

КАФЕДРА Системы обработки информации и управления

Система удаления шума из речи

(Подпись, дата)

(И.О.Фамилия)

2023 Շ.

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

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)

(И.О.Фамилия)
« ____ » _____ 2023 г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине Технологии разработки программного обеспечения

Студент группы ИУ5-25М

Назаров Максим Михайлович
(Фамилия, имя, отчество)

Тема курсового проекта _____ Система удаления шума из речи _____

Направленность КП (учебный, исследовательский, практический, производственный, др.) _____

Источник тематики (кафедра, предприятие, НИР) _____ кафедра _____

График выполнения проекта: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 15 нед.

Задание Выполнить разработку СОИУ в соответствии с описанием ее функциональности на основе моделей унифицированного процесса (RUP). Написать программу, реализующую фрагмент СОИУ, и реализовать в ней паттерны бизнес-логики (domain model), работы с БД (active record) и gof (хранитель).

Оформление курсового проекта:

Расчетно-пояснительная записка на _____ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « 7 » февраля 2023 г.

Руководитель курсового проекта

(Подпись, дата) М.В. Виноградова
(И.О.Фамилия)

Студент

(Подпись, дата) М.М. Назаров
(И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Оглавление

1. Этап анализа и планирования требований

- 1.1. Перечень функциональных и нефункциональных требований
- 1.2. Модель предметной области
- 1.3. Выявленные актёры
- 1.4. Выявленные прецеденты, их приоритеты и описание
- 1.5. Диаграмма прецедентов
- 1.6. Описание возможной архитектуры
- 1.7. Диаграмма уровней подсистем и развертывания
- 1.8. Перечень критических рисков
- 1.9. Перечень экранных форм и их сложность
- 1.10. Расчёт затрат
- 1.11. Расчет длительности и стоимости разработки

2. Этап проектирования

- 2.1. Уточненная диаграмма прецедентов
- 2.2. Расширенное описание основных прецедентов
- 2.3. Прототип пользовательского интерфейса
- 2.4. Диаграммы классов анализа
- 2.5. Диаграммы взаимодействия для основных прецедентов
- 2.6. Диаграммы классов участников
- 2.7. Пакеты анализа и сервисные пакеты в форме обобщенной диаграммы классов
- 2.8. Трассировка пакетов в подсистемы
- 2.9. Модель трассировки классов анализа в классы проектирования
- 2.10. Диаграмма распределения классов проектирования по подсистемам
- 2.11. Диаграмма уровней подсистем
- 2.12. Диаграмма размещения подсистем
- 2.13. Модель трассировки подсистем в компоненты
- 2.14. Трассировка классов проектирования в исходные файлы
- 2.15. Зависимость компонентов от исходных файлов
- 2.16. Переработанный список рисков
- 2.17. Перечень итераций и их состав

3. Этап построения (Конструирования)

- 3.1. Экранные формы работающей программы
- 3.2. Исходный код программы, реализующей архитектурно-значимые прецеденты и паттерны

3.2.1. Реализация классов Domain model

3.2.2. Реализация классов Active Record

3.3. Полная диаграмма классов реализации

3.4. Диаграммы последовательностей для иллюстрации работы паттернов

3.5. Перечень и последовательность проведения тестов

3.6. Примеры модульных и функциональных тестов

4. Этап внедрения

4.1. Перечень программ и рекомендаций по установке

4.2. Перечень документации для пользователей и заказчиков

4.3. Перечень документации для пользователя

4.4. Рекомендации по внедрению

Заключение

Список литературы

1. Этап анализа и планирования требований

Таблица 1. Постановка задачи (задание по варианту)

Тема	Паттерн бизнес-логики	Паттерн работы с БД	Паттерн GOF
Система удаления шума из речи	domain model	active record	хранитель

1.1. Перечень функциональных и нефункциональных требований

Функциональные требования:

Система удаления шума из речи должна выполнять следующие функциональные требования:

- Загрузка аудиофайла формата .wav.
- Удаление из файла частот содержащих шумы, выявленных в процессе анализа аудиофайла при его получении.
- Загрузка обработанного аудиофайла.
- Регистрация и авторизация пользователей.
- История обработанных файлов, содержащая имя пользователя, имя файла и дату загрузки, для каждого авторизованного пользователя.

Нефункциональные требования:

Ограничения среды и реализации:

- Система должна разрабатываться на ЯП Python 3.10.8;
- Система должна запускаться в современном веб-браузере.

Требования по производительности:

- Для работы системы необходимо не более 2 ГБ оперативной памяти;
- Для работы системы требуется стабильное интернет соединение с пропускной способностью 10 Мбит/с.

Требования по зависимости от платформы:

- Система должна работать на ОС Windows.

Требования по надежности:

- Система не должна выдавать ошибок, не предусмотренных работой;
- Система должна надежно и устойчиво функционировать;
- Система должна иметь возможность быть восстановленной в течение часа, в случае непредвиденных проблем функционирования.

Требования к пользовательскому интерфейсу:

- Дизайн системы должен быть лаконичным и понятным любому пользователю.

Требования к форматам и протоколам передачи данных:

- Запросы к внешним сервисам должны осуществляться при помощи сетевого протокола HTTP версии 1.1.

1.2. Модель предметной области

Была составлена модель предметной области, она же диаграмма классов, которая представлена на рисунке 1.

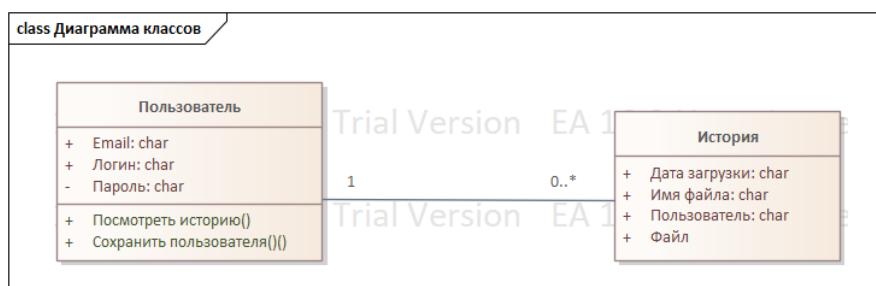


Рисунок 1. Модель предметной области

1.3. Выявленные актёры

Были выявлены следующие актёры:

1. Пользователь. Он может загружать файлы для их обработки, скачивать обработанный файл из системы, зарегистрироваться и смотреть на свою историю обработанных файлов.
2. Администратор. Он отвечает за правильность работы системы и контроль базы данных.

1.4. Выявленные прецеденты, их приоритеты и описание

Были выявлены следующие прецеденты, который расположены в порядке

убывания их приоритета:

1. Удаление шума - получает на вход файл, удаляет из него аудишум, позволяет пользователю его скачать обработанный файл.
2. Загрузить аудиофайл - предоставляет пользователю возможность загрузить свой аудио файл для последующей обработки его системой.
3. Скачать обработанный файл - предоставляет пользователю возможность скачать обработанной системой аудиофайл, ранее загруженный пользователем.
4. Зарегистрироваться - позволяет пользователю зарегистрироваться в системе.
5. Войти в учетную записи - позволяет зарегистрированным пользователям войти в свою учетную запись.
6. Просмотреть историю - позволяет зарегистрированным пользователям увидеть историю обработанных им ранее аудиофайлов.
7. Просмотр текущего состояние веб-приложения - позволяет администратору убедиться в работе системы.
8. Просмотреть базу данных - позволяет администратору просмотреть базу данных.

1.5. Диаграмма прецедентов

Была составлена диаграмма выявленных ранее прецедентов, диаграмма представлена на рисунке 2.

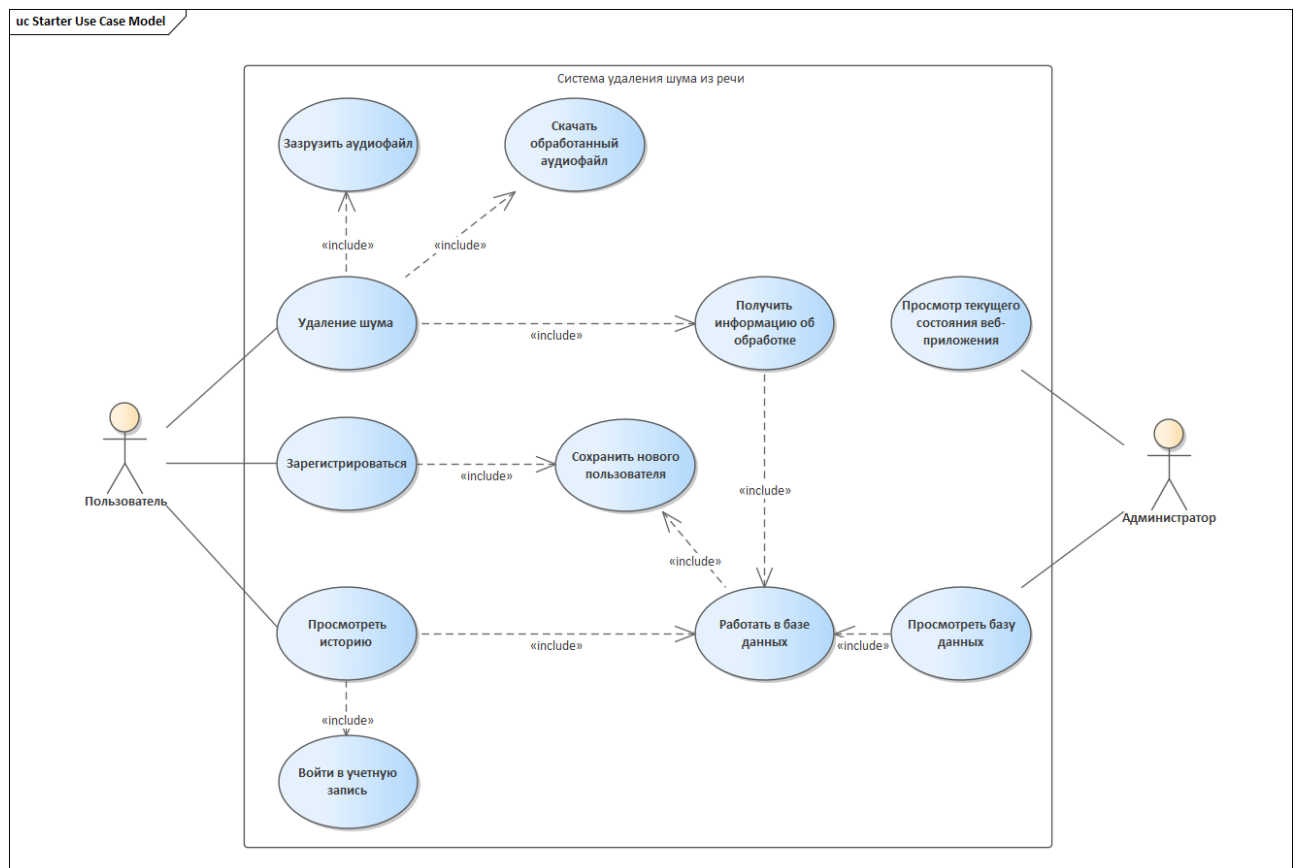


Рисунок 2. Диаграмма прецедентов

1.6. Описание возможной архитектуры

Архитектурно-значимые прецеденты:

1. Удаление шума;
2. Загрузить аудиофайл.

Возможные архитектурные решения:

1. Выбор архитектуры нейронной сети;
2. Разработка веб-интерфейса.

Решение архитектурных задач в аналогах

1. Выбор архитектуры нейронной системы

Перед выбором архитектуры нейронной сети надо решить как архитектуру использовать: RNN или U-Net.

RNN (Recurrent Neural Network) - это класс нейронных сетей, которые позволяют использовать предыдущие выходы в качестве входов, имея скрытое состояние. Они хорошо подходят для обработки последовательных данных, таких как речь или звук. Преимущества RNN заключаются в том, что они могут

учитывать контекст и динамику данных, а также моделировать долгосрочные зависимости. Недостатки RNN заключаются в том, что они подвержены проблемам затухания или взрыва градиентов, а также требуют большого количества вычислительных ресурсов и памяти.

U-Net - это тип сверточной нейронной сети (CNN), которая имеет U-образную структуру с симметричными сверточными и обратно сверточными слоями. Она была изначально разработана для задачи сегментации изображений, но также может быть применена к аудиоданным. Преимущества U-Net заключаются в том, что она может эффективно извлекать признаки на разных масштабах и сочетать локальную и глобальную информацию. Недостатки U-Net заключаются в том, что она требует большого количества обучающих данных.

В качестве архитектуры нейронной системы была выбрана архитектура U-Net. Это связано с тем, что за счет сверточных слоев она может улавливать шумы различных видов и эта архитектура является менее требовательной к вычислительным ресурсам.

2. Разработка веб-интерфейса.

Разработку веб-интерфейса можно выполнить за счет различных инструментов разработки веб-приложений например Django и React. Django - это фреймворк на Python, который следует архитектурному шаблону MVT (model-view-template), а React - это библиотека на JavaScript, которая используется для создания пользовательских интерфейсов. Преимущества Django , по сравнению с React:

- Django предоставляет более высокий уровень безопасности и защиты от таких атак, как SQL-инъекции или XSS;
- Django имеет встроенную поддержку аутентификации, авторизации, кэширования и других функций;
- Django обладает встроенной СУБД SQLite. Эта СУБД имеет синтаксис, который прост в освоении, также она обладает большим быстродействием.

На основе описанных выше преимуществ был выбран фреймворк Django.

1.7. Диаграмма уровней подсистем и развертывания

Была сформирована возможная диаграмма уровней подсистемы (см. рисунок 3), а также возможная диаграмма развертывания (см. рисунок 4).

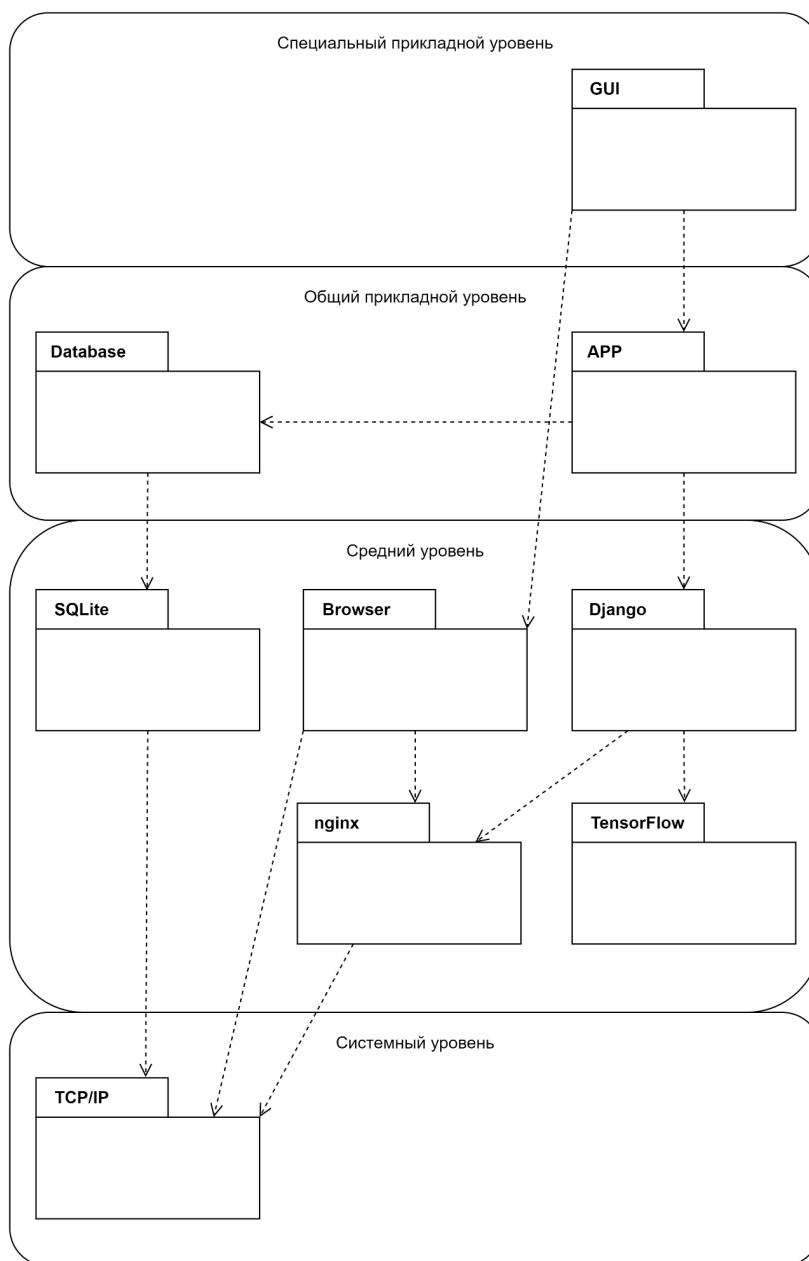


Рисунок 3. Возможная диаграмма уровней подсистем

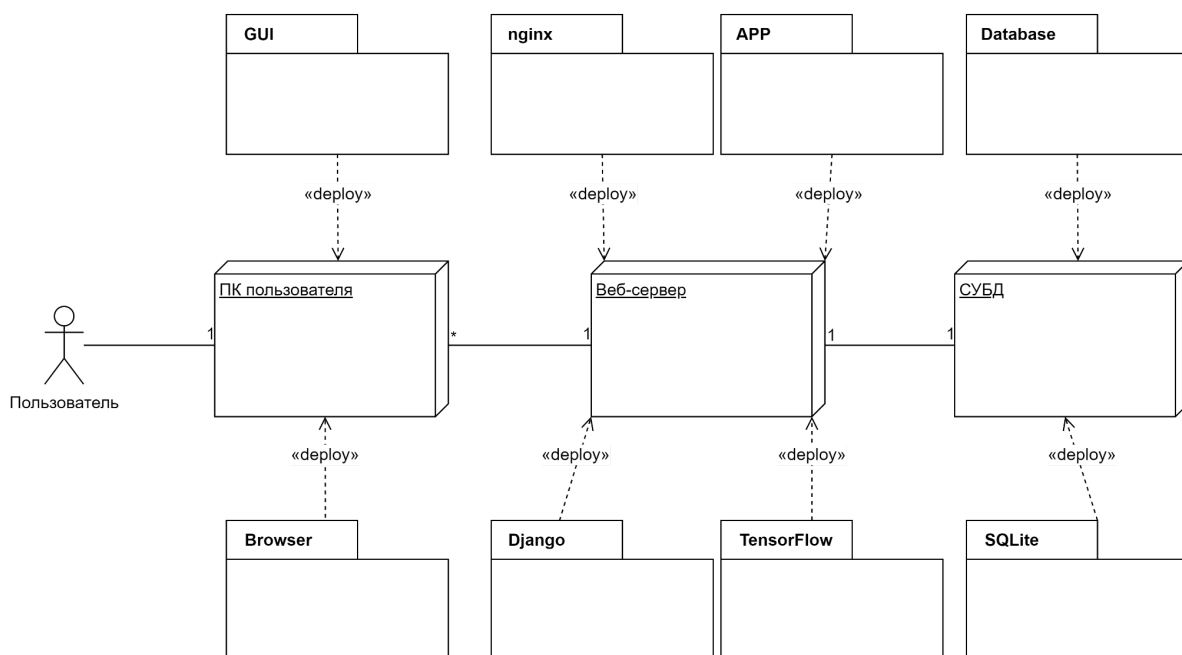


Рисунок 4. Возможная диаграмма развертывания

1.8. Перечень критических рисков

В таблице 2 представлены риски и рекомендации по их управлению.

Таблица 2. Риски и рекомендации по их управлению

Риск	План управления
Недостаточная эффективность удаления шума	Использование алгоритмов машинного обучения для более точного определения шума
Потеря качества звука	Проведение тестирования качества звука, оптимизация аппаратной части системы
Критическая программная ошибка	Поиск ошибки и ее дальнейшее устранение.
Низкая скорость отклика системы	Поиск возможностей для оптимизации и их внедрения в систему
Недостаточно понятный пользовательский интерфейс	Анализ пользовательского интерфейса, разработка новых вариантов интерфейса, их анализ и внедрение лучшего варианта интерфейса в систему.

Ошибка потери пользовательских данных	Поиск возможностей по созданию резервных копий данных и дальнейшему восстановлению из них
Отставание по срокам	Привлечение дополнительной рабочей силы

1.9. Перечень экранных форм и их сложность

Проект содержит следующие веб-страницы:

1. Главная страница

- 1.1. Панель навигации
- 1.2. Кнопка входа
- 1.3. Кнопка регистрации
- 1.4. Кнопка выхода (для авторизованного пользователя)
- 1.5. О проекте
- 1.6. История (для авторизованного пользователя)
- 1.7. Кнопка выбора файла
- 1.8. Кнопка обработки
- 1.9. Информацию о системе
- 1.10. Логин пользователя (для авторизованного пользователя)

2. О проекте

- 2.1. Панель навигации
- 2.2. Информацию о проекте

3. Регистрация

- 3.1. Поле Имя пользователя
- 3.2. Поле Пароль
- 3.3. Поле Повторите пароль
- 3.4. Поле Email
- 3.5. Кнопку Зарегистрироваться

3.6. Проверка введенных данных

3.7. Панель навигации

4. Войти

4.1. Поле Имя пользователя

4.2. Поле Пароль

4.3. Кнопка Войти

4.4. Проверка введенных данных

4.5. Панель навигации

5. История (доступная только авторизованным пользователям)

5.1. Информация о ранее обработанных файлах

5.2. Панель навигации

6. Загрузка

6.1. Панель навигации

6.2. Кнопка Загрузить

7. Страница не найдена

7.1. Панель навигации

7.2. Кнопка Вернуться на главную страницу

Для всех вышеперечисленных страниц, кроме «Страница не найдена», имеется два вида:

1. Авторизованный пользователь

2. Неавторизованного пользователь

1.10. Расчёт затрат

Оценка характеристик масштабных факторов приведена в таблице 3.

Таблица 3. Характеристика масштабных факторов

Масштабные факторы (W_i)	Значение
Предсказуемость PRED	4
Гибкость разработки FLEX	3
Разрешение архитектуры/риска RESL	5

Связанность групп TEAM	2
Зрелость процесса PMAT	3

$$B = 1,01 + 0,01 * (4 + 3 + 5 + 2 + 3) = 1,18$$

Производительности разработки номинальная $PROD = 13$.

Количество объектных указателей равно 12 ($OP = 12$).

В проекте не происходит повторного использования ресурсов, поэтому коэффициент $REUSE = 0\%$, поэтому $NOP = OP$.

Вычисление затрат по модели COCOMO-2 этапа композиции приложения:

$$\text{ЗАТРАТЫ} = \frac{NOP}{PROD} = \frac{12}{13} = 0,92$$

1.11. Расчет длительности и стоимости разработки

Расчет длительности по модели COCOMO-2 этапа композиции приложения:

$$(TDEV) = [3 * (\text{ЗАТРАТЫ})^{(0,33+0,2*(B-1,01))}] * \frac{SCEDPercentage}{100}$$

$$(TDEV) = 3 * 0,92^{0,364} * \frac{140}{100} \approx 4,1 \text{ {мес}}$$

Расчет стоимости по модели COCOMO-2 этапа композиции приложения:

$$\text{Стоимость} = \text{ЗАТРАТЫ} * \text{РАБ_КОЭФ}$$

$$\text{Стоимость} = 0,92 * 110 = 101 \text{ [тыс. руб.]}$$

2. Этап проектирования

2.1. Уточненная диаграмма прецедентов

Уточненная диаграмма прецедентов представлена на рисунке 5.

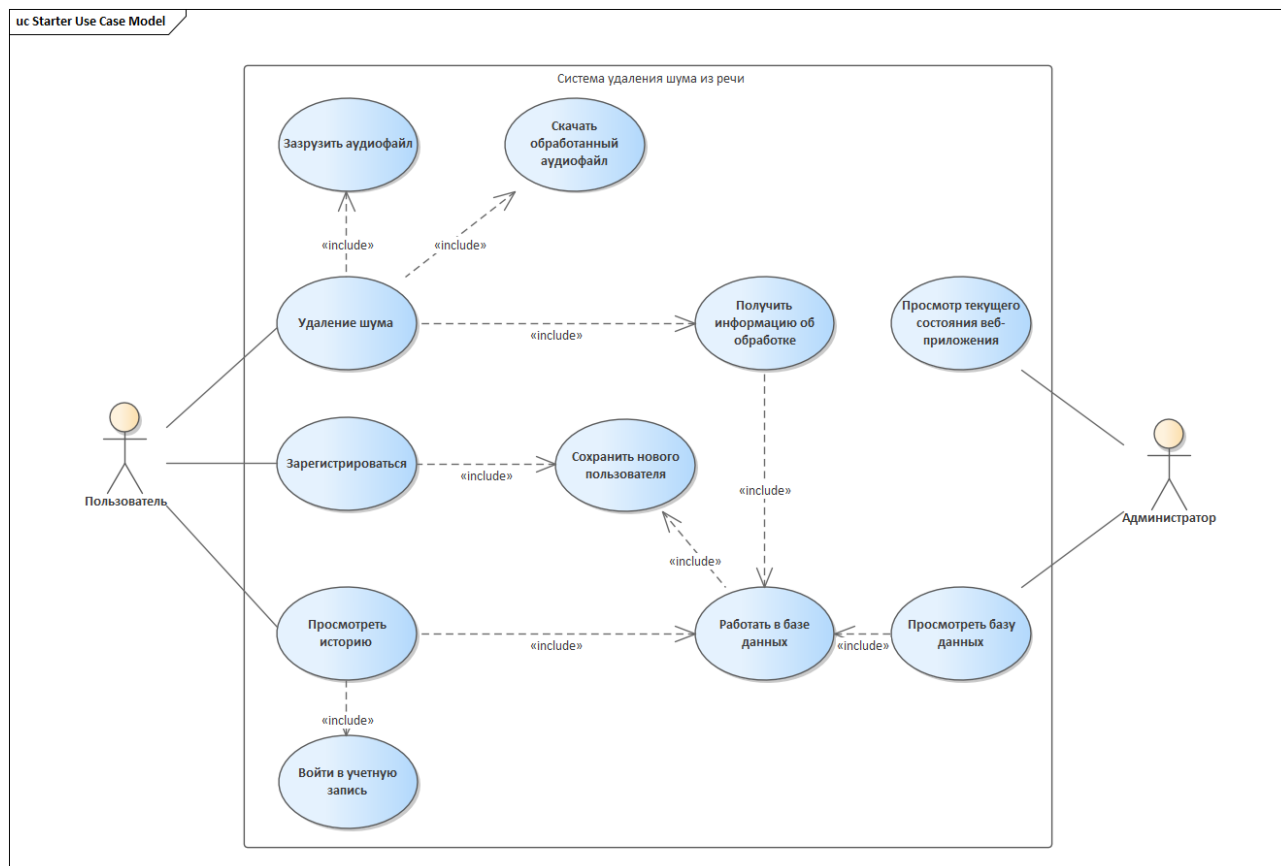


Рисунок 5. Уточненная диаграмма прецедентов

2.2. Расширенное описание основных прецедентов

Было разработано расширенное описание одного из основных прецедентов в виде спецификации. В таблице 5 представлен пример спецификации прецедента “Загрузить аудиофайл”.

Таблица 4. Спецификация основного прецедента “Загрузить аудиофайл”

Прецедент: “Загрузить аудиофайл”
Краткое описание: Загрузка аудиофайла на сайт для его дальнейшей обработки
Главный актёр: Пользователь
Предусловие: Перед началом этого прецедента пользователь должен зайти в систему
Основной поток: Прецедента “Загрузить аудиофайл” начинается, когда пользователь нажимает на кнопку выбрать файл. Поочередно выполняются следующие подпотоки

S-1: выбор файла в системном окне и S-2: обработка файла.

S-1: выбор файла в системном окне. При нажатии на кнопку “Выберите файл” происходит открытие системного окна с каталогом, где пользователю необходимо выбрать файл расширения .wav и нажать в системном окне кнопку “Открыть”. Возврат к началу прецедента.

S-2: обработка файла. Необходимо нажать кнопку “Обработать”. После происходит обработка файла и автоматическая переадресация на страницу загрузки (E-1).

Постусловие: Пользователем переадресован на страницу загрузки, где можно скачать обработанный файл. Если пользователь авторизован, то сохранение данных в БД.

Альтернативные потоки: E-1: нажата кнопка “Обработать”, но не выбран файл. Пользователь будет перенаправлен в начало прецедента.

2.3. Прототип пользовательского интерфейса

Ниже приведены рисунки 6-10 на которых показаны различные страницы прототипа пользовательского интерфейса.

Пользователь: maxzbox

Главная страницаО проектеИстория

Выход

Удаление шума из аудиофайлов

Выберите файл

Файл не выбран

Обработать

Инструкция по работе с системой:

Данная система позволяет вам загрузить аудиофайл формата .wav и скачать обработанную версию

Необходимо нажать на кнопку “Выберите файл” и выбрать аудиофайл в формате .wav, который вы хотите обработать

После выбора файла его название отобразится справа от кнопки “Выберите файл”, после чего необходимо нажать на кнопку “Обработать”

После обработки файла вас автоматически перенаправит на страницу загрузки, где можно скачать обработанную версию

Рисунок 6. Главная страница для авторизованного пользователя

Главная страница О проекте

Войти

Регистрация

Имя пользователя:

Пароль:

Повторите пароль:

Email:

Зарегистрироваться

Рисунок 7. Страница «Регистрация»

Главная страница О проекте

Войти

Регистрация

Имя пользователя:

Пароль:

Войти

Рисунок 8. Страница «Войти»

Пользователь: maxzbox

Главная страница О проекте История

Выход

История обработанных файлов

Пользователь	Имя файла	Дата загрузки
maxzbox	- reverb_fileid_135.wav	2022-05-19 18:50:51
maxzbox	piano.wav	2022-05-19 18:51:04
maxzbox	sample6.wav	2022-06-14 16:06:51

Рисунок 9. Страница «История»

Обработка завершена

Скачайте обработанный файл

sample6_enhanced.wav

Скачать

Обработать больше файлов

Рисунок 10. Страница «Загрузка»

2.4. Диаграммы классов анализа

Было выявлено 3 сущности: Пользователь, Администратор, История. Класс Пользователь отвечает за сохранение информации о пользователях: email, логин и пароль. Класс Администратор наследуется от класс Пользователь, он содержит все поля класса Пользователь, а также содержит уникальное поле - тип пользователя. Класс История отвечает за сохранение информации об обработанном файле, он хранит следующие поля; дата обработки, логин, пароль.

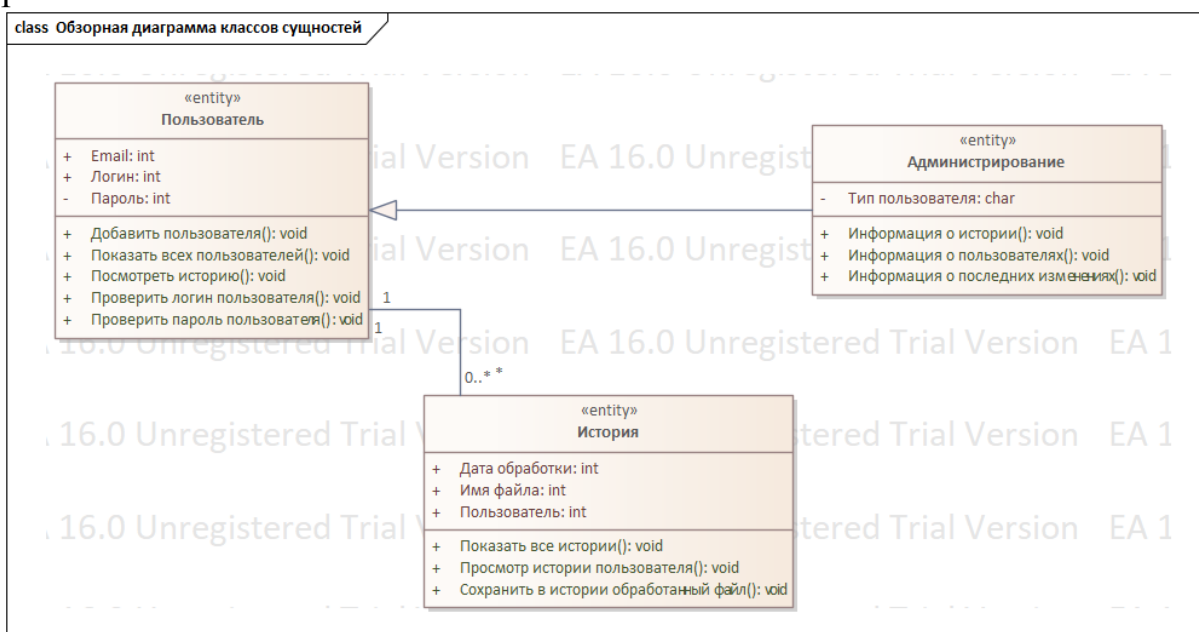


Рисунок 11. Обзорная диаграмма классов сущностей

Обзорная диаграмма управляющих классов приведена на рисунке 12.

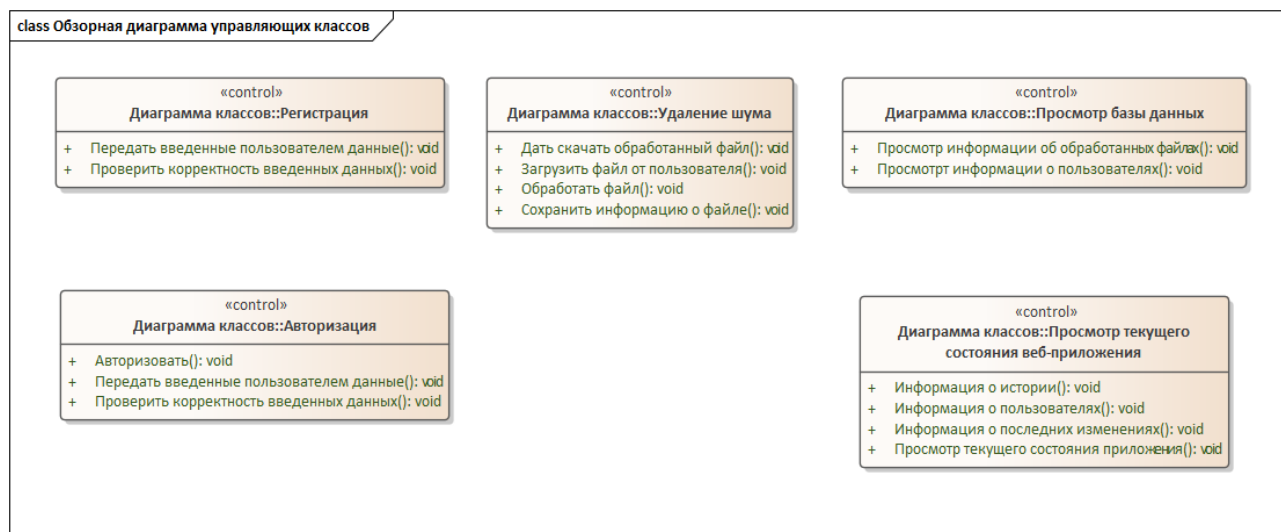


Рисунок 12. Обзорная диаграмма управляющих классов

На рисунке 13 приведена обзорная диаграмма граничных классов

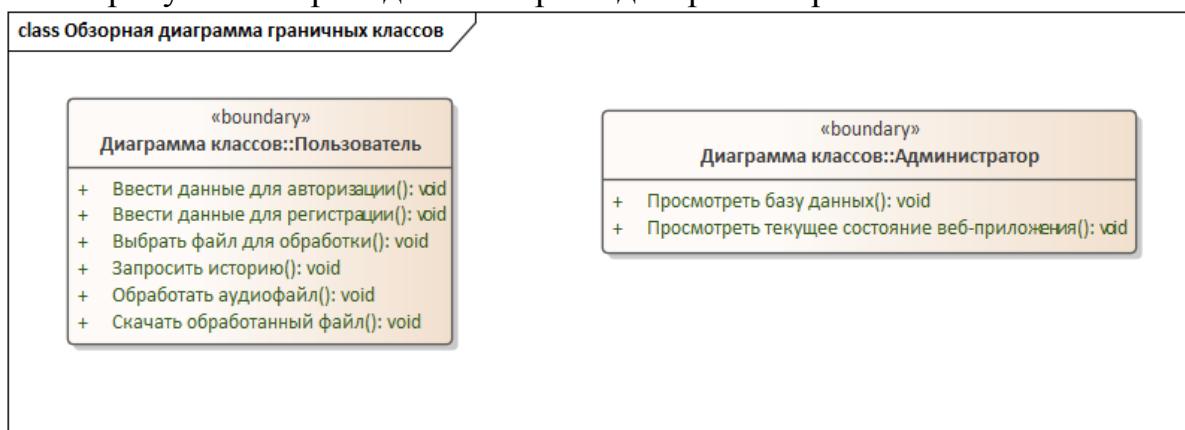


Рисунок 13. Обзорная диаграмма граничных классов

2.5. Диаграммы взаимодействия для основных прецедентов

Были сформированы диаграммы деятельности для прецедентов «Загрузить аудиофайл» (см. рисунок 14) и диаграмма последовательностей для этого прецедента (см. рисунок 15), «Скачать обработанный файл» (см. рисунок 16) и «Войти в учетную запись» (см. рисунок 17).

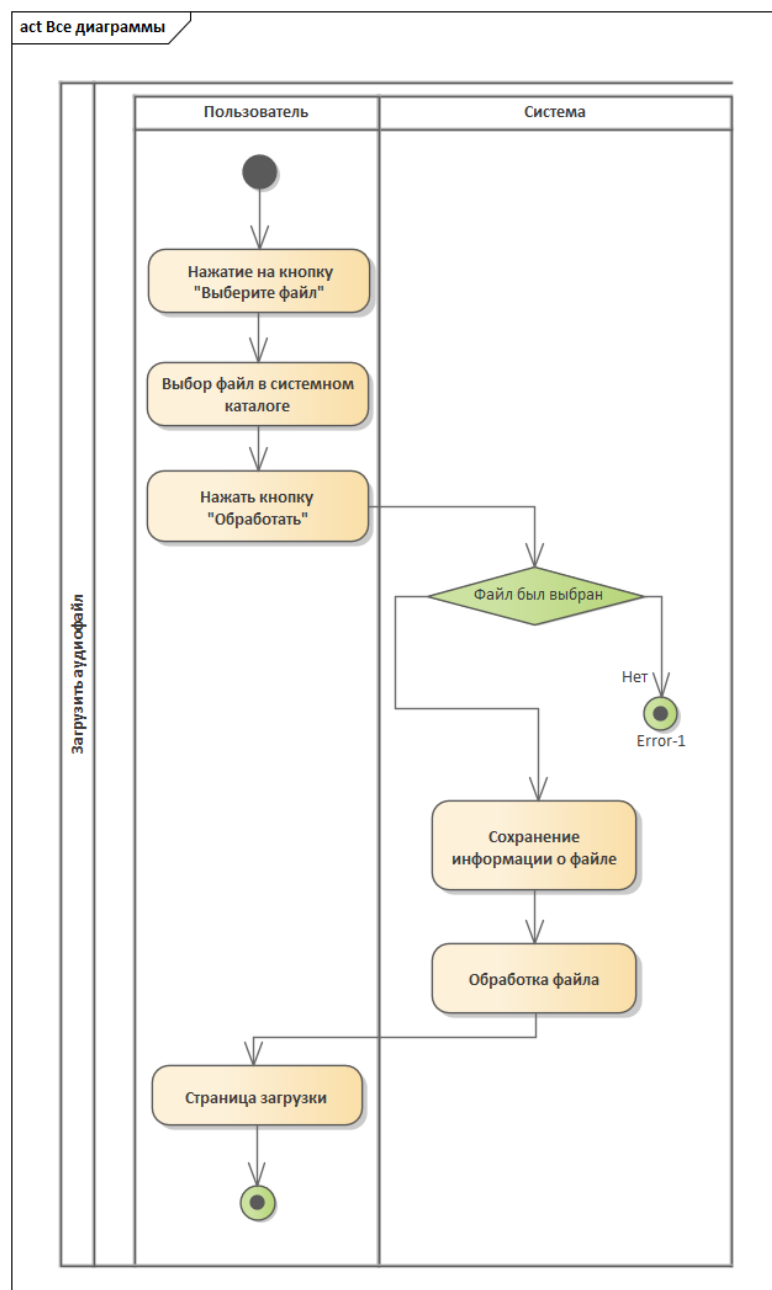


Рисунок 14. Диаграмма деятельности для прецедента «Загрузить аудиофайл»

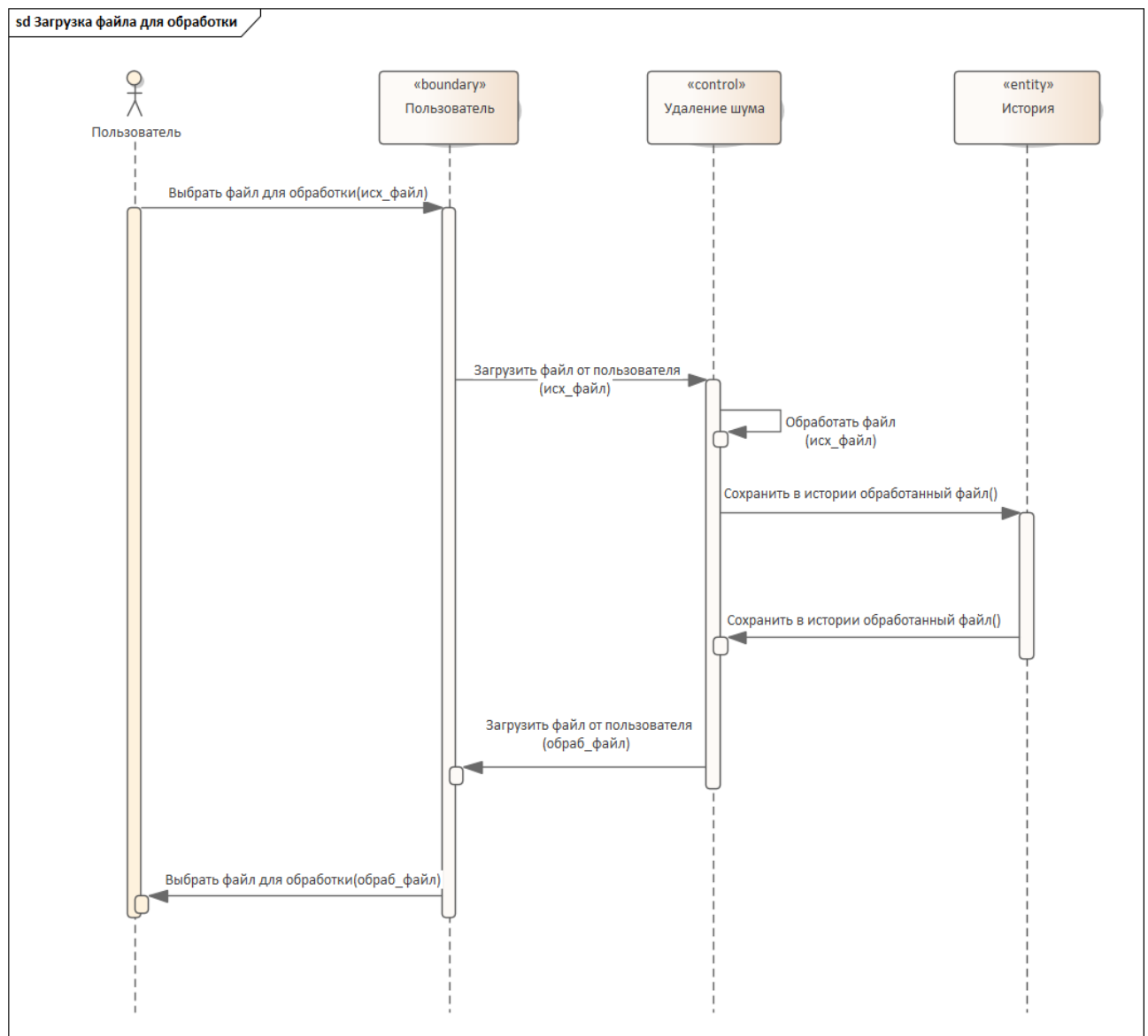


Рисунок 15. Диаграмма последовательностей для прецедента
«Загрузить аудиофайл»

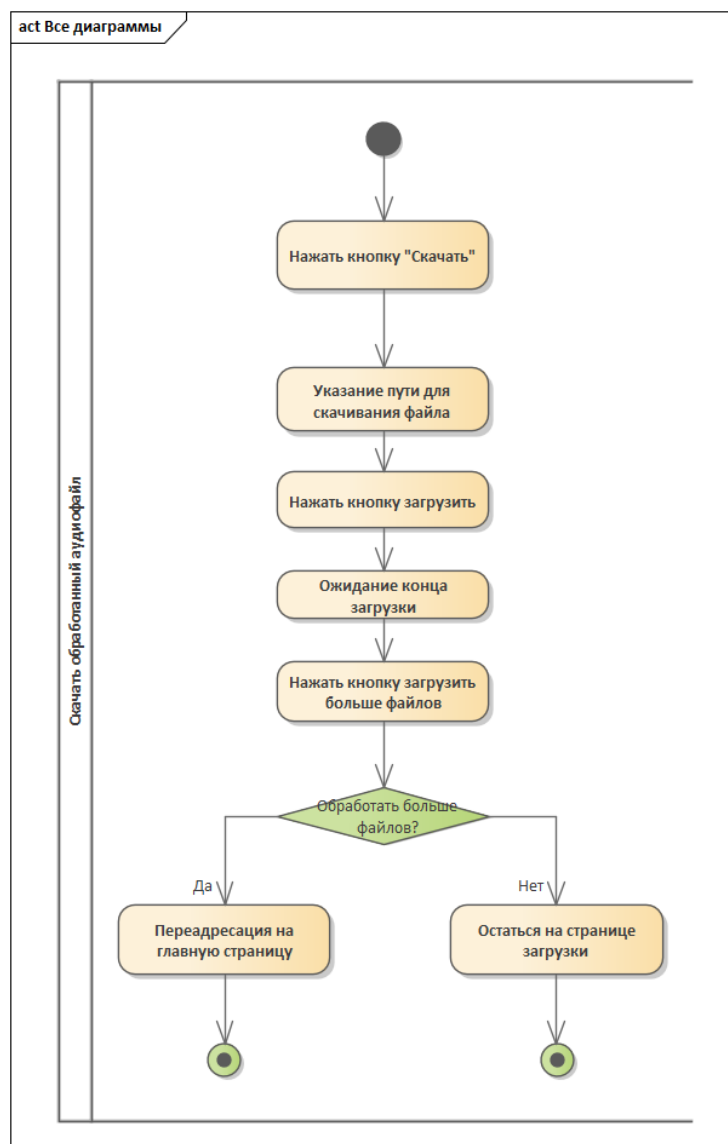


Рисунок 16. Диаграмма деятельности для прецедента
«Скачать обработанный файл»

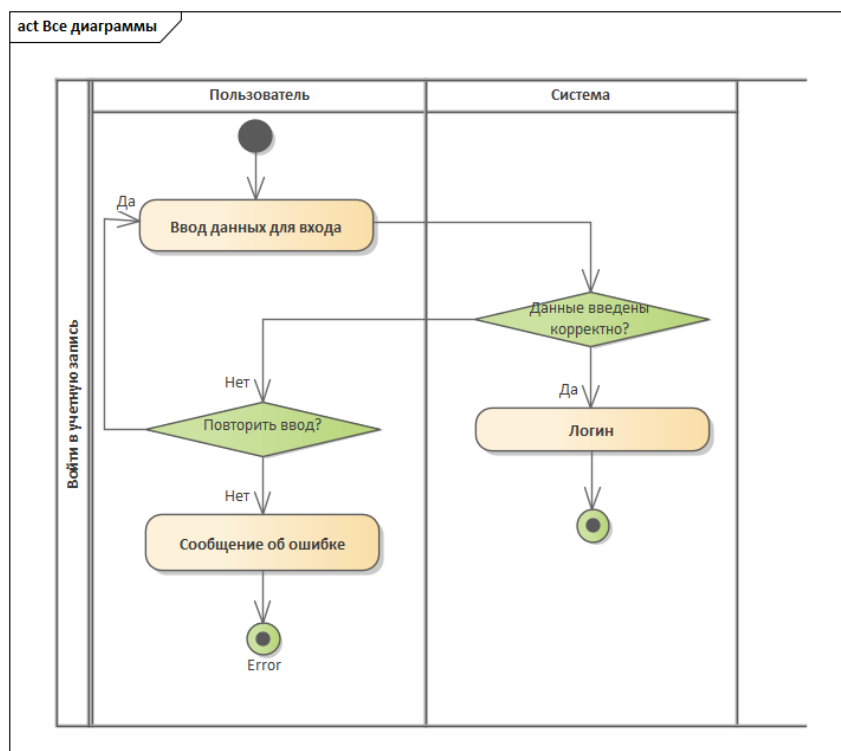


Рисунок 17. Диаграмма деятельности для прецедента «Войти в учетную запись»

2.6. Диаграммы классов участников

На рисунках 18-19 представлены диаграммы классов участников прецедентов зарегистрироваться, скачать обработанный файл.

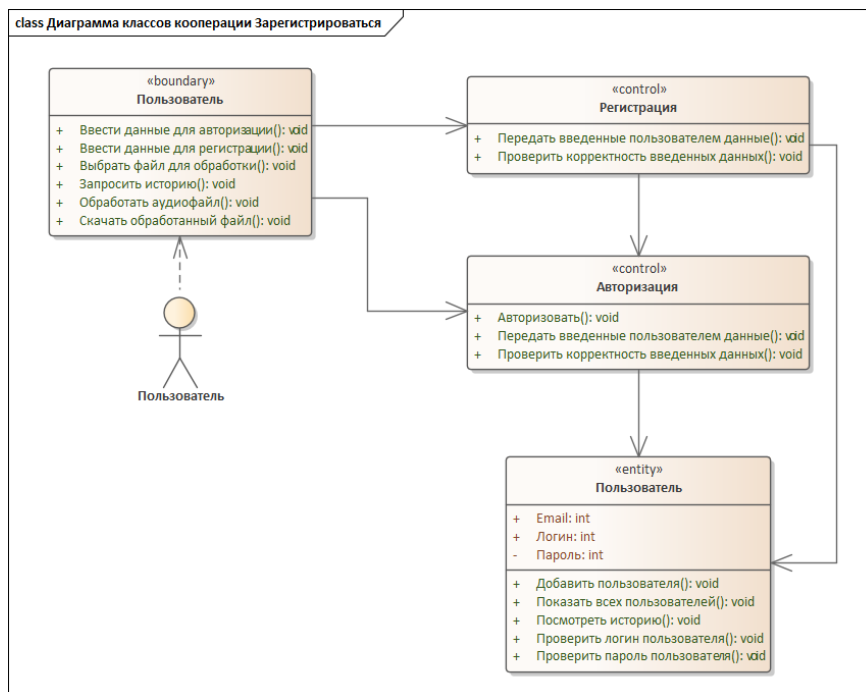


Рисунок 18. Диаграмма классов участников прецедента “Зарегистрироваться”

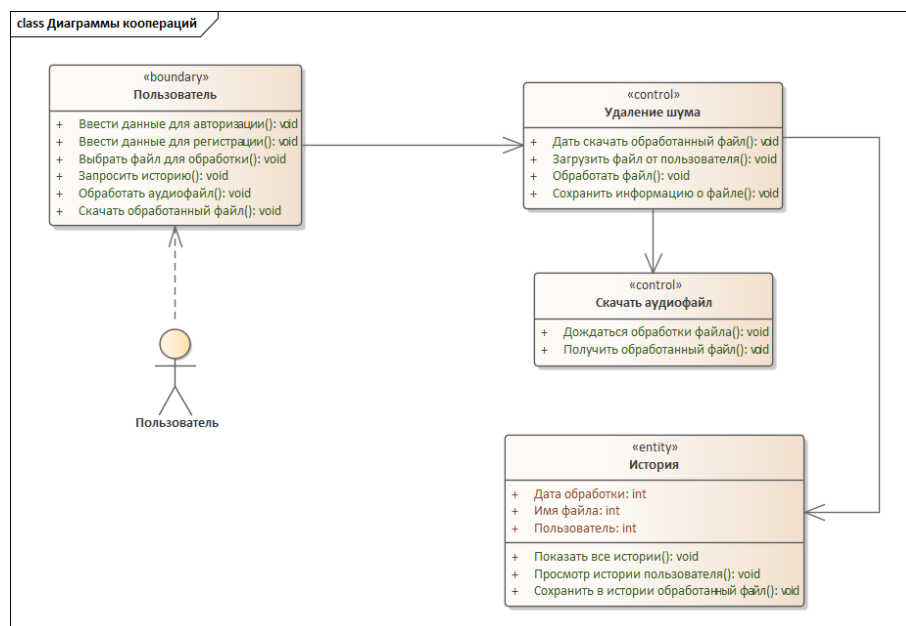


Рисунок 19. Диаграмма классов участников прецедента “Скачать обработанный файл”

2.7. Пакеты анализа и сервисные пакеты в форме обобщенной диаграммы классов

На рисунке 20 представлена диаграмма пакетов в форме обобщенной диаграммы классов. Были выявлены следующие пакеты: “Пакет пользователя”, “Пакет администратора”, “Пакет приложения пользователя”, “Пакет приложения администратора” и “Пакет данных”.

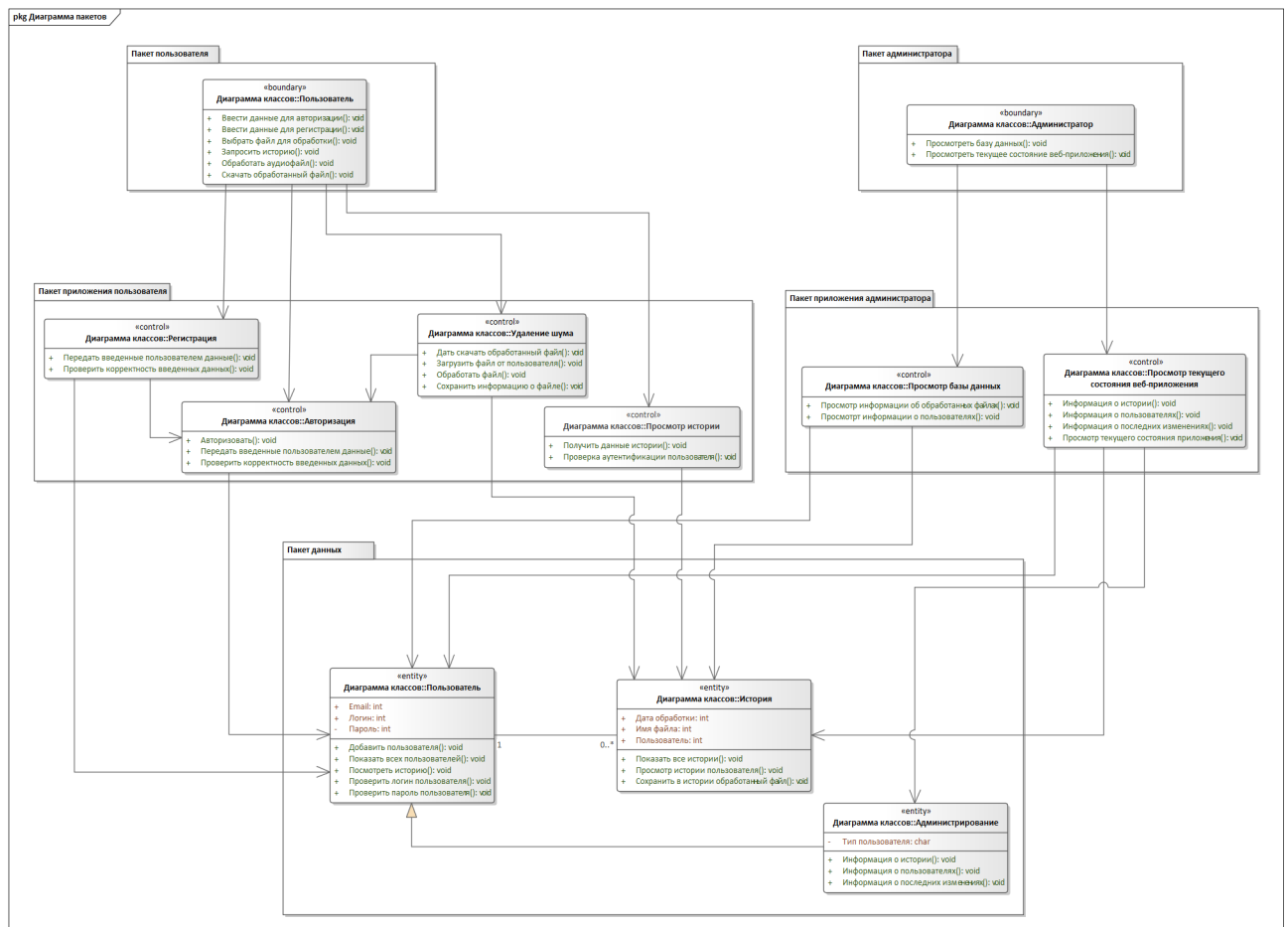


Рисунок 20. Диаграмма пакетов в форме обобщенной диаграммы классов

2.8. Трассировка пакетов в подсистемы

Диаграмма трассировки пакетов в подсистемы представлена на рисунке 21. Были выявлены 3 подсистемы: GUI, APP и Database.

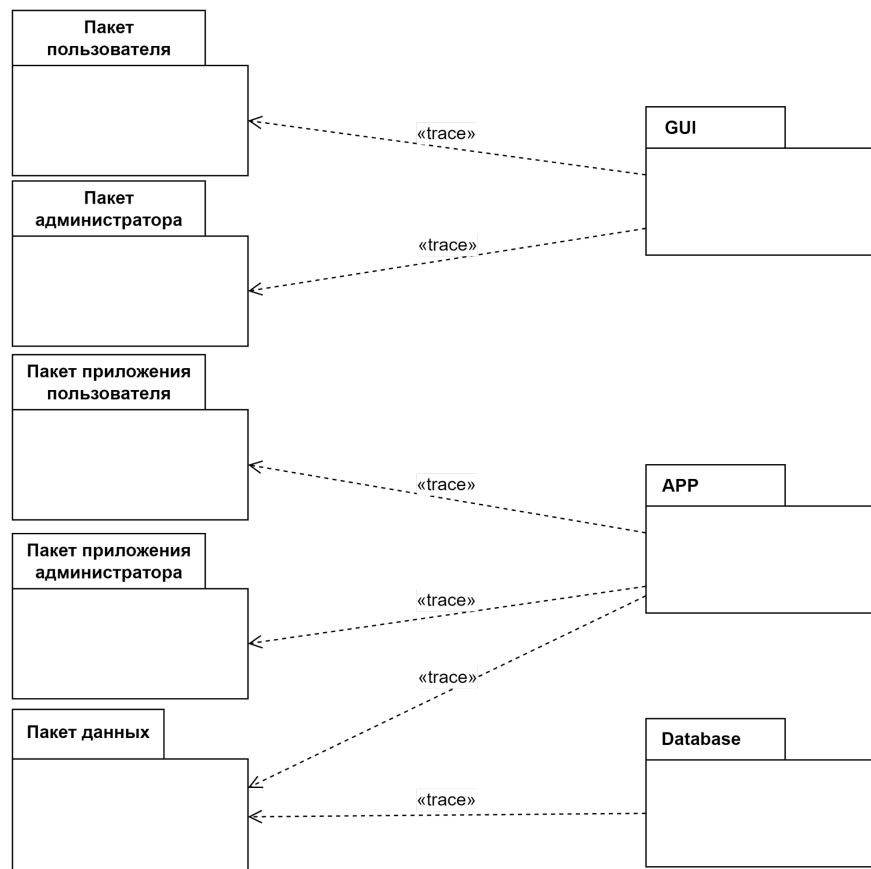


Рисунок 21. Диаграмма трассировки пакетов в подсистемы

2.9. Модель трассировки классов анализа в классы проектирования

На рисунке 22 представлена модель трассировки классов анализа в классы проектирования.

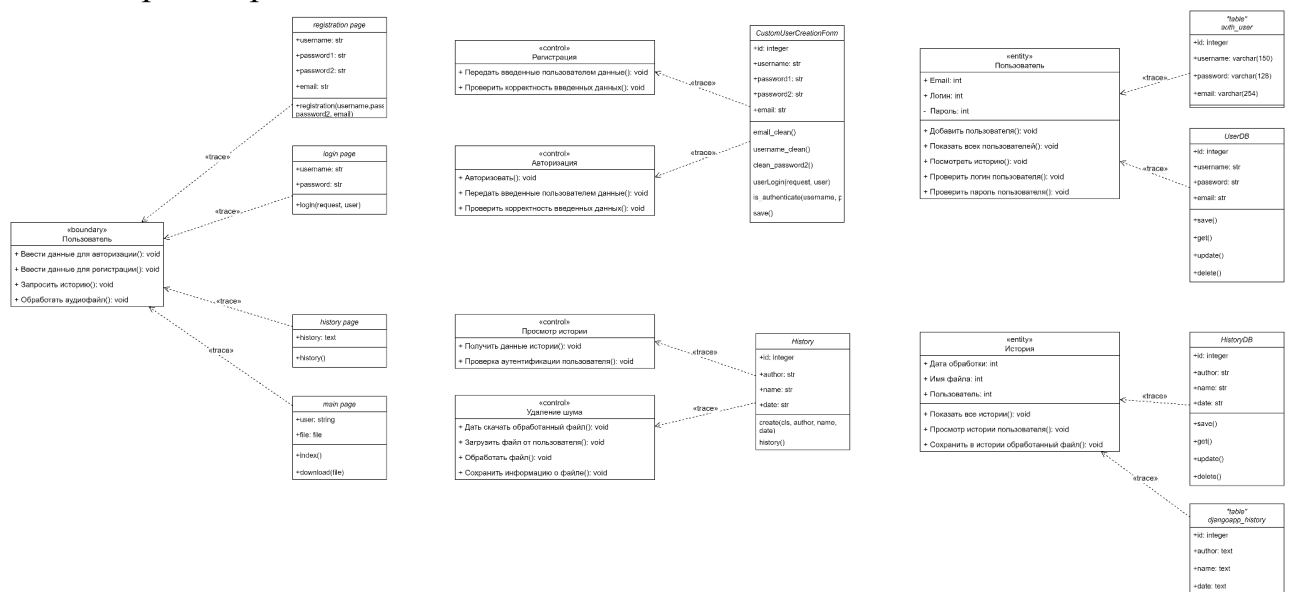


Рисунок 22. Модель трассировки классов анализа в классы проектирования

2.10. Диаграмма распределения классов проектирования по подсистемам

На рисунке 23 представлена диаграмма распределения классов проектирования по подсистемам.

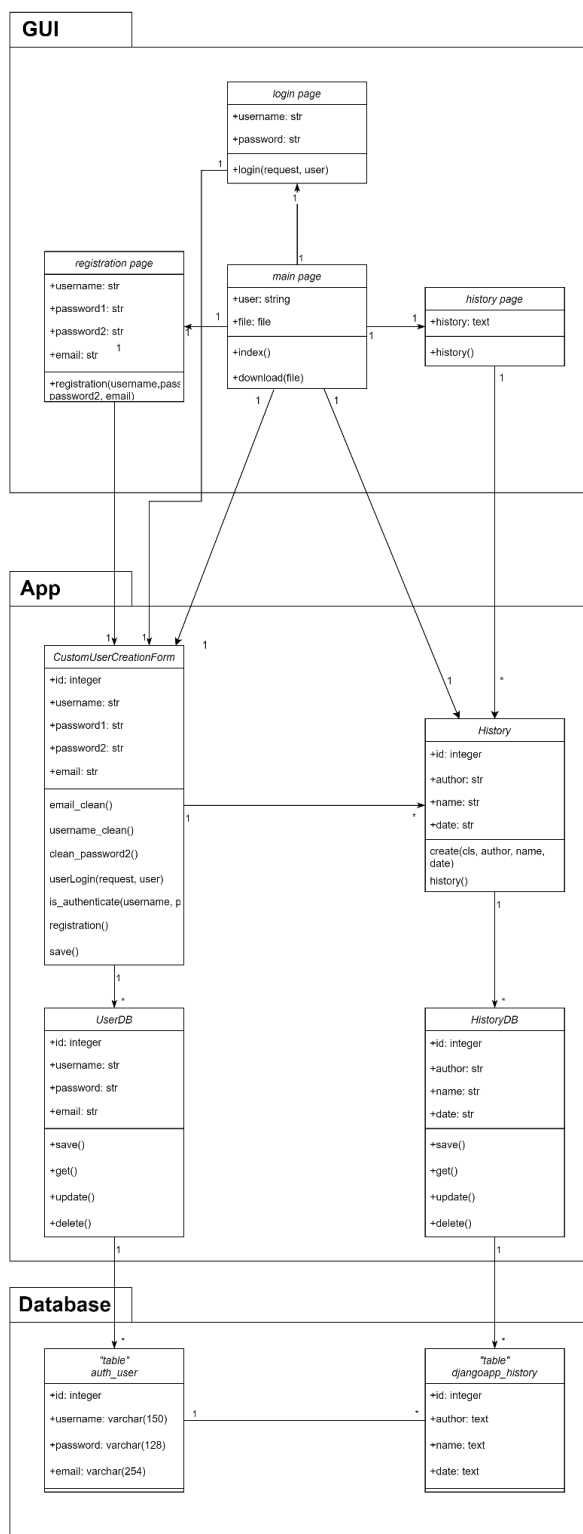


Рисунок 23. Диаграмма распределения классов проектирования по подсистемам

2.11. Диаграмма уровней подсистем

Была сформирована диаграмма уровней подсистемы (см. рисунок 24).

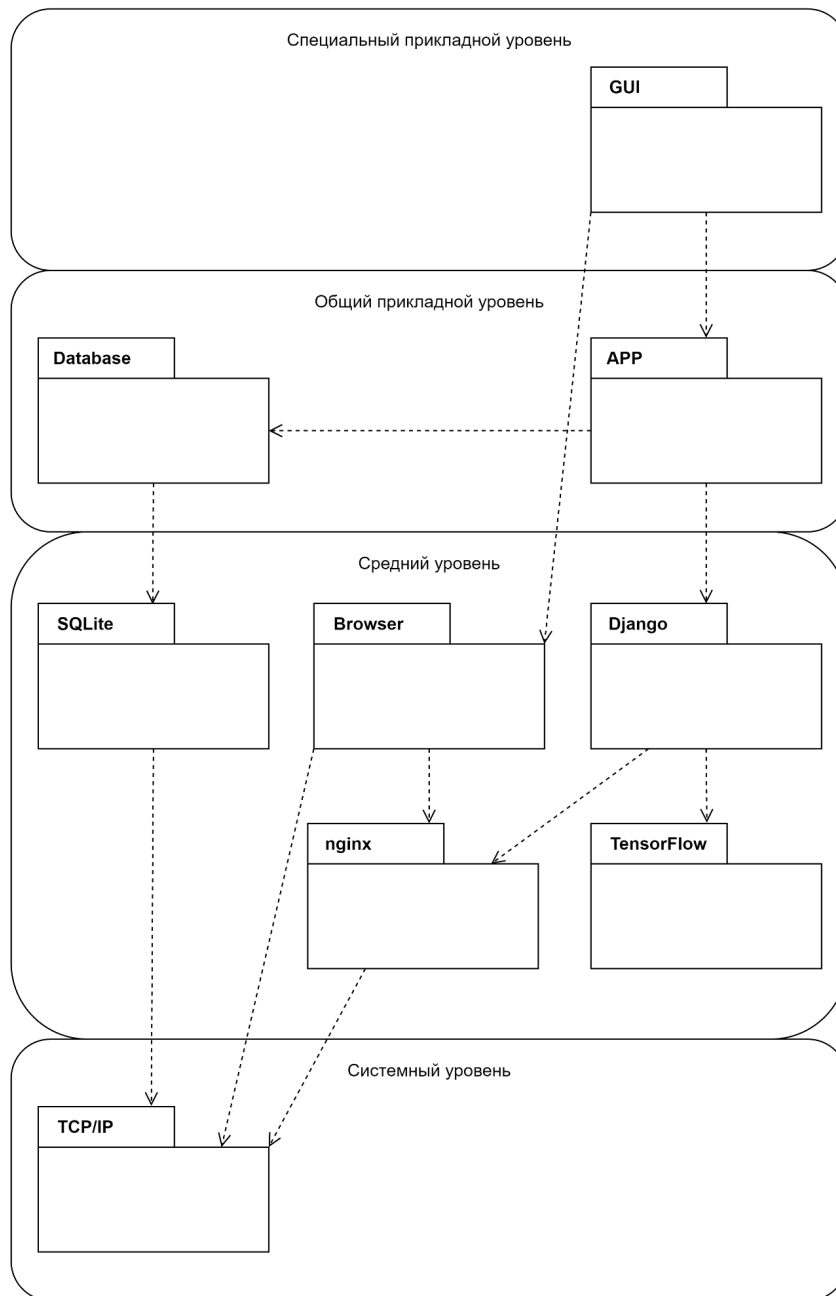


Рисунок 24. Диаграмма уровней подсистемы

2.12. Диаграмма размещения подсистем

На рисунке 25 представлена диаграмма размещения подсистем.

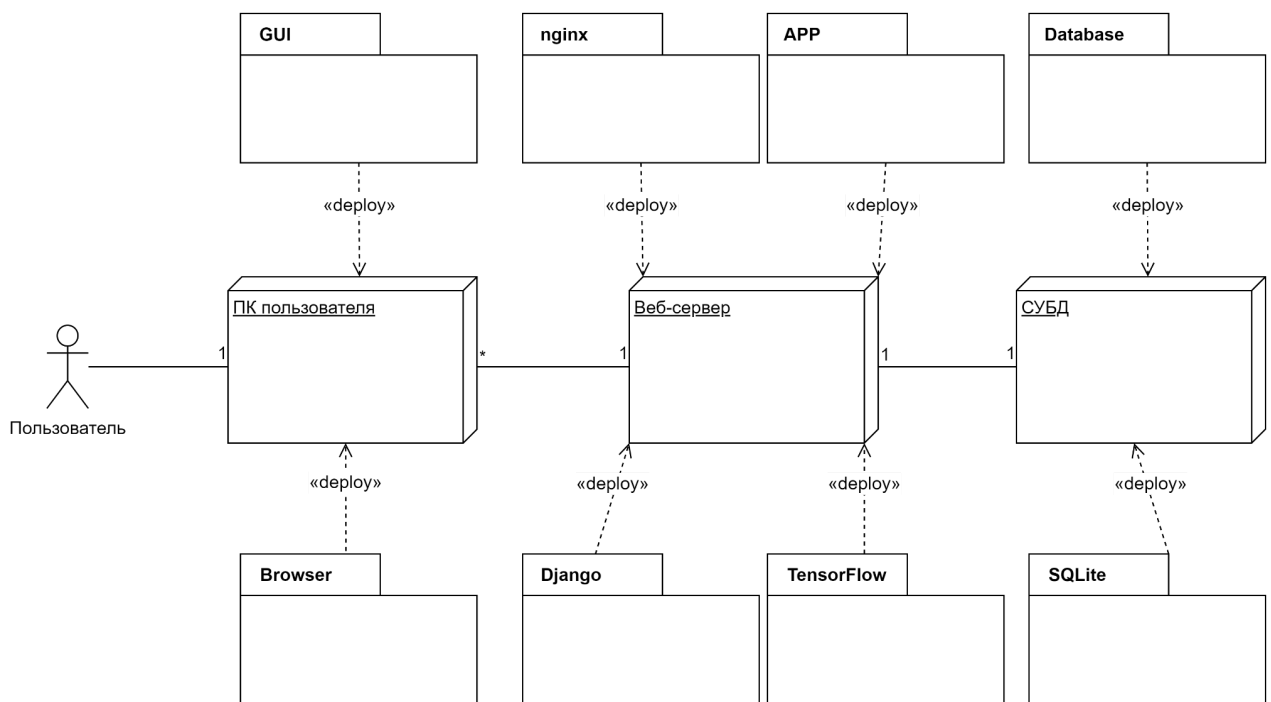


Рисунок 25. Диаграмма размещения подсистем

2.13. Модель трассировки подсистем в компоненты

На рисунке 26 показана модель трассировки подсистем в компоненты.

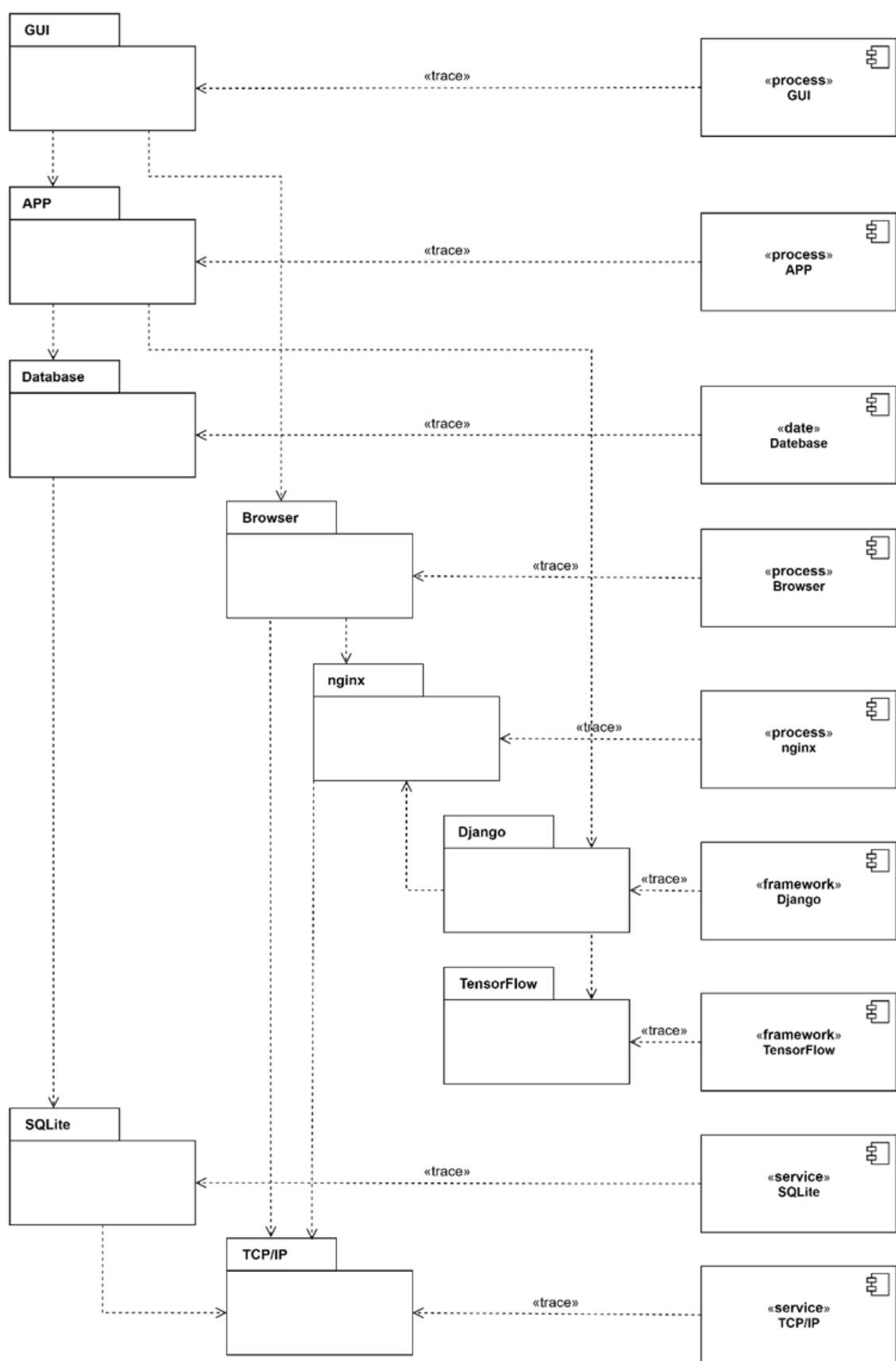


Рисунок 26. Модель трассировки подсистем в компоненты

2.14. Трассировка классов проектирования в исходные файлы

Трассировка классов проектирования в исходные файлы изображена на рисунке 27.

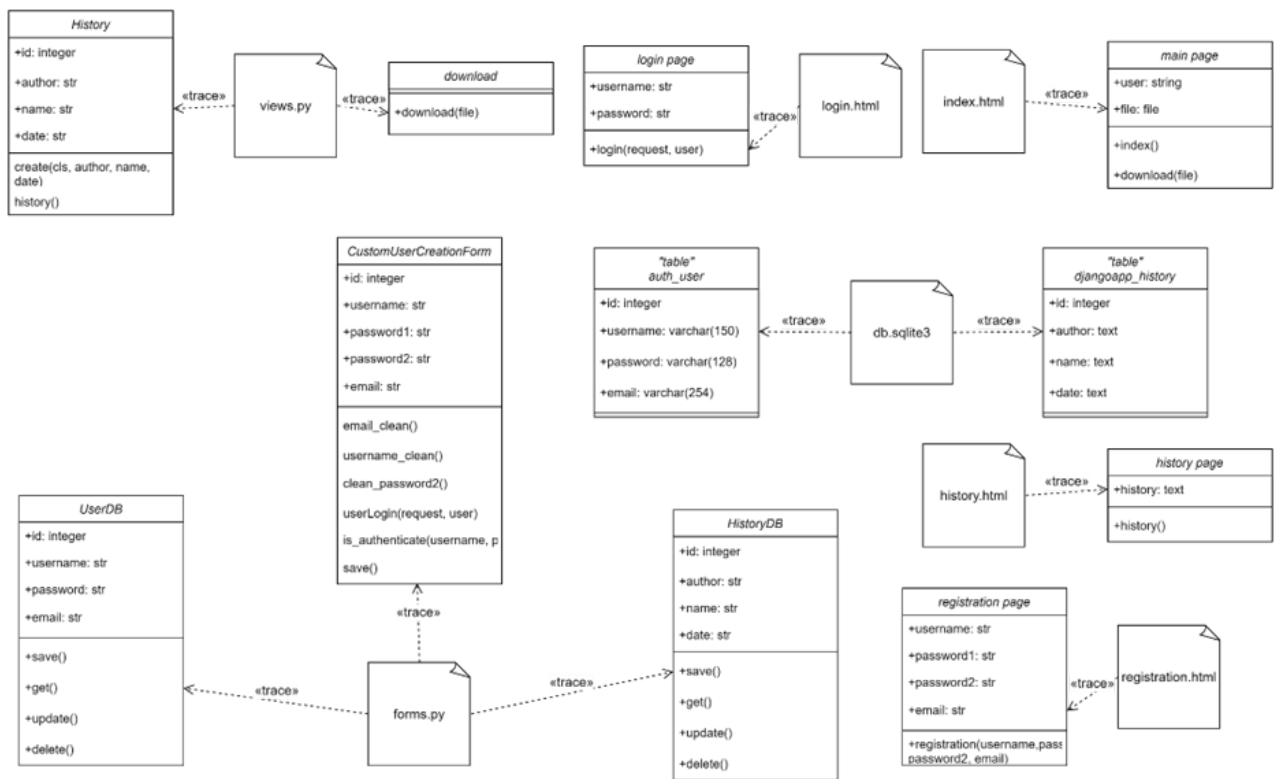


Рисунок 27. Трассировка классов проектирования в исходные файлы

2.15. Зависимость компонентов от исходных файлов

Рисунок 28 показывает зависимость компонентов от исходных файлов.

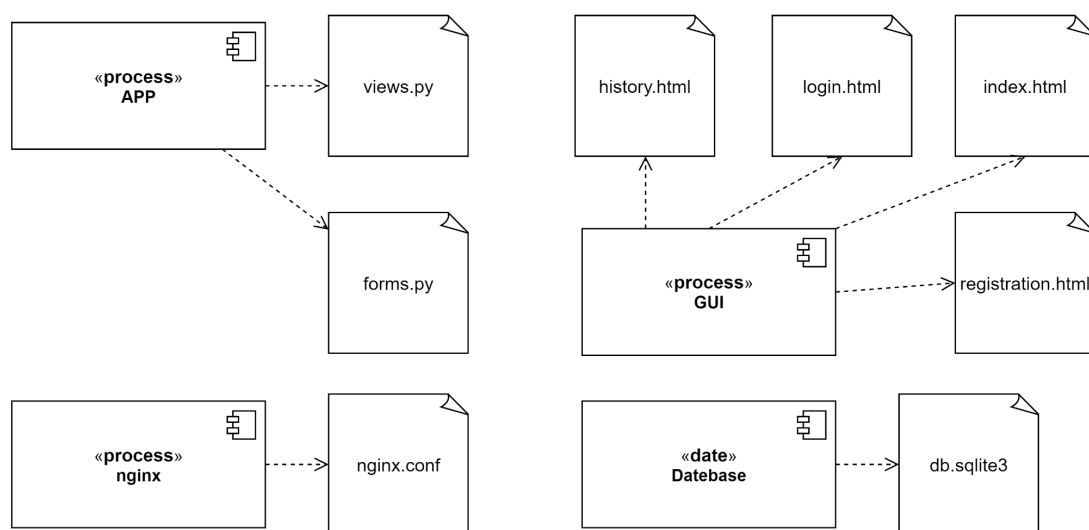


Рисунок 28. Зависимость компонентов от исходных файлов

2.16. Переработанный список рисков

В таблице 5 представлен переработанный список рисков. Был полностью переработан план управления для каждого риска.

Таблица 5. Переработанный список рисков

Риск	Переработанный план управления
Недостаточная эффективность удаления шума	<ul style="list-style-type: none">● Изучите причины шума и определите, какие источники шума могут быть устранены;● Обучите нейронную сеть на большом количестве данных;● Использовать алгоритмы обучения, которые способствуют уменьшению шума.
Потеря качества звука	<ul style="list-style-type: none">● Дообучить модель на частотах, качество которых изменилось;● Переобучить нейронную сеть с данными другой частоты.
Критическая программная ошибка	<ul style="list-style-type: none">● Разработать процедуры тестирования и контроля качества кода;● Использовать инструменты для обнаружения ошибок и автоматической проверки кода;● Внедрить процедуры резервного копирования и восстановления данных.
Низкая скорость отклика системы	<ul style="list-style-type: none">● Провести анализ производительности и оптимизировать систему для улучшения скорости отклика;● Использовать мониторинг производительности для отслеживания проблем и их устранения;● Рассмотреть возможность использования новых технологий и
Недостаточно понятный	<ul style="list-style-type: none">● Провести тестирование

пользовательский интерфейс	<p>пользовательского интерфейса на пользовательском опыте;</p> <ul style="list-style-type: none"> ● Провести исследование пользовательских потребностей для уточнения требований к интерфейсу; ● Разработать документацию и инструкции для пользователя.
Ошибка потери пользовательских данных	<ul style="list-style-type: none"> ● Разработать процедуры резервного копирования и восстановления данных; ● Использовать шифрование данных для защиты от несанкционированного доступа; ● Проводить регулярную проверку системы на уязвимости и устранять их.
Отставание по срокам	<ul style="list-style-type: none"> ● Разработать более детальный план проекта с учетом возможных задержек; ● Использовать методы управления проектами для отслеживания прогресса и сокращения задержек; ● Рассмотреть возможность увеличения ресурсов и перераспределения задач для ускорения проекта.

2.17. Перечень итераций и их состав

Итерация 0:

Прецеденты:

- Удаление шума;
- Загрузить аудиофайл;
- Скачать обработанный файл.

Пакеты:

- Пакет приложения пользователя;
- Пакет пользователя.

Подсистемы:

- GUI;
- APP.

Компоненты:

- GUI;
- APP;
- Django;
- Tensorflow.

Итерация 1:

Прецеденты:

- Зарегистрироваться;
- Войти в учетную запись;
- Просмотреть историю.

Пакеты:

- Пакет данных.

Подсистемы:

- Database.

Компоненты:

- SQLite.

3. Этап построения (Конструирования)

3.1. Экранные формы работающей программы

На рисунках 29-33 представлены экранные формы работающей программы, в которой реализованы прецеденты: загрузить аудиофайл, зарегистрироваться, войти в учетную запись, просмотреть историю, удаление шума и скачать обработанный файл.

The screenshot shows a web application interface for a logged-in user. At the top, a navigation bar includes the username 'maxzbox', links for 'Главная страница', 'О проекте', and 'История', and a 'Выход' button. The main content area is titled 'Удаление шума из аудиофайлов'. It features a file selection interface with a 'Выберите файл' button, a status indicator 'Файл не выбран', and an 'Обработать' button. Below this is an 'Инструкция по работе с системой:' section, which contains three lines of text explaining the process: uploading a .wav file, clicking 'Обработать' to see a preview, and being redirected to a download page after processing.

Рисунок 29. Главная страница для авторизованного пользователя

The screenshot shows the registration page of the web application. The navigation bar at the top has links for 'Главная страница' and 'О проекте', along with 'Войти' and 'Регистрация' buttons. The main content area contains a registration form with four input fields labeled 'Имя пользователя:', 'Пароль:', 'Повторите пароль:', and 'Email:'. A blue 'Зарегистрироваться' button is positioned below the form.

Рисунок 30. Страница «Регистрация»

Главная страница

О проекте

Войти

Регистрация

Имя пользователя:

Пароль:

Войти

Рисунок 31. Страница «Войти»

Пользователь: maxzbox

Главная страница

О проекте

История

Выход

История обработанных файлов

Пользователь	Имя файла	Дата загрузки
maxzbox	- reverb_fileid_135.wav	2022-05-19 18:50:51
maxzbox	piano.wav	2022-05-19 18:51:04
maxzbox	sample6.wav	2022-06-14 16:06:51

Рисунок 32. Страница «История»

Обработка завершена

Скачайте обработанный файл

sample6_enhanced.wav

Скачать

Обработать больше файлов

Рисунок 33. Страница «Загрузка»

3.2. Исходный код программы, реализующей архитектурно-значимые прецеденты и паттерны

Основная страница системы - index.html, состоящая из base.html и index.html. С этой страницы начинается взаимодействия пользователя с системой откуда он может получить доступ ко всем следующим страницам.

```
def index(request):
    return render(request, 'main/index.html')

def faq(request):
    return render(request, 'main/faq.html')

def download(request):

    for file in
os.scandir('D:\\Диплом\\ИУ5-83б_Назаров_М_М_Программа\\media\\'):
    if file.name.endswith(".wav"):
        os.unlink(file.path)

if request.method == 'POST':
    try:
        ticket_attachmet_audio = request.FILES["q_attachment_audio"]
    except:
        ticket_attachmet_audio = None

    if ticket_attachmet_audio == None:
```

```

        try:
            fileObj = request.FILES['file']
        except:
            return redirect('home')

fileObj = request.FILES['file']
fs = FileSystemStorage()
fs.save(fileObj.name, fileObj)
duration = torchaudio.info(
    'D:\\Диплом\\ИУ5-83б_Назаров_М_М_Программа\\media\\'+fileObj.name)
if request.user.id != None:
    size = fileObj.size / 1024 / 1024
    name = fileObj.name
    date = datetime.datetime.now()
    author = request.user
    form = History.create(author, name, str(
        date.strftime("%Y-%m-%d %H:%M:%S")))
    form.save()
fileObj1 = os.path.splitext(fileObj.name)[0] + '_enhanced.wav'
context = {
    'fileObj': fileObj1
}
os.system('python -m denoiser.enhance
--noisy_dir=D:\\Диплом\\ИУ5-83б_Назаров_М_М_Программа\\media\\
--out_dir=D:\\Диплом\\ИУ5-83б_Назаров_М_М_Программа\\media ')
return render(request, 'main/download.html', context)

class registration(View):
    template_name = 'main/registration/registration.html'

    def get(self, request):
        context = {
            'form': CustomUserCreationForm()
        }
        return render(request, self.template_name, context)

    def post(self, request):
        form = CustomUserCreationForm(request.POST)

        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password1')
            '''user =
CustomUserCreationForm.is_authenticate(username=username,
password=password)'''
            CustomUserCreationForm.userLogin(request,

```

```

user=CustomUserCreationForm.is_authenticate(
    username=username, password=password))
    return redirect('home')
context = {
    'form': form
}
return render(request, self.template_name, context)

```

```

def handle_not_found(request, exception):
    return render(request, 'main/not-found.html', status=404)

```

Код страницы history.html, которая отвечает за вывод истории авторизованного пользователя системы.

```

{% block css_additional %}
<link rel="stylesheet" href="../../static/main/css/history.css">
{% endblock %}

{% block title %}
История
{% endblock %}

{% block content %}
<div>{{id}}</div>
<h1>История обработанных файлов</h1>
<table class="table table-bordered">
    <tr>
        <th scope="col">Пользователь</th>
        <th scope="col">Имя файла</th>
        <th scope="col">Дата загрузки</th>
    </tr>
{% for history in object_list %}
    {% if history.author.id == user.id %}
    <tr>
        <td>{{history.author}}</td>
        <td>{{history.name}}</td>
        <td>{{history.date}}</td>
    </tr>
    {% endif %}
{% endfor %}
</table>
{% endblock %}

```

3.2.1. Реализация классов Domain model

Класс доменной модели CustomUserCreationForm отвечает за регистрацию пользователя. Содержит следующие атрибуты: username, email, password1, password2.

```
class CustomUserCreationForm(UserCreationForm):
    username = forms.CharField(
        label='Имя пользователя', min_length=5, max_length=150)
    email = forms.EmailField(label='Email')
    password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput)
    password2 = forms.CharField(
        label='Повторите пароль', widget=forms.PasswordInput)

    def username_clean(self):
        username = self.cleaned_data['username'].lower()
        new = UserDB.get(username=username)
        if new.count():
            raise ValidationError(
                "Пользователь с таким именем уже зарегистрирован")
        return username

    def email_clean(self):
        email = self.cleaned_data['email'].lower()
        new = UserDB.get(email=email)
        if new.count():
            raise ValidationError(
                "Пользователь с таким email уже зарегистрирован")
        return email

    def clean_password2(self):
        password1 = self.cleaned_data['password1']
        password2 = self.cleaned_data['password2']

        if password1 and password2 and password1 != password2:
            raise ValidationError("Пароли не совпадают")
        return password2

    def save(self, commit=True):
        user = UserDB(0, self.cleaned_data["username"],
                      self.cleaned_data["email"],
self.cleaned_data["password1"])
        user.save()

    @staticmethod
    def is_authenticate(username, password1):
        username1 = UserDB.get(username=username)
        password = UserDB.get(password=password1)
        user = authenticate(username=username1, password=password)
```



```

        return user

    def userLogin(self, request, user):
        loginUser = login(request=request, user=user)
        return loginUser

```

Класс доменной модели History отвечает за отображение истории для авторизованных пользователей. Содержит атрибуты: id, author, name, date.

```

class history(ListView):
    model = History
    template_name = 'main/history.html'
    def history(request):
        dataHistory=HistoryDB.get(id=request.user.id)
        history= {
            'id' : dataHistory.id,
            'author' : dataHistory.author,
            'name' : dataHistory.name,
            'date' : dataHistory.date
        }
        return render(request, 'main/history.html', history)

```

3.2.2. Реализация классов Active Record

Класс активной записи UserDB служит для взаимодействия с таблицей БД auth_user.

```

class UserDB(CustomUserCreationForm):

    def __init__(self, id, username, email, password):
        self.id = id
        self.username = username
        self.password = password
        self.email = email

    def save(self, commit=True):
        user = User.objects.create_user(
            self.username, self.email, self.password)
        return user

    def get(self):
        user = User.objects.filter(id=self.id)
        return user

    def update(self):
        user = User.objects.filter(id=self.id).update(
            username=self.username, email=self.email, password=self.password)

```

```
        return user

def delete(self):
    user = User.objects.filter(id=self.id).delete()
    return user
```

Класс активной записи HistoryDB служит для взаимодействия с таблицей БД djangoapp_history.

```
class HistoryDB(History):

    def __init__(self, author, name, date):
        self.id = id
        self.author = author
        self.name = name
        self.date = date

    def create(self, author, name, date, commit=True):
        history = history.objects.create(
            author, name, date)
        return history

    def get(self):
        history = history.objects.filter(id=self.id)
        return history

    def update(self):
        history = history.objects.filter(id=self.id).update(
            author=self.author, name=self.name, date=self.date)
        return history

    def delete(self):
        history = history.objects.filter(id=self.id).delete()
        return history
```

3.3. Полная диаграмма классов реализации

На рисунке 34 представлена диаграмма классов системы.



Рисунок 34. Диаграмма классов системы.

3.4. Диаграммы последовательностей для иллюстрации работы паттернов

На рисунке 35 представлена диаграмма последовательностей для основных функций программы.

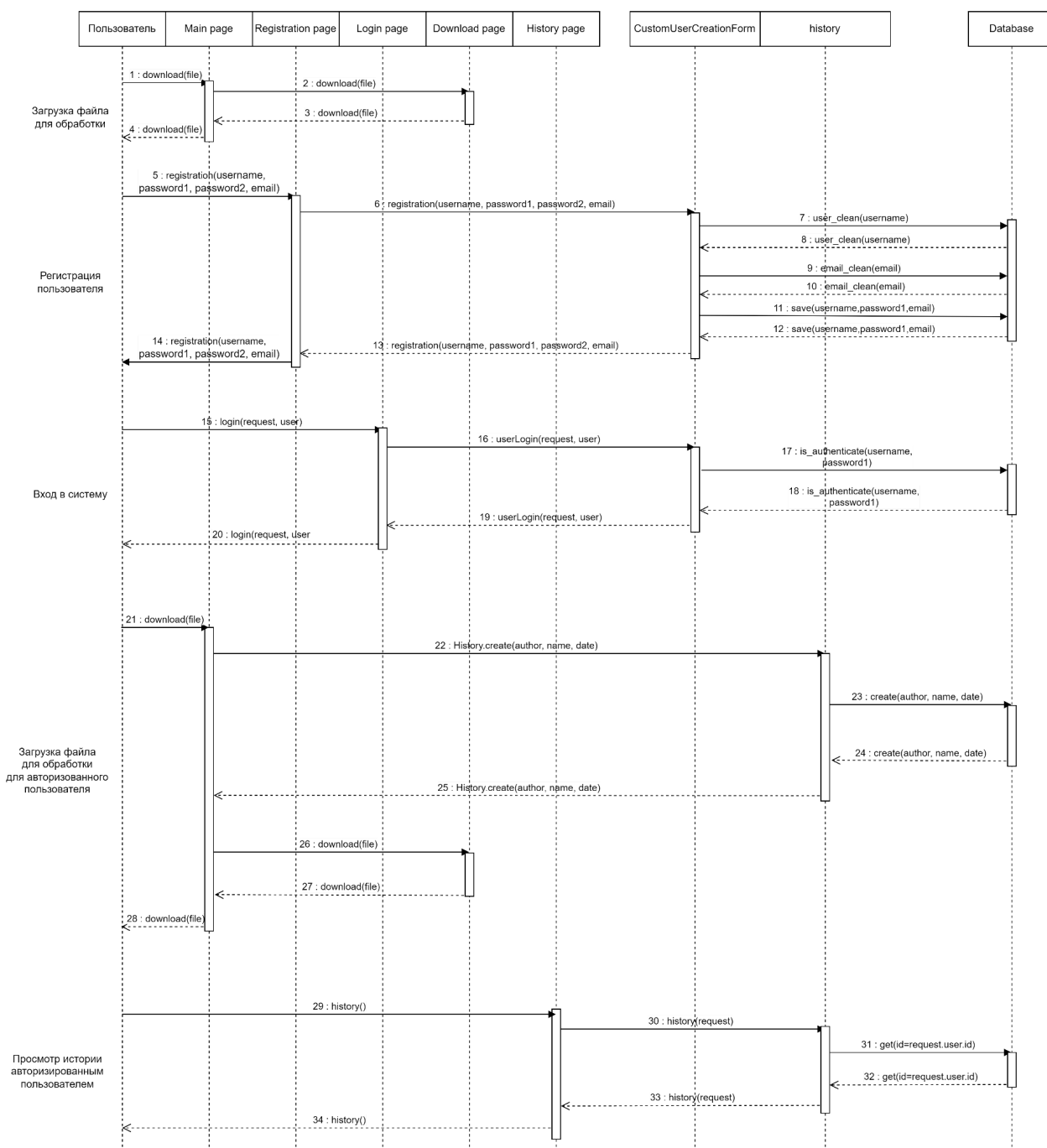


Рисунок 35. Диаграмма последовательностей для основных функций программы.

Рисунок 36 иллюстрирует работу прецедента «Зарегистрироваться».

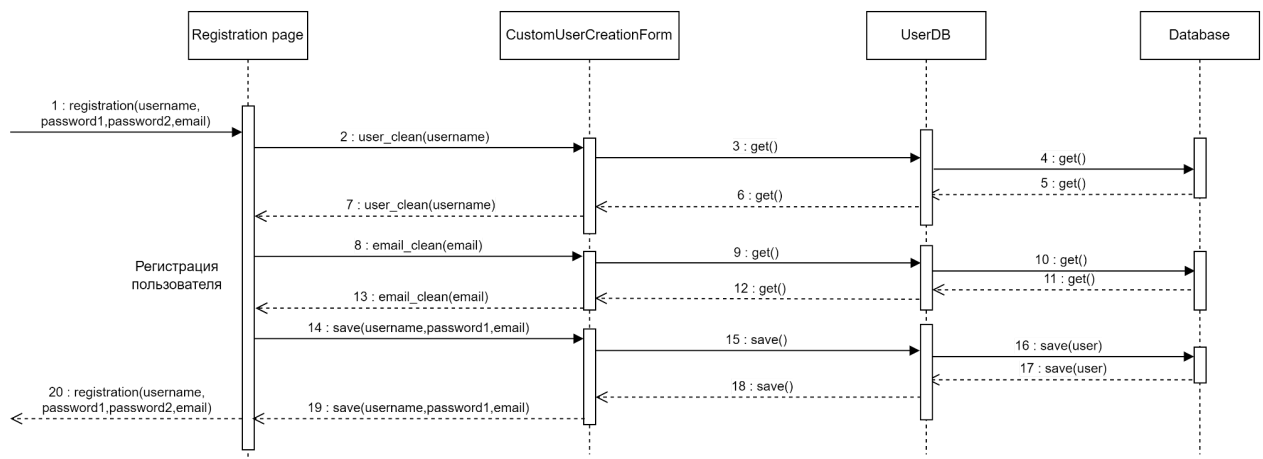


Рисунок 36. Диаграмма последовательностей для прецедента
«Зарегистрироваться».

На рисунке 37 изображает диаграмма последовательностей для прецедента «Просмотреть историю».

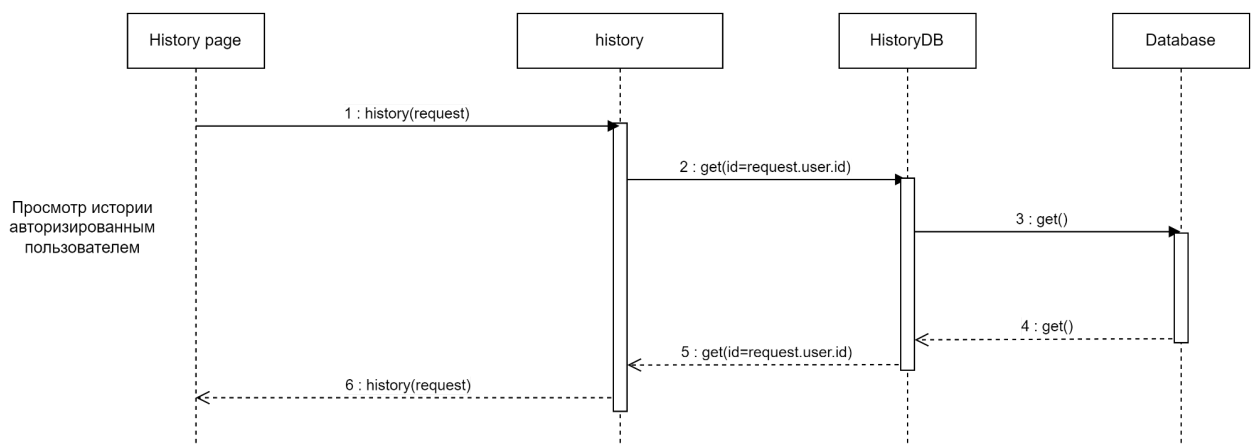


Рисунок 37. Диаграмма последовательностей для прецедента «Просмотреть историю».

3.5. Перечень и последовательность проведения тестов

Тестирование реализованных прецедентов будет проводится в виде модульных тестов и интеграционных.

Модульные тесты:

- Первое тестирование проверяет, что поле username обязательно для заполнения (тестирования CustomUserCreationForm .username_clean());
- Второй тест проверяет, что поле email обязательно для заполнения

(тестирования CustomUserCreationForm.email_clean());

- Третье тестирование проверяет методы username_clean(), email_clean() и clean_password2() класса CustomUserCreationForm;
- Четвертый тест проверяет, что поле username обязательно для заполнения. Он создает экземпляр формы с некорректными данными и проверяет, что форма не проходит валидацию и выводится ошибка “Обязательное поле.”;
- Пятый тест проверяет, что поле email обязательно для заполнения. Он создает экземпляр формы с некорректными данными и проверяет, что форма не проходит валидацию и выводится ошибка “Обязательное поле.”.

После успешного завершения модульных теста необходимо провести интеграционное и функциональное тестирование:

- Тестирование реализованного функционала регистрации.
Интеграция модулей регистрации, авторизации.

Необходимо выполнять подобные тесты на каждом этапе реализации программы. После завершения каждой итерации следует проводить модульное тестирование разработанного модуля, а после завершения всех итераций этапа проводить интеграционное тестирование всех модулей.

После завершения разработки системы и успешного прохождения модульных и интеграционных тестирований, следует выполнить системное тестирование.

В ходе системного тестирования необходимо проверить работу системы в целом, то есть проверить: работает ли удаление шума, загрузка аудиофайла, скачивание обработанного аудиофайла, регистрация, авторизация и работа истории обработки.

3.6. Примеры модульных и функциональных тестов

Модульное тестирование работы класса CustomUserCreationForm. Ниже приведен код данного класса

```
class CustomUserCreationForm(UserCreationForm):
    username = forms.CharField(
        label='Имя пользователя', min_length=5, max_length=150)
    email = forms.EmailField(label='Email')
    password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput)
    password2 = forms.CharField(
        label='Повторите пароль', widget=forms.PasswordInput)

    def username_clean(self):
        username = self.cleaned_data['username'].lower()
        new = UserDB.get(username=username)
        if new.count():
            raise ValidationError(
                "Пользователь с таким именем уже зарегистрирован")
        return username

    def email_clean(self):
        email = self.cleaned_data['email'].lower()
        new = UserDB.get(email=email)
        if new.count():
            raise ValidationError(
                "Пользователь с таким email уже зарегистрирован")
        return email

    def clean_password2(self):
        password1 = self.cleaned_data['password1']
        password2 = self.cleaned_data['password2']

        if password1 and password2 and password1 != password2:
            raise ValidationError("Пароли не совпадают")
        return password2

    def save(self, commit=True):
        user = UserDB(0, self.cleaned_data["username"],
                      self.cleaned_data["email"], self.cleaned_data["password1"])
        user.save()

    @staticmethod
    def is_authenticate(username, password1):
        username1 = UserDB.get(username=username)
        password = UserDB.get(password=password1)
        user = authenticate(username=username1, password=password)
        return user

    def userLogin(self, request, user):
```

```
loginUser = login(request=request, user=user)
return loginUser
```

Ниже приведен код тестов для класса CustomUserCreationForm:

```
from django.test import TestCase
from .forms import CustomUserCreationForm

class TestCustomUserCreationForm(TestCase):
    def test_username_clean(self):
        form = CustomUserCreationForm({'username': 'testuser', 'email':
'testuser@example.com', 'password1': 'testpass123', 'password2': 'testpass123'})
        self.assertTrue(form.is_valid())

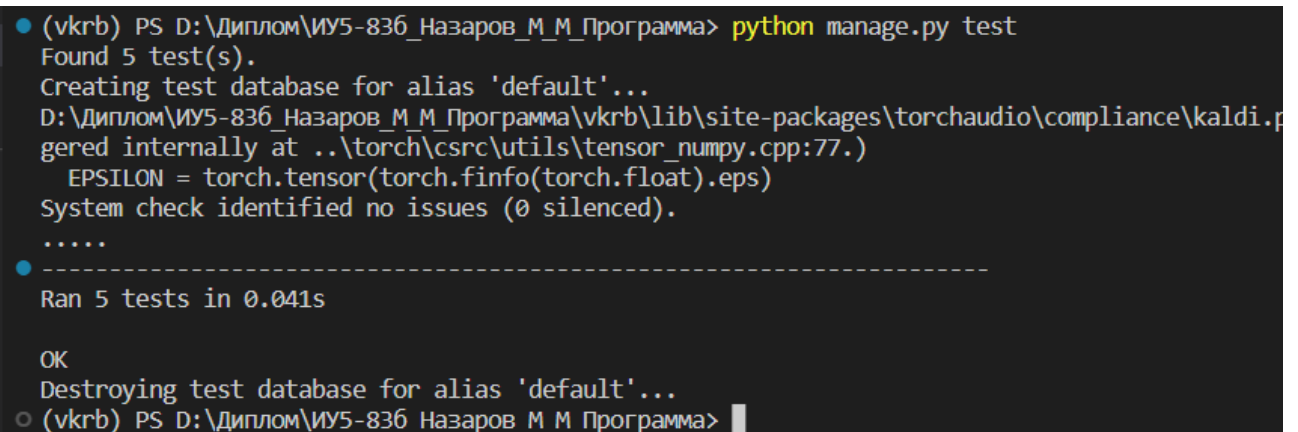
    def test_email_clean(self):
        form = CustomUserCreationForm({'username': 'testuser', 'email':
'testuser@example.com', 'password1': 'testpass123', 'password2': 'testpass123'})
        self.assertTrue(form.is_valid())

    def test_clean_password2(self):
        form = CustomUserCreationForm({'username': 'testuser', 'email':
'testuser@example.com', 'password1': 'testpass123', 'password2': 'testpass123'})
        self.assertTrue(form.is_valid())

    def test_username_required(self):
        form = CustomUserCreationForm({'email': 'testuser@example.com',
'password1': 'testpass123', 'password2': 'testpass123'})
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['username'], ['Обязательное поле.'])

    def test_email_required(self):
        form = CustomUserCreationForm({'username': 'testuser', 'password1':
'testpass123', 'password2': 'testpass123'})
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['email'], ['Обязательное поле.'])
```

Результат выполнения тестов приведен на рисунке 38.



```
● (vkrb) PS D:\Диплом\ИУ5-836_Назаров_М_М_Программа> python manage.py test
Found 5 test(s).
Creating test database for alias 'default'...
D:\Диплом\ИУ5-836_Назаров_М_М_Программа\vkrb\lib\site-packages\torchaudio\compliance\kaldi.p
gered internally at ..\torch\csrc\utils\tensor_numpy.cpp:77.)
  EPSILON = torch.tensor(torch.finfo(torch.float).eps)
System check identified no issues (0 silenced).
.....
● -----
Ran 5 tests in 0.041s

OK
Destroying test database for alias 'default'...
○ (vkrb) PS D:\Диплом\ИУ5-836_Назаров_М_М_Программа> █
```

Рисунок 38. Результаты модульного тестирования.

Функциональное тестирование интеграции модели, реализующих функционал регистрации и авторизации, было выполнено через интерфейс пользователя на локально запущенном сервере. В таблице 6 представлены тестовые варианты.

Таблица 6. Результаты функционального тестирования

Действие(входные значения)	Ожидаемый результат
Страница регистрации: оставить пустыми все поля. Нажать кнопку зарегистрироваться.	Вывод подсказки с указанием заполнить поле.
Страница регистрации: ввести в поле имени пользователя admin (уже существующий пользователь), оставшиеся поля заполнить правильно. Нажать кнопку зарегистрироваться.	Вывод текста о существовании пользователя с таким паролем.
Страница регистрации: ввести разные пароли в полях пароль и повторите пароль, оставшиеся поля заполнить нормально. Нажать кнопку зарегистрироваться.	Вывод текста о несовпадении паролей.
Страница регистрации: ввести в поле имени Email admin1@mail.ru (уже существующий пользователь с таким email'ом), оставшиеся поля заполнить правильно. Нажать кнопку зарегистрироваться.	Вывод текста о существовании пользователя с такой электронной почтой.
Страница регистрации: заполнить все поля правильно. Нажать кнопку зарегистрироваться.	Регистрация и переадресация на главную страницу. Указание имени пользователя в шапки сайта.
Страница войти: указать неверные данные. Нажать кнопку войти.	Вывод текста на экран: Пожалуйста, введите правильные имя пользователя и пароль. Оба поля могут быть чувствительны к регистру
Страница войти: заполнить все поля правильно. Нажать кнопку войти.	Авторизация и переадресация на главную страницу. Указание имени пользователя в шапки сайта.

4. Этап внедрения

4.1. Перечень программ и рекомендаций по установке

Для работы системы удаления шума со стороны пользователя необходимы:

- Стабильное подключение к сети интернет;
- Браузер на ядре Chromium.

Для работы системы удаления шума необходима следующая возможная конфигурация сервера:

- Интернет: подключение к интернету;
- ОС: Windows 10+, Ubuntu LTS2020+;
- ОЗУ: >2 ГБ;
- SSD: 4 ГБ свободного места.

4.2. Перечень документации для пользователей и заказчиков

Перечень документации для заказчика:

- Техническое задание;
- Программа и методика испытаний;
- Описание программы.

4.3. Перечень документации для пользователя

Перечень документации для пользователя:

- Руководство пользователя.

4.4. Рекомендации по внедрению

Для улучшения понимания работы системы пользователем можно записать серию обучающих видеороликов по работе с системой.

Заключение

В результате выполнения курсовой работы была сделана рабочая программа для удаления шума из речи. Для ее создания были выполнены следующие этапы: анализ и планирование требований, проектирование, построение и внедрение. В качестве паттерна бизнес-логики использовался domain model, а для работы с базой данных - active record.

Программа была написана на языке Python с использованием библиотеки TensorFlow. Был написан сайт на Django для удобного использования программы.

Список литературы

1. Унифицированный процесс разработки программного обеспечения: учебное пособие / Виноградова М.В., Белоусова В.И. – М.: МГТУ им.Н.Э. Баумана. – 2015 г.
2. Орлов С.А. Технологии разработки программного обеспечения. - СПб: Питер, - 2002 г
3. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования. - СПб.: Питер. - 2009 г.
4. Django documentation. – Текст. Изображение: электронные // Django: [сайт]. – URL: <https://docs.djangoproject.com/en/4.1/> (Дата обращения: 23.03.2023)
5. Фаулер М. Архитектура корпоративных приложений. - М.:Изд.дом Вильямс. - 2008 г.