МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Модернизация ИТ-инфраструктуры предприятия для защиты данных и обеспечения бесперебойной работы

Студент _____ Моисеев Матвей Максимович
(подпись, дата) (ФИО полностью)

Руководитель _____ Попова Н.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026г № 12)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«26» декабря 2025 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент _____Моисеев Матвей Максимович_____гр.___22ВИ1_____факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы _Модернизация ИТ-инфраструктуры предприятия для защиты данных и_ обеспечения бесперебойной работы

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

2. Руководитель работы __Попова Н. А.

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

Назначение: Обеспечение защищенного обмена файлами внутри предприятия с возможностью адресной передачи документов конкретному сотруднику, с аудитом операций и мониторингом состояния сервиса для повышения защищённости и управляемости ИТ-инфраструктуры

Основные функции системы:

1. Авторизация пользователей и разграничение прав доступа
2. Загрузка файлов пользователей через веб-интерфейс
3. Шифрование файлов перед сохранением на сервере
4. Адресная передача файла сотруднику
5. Скачивание файла получателем
6. Ограничение доступа к передаче
7. Введение журнала аудита операций в базе данных
8. Дублирование событий аудита в файловый журнал
9. Мониторинг состояния сервера и просмотр аудита

Технология разработки: объектно-ориентированный и модульный подход клиент-серверной и веб-архитектуры

Языки программирования: Python, JavaScript

Среда программирования: PyCharm

4. Перечень подлежащих разработке вопросов:

1. анализ предметной области и постановка задачи;
2. проектирование программных средств;
3. реализация программных средств;
4. тестирование программных средств;
5. оформление ВКР.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 116 листов, 21 таблицу, 38 рисунков, 22 использованных источника и 3 приложения

ЗАЩИТА ДАННЫХ, WEB-СЕРВИС, PYTHON, FLASK, JAVASCRIPT WSGI, SQLITE, HTML, CSS, HTTP, WINDOWS, АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР, БАЗЫ ДАННЫХ. ВНУТРЕННИЙ КОРПОРАТИВНЫЙ СЕРВИС

Объект разработки: Модернизация ИТ-инфраструктуры предприятия для защиты данных и обеспечения бесперебойной работы

Предмет разработки: разработка веб-сервиса для обеспечения защищенного обмена файлами внутри предприятия с возможностью адресной передачи документов конкретному сотруднику, с аудитом операций и мониторингом состояния сервиса для повышения защищённости и управляемости ИТ-инфраструктуры.

Цель работы: разработать программное обеспечение для реализации веб-сервиса для защищённой передачи данных.

Результаты работы: Спроектирован веб-сервис для защищённой передачи данных внутри предприятия

Средства разработки: PyCharm, Flask, WSGI, SQLite, Python, JavaScript, HTML, CSS

Содержание

Введение 6
1 Постановка задачи и анализ требований 8
1.1 Постановка задачи 8
1.2 Анализ программного обеспечения аналогичного назначения 9
1.3 Формулировка и анализ требований к разрабатываемому программному обеспечению 11
2 Проектирование 13
2.1 Проектирование бизнес-процессов системы 13
2.2 Проектирование базы данных 14
2.3 Проектирование интерфейса 18
3 Программная реализация 24
3.1 Программная реализация клиентской части системы 24
3.2 Программная реализация серверной части 30
3.3 Программная реализация базы данных 39
4 Тестирование 45
4.1 Тестирование методом белого ящика 45
4.3 Расчёт метрик кода тестирования 48
4.4 Нагрузочное тестирование 56
4.4 Тестирование интерфейса веб-сервиса 59
5. Планирование и расчёт бюджета разрабатываемой системы 62
Заключение 65
Список использованных источников 67
Приложение А Программный код – клиентская часть (обязательное) 69
Приложение Б Программный код – серверная часть (обязательное) 85
Приложение В Тестирование интерфейса (рекомендуемое) 112

Введение

Работа любой компании держится на файлах. Договоры, отчёты, технические спецификации, медиа. Всё это постоянно перемещается между людьми. Проблемы начинаются там, где для этого перемещения используют личную почту, мессенджеры и публичные файлообменники. Контроль исчезает мгновенно. Нельзя ограничить срок доступа, отозвать уже отправленную ссылку, понять, кто и когда скачал документ. Одна ошибка в адресе, одна лишняя пересылка и служебный файл превращается в неконтролируемую копию, о существовании которой компания уже не знает. Это прямой удар по информационной безопасности и по соблюдению внутренних правил.

Утечками дело не ограничивается. Даже внутри корпоративной сети система может отказать. Например, из-за переполненного диска, или зависшей фоновой задачи. Без нормального мониторинга и журнала событий администратор видит лишь факт недоступности сервера. Причины он ищет вслепую, полагаясь на свой опыт. Разбор инцидента затягивается, простои растут, а пользователи тем временем переходят в неофициальные каналы и усугубляют то, что уже пошло не так.

Облачное хранилище проблему не решает. Оно добавляет зависимость от третьих сторон, что для многих организаций неприемлемо. Требования по защите информации и режиму обработки данных это исключают. Поэтому выбор делают в пользу решений, которые разворачиваются внутри корпоративной сети, которые строятся на свободно распространяемых и отечественных компонентах и позволяют управлять доступом.

Именно под такую среду проектируется внутренний сервис обмена файлами с понятной логикой работы. Доступ назначается через роли и явные разрешения. Внешняя передача реализована через адресные ссылки с ограничением по времени и числу скачиваний. Все значимые действия фиксируются в журнале так, чтобы любая подмена была заметна. Состояние

системы администратор видит без дополнительных инструментов, прямо из интерфейса.

Основная задача не создать конкурента полноценным системам электронного документа оборота. Главное закрыть потребность безопасного и управляемого обмена файлами в масштабах малого и среднего предприятия.

Объект исследования ИТ-инфраструктура предприятия среднего масштаба, где требуется организованный и контролируемый обмен служебными файлами.

Предметом исследования, являются программные средства построения внутреннего сервиса обмена с разграничением прав, аудитом действий пользователей и базовым мониторингом.

Выпускная квалификационная работа ставит одну конкретную цель – разработать программный комплекс для защищённого обмена файлами внутри предприятия, который обеспечивает контроль доступа, проверяемый журнал всех операций и средства наблюдения за состоянием системы в реальном времени.

1 Постановка задачи и анализ требований

1.1 Постановка задачи

Сервис разрабатывается для развёртывания внутри корпоративной сети предприятия среднего масштаба. Главная задача, это убрать риски, которые возникают, когда сотрудники передают служебные файлы через личную почту, мессенджеры и внешние файлообменники. Нецентрализованные каналы не оставляют следов, а сервис оставляет.

Все файлы шифруются на сервере перед сохранением. Клиентское шифрование не применяется, ключи хранятся на стороне сервера. Права разграничены по двум ролям: пользователь и администратор. Для пользователей, которым нужно отослать файл вне системы, предусмотрена возможность создания уникальной ссылкой. Каждая такая ссылка живёт ограниченное время, имеет лимит по объёму и допустимым типам данных. Доступ к любому передаваемому объекту временный и отзывной. Используется TTL, лимит скачиваний, немедленный отзыв при необходимости. Каждое действие фиксируется в журнале: время, отправитель, получатель, совершённое действие, результат, атрибуты клиента. Вся информация хранится в СУБД. Параллельно ведётся файловый журнал с хеш-цепочкой. Если кто-то попытается подправить записи, это будет видно.

Администратор наблюдает за состоянием системы через веб-панель. Ему видна полная информация о всём: загрузка процессора, память, дисковое пространство. Показатели обновляются каждые 5-10 секунд. Никаких сторонних инструментов мониторинга не требуется.

Сервис разворачивается на операционной системе Windows, но также возможен запуск на GNU/Linux, работает в браузере. В текущем варианте сервиса, трафик идёт по HTTP внутри доверенного сегмента сети – это упрощение для показа программного обеспечения. При реальном внедрении система публикуется через reverse-proxy с HTTPS/TLS. Расчётная нагрузка,

предполагает до нескольких десятков одновременно активных пользователей, до 20 параллельных скачиваний, максимальный размер файла – 16 МБ.

Система развёртывалась на ОС Windows 10-11. Возможна установка на системы типа Linux, для этого необходим несколько другой подход сборки разрабатываемого сервиса.

Итогом работы станет готовый программный продукт, веб-сервис с пользовательским интерфейсом и полным комплектом испытаний: модульных по методу белого ящика и нагрузочного. Тесты подтверждают, что система делает именно то, для чего проектировалась.

1.2 Анализ программного обеспечения аналогичного назначения

Готовых решений для обмена файлами внутри корпоративной сети предприятия довольно много. Для сравнения выбраны два наиболее распространённых продукта, которые разворачиваются в закрытой сети и работают на Windows и Linux – Nextcloud и Seafile.

Nextcloud – это платформа частного облачного хранения, с большой экосистемой модулей. Умеет создавать ссылки с паролем и сроком действия, поддерживает сценарий загрузки без доступа к остальному содержимому, о за этой широтой есть обратная сторона. Лимит скачиваний на ссылку не базовая функция, его подключают через сторонние приложения. Детальный аудит с проверяемой целостностью и мониторинг выносятся во внешние системы. Чем богаче функциональность, тем выше требования к администрированию, и тем сложнее удерживать конфигурацию безопасности в минимально достаточном, но строгом состоянии [1].

Seafile сделан иначе. Акцент делается на хранение и синхронизацию при скромных ресурсных требованиях. Публичные и загрузочные ссылки со сроком действия есть. Встроенного лимита скачиваний на ссылку нет. Аудита с проверяемой целостностью записей также не предусмотрено. Мониторинг присутствует в базовом виде. Администрировать проще, но гибкость адресной передачи уступает Nextcloud [2].

Разрабатываемый сервис идёт другим путём. Сужается область до одной задачи, а именно: контролировать адресную передачу файлов с серверным шифрованием при хранении, простым RBAC, TTL и лимитом скачиваний. Никакой экосистемы модулей, внешних зависимостей для критичных функций. Аудит встроен с минимально необходимым набором полей и хеш-цепочкой, которая делает подмену записей обнаруживаемой. Мониторинг ключевых метрик доступен прямо в веб-панели, без подключения сторонних систем. Детальное сравнение по признакам приведено в таблице 1.

Таблица 1 – Сравнение разрабатываемой системы с аналогичными решениями

Критерий сравнения	Nextcloud	Seafile	Разрабатываемая система
Развёртывание внутри корпоративного контура	+	+	+
Разграничение прав (пользователи/роли)	+	+	+
Адресная передача через «Входящие»	+	+	+
Срок действия ссылок (TTL)	+	+	+
Лимит числа скачиваний	-	-	+
Пароль/одноразовый токен на ссылку	+	+	+
Шифрование при хранении	-	+	+
Аудит операций (вход, загрузка, скачивание, ошибки)	+	-	+
Контроль целостности журнала	-	-	+
Требования к СУБД	+	+	+
Поддержка мобильных клиентов	+	+	-
Экосистема расширений и интеграций	+	+	-

В ходе проведённого сравнения разрабатываемой системы с сервисами-аналогами, было выявлено, что разрабатываемая система обладает почти всеми основными функциями из аналогичных систем, а также имеет в своем составе

уникальные функции, которые выделяют ее от аналогичных систем. К данным функциям можно отнести контроль целостности журнала и поддержку лимита числа скачиваний.

1.3 Формулировка и анализ требований к разрабатываемому программному обеспечению

На основе рассмотренных выше разделов, можно подвести итог и описать разрабатываемую систему и набор ее основных функций в виде диаграммы вариантов использования. Диаграмма вариантов использования приведена на рисунке 1.

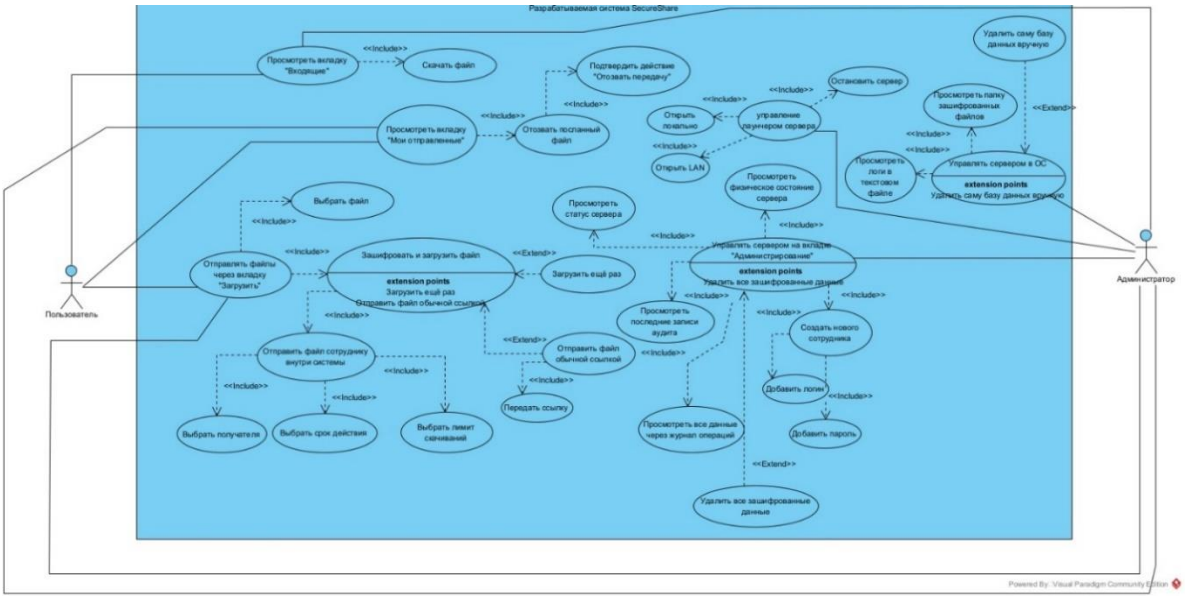


Рисунок 1 – Диаграмма вариантов использования

В системе два вида пользователей. Обычный пользователь и Администратор. Администратор наследует всё, что доступно пользователю, и получает сверху три дополнительные точки управления: вкладку «Администрирование» в веб-интерфейсе, прямой доступ к серверу через ОС и управление лаунчером сервиса. Разделение намеренно жёсткое, рядовой пользователь раздел администрирования не видит вообще.

Пользователь работает в трёх сценариях. Загружает файл и отправляет его через ссылку, с указанием адресата, TTL и лимита скачиваний. Просматривает «Мои отправленные», там исходящие передачи и внешние ссылки со статусами

и счётчиками, оттуда же можно отозвать доступ к файлу. Работает со вкладкой «Входящие», где виден список полученных передач с переходом к скачиванию. Отзыв работает немедленно, после него последующие скачивания блокируются, факт отзыва фиксируется в аудите.

Отдельный сценарий – внешняя ссылка. Она генерируется с TTL в одну минуту, передаётся за пределы системы, например, просто копируется в буфер и вставляется в строку браузера. Получатель без учётной записи скачивает файл. Никакого доступа к остальным ресурсам у него нет.

Все файлы хранятся в зашифрованном виде в выделенной папке сервера. Учётные записи создаёт только администратор. Новые пользователи получают права уровня «Пользователь».

Функциональные требования:

– авторизация и роли;
– управление учётными записями;
– шифрование на хранении;
– управление файлами;
– отправка с параметрами;
– внешний доступ;
– отзыв доступа;
– аудит.

Нефункциональные требования:

– наблюдаемость: структурированные логи, доступные администратору в полном объёме;
– безопасность хранения: файлы на диске только в зашифрованном виде;
– среда эксплуатации: система разворачивается исключительно внутри корпоративной сети, работает только по HTTP.

2 Проектирование

2.1 Проектирование бизнес-процессов системы

Для проектирования основного алгоритма, реализующего набор главных функций приложения, была выбрана диаграмма деятельности, которая описывает алгоритмы, необходимые для осуществления основной деятельности пользователя в разрабатываемой системе. Диаграмма деятельности приведена на рисунке 2.

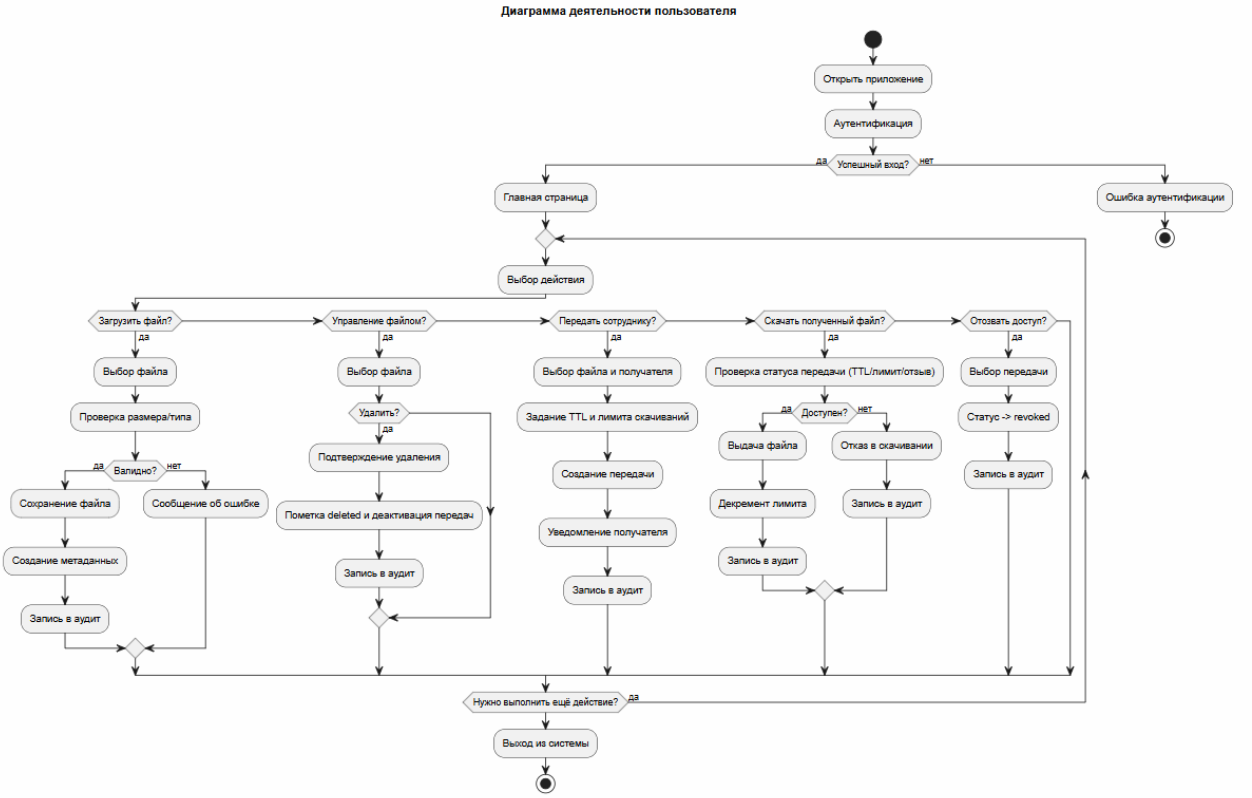


Рисунок 2 – Диаграмма деятельности

Диаграмма фиксирует базовый сценарий работы пользователя от входа до выхода из системы. Каждая функция расписана пошагово, что происходит, в какой последовательности, и что записывается в аудит.

Система покрывает несколько сценариев:

– аутентификация;
– работа с файлами;
– передача сотруднику;

– скачивание;

– отзыв доступа.

При аутентификации делается проверка подлинности пользователя через логин и пароль. При Загрузке проверяется размер и тип, файл сохраняется, создаются метаданные. Удаляя, пользователь подтверждает действие, файл получает метку deleted, связанные передачи деактивируются. Чтобы передать сотруднику данные, выбирается файл и получатель, задаётся TTL и лимит скачиваний, система создаёт передачу и уведомляет получателя. перед выдачей файла для скачивания, система проверяет TTL, лимит и статус отзыва, если условия нарушены, в скачивании отказывается. Отзыв доступа предполагает перевод передачи файла в состояние revoked немедленно. Каждое из этих действий оставляет запись в аудите.

2.2 Проектирование базы данных

В качестве базы данных для разрабатываемой системы была выбрана база данных SQLite. Такое решение обусловлено одноузловой архитектурой приложения и требованием к простому развертыванию без отдельного СУБД-сервера. SQLite хранит данные в одном файле, поддерживает полноценные ACID-транзакции, внешние ключи и журнальный режим, а также интегрируется с Python/Flask. Это упрощает установку при упаковке приложения в единый исполняемый файл, резервное копирование и перенос между машинами. При этом производительности и функциональности SQLite достаточно для сценариев обмена файлами и аудита в рамках локального или малонагруженного сервера.

Логическая структура данных отражает четыре ключевые сущности предметной области. Учетные записи пользователей, сами файлы как объекты с метаданными, передачи, а также журнал действий. Каждая сущность вынесена в отдельную таблицу с явными связями и ограничениями целостности. Такой подход обеспечивает читаемость модели, упрощает сопровождение и позволяет точно контролировать бизнес-правила на уровне БД.

Таблица user хранит учетные записи. Уникальное имя пользователя, хэш пароля, признак администратора и активности, даты создания и последнего входа. Уникальный индекс по username обеспечивает быструю аутентификацию и предотвращает дублирование логинов. Подробная информация о file_record представлена в таблице 2.

Таблица 2 – Описание таблицы базы данных user

id	username	password_hash	is_admin
Идентификатор записи	Имя учётной записи	Зашифрованный пароль учётной записи	Фла,г показывающий является ли текущий пользователь администратором, 0 – нет, 1 – да

Таблица file_record описывает загруженные файлы. Для каждого файла сохраняется публичный токен, владелец, исходное имя и тип содержимого, размер и контрольная сумма, а также безопасное имя на диске в каталоге uploads. Здесь же ведутся служебные поля – количество скачиваний, время последнего доступа и отметка мягкого удаления, позволяющая сохранять историю аудита, даже после физической очистки. Подробная информация о file_record представлена в таблице 3.

Таблица 3 – Описание таблицы базы данных file_record

id	Идентификатор записи
token	Уникальный токен
file_name	Имя файла
secure_name	Зашифрованный файл, специального типа данных
upload_date	Дата загрузки
download_count	Количество скачиваний
user_id	Идентификатор пользователя

Таблица file_shares фиксирует передачи, независимые от файла объекты доступа с уникальным share_token. В них задаются параметры жизненного цикла

и состояние активна или отозвана ссылка. Для оперативной обработки запросов используются индексы по токену, файлу и времени создания, что ускоряет валидацию ссылок и выборку активных передач. Подробная информация о file_shares представлена на в таблице 4.

Таблица 4 – Описание таблицы базы данных file_shares

is_new	Флаг новой передачи, если получатель ещё не открывал входящий файл в строке 0, если запись просмотрена, то 1
id	Внутренний идентификатор записи
share_token	Уникальный токен передачи, с помощью него формируется ссылка на скачивание
file_token	Токен файла показывает ссылку на запись в таблице file_record
filename	Имя файла
sender_id	Идентификатор отправителя
max_downloads	Максимально допустимое количество скачиваний
downloads_done	Фактическое количество уже выполненных скачиваний
revoked	Флаг отзыва доступа, если 1 – ссылка отозвана, если 0, то ссылка действует

Таблица download_logs хранит полный аудит. Все действия, совершённые пользователями на сервере. Метку времени, инициатора, при условии, что он известен, ссылку на файл и передачу, тип действия, результат операции, IP-адрес, клиент пользователя и дополнительные детали. Журнал спроектирован так, чтобы не терять связность при удалении объектов, внешние ключи допускают обнуление ссылок, а сами записи остаются доступными для последующего анализа. Подробная информация о download_logs представлена на в таблице 5

Таблица 5 – Описание таблицы базы данных download_logs

id	Внутренний идентификатор записи
ts	Временная метка события
file_token	Токен файла ссылающийся на конкретный файл в таблице file_record
filename	Имя файла, с которым выполнялось действие
username	Имя пользователя, совершившего действие
ip	IP-адрес клиента на момент действия
user_agent	Информация о клиенте пользователя

Продолжение таблицы 5

Result	Результат операции, успешно или неудачно
Error	Описание ошибки если операция не удалась
Action	Тип выполненного действия

Связи между таблицами построены на внешних ключах с жёсткой семантикой удаления. Файл или передачу нельзя удалить, пока это затрагивает активные объекты. Журнал при этом не трогается, он должен оставаться полным независимо от любых операций. Индексы закрывают три основных сценария: проверку токенов и параметров доступа при скачивании, вывод списков файлов и передач конкретного пользователя, построение аудиторских отчётов с сортировкой по времени.

Каждый рабочий процесс обёрнут в транзакцию. При загрузке файл сохраняется на диск под безопасным именем, и в тот же момент в file_record появляется запись, а журнал фиксирует событие upload. При создании передачи, система генерирует токен, сохраняет параметры доступа и пишет в аудит share_create. Запрос на скачивание проходит последовательную проверку, сначала проверяется срок действия, лимит, статус отзыва. Если всё в порядке файл отдаётся, а счётчики в file_shares и file_record увеличиваются, в журнал пишется download с результатом и данными клиента. Неуспешная попытка тоже протоколируется. Отзыв передачи меняет её статус и оставляет запись в журнале. Удаление файла работает просто. Запись помечается флагом, но из базы не исчезает, аудит остаётся нетронутым. Для наглядности была сформирована диаграмма. ER-диаграмма – это графический способ описать структуру данных ещё до того, как написана первая строка кода. На диаграмме показывают сущности – объекты предметной области, которые нужно хранить, их атрибуты – свойства этих объектов, и связи между ними – то, как сущности соотносятся друг с другом. Такой подход позволяет проектировщику увидеть всю модель данных целиком, обнаружить избыточность и логические противоречия на раннем этапе, до реализации. В контексте реляционных баз данных ER-диаграмма служит прямым прообразом схемы таблиц: каждая сущность превращается в таблицу, атрибуты в столбцы, а связи определяют наличие

внешних ключей и промежуточных таблиц. На рисунке 4 показана ER-диаграмма, показывающая ключевые сущности разрабатываемой системы.

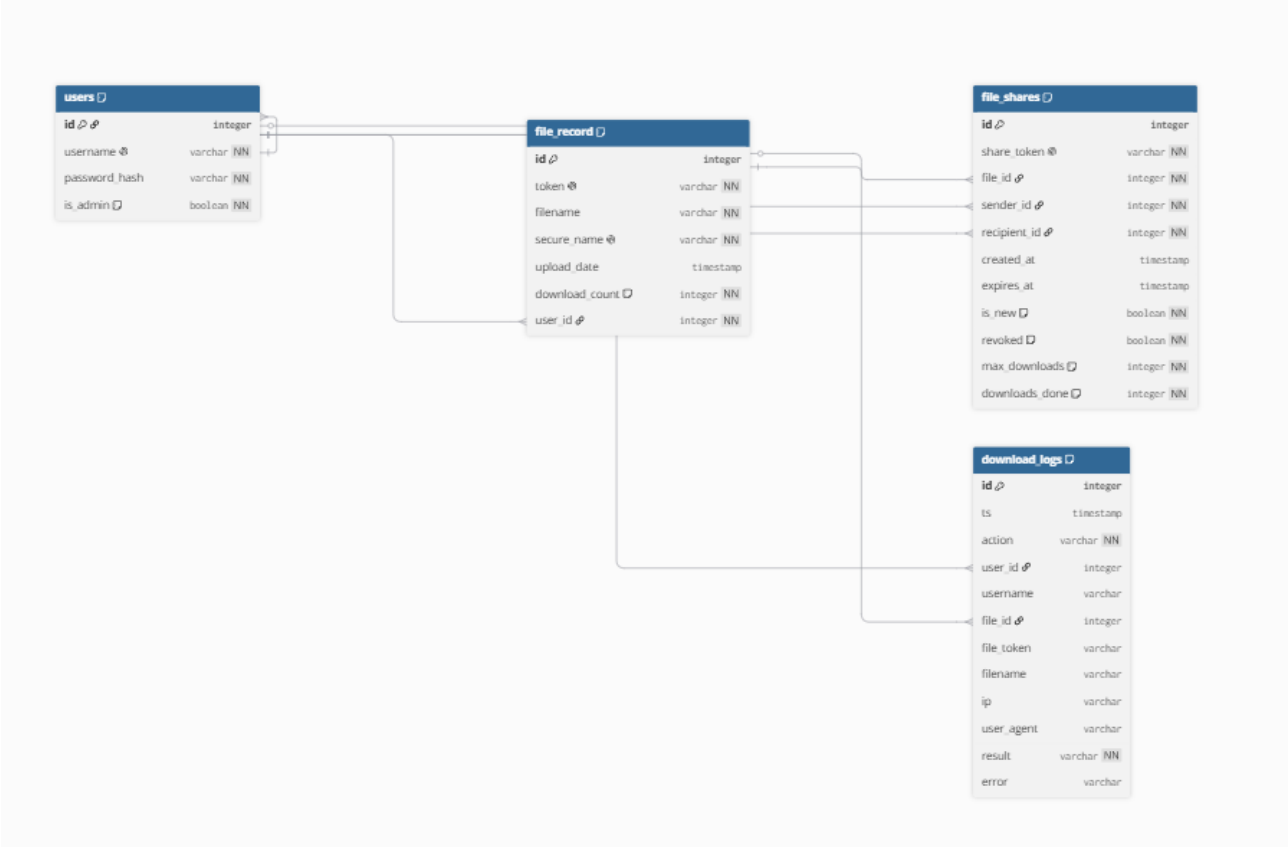


Рисунок 4 – ER-диаграмма ключевых сущностей

Данная диаграмма демонстрирует взаимодействие сущностей разрабатываемой системы.

2.3 Проектирование интерфейса

Интерфейс неотъемлемая часть современного программного обеспечения. В рамках выпускной квалификационной работы необходимо спроектировать интерфейс, обеспечивающий удобное взаимодействие между пользователем и системой. При проектировании, упор должен быть сделан на простоту использования и функциональность. У пользователя не должно возникать вопросов касательно назначения тех или иных вкладок, кнопок и других элементов разрабатываемой системы. Основная аудитория – сотрудники

небольшого предприятия, не обладающие навыками работы со сложными программными продуктами, и не обязательно имеющие технические знания.

Должны быть реализованы:

– окно авторизации;
– разделение ролей на администратора и пользователя;
– форма загрузки и отправки файлов;
– вкладка входящих сообщений от других пользователей;
– вкладка отправленных файлов текущим пользователем;
– специальная вкладка управления системой для администратора.

В ходе проектирования, был получен интерфейс, представляющий собой несколько HTML страниц, связанных между собой. Навигация построена понятным образом, пользователь всегда понимает, на каком этапе находится. Формы снабжены понятными подсказками и сообщениями об ошибках. Все критические действия, такие как отзыв передачи или удаление файла, требуют явного подтверждения, что исключает случайные операции.

Интерфейс поддерживает две роли: администратора и обычного пользователя. Окно авторизации позволяет заходить под разными ролями в систему, окно авторизации представлено на рисунке 5.

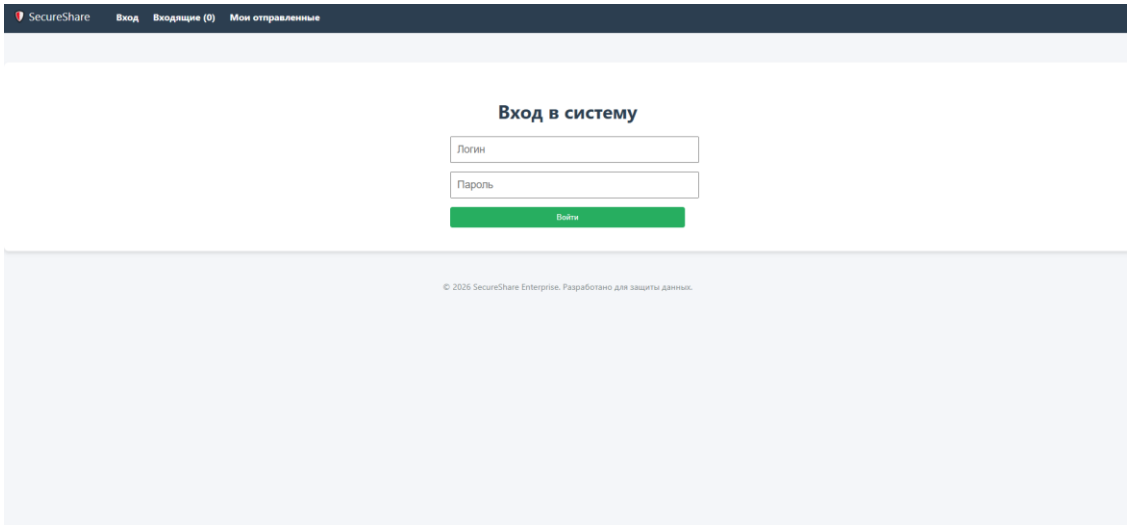


Рисунок 5 – Окно авторизации

По функционалу роли имеют некоторые различия. У администратора есть отдельная вкладка «Администрирование», где осуществляется работа с сервером. Пользователь не имеет такой вкладки. Разница в интерфейсе по ролям показана на рисунках 6 – 7.

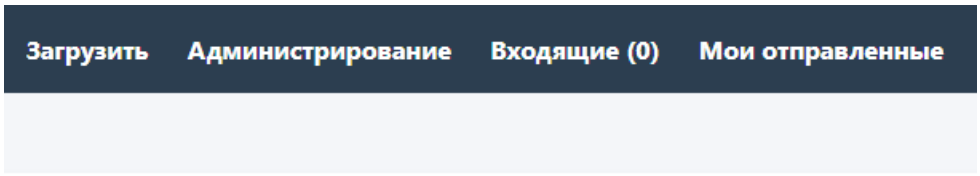


Рисунок 6 – Интерфейс администратора

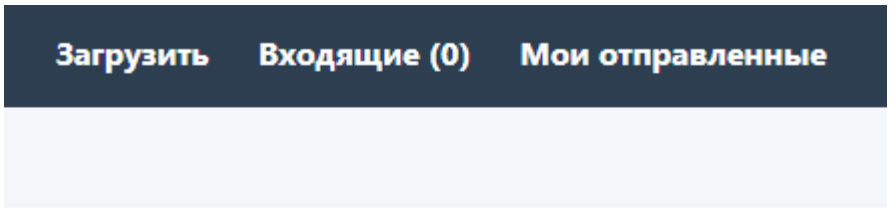


Рисунок 7 – Интерфейс пользователя

Главная страница спроектированного интерфейса выглядит понятно для рядового пользователя и не вызовет каких-либо вопросов. Страница намеренно лишена избыточных элементов: пользователь видит только то, что необходимо для выполнения конкретного действия. Зона загрузки файла расположена в центральной части экрана и визуально выделена, что исключает необходимость дополнительных инструкций. После выбора файла пользователю предлагается указать параметры передачи, срок действия ссылки и допустимое число скачиваний, после чего система генерирует уникальную ссылку, готовую к отправке адресату.

Такой подход к проектированию интерфейса соответствует принципу минимальной достаточности: каждый элемент присутствует по конкретной

причине и выполняет единственную функцию. Это снижает вероятность ошибки со стороны пользователя и сокращает время на освоение системы до минимума.

На рисунке 8 представлена основная форма интерфейса, где осуществляется загрузка и отправка файлов.

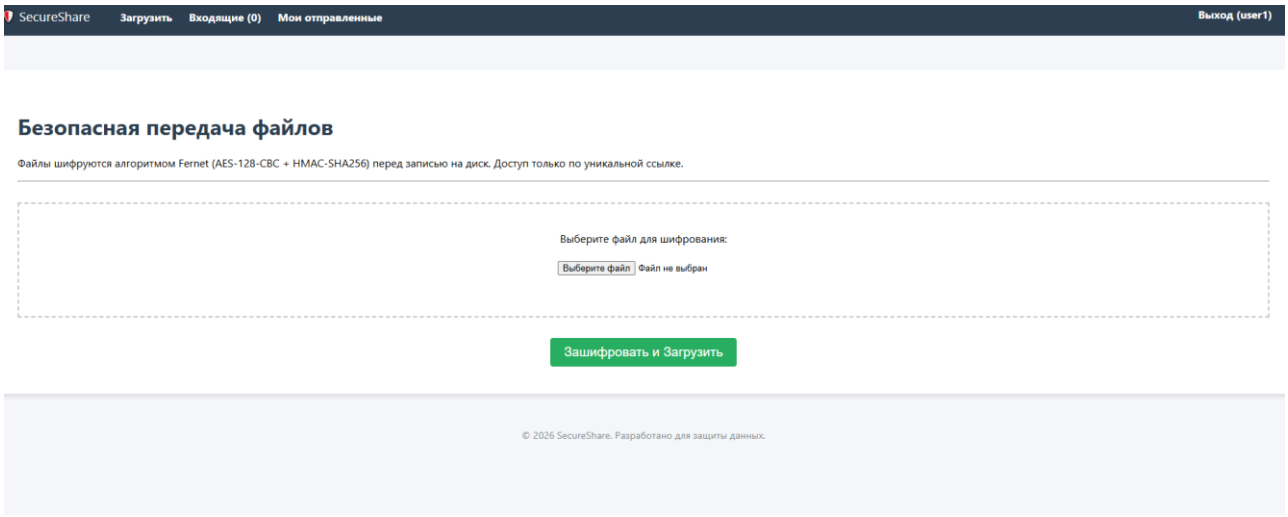


Рисунок 8 – Основная форма интерфейса

Для принятия входящих файлов была спроектирована вкладка «Входящие», где находятся все поступившие файлы от других пользователей. Данный элемент интерфейса показан на рисунке 9.

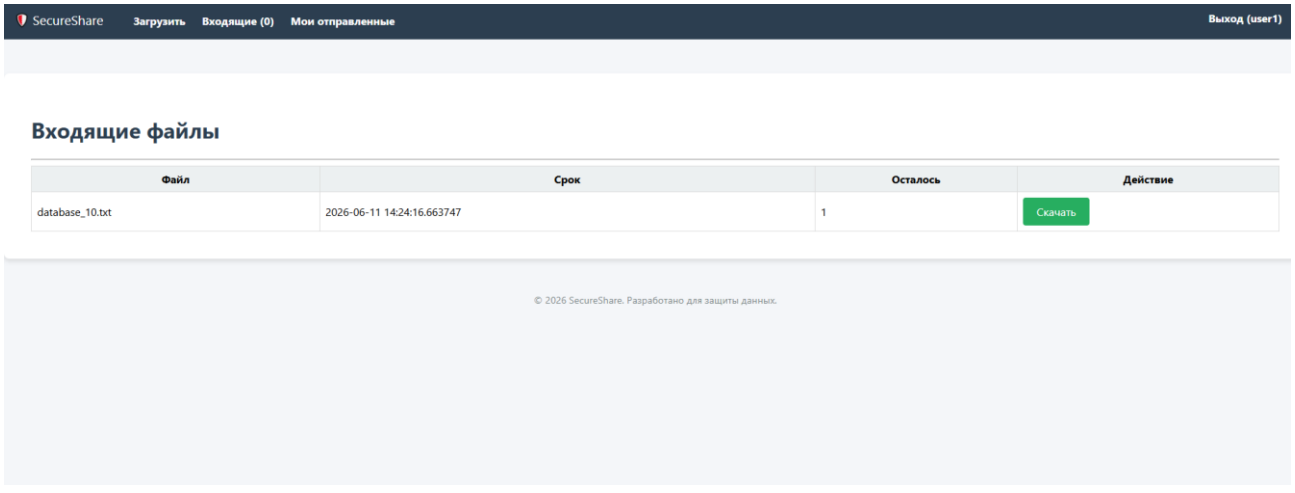


Рисунок 9 – Вкладка «Входящие»

Администратор может управлять системой и контролировать её работу. Панель администрирования даёт полную картину происходящего в веб-сервисе,

позволяя избегать различных ошибок. На рисунках 10 – 11 представлена вкладка управления.

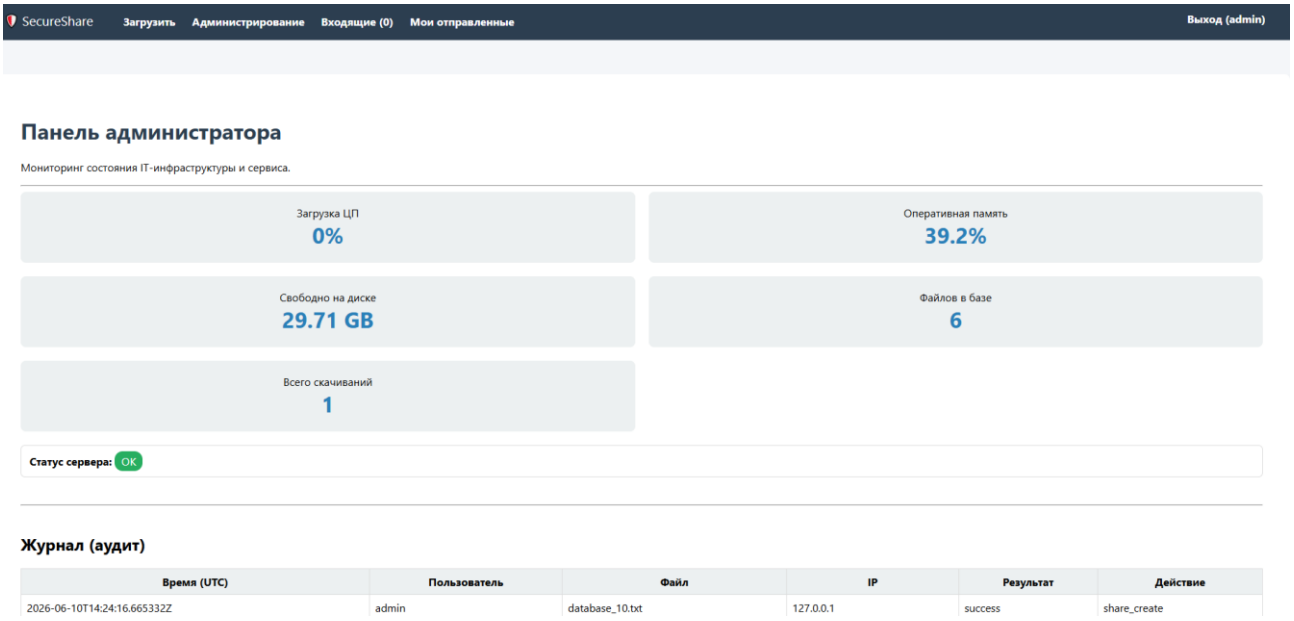


Рисунок 10 – Вкладка «Администрирование» для управления системой

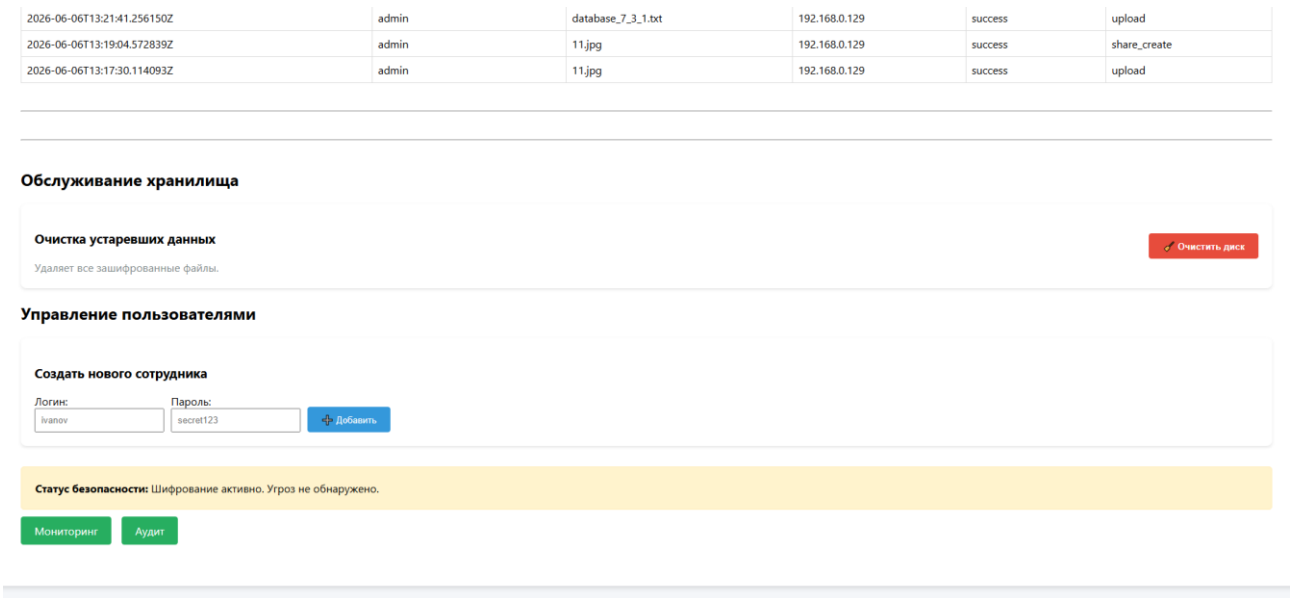


Рисунок 11 – Вкладка «Администрирование» для управления системой

Каждый пользователь может просмотреть список всех отправленных файлов, и при необходимости отозвать доступ. Такой подход, даёт возможность каждому пользователю управлять своими передачами файлов, что позволяет усилить безопасность и избежать утечки данных, любой сотрудник знает кому и

какой файл он отправил и в любой момент может отменить передачу. На рисунке 12 показана вкладка просмотра отправленных файлов.

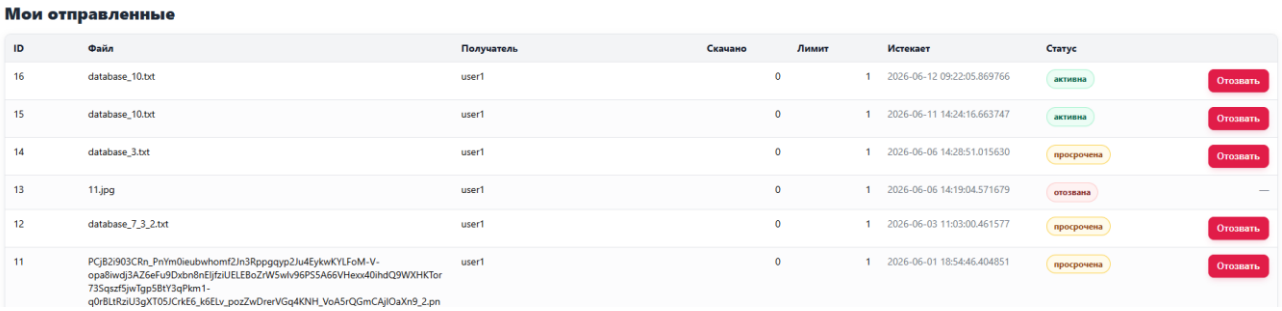


Рисунок 12 – Вкладка просмотра отправленных файлов другим пользователям

Результатом этапа проектирования интерфейса является полностью спроектированный интерфейс разработанной системы. Подробное описание работы интерфейса и все его функции описаны в программной реализации клиентской части системы.

3 Программная реализация

3.1 Программная реализация клиентской части системы

Клиентская часть реализована как набор HTML-шаблонов Jinja2, отдаваемых микро-фреймворком Flask в связке с WSGI-сервером. Вёрстка выполнена на HTML и CSS без сторонних библиотек для интерфейса. Для динамического поведения используются короткие сценарии на языке JavaScript. Доступ к функционалу управляется ролями, обычный пользователь и администратор, что отражается в навигации и доступных действиях на страницах.

Клиентская часть состоит из следующих шаблонов:

- базового шаблона base.html;
- формы входа login.html;
- страницы загрузки файла index.html;
- страницы подтверждения success.html;
- раздела входящих передач inbox.html;
- раздела исходящих передач shares_outbox.html;
- административной панели admin.html;
- расширенного журнала аудита admin_audit.html.

Шаблон base.html формирует единую навигационную панель, которая подключается ко всем остальным страницам. Меню формируется динамически в зависимости от роли текущего пользователя. Обычный пользователь видит пункты «Загрузить», «Входящие» и «Мои отправленные», тогда как администратор дополнительно получает пункт «Администрирование». Счётчик непрочитанных входящих передач подставляется сервером автоматически используя инструменты Flask и отображается рядом с соответствующим пунктом меню. Не прошедшему аутентификацию пользователю, доступна только ссылка на страницу входа. Также шаблон обеспечивает вывод сообщений

об успехе или ошибке операций и содержит общий «подвал» страницы. Краткий программный код данного компонента приведен в листинге 1.

Листинг 1 – Программный код навигационной панели

```
<div class="navbar">
  <span style="font-size: 1.2em; margin-right: 40px;">
SecureShare</span>
  {% if current_user.is_authenticated %}
      <a href="{{ url_for('main.index') }}">Загрузить</a>
      {% if current_user.is_admin %}
          <a href="{{ url_for('main.admin_dashboard')
}}">Администрирование</a>
      {% endif %}
      <a href="{{ url_for('main.logout') }}" style="float:
right;">Выход ({{ current_user.username }})</a>
  {% else %}
      <a href="{{ url_for('main.login') }}">Вход</a>
  {% endif %}
  <a href="{{ url_for('main.inbox') }}">Входящие ({{ inbox_count
}})</a>
  <a href="{{ url_for('main.shares_outbox') }}">Мои
отправленные</a>
</div>
```

Страница login.html содержит форму входа с двумя полями: логин и пароль. Форма отправляется методом POST. Браузер выполняет базовую валидацию заполненности полей. При успешной аутентификации сервер устанавливает сессию и перенаправляет пользователя на главную страницу загрузки. При неверных учётных данных на странице отображается соответствующее сообщение об ошибке. Краткий программный код данного компонента приведен в листинге 2.

Листинг 2 – Программный код формы входа

```
<form action="" method="post">
  <div style="margin-bottom: 15px;">
    <input type="text" name="username" placeholder="Логин"
required style="padding: 10px; width: 100%;">
  </div>
  <div style="margin-bottom: 15px;">
    <input type="password" name="password" placeholder="Пароль"
required style="padding: 10px; width: 100%;">
  </div>
  <button type="submit" class="btn" style="width:
100%;">Войти</button>
</form>
```

Страница index.html предоставляет форму для выбора и отправки файла. Файл упаковывается в специальный формат и отправляется на веб-сервер через HTTP-запрос. На стороне клиента выполняется только проверка браузером факта выбора файла перед отправкой. Непосредственно шифрование выполняется на самом сервере. После получения файла он шифруется и сохраняется в хранилище, а пользователю возвращается уникальный токен. После успешной загрузки браузер автоматически перенаправляется на страницу подтверждения. Краткий программный код данного компонента приведен в листинге 3.

Листинг 3 – Программный код формы для выбора и отправки файла

```
<form action="" method="post">
  <div style="margin-bottom: 15px;">
    <input type="text" name="username" placeholder="Логин"
required style="padding: 10px; width: 100%;">
  </div>
  <div style="margin-bottom: 15px;">
    <input type="password" name="password" placeholder="Пароль"
required style="padding: 10px; width: 100%;">
  </div>
  <button type="submit" class="btn" style="width:
100%;">Войти</button>
</form>
```

Страница success.html отображается после успешной загрузки файла и выполняет две самостоятельные функции, логично объединённые в рамках одного экрана. В верхней части страницы пользователю отображается сгенерированная сервером защищённая временная ссылка для скачивания файла.

Эта ссылка уникальна для каждой загрузки и может быть передана получателю любым удобным способом, например, скопирована и отправлена вручную по внутренним каналам связи организации.

Ниже, на этой же странице, расположена форма адресной внутренней передачи, которая предоставляет более удобный и контролируемый способ доставки файла. С её помощью пользователь выбирает получателя из списка сотрудников организации, устанавливает срок действия передачи от одной минуты до бессрочного доступа и задаёт допустимый лимит скачиваний: один

раз, три раза или без ограничений. Такой набор параметров позволяет гибко управлять условиями доступа к файлу в зависимости от конкретной ситуации.

После отправки формы файл незамедлительно появляется в разделе «Входящие» у выбранного получателя, не требуя от него каких-либо дополнительных действий для получения доступа. На рисунке 13 показана форма отправки файлов.

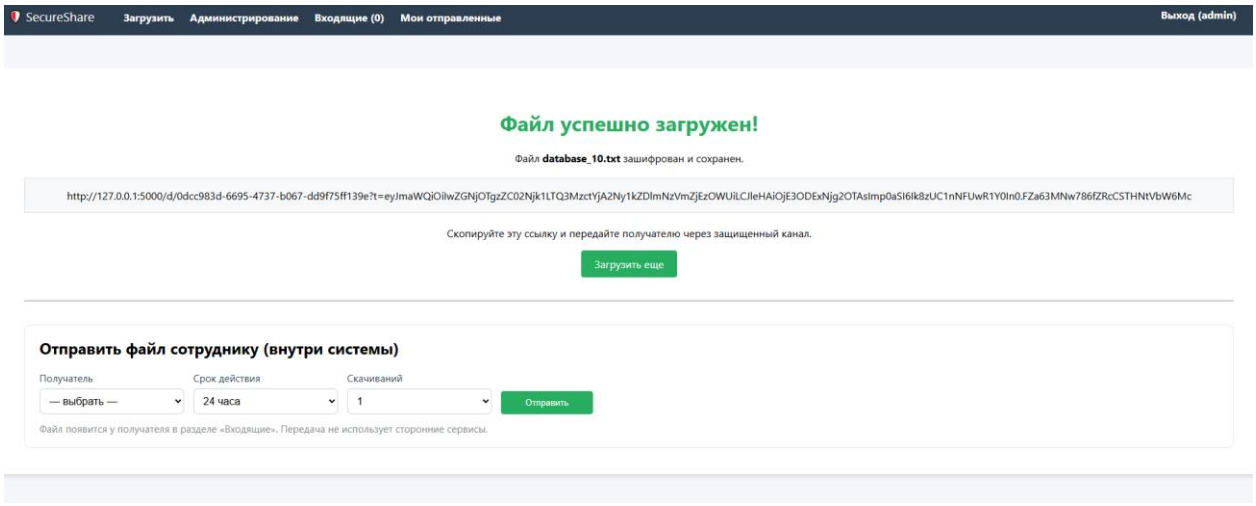


Рисунок 13 – Страница подтверждения загрузки с формой передачи

Чтобы просмотреть присланные файлы была создана страница inbox.html, которая отображает список активных передач, адресованных текущему пользователю. Для каждой передачи в таблице показывается имя файла, срок действия и количество оставшихся скачиваний. Кнопка «Скачать» ведёт на защищённый маршрут, при обращении к которому сервер проверяет аутентификацию пользователя и соответствие получателя, после чего выдаёт расшифрованный файл. Если срок действия или лимит скачиваний исчерпан, передача в списке не отображается. Страница входящих была показана на рисунке 9.

Страница shares_outbox.html содержит список всех созданных пользователем передач. Для каждой передачи отображается имя файла, получатель, дата создания, актуальный статус и число выполненных скачиваний. Статус передачи определяется автоматически и может принимать три значения: «активна» – передача действует и доступна получателю, «просрочена» – истёк

срок действия, «отозвана» – пользователь вручную аннулировал доступ. Для активных передач доступна кнопка «Отозвать», при её нажатии браузер запрашивает подтверждение действия, после чего передача немедленно блокируется и исчезает из раздела «Входящие» у получателя. На рисунке 12 показа вкладка просмотра отправленных сообщений.

Страница admin.html доступна исключительно пользователям с ролью администратора. Панель разделена на два блока. Первый блок отображает метрики сервера в режиме реального времени: загрузка процессора, использование оперативной памяти, свободное место в хранилище, общее количество файлов, скачиваний и зарегистрированных пользователей. Данные обновляются автоматически каждые две секунды посредством асинхронных запросов к внутреннему JSON-API сервера. Дополнительно на панели отображается индикатор «Статус сервера», который меняет цвет в зависимости от уровня нагрузки: зелёный соответствует норме, жёлтый сигнализирует о предупреждении, красный – о критическом состоянии. Второй блок содержит таблицу последних событий аудита, которая также обновляется автоматически каждые две секунды без перезагрузки страницы. Вместе с кодом в формате HTML встроен код на языке JavaScript. Фрагмент кода обновления состояния сервера, встроенного в файл admin.html представлен в листинге 4. Сама панель администратора была показана на рисунках 10 – 11.

Листинг 4 – Фрагмент кода на языке JavaScript

```
async function refreshStats() {
  try {
    const resp = await fetch("{{ url_for('main.api_admin_stats')
}}", { cache: "no-store" });
    if (!resp.ok) return;

    const data = await resp.json();

    document.getElementById("cpuVal").textContent = data.cpu;
    document.getElementById("ramVal").textContent =
data.ram_percent;
    document.getElementById("diskVal").textContent =
data.disk_free_gb;
```

```
    document.getElementById("filesVal").textContent =
data.files_count;
    document.getElementById("downloadsVal").textContent =
data.downloads_count;
    const u = document.getElementById("usersVal");
    if (u) u.textContent = data.users_count;

    const badge = document.getElementById("healthBadge");
    const issues = document.getElementById("healthIssues");
    issues.textContent = (data.issues && data.issues.length) ?
("(" + data.issues.join(", ") + ")") : "";

    if (data.health === "OK") {
      badge.textContent = "OK";
      badge.style.background = "#27ae60";
    } else if (data.health === "WARN") {
      badge.textContent = "WARN";
      badge.style.background = "#f39c12";
    } else {
      badge.textContent = "CRIT";
      badge.style.background = "#e74c3c";
    }
  } catch (e) {
  }
}
refreshStats();
setInterval(refreshStats, 2000); // 2 секунды
```

Страница admin_audit.html предоставляет полный журнал всех событий системы в табличном виде и является одним из ключевых инструментов в распоряжении администратора. В отличие от краткого аудита, доступного непосредственно на административной панели, данная страница не обновляется автоматически и ориентирована на вдумчивую работу с накопленными данными. Она предназначена прежде всего для детального анализа истории событий и расследования возможных инцидентов безопасности, когда требуется восстановить хронологию действий или установить источник нежелательной активности.

Каждая запись в таблице содержит исчерпывающий набор сведений: дату и точное время события, имя пользователя, совершившего действие, имя задействованного файла, IP-адрес, с которого был выполнен запрос, результат операции – успешно или с ошибкой, а также тип выполненного действия. Совокупность этих параметров позволяет администратору получить полную

картину произошедшего без необходимости обращаться к сторонним источникам информации. На рисунке 14 показана основная панель мониторинга, представляющая полный обзор всех действий, происходящих на сервере.

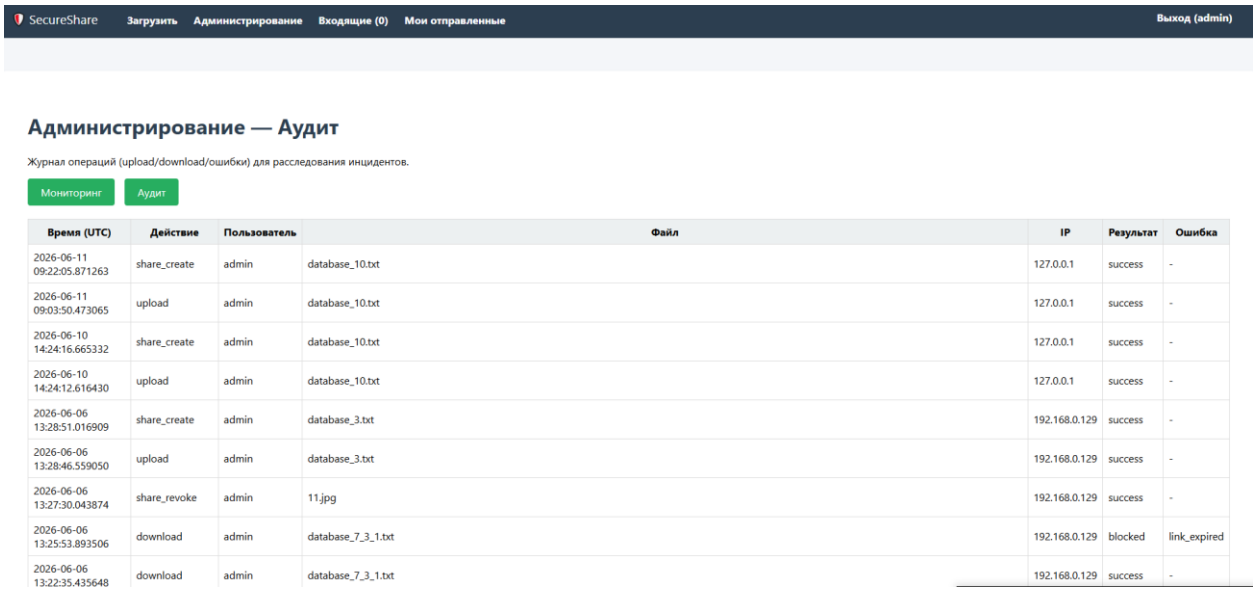


Рисунок 14 – Расширенный журнал аудита

Клиентская часть системы построена на серверных шаблонах Jinja2 с минимальным использованием JavaScript только для динамического обновления данных на административной панели. Реализованы все ключевые пользовательские сценарии:

– загрузка и хранение зашифрованных файлов с формированием защищённой ссылки;

– адресная передача файлов внутри организации с разделами «Входящие» и «Мои отправленные» и возможностью отзыва доступа;

– административный мониторинг серверных ресурсов и аудит действий пользователей в режиме реального времени.

Реализованное решение соответствует требованиям к защищённой передаче данных во внутренней сети предприятия.

3.2 Программная реализация серверной части

Архитектура серверной части разработана на языке программирования Python с применением расширения Flask. Для обеспечения гибкости

конфигурации и изоляции компонентов системы внедрён паттерн проектирования «Фабрика приложения». Взаимодействие между веб-сервером и программным кодом осуществляется на базе стандартного интерфейса WSGI. В качестве системы управления базами данных задействована SQLite, взаимодействие с которой реализовано посредством объектно-реляционного отображения SQLAlchemy. Управление сессиями пользователей, механизмы авторизации и аутентификации построены на базе расширения Flask-Login. Безопасность данных обеспечивается криптографическим модулем cryptography, реализующим симметричный алгоритм шифрования Fernet на стороне сервера. Сбор телеметрии и мониторинг аппаратных ресурсов сервера реализованы с помощью библиотеки psutil.

Серверная часть состоит из следующих модулей:

– модуля инициализации;
– модуля работы с данными;
– модуля маршрутизации;
– модуля шифрования;
– модуля конфигурации;
– модулей запуска системы.

Модуль инициализации является основой программного пакета и реализует паттерн «Фабрика приложения». В рамках данного модуля осуществляется подключение к базе данных, конфигурирование подсистемы аутентификации и службы логирования событий. На уровне данного модуля задаются политики безопасности заголовков HTTP-ответов и инициализируется параллельный фоновый поток, выполняющий плановую очистку хранилища от устаревших данных. Ключевой особенностью подсистемы логирования является внедрение криптографического фильтра AuditChainFilter, который формирует непрерывную цепочку хеш-записей для событий аудита, связывая каждое новое событие с предыдущим. Принцип работы фильтра заключается в следующем: при поступлении каждой новой записи аудита вычисляется хеш SHA-256 на

основе содержимого текущего сообщения и хеша предыдущей записи, после чего полученное значение сохраняется как якорь для следующей записи. Это гарантирует защиту журналов регистрации от несанкционированного изменения и ретроспективной фальсификации. Программный код криптографического фильтра приведён в листинге 5. Полный листинг кода приведён в приложении Б.

Листинг 5 – Криптографический фильтр

```
class AuditChainFilter(logging.Filter):
        def __init__(self, anchor_path: Path):
            super().__init__()
            self.anchor_path = anchor_path
            try:
                self.prev =
self.anchor_path.read_text(encoding="utf-8").strip()
            except Exception:
                self.prev = ""
        def filter(self, record: logging.LogRecord) -> bool:
            msg = str(record.getMessage())
            if "AUDIT " in msg:
                base = (self.prev or "").encode() + msg.encode()
                h = hashlib.sha256(base).hexdigest()
                record.msg = f"{msg} h={h}"
                self.prev = h
                try:
                    self.anchor_path.write_text(self.prev,
encoding="utf-8")
                except Exception:
                    pass
            return True
```

Модуль работы с данными описывает схемы и структуры таблиц реляционной базы данных. Содержит описание четырёх базовых сущностей. Сущность User хранит учётные записи пользователей, включая имя пользователя, хеш пароля и флаг роли администратора. Пароли никогда не сохраняются в открытом виде – при создании учётной записи формируется безопасный хеш, а при входе выполняется его проверка. Сущность FileRecord содержит метаданные загруженных файлов, а именно оригинальное имя, путь к зашифрованному файлу на диске и уникальный токен доступа. Сущность FileShare фиксирует записи о предоставлении адресного доступа к файлам с указанием получателя, срока действия, лимита скачиваний и статуса отзыва. Сущность DownloadLog ведёт журнал фактов скачивания с фиксацией времени

и IP-адреса. Программный код создания сущности приведён в листинге 6. Полный листинг создания сущностей находится в приложении Б.

Листинг 6 – Пример создании сущности пользователя

```
class User(UserMixin, db.Model):
    """
    Модель пользователя системы.
    Наследуется от UserMixin, чтобы Flask-Login умел с ней
работать.
    """
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    is_admin = db.Column(db.Boolean, default=False)  # Флаг: админ
или нет

    def set_password(self, password):
        """Создает безопасный хеш пароля"""
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        """Проверяет пароль по хешу"""
        return check_password_hash(self.password_hash, password)

    def __repr__(self):
        return f'<User {self.username}>'
```

Модуль маршрутизации реализует всю бизнес-логику веб-приложения и обеспечивает обработку входящих HTTP-запросов. В рамках модуля реализованы следующие подсистемы:

– авторизация пользователей с проверкой учётных данных и управлением сессией;

– приём файлов от пользователя с последующей передачей в модуль шифрования и сохранением метаданных в базе данных;

– выдача файлов по защищённому маршруту с проверкой аутентификации, соответствия получателя, срока действия и лимита скачиваний;

– управление адресными передачами с возможностью создания и отзыва;

– интерфейс администратора, предоставляющий метрики системы и системные журналы через внутренний JSON-API.

Листинг кода модуля маршрутизации приведён в приложении Б.

Для безопасности предусмотрен модуль шифрования, он является подсистемой обеспечения информационной безопасности и предоставляет программный интерфейс для выполнения обратимых криптографических преобразований файлов. Шифрование выполняется алгоритмом Fernet перед записью файла на постоянное запоминающее устройство. Расшифрование производится при санкционированном запросе пользователя непосредственно перед выдачей файла, без сохранения открытых данных на диске. Модуль поддерживает ротацию ключей, при наличии нескольких ключей шифрование всегда выполняется актуальным ключом, тогда как расшифрование последовательно перебирает все доступные ключи, что обеспечивает совместимость с ранее зашифрованными файлами в процессе смены ключей. Ключи передаются через переменные окружения, исключая их хранение в исходном коде. Листинг программного кода модуля шифрования приведён в приложении Б.

Модуль конфигурации содержит параметры окружения, секретные ключи для подписи пользовательских сессий, ограничения на максимальный размер загружаемых файлов, а также глобальные настройки безопасности файлов, флаги HttpOnly и SameSite, препятствующие межсайтовым атакам. Ключевой особенностью модуля является реализация кроссплатформенного механизма динамического определения путей к директориям хранения данных и базы данных SQLite. Алгоритм автоматически адаптирует пути под целевую операционную среду: для семейства Windows используются системные переменные окружения LOCALAPPDATA и APPDATA, для систем семейства Linux применяется спецификация XDG Base Directory. Это обеспечивает переносимость и корректное развёртывание системы в различных инфраструктурах без изменения исходного кода. Листинг программного кода модуля конфигурации приведён в приложении Б.

Модули запуска предназначены для инициализации WSGI-сервера и запуска веб-приложения в целевой операционной системе. Реализовано три способа запуска. Первый предназначен для тестирования и отладки. приложение

запускается встроенным сервером Flask с выводом отладочной информации в консоль. На рисунке 15 показан первый способ запуска.

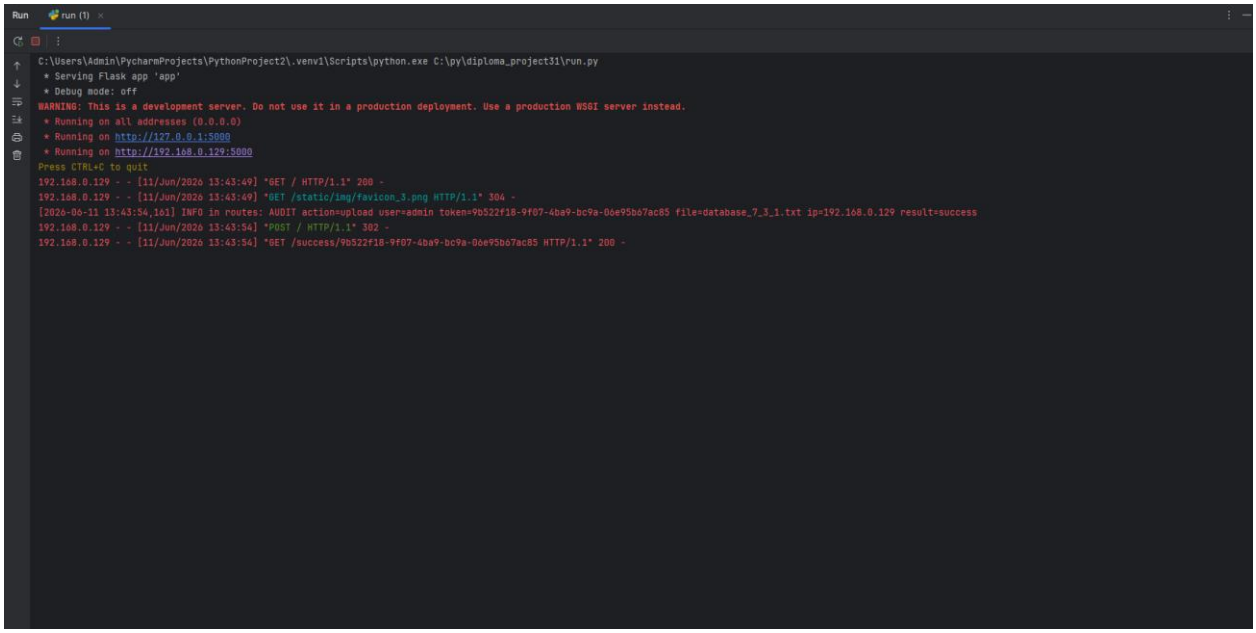


Рисунок 15 – Способ запуска для тестирования и отладки

Второй способ ориентирован на эксплуатацию без отладочного вывода. Приложение обслуживается сервером Waitress в многопоточном режиме, что обеспечивает стабильную работу при параллельных запросах. Такой способ запуска показан на рисунке 16.

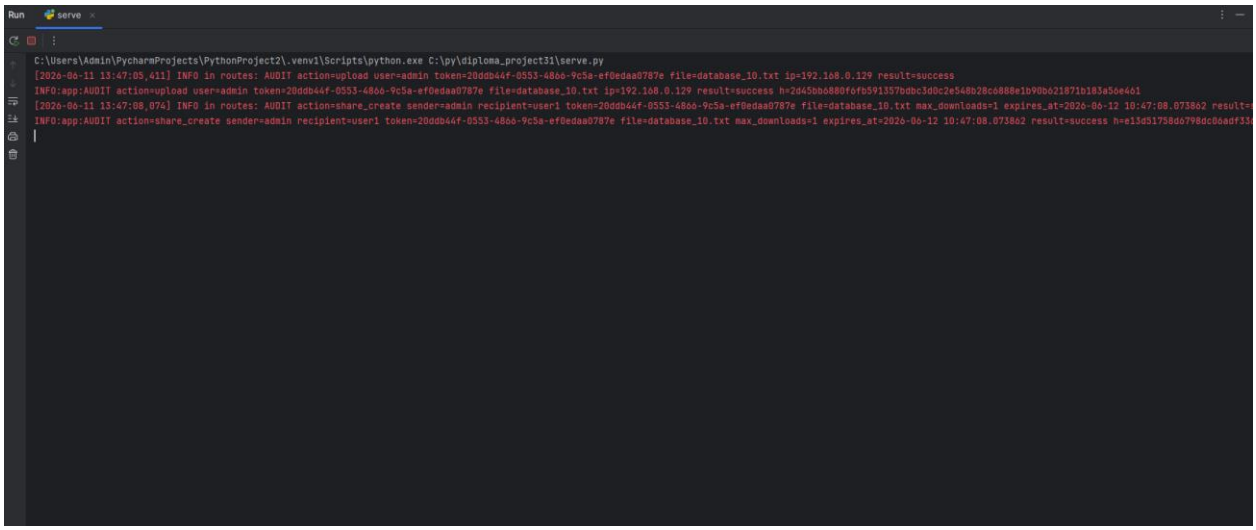


Рисунок 16 – Способ запуска без отладки

Третий способ представляет собой основной исполняемый модуль с графическим интерфейсом, позволяющий запускать и останавливать сервер

нажатием кнопки, а также выбирать режим работы – локальный или сетевой. На рисунке 17 показан способ запуска с графическим интерфейсом.

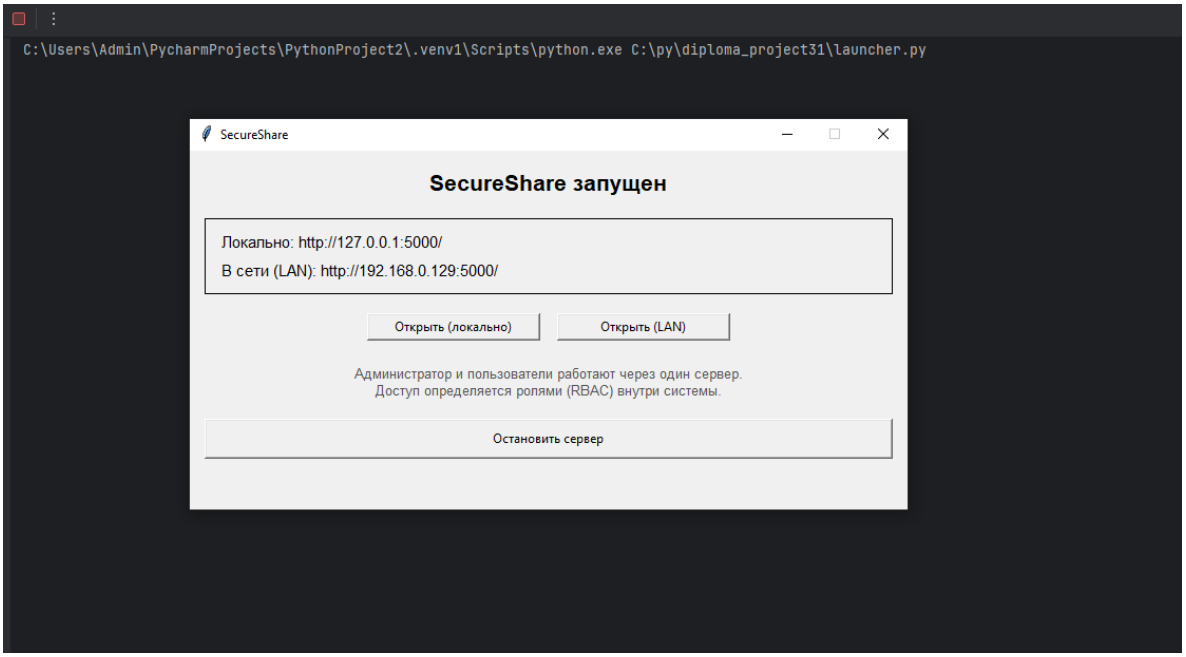


Рисунок 17 – Способ запуска с графическим интерфейсом

Листинг модулей запуска приведён в приложении Б.

Чтобы детально спроектировать физическую структуру системы, была разработана диаграмма развёртывания, которая показывает, из каких компонентов состоит система и как они связаны между собой.

Диаграмма развёртывания – это один из видов структурных диаграмм языка моделирования UML, предназначенный для описания физической архитектуры программной системы. Она отображает, каким образом программные компоненты и артефакты распределяются по аппаратным узлам, а также демонстрирует коммуникационные связи между этими узлами. Иными словами, диаграмма развёртывания отвечает на вопрос не о том, что делает система, а о том, где и на чём она работает.

В контексте данного проекта диаграмма развёртывания позволяет наглядно представить взаимодействие между клиентской частью приложения, серверной стороной и базой данных. На ней отображаются узлы системы, такие как рабочая станция пользователя, веб-сервер и сервер хранения данных, а также

протоколы и каналы, по которым осуществляется обмен информацией между ними. Подобное представление архитектуры упрощает понимание системы в целом, облегчает её последующее сопровождение и масштабирование, а также помогает выявить потенциальные узкие места ещё на этапе проектирования. Диаграмма развёртывания представлена на рисунке 18.

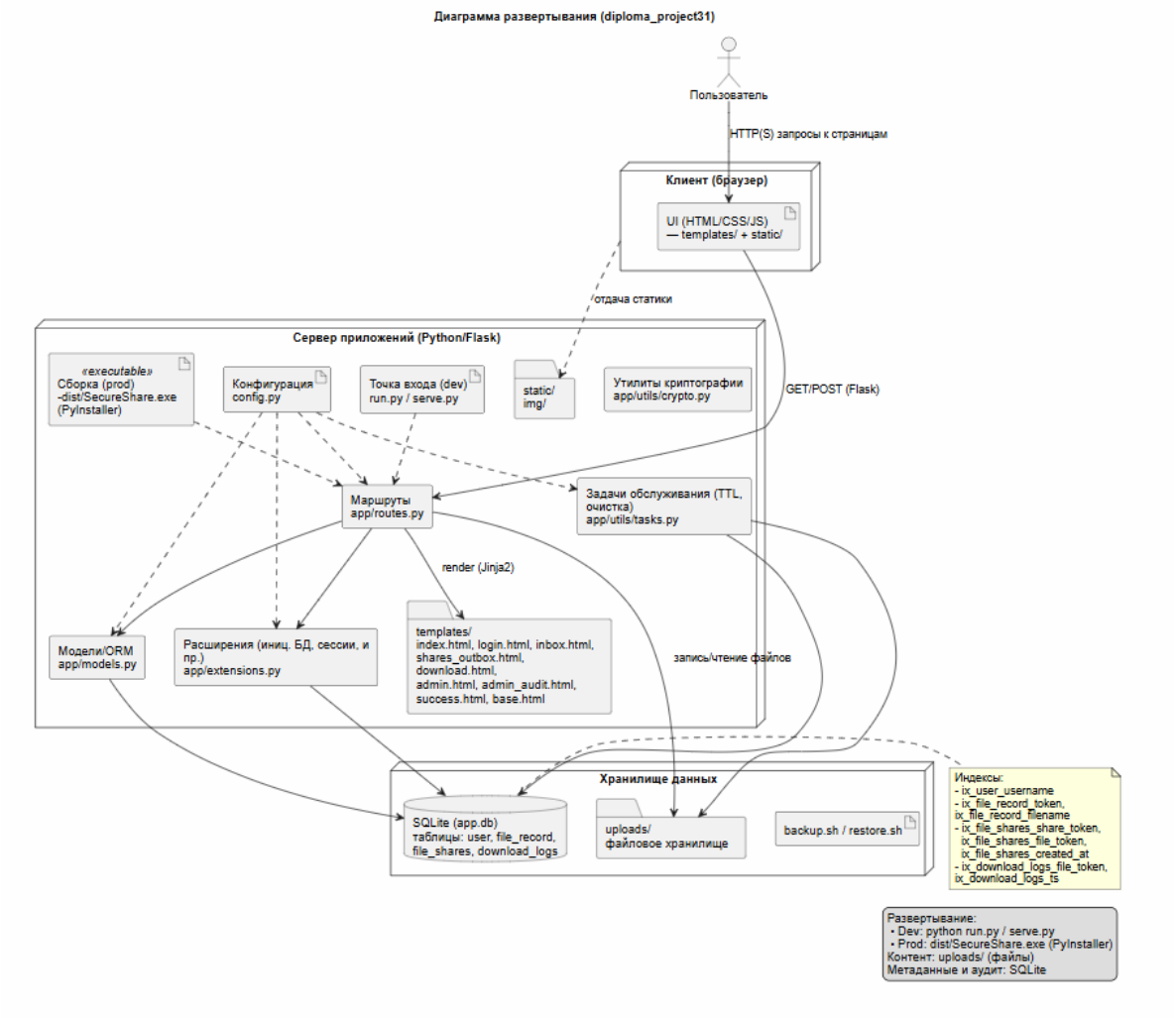


Рисунок 18 – Диаграмма развёртывания

Диаграмма развёртывания строится вокруг трёх узлов: браузер пользователя, сервер приложений и узел данных.

Пользователь работает через веб-интерфейс. Страницы формируются на сервере из шаблонов каталога templates в формате HTML – index, login, inbox, shares_outbox, download, admin, admin_audit, success. Статика отдаётся из каталога static. Любое действие пользователя, переход по странице, отправка

формы, загрузка файла, создание или отзыв передачи, всё это проходит через маршруты Flask в файле app/routes.py.

Сервер реализован на Python/Flask и запускается двумя способами. В среде разработки, через run.py или serve.py. В корпоративной среде упакованным исполняемым файлом SecureShare.exe из каталога dist, собранным с помощью PyInstaller. Конфигурация задаётся в config.py и подхватывается модулями инициализации, моделями и утилитами. Криптографика и токены живут в utils/crypto.py. Фоновые задачи такие как проверка сроков действия ссылок, обслуживание хранилища специально вынесены в каталог utils в tasks.py.

Все структурированные данные хранятся в локальной базе SQLite, в файле app.db. Между маршрутами и базой стоит слой моделей: он создаёт записи о файлах, фиксирует передачи с уникальным share_token, обновляет счётчики скачиваний и пишет аудит. Для ускорения выборок настроены индексы:

– для пользователей (ix_user_username);

– для файлов (ix_file_record_token, ix_file_record_filename);

– для передач (ix_file_shares_share_token, ix_file_shares_file_token, ix_file_shares_created_at);

– для журнала аудита (ix_download_logs_file_token, ix_download_logs_ts).

Это позволяет быстро искать по токенам, именам и временным меткам, что критично при скачивании и аудиторских проверках.

Потоки данных чётко разделены по ролям. При загрузке маршрут принимает файл, сохраняет его на диск под безопасным именем, записывает метаданные в таблицу file_record и фиксирует событие upload в журнале. При создании передачи сервер генерирует share_token, сохраняет TTL и лимит скачиваний в таблице file_shares, отмечает событие в аудите. При скачивании система последовательно проверяет отзыв, срок действия и лимит, и только потом отдаёт файл. Счётчики обновляются атомарно, IP-адрес клиента пишется

в журнал. Любая попытка скачать файл, в том числе отклонённая, оставляет след в аудите.

Сетевая граница намеренно узкая, HTTP только между браузером и сервером. База данных и каталог uploads наружу не публикуются. Это упрощает развёртывание и сокращает поверхность атаки. Для резервного копирования рядом с узлом данных лежат скрипты backup.sh и restore.sh они работают на GNU/Linux-системах и позволяют восстановить как базу SQLite, так и каталог uploads целиком.

По итогу диаграмма развёртывания описывает компактную одноузловую систему, в которой интерфейс, логика, метаданные и файлы выстроены в единую цепочку начиная от аутентификации, заканчивая полным аудитом каждого действия.

Серверная часть разрабатываемой системы, реализована на языке Python с применением микро-фреймворка Flask и построена по модульному принципу. Паттерн «Фабрика приложения» обеспечивает гибкость конфигурации и изоляцию компонентов. Шифрование файлов алгоритмом Fernet с поддержкой ротации ключей гарантирует конфиденциальность данных на диске. Цепочечное хеширование записей аудита защищает журналы от фальсификации. Кроссплатформенный модуль конфигурации обеспечивает корректное развёртывание системы как на Windows, так и на Linux. В совокупности реализованные модули обеспечивают надёжную, безопасную и переносимую серверную основу для защищённой передачи файлов.

3.3 Программная реализация базы данных

В проектируемой системе используется база данных SQLite. Это лёгкое и надёжное решение, которое не требует отдельного сервера и хорошо подходит для сервиса, где в базе хранятся служебные сведения, такие как пользователи, метаданные о загруженных файлах, записи об адресных передачах, а также журнал операций для аудита. Сами файлы в базе данных не помещаются – они сохраняются на диск в зашифрованном виде. В базе остаются только метаданные

и токены доступа. Благодаря этому система остаётся простой, а хранение предсказуемое и управляемое. Доступ к данным реализован через ORM, это делает код чище. При необходимости можно перейти на другую СУБД без серьёзной переделки логики, например PostgreSQL. Подход удобен при развёртывании прозрачен для администрирования и хорошо работает как в разработке, так и в эксплуатации при умеренной нагрузке. Если со временем требования вырастут, выбранная архитектура позволит масштабироваться без потери накопленных данных и без серьёзного пересмотра модели.

Сущность пользователей является одной из центральных в структуре базы данных и предназначена для регистрации и хранения учётных записей всех субъектов системы, взаимодействующих с ней в той или иной роли.

В структуре данной сущности фиксируется исчерпывающий набор атрибутов, необходимых для идентификации и управления учётной записью. В частности, хранится уникальный идентификатор пользователя в виде логина, криптографический хеш пароля, исключающий возможность его восстановления в открытом виде даже при несанкционированном доступе к базе данных, метка административных привилегий, определяющая уровень прав конкретного субъекта в системе, а также временная метка создания профиля, позволяющая отслеживать историю регистраций.

Записи данной таблицы активно задействуются ключевыми механизмами системы – аутентификацией при входе, контролем доступа к защищённым ресурсам и разграничением прав между рядовыми пользователями и администраторами. Таким образом, сущность пользователей выступает фундаментом, на котором строится вся политика безопасности приложения.

Посредством внешних ключей идентификатор пользователя связывается с объектами других подсистем, что позволяет однозначно определить:

– владельца загруженного файла, то есть пользователя, осуществившего его передачу в систему;

– отправителя и адресата при реализации адресного доступа к ресурсам, когда файл направляется конкретному сотруднику организации;

– инициатора операций, фиксируемых в системном журнале событий, что обеспечивает полную прослеживаемость действий каждого субъекта.

На рисунке 19 представлена таблица внутри базы данных с тестовыми данными.

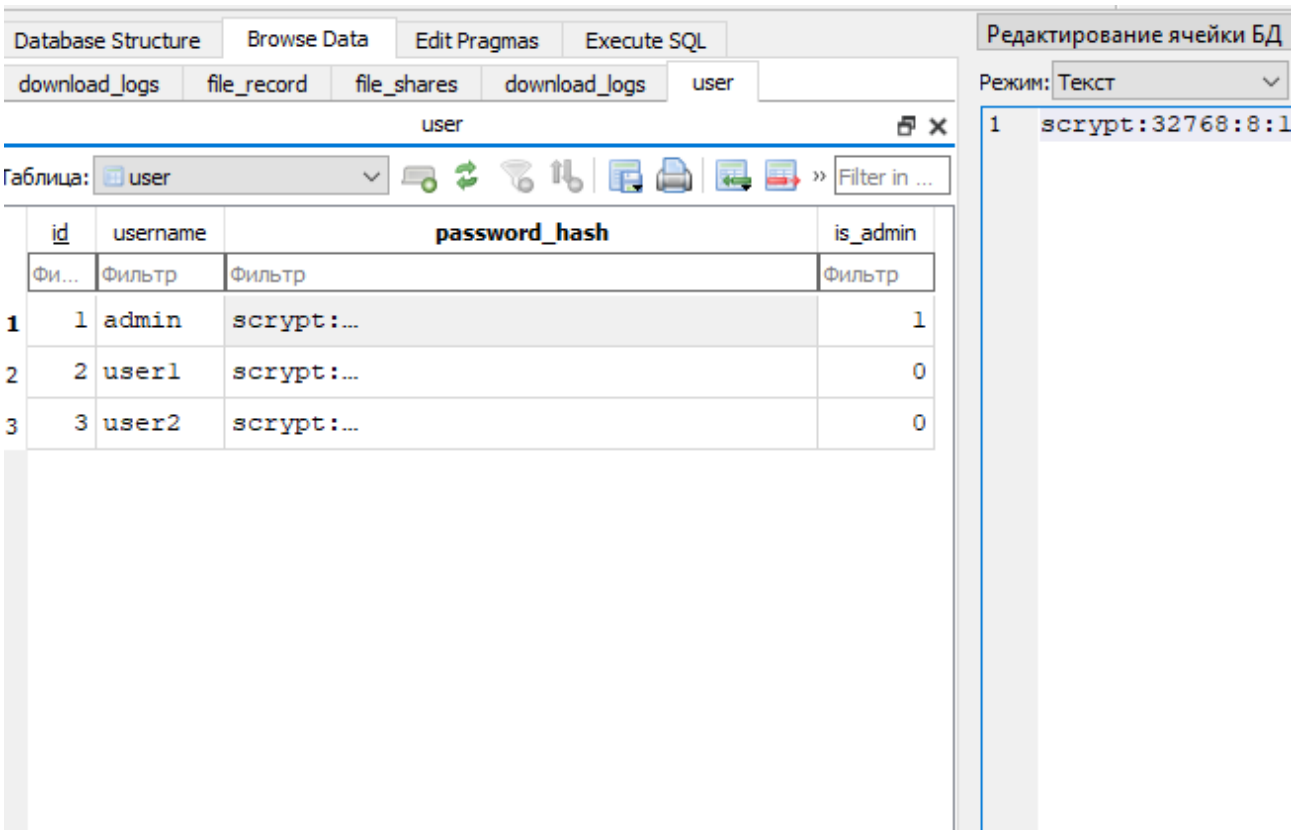
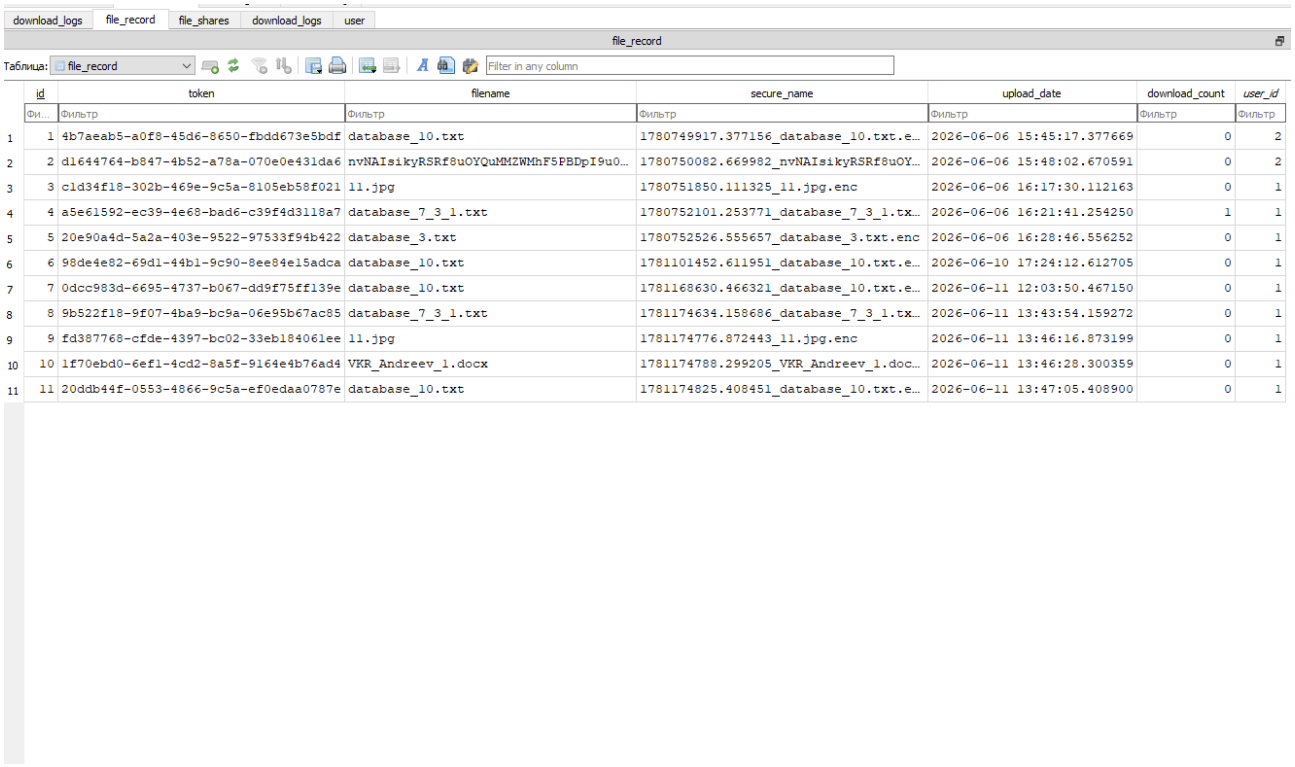


Рисунок 19 – Таблица с тестовыми данными пользователей системы

Сущность метаданных файлов предназначена для упорядочивания и хранения атрибутов загруженных в систему информационных объектов. Структура сущности включает идентификатор владельца, исходное наименование файла, уникальное служебное имя для физического размещения на диске, публичный токен доступа, временную метку загрузки и счетчик обращений. Данный компонент обеспечивает отображение структуры данных в пользовательском интерфейсе, разграничение прав доступа и установление однозначного соответствия между логической записью в базе данных и зашифрованным двоичным файлом на постоянном запоминающем устройстве. Сущность является центральным узлом схемы данных, связывающим файловые

объекты с подсистемами адресной передачи и аудита. На рисунке 20 представлена таблица внутри базы данных с тестовыми данными.



download_logs | file_record | file_shares | download_logs | user

file_record

Таблица: file_record — Filter in any column

	id	token	filename	secure_name	upload_date	download_count	user_id
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	4b7aeab5-a0f8-45d6-8650-fbdd673e5bdf	database_10.txt	1780749917.377156_database_10.txt.e…	2026-06-06 15:45:17.377669	0	2
2	2	d1644764-b847-4b52-a78a-070e0e431da6	nvNAIsikyRSRf8uOYQuMMZWMhF5PBDpI9u0…	1780750082.669982_nvNAIsikyRSRf8uOY…	2026-06-06 15:48:02.670591	0	2
3	3	c1d34f18-302b-469e-9c5a-8105eb58f021	11.jpg	1780751850.111325_11.jpg.enc	2026-06-06 16:17:30.112163	0	1
4	4	a5e61592-ec39-4e68-bad6-c39f4d3118a7	database_7_3_1.txt	1780752101.253771_database_7_3_1.tx…	2026-06-06 16:21:41.254250	1	1
5	5	20e90a4d-5a2a-403e-9522-97533f94b422	database_3.txt	1780752526.555657_database_3.txt.enc	2026-06-06 16:28:46.556252	0	1
6	6	98de4e82-69d1-44b1-9c90-8ee84e15adca	database_10.txt	1781101452.611951_database_10.txt.e…	2026-06-10 17:24:12.612705	0	1
7	7	0dcc983d-6695-4737-b067-dd9f75ff139e	database_10.txt	1781168630.466321_database_10.txt.e…	2026-06-11 12:03:50.467150	0	1
8	8	9b522f18-9f07-4ba9-bc9a-06e95b67ac85	database_7_3_1.txt	1781174634.158686_database_7_3_1.tx…	2026-06-11 13:43:54.159272	0	1
9	9	fd387768-cfde-4397-bc02-33eb18406lee	11.jpg	1781174776.872443_11.jpg.enc	2026-06-11 13:46:16.873199	0	1
10	10	1f70ebd0-6ef1-4cd2-8a5f-9164e4b76ad4	VKR_Andreev_1.docx	1781174788.299205_VKR_Andreev_1.doc…	2026-06-11 13:46:28.300359	0	1
11	11	20ddb44f-0553-4866-9c5a-ef0edaa0787e	database_10.txt	1781174825.408451_database_10.txt.e…	2026-06-11 13:47:05.408900	0	1

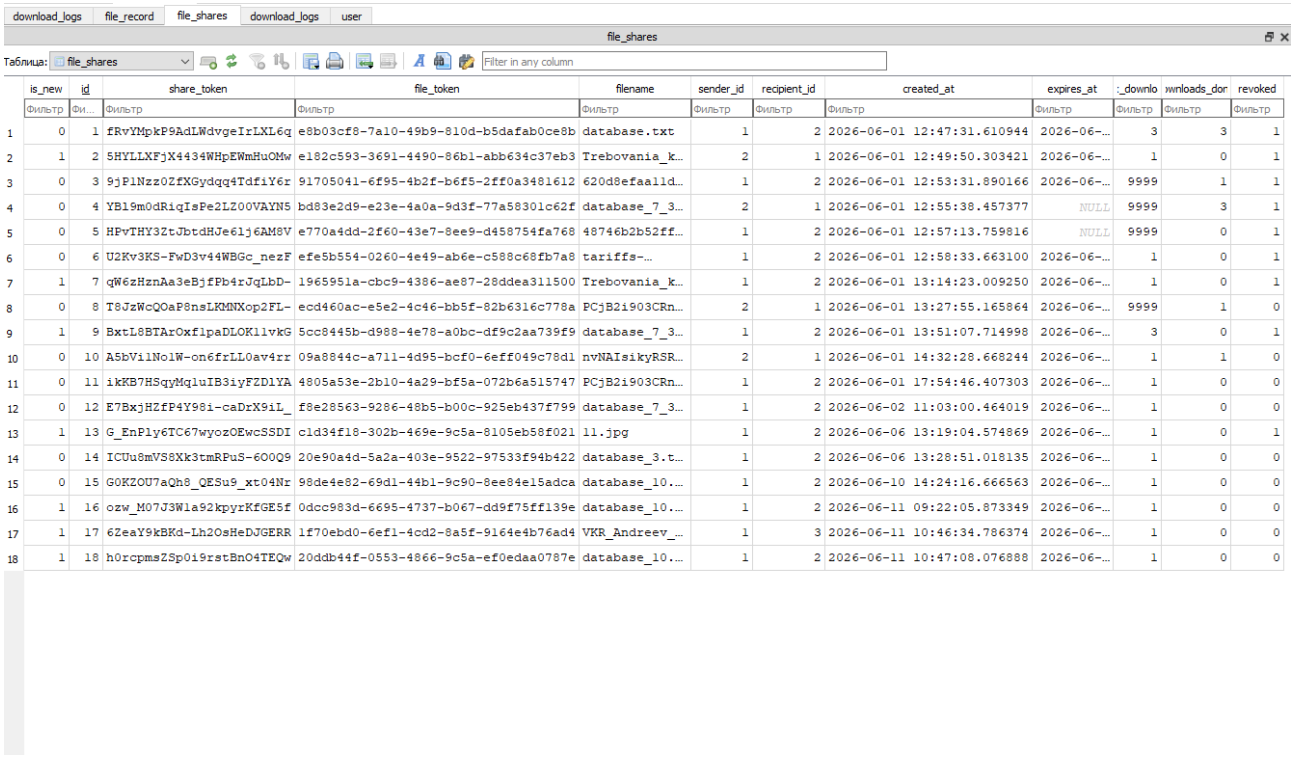
Рисунок 20 – Таблица загрузки файлов

Сущность адресного доступа формализует параметры адресной передачи информационных ресурсов между субъектами системы и является ключевым элементом механизма управления правами на получение файлов.

Набор атрибутов сущности описывает всех участников транзакции отправителя и получателя, идентификатор передаваемого файла, уникальный токен авторизации ссылки, генерируемый для каждой передачи индивидуально, срок действия предоставленного доступа, предельно допустимое количество скачиваний, а также статусные флаги, отражающие текущее состояние записи.

Сущность активно задействуется механизмами генерации и валидации персональных URL-адресов, посредством которых получатель может обратиться к ресурсу без необходимости прямой авторизации в системе. Помимо этого, она обеспечивает фильтрацию списков входящих и исходящих отправлений для каждого пользователя, контролирует соблюдение установленных лимитов на количество обращений к файлу и позволяет инициатору оперативно

аннулировать доступ к ресурсу в любой момент времени, не дожидаясь истечения срока действия ссылки. На рисунке 21 представлена таблица внутри базы данных с тестовыми данными.



download_logs | file_record | file_shares | download_logs | user

file_shares

Таблица: file_shares — Filter in any column

	is_new	id	share_token	file_token	filename	sender_id	recipient_id	created_at	expires_at	:_downlo	wnloads_dor	revoked
	Фильтр	Фи…	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	0	1	fRvYMpkP9AdLWdvgeIrLXL6q	e8b03cf8-7a10-49b9-810d-b5dafab0ce8b	database.txt	1	2	2026-06-01 12:47:31.610944	2026-06-…	3	3	1
2	1	2	5HYLLXFjX4434WHpEWmHuOMw	e182c593-3691-4490-86b1-abb634c37eb3	Trebovania_k…	2	1	2026-06-01 12:49:50.303421	2026-06-…	1	0	1
3	0	3	9jPlNzz0ZfXGydqq4TdfiY6r	91705041-6f95-4b2f-b6f5-2ff0a3481612	620d8efaalld…	1	2	2026-06-01 12:53:31.890166	2026-06-…	9999	1	1
4	0	4	YBl9m0dRiqIsPe2LZ00VAYN5	bd83e2d9-e23e-4a0a-9d3f-77a58301c62f	database_7_3…	2	1	2026-06-01 12:55:38.457377	NULL	9999	3	1
5	0	5	HPvTHY3ZtJbtdHJe6lj6AM8V	e770a4dd-2f60-43e7-8ee9-d458754fa768	48746b2b52ff…	1	2	2026-06-01 12:57:13.759816	NULL	9999	0	1
6	0	6	U2Kv3KS-FwD3v44WBGc_nezF	efe5b554-0260-4e49-ab6e-c588c68fb7a8	tariffs-…	1	2	2026-06-01 12:58:33.663100	2026-06-…	1	0	1
7	1	7	qW6zHznAa3eBjfPb4rJqLbD-	196595la-cbc9-4386-ae87-28ddea311500	Trebovania_k…	1	2	2026-06-01 13:14:23.009250	2026-06-…	1	0	1
8	0	8	T8JzWcQOaP8nsLKMNXop2FL-	ecd460ac-e5e2-4c46-bb5f-82b6316c778a	PCjB2i903CRn…	2	1	2026-06-01 13:27:55.165864	2026-06-…	9999	1	0
9	1	9	BxtL8BTArOxflpaDLOKllvkG	5cc8445b-d988-4e78-a0bc-df9c2aa739f9	database_7_3…	1	2	2026-06-01 13:51:07.714998	2026-06-…	3	0	1
10	0	10	A5bVilNolW-on6frLL0av4rr	09a8844c-a711-4d95-bcf0-6eff049c78d1	nvNAIsikyRSR…	2	1	2026-06-01 14:32:28.668244	2026-06-…	1	1	0
11	0	11	ikKB7HSqyMqluIB3iyFZDlYA	4805a53e-2b10-4a29-bf5a-072b6a515747	PCjB2i903CRn…	1	2	2026-06-01 17:54:46.407303	2026-06-…	1	0	0
12	0	12	E7BxjHZfP4Y98i-caDrX9iL_	f8e28563-9286-48b5-b00c-925eb437f799	database_7_3…	1	2	2026-06-02 11:03:00.464019	2026-06-…	1	0	0
13	1	13	G_EnPly6TC67wyozOEwcSSDI	c1d34f18-302b-469e-9c5a-8105eb58f021	11.jpg	1	2	2026-06-06 13:19:04.574869	2026-06-…	1	0	1
14	0	14	ICUu8mVS8Xk3tmRPuS-6O0Q9	20e90a4d-5a2a-403e-9522-97533f94b422	database_3.t…	1	2	2026-06-06 13:28:51.018135	2026-06-…	1	0	0
15	0	15	G0KZOU7aQh8_QESu9_xt04Nr	98de4e82-69d1-44b1-9c90-8ee84e15adca	database_10.…	1	2	2026-06-10 14:24:16.666563	2026-06-…	1	0	0
16	1	16	ozw_M07J3W1a92kpyrKfGE5f	0dcc983d-6695-4737-b067-dd9f75ff139e	database_10.…	1	2	2026-06-11 09:22:05.873349	2026-06-…	1	0	0
17	1	17	6ZeaY9kBKd-Lh2OsHeDJGERR	1f70ebd0-6ef1-4cd2-8a5f-9164e4b76ad4	VKR_Andreev_…	1	3	2026-06-11 10:46:34.786374	2026-06-…	1	0	0
18	1	18	h0rcpmsZSp0i9rstBnO4TEQw	20ddb44f-0553-4866-9c5a-ef0edaa0787e	database_10.…	1	2	2026-06-11 10:47:08.076888	2026-06-…	1	0	0

Рисунок 21 – Таблица с тестовыми метаданными файлов

Сущность журнала регистрации событий представляет собой неизменяемый реестр системных инцидентов, последовательно фиксирующий все значимые операции, происходящие в системе: авторизацию пользователей, загрузку и выгрузку файлов, а также модификацию прав доступа к ресурсам.

Для каждой записи журнала регистрируется исчерпывающий набор атрибутов, обеспечивающий полноту и достоверность хранимых сведений. В частности, фиксируются точное время совершения операции, тип транзакции, идентификатор и текстовый снимок имени инициатора, связь с соответствующим файловым объектом, сетевой IP-адрес, с которого был выполнен запрос, сигнатура клиентского приложения, итоговый результат выполнения операции, а также код возникшей ошибки в случае её наступления.

Принципиально важной особенностью данной подсистемы является её автономность: журнал функционирует независимо от жизненного цикла всех

остальных сущностей базы данных. Это делает журнал надёжным и самодостаточным источником исторических данных.

Сведения, накопленные в данной таблице, находят практическое применение в нескольких направлениях: расследовании инцидентов информационной безопасности, проведении системной аналитики и отладке программного комплекса. На рисунке 22 представлена таблица внутри базы данных с тестовыми данными.

download_logs | file_record | file_shares | download_logs | user

download_logs

Таблица: download_logs — Filter in any column

	id	ts	file_token	filename	username	ip	user_agent	result	error	action
1	1	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	admin	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
2	2	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	admin	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
3	3	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
4	4	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
5	5	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
6	6	2026-06-01 …	e182c593-3691-4490-…	Trebovania_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
7	7	2026-06-01 …	e182c593-3691-4490-…	Trebovania_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
8	8	2026-06-01 …	e182c593-3691-4490-…	Trebovania_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
9	9	2026-06-01 …	e8b03cf8-7a10-49b9-…	database.txt	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
10	10	2026-06-01 …	91705041-6f95-4b2f-…	620d8efaa11…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
11	11	2026-06-01 …	91705041-6f95-4b2f-…	620d8efaa11…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
12	12	2026-06-01 …	91705041-6f95-4b2f-…	620d8efaa11…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
13	13	2026-06-01 …	91705041-6f95-4b2f-…	620d8efaa11…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
14	14	2026-06-01 …	bd83e2d9-…	database_7_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
15	15	2026-06-01 …	bd83e2d9-…	database_7_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
16	16	2026-06-01 …	bd83e2d9-…	database_7_…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
17	17	2026-06-01 …	bd83e2d9-…	database_7_…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
18	18	2026-06-01 …	bd83e2d9-…	database_7_…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	download_inbox
19	19	2026-06-01 …	bd83e2d9-…	database_7_…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
20	20	2026-06-01 …	e770a4dd-2f60-43e7-…	48746b2b52f…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
21	21	2026-06-01 …	e770a4dd-2f60-43e7-…	48746b2b52f…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
22	22	2026-06-01 …	efe5b554-0260-4e49-…	tariffs-…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
23	23	2026-06-01 …	efe5b554-0260-4e49-…	tariffs-…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
24	24	2026-06-01 …	efe5b554-0260-4e49-…	tariffs-…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
25	25	2026-06-01 …	e770a4dd-2f60-43e7-…	48746b2b52f…	admin	127.0.0.1	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
26	26	2026-06-01 …	1965951a-cbc9-4386-…	Trebovania_…	admin	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
27	27	2026-06-01 …	1965951a-cbc9-4386-…	Trebovania_…	admin	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create
28	28	2026-06-01 …	1965951a-cbc9-4386-…	Trebovania_…	admin	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_revoke
29	29	2026-06-01 …	ecd460ac-e5e2-4c46-…	PCjB2i903CR…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	upload
30	30	2026-06-01 …	ecd460ac-e5e2-4c46-…	PCjB2i903CR…	user1	192.168.0.129	Mozilla/5.0 (Windows NT 10.0; Win64…	success	NULL	share_create

Рисунок 22 – Таблица журнала с тестовыми данными

Спроектированная реляционная структура базы данных обеспечивает полный жизненный цикл хранения и обработки информации в системе. Разделение данных на специализированные сущности, позволяет эффективно разграничивать права доступа, организовывать безопасный обмен файлами и вести непрерывный аудит действий пользователей. Архитектурное решение по сохранению текстовых снимков метаданных в журнале событий гарантирует целостность и неизменяемость истории транзакций, что полностью удовлетворяет требованиям к подсистемам информационной безопасности веб-сервисов.

4 Тестирование

4.1 Тестирование методом белого ящика

Для проверки корректности работы модуля валидации загрузки выполнено модульное тестирование. В качестве критерия полноты использован метод модифицированного покрытия условий и решений, обеспечивающий что каждое элементарное условие независимо влияет на итоговое решение. Программный код приведён в листинге 7.

Листинг 7 – Тестируемый код

```
# tests/test_validate_upload_mc_dc.py
import pytest

def validate_upload(file_present: bool, filename_nonempty: bool,
size_ok: bool, ext_ok: bool) -> str:
    if not file_present:
        return "E_NO_FILE"
    if not filename_nonempty:
        return "E_EMPTY_NAME"
    if (not size_ok) or (not ext_ok):
        return "E_POLICY"
    return "OK"

@pytest.mark.parametrize(
    "file_present, filename_nonempty, size_ok, ext_ok, expected",
    [
        (0, 1, 1, 1, "E_NO_FILE"),   # T_A
        (1, 0, 1, 1, "E_EMPTY_NAME"),# T_B
        (1, 1, 0, 1, "E_POLICY"),    # T_C
        (1, 1, 1, 0, "E_POLICY"),    # T_D
        (1, 1, 1, 1, "OK"),          # T_OK
    ],
)
def test_validate_upload_mc_dc(file_present, filename_nonempty,
size_ok, ext_ok, expected):
    assert validate_upload(bool(file_present),
bool(filename_nonempty), bool(size_ok), bool(ext_ok)) == expected
```

Код содержит функцию проверки файла перед загрузкой и тесты для неё. Функция принимает четыре параметра: наличие файла, непустое имя, допустимый размер и разрешённое расширение. Если хотя бы одно из условий нарушено, функция возвращает соответствующий код ошибки, например E_NO_FILE или E_POLICY. Если все условия выполнены, возвращается OK.

Тесты проверяют каждый возможный сбой отдельно: в каждом тест-кейсе ровно один параметр является ложным, а остальные – истинными. Это позволяет убедиться, что каждое условие самостоятельно влияет на итог проверки. На рисунке 23 представлена блок схема, описывающая алгоритм работы тестирования

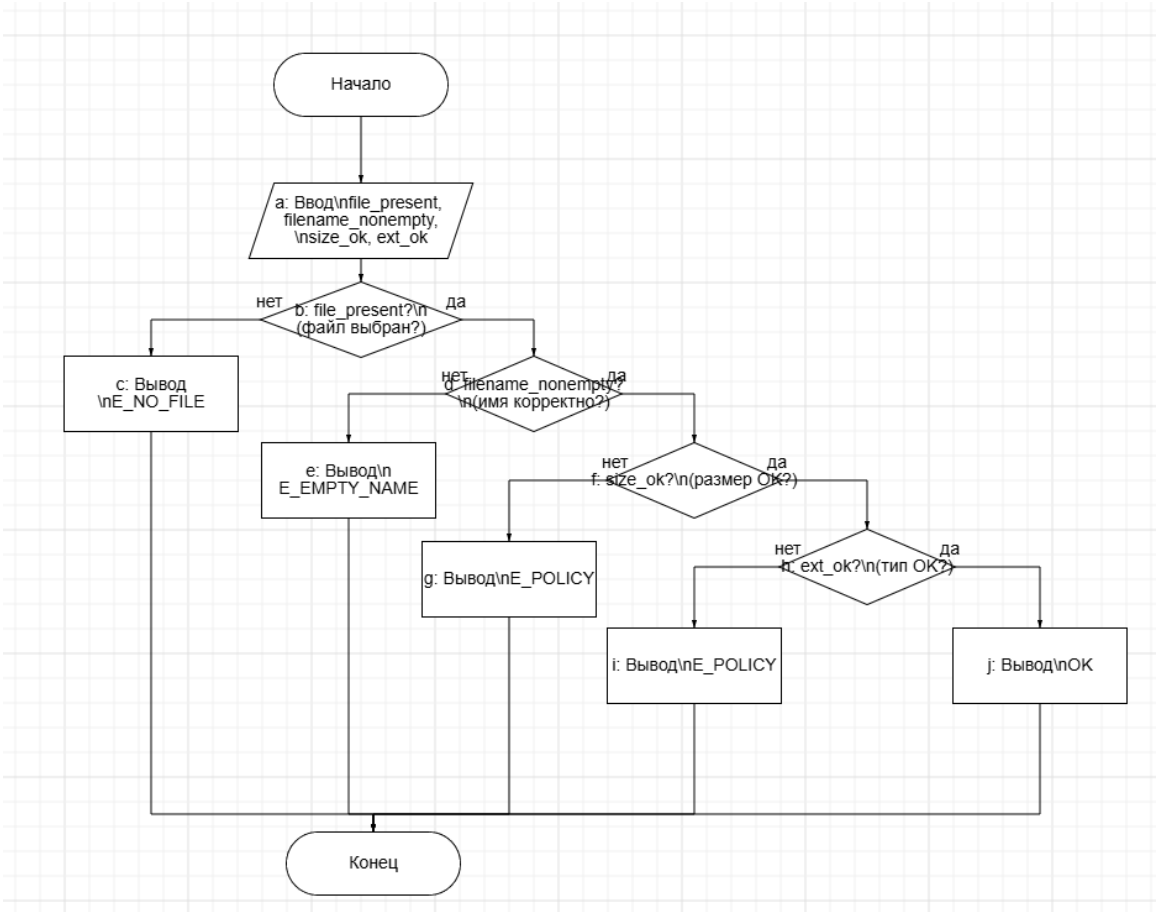


Рисунок 23 – Блок схема

Для документирования результатов тестирования методом «белого ящика» и демонстрации полноты покрытия логических путей были сформированы таблицы испытаний. В качестве объекта тестирования выступает предикат валидации загружаемых файлов, содержащий четыре последовательных условия:

– присутствие файла на сервере;
– наличие непустого имени файла;
– соответствие лимитам размера;

– корректность расширения файла.

Результаты тестирования при различных входных данных операторов приведены в таблице 6.

Таблица 6 – Результаты тестирования операторов

№	Операторы	Входные данные	Выходные данные
1	S1 return E_NO_FILE	0, 0, 0, 0	E_NO_FILE
2	S2 return E_EMPTYNAME	1, 0, 1, 1	E_EMPTY_NAME
3	S3 return E_POLICY	1, 1, 0, 1	E_POLICY
4	S4 return OK	1, 1, 1, 1	OK

Таблица фиксирует сквозную проверку выполнения отдельных операторов возврата значений в зависимости от комбинации входных булевых векторов. Необходимо доказать, что каждый изолированный блок обработки ошибок и успешного завершения функции физически достижим и возвращает код ответа

Результаты тестирования покрытия решений приведены в таблице 7.

Таблица 7 – Покрытие решений

№	Решения if not file_present (D1), if not filename_nonempty (D2), if (not size_ok) or (not ext_ok) (D3)	Входные данные	Выходные данные
1	D1=Да	0, 0, 0, 0	E_NO_FILE
2	D1=Нет; D2=Да	1, 0, 1, 1	E_EMPTY_NAME
3	D1=Нет; D2=Нет; D3=Да	1, 1, 0, 1	E_POLICY
4	D1=Нет; D2=Нет; D3=Нет	1, 1, 1, 1	OK

Таблица описывает верификацию обобщенного логического решения, описываемого составной формулой:

«if_not file_present or not filename_nonempty or not size_ok or not ext_ok»

Проверить реакцию системы на обработку как общего отрицательного исхода (когда хотя бы одна проверка возвращает False), так и единственного

полностью валидного пути, когда все условия принимают значение «Истина». Результаты тестирования покрытия условий приведены в таблице 8.

Таблица 8 – Покрытие условий

№	Условие C1: not file_present (в D1), C2: not filename_nonempty (в D2), C3: not size_ok (часть D3), C4: not ext_ok (часть D3)	Входные данные	Выходные данные
1	C1=True	0, 0, 0, 0	E_NO_FILE
2	C1=False; C2=True	1, 0, 1, 1	E_EMPTY_NAME
3	C2=False; C3=True	1, 1, 0, 1	E_POLICY
4	C3=False; C4=True (C4 вычислился)	1, 1, 1, 0	E_POLICY
5	C4=False	1, 1, 1, 1	OK

Данная таблица демонстрирует наиболее глубокий уровень верификации «белого ящика», отражая атомарное переключение каждого из четырех булевых условий в изолированные состояния True и False.

4.3 Расчёт метрик кода тестирования

Для оценки качества разработанного программного обеспечения был проведён анализ ключевого модуля серверной части – routes.py, реализующего Flask-маршруты, обработку загрузки и скачивания файлов, а также функции администрирования. Анализ выполнен по шести группам метрик.

Метрика SLOC показывает объём программного модуля и косвенно характеризует сложность его сопровождения. Результаты анализа модуля представлены в таблице 9.

Таблица 9 – Результаты SLOC-анализа модуля

Показатель	Значение
Total LOC (физические строки)	255
Blank lines (пустые строки)	57
Comments (комментарии)	37
в том числе однострочные	32
в том числе многострочные	24

Продолжение таблицы 9

Показатель	Значение
SLOC (значимые строки кода)	142
LLOC (логические строки)	135
Доля комментариев к физическим строкам (C % L)	15%
Доля комментариев к значимым строкам (C % S)	26%
Доля комментариев и многострочных к общему объёму (C + M % L)	24%

Модуль имеет умеренный объём в 142 значимые строки кода. Доля комментариев составляет 26% от значимых строк, что положительно сказывается на читаемости и понимании кода при сопровождении. При дальнейшем росте функциональности целесообразно разделить слой маршрутов и бизнес-логику, вынося части кода в отдельные модули или сервисные классы.

Метрики Холстеда оценивают информационный объём программы, её сложность и трудоёмкость сопровождения на основе подсчёта уникальных и общих операторов и операндов. Входные значения и расчётные показатели представлены в таблицах 10 – 11.

Таблица 10 – Входные значения

Параметр	Обозначение	Значение
Уникальные операторы	n1	9
Уникальные операнды	n2	28
Общее число операторов	N1	17
Общее число операндов	N2	29

Таблица 11 – Расчётные показатели

Показатель	Формула	Значение
Словарь программы	n = n1 + n2	37
Длина программы	N = N1 + N2	46
Объём	$V = N \times \log_2(n)$	239,63
Сложность	D = (n1/2) × (N2/n2)	4,66

Продолжение таблицы 11

Показатель	Формула	Значение
Усилие	E = V × D	1116,87
Оценка времени разработки	T = E / 18, сек	62,05
Оценка числа дефектов	B = V / 3000	0,0799

В таблице 12 приведены операторы, выявленные при анализе модуля.

Таблица 12 – Операторы модуля routes.py

№	Оператор	Назначение	Частота
1	If	Ветвление/решение	6
2	Return	Возврат результата	4
3	Not	Логическое отрицание	1
4	Or	Логическое ИЛИ	1
5	And	Логическое И	1
6	With	Контекстный менеджер	1
7	Try	Обработка исключений	1
8	Except	Ветка исключений	1
9	=	Присваивание	1
Итого	N1 = 17		

В таблице 13 приведены основные операнды, выявленные при анализе модуля.

Таблица 13 – Операнды модуля routes.py

№	Операнд	Роль в модуле	Частота
1	Db	Объект базы данных (SQLAlchemy session)	2
2	Request	HTTP-запрос (форма/файлы/заголовки)	1
3	current_user	Текущий пользователь (Flask-Login)	1
4	User	Модель пользователя	1
5	FileRecord	Модель файла	1
6	DownloadLog	Модель журнала аудита	1

Продолжение таблицы 13

№	Операнд	Роль в модуле	Частота
7	token	Уникальный идентификатор ссылки	1
8	file	Загружаемый файл (FileStorage)	1
9	filename	Исходное имя файла	1
10	secure_name	Имя файла на диске	1
11	upload_folder	Каталог хранения файлов	1
12	file_path	Путь к файлу на диске	1
13	encrypted_data	Зашифрованные байты	1
14	decrypted_data	Расшифрованные байты	1
15	cpu_usage	Метрика CPU (psutil)	1
16	memory	Метрика RAM (psutil)	1
17	disk	Метрика диска (psutil)	1
18	system_stats	Агрегированные метрики в администрировании	1
19	flash	Уведомления пользователю	1
20	redirect	Редирект после действий	1
21	url_for	Генерация URL-адресов	1
22	render_template	Рендер HTML-шаблона	1
23	send_file	Отдача файла клиенту	1
24	io.BytesIO	Выдача файла из памяти	1
25	psutil	Сбор системных метрик	1
26	datetime	Работа с временными метками	1
27	os.path.exists	Проверка существования файла	1
28	«OK»	Строковая константа результата	1
Итого	N2 = 29		

Для подтверждения наличия указанных операторов и операндов в коде ниже в листингах 8 – 9, приведены характерные фрагменты листинга модуля routes.py.

Листинг 8 – Фрагмент обработки загрузки файла

```
file = request.files.get("file")              # request, file
filename = file.filename if file else ""      # filename, "="
if not file:                                  # if, not
    flash("Файл не выбран")                   # flash
    return redirect(url_for("main.index"))    # return, redirect,
url_for
cpu_usage = psutil.cpu_percent(interval=1)          # psutil,
cpu_usage
memory    = psutil.virtual_memory()                 # memory
disk      = psutil.disk_usage(os.path.abspath(os.sep))  # disk
system_stats = {                                    # system_stats
    "cpu":          cpu_usage,
    "ram_percent":  memory.percent,
    "disk_free_gb": round(disk.free / (1024**3), 2)
}
return render_template("admin.html", stats=system_stats)  #
render_template
```

Полученные значения объёма V = 239,63 и сложности D = 4,66 находятся на низком и умеренном уровнях соответственно, что в целом свидетельствует о хорошем качестве анализируемого кода. Код не перегружен операторами, легко читается и не создаёт значительной когнитивной нагрузки при сопровождении, что упрощает его дальнейшую поддержку и развитие. Оценка числа дефектов B = 0,08 свидетельствует о низкой вероятности скрытых ошибок в анализируемом фрагменте, что дополнительно подтверждает его структурную простоту и корректность реализации.

Метрика Чепина предоставляет более детальный взгляд на качество кода, классифицируя все переменные программы по их функциональной роли. В рамках данной метрики выделяются четыре категории: входные данные (P), непосредственно участвующие в вычислениях, модифицируемые данные (M), значения которых изменяются в ходе выполнения программы, управляющие переменные (C), влияющие на логику ветвления, а также временные и служебные переменные (T), используемые для вспомогательных операций. В

таблице 14 представлена классификация переменных по метрике Чепина.
Таблица 14 – Классификация переменных по метрике Чепина

Класс	Описание	Переменные модуля
P (Input)	Входные данные	request, request.files, request.form, token (параметр URL), current_user, результаты запросов к БД
M (Modified)	Модифицируемые данные	encrypted_data, decrypted_data, storage_name, объекты моделей при создании записей, download_count, system_stats
C (Control)	Управляющие переменные	current_user.is_admin, проверки корректности входных данных, проверки наличия файла на диске, ветвления по HTTP-методу
T (Temporary)	Временные/служебные	file_path, промежуточные переменные вычисления CPU/RAM/Disk, переменные циклов

Преобладание переменных классов P, M и T является нормальным для контроллерного слоя серверного приложения, связывающего HTTP-запросы, файловую систему и базу данных. Умеренное число управляющих переменных класса C указывает на контролируемую ветвистость модуля без признаков избыточной условной логики.

Цикломатическая сложность отражает число независимых путей выполнения в программе и напрямую связана с тестируемостью и риском ошибок. Анализ выполнен инструментом radon. Результаты приведены в таблице 15.

Таблица 15 – Цикломатическая сложность функций модуля routes.py

Функция	Оценка	Значение	Интерпретация
admin_cleanup	B	6	Допустимый уровень
Login	A	5	Хороший уровень
index	A	5	Хороший уровень
download_file	A	3	Хороший уровень
admin_dashboard	A	3	Хороший уровень

Продолжение таблицы 15

Функция	Оценка	Значение	Интерпретация
create_user	A	3	Хороший уровень
logout	A	1	Отличный уровень
success	A	1	Отличный уровень

Исходя из проверки сложности, 7 из 8 функций получили оценку A с цикломатической сложностью от 1 до 5, что соответствует хорошему уровню тестируемости. Функция admin_cleanup (B = 6) находится в допустимых пределах. При дальнейшем усложнении логики автоматической очистки устаревших данных рекомендуется вынести часть кода в отдельную функцию или сервисный модуль.

Индекс поддерживаемости – это сводная метрика, учитывающая объём кода, цикломатическую сложность и степень комментирования. Анализ выполнен инструментом radon mi. Результат: routes.py – уровень A, MI = 62,76

Модуль получил наивысший уровень поддерживаемости A. Это означает, что код пригоден для длительного сопровождения без значительного роста трудозатрат. Дальнейшее повышение MI достижимо за счёт снижения связанности модуля и вынесения бизнес-логики из обработчиков маршрутов в отдельные сервисные классы.

Помимо количественных метрик, модуль был оценён экспертно по четырём критериям качества программного кода, что позволяет составить более полное и всестороннее представление о его состоянии. Экспертная оценка даёт возможность выявить те аспекты качества, которые сложно выразить числовыми показателями, – такие как читаемость, устойчивость к сбоям, архитектурная связность и безопасность реализованных решений.

По критерию читаемости модуль получил оценку «Хорошо»: функции имеют понятные и говорящие имена, комментарии присутствуют, а код структурирован по зонам ответственности. Отказоустойчивость оценена как «Удовлетворительно / Хорошо» – базовые проверки реализованы, однако обработку исключений при работе с файлами рекомендуется усилить. Связность

модуля оценена как средняя, поскольку он совмещает обработку HTTP-запросов и бизнес-логику, что в перспективе желательно устранить путём разделения ответственности. Безопасность получила оценку «Хорошо» благодаря реализованному шифрованию файлов и контролю доступа, хотя для промышленного развёртывания рекомендуется вынести ключи шифрования из кодовой базы. В таблице 16 представлена экспертная оценка качества модуля.

Таблица 16 – Экспертная оценка качества модуля routes.py

Критерий	Оценка	Обоснование
Читаемость	Хорошо	Функции имеют понятные имена (index, download_file, admin_dashboard), комментарии присутствуют, код структурирован по зонам ответственности
Отказоустойчивость	Удовлетворительно / Хорошо	Реализованы проверки на отсутствие файла и некорректные входные данные; для повышения надёжности рекомендуется усилить обработку исключений при чтении и расшифровании файлов
Связность	Средняя	Модуль совмещает обработку HTTP-запросов и бизнес-логику. Рекомендуется постепенный вынос валидации, работы с файлами и аудита в отдельные каталоги

Продолжение таблицы 16

Безопасность	Хорошо	Реализованы шифрование файлов при хранении и контроль доступа к административным функциям. Для промышленного развёртывания необходимо вынести ключи шифрования из кодовой базы и отключить режим отладки

Сводные результаты анализа метрик модуля routes.py представлены в таблице 17.

Таблица 17 – Сводные результаты анализа метрик

Метрика	Результат	Оценка
SLOC	142 строки	Умеренный объём
Доля комментариев	26% от SLOC	Достаточная
Объём по Холстеду (V)	239,63	Низкий
Сложность по Холстеду (D)	4,66	Умеренная
Оценка дефектов (B)	0,08	Низкая
Цикломатическая сложность	A (макс. B=6)	Хорошая
Индекс поддерживаемости (MI)	A (62,76)	Хороший

Анализ метрик подтверждает, что модуль routes.py обладает умеренным размером, невысокой цикломатической сложностью и хорошим индексом поддерживаемости. Код пригоден для дальнейшего расширения и сопровождения. Основной вектор улучшения, это постепенная декомпозиция обработчиков маршрутов с выносом бизнес-логики в отдельные сервисные модули по мере роста функциональности системы.

4.4 Нагрузочное тестирование

Для проверки работоспособности разрабатываемого веб-сервиса, необходимо оценить, как система будет справляться с нагрузкой. Нагрузочное тестирование позволяет оценить, как программное обеспечение справиться с

нагрузкой. В данной работе использовался инструмент Apache Jmeter, позволяющий имитировать работу сотен или тысяч пользователей.

Целью нагрузочного тестирования является оценка производительности и устойчивости веб сервиса при конкурентном доступе к основной сервисной функции – скачиванию файла по уникальной ссылке (GET /d/<token>). Дополнительно оценивается отсутствие ошибок (HTTP) при нагрузке.

В качестве основного объекта экспериментального исследования производительности разработанного программного комплекса был выбран точка входа GET /d/${token}, обеспечивающая выгрузку зашифрованных файлов по уникальным идентификаторам. Данный интерфейс является критически важным компонентом системы, так как реализует ключевой пользовательский сценарий – предоставление санкционированного доступа к защищенным информационным объектам подсистемы хранения. Тестирование направлено на определение пропускной способности сервера, оценку времени отклика и выявление потенциальных узких мест архитектуры при масштабировании сетевой нагрузки. Для имитации запросов и последующему объединению статистических метрик производительности использовалась специализированная среда тестирования Apache JMeter версии 5.6.3. Инструмент позволяет генерировать синтетическую многопоточную нагрузку на исследуемые сетевые узлы в соответствии с заданным профилем.

Дополнительным средством верификации выступает встроенный системный журнал веб-приложения на базе Flask, выполняющий сквозную регистрацию транзакций для подтверждения корректности обработки запросов и фиксации кодов HTTP-ответов сервера 200 OK. Испытания проводились в изолированной тестовой среде на локальном сетевом интерфейсе по адресу http://127.0.0.1:5000.

С целью имитации реального поведения множества распределенных клиентов был подготовлен структурированный текстовый файл tokens.csv, содержащий репрезентативный набор из 5 валидных токенов доступа. Динамическая подстановка параметров в структуру HTTP-запроса метода GET в

процессе тестирования была автоматизирована посредством специализированного конфигурационного модуля интеграции внешних данных Apache JMeter.

Структурная схема сценария тестирования в среде JMeter включает в себя комплекс связанных функциональных компонентов. Управление интенсивностью и характером нагрузки возложено на группу потоков, задающую количество виртуальных пользователей и параметры генерации параллельных сессий. Глобальные сетевые параметры такие как целевой хост и порт, определены в элементе HTTP Request Defaults, что оптимизирует структуру запроса HTTP Request, непосредственно вызывающего метод GET /d/${token}. Для чтения данных из внешнего источника используется контроллер CSV Data Set Config, передающий переменную token в тело запроса. Финальный сбор, агрегация и визуализация расчетных метрик осуществляются с помощью сводного компонента анализа Aggregate Report. Результаты тестирования представлены в таблице 18.

Таблица 18 – Результаты нагрузочного тестирования

Показатель	Значение
Количество запросов	1200
Ошибки	0.00%
Среднее время ответа	11 ms
Медиана	8 ms
90% перцентиль	18 ms
95% перцентиль	19 ms
99% перцентиль	35 ms
Минимум (Min)	5 ms
Максимум (Max)	91 ms
Пропускная способность	2.2 req/sec
Скорость получения данных	817.40 KB/sec
Скорость отправки данных	0.33 KB/sec

Также журнал сервера подтверждает успешные ответы. Результаты тестирования представлены на рисунке 24.

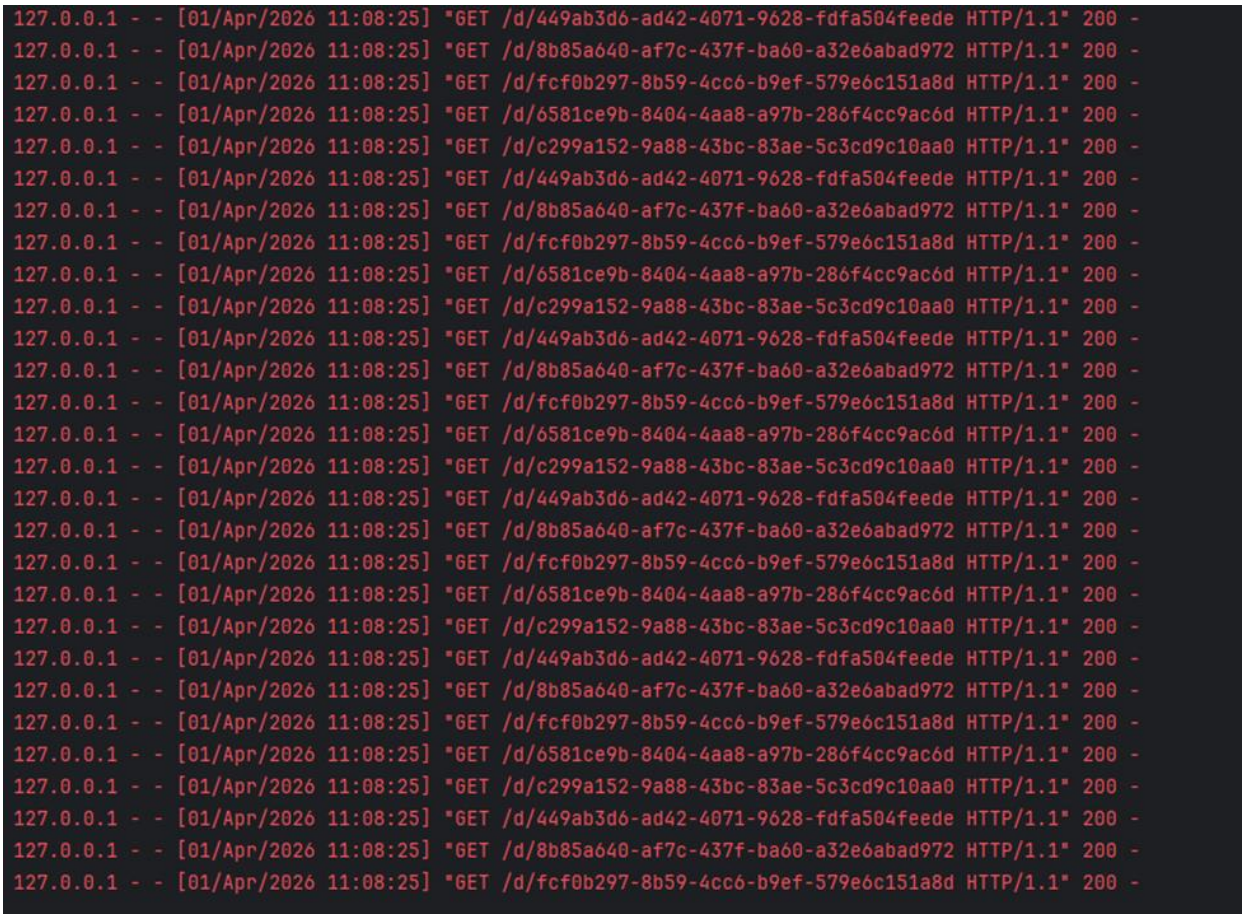


Рисунок 24 – Результаты тестирования

Все зарегистрированные HTTP-запросы завершаются с кодом ответа 200 что означает «Успешно» и флагом успешной передачи содержимого. Отсутствие кодов ошибок доказывает, что при заданном профиле многопоточной нагрузки сервер успешно справляется с аутентификацией сессий, обработкой маршрутов и отдачей зашифрованных файлов без отказов в обслуживании.

4.4 Тестирование интерфейса веб-сервиса

Тестирование интерфейса является ключевой задачей для приложения. Рядовой пользователь или администратор не должен сталкиваться с различными проблемами во время пользования программным обеспечением. Разрабатываемый веб-сервис будет протестирован на наличие ошибок интерфейса. Будут протестированы ситуации, при которых могут возникнуть ошибки. Все графические элементы системы пройдут тестирование. В таблице 19 представлены все тесты интерфейса системы и их результат.

Таблица 19 – Описание проведённых тестов интерфейса системы

Описание теста	Ожидание	Фактический результат	Итог
Вход пользователя в свою учётную запись	Пользователь вводит данные своей учётной записи (логин и пароль) и получает доступ к системе	При вводе правильных данных получает доступ к системе	Результат тестирования представлен на рисунке В.1
Попытка пользователя ввести неверные данные	Пользователь пытается ввести неправильные данные (логин и пароль), спустя несколько попыток система запретит ввод данных и попросит подождать минуту	Система не пускает пользователя и ограничивает ввод логин и пароля	Результат тестирования представлен на рисунке В.2
Проверка загрузки файла	Файлы загружаются и шифруются в системе	Файл успешно загружен	Результат тестирования представлен на рисунке В.3
Проверка отправки файлов	Файлы успешно отправляются с помощью меню отправки	Файл успешно отправлен	Результат тестирования представлен на рисунке В.4
Попытка отправить файл запрещённого типа	Система не позволит загрузить потенциально опасный файл	Файл не загружен	Результат тестирования представлен на рисунке В.5
Генерация уникальной ссылки для отправки вне системы	Ссылка успешно создана	Ссылка успешно создана	Результат тестирования представлен на рисунке В.6

Продолжение таблицы 19

Описание теста	Ожидание	Фактический результат	Итог
Ограничение срока жизни ссылки	Ссылка доступна определённое время, после чего перестаёт работать	Ссылка недоступна через определённое время	Результат тестирования представлен на рисунке В.7
Возможность отозвать отправленную ссылку	Пользователь может отозвать доступ к отправленному файлу	Ссылка успешно отозвана	Результат тестирования представлен на рисунке В.8
Проверка входящих сообщений	Возможность видеть файлы, отправленные другим пользователем	Входящие файлы доступны и могут быть скачены	Результат тестирования представлен на рисунке В.9
Возможность создавать новых пользователей	Администратор может добавлять пользователей	Пользователь успешно добавлен	Результат тестирования представлен на рисунках В.10 – В.11
Панель мониторинга состояния сервера обновляется в реальном времени	Панель мониторинга сервера обновляется каждые несколько секунд	Панель мониторинга сервера обновляется каждые несколько секунд	Результат тестирования представлен на рисунках В.12 – В.13
Журнал аудирования записывает все действия на сервере	Журнал обновляется и записывает данные	Журнал обновляется и записывает данные	Результат тестирования представлен на рисунке В.14

Результаты тестирования графического интерфейса представлены на рисунках в приложении В.

5. Планирование и расчёт бюджета разрабатываемой системы

Для планирования и расчета бюджета была выбрана диаграмма Ганта. Разработанная диаграмма приведена на рисунке 25.

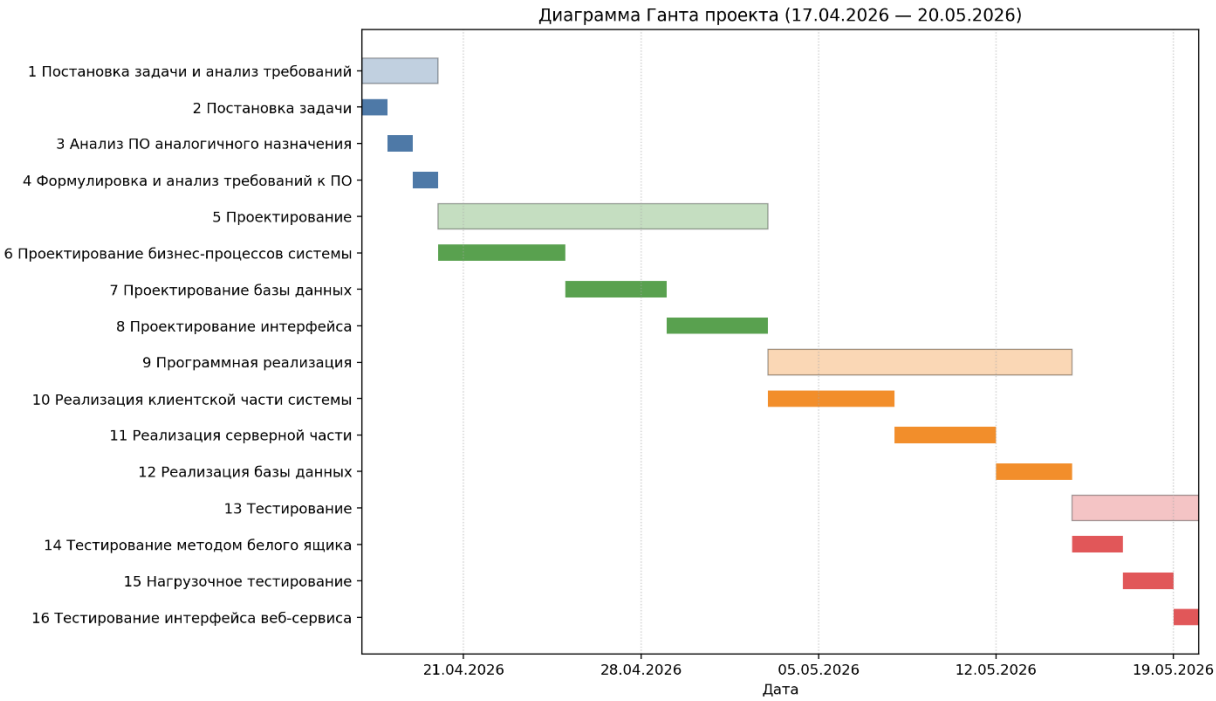


Рисунок 25 – Диаграмма Ганта

На представленной диаграмме показаны все задачи выполненные за период разработки системы. Присутствует 4 основных задачи: «Постановка задачи и анализ требований», «Проектирование», «Программная реализация» и «Тестирование». Каждая из этих задач состоит из подзадач.

Информация об исполнителях каждого этапа представлена в таблице 20.

Таблица 20 – Исполнители этапов проектирования системы

Исполнитель	Стоимость
Системный аналитик	300 рублей в час
Разработчик клиента (Fronted-разработчик)	250 рублей в час
Разработчик серверной части (Backend-разработчик)	300 рублей в час
Тестировщик	150 рублей в час

Также был рассчитан бюджет разработки и проектирования системы. В таблице 21 представлен расчет бюджета разработанной системы.

Таблица 21 – Расчёт бюджета разработанной системы

Задача	Исполнитель	Фактическая стоимость	Статус	Затраченное время
Постановка задачи и анализ требований		8400 руб.		28 часов
Постановка задачи	Системный аналитик	3000 руб.	Выполнено	10 часов
Анализ программного обеспечения аналогичного назначения	Системный аналитик	3600 руб.	Выполнено	12 часов
Формулировка и анализ требований к разрабатываемому программному обеспечению	Системный аналитик	1800 руб.	Выполнено	6 часов
Проектирование		10200 руб.		34 часа
Проектирование бизнес-процессов системы	Системный аналитик	4800 руб.	Выполнено	16 часов
Проектирование базы данных	Разработчик серверной части	1200 руб.	Выполнено	4 часа
Проектирование интерфейса	Разработчик клиента	4200 руб.	Выполнено	14 часов
Программная реализация		14800 руб.		52 часа
Программная реализация клиентской части системы	Разработчик клиента	4000 руб.	Выполнено	16 часа
Программная реализация серверной части	Разработчик серверной части	7200 руб.	Выполнено	24 часа

Продолжение таблицы 21

Задача	Исполнитель	Фактическая стоимость	Статус	Затраченное время
Программная реализация базы данных	Системный аналитик	3600 руб.	Выполнено	12 часа
Тестирование		2950 руб.		33 часа
Тестирование методом белого ящика	Тестировщик	1200 руб.	Выполнено	8 часов
Нагрузочное тестирование	Тестировщик	1000 руб.	Выполнено	20 часов
Тестирование интерфейса веб-сервиса	Тестировщик	750 руб.	Выполнено	5 часов

По итогами расчета бюджета разработанной системы, итоговая стоимость проектирования и разработки приложения равняется 38 350 рублей.

Был спроектирован бюджет разработки системы, установлены сроки выполнения проекта.

Заключение

Разработанная система решает проблему безопасности предприятия. Сотрудники предприятий передают файлы через личную почту и публичные мессенджеры, а это прямая утечка корпоративных данных. Программное обеспечение закрывает этот пробел.

Изучены риски и сформулированы требования. Сравнение с Nextcloud и Seafile показало, что ни одно из готовых решений не сочетает встроенный контроль целостности журнала аудита через хеш-цепочку с лимитом скачиваний.

Спроектирована архитектура. Построены диаграммы вариантов использования, деятельности и развёртывания. Определены два типа пользователей, обычный и администратор. База данных SQLite включает четыре таблицы: учётные записи, метаданные файлов, адресные передачи и журнал событий. Связи и ограничения целостности прописаны явно.

Разработан клиентский интерфейс. Восемь HTML-шаблонов на Jinja2 покрывают весь пользовательский сценарий. Вход, загрузка файла, адресная передача, входящие и исходящие передачи, отзыв доступа, административная панель и журнал аудита. Метрики сервера обновляются в реальном времени через асинхронные JavaScript-запросы с интервалом две секунды.

Написана серверная часть. Шесть модулей на Python и Flask. Модуль инициализации запускает фоновую очистку устаревших данных и фильтр AuditChainFilter который связывает каждую запись аудита с предыдущей через SHA-256, делая журнал безопасным. Модуль маршрутизации реализует всю бизнес-логику, и препятствует перебору паролей. Файлы шифруются алгоритмом Fernet. Ротация ключей поддерживается без потери доступа к ранее зашифрованным данным. Конфигурация автоматически адаптирует пути к директориям под Windows и Linux, без правки кода при переносе. Сервис запускается тремя способами:

– отладочный режим;

– режим без отладки;

– графический интерфейс.

Проведено тестирование. Модульные тесты методом белого ящика проверили все четыре ветви валидации загружаемых файлов. Нагрузочное тестирование в Apache JMeter показало, что за 1200 запросов, ошибок 0. среднее время ответа 11 мс, медиана 8 мс. 12 сценариев пользовательского интерфейса прошли без замечаний.

Рассчитан бюджет. Построена диаграмма Ганта, распределены исполнители. Общая трудоёмкость 147 часов. Итоговая стоимость разработки 38350 рублей.

Система полностью реализует требования:

– шифрование файлов при хранении;

– разграничение прав по ролям;

– адресная передача с настраиваемым сроком и лимитом скачиваний;

– мгновенный отзыв доступа, проверяемый журнал аудита и мониторинг сервера в режиме реального времени.

Список использованных источников

1. Nextcloud Documentation // Nextcloud [Электронный ресурс]. URL: https://docs.nextcloud.com/server/stable/admin_manual/index.html (дата обращения: 25.05.2026).

2. Seafile Documentation // Seafile [Электронный ресурс]. URL: https://manual.seafile.com/latest/ (дата обращения: 25.06.2026)

3. Сваруп, К. A Byte of Python / К. Сваруп; пер. с англ. В. Смоляр. – [Электронный ресурс]. URL: https://www.swaroopch.com (дата обращения: 25.05.2026).

4. Шоттс, У. Командная строка Linux. Полное руководство / У. Шоттс; пер. с англ. А. Н. Киселев. – СПб.: Питер, 2024. – 544 с.

5. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В. Г. Олифер, Н. А. Олифер. – 6-е изд. – СПб.: Питер, 2023. – 1008 с.

6. Flask Documentation // Pallets Projects [Электронный ресурс]. URL: https://flask.palletsprojects.com/ (дата обращения: 19.05.2026).

7. Flask-Login Documentation // Read the Docs [Электронный ресурс]. URL: https://flask-login.readthedocs.io/ (дата обращения: 19.05.2026).

8. PostgreSQL Documentation // PostgreSQL [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 19.05.2026).

9. SQLite Documentation // SQLite [Электронный ресурс]. URL: https://www.sqlite.org/docs.html (дата обращения: 20.05.2026).

10. Федеральный закон РФ от 27.07.2006 № 152-ФЗ «О персональных данных». // Официальный интернет-портал правовой информации [Электронный ресурс]. URL: http://pravo.gov.ru/ (дата обращения: 19.05.2026).

11. Astra Linux Wiki (документация и руководства) // Astra Linux [Электронный ресурс]. URL: https://wiki.astralinux.ru/ (дата обращения: 19.05.2026).

12. Apache JMeter User’s Manual // Apache JMeter [Электронный ресурс]. URL: https://jmeter.apache.org/usermanual/ (дата обращения: 19.05.2026).

13. Федеральный закон РФ от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации». // Официальный интернет-портал правовой информации [Электронный ресурс]. URL: http://pravo.gov.ru/ (дата обращения: 19.05.2026).

14. Python 3 Documentation // Python.org [Электронный ресурс]. URL: https://docs.python.org/3/ (дата обращения: 20.05.2026).

15. Рамальо, Л. Python. Исчерпывающее руководство / Л. Рамальо; пер. с англ. А. Д. Васильев. – СПб.: Питер, 2016. – 800 с.

16. MDN Web Docs (HTML) // Mozilla [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/HTML (дата обращения: 20.05.2026).

17. MDN Web Docs (CSS) // Mozilla [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата обращения: 20.05.2026).

18. RFC 9110. HTTP Semantics // RFC Editor [Электронный ресурс]. URL: https://www.rfc-editor.org/rfc/rfc9110.html (дата обращения: 20.05.2026).

19. PEP 3333 – Python Web Server Gateway Interface v1.0.1 // Python.org [Электронный ресурс]. URL: https://www.python.org/dev/peps/pep-3333/ (дата обращения: 20.05.2026).

20. Федеральный закон РФ от 26.07.2017 № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации». // Официальный интернет-портал правовой информации [Электронный ресурс]. URL: http://pravo.gov.ru/ (дата обращения: 20.05.2026).

21. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения. – М.: ФСТЭК России, 1992.

22. Доктрина информационной безопасности Российской Федерации (Утверждена Указом Президента Российской Федерации от 5 декабря 2016 г. № 646). // Официальный интернет-портал правовой информации [Электронный ресурс]. URL: http://pravo.gov.ru/ (дата обращения: 20.05.2026).

Приложение А
Программный код – клиентская часть
(обязательное)

Листинг А.1 – Файл admin.html

```
{% extends "base.html" %}

{% block content %}
<h1>Панель администратора</h1>
<p>Мониторинг состояния IT-инфраструктуры и сервиса.</p>
<hr>

<div class="stats-grid">
    <div class="stat-card">
        <div>Загрузка ЦП</div>
        <div class="stat-value"><span id="cpuVal">{{ stats.cpu
}}</span>%</div>
    </div>

    <div class="stat-card">
        <div>Оперативная память</div>
        <div class="stat-value"><span id="ramVal">{{
stats.ram_percent }}</span>%</div>
    </div>

    <div class="stat-card">
        <div>Свободно на диске</div>
        <div class="stat-value"><span id="diskVal">{{
stats.disk_free_gb }}</span> GB</div>
    </div>

    <div class="stat-card">
        <div>Файлов в базе</div>
        <div class="stat-value"><span id="filesVal">{{
stats.files_count }}</span></div>
    </div>

    <div class="stat-card">
        <div>Всего скачиваний</div>
        <div class="stat-value"><span id="downloadsVal">{{
stats.downloads_count }}</span></div>
    </div>
</div>

<div style="margin-top: 20px; padding: 12px; border-radius: 6px;
border:1px solid #ddd;">
  <strong>Статус сервера:</strong>
  <span id="healthBadge" style="padding:4px 10px; border-radius:
12px; background:#27ae60; color:#fff;">OK</span>
  <span id="healthIssues" style="margin-left:10px;
color:#7f8c8d;"></span>
</div>

<hr style="margin: 40px 0;">
<h2>Журнал (аудит)</h2>
```

```
<table style="width:100%; border-collapse: collapse;">
  <thead>
    <tr style="background:#ecf0f1;">
      <th style="padding:8px; border:1px solid #ddd;">Время
(UTC)</th>
      <th style="padding:8px; border:1px solid
#ddd;">Пользователь</th>
      <th style="padding:8px; border:1px solid #ddd;">Файл</th>
      <th style="padding:8px; border:1px solid #ddd;">IP</th>
      <th style="padding:8px; border:1px solid
#ddd;">Результат</th>
      <th style="padding:8px; border:1px solid
#ddd;">Действие</th>
    </tr>
  </thead>

  <tbody id="logsBody">
    {% for row in logs %}
    <tr>
      <td style="padding:8px; border:1px solid #ddd;">{{ row.ts
}}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
row.username }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
row.filename or "-" }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{ row.ip or
"-" }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
row.result }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
row.action or "-" }}</td>
    </tr>
    {% endfor %}
    {% if not last_logs %}
    <tr>
      <td colspan="5" style="padding:10px; border:1px solid #ddd;
color:#7f8c8d;">
        Событий аудита пока нет.
      </td>
    </tr>
    {% endif %}
  </tbody>
</table>

<hr style="margin: 40px 0;">

<hr style="margin: 40px 0;">

<h2> Обслуживание хранилища</h2>
<div style="background: white; padding: 20px; border-radius: 8px;
box-shadow: 0 2px 4px rgba(0,0,0,0.1); display: flex; justify-
content: space-between; align-items: center;">
```

```
    <div>
        <h3>Очистка устаревших данных</h3>
        <p style="color: #7f8c8d; margin: 0;">Удаляет все
зашифрованные файлы.</p>
    </div>

    <!-- Форма с кнопкой -->
    <form action="{{ url_for('main.admin_cleanup') }}"
method="post">
        <button type="submit" class="btn" style="background-color:
#e74c3c; font-weight: bold;">
            Очистить диск
        </button>
    </form>
</div>

<h2> Управление пользователями</h2>
<div style="background: white; padding: 20px; border-radius: 8px;
box-shadow: 0 2px 4px rgba(0,0,0,0.1);">
    <h3>Создать нового сотрудника</h3>
    <form action="{{ url_for('main.create_user') }}" method="post"
style="display: flex; gap: 10px; align-items: flex-end;">
        <div>
            <label>Логин:</label><br>
            <input type="text" name="username" required
placeholder="ivanov" style="padding: 8px;">
        </div>
        <div>
            <label>Пароль:</label><br>
            <input type="text" name="password" required
placeholder="secret123" style="padding: 8px;">
        </div>
        <button type="submit" class="btn" style="background-color:
#3498db;">Добавить</button>
    </form>
</div>

<div style="margin-top: 30px; padding: 20px; background: #fff3cd;
border: 1px solid #ffeeba; border-radius: 5px;">
    <strong>Статус безопасности:</strong> Шифрование активно.
Угроз не обнаружено.
</div>

<div style="margin: 10px 0 20px 0;">
  <a class="btn" href="{{ url_for('main.admin_dashboard')
}}">Мониторинг</a>
  <a class="btn" href="{{ url_for('main.admin_audit') }}"
style="margin-left:10px;">Аудит</a>
</div>

<script>
async function refreshStats() {
```

```
  try {
    const resp = await fetch("{{ url_for('main.api_admin_stats')
}}", { cache: "no-store" });
    if (!resp.ok) return;

    const data = await resp.json();

    document.getElementById("cpuVal").textContent = data.cpu;
    document.getElementById("ramVal").textContent =
data.ram_percent;
    document.getElementById("diskVal").textContent =
data.disk_free_gb;
    document.getElementById("filesVal").textContent =
data.files_count;
    document.getElementById("downloadsVal").textContent =
data.downloads_count;
    const u = document.getElementById("usersVal");
    if (u) u.textContent = data.users_count;

    const badge = document.getElementById("healthBadge");
    const issues = document.getElementById("healthIssues");
    issues.textContent = (data.issues && data.issues.length) ?
("(" + data.issues.join(", ") + ")") : "";

    if (data.health === "OK") {
      badge.textContent = "OK";
      badge.style.background = "#27ae60";
    } else if (data.health === "WARN") {
      badge.textContent = "WARN";
      badge.style.background = "#f39c12";
    } else {
      badge.textContent = "CRIT";
      badge.style.background = "#e74c3c";
    }
  } catch (e) {
    //при сбое просто не обновляем
  }
}

refreshStats();
setInterval(refreshStats, 2000); // 2 секунды (можно 5 сек)
</script>

<script>
async function refreshLogs() {
  try {
    const resp = await fetch("{{ url_for('main.api_admin_logs')
}}", { cache: "no-store" });
    if (!resp.ok) return;
    const rows = await resp.json();

    const tbody = document.getElementById("logsBody");
```

```
    if (!tbody) return;

    tbody.innerHTML = rows.map(r => `
      <tr>
        <td style="padding:8px; border:1px solid
#ddd;">${r.ts}</td>
        <td style="padding:8px; border:1px solid
#ddd;">${r.username || "-"}</td>
        <td style="padding:8px; border:1px solid
#ddd;">${r.filename || "-"}</td>
        <td style="padding:8px; border:1px solid #ddd;">${r.ip ||
"-"}</td>
        <td style="padding:8px; border:1px solid
#ddd;">${r.result}</td>
        <td style="padding:8px; border:1px solid #ddd;">${r.action
|| "-"}</td>
      </tr>
    `).join("");
  } catch (e) {}
}

//refreshLogs(); если что потом включить
setInterval(refreshLogs, 2000);
</script>
{% endblock %}
```

Листинг А.2 – Файл admin_audit.html

```
{% extends "base.html" %}
{% block content %}

<h1>Администрирование — Аудит</h1>
<p>Журнал операций (upload/download/ошибки) для расследования
инцидентов.</p>

<div style="margin: 10px 0 20px 0;">
  <a class="btn" href="{{ url_for('main.admin_dashboard')
}}">Мониторинг</a>
  <a class="btn" href="{{ url_for('main.admin_audit') }}"
style="margin-left:10px;">Аудит</a>
</div>

<table style="width:100%; border-collapse: collapse;">
  <thead>
    <tr style="background:#ecf0f1;">
      <th style="padding:8px; border:1px solid #ddd;">Время
(UTC)</th>
      <th style="padding:8px; border:1px solid
#ddd;">Действие</th>
      <th style="padding:8px; border:1px solid
#ddd;">Пользователь</th>
      <th style="padding:8px; border:1px solid #ddd;">Файл</th>
      <th style="padding:8px; border:1px solid #ddd;">IP</th>
```

```
      <th style="padding:8px; border:1px solid
#ddd;">Результат</th>
      <th style="padding:8px; border:1px solid #ddd;">Ошибка</th>
    </tr>
  </thead>
  <tbody>
    {% for r in logs %}
    <tr>
      <td style="padding:8px; border:1px solid #ddd;">{{ r.ts
}}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{ r.action
}}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
r.username }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{
r.filename or "-" }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{ r.ip or
"-" }}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{ r.result
}}</td>
      <td style="padding:8px; border:1px solid #ddd;">{{ r.error
or "-" }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

{% endblock %}
```

Листинг А.3 – Файл base.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>SecureShare | Защищенный обмен</title>
    <link rel="icon" type="image/png" href="{{ url_for('static',
filename='img/favicon_3.png') }}">
    <style>
        body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif; background-color: #f4f6f9; margin: 0; }
        .navbar { background-color: #2c3e50; color: white;
padding: 1rem; }
        .navbar a { color: white; text-decoration: none; margin-
right: 20px; font-weight: bold; }
        .container { max_width: 800px; margin: 50px auto;
background: white; padding: 40px; border-radius: 8px; box-shadow:
0 4px 6px rgba(0,0,0,0.1); }
        .btn { background-color: #27ae60; color: white; padding:
10px 20px; border: none; border-radius: 4px; cursor: pointer;
text-decoration: none; display: inline-block;}
        .btn:hover { background-color: #219150; }
```

```
        .alert { padding: 10px; background-color: #e74c3c; color:
white; margin-bottom: 20px; border-radius: 4px; }
        h1 { color: #2c3e50; }
        input[type=file] { margin-bottom: 20px; }
        .stats-grid { display: grid; grid-template-columns: 1fr
1fr; gap: 20px; }
        .stat-card { background: #ecf0f1; padding: 20px; border-
radius: 8px; text-align: center; }
        .stat-value { font-size: 2em; font-weight: bold; color:
#2980b9; }
        footer { text-align: center; margin-top: 50px; color:
#7f8c8d; font-size: 0.9em; }
        .share-card {
          margin-top: 20px;
          padding: 18px 22px;
          background: #ffffff;
          border: 1px solid #e6e8eb;
          border-radius: 10px;
        }

        .share-form .share-row {
          display: flex;
          gap: 14px;
          align-items: end;
          flex-wrap: wrap;
        }

        .share-field {
          display: flex;
          flex-direction: column;
          min-width: 220px;
        }

        .share-field label {
          font-size: 0.95em;
          color: #2c3e50;
          margin-bottom: 6px;
        }

        .share-input {
          padding: 9px 10px;
          border: 1px solid #ccd2d8;
          border-radius: 7px;
          font-size: 1em;
        }

        .share-btn-wrap {
          min-width: 140px;
        }

        .share-hint {
          margin: 12px 0 0 0;
```

```
          color: #7f8c8d;
          font-size: 0.95em;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <span style="font-size: 1.2em; margin-right: 40px;">
SecureShare</span>
        {% if current_user.is_authenticated %}
            <a href="{{ url_for('main.index') }}">Загрузить</a>
            {% if current_user.is_admin %}
                <a href="{{ url_for('main.admin_dashboard')
}}">Администрирование</a>
            {% endif %}
            <a href="{{ url_for('main.logout') }}" style="float:
right;">Выход ({{ current_user.username }})</a>
        {% else %}
            <a href="{{ url_for('main.login') }}">Вход</a>
        {% endif %}
        <a href="{{ url_for('main.inbox') }}">Входящие ({{
inbox_count }})</a>
        <!-- где-нибудь в навигации -->
        <a href="{{ url_for('main.shares_outbox') }}">Мои
отправленные</a>
    </div>

    <div class="container">
        <!-- Блок для сообщений об ошибках -->
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                {% for message in messages %}
                    <div class="alert">{{ message }}</div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        {% block content %}{% endblock %}
    </div>

    <footer>
        &copy; 2026 SecureShare. Разработано для защиты данных.
    </footer>
</body>
</html>
```

Листинг А.4 – Файл inbox.html

```
{% extends "base.html" %}
{% block content %}
<h1>Входящие файлы</h1>
<hr>

{% if not shares %}
```

```
  <p>Нет входящих файлов.</p>
{% else %}
  <table style="width:100%; border-collapse: collapse;">
    <thead>
      <tr style="background:#ecf0f1;">
        <th style="padding:8px; border:1px solid #ddd;">Файл</th>
        <th style="padding:8px; border:1px solid #ddd;">Срок</th>
        <th style="padding:8px; border:1px solid
#ddd;">Осталось</th>
        <th style="padding:8px; border:1px solid
#ddd;">Действие</th>
      </tr>
    </thead>
    <tbody>
      {% for s in shares %}
      <tr>
        <td style="padding:8px; border:1px solid #ddd;">{{
s.filename }}</td>
        <td style="padding:8px; border:1px solid #ddd;">
          {{ s.expires_at if s.expires_at else "∞" }}
        </td>
        <td style="padding:8px; border:1px solid #ddd;">
          {{ s.max_downloads - s.downloads_done }}
        </td>
        <td style="padding:8px; border:1px solid #ddd;">
          <a class="btn" href="{{ url_for('main.download_shared',
share_token=s.share_token) }}">Скачать</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
{% endif %}
{% endblock %}
```

Листинг А.5 – Файл index.html

```
{% extends "base.html" %}

{% block content %}
<h1>Безопасная передача файлов</h1>
<p>Файлы шифруются алгоритмом Fernet (AES-128-CBC + HMAC-SHA256)
перед записью на диск. Доступ только по уникальной ссылке.</p>
<hr>

<form method="post" enctype="multipart/form-data">
    <div style="margin: 30px 0; border: 2px dashed #bdc3c7;
padding: 40px; text-align: center;">
        <label for="file">Выберите файл для
шифрования:</label><br><br>
        <input type="file" name="file" id="file" required>
    </div>

    <div style="text-align: center;">
```

```
        <button type="submit" class="btn" style="font-size:
1.2em;"> Зашифровать и Загрузить</button>
    </div>
</form>
{% endblock %}
```

Листинг А.6 – Файл login.html

```
{% extends "base.html" %}

{% block content %}
<div style="max-width: 400px; margin: 0 auto; text-align:
center;">
    <h1>Вход в систему</h1>
    <form action="" method="post">
        <div style="margin-bottom: 15px;">
            <input type="text" name="username" placeholder="Логин"
required
                   style="padding: 10px; width: 100%; font-size:
1.1em;">
        </div>
        <div style="margin-bottom: 15px;">
            <input type="password" name="password"
placeholder="Пароль" required
                   style="padding: 10px; width: 100%; font-size:
1.1em;">
        </div>
        <button type="submit" class="btn" style="width:
100%;">Войти</button>
    </form>
</div>
{% endblock %}
```

Листинг А.7 – Файл shares_outbox.html

```
<!-- templates/shares_outbox.html -->
{% extends "base.html" %}
{% block content %}

<style>
  :root {
    --text: #1f2937;
    --muted: #6b7280;
    --bg: #ffffff;
    --border: #e5e7eb;
    --danger: #e11d48;
    --danger-600: #be123c;
    --success-ink: #065f46;
    --success-bg: #ecfdf5;
    --success-br: #a7f3d0;
    --warn-ink: #92400e;
    --warn-bg: #fffbeb;
    --warn-br: #fcd34d;
    --rev-ink: #7f1d1d;
    --rev-bg: #fef2f2;
```

```
  --rev-br: #fecaca;
}

h2 {
  margin: 6px 0 14px;
  color: var(--text);
  font-weight: 800;
  letter-spacing: .2px;
}

.table-wrap {
  overflow-x: auto;
  background: var(--bg);
  border: 1px solid var(--border);
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(17,24,39,.05);
}

table.outbox {
  width: 100%;
  border-collapse: collapse;
  table-layout: fixed;
  font-size: 14px;
}

.outbox th, .outbox td {
  padding: 10px 12px;
  border-bottom: 1px solid var(--border);
  vertical-align: top;
}

.outbox thead th {
  background: #f3f4f6;
  text-align: left;
  font-weight: 700;
  color: #111827;
}

.outbox tbody tr:nth-child(even) td {
  background: #fcfcfd;
}

.outbox tbody tr:hover td {
  background: #f9fafb;
}

.col-id { width: 72px; }
.col-file { width: 28%; word-break: break-word; }
.col-recipient { width: 18%; word-break: break-word; }
.col-done, .col-limit { width: 92px; text-align: right; }
.col-exp { width: 180px; color: var(--muted); }
.col-status { width: 120px; }
```

```
  .col-actions { width: 140px; text-align: right; white-space:
nowrap; }

  /* Бейджи статусов */
  .badge {
    display: inline-block;
    padding: 2px 10px;
    border-radius: 999px;
    font-weight: 700;
    font-size: 12px;
    line-height: 1.8;
    border: 1px solid transparent;
  }
  .badge--active  { color: var(--success-ink); background: var(--
success-bg); border-color: var(--success-br); }
  .badge--expired { color: var(--warn-ink);    background: var(--
warn-bg);    border-color: var(--warn-br); }
  .badge--revoked { color: var(--rev-ink);     background: var(--
rev-bg);     border-color: var(--rev-br); }

  /* Кнопки */
  .btn {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    gap: 8px;
    padding: 8px 12px;
    font-weight: 700;
    border-radius: 8px;
    border: 1px solid transparent;
    cursor: pointer;
    transition: transform .05s, background .15s, box-shadow .15s,
border-color .15s;
  }
  .btn:active { transform: translateY(1px); }
  .btn-danger {
    background: var(--danger);
    border-color: var(--danger);
    color: #fff;
    box-shadow: 0 1px 0 rgba(0,0,0,.05), 0 4px 10px -4px
rgba(225,17,72,.35);
  }
  .btn-danger:hover {
    background: var(--danger-600);
    border-color: var(--danger-600);
  }

  .muted { color: var(--muted); }
</style>

<h2>Мои отправленные</h2>
```

```
<div class="table-wrap">
  <table class="outbox">
    <thead>
      <tr>
        <th class="col-id">ID</th>
        <th class="col-file">Файл</th>
        <th class="col-recipient">Получатель</th>
        <th class="col-done">Скачано</th>
        <th class="col-limit">Лимит</th>
        <th class="col-exp">Истекает</th>
        <th class="col-status">Статус</th>
        <th class="col-actions"></th>
      </tr>
    </thead>
    <tbody>
      {% for s in shares %}
      <tr>
        <td class="col-id">{{ s.id }}</td>
        <td class="col-file" title="{{ s.filename }}">{{
s.filename }}</td>
        <td class="col-recipient">{{ s.recipient.username if
s.recipient else s.recipient_id }}</td>
        <td class="col-done">{{ s.downloads_done }}</td>
        <td class="col-limit">{{ s.max_downloads }}</td>
        <td class="col-exp">{{ s.expires_at or "—" }}</td>
        <td class="col-status">
          {% if s.revoked %}
            <span class="badge badge--revoked">отозвана</span>
          {% elif s.expires_at and s.expires_at < utcnow() %}
            <span class="badge badge--expired">просрочена</span>
          {% else %}
            <span class="badge badge--active">активна</span>
          {% endif %}
        </td>
        <td class="col-actions">
          {% if not s.revoked %}
          <form method="post"
                action="{{ url_for('main.revoke_share',
share_id=s.id) }}"
                onsubmit="return confirm('Отозвать передачу?');"
                style="display:inline;">
            <button type="submit" class="btn btn-
danger">Отозвать</button>
          </form>
          {% else %}
            <span class="muted">—</span>
          {% endif %}
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
```

```
</div>

{% endblock %}
```

Листинг А.8 – Файл success.html

```
{% extends "base.html" %}

{% block content %}
<div style="text-align: center;">
    <h1 style="color: #27ae60;">Файл успешно загружен!</h1>
    <p>Файл <strong>{{ filename }}</strong> зашифрован и
сохранен.</p>

    <div style="background: #f8f9fa; padding: 15px; border: 1px
solid #ddd; margin: 20px 0; word-break: break-all;">
        {{ link }}
    </div>

    <p>Скопируйте эту ссылку и передайте получателю через
защищенный канал.</p>

    <a href="{{ url_for('main.index') }}" class="btn">Загрузить
еще</a>

    <!-- разделитель для тестирования итд -->
</div>
<hr style="margin: 35px 0;">

<div class="share-card">
  <h2 style="margin-top:0;">Отправить файл сотруднику (внутри
системы)</h2>

  <form method="post" action="{{ url_for('main.create_share') }}"
class="share-form">
    <input type="hidden" name="file_token" value="{{ token }}">

    <div class="share-row">
      <div class="share-field">
        <label>Получатель</label>
        <select name="recipient_id" required class="share-input">
          <option value="">— выбрать —</option>
          {% for u in users %}
            <option value="{{ u.id }}">{{ u.username }}</option>
          {% endfor %}
        </select>
      </div>

      <div class="share-field">
        <label>Срок действия</label>
        <select name="ttl_minutes" class="share-input">
          <option value="1">1 минута (тест)</option>
          <option value="60">1 час</option>
          <option value="1440" selected>24 часа</option>
```

```
            <option value="10080">7 дней</option>
            <option value="0">бессрочно</option>
          </select>
        </div>

        <div class="share-field">
          <label>Скачиваний</label>
          <select name="max_downloads" class="share-input">
            <option value="1" selected>1</option>
            <option value="3">3</option>
            <option value="9999">без ограничений</option>
          </select>
        </div>

        <div class="share-field share-btn-wrap">
          <button type="submit" class="btn">Отправить</button>
        </div>
      </div>
    </form>

    <p class="share-hint">
      Файл появится у получателя в разделе «Входящие». Передача не
использует сторонние сервисы.
    </p>
</div>
{% endblock %}
```

Приложение Б

Программный код – серверная часть

(обязательное)

Листинг Б.1 – Файл crypto.py

```
# app/utils/crypto.py
from cryptography.fernet import Fernet, InvalidToken
import os

DEFAULT_KEY = b'Z7wz89_P4s0yJ1kL2mN3oP5qR6sT8uV9wX0yZ1aB2cD='

def _load_keys():
    raw = os.getenv("APP_FERNET_KEYS", "").strip()
    if not raw:
        return [Fernet(DEFAULT_KEY)]
    keys = []
    for k in [s.strip() for s in raw.split(",") if s.strip()]:
        keys.append(Fernet(k.encode() if not isinstance(k, (bytes,
bytearray)) else k))
    return keys

FERNETS = _load_keys()

def encrypt_data(data: bytes) -> bytes:
    return FERNETS[0].encrypt(data)

def decrypt_data(data: bytes) -> bytes:
    last_err = None
    for f in FERNETS:
        try:
            return f.decrypt(data)
        except Exception as e:
            last_err = e
            continue
    raise last_err or InvalidToken()

def generate_key():
    return Fernet.generate_key()
```

Листинг Б.2 – Файл __init__.py

```
# app/__init__.py
import os
import logging
import threading
import time
import hashlib
from pathlib import Path
from datetime import datetime, timedelta
from flask import Flask
from flask_login import LoginManager
from logging.handlers import RotatingFileHandler
from config import Config
from app.extensions import db

def create_app():
    app = Flask(__name__)
```

```
    app.config.from_object(Config)

    data_dir = Path(app.config.get("DATA_DIR", "data"))
    data_dir.mkdir(parents=True, exist_ok=True)
    upload_dir = Path(app.config.get("UPLOAD_FOLDER", str(data_dir
/ "uploads")))
    upload_dir.mkdir(parents=True, exist_ok=True)
    app.config["DATA_DIR"] = str(data_dir)
    app.config["UPLOAD_FOLDER"] = str(upload_dir)

    log_dir = data_dir / "logs"
    log_dir.mkdir(parents=True, exist_ok=True)
    log_file = log_dir / "secure_share_audit.log"

    handler = RotatingFileHandler(
        log_file,
        maxBytes=2_000_000,
        backupCount=5,
        encoding="utf-8"
    )
    formatter = logging.Formatter("%(asctime)s | %(levelname)s |
%(message)s")
    handler.setFormatter(formatter)

    class AuditChainFilter(logging.Filter):
        def __init__(self, anchor_path: Path):
            super().__init__()
            self.anchor_path = anchor_path
            try:
                self.prev =
self.anchor_path.read_text(encoding="utf-8").strip()
            except Exception:
                self.prev = ""

        def filter(self, record: logging.LogRecord) -> bool:
            msg = str(record.getMessage())
            if "AUDIT " in msg:
                base = (self.prev or "").encode() + msg.encode()
                h = hashlib.sha256(base).hexdigest()
                record.msg = f"{msg} h={h}"
                self.prev = h
                try:
                    self.anchor_path.write_text(self.prev,
encoding="utf-8")
                except Exception:
                    pass
            return True

    handler.addFilter(AuditChainFilter(log_dir /
"audit_anchor.txt"))
    if not any(isinstance(h, RotatingFileHandler) for h in
app.logger.handlers):
```

```
        app.logger.setLevel(logging.INFO)
        app.logger.addHandler(handler)

    db.init_app(app)

    login = LoginManager(app)
    login.login_view = 'main.login'
    login.login_message = 'Пожалуйста, войдите в систему.'

    from app.models import User

    @login.user_loader
    def load_user(user_id):
        try:
            return User.query.get(int(user_id))
        except Exception:
            return None

    @app.after_request
    def set_security_headers(resp):
        resp.headers["Strict-Transport-Security"] = "max-
age=31536000; includeSubDomains; preload"
        resp.headers["X-Content-Type-Options"] = "nosniff"
        resp.headers["X-Frame-Options"] = "DENY"
        resp.headers["Referrer-Policy"] = "strict-origin-when-
cross-origin"
        resp.headers["Permissions-Policy"] = "geolocation=(),
microphone=(), camera=()"
        resp.headers["Content-Security-Policy"] = (
            "default-src 'self'; "
            "img-src 'self' data:; "
            "font-src 'self' data:; "
            "style-src 'self' 'unsafe-inline'; "
            "script-src 'self' 'unsafe-inline'; "
            "base-uri 'none'; "
            "frame-ancestors 'none'"
        )
        return resp
    def start_cleanup_thread(app_):
        EXP_SEC = int(os.getenv("FILE_TTL_SEC", "604800"))
        PERIOD = int(os.getenv("CLEANUP_PERIOD_SEC", "900"))
        if EXP_SEC <= 0 or PERIOD <= 0:
            return

        def worker():
            from app.extensions import db as _db
            from app.models import FileRecord

            while True:
                try:
                    with app_.app_context():
```

```
                        cutoff = datetime.utcnow() -
timedelta(seconds=EXP_SEC)
                        old =
FileRecord.query.filter(FileRecord.upload_date < cutoff).all()
                        if old:
                            for fr in old:
                                p =
Path(app_.config['UPLOAD_FOLDER']) / fr.secure_name
                                try:
                                    if p.exists():
                                        p.unlink()
                                except Exception:
                                    pass
                                _db.session.delete(fr)
                            try:
                                _db.session.commit()
                                app_.logger.info(f"AUDIT
action=cleanup result=success count={len(old)}")
                            except Exception:
                                _db.session.rollback()
                except Exception:
                    pass

                time.sleep(PERIOD)

        t = threading.Thread(target=worker, daemon=True)
        t.start()
    with app.app_context():
        from app import routes, models  # noqa: F401
        app.register_blueprint(routes.bp)
        db.create_all()

        # Создаём администратора, если его ещё нет
        if not User.query.filter_by(username='admin').first():
            admin = User(username='admin', is_admin=True)
            admin.set_password('admin123')
            db.session.add(admin)
            db.session.commit()

    start_cleanup_thread(app)

    @app.context_processor
    def inject_helpers():
        from datetime import datetime
        return {"utcnow": lambda: datetime.utcnow()}

    return app
```

Листинг Б.2 – Файл extensions.py

```
from flask_sqlalchemy import SQLAlchemy
from flask_apscheduler import APScheduler

# База данных
```

```
db = SQLAlchemy()

# Планировщик задач
scheduler = APScheduler()
```

Листинг Б.3 – Файл models.py

```
from app.extensions import db
from datetime import datetime, timedelta
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,
check_password_hash
import uuid
import secrets

class User(UserMixin, db.Model):
    """
    Модель пользователя системы.
    Наследуется от UserMixin, чтобы Flask-Login умел с ней
работать.
    """
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    is_admin = db.Column(db.Boolean, default=False)  # Флаг: админ
или нет

    def set_password(self, password):
        """Создает безопасный хеш пароля"""
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        """Проверяет пароль по хешу"""
        return check_password_hash(self.password_hash, password)

    def __repr__(self):
        return f'<User {self.username}>'


class FileRecord(db.Model):
    """
    Модель для хранения метаданных о файлах.
    """
    id = db.Column(db.Integer, primary_key=True)

    # default=uuid.uuid4 генерирует уникальный код автоматически
при создании записи
    token = db.Column(db.String(36), unique=True, index=True,
default=lambda: str(uuid.uuid4()))

    filename = db.Column(db.String(128), index=True)
    secure_name = db.Column(db.String(128), unique=True)
    upload_date = db.Column(db.DateTime, default=datetime.utcnow)
    download_count = db.Column(db.Integer, default=0)
```

```
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

class DownloadLog(db.Model):
    __tablename__ = "download_logs"

    id = db.Column(db.Integer, primary_key=True)

    ts = db.Column(db.DateTime, default=datetime.utcnow,
index=True)

    file_token = db.Column(db.String(128), index=True,
nullable=True)
    filename = db.Column(db.String(256), nullable=True)
    username = db.Column(db.String(64), nullable=False,
default="anonymous")
    ip = db.Column(db.String(64), nullable=True)
    user_agent = db.Column(db.String(256), nullable=True)
    result = db.Column(db.String(32), nullable=False,
default="success")
    error = db.Column(db.String(256), nullable=True)
    action = db.Column(db.String(16), nullable=False,
default="download")

def generate_share_token():
    return secrets.token_urlsafe(18)

class FileShare(db.Model):
    is_new = db.Column(db.Boolean, default=True, nullable=False)
    __tablename__ = "file_shares"

    id = db.Column(db.Integer, primary_key=True)

    share_token = db.Column(db.String(128), unique=True,
index=True, default=generate_share_token)

    file_token = db.Column(db.String(128), index=True,
nullable=False)
    filename = db.Column(db.String(256), nullable=True)

    sender_id = db.Column(db.Integer, nullable=False)
    #recipient_id = db.Column(db.Integer, nullable=False)

    created_at = db.Column(db.DateTime, default=datetime.utcnow,
index=True)
    expires_at = db.Column(db.DateTime, nullable=True)  # можно
None = бессрочно

    max_downloads = db.Column(db.Integer, default=1)
    downloads_done = db.Column(db.Integer, default=0)

    revoked = db.Column(db.Boolean, default=False)
```

```
    recipient_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
    recipient = db.relationship('User',
foreign_keys=[recipient_id], lazy='selectin')
```

Листинг Б.4 – Файл routes.py

```
import os
import io
import psutil  # Для мониторинга системы
#from datetime import datetime
from werkzeug.utils import secure_filename
from flask_login import current_user, login_user, logout_user,
login_required
from flask import Blueprint, render_template, request, redirect,
url_for, send_file, current_app, flash, jsonify
from app.models import FileRecord, User, DownloadLog, FileShare
from datetime import datetime, timedelta

from app.extensions import db
from app.models import FileRecord, User
from app.utils.crypto import encrypt_data, decrypt_data
from flask import abort
from app.models import FileRecord, User, DownloadLog

bp = Blueprint('main', __name__)

import time
import secrets
from collections import deque, defaultdict
from itsdangerous import URLSafeSerializer, BadSignature
import mimetypes

APP_SECRET = os.getenv("SECRET_KEY", "secret-key-123")  # уже есть
в Config, но дублировать безопасно
TOKEN_SALT = "download-link"
def _ser() -> URLSafeSerializer:
    return URLSafeSerializer(APP_SECRET, salt=TOKEN_SALT)

def make_download_token(file_id: str, ttl_seconds: int) -> str:
    payload = {"fid": file_id, "exp": int(time.time()) +
int(ttl_seconds), "jti": secrets.token_urlsafe(8)}
    return _ser().dumps(payload)

def verify_download_token(token: str, expected_fid: str) ->
tuple[bool, str]:
    try:
        p = _ser().loads(token)
    except BadSignature:
        return False, "bad_signature"
    if p.get("fid") != expected_fid:
        return False, "fid_mismatch"
    if time.time() > p.get("exp", 0):
```

```
        return False, "expired"
    return True, "ok"

BUCKETS = defaultdict(deque)  # key -> deque[timestamps]
def allow_rate(key: str, limit: int, window_sec: int) -> bool:
    now = time.time()
    dq = BUCKETS[key]
    # очистка старых записей
    while dq and dq[0] <= now - window_sec:
        dq.popleft()
    if len(dq) >= limit:
        return False
    dq.append(now)
    return True

LOGIN_LIMIT = int(os.getenv("RATE_LOGIN_LIMIT", "5"))
LOGIN_WINDOW = int(os.getenv("RATE_LOGIN_WINDOW", "60"))      # 5
в минуту
DL_LIMIT = int(os.getenv("RATE_DL_LIMIT", "200"))
DL_WINDOW = int(os.getenv("RATE_DL_WINDOW", "3600"))          #
200 в час

ALLOWED_EXTS = set((os.getenv("ALLOWED_EXTS") or
"pdf,png,jpg,jpeg,txt,docx,xlsx,csv,zip").split(","))
def ext_allowed(filename: str) -> bool:
    ext = filename.rsplit(".", 1)[-1].lower() if "." in filename
else ""
    return ext in ALLOWED_EXTS

@bp.route('/login', methods=['GET', 'POST'])
def login():
    """
    Страница входа. Добавлен rate limiting по IP.
    """
    if current_user.is_authenticated:
        return redirect(url_for('main.index'))

    if request.method == 'POST':
        ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
        if not allow_rate(f"login:{ip}", LOGIN_LIMIT,
LOGIN_WINDOW):
            flash("Слишком много попыток входа. Подождите минуту и
попробуйте снова.")
            current_app.logger.info(f"AUDIT action=login user=-
ip={ip} result=rate_limited")
            return redirect(url_for('main.login'))

        username = request.form.get('username', '')
        password = request.form.get('password', '')

        user = User.query.filter_by(username=username).first()
```

```
        if user is None or not user.check_password(password):
            flash('Ошибка: Неверное имя пользователя или пароль')
            current_app.logger.info(f"AUDIT action=login
user={username} ip={ip} result=fail")
            return redirect(url_for('main.login'))

        login_user(user)
        current_app.logger.info(f"AUDIT action=login
user={username} ip={ip} result=success")
        return redirect(url_for('main.index'))

    return render_template('login.html')

@bp.route('/logout')
def logout():
    """Выход из системы"""
    logout_user()
    return redirect(url_for('main.login'))

@bp.route('/', methods=['GET', 'POST'])
@login_required
def index():
    """
    Главная: загрузка файла с проверкой расширения.
    """
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('Нет файла')
            return redirect(request.url)

        file = request.files['file']
        if file.filename == '':
            flash('Файл не выбран')
            return redirect(request.url)

        if file:
            filename = secure_filename(file.filename)

            # Политика загрузки по расширениям
            if not ext_allowed(filename):
                flash('Запрещённый тип файла. Разрешены: ' + ",
".join(sorted(ALLOWED_EXTS)))
                return redirect(request.url)

            # Шифрование и сохранение
            file_data = file.read()
            encrypted_data = encrypt_data(file_data)

            storage_name =
f"{datetime.now().timestamp()}_{filename}.enc"
            save_path =
os.path.join(current_app.config['UPLOAD_FOLDER'], storage_name)
```

```
            with open(save_path, 'wb') as f:
                f.write(encrypted_data)

            # Запись в БД
            new_file = FileRecord(
                filename=filename,
                secure_name=storage_name,
                upload_date=datetime.now(),
                user_id=current_user.id
            )
            db.session.add(new_file)
            db.session.flush()  # чтобы был заполнен token до
commit

            ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
            ua = request.headers.get("User-Agent", "")[:250]
            db.session.add(DownloadLog(
                action="upload",
                file_token=new_file.token,
                filename=new_file.filename,
                username=current_user.username,
                ip=ip,
                user_agent=ua,
                result="success",
                error=None
            ))
            current_app.logger.info(
                f"AUDIT action=upload user={current_user.username}
token={new_file.token} file={new_file.filename} ip={ip}
result=success"
            )

            db.session.commit()
            return redirect(url_for('main.success',
token=new_file.token))

    return render_template('index.html')

@bp.route('/success/<token>')
@login_required
def success(token):
    file_record =
FileRecord.query.filter_by(token=token).first_or_404()

    ttl_seconds = int(os.getenv("LINK_TTL_SEC", "60"))

    base = url_for('main.download_file', token=token,
_external=True)
    t = make_download_token(token, ttl_seconds)
    download_link = f"{base}?t={t}"
```

```
    users = User.query.filter(User.id !=
current_user.id).order_by(User.username.asc()).all()

    return render_template(
        'success.html',
        link=download_link,
        filename=file_record.filename,
        token=token,
        users=users
    )

@bp.route('/d/<token>')
def download_file(token: str):
    """
    Скачивание файла:
    - rate limit по IP;
    - проверка подписанного временного токена из query (?t=...);
    - аудит всех исходов.
    """
    ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
    ua = request.headers.get("User-Agent", "")[:250]
    username = current_user.username if getattr(current_user,
"is_authenticated", False) else "anonymous"

    if not allow_rate(f"dl:{ip}", DL_LIMIT, DL_WINDOW):
        current_app.logger.info(
            f"AUDIT action=download user={username} token={token}
file=- ip={ip} result=rate_limited"
        )
        return "Too Many Requests", 429

    file_record = FileRecord.query.filter_by(token=token).first()
    if file_record is None:
        try:
            db.session.add(DownloadLog(
                action="download",
                file_token=token,
                filename=None,
                username=username,
                ip=ip,
                user_agent=ua,
                result="not_found",
                error="token_not_found"
            ))
            current_app.logger.info(
                f"AUDIT action=download user={username}
token={token} file=- ip={ip} result=not_found
error=token_not_found"
            )
            db.session.commit()
        except Exception:
```

```
            db.session.rollback()
        return "Not found", 404

    t = request.args.get("t", "")
    ok, reason = verify_download_token(t, token)
    if not ok:
        try:
            db.session.add(DownloadLog(
                action="download",
                file_token=file_record.token,
                filename=file_record.filename,
                username=username,
                ip=ip,
                user_agent=ua,
                result="blocked",
                error=f"link_{reason}"
            ))
            current_app.logger.info(
                f"AUDIT action=download user={username}
token={file_record.token} file={file_record.filename} ip={ip}
result=blocked error=link_{reason}"
            )
            db.session.commit()
        except Exception:
            db.session.rollback()

        return ("Срок действия ссылки истёк", 410) if reason ==
"expired" else ("Доступ по ссылке запрещён", 403)

    file_path = os.path.join(current_app.config["UPLOAD_FOLDER"],
file_record.secure_name)
    if not os.path.exists(file_path):
        try:
            db.session.add(DownloadLog(
                action="download",
                file_token=file_record.token,
                filename=file_record.filename,
                username=username,
                ip=ip,
                user_agent=ua,
                result="error",
                error="file_missing_on_disk"
            ))
            current_app.logger.info(
                f"AUDIT action=download user={username}
token={file_record.token} file={file_record.filename} ip={ip}
result=error error=file_missing_on_disk"
            )
            db.session.commit()
        except Exception:
            db.session.rollback()
        return "Файл отсутствует на диске", 404
```

```
    try:
        with open(file_path, "rb") as f:
            encrypted_data = f.read()
        decrypted_data = decrypt_data(encrypted_data)
    except Exception as e:
        try:
            db.session.add(DownloadLog(
                action="download",
                file_token=file_record.token,
                filename=file_record.filename,
                username=username,
                ip=ip,
                user_agent=ua,
                result="error",
                error=f"decrypt_failed:{type(e).__name__}"
            ))
            current_app.logger.info(
                f"AUDIT action=download user={username}
token={file_record.token} file={file_record.filename} ip={ip}
result=error error=decrypt_failed:{type(e).__name__}"
            )
            db.session.commit()
        except Exception:
            db.session.rollback()
        return "Ошибка расшифрования", 500

    try:
        file_record.download_count += 1
        db.session.add(DownloadLog(
            action="download",
            file_token=file_record.token,
            filename=file_record.filename,
            username=username,
            ip=ip,
            user_agent=ua,
            result="success",
            error=None
        ))
        current_app.logger.info(
            f"AUDIT action=download user={username}
token={file_record.token} file={file_record.filename} ip={ip}
result=success"
        )
        db.session.commit()
    except Exception:
        db.session.rollback()

    return send_file(
        io.BytesIO(decrypted_data),
        download_name=file_record.filename,
        as_attachment=True
```

```
    )

@bp.route('/admin')
@login_required  # <-- ЗАЩИТА: Сначала проверяем логин
def admin_dashboard():
    """
    Панель мониторинга. Доступна ТОЛЬКО администраторам.
    """
    # Вторая проверка: проверяем флаг is_admin в базе
    if not current_user.is_admin:
        # Ошибка 403 Forbidden (Доступ запрещен)
        return render_template('base.html', content="<h1>Ошибка
403</h1><p>У вас нет прав администратора.</p>"), 403

    cpu_usage = psutil.cpu_percent(interval=1)
    memory = psutil.virtual_memory()
    disk = psutil.disk_usage('/')

    total_files = FileRecord.query.count()
    total_downloads =
db.session.query(db.func.sum(FileRecord.download_count)).scalar()
or 0

    users_count = User.query.count()

    system_stats = {
        'cpu': cpu_usage,
        'ram_percent': memory.percent,
        'disk_free_gb': round(disk.free / (1024 ** 3), 2),
        'files_count': total_files,
        'downloads_count': total_downloads,
        'users_count': users_count
    }
    last_logs =
DownloadLog.query.order_by(DownloadLog.ts.desc()).limit(10).all()

    return render_template("admin.html", stats=system_stats,
logs=last_logs)

@bp.route("/api/admin/stats")
@login_required
def api_admin_stats():
    """JSON API: динамическая статистика сервера для админки."""
    if not current_user.is_admin:
        return jsonify({"error": "forbidden"}), 403

    cpu = psutil.cpu_percent(interval=0.1)
    mem = psutil.virtual_memory()

    # Диск считаем по папке uploads (логично для твоего сервиса)
    disk = psutil.disk_usage(current_app.config["UPLOAD_FOLDER"])
```

```
    total_files = FileRecord.query.count()
    total_downloads =
db.session.query(db.func.sum(FileRecord.download_count)).scalar()
or 0
    users_count = User.query.count()

    disk_free_gb = round(disk.free / (1024**3), 2)

    issues = []
    if cpu >= 85:
        issues.append("Высокая загрузка CPU")
    if mem.percent >= 85:
        issues.append("Высокое потребление RAM")
    if disk_free_gb <= 1:
        issues.append("Мало свободного места на диске")

    if disk_free_gb <= 0.2 or cpu >= 95 or mem.percent >= 95:
        health = "CRIT"
    elif issues:
        health = "WARN"
    else:
        health = "OK"

    return jsonify({
        "health": health,
        "issues": issues,
        "cpu": round(cpu, 1),
        "ram_percent": round(mem.percent, 1),
        "disk_free_gb": disk_free_gb,
        "files_count": total_files,
        "downloads_count": int(total_downloads),
        "users_count": users_count,
        "ts": datetime.utcnow().isoformat() + "Z"
    })

@bp.route('/admin/create_user', methods=['POST'])
@login_required
def create_user():
    """
    Создание нового пользователя администратором.
    """
    if not current_user.is_admin:
        return "У вас нет прав!", 403

    username = request.form.get('username')
    password = request.form.get('password')

    if User.query.filter_by(username=username).first():
        flash('Ошибка: Пользователь с таким именем уже
существует!')
        return redirect(url_for('main.admin_dashboard'))
```

```
    # Создаем нового пользователя
    new_user = User(username=username)
    new_user.set_password(password)  # Хешируем пароль

    db.session.add(new_user)
    db.session.commit()

    flash(f'Пользователь {username} успешно создан!')
    return redirect(url_for('main.admin_dashboard'))

@bp.route('/admin/cleanup', methods=['POST'])
@login_required
def admin_cleanup():
    """
    ПОЛНАЯ ОЧИСТКА.
    Удаляет ВСЕ файлы из базы и с диска.
    Идеально для демонстрации работы функции.
    """
    if not current_user.is_admin:
        flash("У вас нет прав администратора!")
        return redirect(url_for('main.index'))

    files_to_delete = FileRecord.query.all()

    deleted_count = 0

    for file_record in files_to_delete:
        # Полный путь
        file_path =
os.path.join(current_app.config['UPLOAD_FOLDER'],
file_record.secure_name)

        # 1. Удаляем с диска
        if os.path.exists(file_path):
            try:
                os.remove(file_path)
            except Exception:
                pass  # Если файла нет, не страшно

        # 2. Удаляем из БД
        db.session.delete(file_record)
        deleted_count += 1

    db.session.commit()

    if deleted_count > 0:
        flash(f' Хранилище полностью очищено! Удалено
{deleted_count} файлов.', 'success')
    else:
        flash('Хранилище уже пустое.', 'info')

    return redirect(url_for('main.admin_dashboard'))
```

```
@bp.route("/api/admin/logs")
@login_required
def api_admin_logs():
    if not current_user.is_admin:
        return jsonify({"error": "forbidden"}), 403

    rows =
DownloadLog.query.order_by(DownloadLog.ts.desc()).limit(10).all()
    return jsonify([
        {
            "ts": r.ts.isoformat() + "Z",
            "username": r.username,
            "filename": r.filename,
            "ip": r.ip,
            "result": r.result,
            "action": r.action
        } for r in rows
    ])

@bp.app_context_processor
def inject_inbox_count():
    if not getattr(current_user, "is_authenticated", False):
        return {"inbox_count": 0}

    now = datetime.utcnow()
    cnt = (FileShare.query
           .filter(FileShare.recipient_id == current_user.id)
           .filter(FileShare.revoked == False)
           .filter(FileShare.is_new == True)
           .filter((FileShare.expires_at.is_(None)) |
(FileShare.expires_at > now))
           .filter(FileShare.downloads_done <
FileShare.max_downloads)
           .count())
    return {"inbox_count": cnt}

@bp.route("/share/create", methods=["POST"])
@login_required
def create_share():
    """
    Создание «внутренней передачи» файла конкретному пользователю.
    TTL задаётся в минутах (name="ttl_minutes" в шаблоне
success.html).
    """
    file_token = request.form.get("file_token", "").strip()
    recipient_id = request.form.get("recipient_id", "").strip()
    ttl_minutes = request.form.get("ttl_minutes", "1440").strip()
    max_downloads = request.form.get("max_downloads", "1").strip()

    if not file_token or not recipient_id:
        flash("Не указан получатель или файл")
```

```
        return redirect(request.referrer or url_for("main.index"))

    fr = FileRecord.query.filter_by(token=file_token).first()
    if fr is None or fr.user_id != current_user.id:
        flash("Нельзя отправить: файл не найден или не принадлежит
вам")
        return redirect(request.referrer or url_for("main.index"))

    # recipient_id
    try:
        recipient_id_i = int(recipient_id)
    except ValueError:
        flash("Некорректный получатель")
        return redirect(request.referrer or url_for("main.index"))

    recip = User.query.get(recipient_id_i)
    if recip is None:
        flash("Получатель не найден")
        return redirect(request.referrer or url_for("main.index"))
    expires_at = None
    try:
        ttl_i = int(ttl_minutes)
        if ttl_i > 0:
            expires_at = datetime.utcnow() +
timedelta(minutes=ttl_i)
    except ValueError:
        expires_at = datetime.utcnow() + timedelta(hours=24)

    try:
        max_d = int(max_downloads)
        if max_d < 1:
            max_d = 1
    except ValueError:
        max_d = 1

    share = FileShare(
        file_token=fr.token,
        filename=fr.filename,
        sender_id=current_user.id,
        recipient_id=recipient_id_i,
        expires_at=expires_at,
        max_downloads=max_d,
        downloads_done=0,
        revoked=False
    )
    db.session.add(share)

    ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
    ua = request.headers.get("User-Agent", "")[:250]
    db.session.add(DownloadLog(
        action="share_create",
```

```
        file_token=fr.token,
        filename=fr.filename,
        username=current_user.username,
        ip=ip,
        user_agent=ua,
        result="success",
        error=None
    ))
    current_app.logger.info(
        f"AUDIT action=share_create sender={current_user.username}
recipient={recip.username} "
        f"token={fr.token} file={fr.filename}
max_downloads={max_d} expires_at={expires_at} result=success"
    )

    db.session.commit()
    flash(f"Файл отправлен пользователю {recip.username}")
    return redirect(url_for("main.inbox"))

@bp.route("/inbox")
@login_required
def inbox():
    now = datetime.utcnow()

    (FileShare.query
     .filter(FileShare.recipient_id == current_user.id)
     .filter(FileShare.revoked == False)
     .filter(FileShare.is_new == True)
     .filter((FileShare.expires_at.is_(None)) |
(FileShare.expires_at > now))
     .filter(FileShare.downloads_done < FileShare.max_downloads)
     .update({FileShare.is_new: False},
synchronize_session=False))
    db.session.commit()

    shares = (FileShare.query
              .filter_by(recipient_id=current_user.id,
revoked=False)
              .order_by(FileShare.created_at.desc())
              .all())

    active = []
    for s in shares:
        if s.expires_at and s.expires_at < now:
            continue
        if s.downloads_done >= s.max_downloads:
            continue
        active.append(s)

    return render_template("inbox.html", shares=active)

@bp.route("/s/<share_token>")
```

```
@login_required
def download_shared(share_token: str):
    ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
    ua = request.headers.get("User-Agent", "")[:250]
    s =
FileShare.query.filter_by(share_token=share_token).first_or_404()

    if s.recipient_id != current_user.id:
        return "Доступ запрещён", 403

    if s.revoked:
        return "Доступ отозван", 410

    now = datetime.utcnow()
    if s.expires_at and s.expires_at < now:
        return "Срок действия истёк", 410

    if s.downloads_done >= s.max_downloads:
        current_app.logger.info(
            f"AUDIT action=download_inbox
user={current_user.username} share_token={share_token} ip={ip}
result=blocked error=expired_or_limit"
        )
        return "Лимит скачиваний исчерпан", 410

    s.is_new = False

    fr = FileRecord.query.filter_by(token=s.file_token).first()
    if fr is None:
        return "Файл не найден", 404

    file_path = os.path.join(current_app.config["UPLOAD_FOLDER"],
fr.secure_name)
    if not os.path.exists(file_path):
        return "Файл отсутствует на диске", 404

    with open(file_path, "rb") as f:
        encrypted_data = f.read()

    try:
        decrypted_data = decrypt_data(encrypted_data)
    except Exception:
        return "Ошибка расшифрования", 500

    ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
    ua = request.headers.get("User-Agent", "")[:250]

    s.downloads_done += 1
    db.session.add(DownloadLog(
        action="download_inbox",
```

```
            file_token=fr.token,
            filename=fr.filename,
            username=current_user.username,
            ip=ip,
            user_agent=ua,
            result="success",
            error=None
        ))

        current_app.logger.info(
            f"AUDIT action=download_inbox user={current_user.username}
share_token={share_token} "
            f"token={fr.token} file={fr.filename} ip={ip}
result=success"
        )

        db.session.commit()

        return send_file(
            io.BytesIO(decrypted_data),
            download_name=fr.filename,
            as_attachment=True
        )

@bp.route("/admin/audit")
@login_required
def admin_audit():
    if not current_user.is_admin:
        return "Доступ запрещён", 403

    # последние 200 событий (можно меньше/больше)
    logs = (DownloadLog.query
            .order_by(DownloadLog.ts.desc())
            .limit(200)
            .all())

    return render_template("admin_audit.html", logs=logs)

@bp.route("/shares/outbox")
@login_required
def shares_outbox():
    shares = FileShare.query.filter_by(sender_id=current_user.id)\
                            .order_by(FileShare.id.desc()).all()
    return render_template("shares_outbox.html", shares=shares)

@bp.route("/share/revoke/<int:share_id>", methods=["POST"])
@login_required
def revoke_share(share_id):
    share = FileShare.query.get_or_404(share_id)
    if share.sender_id != current_user.id and not
current_user.is_admin:
        abort(403)
```

```
    if not share.revoked:
        share.revoked = True
        db.session.add(share)

        ip = request.headers.get("X-Forwarded-For",
request.remote_addr) or ""
        ua = request.headers.get("User-Agent", "")[:250]
        db.session.add(DownloadLog(
            action="share_revoke",
            file_token=share.file_token,
            filename=share.filename,
            username=current_user.username,
            ip=ip,
            user_agent=ua,
            result="success",
            error=None
        ))
        current_app.logger.info(
            f"AUDIT action=share_revoke
sender={current_user.username} "
            f"recipient_id={share.recipient_id}
token={share.file_token} file={share.filename} result=success"
        )
        db.session.commit()

    flash("Передача отозвана")
    return redirect(url_for("main.shares_outbox"))
```

Листинг Б.5 – Файл tasks.py

```
import os
from datetime import datetime, timedelta
from app.extensions import db, scheduler
from app.models import FileRecord

def delete_old_files():
    """Фоновая задача очистки"""
    # Важно: используем scheduler.app для контекста
    with scheduler.app.app_context():
        expiration_time = datetime.utcnow() -
timedelta(seconds=30)

        files_to_delete =
FileRecord.query.filter(FileRecord.upload_date <
expiration_time).all()

        if files_to_delete:
            print(f"[{datetime.now().strftime('%H:%M:%S')}]
Найдено файлов: {len(files_to_delete)}")

            for file_record in files_to_delete:
```

```
                upload_folder =
scheduler.app.config['UPLOAD_FOLDER']
                file_path = os.path.join(upload_folder,
file_record.secure_name)

                if os.path.exists(file_path):
                    try:
                        os.remove(file_path)
                        print(f"   --> Удален с диска:
{file_record.filename}")
                    except Exception as e:
                        print(f"   --> Ошибка диска: {e}")

                db.session.delete(file_record)

            try:
                db.session.commit()
                print(f" Удалено из БД.")
            except Exception as e:
                print(f" Ошибка БД: {e}")
```

Листинг Б.6 – Файл test_validate_upload_mc_dc.py

```
# tests/test_validate_upload_mc_dc.py
import pytest

def validate_upload(file_present: bool, filename_nonempty: bool,
size_ok: bool, ext_ok: bool) -> str:
    if not file_present:
        return "E_NO_FILE"
    if not filename_nonempty:
        return "E_EMPTY_NAME"
    if (not size_ok) or (not ext_ok):
        return "E_POLICY"
    return "OK"

@pytest.mark.parametrize(
    "file_present, filename_nonempty, size_ok, ext_ok, expected",
    [
        (0, 1, 1, 1, "E_NO_FILE"),   # T_A
        (1, 0, 1, 1, "E_EMPTY_NAME"),# T_B
        (1, 1, 0, 1, "E_POLICY"),    # T_C
        (1, 1, 1, 0, "E_POLICY"),    # T_D
        (1, 1, 1, 1, "OK"),          # T_OK
    ],
)
def test_validate_upload_mc_dc(file_present, filename_nonempty,
size_ok, ext_ok, expected):
    assert validate_upload(bool(file_present),
bool(filename_nonempty), bool(size_ok), bool(ext_ok)) == expected
```

Листинг Б.7 – Файл config.py

```
import os
from pathlib import Path

SESSION_COOKIE_HTTPONLY = True
SESSION_COOKIE_SAMESITE = "Lax"
REMEMBER_COOKIE_HTTPONLY = True

def get_data_dir(app_name: str = "SecureShare") -> Path:
    if os.name == "nt":
        base = os.environ.get("LOCALAPPDATA") or
os.environ.get("APPDATA")
        if base:
            return Path(base) / app_name
        return Path.home() / app_name

    xdg = os.environ.get("XDG_DATA_HOME")
    if xdg:
        return Path(xdg) / app_name
    return Path.home() / ".local" / "share" / app_name

class Config:
    SECRET_KEY = os.environ.get("SECRET_KEY") or "secret-key-123"

    DATA_DIR = get_data_dir("SecureShare")
    DB_PATH = (DATA_DIR / "app.db").as_posix()

    SQLALCHEMY_DATABASE_URI = f"sqlite:///{DB_PATH}"
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    UPLOAD_FOLDER = (DATA_DIR / "uploads").as_posix()
    MAX_CONTENT_LENGTH = 16 * 1024 * 1024
```

Листинг Б.8 – Файл launcher.py

```
import threading
import time
import webbrowser
import socket
import tkinter as tk

from waitress import serve
from app import create_app

PORT = 5000
LISTEN_HOST = "0.0.0.0"
OPEN_HOST = "127.0.0.1"

def get_lan_ip() -> str:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        s.connect(("192.168.0.1", 80))
        ip = s.getsockname()[0]
    except Exception:
```

```
        ip = "127.0.0.1"
    finally:
        s.close()
    return ip

def run_server():
    app = create_app()
    serve(app, host=LISTEN_HOST, port=PORT, threads=8)

def main():
    lan_ip = get_lan_ip()
    local_url = f"http://{OPEN_HOST}:{PORT}/"
    lan_url = f"http://{lan_ip}:{PORT}/"

    t = threading.Thread(target=run_server, daemon=True)
    t.start()

    time.sleep(0.8)
    webbrowser.open(local_url)

    root = tk.Tk()
    root.title("SecureShare")
    root.geometry("680x340")
    root.minsize(680, 340)          # <-- важное: не даём окну
схлопнуться
    root.resizable(False, False)

    root.configure(padx=14, pady=12)

    title = tk.Label(root, text="SecureShare запущен",
font=("Arial", 16, "bold"))
    title.pack(pady=(4, 10))

    box = tk.Frame(root, bd=1, relief="solid")
    box.pack(pady=8, fill="x")

    tk.Label(box, text=f"Локально: {local_url}", font=("Arial",
11)).pack(anchor="w", padx=12, pady=(10, 4))
    tk.Label(box, text=f"В сети (LAN): {lan_url}", font=("Arial",
11)).pack(anchor="w", padx=12, pady=(0, 10))

    btn_frame = tk.Frame(root)
    btn_frame.pack(pady=10)

    tk.Button(btn_frame, text="Открыть (локально)", width=22,
              command=lambda:
webbrowser.open(local_url)).grid(row=0, column=0, padx=8)

    tk.Button(btn_frame, text="Открыть (LAN)", width=22,
              command=lambda:
webbrowser.open(lan_url)).grid(row=0, column=1, padx=8)
```

```
    hint = tk.Label(
        root,
        text="Администратор и пользователи работают через один
сервер.\nДоступ определяется ролями (RBAC) внутри системы.",
        fg="#555",
        font=("Arial", 10),
        justify="center"
    )
    hint.pack(pady=(10, 10))

    # <-- делаем кнопку заметной и высокой
    btn_stop = tk.Button(root, text="Остановить сервер",
command=root.destroy)
    btn_stop.pack(fill="x", pady=(6, 0), ipady=6)

    root.mainloop()

if __name__ == "__main__":
    main()
```

Листинг Б.9 – Файл run.py

```
from app import create_app
app = create_app()

if __name__ == '__main__':
    # Обычный запуск
    app.run(host="0.0.0.0", port=5000, debug=False,
use_reloader=False)
```

Листинг Б.8 – Файл serve.py

```
from waitress import serve
from app import create_app
app = create_app()
if __name__ == "__main__":
    serve(app, host="0.0.0.0", port=5000, threads=8
```

Приложение В

Тестирование интерфейса

(обязательное)

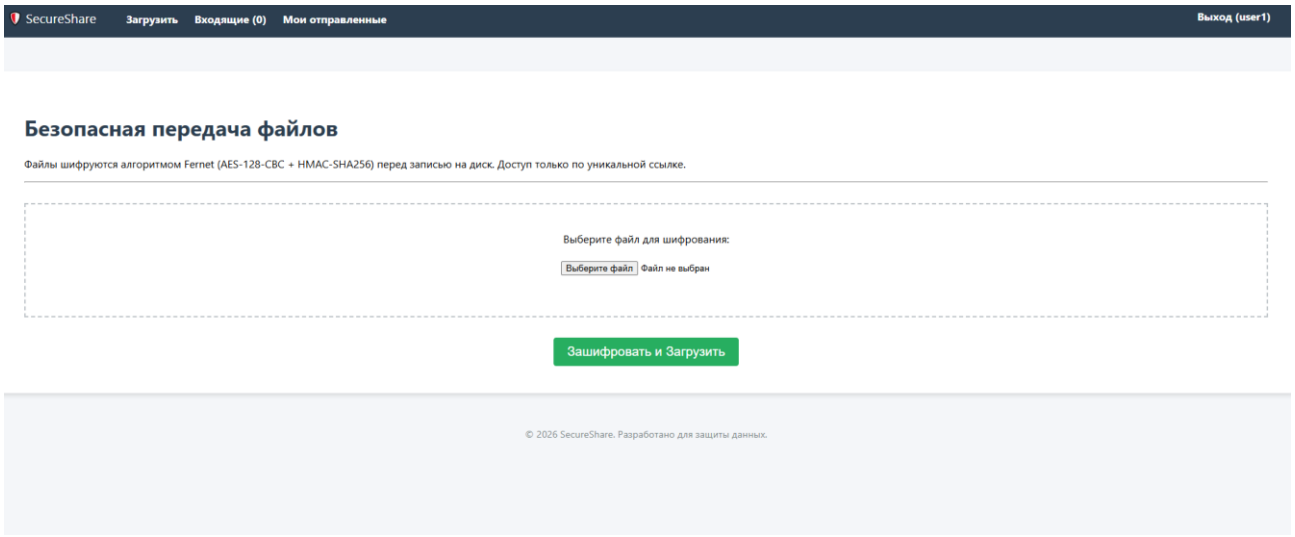


Рисунок В.1 – Успешный вход в учётную запись

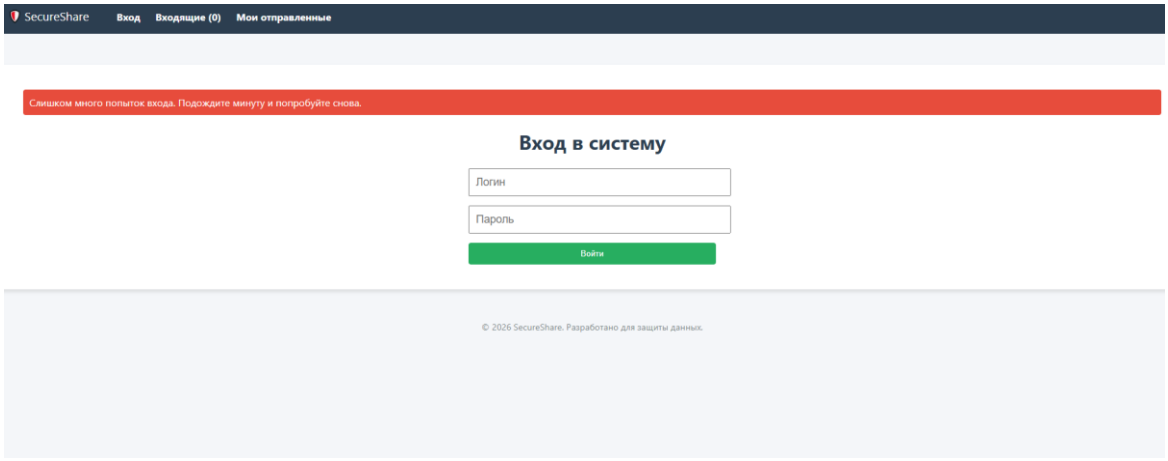


Рисунок В.2 – Проверка на корректность ввода

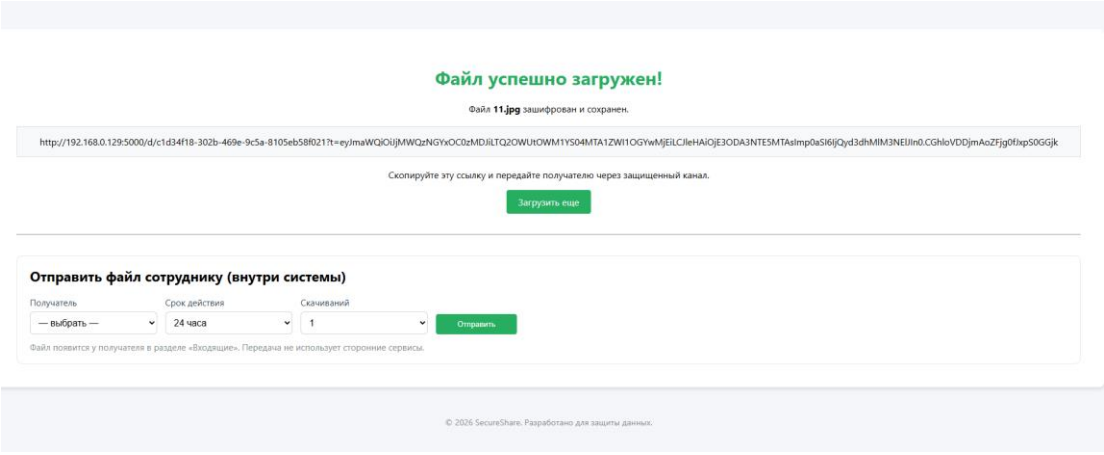


Рисунок В.3 – Загрузка файла в систему

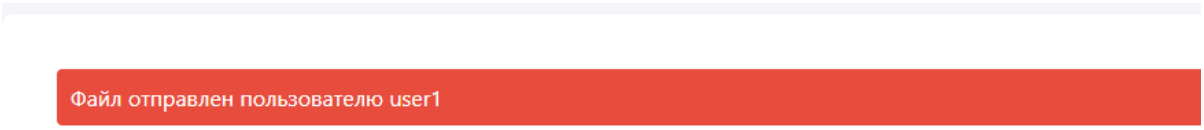


Рисунок В.4 – Успешная отправка файла пользователю

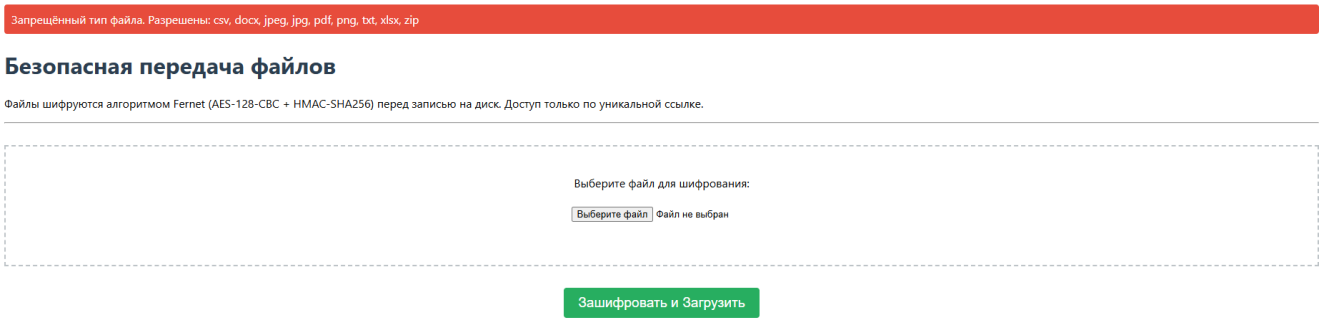


Рисунок В.5 – Попытка загрузить файл запрещённого типа

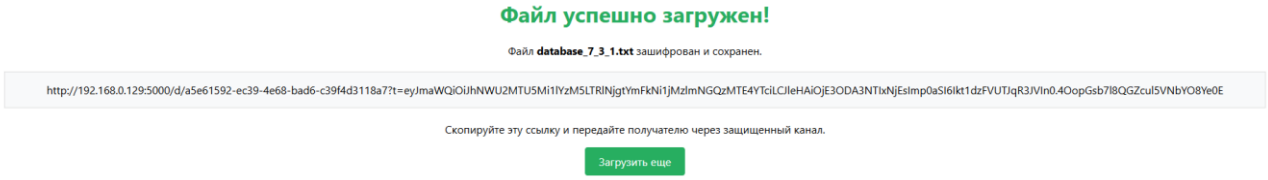


Рисунок В.6 – Генерация уникальной ссылки для отправки

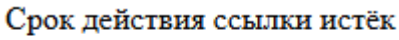


Рисунок В.7 – Ограничение времени работы ссылки

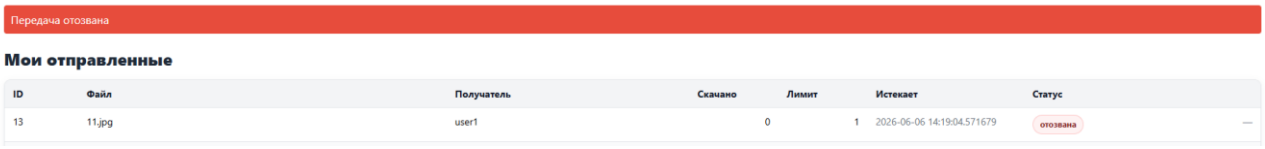


Рисунок В.8 – Успешная отмена передачи файла

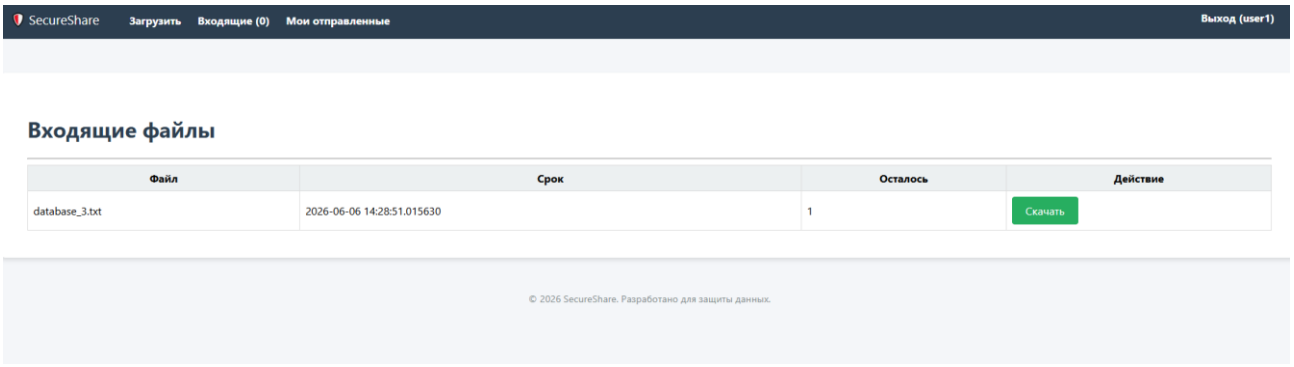


Рисунок В.9 – Проверка входящих файлов

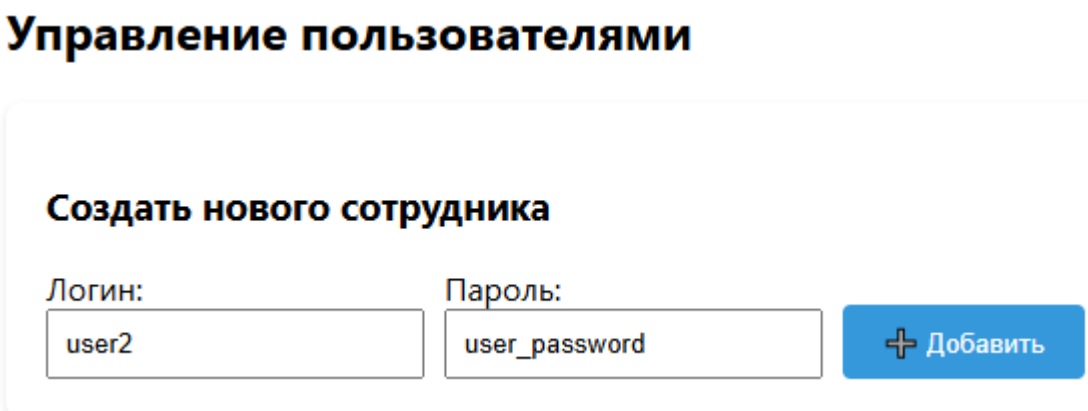


Рисунок В.10 – Создание нового пользователя

Рисунок В.11 – Успешное создание пользователя

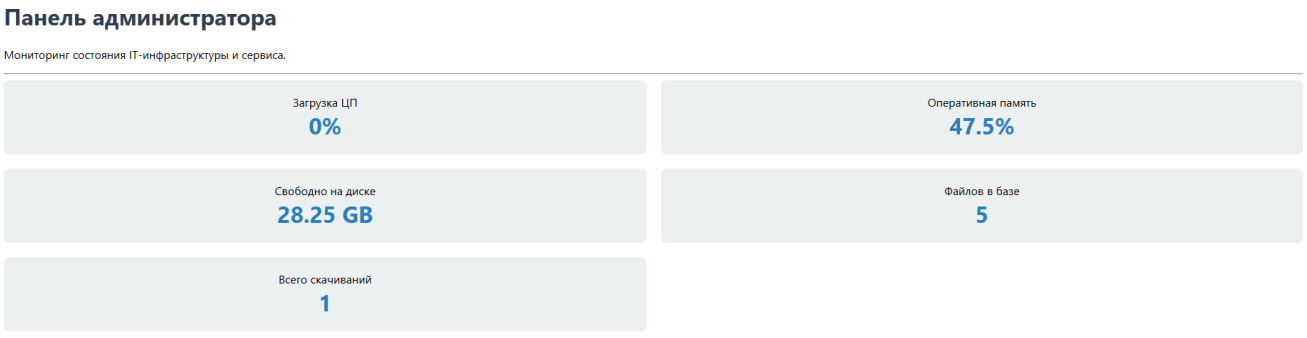


Рисунок В.12 – Состояние сервера

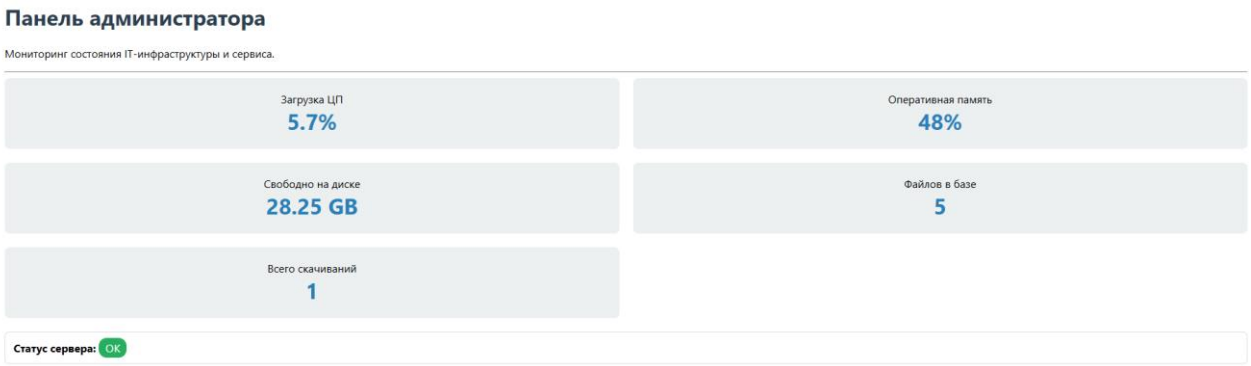
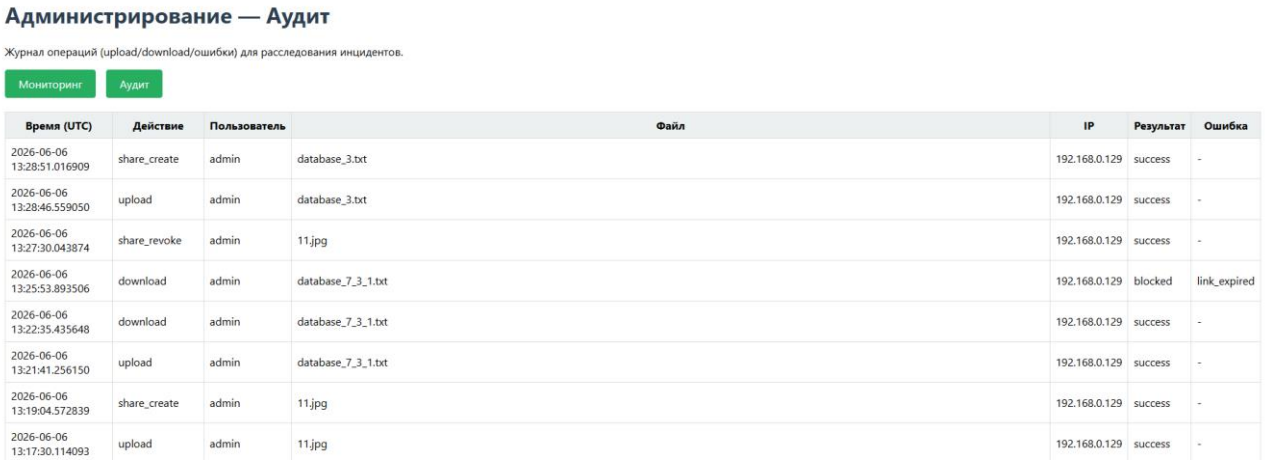


Рисунок В.13 – Состояние сервера спустя несколько секунд



Время (UTC)	Действие	Пользователь	Файл	IP	Результат	Ошибка
2026-06-06 13:28:51.016909	share_create	admin	database_3.txt	192.168.0.129	success	-
2026-06-06 13:28:46.559050	upload	admin	database_3.txt	192.168.0.129	success	-
2026-06-06 13:27:30.043874	share_revoke	admin	11.jpg	192.168.0.129	success	-
2026-06-06 13:25:53.893506	download	admin	database_7_3_1.txt	192.168.0.129	blocked	link_expired
2026-06-06 13:22:35.435648	download	admin	database_7_3_1.txt	192.168.0.129	success	-
2026-06-06 13:21:41.256150	upload	admin	database_7_3_1.txt	192.168.0.129	success	-
2026-06-06 13:19:04.572839	share_create	admin	11.jpg	192.168.0.129	success	-
2026-06-06 13:17:30.114093	upload	admin	11.jpg	192.168.0.129	success	-

Рисунок В.14 – Журнал администрирования (аудит)