**Описание структуры программного кода (На курсовую сделать скриншоты: Дерево коммитов.**

**1. Общая структура проекта**

Проект разделён на три основные части:

* **Database**: Модуль для работы с базой данных.
  + DatabaseHandler.cs: Класс для подключения и выполнения запросов к базе данных.
  + PlaylistManager.cs: Класс для управления плейлистами (добавление, удаление, сортировка песен).
  + AdvertisementManager.cs: Класс для управления рекламными блоками (добавление, планирование).
* **API**: Модуль для обработки запросов.
  + MusicAPI.cs: Класс для поиска и управления музыкой (поиск песен, исполнителей, плейлистов).
  + AdAPI.cs: Класс для работы с рекламными блоками (запросы на добавление и удаление рекламы).
* **Services**: Модуль для реализации бизнес-логики.
  + PlaybackService.cs: Класс для автоматизации воспроизведения контента по расписанию.
  + SchedulingService.cs: Класс для планирования очереди песен и рекламных блоков.
  + AnalyticsService.cs: Класс для сбора статистики и аналитики (например, метрики прослушиваний).

**2. Пример структуры проекта в Visual Studio**

/RadioApp

├── /Database

│ ├── DatabaseHandler.cs

│ ├── PlaylistManager.cs

│ └── AdvertisementManager.cs

├── /API

│ ├── MusicAPI.cs

│ └── AdAPI.cs

├── /Services

│ ├── PlaybackService.cs

│ ├── SchedulingService.cs

│ └── AnalyticsService.cs

├── RadioApp.csproj

└── Program.cs

**3. Описание классов**

**Database**

* **DatabaseHandler.cs**:
  + Подключение к базе данных.
  + Выполнение SQL-запросов для работы с песнями, плейлистами и рекламой.
* **PlaylistManager.cs**:
  + Методы для добавления, удаления и перемещения песен внутри очереди.
  + Хранение информации о текущем состоянии плейлистов.
* **AdvertisementManager.cs**:
  + Управление рекламными блоками: добавление, удаление и планирование.
  + Хранение данных о рекламных кампаниях.

**API**

* **MusicAPI.cs**:
  + Методы для поиска песен, исполнителей и плейлистов.
  + Интерфейс для добавления песен в очередь.
* **AdAPI.cs**:
  + Методы для интеграции рекламы в плейлист.
  + Интерфейс для управления рекламодателями.

**Services**

* **PlaybackService.cs**:
  + Логика воспроизведения контента по расписанию.
  + Управление текущим воспроизведением (старт/пауза, пропуск).
* **SchedulingService.cs**:
  + Планирование очереди воспроизведения.
  + Поддержка автоматической расстановки рекламы между песнями.
* **AnalyticsService.cs**:
  + Сбор данных о времени прослушивания.
  + Формирование отчётов о прослушивании и эффективности рекламы.

**4. Репозиторий проекта**

**Основные ветки:**

* main: Основная рабочая версия проекта.
* feature/ui: Разработка пользовательского интерфейса.
* feature/analytics: Реализация аналитической части.
* feature/ad\_management: Управление рекламой.

**Структура файлов:**

* .cs: Файлы кода на C#.
* .json: Конфигурационные файлы для API и базы данных.
* README.md: Документация по проекту.

**Пример файла README.md:**

RadioApp

Описание

Приложение для управления радиостанцией, включая воспроизведение музыки, автоматизацию расписания и интеграцию рекламы.

Функциональность

- Планирование эфира.

- Автоматизация воспроизведения.

- Управление рекламными кампаниями.

- Мониторинг и аналитика.

Структура проекта

- \*\*Database\*\*: Работа с базой данных.

- \*\*API\*\*: Обработка запросов к системе.

- \*\*Services\*\*: Реализация бизнес-логики.

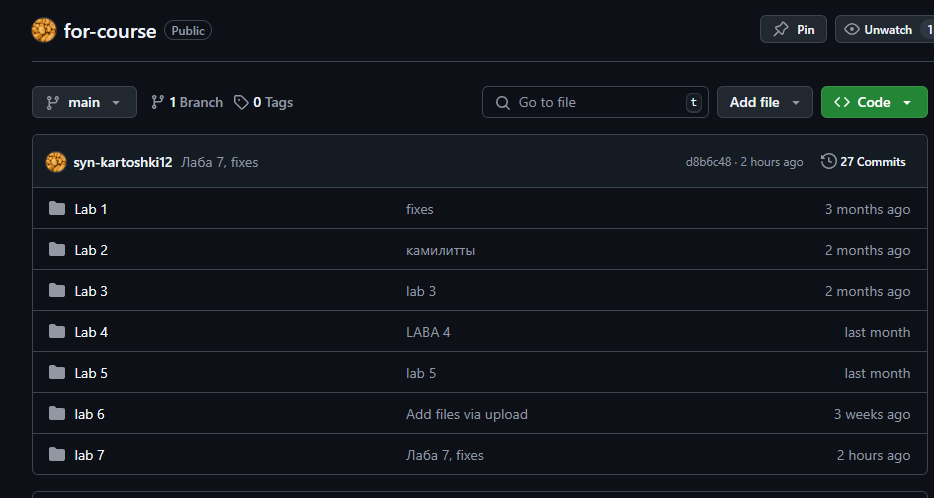
Установка

1. Склонируйте репозиторий: `git clone <repository\_url>`

2. Откройте проект в Visual Studio.

3. Настройте базу данных, используя файл `appsettings.json`.

4. Запустите приложение.

  
  
Расписать, что находится в каждой папке отдельно!