KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Saityno taikomųjų programų projektavimas (T120B165)

Playlist Player projekto ataskaita

Atliko:

IFF-1/5 gr. studentas

Jurgis Andziulis

Priėmė:

Prof. Blažauskas Tomas

Lekt. prakt. Baltulionis Simonas

KAUNAS 2024

TURINYS

1. **Užduoties aprašymas 3**
2. **Sistemos paskirtis 3**
3. **Funkciniai reikalavimai 3**
4. **Sistemos architektūra 3**
5. **Vartotojo sąsajos dizainas 4**
6. **API specifikacija (OpenAPI) 9**
7. **Projekto išvados 25**

1. Užduoties aprašymas

„PlaylistPlayer“ yra muzikos kategorijų, grojaraščių ir dainų tvarkymo programa. Joje užtikrinama vaidmenimis pagrįsta prieigos kontrolė, leidžianti svečiams peržiūrėti turinį, nariams tvarkyti grojaraščius, o administratoriams - valdyti visas kategorijas, grojaraščius ir dainas.

Šiuolaikinėse muzikos paslaugose įprasta tvarkyti skirtingų kategorijų grojaraščius. Vartotojams reikia struktūrizuotos sąsajos, kad jie galėtų tvarkyti savo muzikos kolekcijas ir rasti ar keisti grojaraščius bei dainas.

2. Sistemos paskirtis

Naudotojai gali kurti ir tvarkyti kategorijas bei dainų grojaraščius.

Skirtingi naudotojų vaidmenys turi skirtingas teises: svečiai (peržiūrėti), nariai (tvarkyti savo), administratoriai (visiška prieiga).

Sistemoje įdiegtas JWT pagrįstas autentiškumo patvirtinimas ir integruota REST API su priekine vartotojo sąsaja.

3. Funkciniai reikalavimai

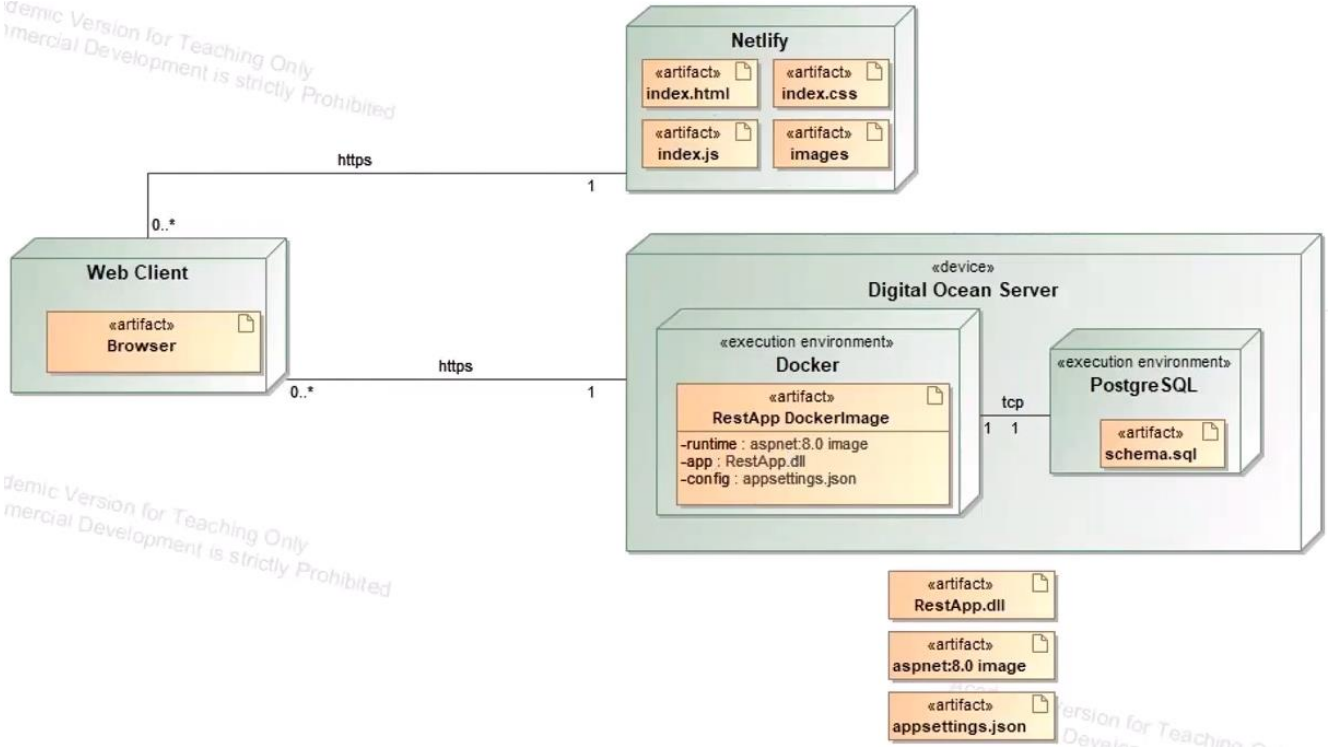
CRUD kategorijoms, grojaraščiams ir dainoms.
Prieiga pagal vaidmenis: Svečias, narys, administratorius.
Autentiškumo nustatymas pagal žetoną (JWT).
Galimybė peržiūrėti hierarchinius duomenis (grojaraščius kategorijoje, dainas grojaraštyje).

4. Sistemos architektūra

Sistemos architektūra susideda iš kelių pagrindinių komponentų:

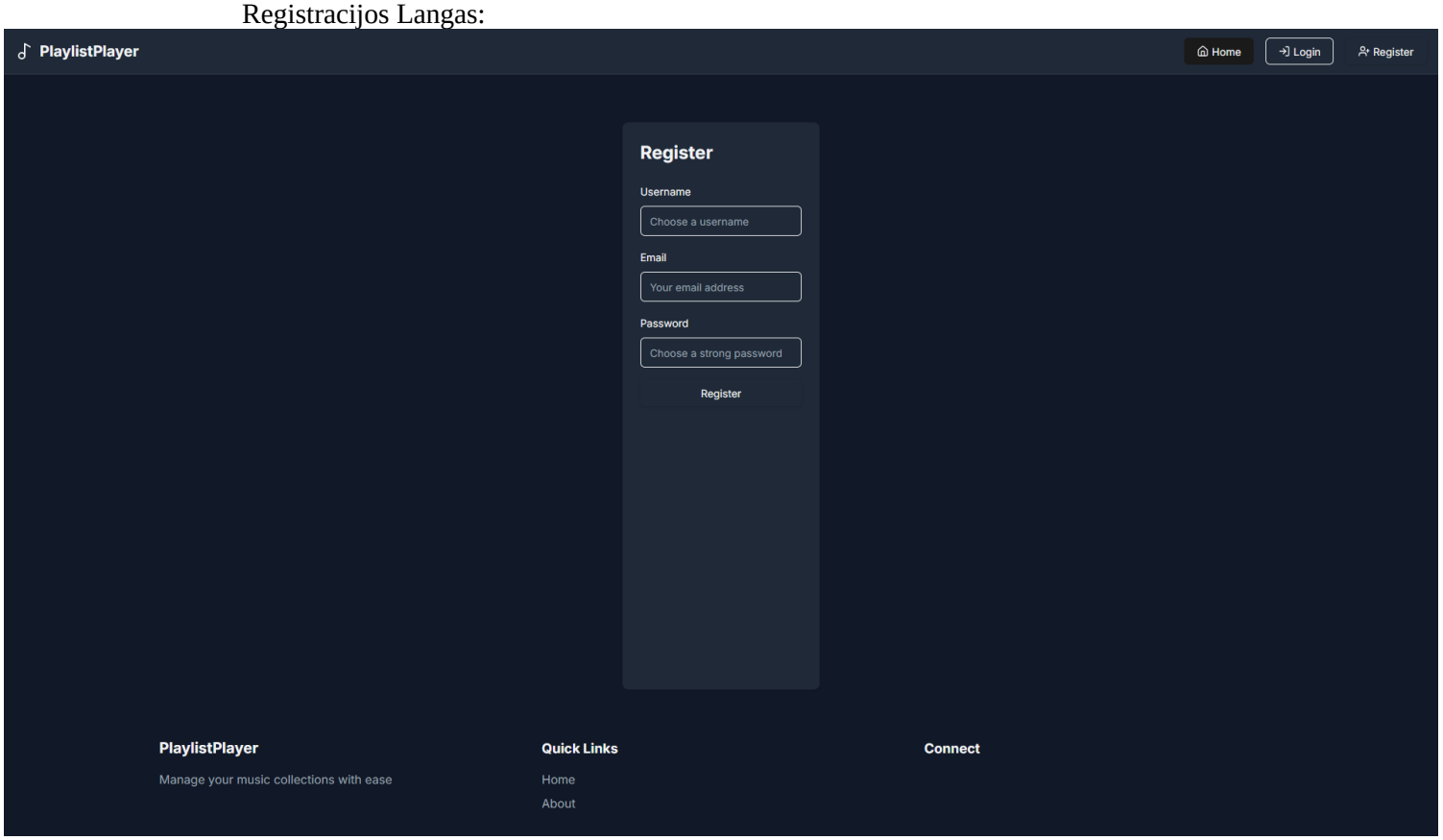
- Frontend: Sukurtas naudojant React ir Tailwind CSS, atsakingas už vartotojo sąsajos pateikimą ir sąveiką su vartotoju.
- Backend: ASP.NET Core Web API, teikiantis REST API endpointus ir valdantis verslo logiką bei duomenų apdorojimą.
- Duomenų Bazė: SQL Server, saugantis visus duomenis apie kategorijas, grojaraščius ir dainas.
- Autentifikacija: JWT pagrindu veikianti autentifikacija, užtikrinanti saugų vartotojų prisijungimą ir autorizaciją.
- Hostingas: DigitalOcean, kuriame yra talpinama visa sistema.

Diagrama:

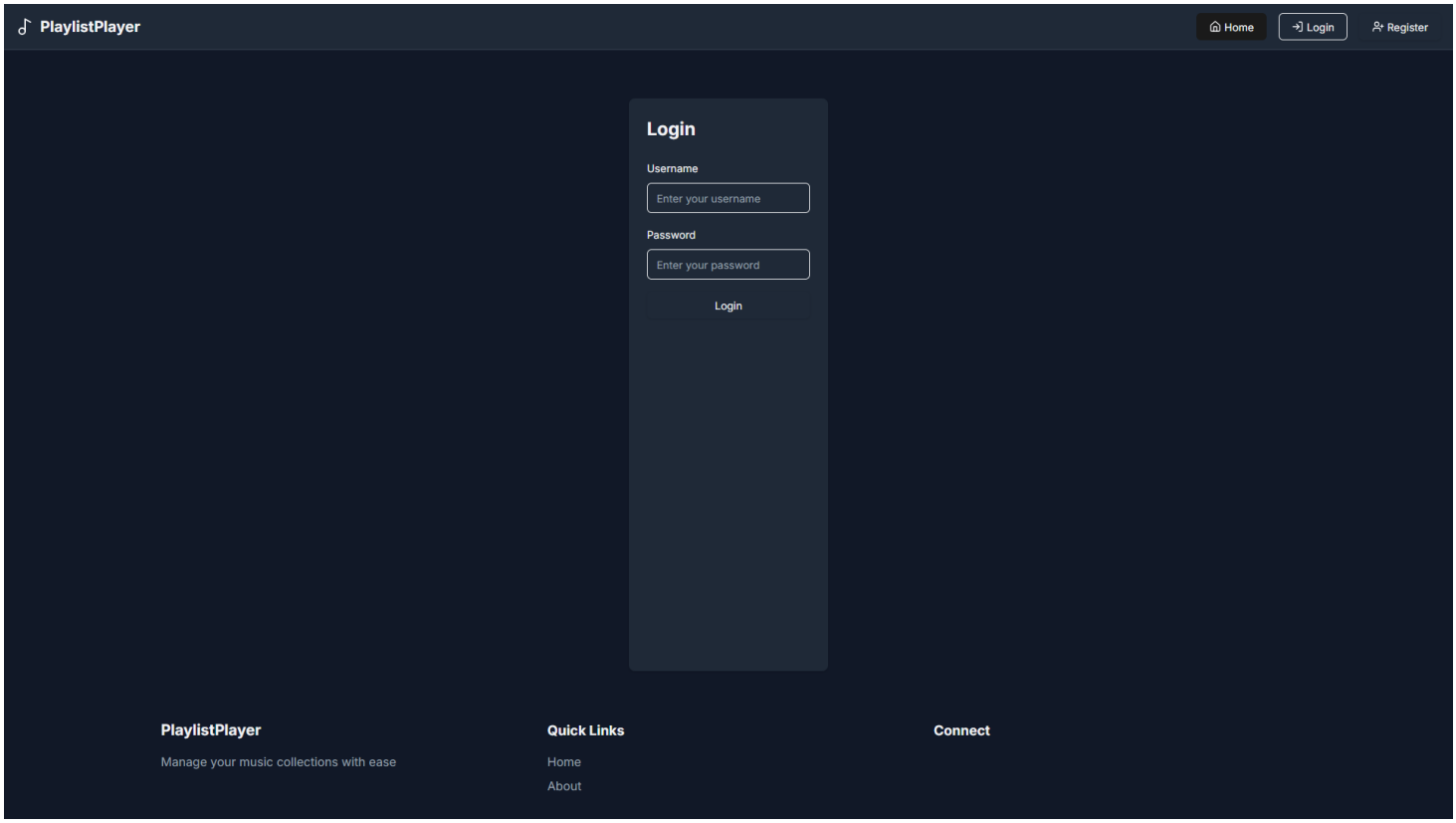


5. Vartotojo sąsajos dizainas

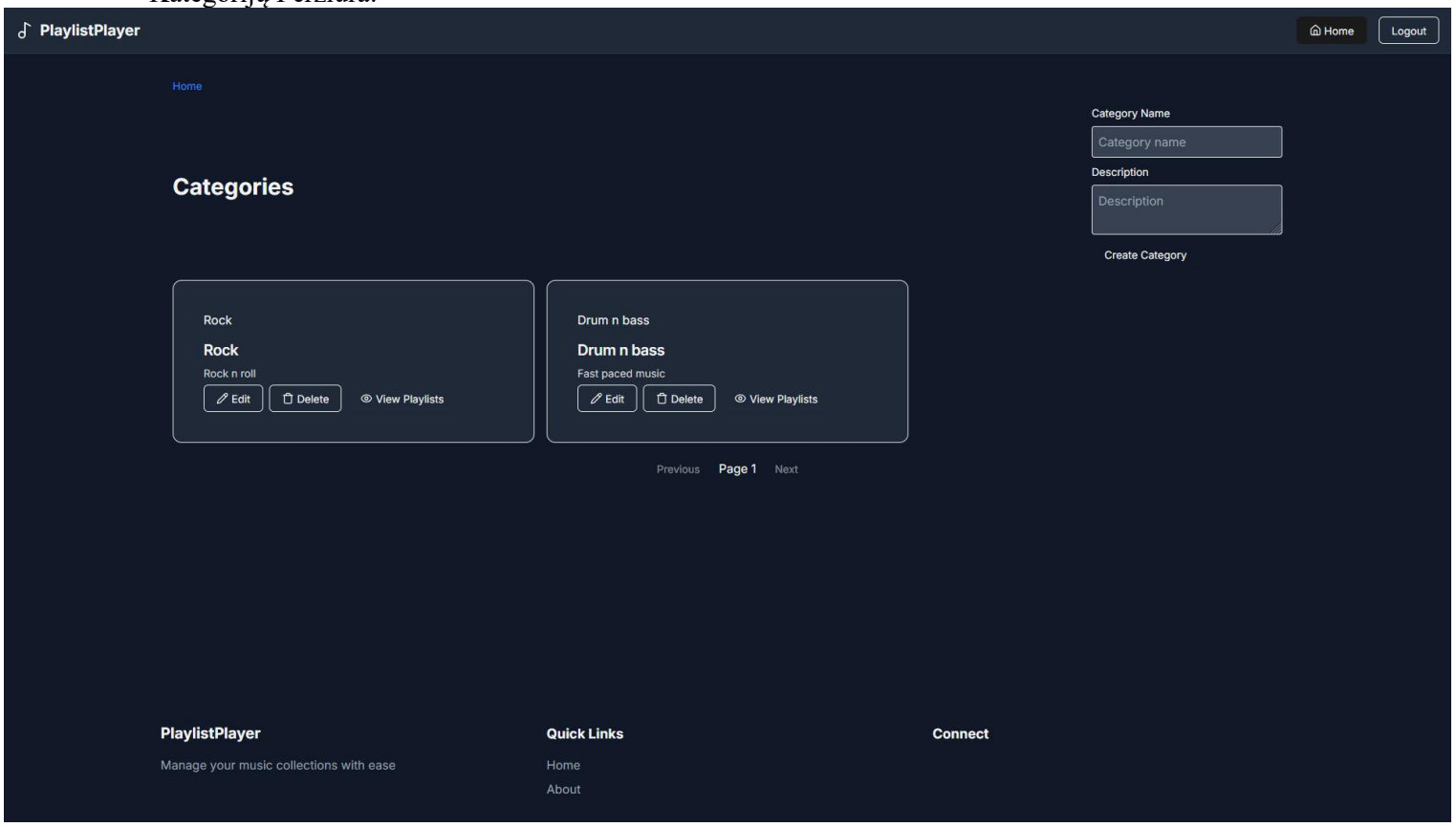
Registracijos Langas:



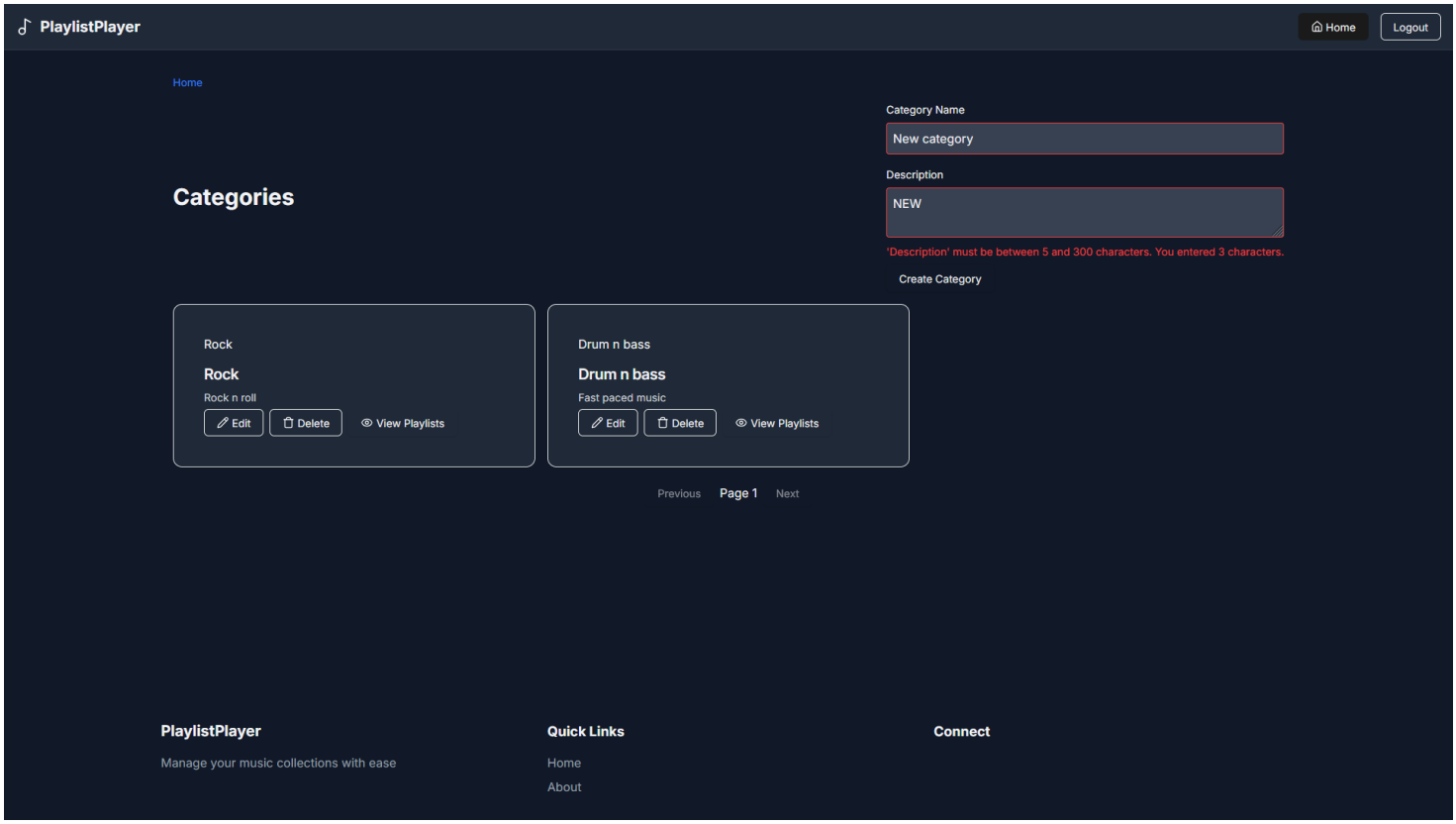
Prisijungimo Langas:



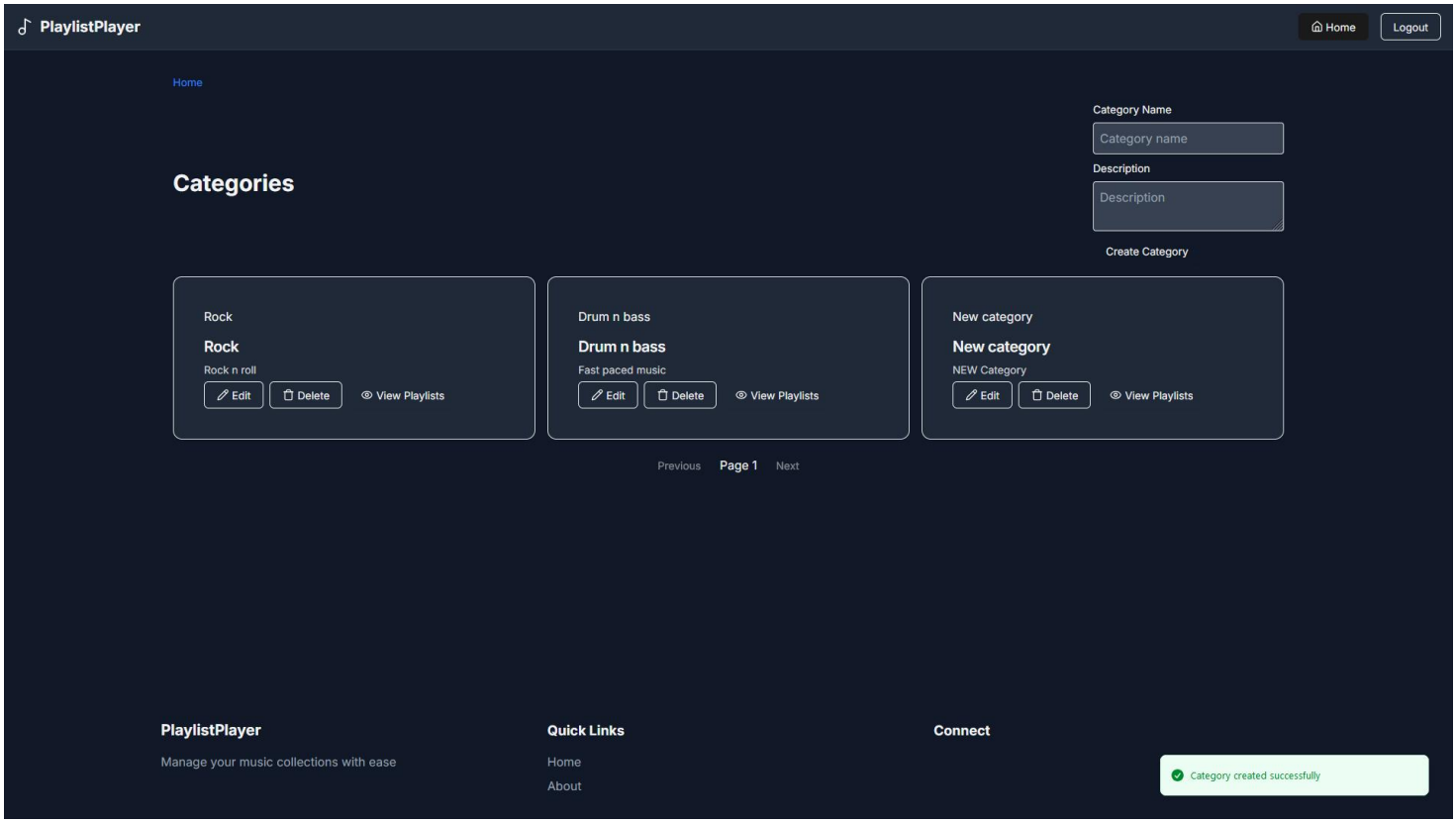
Kategorijų Peržiūra:



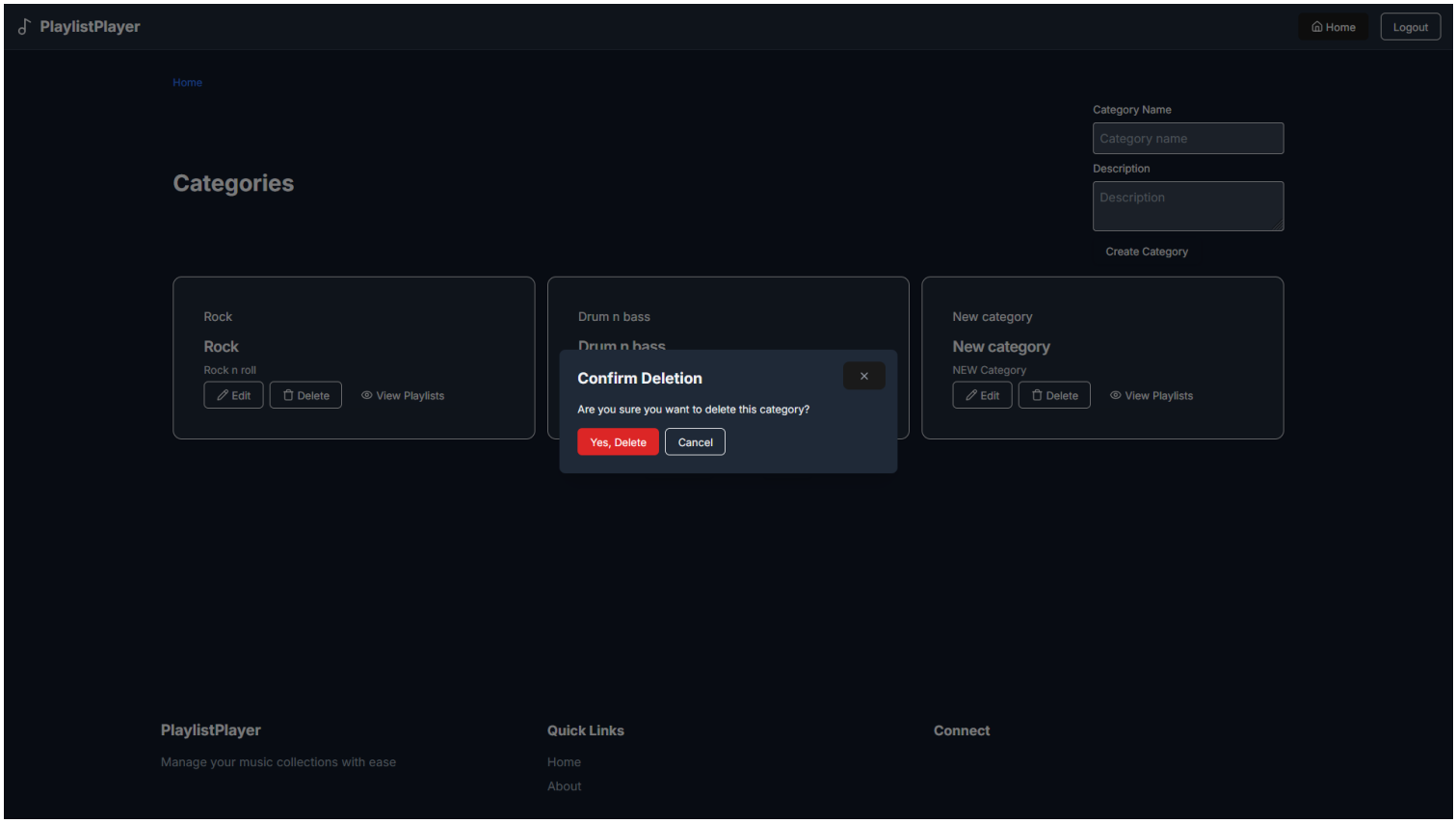
Validacija:



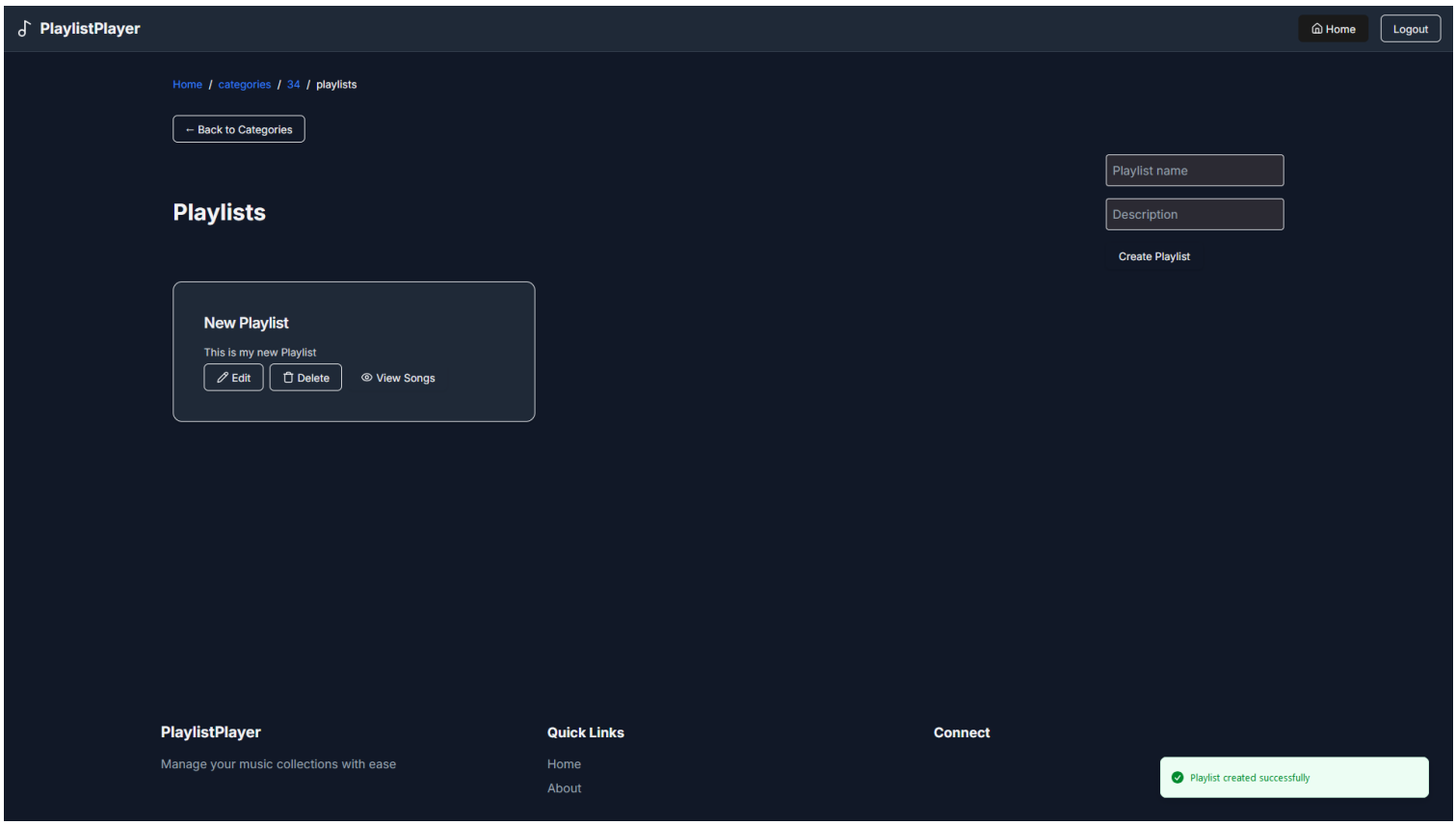
Sukurta kategorija:



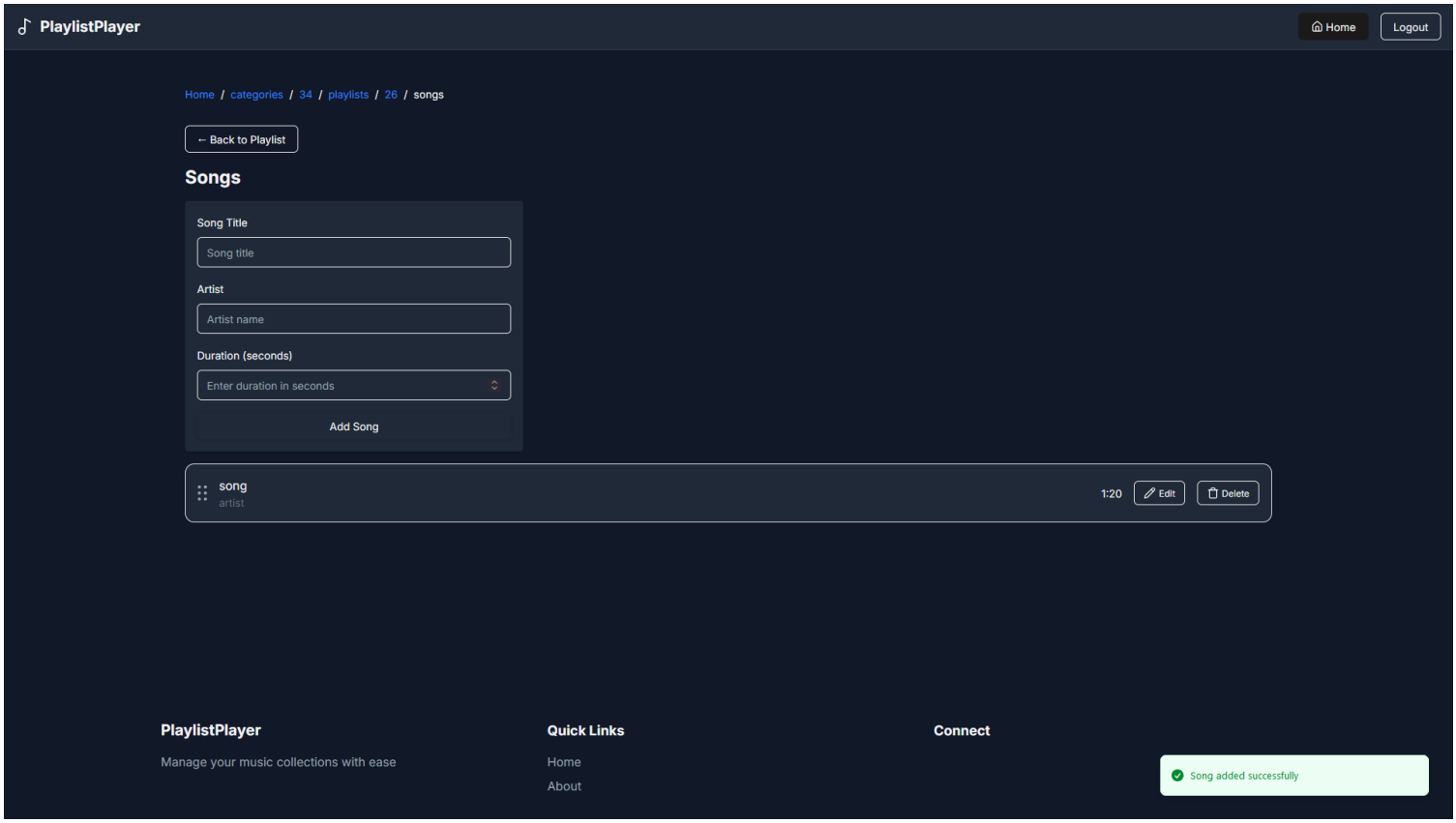
Ištrinama kategorija:



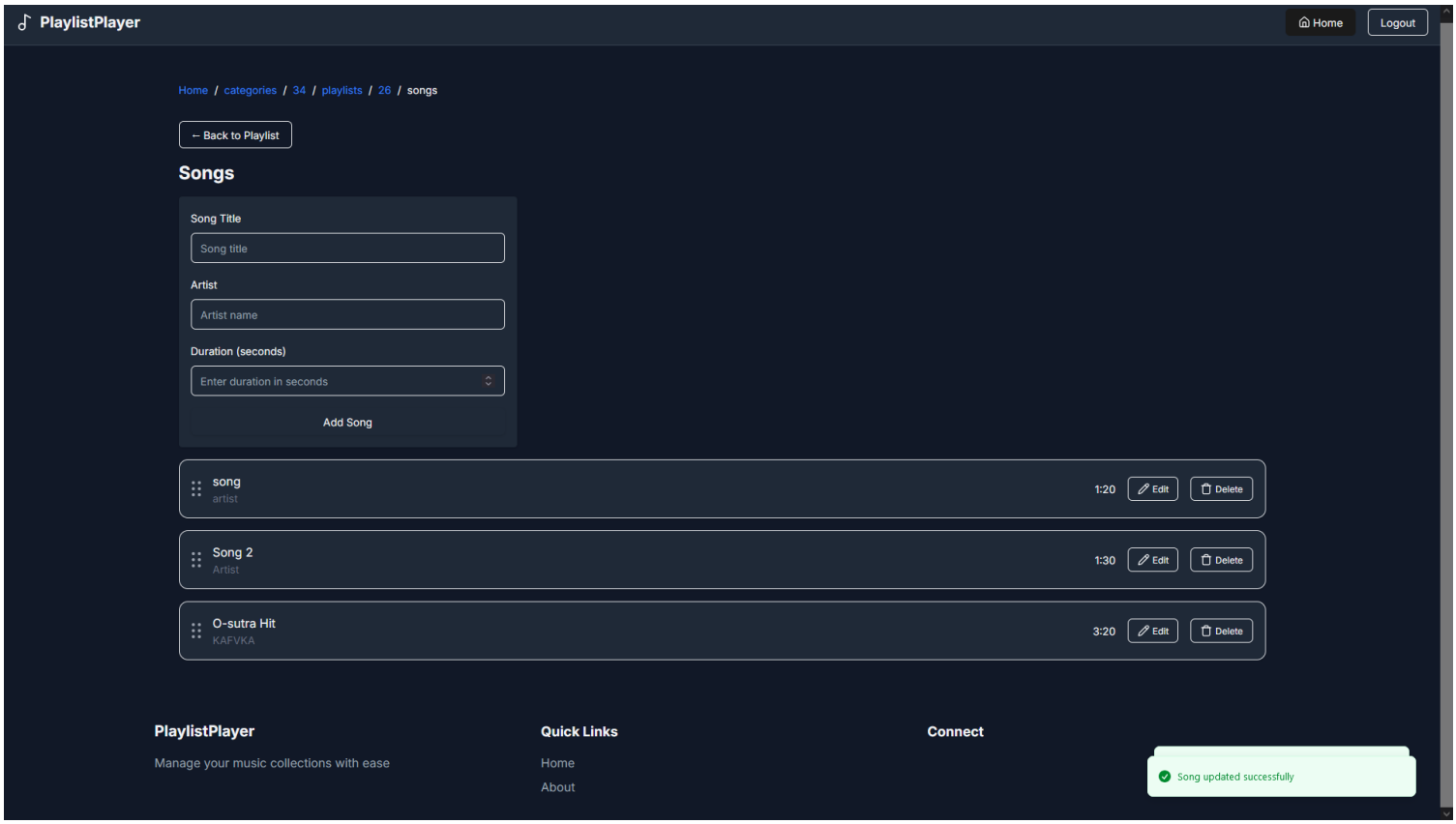
Grojaraščių Tvarkymas:



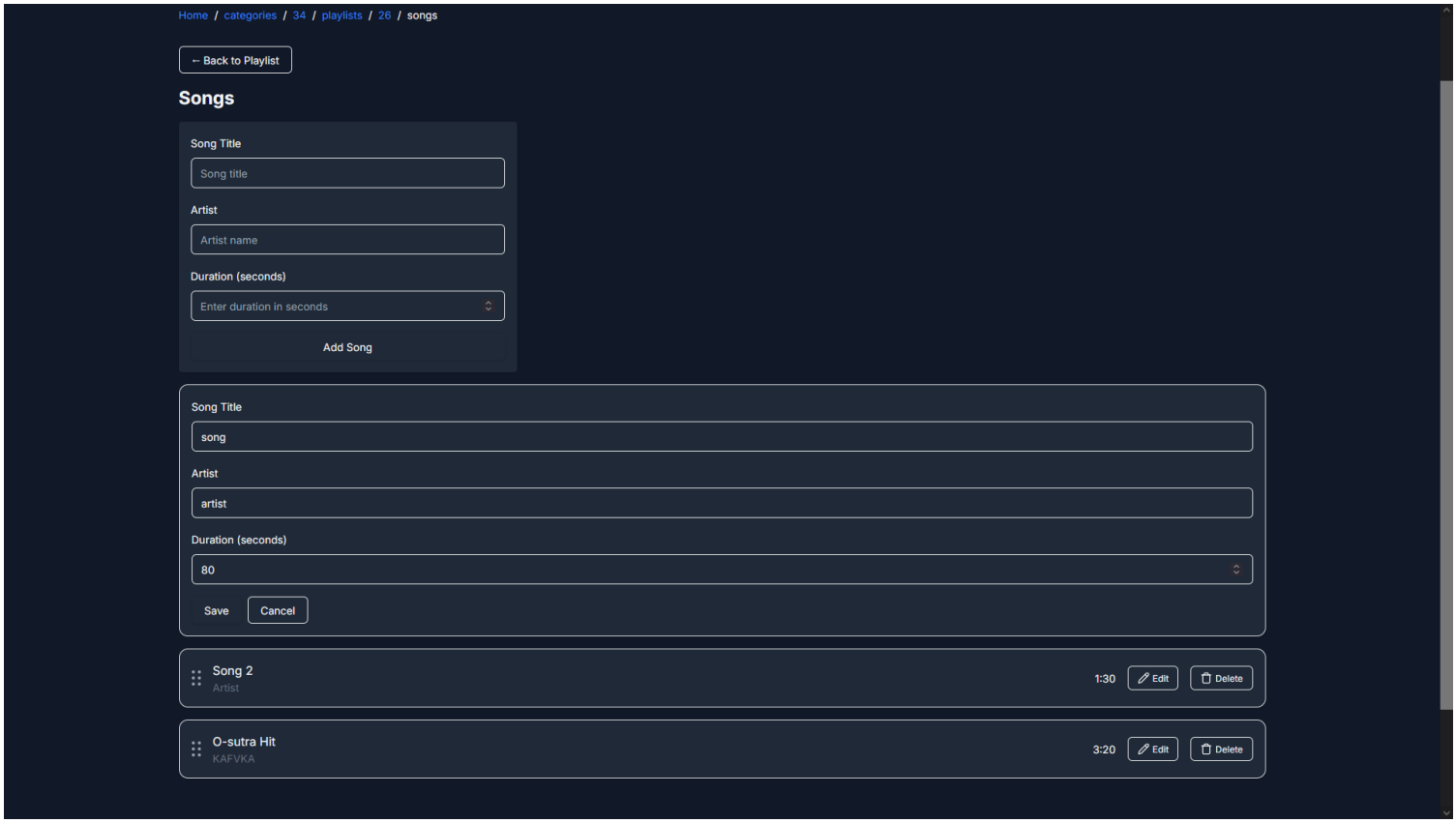
Dainų Tvarkymas:



Galima daina perrūšiuot:



Galima redaguoti:



6. API specifikacija (OpenAPI)

API aprašyta naudojant „OpenAPI 3.0.0“ specifikaciją. Visą specifikaciją galima rasti atskirame api-spec.yaml faile.

https://editor.swagger.io

openapi.yaml:

```
openapi: 3.0.0
info:
  title: Music Playlist Manager API
  description: An API for managing music categories, playlists, and songs
  version: 1.0.0

servers:
  - url: https://octopus-app-3t93j.ondigitalocean.app/api

paths:
  /categories:
    get:
      summary: Get a list of categories
      tags:
        - category
      operationId: getAllCategories
      parameters:
        - $ref: "#/components/parameters/pageNumber"
        - $ref: "#/components/parameters/pageSize"
      responses:
        "200":
```

```
          description: A list of categories
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/pagedResourceResponse"
              example:
                resources:
                  - resource:
                      id: 1
                      name: "Rock"
                      description: "Rock music collection"
                      createdOn: "2024-03-15T14:30:00Z"
                    links:
                      - href: "/categories/1"
                        rel: "self"
                        method: "GET"
                      - href: "/categories/1/playlists"
                        rel: "playlists"
                        method: "GET"
                  - resource:
                      id: 2
                      name: "Jazz"
                      description: "Jazz music collection"
                      createdOn: "2024-03-15T14:35:00Z"
                    links:
                      - href: "/categories/2"
                        rel: "self"
                        method: "GET"
                      - href: "/categories/2/playlists"
                        rel: "playlists"
                        method: "GET"
                links:
                  - href: "/categories?pageNumber=2&pageSize=10"
                    rel: "next"
                    method: "GET"
                  - href: "/categories?pageNumber=1&pageSize=10"
                    rel: "self"
                    method: "GET"
          headers:
            Pagination:
              $ref: "#/components/headers/Pagination"
              example: "Total-Count: 25, Page-Count: 3, Page-Size: 10, Page-
Number: 1"

    post:
      description: "Create a category"
      tags:
        - category
      operationId: createCategory
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/createCategory"
            example:
              name: "Classical"
              description: "Classical music collection featuring orchestral and
chamber music"
      responses:
        "201":
          $ref: "#/components/responses/singleCategory"
          content:
            application/json:
              example:
                resource:
                  id: 3
```

```
                  name: "Classical"
                  description: "Classical music collection featuring orchestral
and chamber music"
                  createdOn: "2024-03-15T15:00:00Z"
                links:
                  - href: "/categories/3"
                    rel: "self"
                    method: "GET"
                  - href: "/categories/3/playlists"
                    rel: "playlists"
                    method: "GET"
        "400":
          $ref: "#/components/responses/badRequest"
        "422":
          $ref: "#/components/responses/unprocessableEntity"

  /categories/{categoryId}:
    parameters:
      - $ref: "#/components/parameters/categoryId"
    get:
      summary: Get a category by ID
      tags:
        - category
      operationId: getCategory
      responses:
        "200":
          $ref: "#/components/responses/singleCategory"
          content:
            application/json:
              example:
                resource:
                  id: 1
                  name: "Rock"
                  description: "Rock music collection"
                  createdOn: "2024-03-15T14:30:00Z"
                links:
                  - href: "/categories/1"
                    rel: "self"
                    method: "GET"
                  - href: "/categories/1/playlists"
                    rel: "playlists"
                    method: "GET"
        "404":
          $ref: "#/components/responses/notFound"

    put:
      summary: Update a category by ID
      tags:
        - category
      operationId: updateCategory
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/updateCategory"
            example:
              description: "Updated rock music collection featuring classic and
modern rock"
      responses:
        "200":
          $ref: "#/components/responses/singleCategory"
          content:
            application/json:
              example:
                resource:
                  id: 1
```

```
                  name: "Rock"
                  description: "Updated rock music collection featuring classic
and modern rock"
                  createdOn: "2024-03-15T14:30:00Z"
                links:
                  - href: "/categories/1"
                    rel: "self"
                    method: "GET"
                  - href: "/categories/1/playlists"
                    rel: "playlists"
                    method: "GET"
        "400":
          $ref: "#/components/responses/badRequest"
        "404":
          $ref: "#/components/responses/notFound"
        "422":
          $ref: "#/components/responses/unprocessableEntity"

    delete:
      summary: Delete a category by ID
      tags:
        - category
      operationId: deleteCategory
      responses:
        "204":
          description: Category deleted
        "404":
          $ref: "#/components/responses/notFound"

  /categories/{categoryId}/playlists:
    parameters:
      - $ref: "#/components/parameters/categoryId"
    get:
      summary: Get playlists for a category
      tags:
        - playlist
      operationId: getPlaylists
      parameters:
        - $ref: "#/components/parameters/pageNumber"
        - $ref: "#/components/parameters/pageSize"
      responses:
        "200":
          description: List of playlists
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/pagedResourceResponse"
              example:
                resources:
                  - resource:
                      id: 1
                      name: "Classic Rock Hits"
                      description: "Best rock songs from the 70s and 80s"
                      createdOn: "2024-03-15T16:00:00Z"
                      categoryId: 1
                    links:
                      - href: "/categories/1/playlists/1"
                        rel: "self"
                        method: "GET"
                      - href: "/categories/1/playlists/1/songs"
                        rel: "songs"
                        method: "GET"
                links:
                  - href: "/categories/1/playlists?pageNumber=1&pageSize=10"
                    rel: "self"
                    method: "GET"
```

```
        headers:
          Pagination:
            $ref: "#/components/headers/Pagination"

  post:
    description: "Add a playlist"
    tags:
      - playlist
    operationId: createPlaylist
    requestBody:
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/createPlaylist"
          example:
            name: "Progressive Rock Classics"
            description: "A collection of the best progressive rock songs"
    responses:
      "201":
        $ref: "#/components/responses/singlePlaylist"
        content:
          application/json:
            example:
              resource:
                id: 2
                name: "Progressive Rock Classics"
                description: "A collection of the best progressive rock songs"
                createdOn: "2024-03-15T16:30:00Z"
                categoryId: 1
              links:
                - href: "/categories/1/playlists/2"
                  rel: "self"
                  method: "GET"
                - href: "/categories/1/playlists/2/songs"
                  rel: "songs"
                  method: "GET"
      "400":
        $ref: "#/components/responses/badRequest"
      "422":
        $ref: "#/components/responses/unprocessableEntity"

/categories/{categoryId}/playlists/{playlistId}:
  parameters:
    - $ref: "#/components/parameters/categoryId"
    - $ref: "#/components/parameters/playlistId"
  get:
    summary: Get a specific playlist for a category
    tags:
      - playlist
    operationId: getPlaylist
    responses:
      "200":
        $ref: "#/components/responses/singlePlaylist"
        content:
          application/json:
            example:
              resource:
                id: 1
                name: "Classic Rock Hits"
                description: "Best rock songs from the 70s and 80s"
                createdOn: "2024-03-15T16:00:00Z"
                categoryId: 1
              links:
                - href: "/categories/1/playlists/1"
                  rel: "self"
                  method: "GET"
```

```
                - href: "/categories/1/playlists/1/songs"
                  rel: "songs"
                  method: "GET"
      "404":
        $ref: "#/components/responses/notFound"

  put:
    summary: Update a playlist for a category
    tags:
      - playlist
    operationId: updatePlaylist
    requestBody:
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/updatePlaylist"
          example:
            name: "Classic Rock Greatest Hits"
            description: "Updated collection of the best classic rock songs"
    responses:
      "200":
        $ref: "#/components/responses/singlePlaylist"
        content:
          application/json:
            example:
              resource:
                id: 1
                name: "Classic Rock Greatest Hits"
                description: "Updated collection of the best classic rock songs"
                createdOn: "2024-03-15T16:00:00Z"
                categoryId: 1
              links:
                - href: "/categories/1/playlists/1"
                  rel: "self"
                  method: "GET"
                - href: "/categories/1/playlists/1/songs"
                  rel: "songs"
                  method: "GET"
      "400":
        $ref: "#/components/responses/badRequest"
      "404":
        $ref: "#/components/responses/notFound"
      "422":
        $ref: "#/components/responses/unprocessableEntity"

  delete:
    summary: Delete a playlist for a category
    tags:
      - playlist
    operationId: deletePlaylist
    responses:
      "204":
        description: Playlist deleted
      "404":
        $ref: "#/components/responses/notFound"

/categories/{categoryId}/playlists/{playlistId}/songs:
  parameters:
    - $ref: "#/components/parameters/categoryId"
    - $ref: "#/components/parameters/playlistId"
  get:
    summary: Get songs for a specific playlist
    tags:
      - song
    operationId: getSongs
    parameters:
```

```
        - $ref: "#/components/parameters/pageNumber"
        - $ref: "#/components/parameters/pageSize"
      responses:
        "200":
          description: List of songs
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/pagedResourceResponse"
              example:
                resources:
                  - resource:
                      id: 1
                      title: "Stairway to Heaven"
                      artist: "Led Zeppelin"
                      duration: 482
                      orderId: 1
                      createdOn: "2024-03-15T17:00:00Z"
                      playlistId: 1
                    links:
                      - href: "/categories/1/playlists/1/songs/1"
                        rel: "self"
                        method: "GET"
                  - resource:
                      id: 2
                      title: "Bohemian Rhapsody"
                      artist: "Queen"
                      duration: 354
                      orderId: 2
                      createdOn: "2024-03-15T17:01:00Z"
                      playlistId: 1
                    links:
                      - href: "/categories/1/playlists/1/songs/2"
                        rel: "self"
                        method: "GET"
                links:
                  - href:
"/categories/1/playlists/1/songs?pageNumber=1&pageSize=10"
                    rel: "self"
                    method: "GET"
          headers:
            Pagination:
              $ref: "#/components/headers/Pagination"

    post:
      description: "Add a song to a specific playlist"
      tags:
        - song
      operationId: createSong
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/createSong"
            example:
              title: "Hotel California"
              artist: "Eagles"
              duration: 391
      responses:
        "201":
          $ref: "#/components/responses/singleSong"
          content:
            application/json:
              example:
                resource:
                  id: 3
```

```
                title: "Hotel California"
                artist: "Eagles"
                duration: 391
                orderId: 3
                createdOn: "2024-03-15T17:02:00Z"
                playlistId: 1
              links:
                - href: "/categories/1/playlists/1/songs/3"
                  rel: "self"
                  method: "GET"
      "400":
        $ref: "#/components/responses/badRequest"
      "422":
        $ref: "#/components/responses/unprocessableEntity"

/categories/{categoryId}/playlists/{playlistId}/songs/{songId}:
  parameters:
    - $ref: "#/components/parameters/categoryId"
    - $ref: "#/components/parameters/playlistId"
    - $ref: "#/components/parameters/songId"
  get:
    summary: Get a specific song
    tags:
      - song
    operationId: getSong
    responses:
      "200":
        $ref: "#/components/responses/singleSong"
        content:
          application/json:
            example:
              resource:
                id: 1
                title: "Stairway to Heaven"
                artist: "Led Zeppelin"
                duration: 482
                orderId: 1
                createdOn: "2024-03-15T17:00:00Z"
                playlistId: 1
              links:
                - href: "/categories/1/playlists/1/songs/1"
                  rel: "self"
                  method: "GET"
      "404":
        $ref: "#/components/responses/notFound"

  put:
    summary: Update a song
    tags:
      - song
    operationId: updateSong
    requestBody:
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/updateSong"
          example:
            title: "Stairway to Heaven (Remastered)"
            artist: "Led Zeppelin"
            duration: 482
            orderId: 1
    responses:
      "200":
        $ref: "#/components/responses/singleSong"
        content:
          application/json:
```

```
            example:
              resource:
                id: 1
                title: "Stairway to Heaven (Remastered)"
                artist: "Led Zeppelin"
                duration: 482
                orderId: 1
                createdOn: "2024-03-15T17:00:00Z"
                playlistId: 1
              links:
                - href: "/categories/1/playlists/1/songs/1"
                  rel: "self"
                  method: "GET"
      "400":
        $ref: "#/components/responses/badRequest"
      "404":
        $ref: "#/components/responses/notFound"
      "422":
        $ref: "#/components/responses/unprocessableEntity"

  delete:
    summary: Delete a song
    tags:
      - song
    operationId: deleteSong
    responses:
      "204":
        description: Song deleted
      "404":
        $ref: "#/components/responses/notFound"

/accounts:
  post:
    summary: Register a new user account
    tags:
      - auth
    operationId: registerUser
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/registerUserDto"
    responses:
      "201":
        description: User successfully created
      "422":
        description: Username already taken or validation failed
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/problemDetails"
            example:
              type: "https://api.example.com/errors/validation"
              title: "Validation Error"
              status: 422
              detail: "Username already taken"
              instance: "/api/accounts"

/login:
  post:
    summary: Login to get access token
    tags:
      - auth
    operationId: login
    requestBody:
```

```
      required: true
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/loginDto"
    responses:
      "200":
        description: Successfully logged in
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/successfulLoginDto"
        headers:
          Set-Cookie:
            schema:
              type: string
              example: "RefreshToken=xyz...; HttpOnly; SameSite=Lax"
      "422":
        description: Invalid credentials
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/problemDetails"
            example:
              type: "https://api.example.com/errors/validation"
              title: "Authentication Error"
              status: 422
              detail: "Invalid username or password"
              instance: "/api/login"

/accessToken:
  post:
    summary: Refresh access token using refresh token
    tags:
      - auth
    operationId: refreshToken
    responses:
      "200":
        description: New access token generated
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/successfulLoginDto"
        headers:
          Set-Cookie:
            schema:
              type: string
              example: "RefreshToken=xyz...; HttpOnly; SameSite=Lax"
      "422":
        description: Invalid or expired refresh token
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/problemDetails"

/logout:
  post:
    summary: Logout and invalidate refresh token
    tags:
      - auth
    operationId: logout
    responses:
      "200":
        description: Successfully logged out
      "422":
        description: Invalid refresh token
```

```
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/problemDetails"

components:
  schemas:
    category:
      type: object
      properties:
        id:
          type: integer
          example: 1
        name:
          type: string
          example: "Rock"
        description:
          type: string
          example: "Rock music collection"
        createdOn:
          type: string
          format: date-time
          example: "2024-03-15T14:30:00Z"

    createCategory:
      type: object
      required:
        - name
        - description
      properties:
        name:
          type: string
          minLength: 2
          maxLength: 100
          example: "Classical"
        description:
          type: string
          minLength: 5
          maxLength: 300
          example: "Classical music collection featuring orchestral and chamber
music"

    updateCategory:
      type: object
      required:
        - description
      properties:
        description:
          type: string
          minLength: 5
          maxLength: 300
          example: "Updated classical music collection"

    playlist:
      type: object
      properties:
        id:
          type: integer
          example: 1
        name:
          type: string
          example: "Classic Rock Hits"
        description:
          type: string
          example: "Best rock songs from the 70s and 80s"
        createdOn:
```

```
      type: string
      format: date-time
      example: "2024-03-15T16:00:00Z"
    categoryId:
      type: integer
      example: 1

createPlaylist:
  type: object
  required:
    - name
    - description
  properties:
    name:
      type: string
      minLength: 2
      maxLength: 100
      example: "Progressive Rock Classics"
    description:
      type: string
      minLength: 5
      maxLength: 500
      example: "A collection of the best progressive rock songs"

updatePlaylist:
  type: object
  required:
    - name
    - description
  properties:
    name:
      type: string
      minLength: 2
      maxLength: 100
      example: "Classic Rock Greatest Hits"
    description:
      type: string
      minLength: 5
      maxLength: 500
      example: "Updated collection of the best classic rock songs"

song:
  type: object
  properties:
    id:
      type: integer
      example: 1
    title:
      type: string
      example: "Stairway to Heaven"
    artist:
      type: string
      example: "Led Zeppelin"
    duration:
      type: integer
      description: Duration in seconds
      example: 482
    orderId:
      type: integer
      description: The order of the song within the playlist
      example: 1
    createdOn:
      type: string
      format: date-time
      example: "2024-03-15T17:00:00Z"
    playlistId:
```

```
      type: integer
      example: 1

createSong:
  type: object
  required:
    - title
    - artist
    - duration
  properties:
    title:
      type: string
      minLength: 1
      maxLength: 200
      example: "Hotel California"
    artist:
      type: string
      minLength: 1
      maxLength: 100
      example: "Eagles"
    duration:
      type: integer
      minimum: 1
      description: Duration in seconds
      example: 391

updateSong:
  type: object
  required:
    - title
    - artist
    - duration
    - orderId
  properties:
    title:
      type: string
      minLength: 1
      maxLength: 200
      example: "Stairway to Heaven (Remastered)"
    artist:
      type: string
      minLength: 1
      maxLength: 100
      example: "Led Zeppelin"
    duration:
      type: integer
      minimum: 1
      description: Duration in seconds
      example: 482
    orderId:
      type: integer
      minimum: 1
      description: The new order of the song within the playlist
      example: 1

link:
  type: object
  properties:
    href:
      type: string
      example: "/categories/1"
    rel:
      type: string
      example: "self"
    method:
      type: string
```

```
      example: "GET"

resourceWithLinks:
  type: object
  properties:
    resource:
      type: object
    links:
      type: array
      items:
        $ref: "#/components/schemas/link"

pagedResourceResponse:
  type: object
  properties:
    resources:
      type: array
      items:
        $ref: "#/components/schemas/resourceWithLinks"
    links:
      type: array
      items:
        $ref: "#/components/schemas/link"

problemDetails:
  type: object
  properties:
    type:
      type: string
      format: uri
      example: "https://api.example.com/errors/validation"
    title:
      type: string
      example: "Validation Error"
    status:
      type: integer
      example: 400
    detail:
      type: string
      example: "The request contains invalid parameters"
    instance:
      type: string
      format: uri
      example: "/categories/1/playlists"
    errors:
      type: object
      additionalProperties:
        type: array
        items:
          type: string
      example:
        name:
          ["Name is required", "Name must be between 2 and 100 characters"]
        description: ["Description must be between 5 and 500 characters"]

registerUserDto:
  type: object
  required:
    - userName
    - email
    - password
  properties:
    userName:
      type: string
      example: "john.doe"
    email:
```

```
        type: string
        format: email
        example: "john.doe@example.com"
      password:
        type: string
        format: password
        example: "MySecurePassword123"

  loginDto:
    type: object
    required:
      - userName
      - password
    properties:
      userName:
        type: string
        example: "john.doe"
      password:
        type: string
        format: password
        example: "MySecurePassword123"

  successfulLoginDto:
    type: object
    required:
      - accessToken
    properties:
      accessToken:
        type: string
        example: "eyJhbGciOiJIUzI1NiIs..."

responses:
  singleCategory:
    description: A single category
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/resourceWithLinks"

  singlePlaylist:
    description: A single playlist
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/resourceWithLinks"

  singleSong:
    description: A single song
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/resourceWithLinks"

  badRequest:
    description: Bad request response
    content:
      application/problem+json:
        schema:
          $ref: "#/components/schemas/problemDetails"
        example:
          type: "https://api.example.com/errors/validation"
          title: "Validation Error"
          status: 400
          detail: "The request contains invalid parameters"
          instance: "/categories/1/playlists"
          errors:
```

```
              name:
                [
                  "Name is required",
                  "Name must be between 2 and 100 characters",
                ]
              description: ["Description must be between 5 and 500 characters"]

    unprocessableEntity:
      description: Unprocessable entity response
      content:
        application/problem+json:
          schema:
            $ref: "#/components/schemas/problemDetails"
          example:
            type: "https://api.example.com/errors/business-rule"
            title: "Business Rule Violation"
            status: 422
            detail: "The operation cannot be completed due to business rule
violations"
            instance: "/categories/1/playlists/1/songs"
            errors:
              playlist:
                ["Maximum number of songs (100) reached for this playlist"]

    notFound:
      description: Resource not found
      content:
        application/problem+json:
          schema:
            $ref: "#/components/schemas/problemDetails"
          example:
            type: "https://api.example.com/errors/not-found"
            title: "Resource Not Found"
            status: 404
            detail: "The requested resource was not found"
            instance: "/categories/999"

  parameters:
    categoryId:
      name: categoryId
      in: path
      required: true
      schema:
        type: integer
        minimum: 1
      description: The unique identifier of the category
      example: 1

    playlistId:
      name: playlistId
      in: path
      required: true
      schema:
        type: integer
        minimum: 1
      description: The unique identifier of the playlist
      example: 1

    songId:
      name: songId
      in: path
      required: true
      schema:
        type: integer
        minimum: 1
      description: The unique identifier of the song
```

```
      example: 1

    pageNumber:
      name: pageNumber
      in: query
      required: false
      schema:
        type: integer
        minimum: 1
        default: 1
      description: The page number for pagination
      example: 1

    pageSize:
      name: pageSize
      in: query
      required: false
      schema:
        type: integer
        minimum: 1
        maximum: 50
        default: 10
      description: The number of items per page for pagination
      example: 10

  headers:
    Pagination:
      description: Pagination metadata
      schema:
        type: string
      example: "Total-Count: 25, Page-Count: 3, Page-Size: 10, Page-Number: 1"

  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT

security:
  - bearerAuth: []

tags:
  - name: category
    description: Operations related to categories
  - name: playlist
    description: Operations related to playlists
  - name: song
    description: Operations related to songs
  - name: auth
    description: Authentication operations

externalDocs:
  description: Find more info here
  url: https://example.com/docs
```

7. Projekto išvados

Projekto metu buvo sėkmingai sukurta pilnavertė Playlist Player programa. Sistema leidžia vartotojams kurti ir tvarkyti muzikos kategorijas, grojaraščius bei dainas, taip pat užtikrina vaidmenimis pagrįstą prieigos kontrolę.

Playlist Player API atitinka pagrindinius muzikos kategorijų, grojaraščių ir dainų valdymo reikalavimus, užtikrinant patikimą vaidmenimis pagrįstą prieigos kontrolę. Pagrindiniai pasiekimai ir pastebėjimai:

- CRUD operacijos: API suteikia visas kategorijų, grojaraščių ir dainų kategorijų, grojaraščių ir dainų kūrimo, skaitymo, atnaujinimo ir šalinimo funkcijas, užtikrinančias, kad naudotojai galėtų veiksmingai valdyti savo muzikos kolekcijas.
- Į vaidmenis orientuota prieigos kontrolė: Įgyvendinant skirtingus vaidmenis (svečiai, nariai, administratoriai) padidinamas saugumas ir naudotojų patirtis, nes veiksmai apribojami pagal leidimus.
- JWT autentiškumo nustatymas: Saugus žetonais pagrįstas autentifikavimas užtikrina saugų naudotojų sesijų valdymą ir apsaugo jautrias operacijas nuo neteisėtos prieigos.
- Dokumentacija ir standartai: Taikant „OpenAPI“ specifikaciją, lengviau parengti aiškią ir standartizuotą API dokumentaciją, todėl palengvėja integracija ir priežiūra.

Pasiektos žinios ir įgūdžiai:

- CRUD operacijų įgyvendinimas: Išmokau efektyviai realizuoti CRUD (Create, Read, Update, Delete) operacijas su vaidmenimis pagrįsta prieigos kontrole, naudojant ASP.NET Core ir React.
- Front-end ir Back-end integracija: Įgyvendinau sklandžią front-end ir back-end bendradarbiavimą, užtikrinant duomenų srautą ir funkcionalumą.
- API struktūrizavimas: Supratau ir pritaikiau gerai struktūrizuotos API (OpenAPI specifikacijos) privalumus, siekiant aiškumo ir palaikomumo.
- Autentifikacija ir autorizacija: Įgyvendinau saugią JWT pagrindu veikiančią autentifikaciją, užtikrinant vartotojų duomenų apsaugą ir sistemos saugumą.
- Deploy’inau Back-end ant DigitalOcean ir Front-end ant Netlify.