:query AS TEXT) IS NULL OR so.name ILIKE '%' ||
CAST(:query AS TEXT) || '%')
            OR (CAST(:query AS TEXT) IS NOT NULL AND CAST(:query AS TEXT)
<> ''
                AND so.content_tsv @@ plainto_tsquery('russian',
CAST(:query AS TEXT)))
        )
        AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
        AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP WITH TIME ZONE))
        AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP WITH TIME ZONE))
        ORDER BY so.updated_at DESC
        LIMIT 100
    """, nativeQuery = true)
List<StorageObject> advancedSearchInFolder(
    @Param("ownerId") Long ownerId,
    @Param("folderId") Long folderId,
    @Param("query") String query,
    @Param("mimeType") String mimeType,
    @Param("dateFrom") String dateFrom,
    @Param("dateTo") String dateTo
);

@Query(value = ""
    WITH RECURSIVE subtree AS (
        SELECT id FROM storage_objects WHERE id = :folderId AND
is_deleted = false
        UNION ALL
        SELECT so.id FROM storage_objects so
        JOIN subtree s ON so.parent_id = s.id
        WHERE so.is_deleted = false
    )
    SELECT so.* FROM storage_objects so
    JOIN subtree st ON st.id = so.id
    WHERE so.is_deleted = false
        AND so.name ILIKE '%' || :query || '%'
        AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
        AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP WITH TIME ZONE))
        AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP WITH TIME ZONE))
        ORDER BY so.updated_at DESC
        LIMIT 100
    """, nativeQuery = true)
List<StorageObject> searchByNameSharedInFolder(
    @Param("folderId") Long folderId,

```

```

        @Param("query") String query,
        @Param("mimeType") String mimeType,
        @Param("dateFrom") String dateFrom,
        @Param("dateTo") String dateTo
    );

    @Query(value = """
        WITH RECURSIVE subtree AS (
            SELECT id FROM storage_objects WHERE id = :folderId AND
is_deleted = false
            UNION ALL
            SELECT so.id FROM storage_objects so
            JOIN subtree s ON so.parent_id = s.id
            WHERE so.is_deleted = false
        )
        SELECT so.* FROM storage_objects so
        JOIN subtree st ON st.id = so.id
        WHERE so.is_deleted = false
            AND (
                (CAST(:query AS TEXT) IS NULL OR so.name ILIKE '%' ||
CAST(:query AS TEXT) || '%')
                OR (CAST(:query AS TEXT) IS NOT NULL AND CAST(:query AS TEXT)
<> ''
                    AND so.content_tsv @@ plainto_tsquery('russian',
CAST(:query AS TEXT)))
            )
            AND (CAST(:mimeType AS TEXT) IS NULL OR so.mime_type ILIKE
CAST(:mimeType AS TEXT) || '%')
            AND (CAST(:dateFrom AS TEXT) IS NULL OR so.updated_at >=
CAST(:dateFrom AS TIMESTAMP WITH TIME ZONE))
            AND (CAST(:dateTo AS TEXT) IS NULL OR so.updated_at <=
CAST(:dateTo AS TIMESTAMP WITH TIME ZONE))
            ORDER BY so.updated_at DESC
            LIMIT 100
        """, nativeQuery = true)
    List<StorageObject> advancedSearchSharedInFolder(
        @Param("folderId") Long folderId,
        @Param("query") String query,
        @Param("mimeType") String mimeType,
        @Param("dateFrom") String dateFrom,
        @Param("dateTo") String dateTo
    );

    boolean existsByNameAndParentAndOwnerAndIsDeletedFalse(String name,
StorageObject parent, User owner);

    List<StorageObject> findByOwnerAndIsDeletedTrue(User owner);
}

```

Файл SearchServiceClient.java

```
package com.cloudstorage.service;
```

```

import com.cloudstorage.dto.SemanticSearchResult;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpMethod;

```

					ИГУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		159

```

import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;

import java.net.URI;
import java.util.List;

@Service
@Slf4j
public class SearchServiceClient {

    private final RestTemplate restTemplate;
    private final String searchServiceUrl;

    public SearchServiceClient(
        @Value("${app.search-service.url:http://search-service:8081}")
        String searchServiceUrl) {
        this.restTemplate = new RestTemplate();
        this.searchServiceUrl = searchServiceUrl;
    }

    public List<SemanticSearchResult> semanticSearch(Long ownerId, String
query) {
        return call("/internal/search/semantic", ownerId, query);
    }

    public List<SemanticSearchResult> hybridSearch(Long ownerId, String
query) {
        return call("/internal/search/hybrid", ownerId, query);
    }

    public List<String> getChatContext(Long objectId, Long ownerId, String
query, int topK) {
        try {
            URI uri = UriComponentsBuilder
                .fromHttpUrl(searchServiceUrl +
"/internal/search/chat/context")
                .queryParams("objectId", objectId)
                .queryParams("ownerId", ownerId)
                .queryParams("query", query)
                .queryParams("topK", topK)
                .build()
                .encode()
                .toUri();

            ResponseEntity<List<String>> response = restTemplate.exchange(
                uri,
                HttpMethod.GET,
                null,
                new ParameterizedTypeReference<>() {}
            );
            List<String> result = response.getBody();
            return result != null ? result : List.of();
        } catch (Exception e) {
            log.warn("getChatContext call unavailable: {}",
e.getMessage());
        }
    }
}

```

					ИПУ 1.090301.24.001 ИЗ	Лист
						160
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return List.of();
    }
}

private List<SemanticSearchResult> call(String path, Long ownerId,
String query) {
    try {

        URI uri = UriComponentsBuilder
            .fromHttpUrl(searchServiceUrl + path)
            .queryParams("ownerId", ownerId)
            .queryParams("query", query)
            .build()
            .encode()
            .toUri();

        ResponseEntity<List<SemanticSearchResult>> response =
restTemplate.exchange(
            uri,
            HttpMethod.GET,
            null,
            new ParameterizedTypeReference<>() {}
        );

        List<SemanticSearchResult> results = response.getBody();
        return results != null ? results : List.of();

    } catch (Exception e) {
        log.warn("Search call to {} unavailable: {}", path,
e.getMessage());
        return List.of();
    }
}
}

```

					<p style="text-align: center;"><i>ПГУ 1.090301.24.001 ПЗ</i></p>	Лист
						161
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг микросервиса search-service

Файл HybridSearchService.java

```
package com.cloudstorage.search.service;

import com.cloudstorage.search.dto.SemanticSearchResult;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Service;

import java.util.*;

@Service
@RequiredArgsConstructor
@Slf4j
public class HybridSearchService {

    private static final int MAX_BM25_RESULTS = 50;
    private static final int MAX_SEMANTIC_RESULTS = 50;
    private static final int MAX_FINAL_RESULTS = 20;

    private static final int RRF_K = 60;

    private final JdbcTemplate jdbc;
    private final SemanticSearchService semanticSearchService;
    private final RerankService rerankService;

    public List<SemanticSearchResult> search(Long ownerId, String query) {
        if (query == null || query.isBlank()) return List.of();

        List<Long> bm25Ids = bm25Search(ownerId, query);
        log.debug("BM25 returned {} results", bm25Ids.size());

        List<SemanticSearchResult> semanticResults =
            semanticSearchService.searchWithoutRerank(ownerId, query);
        log.debug("Semantic returned {} results (no rerank)",
            semanticResults.size());

        Map<Long, Double> rrfScores = new HashMap<>();

        for (int i = 0; i < bm25Ids.size(); i++) {
            long id = bm25Ids.get(i);
            rrfScores.merge(id, 1.0 / (RRF_K + i + 1), Double::sum);
        }

        for (int i = 0; i < semanticResults.size(); i++) {
            long id = semanticResults.get(i).getId();
            rrfScores.merge(id, 1.0 / (RRF_K + i + 1), Double::sum);
        }

        if (rrfScores.isEmpty()) return List.of();

        Map<Long, SemanticSearchResult> metaMap = new HashMap<>();
        for (SemanticSearchResult r : semanticResults)
            metaMap.put(r.getId(), r);
    }
}
```

					ИГУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		162

```

        Set<Long> missingIds = new HashSet<>(rrfScores.keySet());
        missingIds.removeAll(metaMap.keySet());
        if (!missingIds.isEmpty()) {
            loadMetadata(missingIds).forEach(r -> metaMap.put(r.getId(),
r));
        }

        List<SemanticSearchResult> merged = rrfScores.entrySet().stream()
            .sorted(Map.Entry.<Long,
Double>comparingByValue().reversed())
            .limit(MAX_FINAL_RESULTS)
            .map(e -> {
                SemanticSearchResult meta = metaMap.get(e.getKey());
                if (meta == null) return null;
                return new SemanticSearchResult(
                    meta.getId(), meta.getUuid(),
                    meta.getName(), meta.getMimeType(),
                    e.getValue(),
                    meta.getChunkText());
            })
            .filter(Objects::nonNull)
            .toList();

        log.debug("Hybrid RRF returned {} merged results for query '{}'",
merged.size(), query);

        List<SemanticSearchResult> reranked = rerankService.rerank(query,
merged);
        log.debug("After rerank: {} results for query '{}'",
reranked.size(), query);
        return reranked;
    }

    private List<Long> bm25Search(Long ownerId, String query) {
        try {
            return jdbc.query("""
                SELECT id
                FROM storage_objects
                WHERE owner_id = ?
                AND is_deleted = false
                AND content_tsv IS NOT NULL
                AND content_tsv @@ (
                    plainto_tsquery('russian', ?)
                    || plainto_tsquery('simple', ?)
                )
                ORDER BY ts_rank(content_tsv,
                    plainto_tsquery('russian', ?)
                    || plainto_tsquery('simple', ?)
                ) DESC
                LIMIT ?
            """,
                (rs, rowNum) -> rs.getLong("id"),
                ownerId, query, query, query, query, MAX_BM25_RESULTS);
        } catch (Exception e) {
            log.warn("BM25 search failed: {}", e.getMessage());
        }
    }

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		163

```

        return List.of();
    }
}

private List<SemanticSearchResult> loadMetadata(Set<Long> ids) {
    if (ids.isEmpty()) return List.of();
    String placeholders = String.join(",",
Collections.nCopies(ids.size(), "?"));
    Object[] params = ids.toArray();
    try {
        return jdbc.query(
            "SELECT id, uuid, name, mime_type FROM storage_objects
WHERE id IN (" + placeholders + ")",
            (rs, rowNum) -> new SemanticSearchResult(
                rs.getLong("id"),
                UUID.fromString(rs.getString("uuid")),
                rs.getString("name"),
                rs.getString("mime_type"),
                0.0
            ),
            params);
    } catch (Exception e) {
        log.warn("Failed to load metadata for BM25-only results: {}",
e.getMessage());
        return List.of();
    }
}
}

```

Файл EmbeddingService.java

```

package com.cloudstorage.search.service;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.ai.embedding.EmbeddingModel;
import org.springframework.ai.embedding.EmbeddingRequest;
import org.springframework.ai.embedding.EmbeddingResponse;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@RequiredArgsConstructor
@Slf4j
public class EmbeddingService {

    private static final int MAX_EMBED_CHARS = 20000;

    private final EmbeddingModel embeddingModel;

    public float[] embed(String text) {
        if (text == null || text.isBlank()) return new float[0];
        try {

            String truncated = text.length() > MAX_EMBED_CHARS ?
text.substring(0, MAX_EMBED_CHARS) : text;

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		164

```

        EmbeddingResponse response = embeddingModel.call(
            new EmbeddingRequest(List.of(truncated), null));

        if (response == null || response.getResults() == null ||
response.getResults().isEmpty()) {
            log.warn("Empty embedding response for text (len={})",
truncated.length());
            return new float[0];
        }

        List<Double> outputList =
response.getResults().get(0).getOutput();
        if (outputList == null || outputList.isEmpty()) {
            log.warn("Null/empty output in embedding response");
            return new float[0];
        }
        float[] result = new float[outputList.size()];
        for (int i = 0; i < outputList.size(); i++) {
            result[i] = outputList.get(i).floatValue();
        }
        return result;

    } catch (Exception e) {
        log.warn("Failed to generate embedding: {}", e.getMessage());
        return new float[0];
    }
}

public double cosineSimilarity(float[] a, float[] b) {
    if (a.length == 0 || b.length != a.length) return 0.0;
    double dot = 0, normA = 0, normB = 0;
    for (int i = 0; i < a.length; i++) {
        dot += a[i] * b[i];
        normA += a[i] * a[i];
        normB += b[i] * b[i];
    }
    double denom = Math.sqrt(normA) * Math.sqrt(normB);
    return denom == 0 ? 0.0 : dot / denom;
}
}

```

Файл RerankService.java

```

package com.cloudstorage.search.service;

import com.cloudstorage.search.dto.SemanticSearchResult;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import lombok.Data;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import java.util.*;
import java.util.stream.Collectors;

```

					ИПУ 1.090301.24.001 ИЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		165

```

@Service
@Slf4j
@RequiredArgsConstructor
public class RerankService {

    private final RestTemplate restTemplate;

    @Value("${app.reranker.enabled:true}")
    private boolean enabled;

    @Value("${app.reranker.url:http://localhost:8082}")
    private String rerankerUrl;

    @Value("${app.reranker.top-n:10}")
    private int topN;

    public List<SemanticSearchResult> rerank(String query,
List<SemanticSearchResult> candidates) {
        if (!enabled || candidates.isEmpty() || query == null ||
query.isBlank()) {
            return
candidates.stream().limit(topN).collect(Collectors.toList());
        }

        try {

            List<String> documents = candidates.stream()
                .map(this::resolveDocText)
                .collect(Collectors.toList());

            RerankRequest request = new RerankRequest(query, documents,
topN);
            RerankResponse response = restTemplate.postForObject(
                rerankerUrl + "/rerank", request,
RerankResponse.class);

            if (response == null || response.getResults() == null) {
                log.warn("Empty response from reranker – falling back to
original order");
                return
candidates.stream().limit(topN).collect(Collectors.toList());
            }

            List<SemanticSearchResult> reranked =
response.getResults().stream()
                .map(r -> {
                    SemanticSearchResult orig =
candidates.get(r.getIndex());
                    return new SemanticSearchResult(
                        orig.getId(),
                        orig.getUuid(),
                        orig.getName(),
                        orig.getMimeType(),
                        r.getScore(),
                        orig.getChunkText()
                    );
                });

```

```

        })
        .collect(Collectors.toList());

        log.debug("Reranker: {} → {} results for query '{}' [{}ms]",
            candidates.size(), reranked.size(), query,
response.getMs());
        return reranked;

        } catch (Exception e) {
            log.warn("Reranker unavailable ({}), falling back to original
order", e.getMessage());
            return
candidates.stream().limit(topN).collect(Collectors.toList());
        }
    }

    private String resolveDocText(SemanticSearchResult candidate) {
        if (candidate.getChunkText() != null &&
!candidate.getChunkText().isBlank()) {
            return candidate.getChunkText();
        }
        log.warn("chunkText is null for '{}' (id={}) – using file name as
fallback",
            candidate.getName(), candidate.getId());
        return candidate.getName() != null ? candidate.getName() : "";
    }

    @Data
    static class RerankRequest {
        private final String query;
        private final List<String> documents;
        private final Integer top_n;
    }

    @Data
    @JsonIgnoreProperties(ignoreUnknown = true)
    static class RerankResponse {
        private List<RerankResult> results;
        private String model;
        private int ms;
    }

    @Data
    @JsonIgnoreProperties(ignoreUnknown = true)
    static class RerankResult {
        private int index;
        private double score;
    }
}

```

Файл IndexingConsumer.java

```
package com.cloudstorage.search.consumer;
```

```

import com.cloudstorage.search.config.RabbitConfig;
import com.cloudstorage.search.dto.SearchIndexMessage;
import com.cloudstorage.search.entity.StorageObjectIndex;
import com.cloudstorage.search.repository.ChunkEmbeddingJdbcRepository;

```

					ПГУ 1.090301.24.001 ПЗ	Лист
						167
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.cloudstorage.search.repository.EmbeddingJdbcRepository;
import com.cloudstorage.search.repository.SearchIndexRepository;
import com.cloudstorage.search.service.ChunkingService;
import com.cloudstorage.search.service.EmbeddingService;
import com.cloudstorage.search.service.TextExtractorService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

@Component
@RequiredArgsConstructor
@Slf4j
public class IndexingConsumer {

    private final SearchIndexRepository indexRepository;
    private final EmbeddingJdbcRepository embeddingJdbcRepository;
    private final ChunkEmbeddingJdbcRepository
chunkEmbeddingJdbcRepository;
    private final TextExtractorService textExtractor;
    private final EmbeddingService embeddingService;
    private final ChunkingService chunkingService;

    @Value("${app.storage.path}")
    private String storagePath;

    private static final int MAX_EMBED_CHARS = 20000;

    @RabbitListener(queues = RabbitConfig.SEARCH_INDEX_QUEUE)
    public void handleIndexMessage(SearchIndexMessage message) {
        log.debug("Received index message: action={}, objectId={}",
message.getAction(), message.getObjectId());

        if ("DELETE".equals(message.getAction())) {
            indexRepository.findById(message.getObjectId()).ifPresent(obj -
> {
                indexRepository.updateFullTextIndex(obj.getId(), "");
                chunkEmbeddingJdbcRepository.deleteByObjectId(obj.getId());
            });
            return;
        }

        if ("INDEX".equals(message.getAction())) {
            StorageObjectIndex obj =
indexRepository.findById(message.getObjectId()).orElse(null);
            if (obj == null || Boolean.TRUE.equals(obj.getIsDeleted()))
return;

            String extractedText = "";

            if (message.getStoragePath() != null &&
textExtractor.isIndexable(message.getMimeType())) {

```

					ИПУ 1.090301.24.001 ИЗ	Лист
						168
Изм.	Лист	№ докум.	Подпись	Дата		

```

        Path filePath = Paths.get(storagePath,
message.getStoragePath());
        extractedText = textExtractor.extractText(filePath);
        log.debug("Extracted {} chars from {}",
extractedText.length(), message.getObjectName());
    }

    String fullText = message.getObjectName() + " " +
extractedText;
    indexRepository.updateFullTextIndex(obj.getId(), fullText);

    String embeddingInput =
buildEmbeddingInput(message.getObjectName(), extractedText);
    if (!embeddingInput.isBlank()) {
        float[] embedding = embeddingService.embed(embeddingInput);
        if (embedding.length > 0) {
            embeddingJdbcRepository.updateEmbedding(obj.getId(),
embedding);
            log.debug("Updated whole-doc embedding ({} dims) for
object {}", embedding.length, obj.getId());
        }
    }

    indexChunks(obj, message.getObjectName(), extractedText);

    log.info("Indexed object {} ({{}}) with chunking", obj.getId(),
message.getObjectName());
}

private void indexChunks(StorageObjectIndex obj, String name, String
content) {

    chunkEmbeddingJdbcRepository.deleteByObjectId(obj.getId());

    if (content == null || content.isBlank()) {

        float[] emb = embeddingService.embed(name != null ? name : "");
        if (emb.length > 0) {

            chunkEmbeddingJdbcRepository.upsertChunkEmbedding(obj.getId(),
obj.getOwnerId(), 0, name, emb);
            log.debug("Stored 1 chunk (filename-only) for object {}",
obj.getId());
        }
        return;
    }

    List<String> chunks = chunkingService.chunk(content);
    log.debug("Split object {} into {} chunks", obj.getId(),
chunks.size());

    int stored = 0;
    for (int i = 0; i < chunks.size(); i++) {

        String chunkText = (i == 0 && name != null)
            ? name + "\n\n" + chunks.get(i)

```

```

        : chunks.get(i);

        if (chunkText.length() > MAX_EMBED_CHARS) {
            chunkText = chunkText.substring(0, MAX_EMBED_CHARS);
        }

        float[] emb = embeddingService.embed(chunkText);
        if (emb.length > 0) {

chunkEmbeddingJdbcRepository.upsertChunkEmbedding(obj.getId(),
obj.getOwnerId(), i, chunkText, emb);
            stored++;
        } else {
            log.warn("Empty embedding for chunk {} of object {} -
Ollama unavailable?", i, obj.getId());
        }
    }
    log.debug("Stored {}/{ } chunk embeddings for object {}", stored,
chunks.size(), obj.getId());
}

private String buildEmbeddingInput(String name, String content) {
    if (name == null) name = "";
    if (content == null) content = "";
    String combined = name + "\n" + content;
    return combined.length() > MAX_EMBED_CHARS ? combined.substring(0,
MAX_EMBED_CHARS) : combined;
}
}

```

Листинг микросервиса chat-service

Файл FileChatService.java

```
package com.cloudstorage.chat.service;

import com.cloudstorage.chat.dto.ChatDto.FileInfo;
import com.cloudstorage.chat.dto.ChatDto.MessageResponse;
import com.cloudstorage.chat.repository.ChatMessageRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.servlet.mvc.method.annotation.SseEmitter;

import org.springframework.stereotype.Service;

import java.io.IOException;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

@Service
@RequiredArgsConstructor
@Slf4j
public class FileChatService {

    private static final int RAG_TOP_K = 6;

    public static final long SSE_TIMEOUT_MS = 180_000L;

    private final FileAccessChecker accessChecker;
    private final SearchServiceClient searchClient;
    private final OllamaClient ollamaClient;
    private final ChatMessageRepository messageRepo;

    public List<MessageResponse> getHistory(UUID fileUuid, Long userId) {
        FileInfo file = accessChecker.getFileOrThrow(fileUuid, userId);
        return messageRepo.findByObjectAndUser(file.getId(), userId);
    }

    public void clearHistory(UUID fileUuid, Long userId) {
        FileInfo file = accessChecker.getFileOrThrow(fileUuid, userId);
        int deleted = messageRepo.deleteByObjectAndUser(file.getId(),
userId);
        log.debug("Cleared {} chat messages for object={} user={}",
deleted, file.getId(), userId);
    }

    public SseEmitter streamChat(UUID fileUuid, Long userId, String
question) {
        SseEmitter emitter = new SseEmitter(SSE_TIMEOUT_MS);
        FileInfo file = accessChecker.getFileOrThrow(fileUuid, userId);

        CompletableFuture.runAsync(() -> {
            try {

                messageRepo.save(file.getId(), userId, "user", question);
```

```

        List<String> chunks = searchClient.getChatContext(
            file.getId(), file.getOwnerId(), question,
RAG_TOP_K);

        String context = chunks.isEmpty()
            ? "(контекст не найден - файл ещё не
проиндексирован или не содержит текста)"
            : String.join("\n\n---\n\n", chunks);

        String systemPrompt = """
            Ты - умный ассистент для работы с документами. \
            Тебе предоставлены фрагменты из файла «%s».
            Правила:
            1. Отвечай только на основе предоставленного
контекста.
            2. Если ответа нет в контексте - честно скажи об
этом, \
            не придумывай информацию.
            3. Отвечай на том же языке, что и вопрос
пользователя.
            4. Цитируй конкретные места из текста, если это
уместно.

            Контекст из файла:
            %s
            """.formatted(file.getName(), context);

        StringBuilder fullAnswer = new StringBuilder();

        ollamaClient.streamChat(systemPrompt, question, token -> {
            fullAnswer.append(token);
            try {

                String sseToken = token.replace("\n",
"\n\n").replace(" ", "\\s");

                emitter.send(SseEmitter.event().name("token").data(sseToken));
            } catch (IOException e) {
                throw new RuntimeException("SSE send failed", e);
            }
        });

        messageRepo.save(file.getId(), userId, "assistant",
fullAnswer.toString());

        emitter.send(SseEmitter.event().name("done").data("[DONE]"));
        emitter.complete();

    } catch (Exception e) {
        log.error("Chat stream error for file={} user={}: {}",
fileUuid, userId, e.getMessage(), e);
        try {
            emitter.send(SseEmitter.event()
                .name("error")
                .data("Ошибка генерации ответа: " +
e.getMessage()));

```

```

        } catch (IOException ignored) {}
        emitter.completeWithError(e);
    }
});

return emitter;
}
}

```

Файл SearchServiceClient.java

```
package com.cloudstorage.chat.service;
```

```

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;

```

```

import java.net.URI;
import java.util.List;

```

```
@Service
```

```
@Slf4j
```

```

public class SearchServiceClient {

    private final RestTemplate restTemplate;
    private final String searchServiceUrl;

    public SearchServiceClient(
        RestTemplate restTemplate,
        @Value("${app.search-service.url:http://search-service:8081}")
String searchServiceUrl) {
        this.restTemplate = restTemplate;
        this.searchServiceUrl = searchServiceUrl;
    }

    public List<String> getChatContext(Long objectId, Long ownerId, String
query, int topK) {
        try {

            URI uri = UriComponentsBuilder
                .fromHttpUrl(searchServiceUrl +
"/internal/search/chat/context")
                .queryParams("objectId", objectId)
                .queryParams("ownerId", ownerId)
                .queryParams("query", query)
                .queryParams("topK", topK)
                .build()
                .encode()
                .toUri();

            ResponseEntity<List<String>> response = restTemplate.exchange(
                uri,

```

					ИГУ 1.090301.24.001 ПЗ	Лист
						173
Изм.	Лист	№ докум.	Подпись	Дата		

```

        HttpMethod.GET,
        null,
        new ParameterizedTypeReference<>() {}
    );
    List<String> result = response.getBody();
    return result != null ? result : List.of();

    } catch (Exception e) {
        log.warn("getChatContext call unavailable (objectId={}): {}",
objectId, e.getMessage());
        return List.of();
    }
}
}

```

					<p><i>ПГУ 1.090301.24.001 ПЗ</i></p>	Лист
						174
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг микросервиса reranker-service

Файл main.py

```
import logging
import math
import os
import time
from contextlib import asynccontextmanager
from typing import Optional

import torch
from fastapi import FastAPI, HTTPException
from fastapi.responses import JSONResponse
from pydantic import BaseModel, Field
from transformers import AutoModelForSequenceClassification, AutoTokenizer

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(name)s - %(message)s",
)
log = logging.getLogger("reranker")

MODEL_NAME = os.getenv("RERANKER_MODEL", "BAAI/bge-reranker-v2-m3")
MAX_LENGTH = int(os.getenv("RERANKER_MAX_LENGTH", "512"))
BATCH_SIZE = int(os.getenv("RERANKER_BATCH_SIZE", "32"))
HF_CACHE = os.getenv("HF_HOME", "/cache")

tokenizer: Optional[AutoTokenizer] = None
model: Optional[AutoModelForSequenceClassification] = None
device: str = "cpu"

@asynccontextmanager
async def lifespan(app: FastAPI):
    """Загружаем модель один раз при старте сервиса."""
    global tokenizer, model, device

    device = "cuda" if torch.cuda.is_available() else "cpu"
    log.info("Loading %s on %s (cache: %s)", MODEL_NAME, device, HF_CACHE)
    t0 = time.time()

    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME,
cache_dir=HF_CACHE)
    model = AutoModelForSequenceClassification.from_pretrained(
        MODEL_NAME, cache_dir=HF_CACHE
    )
    model.to(device)
    model.eval()

    log.info("Model ready in %.1fs", time.time() - t0)
    yield

    log.info("Shutting down - releasing model")
    del model
    del tokenizer

app = FastAPI()
```

```

        title="BGE Reranker Service",
        description="Cross-encoder перанкинг для семантического поиска",
        version="1.0.0",
        lifespan=lifespan,
    )

class RerankRequest(BaseModel):
    query: str = Field(..., description="Поисковый запрос")
    documents: list[str] = Field(..., description="Список текстов документов/чанков")
    top_n: Optional[int] = Field(None, description="Вернуть только top-N результатов")

class RerankResult(BaseModel):
    index: int = Field(..., description="Исходный индекс документа в запросе")
    score: float = Field(..., description="Скор релевантности [0, 1]")

class RerankResponse(BaseModel):
    results: list[RerankResult]
    model: str
    ms: int = Field(..., description="Время инференса в миллисекундах")

@app.get("/health")
def health():
    return {"status": "ok", "model": MODEL_NAME, "device": device}

@app.post("/rerank", response_model=RerankResponse)
def rerank(req: RerankRequest):
    if model is None or tokenizer is None:
        raise HTTPException(status_code=503, detail="Model not loaded yet")

    if not req.documents:
        return RerankResponse(results=[], model=MODEL_NAME, ms=0)

    t0 = time.time()

    pairs = [[req.query, doc] for doc in req.documents]

    scores = _score_pairs(pairs)

    def sigmoid(x: float) -> float:
        return 1.0 / (1.0 + math.exp(-x))

    results = [
        RerankResult(index=i, score=sigmoid(s))
        for i, s in enumerate(scores)
    ]
    results.sort(key=lambda r: r.score, reverse=True)

    if req.top_n is not None:
        results = results[: req.top_n]

    elapsed_ms = int((time.time() - t0) * 1000)
    log.info(
        "Reranked %d docs for query %r → top score=%.4f [%dms]",

```

```

        len(req.documents), req.query[:60], results[0].score if results
else 0, elapsed_ms,
    )

    return RerankResponse(results=results, model=MODEL_NAME, ms=elapsed_ms)

def _score_pairs(pairs: list[list[str]]) -> list[float]:
    all_scores: list[float] = []

    for i in range(0, len(pairs), BATCH_SIZE):
        batch = pairs[i : i + BATCH_SIZE]

        inputs = tokenizer(
            batch,
            padding=True,
            truncation=True,
            max_length=MAX_LENGTH,
            return_tensors="pt",
        ).to(device)

        with torch.no_grad():
            logits = model(**inputs).logits

        if logits.dim() == 2:
            logits = logits.squeeze(-1)

        all_scores.extend(logits.cpu().tolist())

    return all_scores

@app.exception_handler(Exception)
async def global_exception_handler(request, exc):
    log.exception("Unhandled error: %s", exc)
    return JSONResponse(status_code=500, content={"detail": str(exc)})

```

Листинг frontend

Файл api.js

```
import axios from 'axios';

const api = axios.create({
  baseURL: '/api',
  timeout: 60000,
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem('accessToken');
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});

api.interceptors.response.use(
  (res) => res,
  async (error) => {
    const original = error.config;
    if (error.response?.status === 401 && !original._retry) {
      original._retry = true;
      const refreshToken = localStorage.getItem('refreshToken');
      if (refreshToken) {
        try {
          const res = await axios.post('/api/auth/refresh', { refreshToken });

          const { accessToken } = res.data.data;
          localStorage.setItem('accessToken', accessToken);
          original.headers.Authorization = `Bearer ${accessToken}`;
          return api(original);
        } catch {
          localStorage.clear();
          window.location.href = '/login';
        }
      } else {
        window.location.href = '/login';
      }
    }
    return Promise.reject(error);
  }
);

export const authApi = {
  register: (data) => api.post('/auth/register', data),
  login: (data) => api.post('/auth/login', data),
  logout: (rt) => api.post('/auth/logout', { refreshToken: rt }),
  refresh: (rt) => api.post('/auth/refresh', { refreshToken: rt }),
  forgotPassword: (email) => api.post('/auth/forgot-password', { email }),
  resetPassword: (data) => api.post('/auth/reset-password', data),
  verifyEmail: (token) => api.get(`/auth/verify-email?token=${token}`),
};

export const userApi = {
  getProfile: () => api.get('/users/me'),
  updateProfile: (data) => api.patch('/users/me', data),
};
```

					ПГУ 1.090301.24.001 ПЗ	Лист
						178
Изм.	Лист	№ докум.	Подпись	Дата		

```

changePassword: (data) => api.post('/users/me/change-password', data),
deleteAccount:  ()      => api.delete('/users/me'),
uploadAvatar:   (file) => {
  const form = new FormData();
  form.append('avatar', file);
  return api.post('/users/me/avatar', form, {
    headers: { 'Content-Type': 'multipart/form-data' },
  });
},
deleteAvatar:  () => api.delete('/users/me/avatar'),
};

export const fileApi = {
  listRoot:      ()          => api.get('/files'),
  listFolder:    (uuid)      => api.get(`/files/folder/${uuid}`),
  listShared:    ()          => api.get('/files/shared'),
  getObject:     (uuid)      => api.get(`/files/${uuid}`),
  createFolder:  (data)      => api.post('/files/folders', data),
  rename:        (uuid, n)   => api.patch(`/files/${uuid}/rename`, { newName:
n  }),
  move:          (uuid, d)   => api.post(`/files/${uuid}/move`, d),
  copy:          (uuid, d)   => api.post(`/files/${uuid}/copy`, d),
  delete:        (uuid)      => api.delete(`/files/${uuid}`),
  search:        (data)      => api.post('/files/search', data),
  downloadUrl:   (uuid)      => `/api/files/${uuid}/download`,

  upload: (file, parentUuid, onProgress) => {
    const form = new FormData();
    form.append('file', file);
    if (parentUuid) form.append('parentUuid', parentUuid);
    return api.post('/files/upload', form, {
      headers: { 'Content-Type': 'multipart/form-data' },
      onUploadProgress: (e) => onProgress &&
onProgress(Math.round((e.loaded * 100) / e.total)),
    });
  },

  listTrash:     ()          => api.get('/files/trash'),
  restore:       (uuid)      => api.post(`/files/trash/${uuid}/restore`),
  deletePerm:    (uuid)      => api.delete(`/files/trash/${uuid}`),
  emptyTrash:    ()          => api.delete('/files/trash'),

  listPerms:     (uuid)      => api.get(`/files/${uuid}/permissions`),
  grantPerm:     (uuid, data) => api.post(`/files/${uuid}/permissions`,
data),
  revokePerm:    (uuid, granteeId) =>
api.delete(`/files/${uuid}/permissions/${granteeId}`),

  createLink:    (uuid, data) => api.post(`/share/${uuid}/links`, data),
  listLinks:     (uuid)      => api.get(`/share/${uuid}/links`),
  deactivateLink: (id)       => api.delete(`/share/links/${id}`),
};

export const chatApi = {
  getHistory:    (uuid)      => api.get(`/files/${uuid}/chat/history`),
  clearHistory:  (uuid)      => api.delete(`/files/${uuid}/chat/history`),
};

```

```

    openStream: (uuid, question) => {
      const token = localStorage.getItem('accessToken');
      const params = new URLSearchParams({ q: question, token });
      return new EventSource(`/api/files/${uuid}/chat/stream?${params}`);
    },
  };

export default api;

```

Файл AuthContext.js

```

import React, { createContext, useContext, useState, useEffect, useCallback } from 'react';
import { authApi, userApi } from '../services/api';

const AuthContext = createContext(null);

export function AuthProvider({ children }) {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  const loadUser = useCallback(async () => {
    try {
      const res = await userApi.getProfile();
      setUser(res.data.data);
    } catch {
      setUser(null);
    } finally {
      setLoading(false);
    }
  }, []);

  useEffect(() => {
    if (localStorage.getItem('accessToken')) loadUser();
    else setLoading(false);
  }, [loadUser]);

  const login = async (email, password) => {
    const res = await authApi.login({ email, password });
    const { accessToken, refreshToken, user: u } = res.data.data;
    localStorage.setItem('accessToken', accessToken);
    localStorage.setItem('refreshToken', refreshToken);
    setUser(u);
    return u;
  };

  const logout = async () => {
    try {
      const rt = localStorage.getItem('refreshToken');
      if (rt) await authApi.logout(rt);
    } finally {
      localStorage.clear();
      setUser(null);
    }
  };

  const refreshUser = async () => {
    await loadUser();
  };

```

```

    };

    return (
      <AuthContext.Provider value={{ user, loading, login, logout,
refreshUser }}>
        {children}
      </AuthContext.Provider>
    );
  }

export const useAuth = () => {
  const ctx = useContext(AuthContext);
  if (!ctx) throw new Error('useAuth must be used within AuthProvider');
  return ctx;
};

```

Файл SearchModal.js

```

import React, { useEffect } from 'react';

const MIME_TYPES = [
  { label: 'Все типы', value: '' },
  { label: 'Изображения', value: 'image/' },
  { label: 'PDF', value: 'application/pdf' },
  { label: 'Word', value: 'application/vnd.openxmlformats-officedocument.wordprocessingml' },
  { label: 'Excel', value: 'application/vnd.openxmlformats-officedocument.spreadsheetml' },
  { label: 'Видео', value: 'video/' },
  { label: 'Аудио', value: 'audio/' },
  { label: 'Текстовые', value: 'text/' },
  { label: 'Архивы', value: 'application/zip' },
];

export const SEARCH_TYPES = [
  { value: 'name', label: 'По названию', description: 'Поиск файлов и папок по имени' },
  { value: 'fulltext', label: 'Полнотекстовый', description: 'Поиск по содержимому файлов' },
  { value: 'semantic', label: 'Семантический', description: 'Поиск по смыслу.' },
  { value: 'all', label: 'Гибридный', description: 'BM25 + семантика, ранжирование через RRF' },
];

const CloseIcon = () => (
  <svg width="16" height="16" viewBox="0 0 16 16" fill="none">
    <path d="M12 4L4 12M4 4L12 12" stroke="currentColor" strokeWidth="1.5" strokeLinecap="round"/>
  </svg>
);

const ChevronDownIcon = () => (
  <svg width="12" height="12" viewBox="0 0 12 12" fill="none">
    <path d="M2.5 4.5L6 8L9.5 4.5" stroke="currentColor" strokeWidth="1.4" strokeLinecap="round" strokeLinejoin="round"/>
  </svg>
);

```

					ПГУ 1.090301.24.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		181

```

const FolderIcon = () => (
  <svg width="13" height="13" viewBox="0 0 16 16" fill="none">
    <path d="M2 4a1 1 0 011-1h4l1.5 2H13a1 1 0 011 1v6a1 1 0 01-1 1H3a1 1 0 01-1-1V4z" fill="currentColor" opacity="0.8"/>
  </svg>
);

const GlobeIcon = () => (
  <svg width="13" height="13" viewBox="0 0 16 16" fill="none">
    <circle cx="8" cy="8" r="6" stroke="currentColor" strokeWidth="1.4"/>
    <path d="M8 2c0 0-3 2-3 6s3 6 3 6M8 2c0 0 3 2 3 6s-3 6-3 6M2 8h12"
stroke="currentColor" strokeWidth="1.4" strokeLinecap="round"/>
  </svg>
);

export default function SearchFiltersModal({
  isOpen, onClose,
  searchType, setSearchType,
  mimeType, setMimeType,
  dateFrom, setDateFrom,
  dateTo, setDateTo,
  onApply,

  currentFolderUuid,
  currentFolderName,
  folderScope,
  setFolderScope,
}) {
  useEffect(() => {
    const handleKey = (e) => { if (e.key === 'Escape') onClose(); };
    if (isOpen) window.addEventListener('keydown', handleKey);
    return () => window.removeEventListener('keydown', handleKey);
  }, [isOpen, onClose]);

  if (!isOpen) return null;

  const isInsideFolder = !!currentFolderUuid;

  return (
    <div
      className="fixed inset-0 z-50 flex items-center justify-center"
      style={{ backgroundColor: 'rgba(0,0,0,0.3)', backdropFilter:
'blur(2px)' }}
      onClick={(e) => { if (e.target === e.currentTarget) onClose(); }}
    >
      <div className="w-full max-w-md mx-4 bg-white rounded-xl shadow-2xl
overflow-hidden">
        {}
        <div className="flex items-center justify-between px-5 py-4 border-
b border-gray-100">
          <h2 className="text-sm font-semibold text-gray-900">Настройки
поиска</h2>
          <button onClick={onClose} className="p-1.5 text-gray-400
hover:text-gray-600 rounded-lg hover:bg-gray-100 transition-colors">
            <CloseIcon />
          </button>
        </div>
      </div>
    </div>
  );
}

```

```

</div>

<div className="px-5 py-5 space-y-5">

  {}
  {isInsideFolder && (
    <div>
      <label className="block text-xs font-medium text-gray-500
uppercase tracking-wide mb-2">Область поиска</label>
      <div className="flex rounded-lg border border-gray-200
overflow-hidden">
        <button
          onClick={() => setFolderScope('current')}
          className={`flex-1 flex items-center justify-center gap-
1.5 px-3 py-2.5 text-xs font-medium transition-colors ${
            folderScope === 'current'
              ? 'bg-blue-600 text-white'
              : 'bg-white text-gray-600 hover:bg-gray-50'
            }`}
        >
          <FolderIcon />
          <span className="truncate max-w-[120px]"
title={currentFolderName}>
            В папке «{currentFolderName}»
          </span>
        </button>
        <button
          onClick={() => setFolderScope('all')}
          className={`flex-1 flex items-center justify-center gap-
1.5 px-3 py-2.5 text-xs font-medium border-l border-gray-200 transition-
colors ${
            folderScope === 'all'
              ? 'bg-blue-600 text-white border-blue-600'
              : 'bg-white text-gray-600 hover:bg-gray-50'
            }`}
        >
          <GlobeIcon />
          Везде
        </button>
      </div>
    </div>
  )}

  {}
  <div>
    <label className="block text-xs font-medium text-gray-500
uppercase tracking-wide mb-2">Режим поиска</label>
    <div className="space-y-1.5">
      {SEARCH_TYPES.map(t => (
        <label key={t.value} className={`flex items-start gap-3 p-3
rounded-lg border cursor-pointer transition-colors ${
          searchType === t.value ? 'border-blue-300 bg-blue-50' :
'border-gray-200 hover:border-gray-300 hover:bg-gray-50'
        }`}
        >
          <input
            type="radio"
            name="searchType"

```

```

        value={t.value}
        checked={searchType === t.value}
        onChange={() => setSearchType(t.value)}
        className="mt-0.5 accent-blue-600 flex-shrink-0"
      />
    <div>
      <div className={`text-xs font-medium ${searchType ===
t.value ? 'text-blue-700' : 'text-gray-700'}`}>{t.label}</div>
      <div className="text-xs text-gray-400 mt-0.5 leading-
snug">{t.description}</div>
    </div>
  </label>
  )}}
</div>
</div>

{}
<div>
  <label className="block text-xs font-medium text-gray-500
uppercase tracking-wide mb-2">Тип файла</label>
  <div className="relative">
    <select
      value={mimeType}
      onChange={e => setMimeType(e.target.value)}
      className="w-full appearance-none pl-3 pr-8 py-2.5 text-sm
text-gray-700 bg-white border border-gray-200 rounded-lg outline-none
focus:ring-2 focus:ring-blue-500 focus:border-transparent"
    >
      {MIME_TYPES.map(t => <option key={t.value}
value={t.value}>{t.label}</option>)}
    </select>
    <span className="pointer-events-none absolute right-3 top-1/2
-translate-y-1/2 text-gray-400">
      <ChevronDownIcon />
    </span>
  </div>
</div>

{}
<div>
  <label className="block text-xs font-medium text-gray-500
uppercase tracking-wide mb-2">Дата изменения</label>
  <div className="flex gap-3">
    <div className="flex-1">
      <label className="block text-xs text-gray-400 mb-
1">C</label>
      <input
        type="date"
        value={dateFrom}
        onChange={e => setDateFrom(e.target.value)}
        className="w-full px-3 py-2.5 text-sm text-gray-700
border border-gray-200 rounded-lg outline-none focus:ring-2 focus:ring-
blue-500 focus:border-transparent"
      />
    </div>
    <div className="flex-1">
      <label className="block text-xs text-gray-400 mb-

```

```

1">По</label>
      <input
        type="date"
        value={dateTo}
        onChange={e => setDateTo(e.target.value)}
        className="w-full px-3 py-2.5 text-sm text-gray-700
border border-gray-200 rounded-lg outline-none focus:ring-2 focus:ring-
blue-500 focus:border-transparent"
      />
    </div>
  </div>
</div>
</div>

{}
<div className="flex items-center justify-between px-5 py-4 border-
t border-gray-100 bg-gray-50">
  <button
    onClick={() => { setMimeType(''); setDateFrom('');
setDateTo(''); setSearchType('name'); if (isInsideFolder)
setFolderScope('current'); }}
    className="text-xs text-gray-400 hover:text-gray-600
transition-colors"
  >
    Сбросить всё
  </button>
  <div className="flex gap-2">
    <button onClick={onClose} className="px-4 py-2 text-xs font-
medium text-gray-600 border border-gray-200 rounded-lg hover:bg-white
transition-colors">
      Отмена
    </button>
    <button
      onClick={() => { onApply(); onClose(); }}
      className="px-4 py-2 text-xs font-medium text-white bg-gray-
900 rounded-lg hover:bg-gray-700 transition-colors"
    >
      Применить
    </button>
  </div>
</div>
</div>
</div>
);
}

```

Файл ChatPanel.js

```

import React, { useState, useEffect, useRef, useCallback } from 'react';
import { chatApi } from '../services/api';
import toast from 'react-hot-toast';

const SendIcon = () => (
  <svg width="15" height="15" viewBox="0 0 15 15" fill="none">
    <path d="M7.5 2V13M7.5 2L3 6.5M7.5 2L12 6.5" stroke="currentColor"
strokeWidth="1.4"
      strokeLinecap="round" strokeLinejoin="round"/>
  </svg>
);

```

					ПГУ 1.090301.24.001 ПЗ	Лист
						185
Изм.	Лист	№ докум.	Подпись	Дата		

```

    </svg>
  );

  const TrashIcon = () => (
    <svg width="14" height="14" viewBox="0 0 14 14" fill="none">
      <path d="M2 3.5H12" stroke="currentColor" strokeWidth="1.2"
strokeLinecap="round"/>
      <path d="M5 3.5V2.5H9V3.5" stroke="currentColor" strokeWidth="1.2"
strokeLinejoin="round"/>
      <path d="M3 3.5L3.8 11H10.2L11 3.5" stroke="currentColor"
strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round"/>
    </svg>
  );

  const CloseIcon = () => (
    <svg width="16" height="16" viewBox="0 0 16 16" fill="none">
      <path d="M4 4L12 12M12 4L4 12" stroke="currentColor" strokeWidth="1.4"
strokeLinecap="round"/>
    </svg>
  );

  const BotIcon = () => (
    <svg width="14" height="14" viewBox="0 0 14 14" fill="none">
      <rect x="2" y="4" width="10" height="7" rx="2" stroke="currentColor"
strokeWidth="1.2"/>
      <path d="M5 7.5C5 7.5 5.5 8.5 7 8.5C8.5 8.5 9 7.5 9 7.5"
stroke="currentColor"
strokeWidth="1.1" strokeLinecap="round"/>
      <circle cx="5" cy="6.5" r="0.6" fill="currentColor"/>
      <circle cx="9" cy="6.5" r="0.6" fill="currentColor"/>
      <path d="M7 4V2.5" stroke="currentColor" strokeWidth="1.2"
strokeLinecap="round"/>
      <circle cx="7" cy="2" r="0.6" fill="currentColor"/>
    </svg>
  );

  function TypingDots() {
    return (
      <span className="inline-flex gap-1 items-center px-1">
        {[0, 150, 300].map(delay => (
          <span
            key={delay}
            className="w-1.5 h-1.5 rounded-full bg-gray-400"
            style={{ animation: `chatBounce 1s ${delay}ms ease-in-out
infinite` }}
          />
        ))}
        <style>{`
          @keyframes chatBounce {
            0%, 80%, 100% { transform: translateY(0); opacity: 0.4; }
            40% { transform: translateY(-4px); opacity: 1; }
          }
        `}</style>
      </span>
    );
  }

```

```

function MessageBubble({ role, content, createdAt, isStreaming }) {
  const isUser = role === 'user';

  const timeLabel = createdAt
    ? new Date(createdAt).toLocaleTimeString([], { hour: '2-digit', minute:
'2-digit' })
    : null;

  const renderContent = (text) => {
    if (!text) return isStreaming ? <TypingDots /> : '';
    return text.split('\n').map((line, i, arr) => (
      <React.Fragment key={i}>
        {line}
        {i < arr.length - 1 && <br />}
      </React.Fragment>
    ));
  };

  return (
    <div className={`flex gap-2 ${isUser ? 'flex-row-reverse' : 'flex-
row'}}`>
      {}
      {!!isUser && (
        <div className="flex-shrink-0 w-6 h-6 rounded-full bg-indigo-100
text-indigo-500
flex items-center justify-center mt-0.5">
          <BotIcon />
        </div>
      )}

      {}
      <div className={`flex flex-col gap-0.5 max-w-[82%] ${isUser ? 'items-
end' : 'items-start'}`>
        <div
          className={`rounded-2xl px-3.5 py-2.5 text-sm leading-relaxed
${isUser
  ? 'bg-indigo-500 text-white rounded-tr-sm'
  : 'bg-gray-100 text-gray-800 rounded-tl-sm'
}`}
        >
          {renderContent(content)}
        </div>
        {timeLabel && (
          <span className="text-[10px] text-gray-400 px-
1">{timeLabel}</span>
        )}
      </div>
    </div>
  );
}

function EmptyState({ fileName }) {
  const suggestions = [
    'О чём этот документ?',
    'Перечисли ключевые пункты',
    'Сделай краткое резюме',
  ];
}

```

```

];
return (
  <div className="flex flex-col items-center justify-center h-full px-6
text-center gap-5">
    <div className="w-12 h-12 rounded-2xl bg-indigo-50 text-indigo-400
flex items-center justify-center text-xl">
      <BotIcon />
    </div>
    <div>
      <p className="font-medium text-gray-700 text-sm mb-1">AI-
ассистент</p>
      <p className="text-xs text-gray-400 leading-relaxed">
        Задавай вопросы по содержанию<br/>
        <span className="font-medium text-gray-500
truncate">{fileName}</span>
      </p>
    </div>
    <div className="flex flex-col gap-2 w-full">
      {suggestions.map(s => (
        <button
          key={s}
          className="text-xs text-left px-3 py-2 rounded-xl border
border-gray-200
                    hover:border-indigo-300 hover:bg-indigo-50 text-
gray-500
                    hover:text-indigo-600 transition-colors"
          onClick={() => {
            document.dispatchEvent(new CustomEvent('chat-suggestion', {
detail: s }));
          }}
        >
          {s}
        </button>
      ) )}
    </div>
  </div>
);
}

export default function ChatPanel({ file, onClose }) {
  const [messages, setMessages] = useState([]);
  const [input, setInput] = useState('');
  const [streaming, setStreaming] = useState(false);
  const [loading, setLoading] = useState(true);

  const bottomRef = useRef(null);
  const inputRef = useRef(null);
  const esRef = useRef(null);

  useEffect(() => {
    let cancelled = false;
    chatApi.getHistory(file.uuid)
      .then(res => {
        if (!cancelled) setMessages(res.data.data || []);
      })
      .catch(() => toast.error('Не удалось загрузить историю чата'));
  });

```

```

        .finally(() => { if (!cancelled) setLoading(false); });

    return () => { cancelled = true; };
}, [file.uuid]);

useEffect(() => {
    bottomRef.current?.scrollIntoView({ behavior: 'smooth' });
}, [messages]);

useEffect(() => {
    const handler = (e) => {
        setInput(e.detail);
        inputRef.current?.focus();
    };
    document.addEventListener('chat-suggestion', handler);
    return () => document.removeEventListener('chat-suggestion', handler);
}, []);

useEffect(() => {
    return () => { esRef.current?.close(); };
}, []);

const send = useCallback(() => {
    const question = input.trim();
    if (!question || streaming) return;
    setInput('');

    setMessages(prev => [
        ...prev,
        { role: 'user', content: question, createdAt: new
Date().toISOString() },
        { role: 'assistant', content: '', createdAt: new
Date().toISOString(), _streaming: true },
    ]);
    setStreaming(true);

    const es = chatApi.openStream(file.uuid, question);
    esRef.current = es;

    es.addEventListener('token', (e) => {
        setMessages(prev => {
            const copy = [...prev];
            const last = copy[copy.length - 1];
            const token = e.data.replace(/\n/g, "\n").replace(/\s/g, " ");
            copy[copy.length - 1] = { ...last, content: last.content + token };
            return copy;
        });
    });

    es.addEventListener('done', () => {
        es.close();
        setStreaming(false);

        setMessages(prev => {
            const copy = [...prev];
            const last = copy[copy.length - 1];
            copy[copy.length - 1] = { ...last, _streaming: false };
        });
    });
}, [file.uuid, chatApi]);

```

```

        return copy;
    });
});

es.addEventListener('error', (e) => {
    es.close();
    setStreaming(false);
    const msg = e.data || 'Ollama недоступен';
    toast.error('Ошибка ответа: ' + msg);
    setMessages(prev => {
        const copy = [...prev];
        copy[copy.length - 1] = {
            ...copy[copy.length - 1],
            content: '⚠ ' + msg,
            _streaming: false,
        };
        return copy;
    });
});

es.onerror = () => {
    if (es.readyState === EventSource.CLOSED) return;
    es.close();
    setStreaming(false);
    toast.error('Соединение прервано');
};

}, [input, streaming, file.uuid]);

const clearHistory = async () => {
    if (!window.confirm('Очистить историю переписки?')) return;
    try {
        await chatApi.clearHistory(file.uuid);
        setMessages([]);
        toast.success('История очищена');
    } catch {
        toast.error('Не удалось очистить историю');
    }
};

return (
    <div className="flex flex-col h-full bg-white border-l border-gray-200"
        style={{ width: 340, minWidth: 300 }}>

        {}
        <div className="flex items-center gap-3 px-4 py-3 border-b border-
gray-200 flex-shrink-0">
            <div className="w-8 h-8 rounded-xl bg-indigo-100 text-indigo-500
                flex items-center justify-center flex-shrink-0">
                <BotIcon />
            </div>

            <div className="flex-1 min-w-0">
                <p className="text-xs font-medium text-gray-800
truncate">{file.name}</p>
                <p className="text-xs text-gray-400">AI-ассистент</p>
            </div>

```

```

<div className="flex items-center gap-1 flex-shrink-0">
  {messages.length > 0 && (
    <button
      onClick={clearHistory}
      title="Очистить историю"
      className="p-1.5 rounded-lg text-gray-400 hover:text-red-500
hover:bg-red-50
      transition-colors"
    >
      <TrashIcon />
    </button>
  )}
  <button
    onClick={onClose}
    title="Закрыть"
    className="p-1.5 rounded-lg text-gray-400 hover:text-gray-600
hover:bg-gray-100
    transition-colors"
  >
    <CloseIcon />
  </button>
</div>
</div>

{}
<div className="flex-1 overflow-y-auto px-4 py-4 space-y-4 min-h-0">
  {loading ? (
    <div className="flex justify-center items-center h-full">
      <div className="w-5 h-5 border-2 border-indigo-300 border-t-
indigo-500
        rounded-full animate-spin"/>
    </div>
  ) : messages.length === 0 ? (
    <EmptyState fileName={file.name} />
  ) : (
    messages.map((m, i) => (
      <MessageBubble
        key={i}
        role={m.role}
        content={m.content}
        createdAt={m.createdAt}
        isStreaming={m._streaming}
      />
    ))
  )}
  <div ref={bottomRef} />
</div>

{}
<div className="flex-shrink-0 px-3 py-3 border-t border-gray-200">
  <div className="flex gap-2 items-end">
    <textarea
      ref={inputRef}
      value={input}
      onChange={e => setInput(e.target.value)}
      onKeyDown={e => {

```

```

        if (e.key === 'Enter' && !e.shiftKey) {
            e.preventDefault();
            send();
        }
    }}
    placeholder="Задай вопрос по документу..."
    disabled={streaming}
    rows={1}
    className="flex-1 text-sm border border-gray-200 rounded-xl px-
3 py-2.5
                focus:outline-none focus:ring-2 focus:ring-indigo-
300 focus:border-indigo-300
                disabled:bg-gray-50 resize-none leading-relaxed"
    style={{ maxHeight: 120, overflowY: 'auto' }}
    onInput={e => {

        e.target.style.height = 'auto';
        e.target.style.height = Math.min(e.target.scrollHeight, 120)
+ 'px';
    }}
    />
    <button
        onClick={send}
        disabled={streaming || !input.trim()}
        title="Отправить (Enter)"
        className="flex-shrink-0 w-9 h-9 bg-indigo-500 text-white
rounded-xl
                flex items-center justify-center
                hover:bg-indigo-600 active:scale-95 transition-all
                disabled:opacity-40 disabled:cursor-not-allowed"
    >
        {streaming
            ? <div className="w-3.5 h-3.5 border-2 border-white/40
border-t-white
                rounded-full animate-spin"/>
            : <SendIcon />
        }
    </button>
</div>
<p className="text-xs text-gray-400 mt-1.5 px-1">
    Enter – отправить · Shift+Enter – новая строка
</p>
</div>
</div>
);
}

```

Файл App.js

```

import React from 'react';
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import { Toaster } from 'react-hot-toast';
import { AuthProvider } from '../context/AuthContext';
import ProtectedRoute from '../components/common/ProtectedRoute';
import DashboardLayout from '../components/common/DashboardLayout';
import LoginPage from '../components/auth/LoginPage';
import RegisterPage from '../components/auth/RegisterPage';

```

```

import { ForgotPasswordPage, ResetPasswordPage } from
'./components/auth/PasswordPages';
import EmailVerificationPage from
'./components/auth/EmailVerificationPage';
import FileListPage from './components/files/FileListPage';
import SharedWithMePage from './components/files/SharedWithMePage';
import TrashPage from './components/files/TrashPage';
import SharingPage from './components/sharing/SharingPage';
import PublicSharePage from './components/sharing/PublicSharePage';
import ProfilePage from './components/profile/ProfilePage';

export default function App() {
  return (
    <AuthProvider>
      <BrowserRouter>
        <Toaster
          position="top-right"
          toastOptions={{
            duration: 3500,
            style: { borderRadius: '10px', fontSize: '14px' }
          }}
        />
        <Routes>
          {}
          <Route path="/login"           element={<LoginPage />} />
          <Route path="/register"        element={<RegisterPage />} />
          <Route path="/forgot-password" element={<ForgotPasswordPage />} />
        />
        <Route path="/reset-password"   element={<ResetPasswordPage />} />
        />
        <Route path="/verify-email"     element={<EmailVerificationPage />} />
      />
    />
    {}
    <Route path="/s/:token" element={<PublicSharePage />} />
    {}
    <Route path="/" element={
      <ProtectedRoute>
        <DashboardLayout />
      </ProtectedRoute>
    }>
      <Route index           element={<FileListPage />} />
      <Route path="folder/:uuid" element={<FileListPage />} />
      <Route path="shared"    element={<SharedWithMePage />} />
      <Route path="trash"     element={<TrashPage />} />
      <Route path="sharing/:uuid" element={<SharingPage />} />
      <Route path="profile"   element={<ProfilePage />} />
    </Route>
    <Route path="*" element={<Navigate to="/" replace />} />
  </Routes>
</BrowserRouter>
</AuthProvider>
);
}

```

Приложение В
(обязательное)

Облачное хранилище с управлением доступом и системой поиска
Презентация

Облачное хранилище с управлением доступом и системой поиска

Автор работы:
Студент гр. 22ВВІІ Хоссейин Нежад Сеййед Али Сеййед Мортеза

Научный руководитель:
к.т.н., доцент кафедры ВТ ПГУ Юрова Ольга Викторовна

Актуальность проблемы

- Низкий уровень доступности локальных данных
- Конфиденциальность и безопасность данных. Необходимо разграничение прав доступа.
- Огромное количество файлов затрудняет их нахождение, необходима продвинутая система поиска

Цели и задачи работы

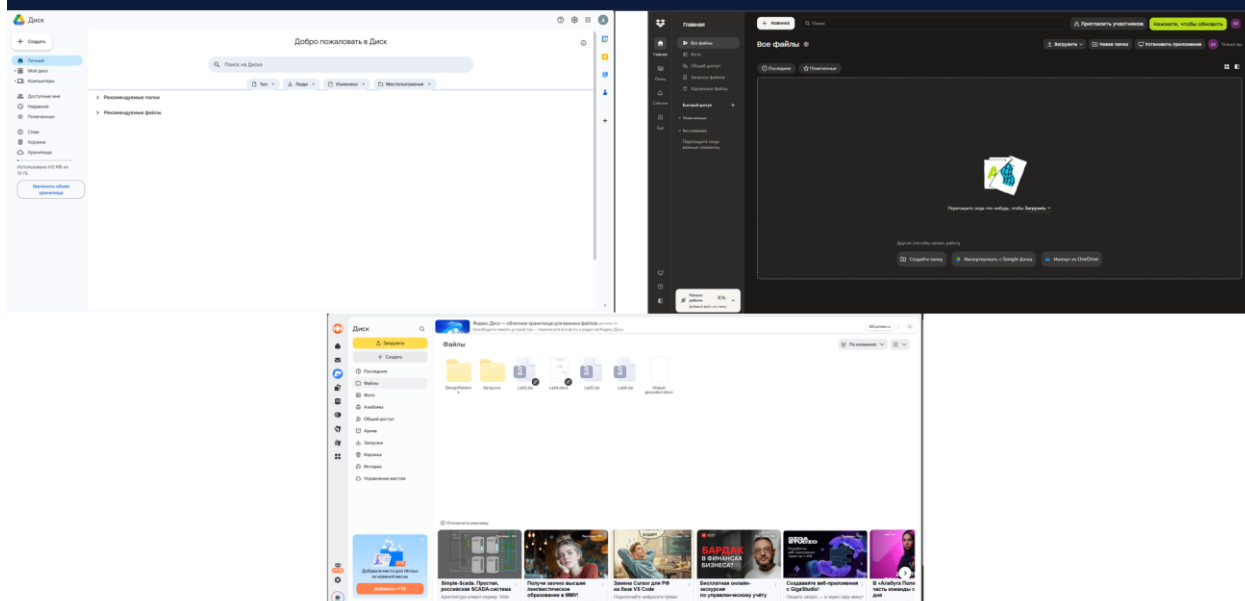
Цель работы:

Разработать веб-приложение облачного хранилища с управлением доступом и системой поиска

Задачи:

- Изучить актуальность темы и существующие аналоги
- Определить функционал
- Определить инструменты и технологии для разработки
- Разработать архитектуру и само веб-приложение
- Протестировать веб-приложение

Аналогичные облачные хранилища



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.24.001 ПЗ

Лист

196

Сравнение аналогов

Google Диск	Dropbox	Яндекс.Диск
Преимущества	Преимущества	Преимущества
Полнотекстовый поиск	Разграничение прав доступа	Доступ по ссылке
Интеграция с Google	Доступ по ссылке	Интеграция с Яндекс
Разграничение прав доступа		
Недостатки	Недостатки	Недостатки
Нет семантического поиска	Ограниченный поиск	Ограниченный поиск
Нет AI-чата	Нет AI-чата	Нет AI-чата
		Ограниченные права доступа

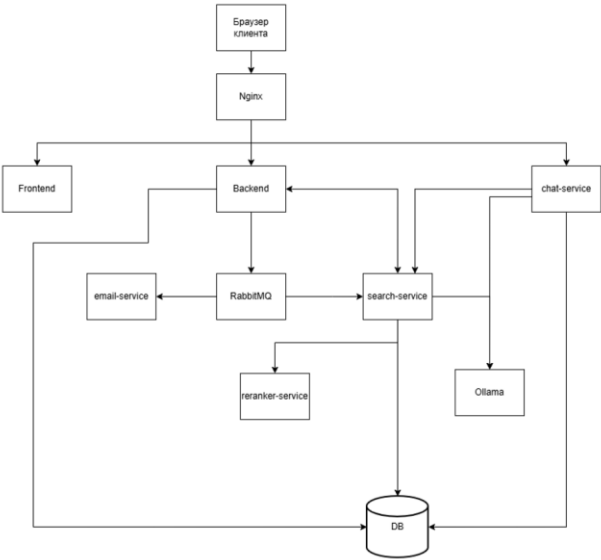
Функционал разработанного веб-приложения

- Регистрация и авторизация пользователей / восстановление пароля;
- Выход / удаление учетной записи;
- Управление профилем пользователем;
- Загрузка, скачивание, удаление, перемещение и переименование файлов;
- Удаление, перемещение и переименование папок;
- Сортировка объектов хранилища;
- Наличие корзины и возможность восстановления объектов из нее;
- Отображение индикатора занятого дискового пространства;
- Предоставление доступа к файлам и папкам другим пользователям с указанием уровня прав (Читатель, Автор, Редактор, Владелец);
- Создание публичных ссылок к объектам с указанием уровня прав (Читатель, Автор, Редактор);
- Система поиска: поиск по имени, типу, дате файла, полнотекстовый и семантический поиск;
- AI-чат по файлу.

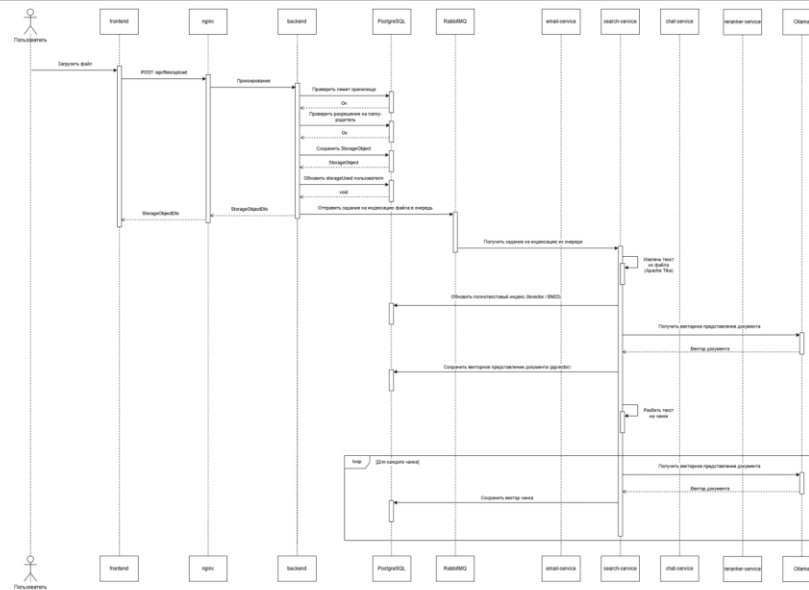
Используемые технологии



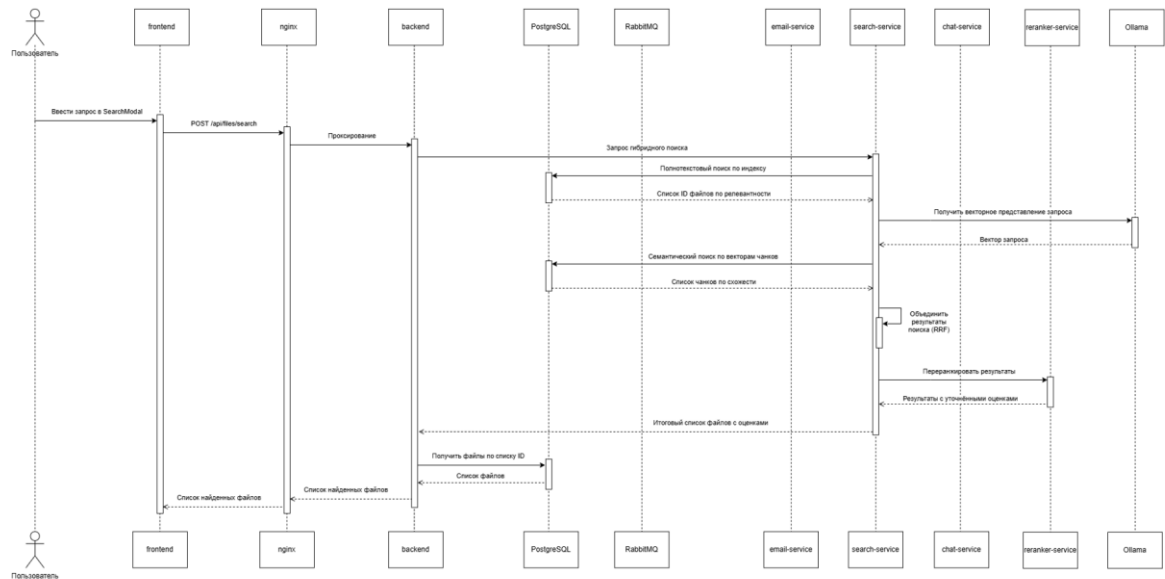
Архитектура веб-приложения



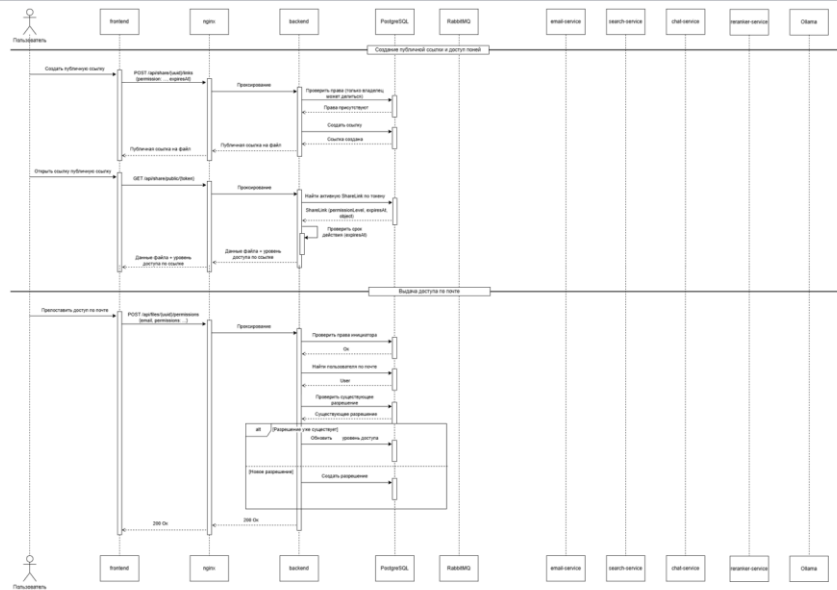
UML-диаграмма последовательности загрузки файла



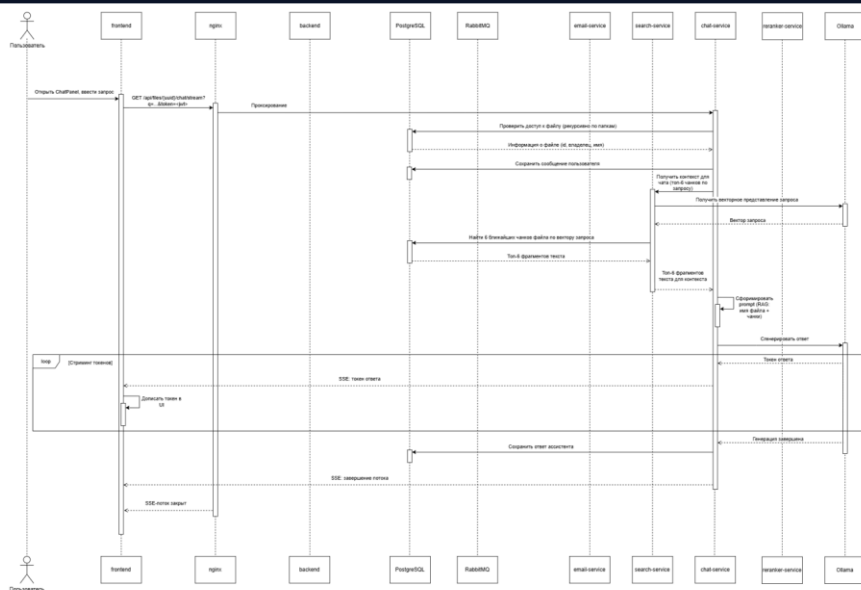
UML-диаграмма последовательности гибридного поиска



UML-диаграмма последовательности выдачи прав



UML-диаграмма последовательности AI-чата



Интерфейс выбора фильтров поиска

Настройки поиска

ОБЛАСТЬ ПОИСКА

В папке «Work»

Везде

РЕЖИМ ПОИСКА

По названию

Поиск файлов и папок по имени

Полнотекстовый

Поиск по содержимому файлов

Семантический

Поиск по смыслу

Гибридный

BM25 + семантика, ранжирование через RRF

ТИП ФАЙЛА

Все типы

ДАТА ИЗМЕНЕНИЯ

С

По

ДД.ММ.ГГГГ

ДД.ММ.ГГГГ

Сбросить все

Отмена

Применить

Интерфейс управления правами доступа

Доступ к файлу

Lab1.docx

Пользователи

Ссылки

Email пользователя

Читатель

Читатель

Автор

Редактор

Владелец

Дать доступ

Мои файлы

Управление доступом

Lab1.docx

Пользователи

Ссылки

Email пользователя

Читатель

Добавить

Доступ не предоставлен ни одному пользователю

Доступ к файлу

Lab1.docx

Пользователи

Ссылки

Читатель

ДД.ММ.ГГГГ --:--

Дать ссылку

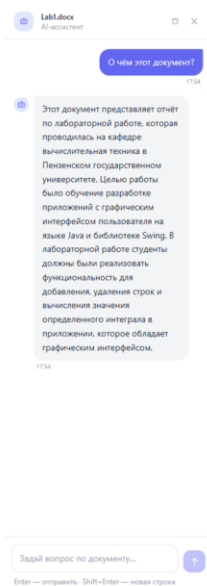
Читатель

Автор

Редактор

Нет активных ссылок

Интерфейс AI-чата



Заключение

В ходе выполнения выпускной квалификационной работы была изучена актуальность облачных хранилищ, анализ аналогов, были выбраны технологии и среды разработки, спроектирована архитектура, реализовано веб-приложение «Облачное хранилище с управлением доступом и системой поиска», а также было проведено тестирование.

Спасибо за внимание!