Концептуальное описание проекта: Веб-приложение «SoundZen»

1. Выбор предметной области

В качестве проекта выбран сайт веб-приложение «SoundZen» для прослушивания фоновой lo-fi музыки и успокаивающих и приятных звуков природы. Цель сайта - помочь пользователям сконцентрироваться, расслабиться или медитировать, предоставив им удобный инструмент для смешивания различных аудиодорожек (например, мелодий, дождя, костра, шума волн) с индивидуальной настройкой громкости для каждого звука.

2. Выбор архитектуры проекта

Для проекта выбрана микросервисная архитектура. Данный выбор обоснован следующими факторами:

* Гибкость и масштабируемость: Разные типы звуков могут быть развернуты в виде независимых сервисов. Это позволяет масштабировать их отдельно друг от друга, например, если нагрузка на сервис с музыкой будет выше.
* Независимость технологий: Микросервисная архитектура позволяет в будущем легко заменить или добавить новые технологии для отдельных сервисов без переписывания всей системы.
* Отказоустойчивость: В случае сбоя в одном микросервисе, основная функциональность приложения продолжит работать.

3. Основные части проекта

Проект состоит из двух основных частей: клиентской (фронтенд) и серверной (бэкенд), которая, в свою очередь, делится на несколько микросервисов.

Фронтенд (Client Layer):

Технология: React.js

Основные компоненты и страницы:

* Главная страница: представляет собой основной аудиоплеер с интерфейсом для управления звуками.
* Страница аутентификации: формы для регистрации нового пользователя и входа в систему.
* Страница библиотеки lo-fi музыки: каталог доступных музыкальных треков.
* Страница библиотеки звуков природы: каталог доступных звуков (дождь, костер, гроза, журчание реки и т.д.).

Бэкенд (Backend Layer) — микросервисы:

* API Gateway (Единая точка входа): сервис на Python FastAPI, который принимает все запросы от фронтенда и перенаправляет их соответствующим микросервисам. Также может заниматься аутентификацией и кешированием.
* Auth Service (Сервис аутентификации): отвечает за регистрацию, вход, проверку JWT-токенов и управление учетными записями пользователей.
* Tracks Service (Сервис управления аудио): предоставляет API для получения списка lo-fi треков и звуков природы, их метаданных (название, автор, длительность).
* Playback Service (Сервис воспроизведения): отвечает за логику управления аудиопотоками (запуск, пауза, смешивание громкости), хотя основное воспроизведение может происходить на клиенте.
* База данных: PostgreSQL используется для хранения данных пользователей, метаданных аудиотреков, плейлистов и настроек.

4. Примерный Pipeline проекта

Процесс разработки и поставки кода (CI/CD) будет организован следующим образом:

1. Хостинг кода: Исходный код всех микросервисов и фронтенда размещается на GitHub в одном репозитории.

2. Ветвление: для внесения изменений разработчик создает новую feature-ветку от ветки «develop»

3. Коммиты и Push: разработчик делает коммиты в свою ветку и запушивает ее на GitHub.

4. Pull Request (PR): создается Pull Request для слияния изменений в ветку «develop».

5. Запуск Pipeline (CI): при создании PR автоматически запускается пайплайн на GitHub Actions, который:

\* Собирает проект (устанавливает зависимости).

\* Запускает модульные и интеграционные тесты.

6. Code Review: один или несколько других разработчиков проводят ревью кода в PR.

7. Слияние в Develop: после успешного прохождения тестов и одобрения ревьюера ветка сливается в «develop».

8. Деплой на Staging: ветка «develop» автоматически деплоится на тестовый сервер (staging) для проверки QA.

9. Релиз и деплой на Production: после тестирования создается PR из «develop» в «main». После его мержа создается релизная версия (tag), и пайплайн (CD) с помощью Docker и Docker Compose развертывает обновленные контейнеры на виртуальной машине (production-сервере).

5. Используемые технологии

* Фронтенд: React.js, HTML5, CSS3, JavaScript (ES6+)
* Бэкенд: Python, FastAPI (для всех микросервисов и API Gateway)
* База данных: PostgreSQL
* Аутентификация: JWT (JSON Web Tokens)
* Контейнеризация: Docker, Docker Compose
* Хостинг кода и CI/CD: GitHub, GitHub Actions
* Деплой: Виртуальная машина
* Инструменты дизайна: Figma
* Прочее: Nginx