**FastAPI 그룹 과제 산출물 2 : API 명세서**

조원 : 나영일 장유민 이재훈 정재철

주제 : 음원 스트리밍 사이트

기본 URL : <http://localhost:8000>

인증 방식

* 로그인(POST /users/login) 성공 시, 응답 헤더를 통해 access\_token과 refresh\_token이 HttpOnly 쿠키로 설정됨
* 인증이 필요한 모든 API는 요청 시 이 쿠키를 자동으로 포함하여 전송해야 함. 인증 실패 시 401 Unauthorized 응답을 반환

1. **User API**

사용자 가입, 로그인, 로그아웃 및 정보 조회를 담당

**1. 회원가입**

* **Endpoint**: POST /users/signup
* **Description**: 새로운 사용자를 생성합니다.
* **Request Body**:

|  |  |  |  |
| --- | --- | --- | --- |
| 필드명 | 타입 | 필수 | 설명 |
| email | string | O | 사용자 이메일 (고유) |
| username | string | O | 사용자 이름 (고유) |
| password | string | O | 비밀번호 |

* **Successful Response**: 200 OK
  + **Body**: 생성된 사용자 정보 (UserRead 스키마)

JSON

{

"email": "user@example.com",

"username": "string",

"id": 1,

"role": "USER"

}

* **Error Responses**:
  + 400 Bad Request: 이메일 또는 사용자 이름이 이미 존재할 경우

**2. 로그인**

* **Endpoint**: POST /users/login
* **Description**: 사용자 로그인 후 인증 쿠키를 발급합니다.
* **Request Body**:

|  |  |  |  |
| --- | --- | --- | --- |
| 필드명 | 타입 | 필수 | 설명 |
| email | string | O | 사용자 이메일 |
| password | string | O | 비밀번호 |

* **Successful Response**: 200 OK
  + **Body**: 로그인한 사용자 정보 (UserRead 스키마)
  + **Headers**: Set-Cookie (access\_token, refresh\_token)
* **Error Responses**:
  + 401 Unauthorized: 이메일 또는 비밀번호가 일치하지 않을 경우

**3. 로그아웃**

* **Endpoint**: POST /users/logout
* **Description**: 인증 쿠키를 삭제하여 로그아웃합니다.
* **Successful Response**: 200 OK
  + **Body**: true
  + **Headers**: Set-Cookie (쿠키 삭제)

**4. 내 정보 조회**

* **Endpoint**: GET /users/me
* **Description**: 현재 로그인된 사용자의 정보를 조회합니다.
* **Authentication**: **Required**
* **Successful Response**: 200 OK
  + **Body**: 현재 사용자 정보 (UserRead 스키마)
* **Error Responses**:
  + 401 Unauthorized: 로그인되어 있지 않은 경우

==========================================================

1. **Song API**

노래 정보의 조회 및 관리를 담당

**1. 노래 생성**

* **Endpoint**: POST /songs/
* **Description**: 새로운 노래 정보를 추가합니다. **(관리자 전용)**
* **Authentication**: **Required (Admin Role)**
* **Request Body**:

|  |  |  |  |
| --- | --- | --- | --- |
| 필드명 | 타입 | 필수 | 설명 |
| title | string | O | 노래 제목 |
| artist | string | O | 아티스트 |
| duration | integer | O | 재생 시간 (초) |

* **Successful Response**: 201 Created
  + **Body**: 생성된 노래 정보 (SongRead 스키마)
* **Error Responses**:
  + 401 Unauthorized: 로그인되어 있지 않은 경우
  + 403 Forbidden: 관리자 권한이 없는 경우

**2. 모든 노래 목록 조회**

* **Endpoint**: GET /songs/
* **Description**: 시스템에 등록된 모든 노래 목록을 조회합니다.
* **Successful Response**: 200 OK
  + **Body**: 노래 정보 배열 (List[SongRead])

**3. 특정 노래 조회**

* **Endpoint**: GET /songs/{song\_id}
* **Description**: ID로 특정 노래의 상세 정보를 조회합니다.
* **Successful Response**: 200 OK
  + **Body**: 노래 상세 정보 (SongRead 스키마)
* **Error Responses**:
  + 404 Not Found: 해당 ID의 노래가 없을 경우

**4. 노래 정보 수정**

* **Endpoint**: PATCH /songs/{song\_id}
* **Description**: 특정 노래의 정보를 수정합니다. **(관리자 전용)**
* **Authentication**: **Required (Admin Role)**
* **Request Body**: SongUpdate 스키마 (모든 필드는 선택적)
* **Successful Response**: 200 OK
  + **Body**: 수정된 노래 정보 (SongRead 스키마)
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found

**5. 노래 삭제**

* **Endpoint**: DELETE /songs/{song\_id}
* **Description**: 특정 노래를 삭제합니다. **(관리자 전용)**
* **Authentication**: **Required (Admin Role)**
* **Successful Response**: 204 No Content
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found

==========================================================

1. **Playlist API**

사용자 플레이리스트의 생성, 조회, 수정, 삭제 및 노래 관리를 담당

**1. 플레이리스트 생성**

* **Endpoint**: POST /playlists/
* **Description**: 현재 로그인된 사용자의 새 플레이리스트를 생성합니다.
* **Authentication**: **Required**
* **Request Body**:

|  |  |  |  |
| --- | --- | --- | --- |
| 필드명 | 타입 | 필수 | 설명 |
| name | string | O | 플레이리스트 이름 |
| desc | string | X | 플레이리스트 설명 |

* **Successful Response**: 201 Created
  + **Body**: 생성된 플레이리스트 정보 (PlaylistRead 스키마)
* **Error Responses**:
  + 401 Unauthorized

**2. 내 플레이리스트 목록 조회**

* **Endpoint**: GET /playlists/
* **Description**: 현재 로그인된 사용자의 모든 플레이리스트 목록을 조회합니다.
* **Authentication**: **Required**
* **Successful Response**: 200 OK
  + **Body**: 플레이리스트 정보 배열 (List[PlaylistRead])

**3. 특정 플레이리스트 조회**

* **Endpoint**: GET /playlists/{playlist\_id}
* **Description**: 특정 플레이리스트의 상세 정보를 조회합니다.
* **Authentication**: **Required (Owner Only)**
* **Successful Response**: 200 OK
  + **Body**: 플레이리스트 상세 정보 (PlaylistRead 스키마)
* **Error Responses**:
  + 401 Unauthorized
  + 403 Forbidden: 소유자가 아닐 경우
  + 404 Not Found

**4. 플레이리스트 정보 수정**

* **Endpoint**: PATCH /playlists/{playlist\_id}
* **Description**: 플레이리스트 정보를 수정합니다.
* **Authentication**: **Required (Owner Only)**
* **Request Body**: PlaylistUpdate 스키마 (모든 필드는 선택적)
* **Successful Response**: 200 OK
  + **Body**: 수정된 플레이리스트 정보 (PlaylistRead 스키마)
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found

**5. 플레이리스트 삭제**

* **Endpoint**: DELETE /playlists/{playlist\_id}
* **Description**: 플레이리스트를 삭제합니다.
* **Authentication**: **Required (Owner Only)**
* **Successful Response**: 204 No Content
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found

**6. 플레이리스트에 노래 추가**

* **Endpoint**: POST /playlists/{playlist\_id}/songs/{song\_id}
* **Description**: 플레이리스트에 노래를 추가합니다.
* **Authentication**: **Required (Owner Only)**
* **Successful Response**: 200 OK
  + **Body**: 노래가 추가된 플레이리스트 정보 (PlaylistRead 스키마)
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found

**7. 플레이리스트에서 노래 제거**

* **Endpoint**: DELETE /playlists/{playlist\_id}/songs/{song\_id}
* **Description**: 플레이리스트에서 노래를 제거합니다.
* **Authentication**: **Required (Owner Only)**
* **Successful Response**: 200 OK
  + **Body**: 노래가 제거된 플레이리스트 정보 (PlaylistRead 스키마)
* **Error Responses**:
  + 401 Unauthorized, 403 Forbidden, 404 Not Found