



Kauno technologijos universitetas

Informatikos Fakultetas

Saityno taikomųjų programų projektavimas (T120B165)

Projekto „Music app“ ataskaita

Vitalijus Pamakštys

Studentas / Studentė

Kaunas 2025

Turinys

1. Sprendžiamo uždavinio aprašymas	3
2. Funkciniai reikalavimai	3
3. Sistemos architektūra	3
4. Naudotojo sąsajos projektas.....	4
projektuojamos sąsajos langų wireframes	4
realizacijos langų iškarpos	9
5. API specifikacija.....	13
6. Projekto išvados.....	32

1. Sprendžiamo uždavinio aprašymas

Projekto tikslas – suteikti galimybę vartotojams tvarkyti muzikos duomenis (atlikėjus, albumus ir dainas), dalintis savo kolekcijomis bei lengvai atrasti naują muziką.

Veikimo principas – pačią kuriamą platformą sudaro dvi dalys: internetinė aplikacija, kuria naudosis vartotojai, „Publisher“ rolę turintys vartotojai bei administratorius ir aplikacijų programavimo sąsaja (API).

Registruotas naudotojas galės naršyti muzikos įrašus, o tie, kuriems administratorius suteiks **Publisher** rolę, galės kurti atlikėjus, albumus bei įkelti dainas.

2. Funkciniai reikalavimai

Neregistruotas sistemos naudotojas galės:

1. Peržiūrėti platformos pradinį puslapį
2. Prisiregistruoti prie internetinės aplikacijos

Registruotas sistemos naudotojas galės:

1. Atsijungti nuo internetinės aplikacijos
2. Prisijungti prie platformos
3. Peržiūrėti kitų naudotojų paskelbtus atlikėjus, albumus ir dainas

Naudotojas, turintis **Publisher** rolę, galės:

1. Kurti / tvarkyti savo muzikos kolekciją:
 - 1.1. Pridėti atlikėją (**Artist**)
 - 1.2. Pridėti albumą (**Album**) ir susieti su atlikėju
 - 1.3. Pridėti dainą (**Song**) ir susieti su albumu
2. Redaguoti ir šalinti savo pridėtus atlikėjus, albumus ir dainas

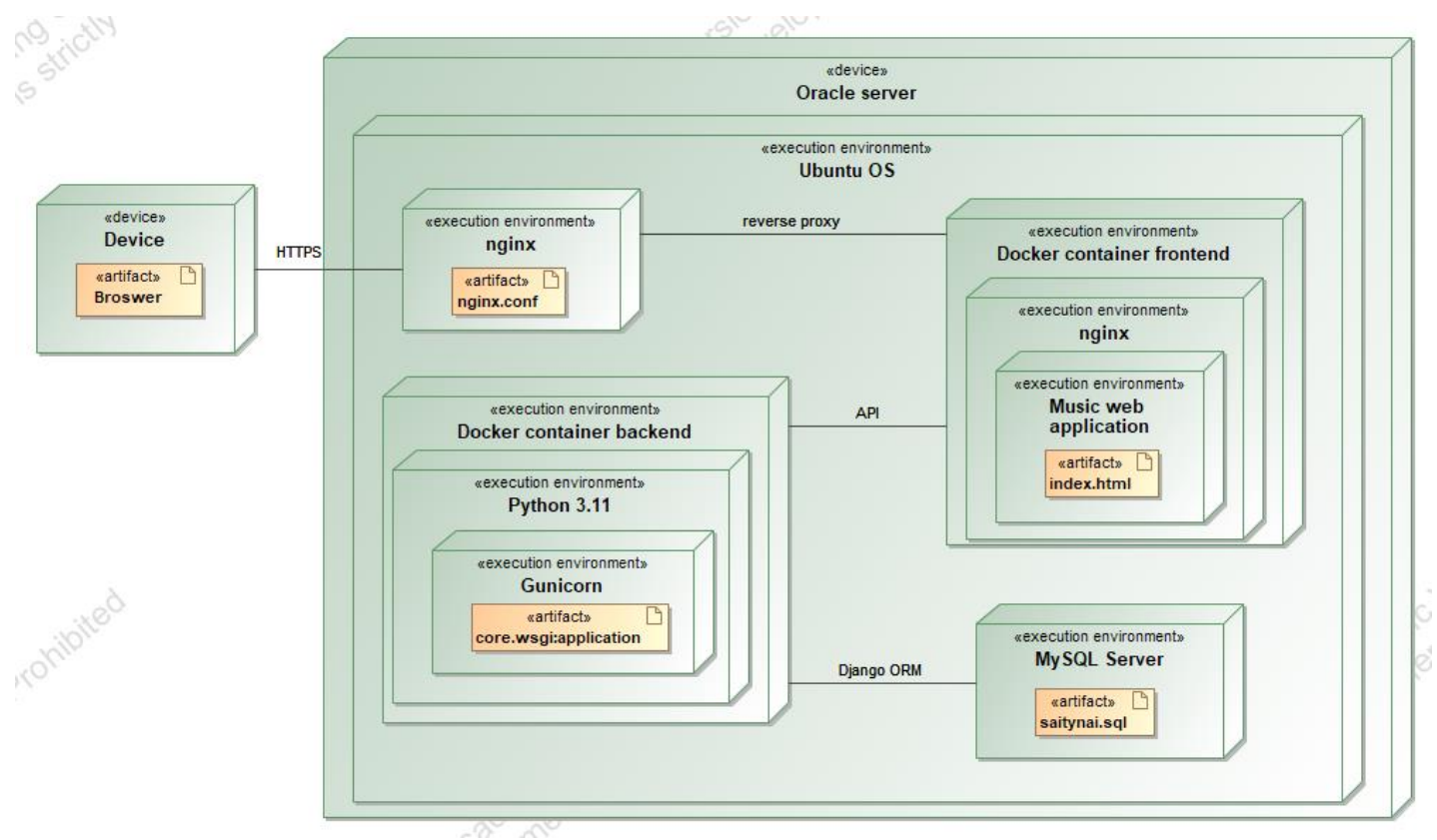
Administratorius galės:

1. Suteikti ar atimti **Publisher** rolę
2. Šalinti naudotojus
3. Šalinti netinkamus įrašus (atlikėjus, albumus, dainas)

3. Sistemos architektūra

Sistemos sudedamosios dalys:

- Kliento pusė (angl. Front-end) – naudojant React bei typescript technologijas.
- Serverio pusė (angl. Back-End) – naudojant Python django technologiją.
- Duomenų bazė (angl. Database) – naudojant MySQL technologiją.



4. Naudotojo sąsajos projektas projektuojamos sąsajos langų wireframes

LOGIN

Email

Password

Login

[Don't have an account? Register here](#)

Prisijungimo langas

REGISTER

Email

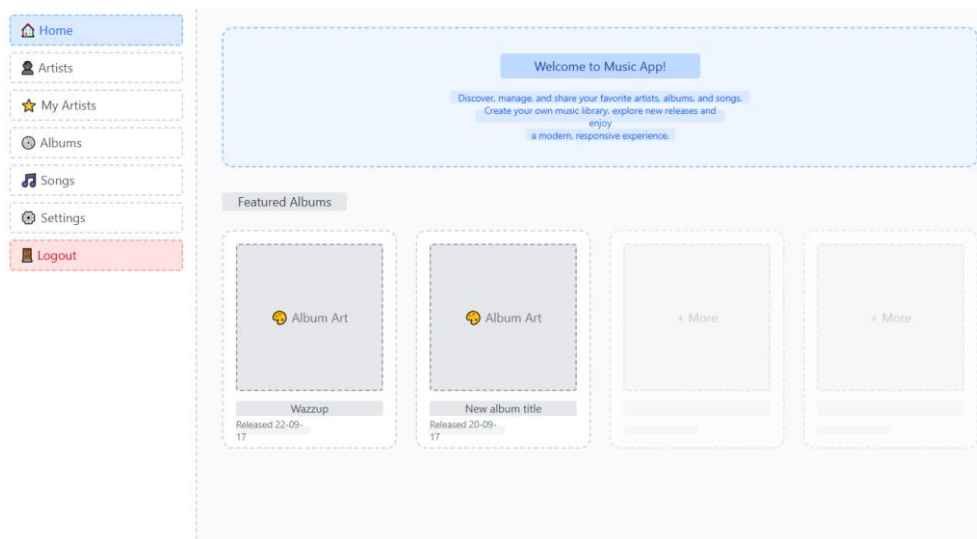
Password

Name

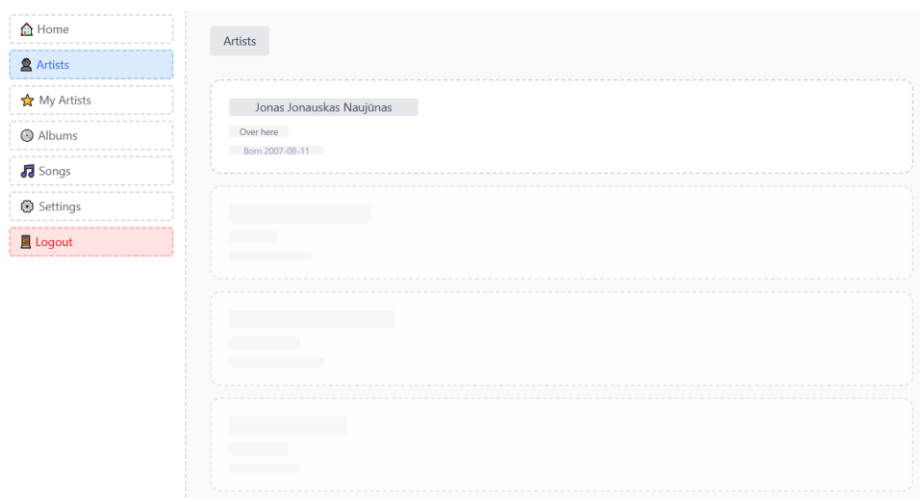
Register

Already have an account? [Login here](#)

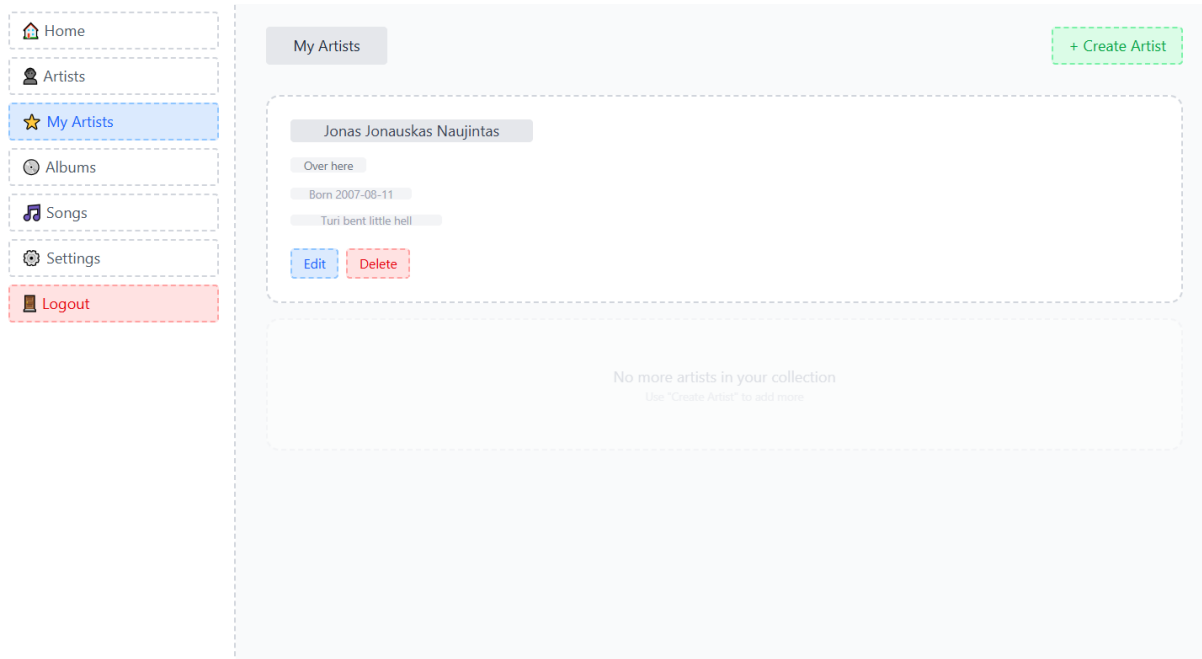
Registracijos langas



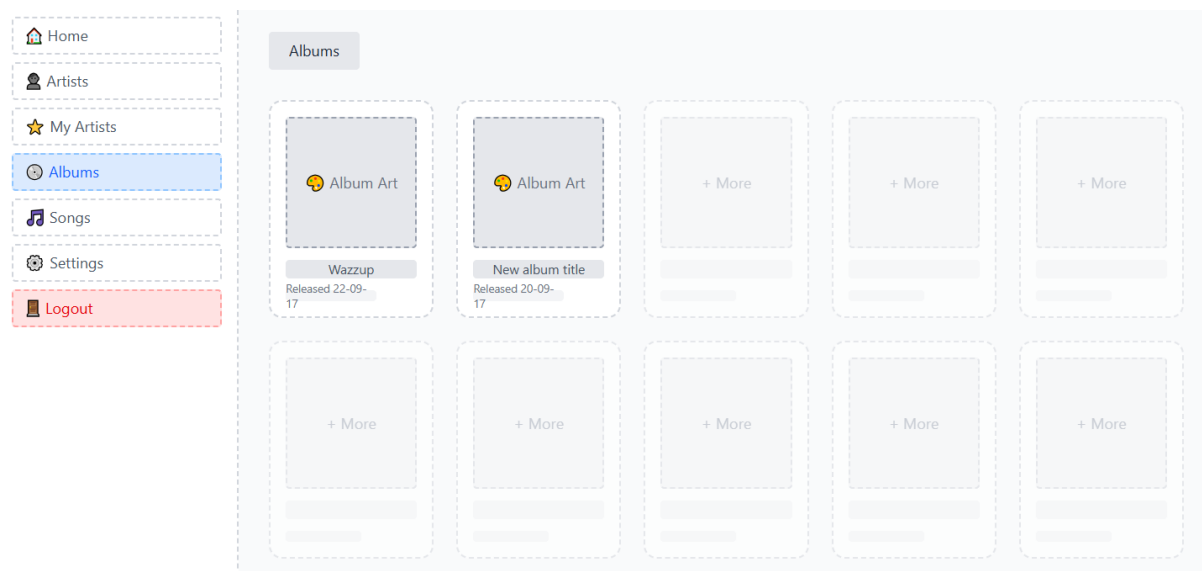
Pagrindinis langas



„Artists“ langas



„My artists“ langas



„Albums“ langas

← Go
Back

Edit Artist

Name

vardas pavarde

Country

salis

Birth Date

data

Bio

Aprasymas

Save

„Edit“ langas

Songs

Art

Melody

Genre: JAZZ

0:00 / 0:05

EditDelete

Art

music

Genre: CLASSICAL

0:00 / 0:04

EditDelete

Art

Song title

Genre: POP

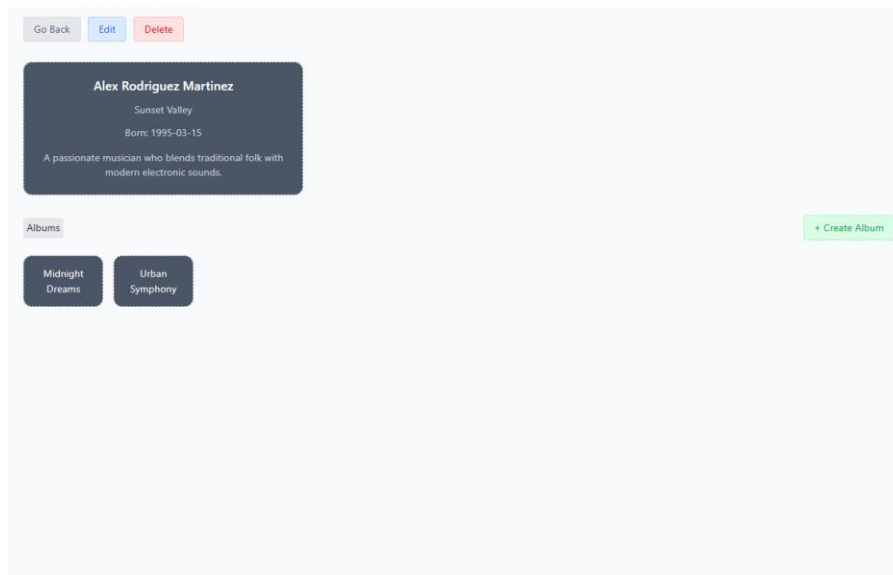
Duration: 00:00:10

0:00 / 0:01

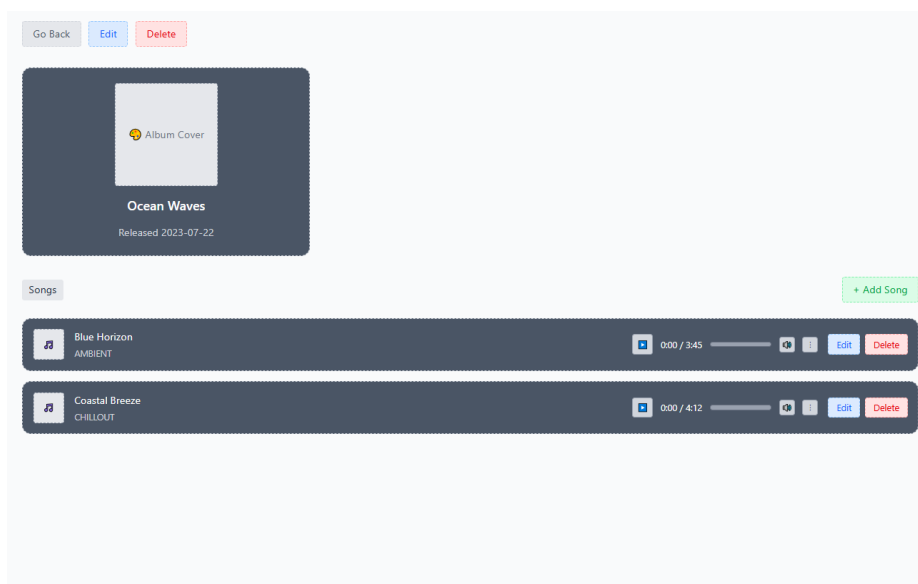
EditDelete

+ Add More Songs

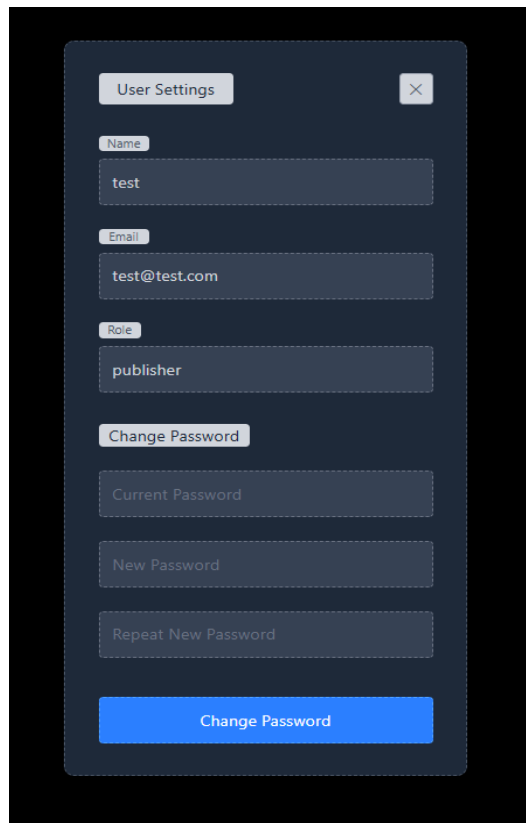
„Songs“ langas



„View artist“ langas



„View album“ langas

A dark-themed user settings form with a title bar 'User Settings' and a close button 'X'. It contains three input fields: 'Name' with the value 'test', 'Email' with the value 'test@test.com', and 'Role' with the value 'publisher'. Below these is a 'Change Password' section with three input fields: 'Current Password', 'New Password', and 'Repeat New Password'. A blue 'Change Password' button is at the bottom.

User Settings

Name

test

Email

test@test.com

Role

publisher

Change Password

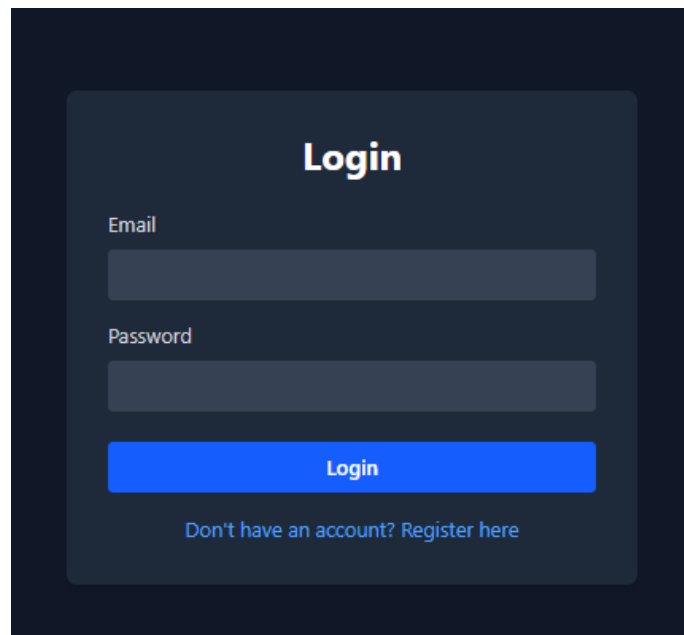
Current Password

New Password

Repeat New Password

Change Password

Nustatymu langas
realizacijos langų iškarpos

A dark-themed login form with the title 'Login'. It has two input fields: 'Email' and 'Password'. A blue 'Login' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? Register here'.

Login

Email

Password

Login

Don't have an account? Register here

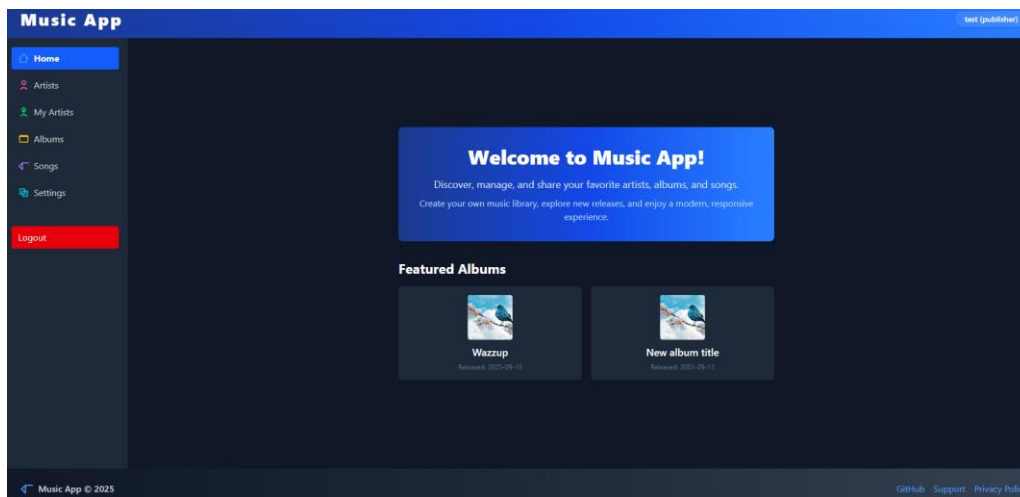
Prisijungimo langas

Register

Register

Already have an account? [Login here](#)

Registracijos langas



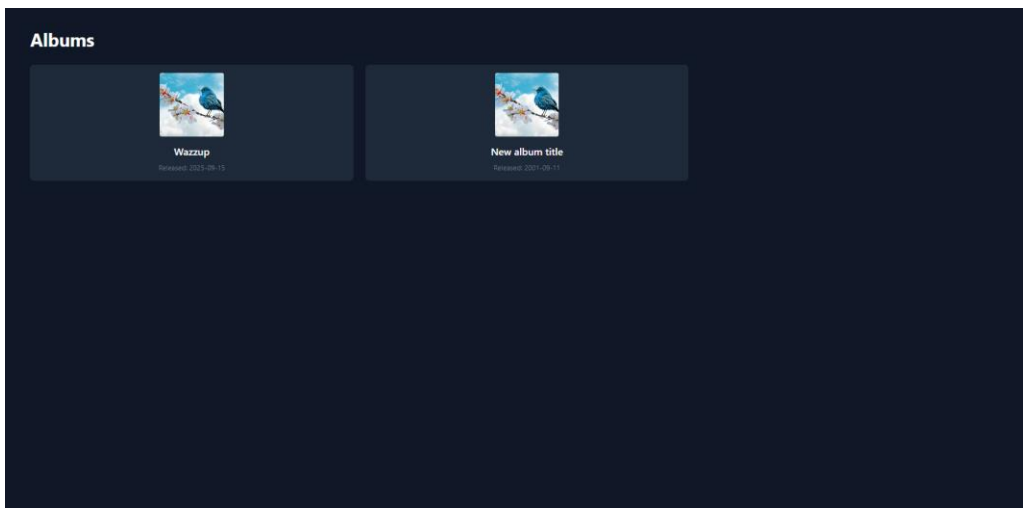
Pagrindinis langas



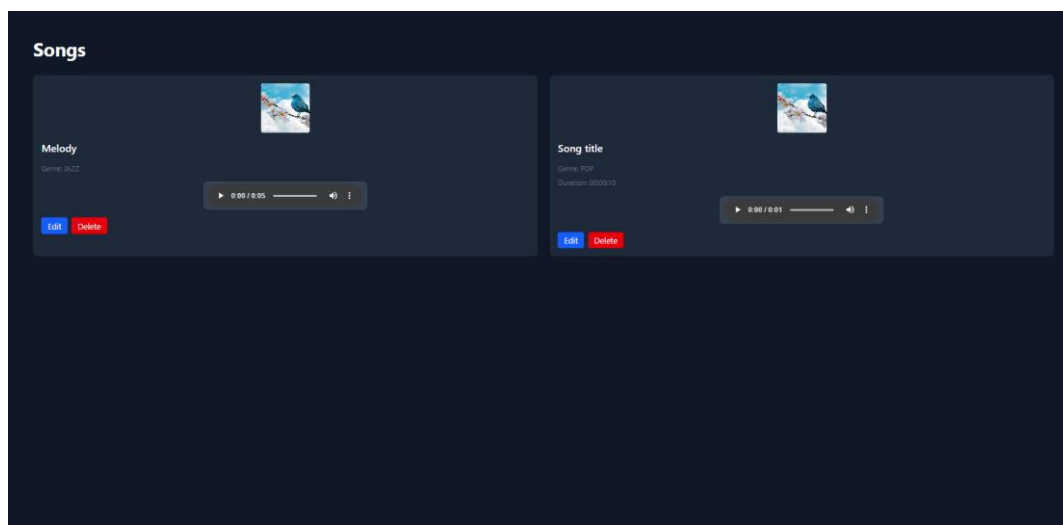
„Artists“ langas



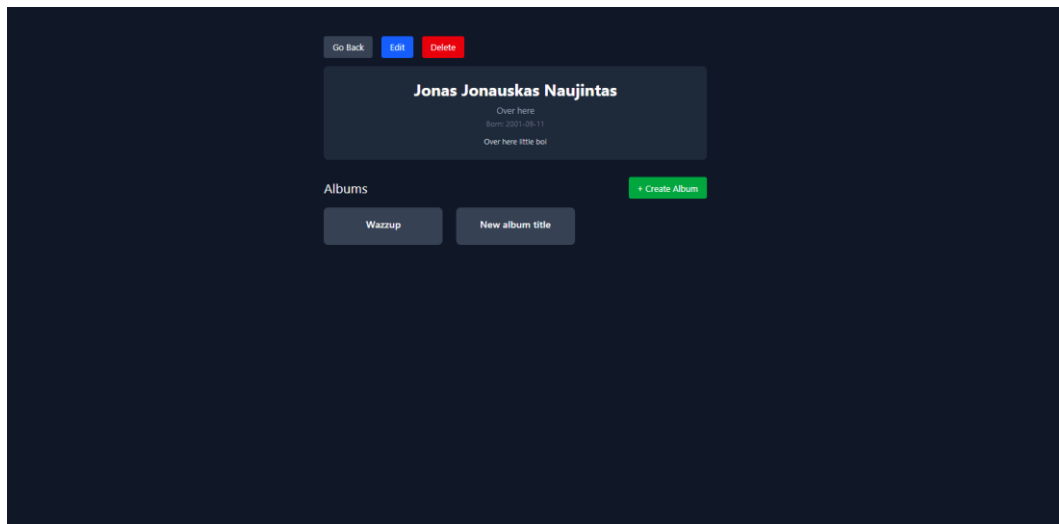
„My artists“ langas



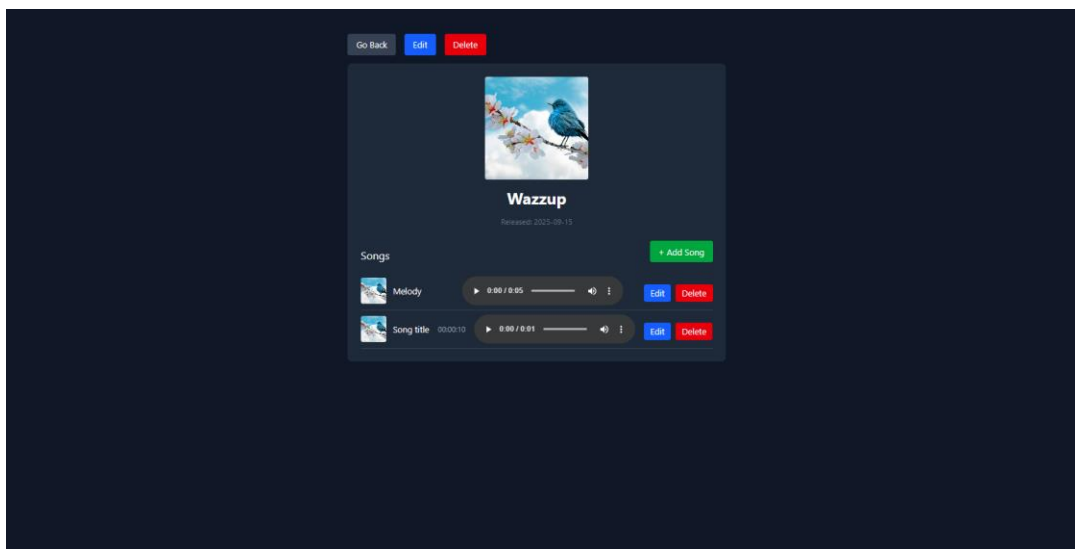
„Albums“ langas



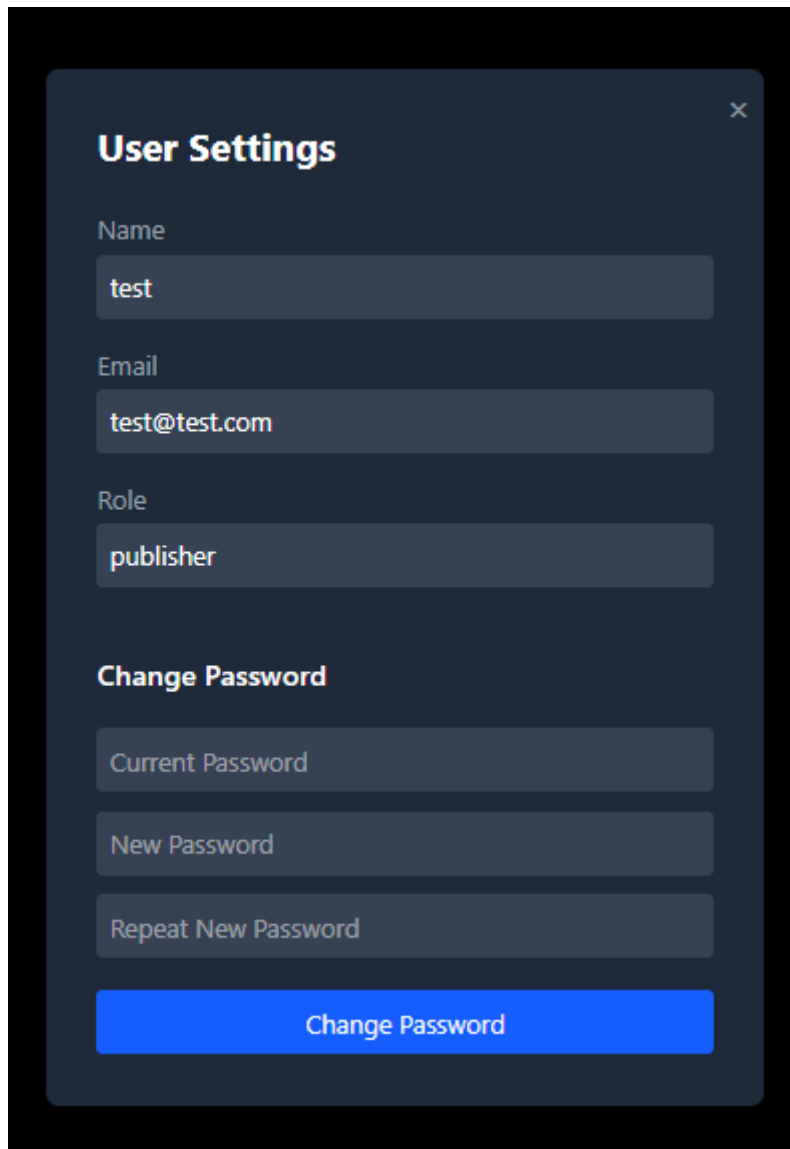
„Songs“ langas



„View artist“ langas



„View album“ langas

A dark-themed modal dialog box titled "User Settings" with a close button (X) in the top right corner. It contains three input fields: "Name" with the value "test", "Email" with the value "test@test.com", and "Role" with the value "publisher". Below these is a section titled "Change Password" which includes three input fields: "Current Password", "New Password", and "Repeat New Password". At the bottom of this section is a blue button labeled "Change Password".

User Settings

Name

test

Email

test@test.com

Role

publisher

Change Password

Current Password

New Password

Repeat New Password

Change Password

Nustatymu langas

5. API specifikacija

Norint peržiūrėti API specifikaciją naudoti šias nuorodas:

- <https://saitynai.pamakstys.tech/api/docs>
- <https://saitynai.pamakstys.tech/docs/redoc>

Api-spec.yaml

```
openapi: 3.0.3
info:
  title: Music App API
  version: 1.0.0
  description: |
    A comprehensive REST API for managing music content including artists,
    albums, and songs.
```

Authentication

This API uses JWT (JSON Web Tokens) for authentication via HTTP-only cookies.

Cookie-based Authentication:

- Login via `/login` endpoint to receive JWT cookies
- Cookies are automatically included in subsequent requests
- Logout via `/logout` to clear authentication cookies

****Note:**** This API does not support Authorization Bearer headers - authentication is handled exclusively through cookies.

Features

- User registration and authentication
- Artist management with publisher permissions
- Album and song CRUD operations
- File uploads for cover images and audio files
- Role-based access control

servers:

- url: <https://saitynai.pamakstys.tech/api>
description: Production server
- url: <http://localhost:8001/api>
description: Development server

paths:

Authentication Endpoints

/register:

post:

summary: Register a new user

description: Create a new user account

tags:

- Authentication

security: []

requestBody:

required: true

content:

application/json:

schema:

type: object

required:

- email
- password
- name

properties:

email:

type: string

```
        format: email
        description: User's email address
    password:
        type: string
        minLength: 8
        description: Password (minimum 8 characters)
    name:
        type: string
        description: User's full name
    example:
        email: "john.doe@example.com"
        password: "securepassword123"
        name: "John Doe"
responses:
  '201':
    description: User created successfully
    content:
      application/json:
        schema:
          type: object
          properties:
            id:
              type: integer
            email:
              type: string
            name:
              type: string
            role:
              type: string
          example:
            id: 1
            email: "john.doe@example.com"
            name: "John Doe"
            role: "user"
  '400':
    description: Validation error
    content:
      application/json:
        schema:
          type: object
          properties:
            email:
              type: array
              items:
                type: string
            password:
              type: array
              items:
```

```

        type: string
    example:
        email: ["This field is required."]
        password: ["This field must be at least 8 characters long."]

/login:
  post:
    summary: User login
    description: Authenticate user and receive JWT tokens
    tags:
      - Authentication
    security: []
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - email
              - password
            properties:
              email:
                type: string
                format: email
              password:
                type: string
            example:
              email: "john.doe@example.com"
              password: "securepassword123"
    responses:
      '200':
        description: Login successful
        content:
          application/json:
            schema:
              type: object
              properties:
                jwt:
                  type: string
                  description: Access token (15 minutes)
                refresh_token:
                  type: string
                  description: Refresh token (7 days)
            example:
              jwt: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
              refresh_token: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
      '401':

```



```
description: Invalid credentials
content:
  application/json:
    schema:
      type: object
      properties:
        detail:
          type: string
    example:
      detail: "Incorrect password"
```

/refresh/:

```
post:
  summary: Refresh JWT token
  description: Get a new access token using refresh token
  tags:
    - Authentication
  security: []
  responses:
    '200':
      description: Token refreshed successfully
      content:
        application/json:
          schema:
            type: object
            properties:
              jwt:
                type: string
                description: New access token
          example:
            jwt: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
    '401':
      description: Invalid or expired refresh token
      content:
        application/json:
          schema:
            type: object
            properties:
              detail:
                type: string
          example:
            detail: "Refresh token expired"
```

/logout:

```
post:
  summary: User logout
  description: Logout user and clear tokens
  tags:
```

```
- Authentication
security: []
responses:
  '200':
    description: Logout successful
    content:
      application/json:
        schema:
          type: object
          properties:
            message:
              type: string
          example:
            message: "success"

/user:
  get:
    summary: Get current user
    description: Retrieve current authenticated user information
    tags:
      - User
    responses:
      '200':
        description: User information retrieved successfully
        content:
          application/json:
            schema:
              type: object
              properties:
                id:
                  type: integer
                email:
                  type: string
                name:
                  type: string
                role:
                  type: string
            example:
              id: 1
              email: "john.doe@example.com"
              name: "John Doe"
              role: "publisher"
      '401':
        description: Unauthorized - invalid or missing token

/change-password:
  post:
    summary: Change user password
```

```
description: Change the current user's password
tags:
  - User
requestBody:
  required: true
  content:
    application/json:
      schema:
        type: object
        required:
          - old_password
          - new_password
        properties:
          old_password:
            type: string
          new_password:
            type: string
            minLength: 8
      example:
        old_password: "oldpassword123"
        new_password: "newsecurepassword456"
responses:
  '200':
    description: Password changed successfully
  '400':
    description: Invalid old password or validation error
  '401':
    description: Unauthorized
```

Artist Endpoints

```
/artists/:
  get:
    summary: List all artists
    description: Retrieve a list of all artists
    tags:
      - Artists
    responses:
      '200':
        description: List of artists retrieved successfully
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Artist'
            example:
              - id: 1
                name: "The Beatles"
```

```
    bio: "British rock band formed in Liverpool in 1960"
    birth_date: "1960-01-01"
    country: "United Kingdom"
    publisher: 1
  - id: 2
    name: "Elvis Presley"
    bio: "American singer and cultural icon"
    birth_date: "1935-01-08"
    country: "United States"
    publisher: 2
```

/artists/create/:

post:

summary: Create a new artist

description: Create a new artist (requires publisher role)

tags:

- Artists

requestBody:

required: true

content:

application/json:

schema:

type: object

required:

- name

properties:

name:

type: string

maxLength: 100

bio:

type: string

birth_date:

type: string

format: date

country:

type: string

maxLength: 100

example:

name: "New Artist"

bio: "An emerging artist in the music industry"

birth_date: "1990-05-15"

country: "Canada"

responses:

'201':

description: Artist created successfully

content:

application/json:

schema:

```

        $ref: '#/components/schemas/Artist'
    '400':
        description: Validation error
    '401':
        description: Unauthorized
    '403':
        description: Forbidden - requires publisher role

/artists/get/:
  get:
    summary: Get artist by ID
    description: Retrieve a specific artist by ID (passed as query
parameter)
    tags:
      - Artists
    parameters:
      - in: query
        name: id
        required: true
        schema:
          type: integer
        description: Artist ID
        example: 1
    responses:
      '200':
        description: Artist retrieved successfully
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Artist'
      '404':
        description: Artist not found

/artists/edit/:
  post:
    summary: Edit artist
    description: Update an existing artist (requires ownership or admin
role)
    tags:
      - Artists
    parameters:
      - in: query
        name: id
        required: true
        schema:
          type: integer
        description: Artist ID to edit
    requestBody:

```

```
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            name:
              type: string
              maxLength: 100
            bio:
              type: string
            birth_date:
              type: string
              format: date
            country:
              type: string
              maxLength: 100
          example:
            name: "Updated Artist Name"
            bio: "Updated biography"
  responses:
    '200':
      description: Artist updated successfully
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Artist'
    '400':
      description: Validation error
    '401':
      description: Unauthorized
    '403':
      description: Forbidden - not owner or admin
    '404':
      description: Artist not found

/artists/delete/:
  delete:
    summary: Delete artist
    description: Delete an artist (requires ownership or admin role)
    tags:
      - Artists
    parameters:
      - in: query
        name: id
        required: true
        schema:
          type: integer
```

```

        description: Artist ID to delete
responses:
  '204':
    description: Artist deleted successfully
  '401':
    description: Unauthorized
  '403':
    description: Forbidden - not owner or admin
  '404':
    description: Artist not found

/artists/my_artists/:
  get:
    summary: Get user's artists
    description: Retrieve artists owned by the current user (publishers
only)
    tags:
      - Artists
    responses:
      '200':
        description: User's artists retrieved successfully
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Artist'
      '401':
        description: Unauthorized
      '403':
        description: Forbidden - requires publisher role

/artists/is_owner/:
  get:
    summary: Check artist ownership
    description: Check if current user owns the specified artist
    tags:
      - Artists
    parameters:
      - in: query
        name: id
        required: true
        schema:
          type: integer
        description: Artist ID
    responses:
      '200':
        description: Ownership status

```

```

        content:
          application/json:
            schema:
              type: object
              properties:
                is_owner:
                  type: boolean
            example:
              is_owner: true
      '401':
        description: Unauthorized
      '404':
        description: Artist not found

# Album Endpoints
/albums/:
  get:
    summary: List all albums
    description: Retrieve a list of all albums
    tags:
      - Albums
    responses:
      '200':
        description: List of albums retrieved successfully
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Album'

/albums/create/:
  post:
    summary: Create a new album
    description: Create a new album for an artist
    tags:
      - Albums
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            type: object
            required:
              - title
              - artist_id
            properties:
              title:

```



```
        type: string
        maxLength: 100
    artist_id:
        type: integer
        description: ID of the artist
    release_date:
        type: string
        format: date
    cover_image:
        type: string
        format: binary
        description: Album cover image file
    example:
        title: "Abbey Road"
        artist_id: 1
        release_date: "1969-09-26"
responses:
    '201':
        description: Album created successfully
        content:
            application/json:
                schema:
                    $ref: '#/components/schemas/Album'
    '400':
        description: Validation error
    '401':
        description: Unauthorized
    '403':
        description: Forbidden - not artist owner

/albums/list_by_artist/:
    get:
        summary: List albums by artist
        description: Retrieve all albums for a specific artist
        tags:
            - Albums
        parameters:
            - in: query
              name: artist_id
              required: true
              schema:
                  type: integer
                  description: Artist ID
        responses:
            '200':
                description: Albums retrieved successfully
                content:
                    application/json:
```

```

        schema:
          type: array
          items:
            $ref: '#/components/schemas/Album'

# Song Endpoints
/songs/:
  get:
    summary: List all songs
    description: Retrieve a list of all songs
    tags:
      - Songs
    responses:
      '200':
        description: List of songs retrieved successfully
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Song'

/songs/create/:
  post:
    summary: Create a new song
    description: Add a new song to an album
    tags:
      - Songs
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            type: object
            required:
              - title
              - album_id
              - genre
            properties:
              title:
                type: string
                maxLength: 100
              album_id:
                type: integer
                description: ID of the album
              genre:
                $ref: '#/components/schemas/GenreEnum'
              release_date:

```

```
        type: string
        format: date
    audio_file:
        type: string
        format: binary
        description: Audio file
    cover_image:
        type: string
        format: binary
        description: Song cover image
    example:
        title: "Come Together"
        album_id: 1
        genre: "ROCK"
        release_date: "1969-09-26"
responses:
  '201':
    description: Song created successfully
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Song'
  '400':
    description: Validation error
  '401':
    description: Unauthorized
  '403':
    description: Forbidden - not album owner

/songs/list_by_album/:
  get:
    summary: List songs by album
    description: Retrieve all songs for a specific album
    tags:
      - Songs
    parameters:
      - in: query
        name: album_id
        required: true
        schema:
          type: integer
        description: Album ID
    responses:
      '200':
        description: Songs retrieved successfully
        content:
          application/json:
            schema:
```

```
        type: array
        items:
          $ref: '#/components/schemas/Song'

components:
  securitySchemes:
    JWTAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: Enter JWT token in the format "Bearer {token}"

  schemas:
    Artist:
      type: object
      properties:
        id:
          type: integer
          readOnly: true
          description: Unique identifier for the artist
        name:
          type: string
          maxLength: 100
          description: Artist name
        bio:
          type: string
          nullable: true
          description: Artist biography
        birth_date:
          type: string
          format: date
          nullable: true
          description: Artist birth date
        country:
          type: string
          nullable: true
          maxLength: 100
          description: Artist's country of origin
        publisher:
          type: integer
          description: ID of the user who published this artist
      required:
        - id
        - name
        - publisher
      example:
        id: 1
        name: "The Beatles"
```

```
bio: "British rock band formed in Liverpool in 1960"
birth_date: "1960-01-01"
country: "United Kingdom"
publisher: 1
```

Album:

```
type: object
properties:
  id:
    type: integer
    readOnly: true
    description: Unique identifier for the album
  cover_image:
    type: string
    format: uri
    nullable: true
    description: URL to album cover image
  title:
    type: string
    maxLength: 100
    description: Album title
  release_date:
    type: string
    format: date
    nullable: true
    description: Album release date
  artist:
    type: integer
    description: ID of the artist who created this album
required:
- artist
- id
- title
example:
  id: 1
  title: "Abbey Road"
  release_date: "1969-09-26"
  cover_image: "http://example.com/covers/abbey-road.jpg"
  artist: 1
```

Song:

```
type: object
properties:
  id:
    type: integer
    readOnly: true
    description: Unique identifier for the song
  audio_file:
```

```
    type: string
    format: uri
    nullable: true
    description: URL to audio file
cover_image:
    type: string
    format: uri
    nullable: true
    description: URL to song cover image
title:
    type: string
    maxLength: 100
    description: Song title
release_date:
    type: string
    format: date
    nullable: true
    description: Song release date
likes:
    type: integer
    default: 0
    description: Number of likes
duration:
    type: string
    nullable: true
    description: Song duration in format MM:SS
genre:
    $ref: '#/components/schemas/GenreEnum'
album:
    type: integer
    nullable: true
    description: ID of the album this song belongs to
required:
  - id
  - title
  - genre
example:
  id: 1
  title: "Come Together"
  release_date: "1969-09-26"
  likes: 1500
  duration: "4:20"
  genre: "ROCK"
  album: 1
  audio_file: "http://example.com/audio/come-together.mp3"
  cover_image: "http://example.com/covers/come-together.jpg"
```

GenreEnum:

```
enum:
  - POP
  - ROCK
  - JAZZ
  - CLASSICAL
  - HIPHOP
  - COUNTRY
  - ELECTRONIC
  - OTHER
type: string
description: Music genre options
```

```
PatchedArtist:
  type: object
  properties:
    name:
      type: string
      maxLength: 100
    bio:
      type: string
      nullable: true
    birth_date:
      type: string
      format: date
      nullable: true
    country:
      type: string
      nullable: true
      maxLength: 100
```

```
PatchedAlbum:
  type: object
  properties:
    cover_image:
      type: string
      format: uri
    title:
      type: string
      maxLength: 100
    release_date:
      type: string
      format: date
      nullable: true
```

```
PatchedSong:
  type: object
  properties:
    audio_file:
```

```
    type: string
    format: uri
  cover_image:
    type: string
    format: uri
  title:
    type: string
    maxLength: 100
  release_date:
    type: string
    format: date
    nullable: true
  genre:
    $ref: '#/components/schemas/GenreEnum'

security:
  - JWTAuth: []

tags:
  - name: Authentication
    description: User authentication and authorization
  - name: User
    description: User profile management
  - name: Artists
    description: Artist management operations
  - name: Albums
    description: Album management operations
  - name: Songs
    description: Song management operations
```

6. Projekto išvados

Sukurto projekto tiklas: sukurti muzikos valdymo sistemą su skirtingais naudotojų vaidmenimis. Nors projektas nesuteikė daug naujų žinių, leido praktiškai pritaikyti jau turimas žinias apie sistemų architektūrą, API kūrimą bei frontend ir backend technologijų naudojimą. Taip pat leido plačiau susipažinti su OpenAPI dokumentacija, kuri leidžia lengvai suprasti ką atlieka kiekvienas sistemos API punktas.