Санкт-Петербургский государственный политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
«Высшая школа программной инженерии»

ОТЧЁТ ПО ВТОРОЙ ЧАСТИ КУРСОВОГО ПРОЕКТА

High-Level Design проекта «Lazy Lecture»

Выполнили студенты
гр. 5130904/10102

Богданов Н.Р.
Головатюков С.А.
Мирончук Ю.Б.
Рыженко Д.А.
Тампио И.С.

Принял работу
преподаватель

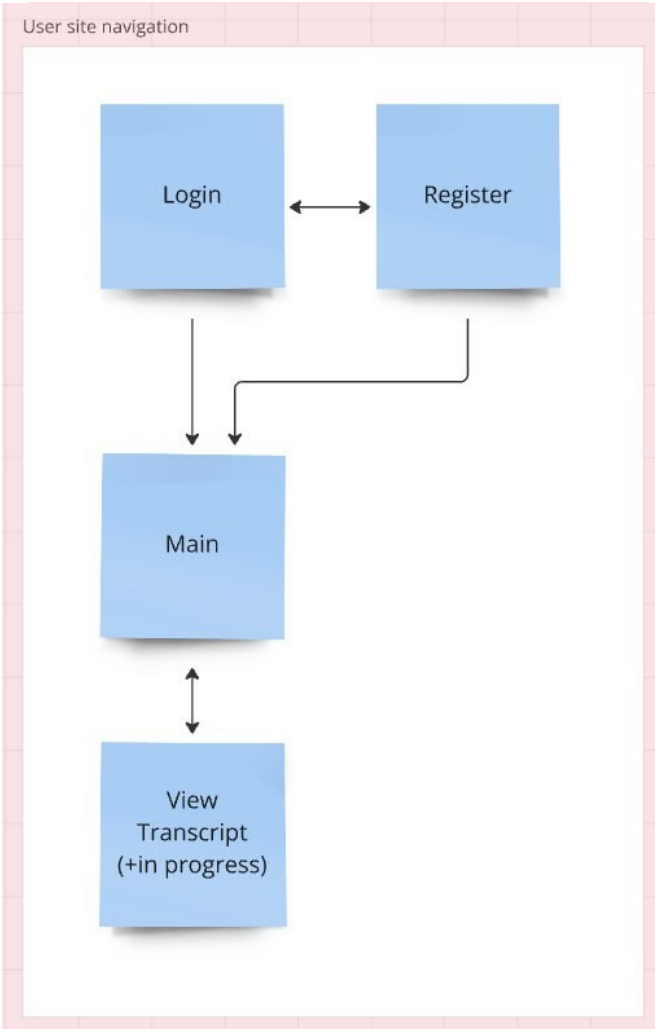
Маслаков А.П.

Санкт-Петербург
2024 г.

Задание:

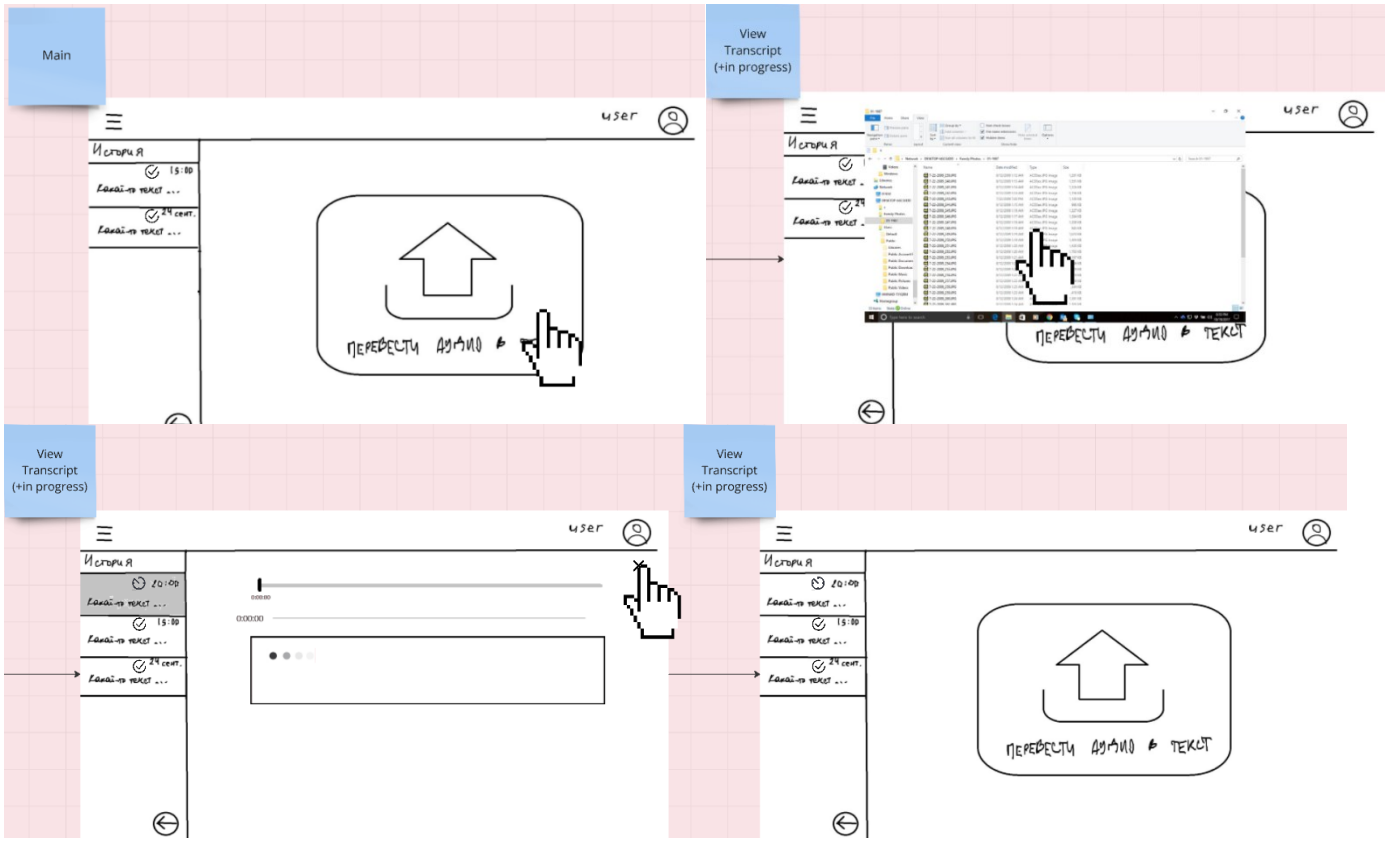
Первый вариант документа High-Level Design (HLD). В документе должны быть описаны макет дизайна интерфейса (при наличии), общая архитектура приложения, стэк технологий, спроектированы диаграммы классов/модулей/etc, схемы баз данных, API. Все схемы должны сопровождаться описанием.

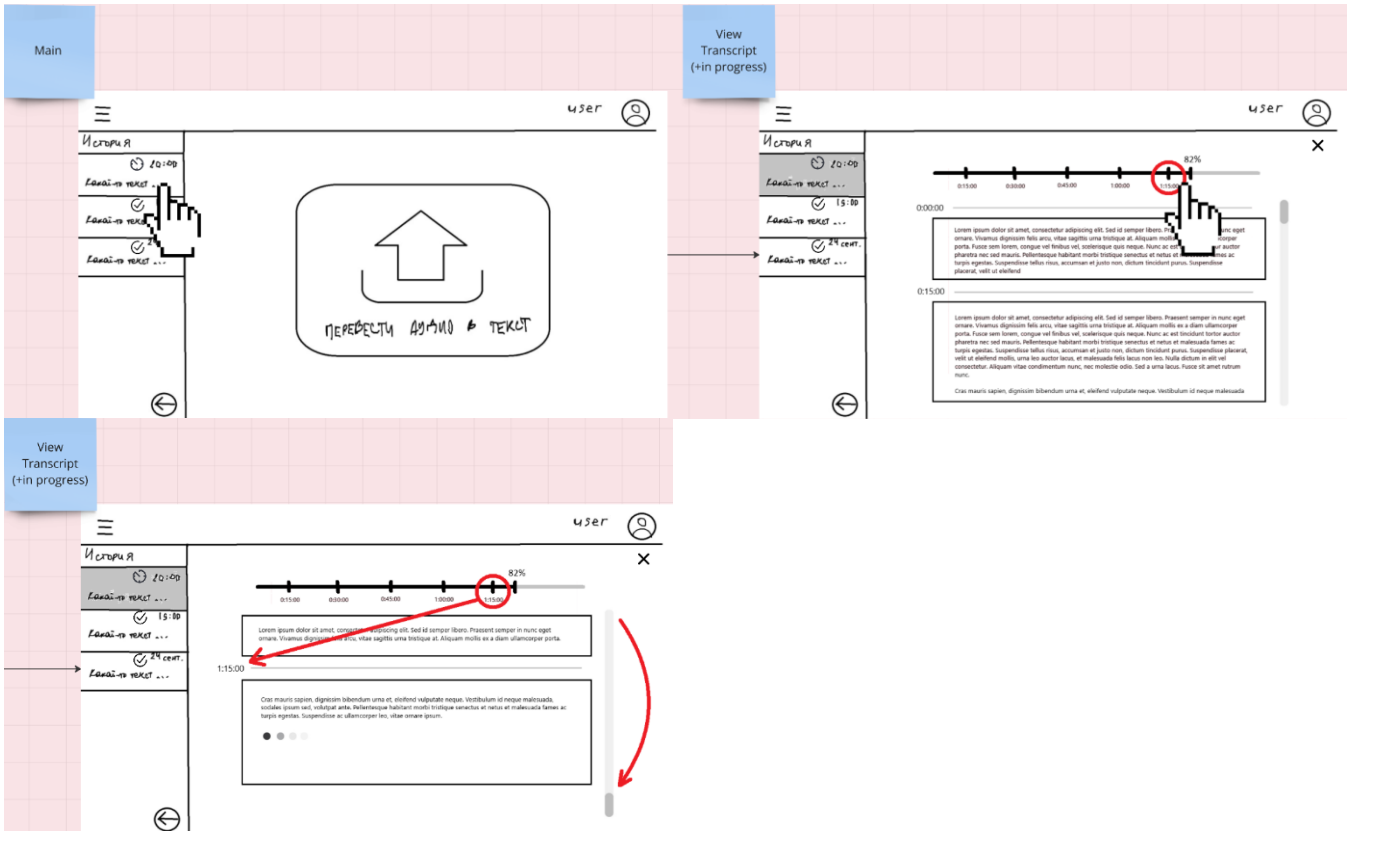
1) Макет дизайна интерфейса

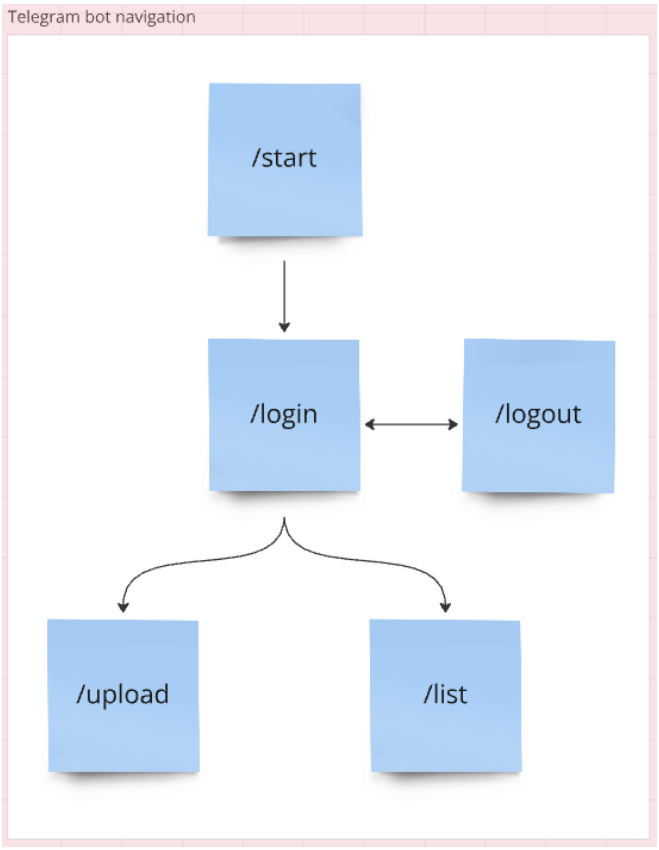


Пользовательская навигация представлена на схеме. Сначала пользователь попадает на окно входа, он может зарегистрироваться, если у него нет аккаунта, или войти под своим логином или паролем. Далее он попадает на главную страницу, откуда может начать процесс транскрипции, либо посмотреть готовые транскрипции.

Далее представлен примерный макет взаимодействий пользователя и системы:







На этой схеме представлена пользовательская навигация в Телеграм-боте.

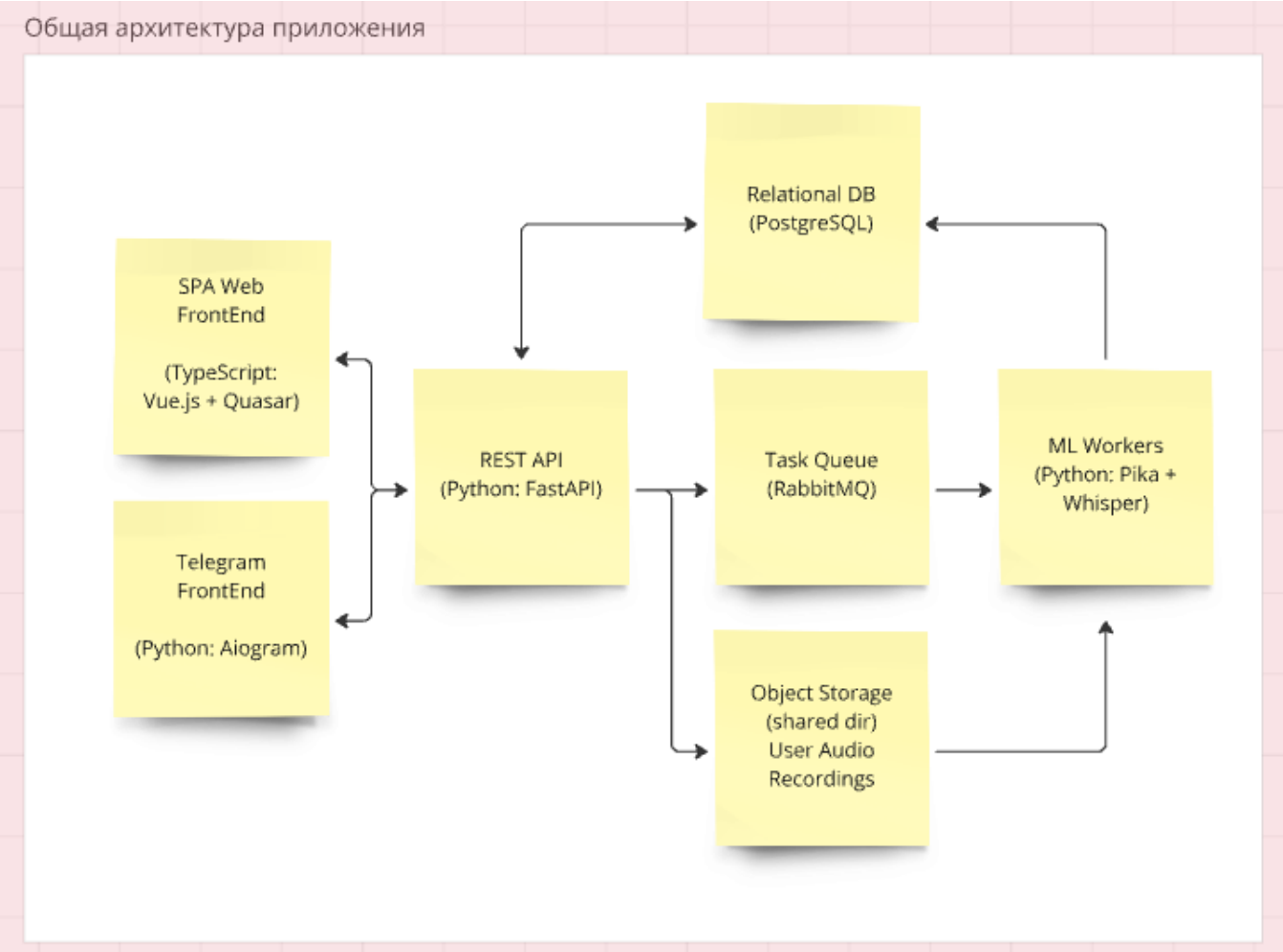
/start – команда запуска бота

/login – команда, с помощью которой пользователь может пойти в свой аккаунт и получить доступ к своим аудиозаписям, как и на сайте

/logout – бот выходит из аккаунта пользователя

После успешного входа можно либо загрузить новый аудиофайл с помощью команды /upload, или просмотреть уже имеющиеся транскрипции с помощью команды /list

2) Общая архитектура приложения + стек технологий



Система реализует транскрипцию аудиофайлов через взаимодействие фронтенда: веб-интерфейс и Telegram-бот, с серверной частью REST API, которая организует управление заданиями на обработку аудио. PI добавляет задачи в очередь обработки и сохраняет аудиофайл в хранилище. Далее задачи обрабатываются машинным обучением (ML Workers) с использованием модели Whisper для транскрипции аудио. Готовые транскрипции сохраняются в реляционной базе данных (PostgreSQL), откуда они могут быть запрошены через API для отображения пользователю.

Описание модулей:

1. SPA Web

- Стек: TypeScript, Vue, Quasar

- Описание: Одностраничное веб-приложение (SPA), предоставляющее пользователю интерфейс для регистрации, авторизации, загрузки аудиофайлов и просмотра истории транскрипций. Управляет данными авторизации и аудиофайлами пользователя, взаимодействует с REST API для передачи запросов на обработку и отображение результатов.

2. Telegram

- Стек: Python, Aiogram

- Описание: Telegram-бот, через который пользователи могут взаимодействовать с системой: загружать аудиофайлы для транскрипции, отслеживать статус обработки и экспортировать готовые транскрипции. Бот также предоставляет функционал для управления сессиями пользователей (логин/логаут) через API.

3. REST API

- Стек: Python, FastAPI

- Описание: Основной интерфейс серверной части, через который все запросы от фронтендов направляются на обработку. Управляет авторизацией пользователей, принимает аудиофайлы, ставит задачи в очередь на обработку, сохраняет файлы в объектное хранилище и предоставляет доступ к истории транскрипций. Работает с базой данных для хранения информации о пользователях и транскрипциях.

4. Relational DB

- Стек: PostgreSQL

- Описание: Реляционная база данных, хранящая учетные записи пользователей, метаданные транскрипций, результаты обработки аудио. Основной источник данных для REST API.

5. Task Queue

- Стек: RabbitMQ

- Описание: Система очередей задач, отвечающая за распределение задач обработки аудиофайлов среди рабочих (ML Workers). Каждый запрос на транскрипцию ставится в очередь и ожидает, пока обработчик заберет его для выполнения.

6. Object Storage

- Реализация: общая папка между ML workers и rest API.

- Описание: Объектное хранилище для аудиофайлов, загружаемых пользователями. Файлы сохраняются до окончания их обработки и получения транскрипций. REST API взаимодействует с этим хранилищем для загрузки и извлечения файлов, а рабочие забирают оттуда файлы для обработки.

7. ML Workers

- Стек: Python, Pika, Whisper

- Описание: Модули обработки аудиофайлов, которые выполняют транскрипцию с помощью модели Whisper. Рабочие получают задачи из очереди (RabbitMQ), загружают аудиофайлы из объектного хранилища, выполняют обработку и сохраняют результаты в реляционную базу данных (PostgreSQL).

3)Диаграмма состояний

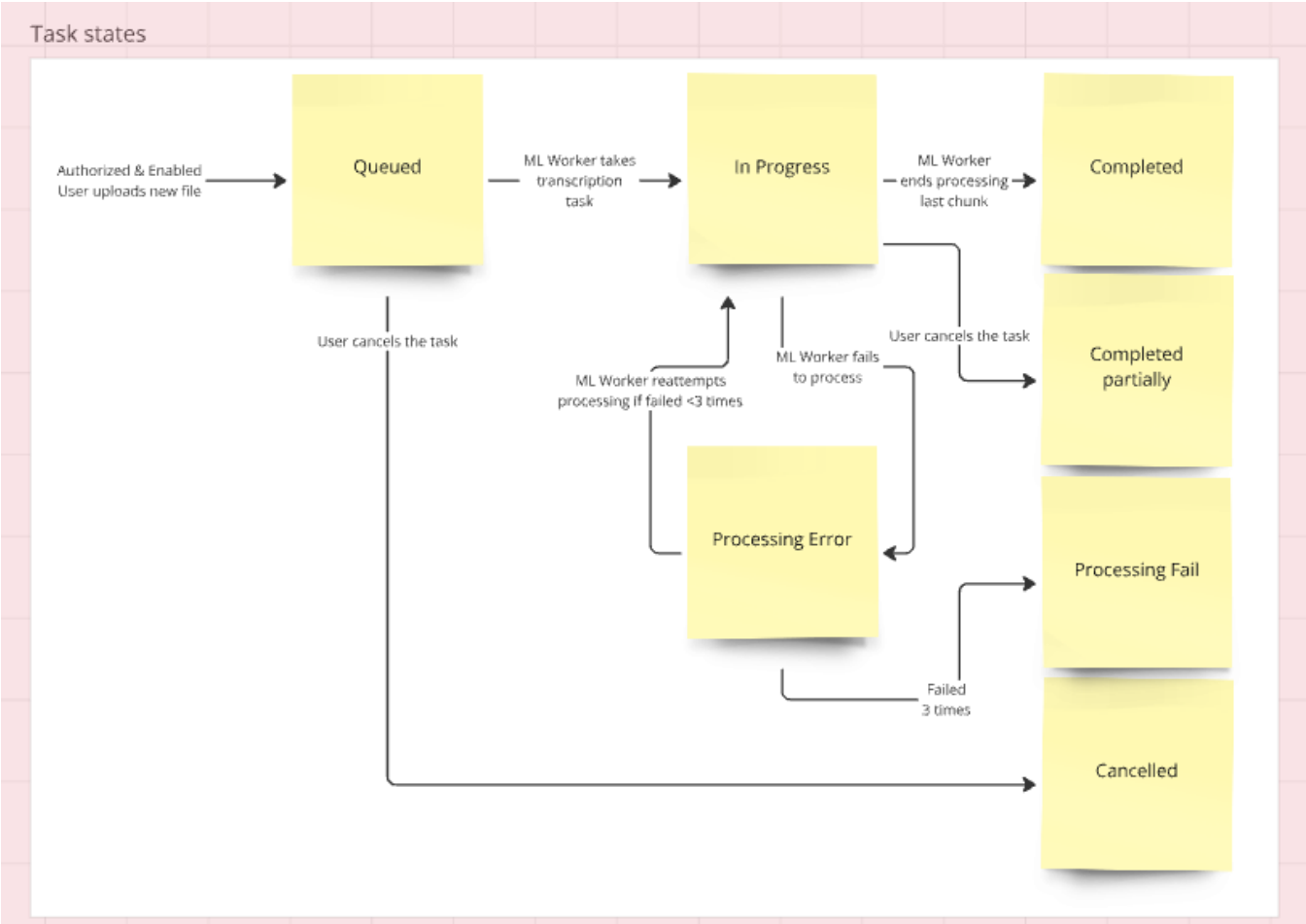


Диаграмма состояний процесса обработки файла для транскрипции аудио. В этой диаграмме определены состояния и события, которые приводят к переходу из одного состояния в другое.

Состояния и связи диаграммы:

1. Authorized & Enabled: User uploads new file

- Описание: Пользователь авторизован и имеет разрешение на загрузку файла для транскрипции. Пользователь загружает новый аудиофайл в систему.

- Переходы:

  - Переход в Queued после успешной загрузки файла.

2. Queued

- Описание: Аудиофайл загружен в систему и находится в очереди задач на обработку.

- Переходы:

  - Переход в In Progress, когда ML Worker берет задачу на транскрипцию.
  - Переход в Cancelled, если user cancels the task (пользователь отменяет задачу до её начала).

3. In Progress

- Описание: Задача обработки аудиофайла в активной фазе, рабочий модуль (ML Worker) выполняет транскрипцию.

- Переходы:

  - Переход в Completed, если рабочий модуль завершает обработку всех частей файла.
  - Переход в Processing Error, если возникает ошибка во время транскрипции.
  - Переход в Completed partially, если пользователь отменяет задачу до завершения ее обработки, пользователю будет возвращена выполненная часть транскрипции.

4. Completed

- Описание: Рабочий завершил обработку всех частей аудиофайла, транскрипция готова.

- Переходы: Нет переходов, это конечное состояние.

5. Processing Error

- Описание: Во время обработки аудиофайла произошла ошибка.

- Переходы:

  - Переход в In Progress, если ошибка произошла меньше трех раз, и система пытается обработать задачу повторно.
  - Переход в Processing Fail, если ошибка произошла три раза.

6. Processing Fail

- Описание: Задача завершилась неудачей после трех попыток обработки аудиофайла.

- Переходы: Нет переходов, это конечное состояние.

7. Cancelled

- Описание: Пользователь отменил задачу на обработку до её завершения, однако транскрипция еще не был начата.

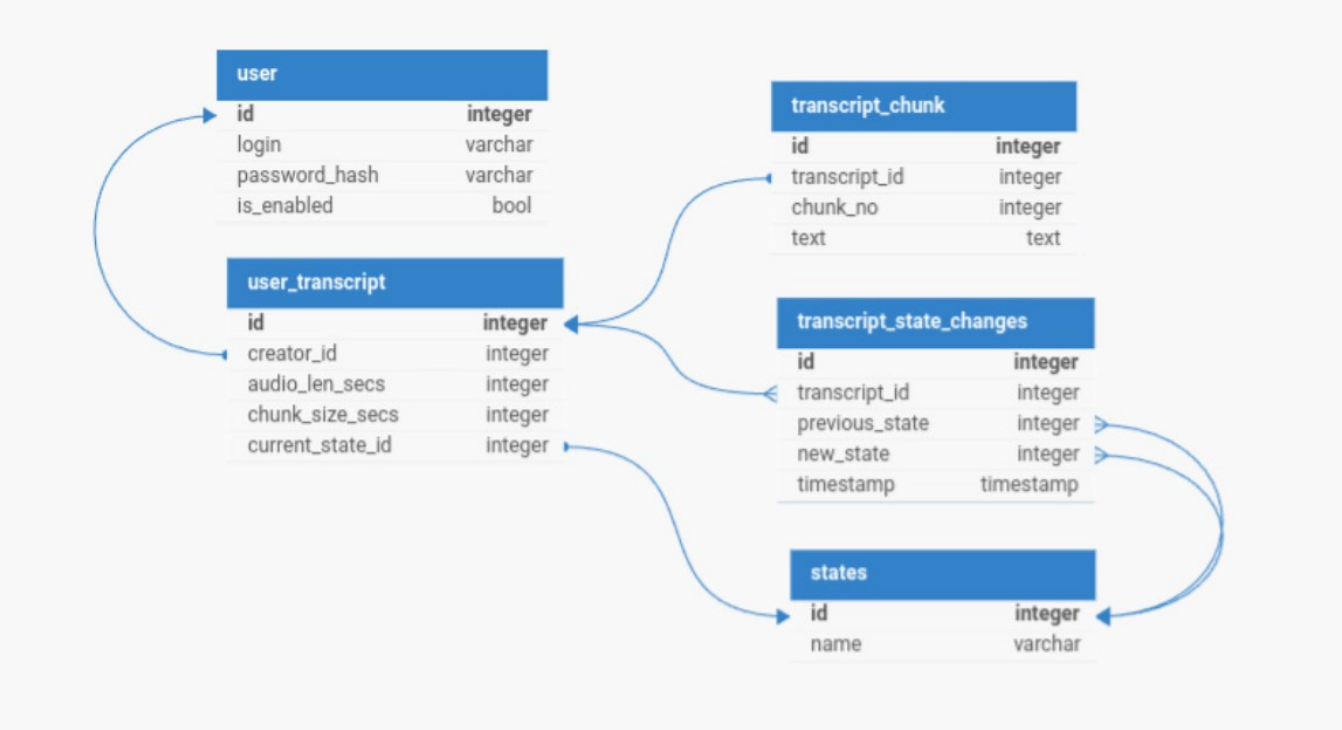
- Переходы: Нет переходов, это конечное состояние.

8. Completed partially

- Описание: Пользователь отменил задачу на обработку до её завершения. Транскрипция не была завершена.

- Переходы: Нет переходов, это конечное состояние.

4) Схемы баз данных



База данных состоит из 5 таблиц: пользователь, пользовательские транскрипты, чанки транскриптов, статус транскрипта, статусы.

Таблицы:

1. **user – пользователь**
    - Хранит информацию о пользователях, имеющих доступ к системе транскрипции.
    - Поля:
        - id (integer): уникальные идентификатор
        - login (varchar): логин пользователя.
        - password_hash (varchar): зашифрованный пароль для аутентификации пользователя.
        - is_enabled (bool): статус, указывающий, активен ли пользователь или отключен.
2. **user_transcript – пользовательские транскрипты**
    - Представляет задачи транскрипции, связанные с каждым пользователем.
    - Поля:
        - id (integer): уникальный идентификатор каждого транскрипта.
        - creator_id (integer): идентификатор пользователя, который создал задачу.
        - audio_len_secs (integer): длительность аудиозаписи в секундах.
        - chunk_size_secs (integer): размер чанка в секундах для аудиозаписи.
        - current_state_id (integer): текущий статус транскрипции, берущийся из таблицы состояний.
3. **transcript_chunk – чанки транскриптов**
    - Разбивает задачу транскрипции на управляемые фрагменты.

- Поля:
    - id (integer): уникальный идентификатор чанков.
    - transcript_id (integer): ссылается на user_transcript, показывая, к какому транскрипту принадлежит каждый чанк.
    - chunk_no (integer): номер чанка в транскрипции.
    - text (text): транскрипция каждого чанка.

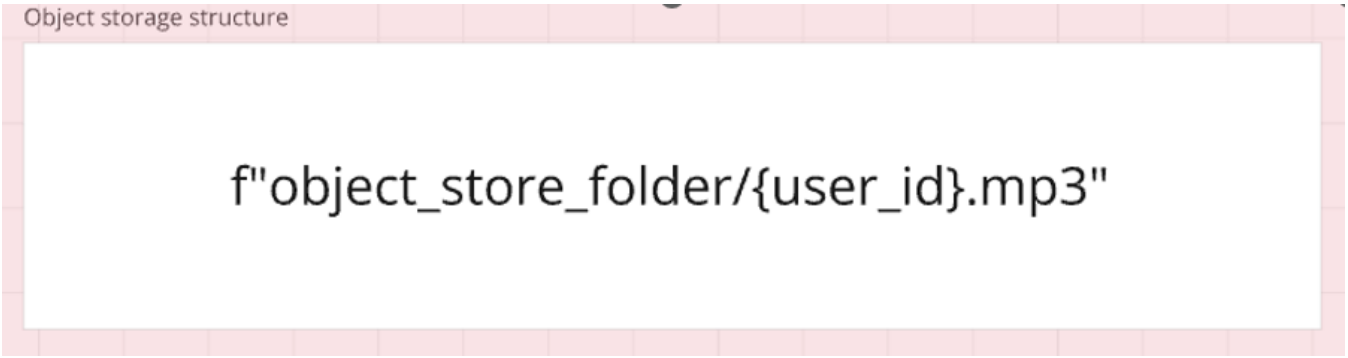
4. **transcript_state_changes – статус транскрипта**
    - Отслеживает историю изменений состояния для каждой задачи транскрипции.
    - Поля:
        - id (integer): уникальный идентификатор для каждой записи об изменении состояния.
        - transcript_id (integer): ссылается на user_transcript, указывает на задачу транскрипции, состояние которой отслеживается.
        - previous_state (integer): предыдущее состояние транскрипта, ссылается на states.
        - new_state (integer): новое состояние после изменения, также ссылается на states.
        - timestamp (timestamp): запись времени изменения состояния.
5. **States – статусы**
    - Определяет различные статусы для задач транскрипции.
    - Поля:
        - id (integer): уникальный идентификатор для каждого состояния.
        - name (varchar): наименование состояния

Связи:

- Таблица user_transcript связывает пользователей с задачами транскрипции, а creator_id указывает на таблицу user.
- transcript_chunk сегментирует каждую задачу транскрипции, связывая ее с user_transcript через transcript_id.
- transcript_state_changes отслеживает переходы состояний для каждой задачи, при этом как previous_state, так и new_state ссылаются на таблицу состояний, которая определяет возможные состояния для задач транскрипции.

5) Структура объектного хранилища

В объектном хранилище используется структура, где каждый аудиофайл сохраняется в формате .mp3 в каталоге object_store_folder. Имя файла определяется уникальным идентификатором пользователя (user_id). Таким образом, каждый аудиофайл будет иметь путь вида:



6) Rest API

1. Авторизация (Login)

Happy Path: Successful Login

```
POST /auth/login HTTP/1.1
Content-Type: application/json

{
  "login": "PovOr",
  "password": "securePassword"
}
```

Response:

```
Content-Type: application/json

{
  "message": "Login successful",
  "token": "your_jwt_token_here"
}
```

Sad Path: Invalid Credentials

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{
  "error": "Invalid login or password"
}
```

2. Регистрация (Registration)

Happy Path: Successful Registration

```
POST /auth/register HTTP/1.1
Content-Type: application/json

{
  "login": "PovOr",
  "password": "securePassword"
}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "message": "Registration successful. Please log in."
}
```

Sad Path: Login Already Exists

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{
  "error": "Login already exists"
}
```

3. Загрузка файла (Upload Audio File)

Happy Path: Successful File Upload

```
POST /upload-audiofile HTTP/1.1
Content-Type: multipart/form-data

--boundary
Content-Disposition: form-data; name="file"; filename="audio.mp3"
Content-Type: audio/mpeg

<binary data>
--boundary--
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "message": "File uploaded successfully",
  "task_id": "12345"
}
```

Sad Path: File Too Large

```
HTTP/1.1 413 Payload Too Large
Content-Type: application/json

{
  "error": "File size exceeds the limit of 200MB"
}
```

4. Получить транскрипции (Get Transcriptions)

Happy Path: Successful Retrieval of Transcriptions
Request:

```
GET /transcriptions?skip=0&limit=10 HTTP/1.1
Authorization: Bearer your_jwt_token_here
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "transcriptions": [
    {
      "task_id": "12345",
      "transcription": "This is the transcribed text of the audio.",
      "timestamp": "2024-10-22T12:34:56Z",

      "transcript_length_secs": 60*120,
      "status": "completed"
```

```
    },
    {
      "task_id": "12346",
      "transcription": "This is another transcribed text of a different
audio.",
      "timestamp": "2024-10-21T11:33:45Z",
      "transcript_length_secs": 60*120,
      "status": "completed"
    }
    // ... up to the limit specified
  ]
}
```

Sad Path: No Transcriptions Found

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "error": "No transcriptions found for the user"
}
```

Sad Path: Invalid Pagination Parameters

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "Invalid pagination parameters. 'skip' must be a non-negative
integer and 'limit' must be a positive integer."
}
```

5. Получить транскрипцию (Get Transcriptions)

Request:

```
GET /transcript?task_id=12345&skip=0&limit=10 HTTP/1.1
Authorization: Bearer your_jwt_token_here
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "task_id": "12345",
  "transcription_chunks": [
    {
      "chunk_order": 1,
      "chunk_size_secs": 60 * 10,
      "transcription": "This is the first part of the transcription."
    },
    {
      "chunk_order": 2,
      "chunk_size_secs": 60 * 10,
      "transcription": "This is the second part of the transcription."
    }
    // ... up to the limit specified
  ]
}
```

Sad path: no transcription found

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "error": "No transcription found for the given task ID"
```

```
}
```

6. Экспорт транскрипции в .doc или .txt формат

Описание

Этот эндпоинт позволяет экспортировать транскрипцию по заданному task_id в формате .doc или .txt.

Happy Path: Экспорт в .doc

```
GET /transcript/export?task_id=12345&format=doc HTTP/1.1
Authorization: Bearer your_jwt_token_here
```

Response:

```
HTTP/1.1 200 OK
Content-Disposition: attachment; filename="transcription_12345.doc"
Content-Type: application/vnd.openxmlformats-
officedocument.wordprocessingml.document

<binary data>
```

Happy Path: Экспорт в .txt

```
GET /transcript/export?task_id=12345&format=txt HTTP/1.1
Authorization: Bearer your_jwt_token_here
```

Response:

```
HTTP/1.1 200 OK
Content-Disposition: attachment; filename="transcription_12345.txt"
Content-Type: text/plain; charset=utf-8

This is the transcribed text of the audio...
```

Sad Path: Некорректный формат

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "Invalid format. Supported formats are 'doc' and 'txt'."
}
```

Sad Path: Транскрипция не найдена

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "error": "No transcription found for the given task ID."
}
```

Параметры запроса

- task_id (string, обязательный) — ID задачи транскрипции.
- format (string, обязательный) — Формат экспорта (doc или txt).