Простой сервис создания аудиокниг

**3. Эскизный проект**

**I. Постановка задачи.**

Разработать веб-сервис A-Book, позволяющий пользователям загружать текстовые файлы в формате epub или fb2 и получать в ответ аудиофайл в формате mp3, сгенерированный на основе технологии Text-to-Speech (TTS). Сервис должен быть удобным, безопасным, масштабируемым, учитывать ограничения TTS сервисов на длину текста, поддерживать регистрацию пользователей, администрирование, резервное копирование данных, объединение mp3 файлов и иметь систему логирования и ошибок.

**II. Компоненты системы.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Компонент | Описание | Технологии | Оценка трудозатрат (человеко-часы) | Связи | Объяснение выбора |
| **Клиентская часть (Frontend)** | Веб-интерфейс для взаимодействия с пользователем. Включает в себя форму для загрузки файлов, индикатор прогресса, кнопку скачивания полученного аудиофайла, страницы регистрации/авторизации, личный кабинет. | HTML, CSS, JavaScript, Bootstrap/Tailwind CSS | 60-80 | Отправляет файлы и пользовательские данные на сервер (Backend). Получает уведомления о прогрессе, результат (аудиофайл) и сообщения об ошибках от сервера. | React/Vue/Angular: Современные фреймворки обеспечивают модульность, повторное использование компонентов и удобство разработки сложных UI. HTML/CSS: Базовые технологии для создания веб-интерфейса. |
| **Серверная часть (Backend)** | Обрабатывает запросы от клиента, управляет процессом конвертации текста в аудио, взаимодействует с TTS сервисами, сохраняет и возвращает готовые аудиофайлы, аутентифицирует пользователей, управляет администрированием. | Python (Flask/Django) | 120-160 | Получает файлы и пользовательские данные от Frontend. Взаимодействует с TTS сервисами. Возвращает аудиофайлы и сообщения на Frontend. Взаимодействует с базой данных. | Python: Широкий выбор библиотек для обработки текста, работы с API, машинного обучения. Flask/Django: Фреймворки облегчают создание веб-приложений и API, Django предоставляет больше готовых решений "из коробки". |
| **TTS Сервис (Text-to-Speech)** | Сервис, преобразующий текст в аудио. | Google Cloud Text-to-Speech | **-** | Получает текст от Backend. Возвращает аудиоданные Backend. | Google Cloud TTS: Высокое качество голоса, удобный API. |
| **Парсер форматов epub/fb2** | Модуль, извлекающий текст из файлов epub и fb2. | Python (ebooklib, fb2lib). | 20-40 | Получает файлы от Backend. Возвращает текст в Backend. | Python (ebooklib/fb2lib): Удобные библиотеки для парсинга epub и fb2 форматов, легко интегрируются с Python Backend. |
| **Хранилище файлов** | Для хранения временно сгенерированных аудиофайлов и метаданных | Google Cloud Storage | **-** | Backend сохраняет аудиофайлы. Backend извлекает аудиофайлы для отдачи пользователю. | Облачные хранилища: Масштабируемость, надежность, доступность, интеграция с другими сервисами облачных платформ. |
| **База данных** | Для хранения информации о книгах, пользователях, и статистике использования сервиса. | MySQL | **-** | Backend сохраняет и извлекает информацию о книгах и пользователях. | MySQL: Реляционные базы данных, обеспечивают надежность, транзакционность, соответствие ACID. |
| **Регистрация пользователей** | Модуль, обеспечивающий создание, аутентификацию и управление учетными записями пользователей. | Интегрируется в Backend. Использование библиотек для аутентификации (Passport.js для Node.js, Flask-Login для Python Flask, Spring Security для Java Spring). Реализация REST API для управления пользователями. | Включено в оценку Backend (в пределах 120-160 часов). | Frontend отправляет запросы на регистрацию/аутентификацию. Backend взаимодействует с базой данных для хранения информации о пользователях. | Passport.js/Flask-Login/Spring Security: Библиотеки для реализации аутентификации, упрощают интеграцию с Backend фреймворками, обеспечивают безопасность |
| **Инструменты администрирования** | Веб-интерфейс для администраторов для управления пользователями, контентом, мониторинга системы и т.д. | Отдельная панель администрирования, построенная с использованием того же стека, что и Frontend и Backend. | 40-60 | Администраторы взаимодействуют через веб-интерфейс. Backend предоставляет API для управления системой. Backend взаимодействует с базой данных и хранилищем файлов. | Отдельная панель: Разделение логики администрирования и пользовательской части, удобство управления системой |
| **Резервное копирование пользовательских данных** | Система автоматического резервного копирования базы данных и хранилища файлов для восстановления в случае сбоя. | Использование инструментов резервного копирования, предоставляемых облачными платформами (AWS Backup, Google Cloud Backup, Azure Backup) или самостоятельная реализация с использованием cron jobs и скриптов. | 20-40 | Backend или отдельный скрипт запускает процесс резервного копирования. Данные сохраняются в безопасном месте (например, отдельный облачный storage). | Облачные инструменты: Удобство использования, автоматизация, надежность. Cron jobs + скрипты: Более гибкий, но требует больше ручной настройки и обслуживания. |
| **Модуль объединения mp3** | Модуль для объединения нескольких mp3 файлов в один. Необходим, так как текст книги разбивается на части для обработки TTS сервисом.  Технологии: Библиотеки для работы с аудиофайлами (например, pydub в Python, ffmpeg). | Библиотеки для работы с аудиофайлами (например, pydub в Python, ffmpeg). | 20-30 | Backend получает отдельные mp3 файлы от TTS сервиса. Backend объединяет файлы с помощью модуля. Backend сохраняет объединенный файл. | pydub/ffmpeg: Библиотеки для работы с аудиофайлами, поддерживают различные форматы, позволяют выполнять объединение, конвертацию и другие операции |
| **Система логирования и ошибок** | Система логирования для записи событий, ошибок и предупреждений в системе. Система мониторинга ошибок для оперативного выявления и устранения проблем. | Logging framewPython's logging module, Log4j for Java). Инструменты мониторинга ошибок (Sentry, Rollbar). | 20-30 | Все компоненты системы отправляют логи и сообщения об ошибках в систему логирования и мониторинга. | Logging modules: Встроены в языки программирования, позволяют записывать события и ошибки. Sentry/Rollbar: Инструменты для мониторинга ошибок, позволяют отслеживать и анализировать ошибки в реальном времени. |
| **Интеграция и тестирование** | Проверка работоспособности всех компонентов вместе, написание юнит-тестов и интеграционных тестов. | Unit-testing frameworks (unittest, pytest, JUnit), integration testing tools. | 40-60 | Тестирование проводится между всеми компонентами системы. | Unit-testing frameworks: Позволяют тестировать отдельные модули кода, обеспечивают надежность. Integration testing tools: Позволяют тестировать взаимодействие между различными компонентами системы |
| ИТОГО | - | - | 420-580+ | - | - |

**III. Набросок реализации.**

**Этап 1**: Базовая функциональность (4-6 недель)

• Разработка Backend (Python/Flask): Прием файла, парсинг, TTS (Google Cloud), склейка mp3, отдача результата.

• Разработка Frontend (HTML/CSS/JS): Загрузка файла, индикатор, скачивание.

• Базовое логирование.

**Этап 2**: Пользователи и безопасность (3-4 недели)

• Реализация регистрации/логина.

• Базовая валидация и защита от атак.

**Этап 3**: Администрирование и стабильность (3-4 недели)

• Разработка панели администрирования (управление пользователями).

• Настройка резервного копирования.

• Тестирование и исправление ошибок.

**Этап 4**: Деплой и запуск (1-2 недели)

• Развертывание на Selectel VDS.

• Запуск и мониторинг.

**Итого:** 11-16 недель (приблизительно)