Задание

1. ***Спроектируйте API по любой предметной области***

Желательно, чтобы у всех тема отличалась - можете "занимать" темы в ветке "Аналитика" чата по практике)

Опишите:
· ключевые ресурсы (entities),
· основные endpoint-ы и HTTP-методы,
· схемы запросов и ответов,
· обязательные поля и статус-коды ошибок,
· правила безопасности (например, токенизация, ограничение запросов).

2. ***Продумайте стратегию сопровождения API***

Кратко опишите концепт, по желанию добавьте Roadmap развития API

Решение:

1. ***Предметная область. Обзор концепции.***

   SoundWave – современный музыкальный сервис с уникальными возможностями.

   Ключевые особенности:

   - AI Music Discovery – умные рекомендации на основе эмоционального состояния и контекста
   - Collaborative Playlists – совместное создание плейлистов в реальном времени
   - Mood Rooms – виртуальные комнаты для прослушивания музыки с друзьями
   - Artist Direct Connect – прямая связь артистов с фанатами
   - Sonic DNA – уникальный музыкальный профиль каждого пользователя

   Базовый URL: https://api.soundwave.io/v1

2. ***Ключевые ресурсы (entities)***

   1. User (Пользователь)

   Представляет зарегистрированного пользователя платформы с его музыкальным профилем и настройками.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор пользователя
username	String	Да	Уникальное имя пользователя
email	String	Да	Уникальный email адрес для входа и уведомлений
displayName	String	Да	Отображаемое имя пользователя (1-100 символов)

avatarUrl	String (URL)	Нет	Ссылка на изображение аватара пользователя
sonicDNA	Object	Нет	Музыкальный профиль пользователя на основе истории прослушивания
sonicDNA.topGenres	Array[String]	Нет	Топ-жанры пользователя (например: [“rock”, “indie”, “electronic”])
sonicDNA.moodProfile	Object	Нет	Профиль настроений (предпочтения по энергичности, валентности музыки)
sonicDNA.listeningHabits	Object	Нет	Статистика прослушивания (любимое время суток, средняя длительность сессии)
subscription	Object	Да	Информация о подписке пользователя
subscription.tier	Enum	Да	Тип подписки: free, premium, family
subscription.expiresAt	DateTime	Нет	Дата окончания платной подписки (null для free tier)
createdAt	DateTime	Да	Дата и время регистрации пользователя
updatedAt	DateTime	Да	Дата и время последнего обновления профиля

2. Track (Трек)

Представляет музыкальную композицию с метаданными и информацией для воспроизведения.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор трека
title	String	Да	Название трека (1-500 символов)
artistId	UUID	Да	Идентификатор основного артиста
albumId	UUID	Нет	Идентификатор альбома (null для синглов вне альбома)
duration	Integer	Да	Длительность трека в секундах
audioUrl	String (URL)	Да	URL для стриминга аудио (защищенный, временный)

coverArtUrl	String (URL)	Да	URL обложки трека
genres	Array[String]	Да	Жанры трека (например: [“rock”, “alternative”])
moodTags	Array[String]	Нет	Теги настроения (например: [“energetic”, “uplifting”, “melancholic”])
explicit	Boolean	Да	Содержит ли трек ненормативную лексику (default: false)
releaseDate	Date	Да	Дата релиза трека
playCount	Integer	Да	Общее количество прослушиваний
likeCount	Integer	Да	Количество лайков от пользователей
metadata	Object	Нет	Дополнительные метаданные трека
metadata.isrc	String	Нет	Международный стандартный код записи (ISRC)
metadata.copyright	String	Нет	Информация об авторских правах
createdAt	DateTime	Да	Дата добавления трека в систему

3. Playlist (Плейлист)

Представляет коллекцию треков, созданную пользователем или системой.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор плейлиста
name	String	Да	Название плейлиста (1-200 символов)
description	String	Нет	Описание плейлиста (до 1000 символов)
ownerId	UUID	Да	Идентификатор создателя плейлиста
coverImageUrl	String (URL)	Нет	URL обложки плейлиста (генерируется автоматически из треков, если не указано)
isPublic	Boolean	Да	Доступен ли плейлист для всех (default: false)
isCollaborative	Boolean	Да	Могут ли другие пользователи редактировать плейлист (default: false)
collaborators	Array[UUID]	Нет	Массив ID пользователей с правами редактирования
tracks	Array[UUID]	Да	Упорядоченный массив ID треков в плейлисте

trackCount	Integer	Да	Общее количество треков (вычисляемое поле)
totalDuration	Integer	Да	Суммарная длительность всех треков в секундах
followers	Integer	Да	Количество пользователей, подписанных на плейлист
tags	Array[String]	Нет	Теги для категоризации (например: [“workout”, “chill”, “party”])
createdAt	DateTime	Да	Дата создания плейлиста
updatedAt	DateTime	Да	Дата последнего изменения (добавление/удаление треков, смена названия)

4. Artist (Артист)

Представляет музыкального исполнителя или группу с их контентом и статистикой.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор артиста
name	String	Да	Имя артиста или название группы (1-200 символов)
bio	String	Нет	Биография артиста (до 5000 символов)
genres	Array[String]	Да	Основные жанры творчества артиста
avatarUrl	String (URL)	Нет	URL фото профиля артиста
coverImageUrl	String (URL)	Нет	URL баннера/обложки для страницы артиста
verified	Boolean	Да	Подтверждена ли личность артиста (синяя галочка, default: false)
monthlyListeners	Integer	Да	Количество уникальных слушателей за последние 28 дней
socialLinks	Object	Нет	Ссылки на социальные сети и сайт
socialLinks.vk	String	Нет	VK handle или URL
socialLinks.tg	String	Нет	Telegram handle или URL
socialLinks.website	String (URL)	Нет	Официальный сайт артиста
createdAt	DateTime	Да	Дата добавления артиста в систему

5. MoodRoom (Комната настроения)

Представляет виртуальную комнату для совместного прослушивания музыки в реальном времени.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор комнаты
name	String	Да	Название комнаты (1-100 символов)
hostId	UUID	Да	ID пользователя-хоста (создателя комнаты, управляет плейбэком)
currentTrackId	UUID	Нет	ID трека, который воспроизводится сейчас (null если остановлено)
playbackPosition	Integer	Да	Текущая позиция воспроизведения в секундах
participants	Array[UUID]	Да	Массив ID пользователей, находящихся в комнате в данный момент
maxParticipants	Integer	Да	Максимальное количество участников (5-500, зависит от подписки хоста)
mood	Enum	Да	Настроение комнаты: chill, party, focus, workout
isPublic	Boolean	Да	Доступна ли комната для всех или только по приглашению (default: false)
chatEnabled	Boolean	Да	Включен ли текстовый чат в комнате (default: true)
queue	Array[UUID]	Да	Очередь треков для воспроизведения
createdAt	DateTime	Да	Дата и время создания комнаты

Примечание: Комнаты автоматически удаляются через 30 минут после выхода последнего участника.

6. Album (Альбом)

Представляет музыкальный альбом, сборник или EP.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор альбома
title	String	Да	Название альбома (1-500 символов)
artistId	UUID	Да	ID основного артиста альбома
releaseDate	Date	Да	Дата официального релиза

coverArtUrl	String (URL)	Да	URL обложки альбома (высокое разрешение)
genres	Array[String]	Да	Жанры альбома
trackIds	Array[UUID]	Да	Упорядоченный массив ID треков в альбоме
totalDuration	Integer	Да	Суммарная длительность всех треков в секундах
label	String	Нет	Название лейбла/студии, выпустившей альбом
type	Enum	Да	Тип релиза: album (7+ треков), single (1-3 трека), ep (4-6 треков)
createdAt	DateTime	Да	Дата добавления альбома в систему

7. Recommendation (Рекомендация)

Представляет персонализированную рекомендацию трека для пользователя, сгенерированную AI.

Поле	Тип	Обязательное	Описание
id	UUID	Да	Уникальный идентификатор рекомендации
userId	UUID	Да	ID пользователя, для которого создана рекомендация
trackId	UUID	Да	ID рекомендуемого трека
score	Float	Да	Оценка релевантности от 0.0 до 1.0 (чем выше, тем лучше совпадение)
reason	String	Да	Объяснение рекомендации (например: “Похоже на треки, которые вы недавно слушали”)
context	Enum	Да	Контекст рекомендации: mood (по настроению), similar_taste (похожие вкусы), trending (популярное), new_release (новинки)
generatedAt	DateTime	Да	Дата и время создания рекомендации
interacted	Boolean	Да	Взаимодействовал ли пользователь с рекомендацией (прослушал, лайкнул, скрыл)

Примечание: Рекомендации автоматически обновляются каждые 6 часов на основе последней активности пользователя.

3. Основные endpoint-ы и HTTP-методы

Ресурс / Эндпоинт	**GET (список)**	**GET (один)**	**POST (создать)**	**PATCH / PUT (обновить)**	**DELETE (удалить)**

/auth/register	–	–	Да	–	–
/auth/login	–	–	Да	–	–
/users/me	–	Да	–	Да (PATCH)	–
/users/{userId}	–	Да	–	–	–
/tracks	Да	–	–	–	–
/tracks/{trackId}	–	Да	–	–	–
/playlists	Да	–	Да	Да	Да
/playlists/{playlistId}	–	Да	–	Да	Да
/artists	Да	–	–	–	–
/artists/{artistId}	–	Да	–	–	–
/rooms	Да	–	Да	–	–
/rooms/{roomId}	–	Да	–	–	–
/recommendations/for-you	Да	–	–	–	–
/recommendations/mood-based	Да	–	–	–	–
/search	Да	–	–	–	–
/me/library/tracks	Да	–	–	–	–
/me/library/playlists	Да	–	–	–	–
/me/library/artists	Да	–	–	–	–
/me/listening-history	Да	–	–	–	–

4. Схемы запросов и ответов

- *Авторизация и пользователи:*
  - POST /auth/register – Регистрация нового пользователя

```
Request:
{
  "email": "user@example.com",
  "password": "SecurePass123!",
  "username": "music_lover",
  "displayName": "Music Lover"
}

Response: 201 Created
{
  "user": { User object },
  "tokens": {
    "accessToken": "jwt_token",
    "refreshToken": "refresh_token",
    "expiresIn": 3600
```

```
  }
}
```

- POST /auth/login – Авторизация пользователя

```
Request:
{
  "email": "user@example.com",
  "password": "SecurePass123!"
}

Response: 200 OK
{
  "user": { User object },
  "tokens": { tokens object }
}
```

- GET /users/me – Получение профиля текущего пользователя

```
Headers: Authorization: Bearer {token}
Response: 200 OK
{ User object }
```

- PATCH /users/me – Обновление профиля пользователя

```
Request:
{
  "displayName": "New Name",
  "avatarUrl": "https://cdn.example.com/avatar.jpg"
}

Response: 200 OK
{ Updated User object }
```

- GET /users/{userId} – Получение публичного профиля пользователя

```
Response: 200 OK
{ User object (public fields only) }
```

- *Треки:*
  - GET /tracks – Поиск и фильтрация треков

```
Query params:
  ?q=search_query
  &genre=rock
  &mood=energetic
  &limit=20
```

```
  &offset=0
  &sortBy=popularity|release_date|title

Response: 200 OK
{
  "tracks": [{ Track objects }],
  "total": 1250,
  "limit": 20,
  "offset": 0
}
```

- GET /tracks/{trackId} – Получение информации о треке

```
Response: 200 OK
{ Track object with extended metadata }
```

- POST /tracks/{trackId}/play – Регистрация воспроизведения трека

```
Request:
{
  "playbackContext": "playlist|album|radio|search",
  "contextId": "uuid (optional)",
  "position": 0
}

Response: 200 OK
{
  "playbackId": "uuid",
  "audioUrl": "https://cdn.soundwave.io/stream/...",
  "expiresAt": "datetime"
}
```

- POST /tracks/{trackId}/like – Лайк трека

```
Response: 200 OK
{
  "liked": true,
  "likeCount": 12543
}
```

- DELETE /tracks/{trackId}/like – Удаление лайка

```
Response: 200 OK
{
  "liked": false,
  "likeCount": 12542
}
```

- GET /tracks/{trackId}/similar – Получение похожих треков

  Query params: ?limit=10

  Response: 200 OK
  ```
  {
   "tracks": [{ Track objects }],
   "similarityScores": {
    "track_id_1": 0.95,
    "track_id_2": 0.89
   }
  }
  ```

- *Плейлисты:*
  - GET /playlists – Получение списка плейлистов

    Query params:
    ```
     ?userId=uuid (optional)
     &isPublic=true
     &limit=20
     &offset=0
    ```

    Response: 200 OK
    ```
    {
     "playlists": [{ Playlist objects }],
     "total": 45,
     "limit": 20,
     "offset": 0
    }
    ```

  - POST /playlists – Создание нового плейлиста

    Request:
    ```
    {
     "name": "My Awesome Playlist",
     "description": "The best tracks ever",
     "isPublic": true,
     "isCollaborative": false,
     "coverImageUrl": "https://..."
    }
    ```

    Response: 201 Created
    ```
    { Playlist object }
    ```

  - GET /playlists/{playlistId} – Получение плейлиста

    Response: 200 OK
    ```
    {
    ```

```
 "playlist": { Playlist object },
 "tracks": [{ Track objects }]
}
```

- PATCH /playlists/{playlistId} – Обновление плейлиста

```
Request:
{
 "name": "Updated Name",
 "isPublic": false
}

Response: 200 OK
{ Updated Playlist object }
```

- DELETE /playlists/{playlistId} – Удаление плейлиста

```
Response: 204 No Content
```

- POST /playlists/{playlistId}/tracks – Добавление треков в плейлист

```
Request:
{
 "trackIds": ["uuid1", "uuid2", "uuid3"],
 "position": 0 (optional, default: end)
}

Response: 200 OK
{
 "playlist": { Updated Playlist object },
 "addedTracks": 3
}
```

- DELETE /playlists/{playlistId}/tracks/{trackId} – Удаление трека из плейлиста

```
Response: 200 OK
{
 "playlist": { Updated Playlist object },
 "removed": true
}
```

- POST /playlists/{playlistId}/follow – Подписка на плейлист

```
Response: 200 OK
{
 "following": true,
```

```
  "followers": 1234
}
```

- POST /playlists/{playlistId}/collaborators – Добавление коллаборатора в плейлист

```
Request:
{
  "userId": "uuid"
}

Response: 200 OK
{
  "collaborators": ["uuid1", "uuid2"],
  "added": true
}
```

- *Артисты:*
  - GET /artists – Поиск артистов

```
Query params:
  ?q=search_query
  &genre=rock
  &verified=true
  &limit=20

Response: 200 OK
{
  "artists": [{ Artist objects }],
  "total": 156
}
```

  - GET /artists/{artistId} – Получение информации об артисте

```
Response: 200 OK
{
  "artist": { Artist object },
  "topTracks": [{ Track objects }],
  "albums": [{ Album objects }]
}
```

  - POST /artists/{artistId}/follow – Подписка на артиста

```
Response: 200 OK
{
  "following": true,
  "followers": 1000000
}
```

- GET /artists/{artistId}/related – Получение похожих артистов

```
Response: 200 OK
{
 "artists": [{ Artist objects }]
}
```

- *Комнаты настроения:*
  - GET /rooms – Получение списка активных комнат

```
Query params:
 ?mood=party
 &isPublic=true
 &hasSpace=true
 &limit=20

Response: 200 OK
{
 "rooms": [{ MoodRoom objects }],
 "total": 234
}
```

  - POST /rooms – Создание новой комнаты

```
Request:
{
 "name": "Chill Evening Vibes",
 "mood": "chill",
 "isPublic": true,
 "maxParticipants": 50,
 "chatEnabled": true,
 "initialTrackId": "uuid (optional)"
}

Response: 201 Created
{ MoodRoom object }
```

  - GET /rooms/{roomId} – Получение информации о комнате

```
Response: 200 OK
{
 "room": { MoodRoom object },
 "currentTrack": { Track object },
 "participants": [{ User objects (minimal) }]
}
```

- POST /rooms/{roomId}/join – Присоединение к комнате

```
Response: 200 OK
{
 "room": { MoodRoom object },
 "wsUrl": "wss://ws.soundwave.io/rooms/{roomId}?token=..."
}
```

- POST /rooms/{roomId}/leave – Выход из комнаты

```
Response: 200 OK
{
 "left": true
}
```

- POST /rooms/{roomId}/queue – Добавление трека в очередь комнаты

```
Request:
{
 "trackId": "uuid"
}

Response: 200 OK
{
 "queue": ["uuid1", "uuid2", "uuid3"],
 "position": 2
}
```

- POST /rooms/{roomId}/skip – Пропуск текущего трека (только хост)

```
Response: 200 OK
{
 "skipped": true,
 "nextTrack": { Track object }
}
```

- *Рекомендации:*
  - GET /recommendations/for-you – Персонализированные рекомендации

```
Query params:
 ?mood=happy (optional)
 &context=morning|workout|commute (optional)
 &limit=20

Response: 200 OK
{
```

```
 "recommendations": [
  {
   "track": { Track object },
   "score": 0.95,
   "reason": "Based on your recent listening to indie rock"
  }
 ],
 "generatedAt": "datetime"
}
```

- GET /recommendations/mood-based – Рекомендации по настроению

```
Request Query params:
 ?mood=energetic&limit=30

Response: 200 OK
{
 "mood": "energetic",
 "tracks": [{ Track objects }],
 "moodProfile": {
  "energy": 0.9,
  "valence": 0.8,
  "tempo": "fast"
 }
}
```

- POST /recommendations/feedback – Отправка обратной связи о рекомендации

```
Request:
{
 "recommendationId": "uuid",
 "feedback": "liked|disliked|skipped",
 "reason": "not_my_taste" (optional)
}

Response: 200 OK
{
 "recorded": true
}
```

- *Поиск:*
  - GET /search – Универсальный поиск

```
Query params:
 ?q=search_query
 &type=all|track|artist|playlist|album
 &limit=20
```

Response: 200 OK
```
{
 "tracks": [{ Track objects }],
 "artists": [{ Artist objects }],
 "playlists": [{ Playlist objects }],
 "albums": [{ Album objects }],
 "total": {
  "tracks": 150,
  "artists": 23,
  "playlists": 45,
  "albums": 67
 }
}
```

- *Библиотека пользователя:*
  - GET /me/library/tracks – Получение любимых треков

    Query params: ?limit=50&offset=0

    Response: 200 OK
    ```
    {
     "tracks": [{ Track objects }],
     "total": 532
    }
    ```

  - GET /me/library/playlists – Получение плейлистов пользователя

    Response: 200 OK
    ```
    {
     "owned": [{ Playlist objects }],
     "following": [{ Playlist objects }],
     "collaborative": [{ Playlist objects }]
    }
    ```

  - GET /me/library/artists – Получение отслеживаемых артистов

    Response: 200 OK
    ```
    {
     "artists": [{ Artist objects }],
     "total": 87
    }
    ```

  - GET /me/listening-history – История прослушивания

    Query params:
     ?from=2024-01-01

```
 &to=2024-12-31
 &limit=100

Response: 200 OK
{
 "history": [
  {
   "track": { Track object },
   "playedAt": "datetime",
   "duration": 180,
   "context": "playlist"
  }
 ],
 "total": 1543
}
```

5. ***Обязательные поля и статус-коды ошибок***
    - При регистрации (POST /auth/register):
        - email – электронная почта (строка, уникальная, валидный email-формат)
        - password – пароль (строка, минимум 8 символов, обязательно должна содержать хотя бы одну заглавную букву, одну строчную и одну цифру)
        - username – имя пользователя (строка, от 3 до 30 символов, только латинские буквы, цифры и подчеркивание _, уникальное)
        - displayName – отображаемое имя (строка от 1 до 100 символов, может содержать пробелы, эмодзи и любые символы)
    - При создании плейлиста (POST /playlists):
        - name – название плейлиста (строка от 1 до 200 символов, обязательно)
        - isPublic – публичный ли плейлист (булево, по умолчанию false = приватный)
        - isCollaborative – совместный ли плейлист (булево, по умолчанию false = только владелец может редактировать)

- description – описание (строка, опционально, до 1000 символов)
- coverImageUrl – обложка (строка-ссылка на картинку, опционально)

- При воспроизведении трека (POST /tracks/{trackId}/play):
  - playbackContext – контекст воспроизведения (обязательное перечисление, возможные значения: "playlist", "album", "radio", "search", "artist", "for-you", "mood-room" и т.п.)
  - contextId – ID контекста (строка-uuid, опционально, если известен плейлист/альбом/комната и т.д.)
  - position – позиция в треке, с которой началось воспроизведение (целое число в секундах, опционально, по умолчанию 0)

- Валидация:

```
{
 "email": {
  "type": "string",
  "format": "email",
  "maxLength": 255,
  "required": true
 },
 "password": {
  "type": "string",
  "minLength": 8,
  "maxLength": 128,
  "pattern": "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d).+$",
  "required": true
 },
 "username": {
  "type": "string",
  "minLength": 3,
  "maxLength": 30,
  "pattern": "^[a-zA-Z0-9_]+$",
  "required": true
 }
}
```

- Коды ошибок приложения:

Код	Значение	Использование
200	OK	Успешные GET, PATCH
201	Created	Успешный POST (creation)
204	No Content	Успешный DELETE
400	Bad Request	Неверный синтаксис/параметры запроса
401	Unauthorized	Отсутствующая или неверная аутентификация
403	Forbidden	Верная аутентификация, но недостаточно прав
404	Not Found	Ресурс не существует
409	Conflict	Конфликт ресурсов(дублирование)
422	Unprocessable Entity	Верный синтаксис, семантические ошибки
429	Too Many Requests	Переизбыток запросов
500	Internal Server Error	Ошибка на стороне сервера
503	Service Unavailable	Временное отсутствие доступа

6. Правила безопасности

- Аутентификация и авторизация:
    - требование JWT-токенов (JSON Web Tokens) для всех endpoints, кроме /auth/register и /auth/login;
    - выдача токенов при регистрации или входе;
    - передача токенов в заголовке Authorization: Bearer {token};
    - срок действия accessToken – 1 час, refreshToken – 7 дней;
    - обновление accessToken через endpoint /auth/refresh (POST).
- Ограничение запросов (Rate Limiting):
    - лимит в 100 запросов в минуту на пользователя для бесплатной подписки (free tier);
    - лимит в 500 запросов в минуту для premium;
    - возврат статуса 429 Too Many Requests при превышении;

    - отслеживание лимитов по IP-адресу и токену.
- Защита от уязвимостей:
    - передача всех запросов только по HTTPS;
    - валидация входных данных с использованием схем для предотвращения инъекций (SQL, XSS);
    - применение библиотек для серверной валидации;
    - настройка CORS для доступа только с доверенных доменов;
    - хэширование чувствительной информации (паролей) с использованием bcrypt;
    - генерация временных и подписанных аудио-URL.
- Мониторинг и логирование:
    - логирование всех запросов (без хранения паролей), включая IP, endpoint и статус;
    - временная блокировка аккаунта при подозрительной активности (много ошибок 401/403).
- Доступ по ролям:
    - доступ обычных пользователей только к собственным ресурсам (профиль, плейлисты);
    - управление артистами с verified=true своими треками и альбомами через дополнительные endpoints;
    - полный доступ администраторов через claims в JWT.

7. Стратегия сопровождения API
    - Версионирование API по URL (/v1, /v2 и т.д.) с сохранением обратной совместимости минимум 12 месяцев для предыдущей версии.
    - Автоматическое обновление документации в формате OpenAPI 3.0 при каждой развертке.

- Непрерывная интеграция и доставка с обязательными тестами (unit, integration, contract).
- Мониторинг здоровья API (SLA 99,9 %), уведомление и сбор метрик.
- Обработка обратной связи через портал разработчиков (developer.soundwave.io) и автоматический сбор ошибок.
- Регулярный аудит безопасности раз в квартал.