Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1
по дисциплине
Сервисо-ориентированная архитектура

Выполнили Студенты группы P3412
Кобелев Р.П. Балин А.А.
к. т. н. Преподаватель:
Кривоносов Е.Д.

г. Санкт-Петербург
2025г.

Содержание

1 Задание **2**

2 Выполнение **3**

2.1 MusicBand Service 3

2.2 Grammy Service 15

3 Вывод **17**

1 Задание

Разработать спецификацию в формате OpenAPI для набора веб-сервисов, реализующего следующую функциональность:

Первый веб-сервис должен осуществлять управление коллекцией объектов. В коллекции необходимо хранить объекты класса ***MusicBand***, описание которого приведено ниже:

MusicBandService.java

```
1  public class MusicBand {
2     private Integer id; //Поле не может быть null, Значение поля должно быть больше
      ↪ 0, Значение этого поля должно быть уникальным, Значение этого поля должно
      ↪ генерироваться автоматически
3     private String name; //Поле не может быть null, Строка не может быть пустой
4     private Coordinates coordinates; //Поле не может быть null
5     private java.time.LocalDate creationDate; //Поле не может быть null, Значение
      ↪ этого поля должно генерироваться автоматически
6     private Integer numberOfParticipants; //Поле не может быть null, Значение поля
      ↪ должно быть больше 0
7     private Integer albumsCount; //Поле не может быть null, Значение поля должно
      ↪ быть больше 0
8     private String description; //Поле не может быть null
9     private MusicGenre genre; //Поле не может быть null
10    private Album bestAlbum; //Поле может быть null
11 }
12 public class Coordinates {
13    private float x; //Значение поля должно быть больше -975
14    private long y;
15 }
16 public class Album {
17    private String name; //Поле не может быть null, Строка не может быть пустой
18    private Long tracks; //Поле не может быть null, Значение поля должно быть
      ↪ больше 0
19 }
20 public enum MusicGenre {
21    PSYCHEDELIC_CLOUD_RAP,
22    SOUL,
23    POP;
24 }
```

Веб-сервис должен удовлетворять следующим требованиям:

- API, реализуемый сервисом, должен соответствовать рекомендациям подхода RESTful.
- Необходимо реализовать следующий базовый набор операций с объектами коллекции: добавление нового элемента, получение элемента по ИД, обновление элемента, удаление элемента, получение массива элементов.
- Операция, выполняемая над объектом коллекции, должна определяться методом HTTP-запроса.
- Операция получения массива элементов должна поддерживать возможность сортировки и фильтрации по любой комбинации полей класса, а также возможность постраничного вывода результатов выборки с указанием размера и порядкового номера выводимой страницы.
- Все параметры, необходимые для выполнения операции, должны передаваться в URL запроса.
- Информация об объектах коллекции должна передаваться в формате **xml**.
- В случае передачи сервису данных, нарушающих заданные на уровне класса ограничения целостности, сервис должен возвращать код ответа http, соответствующий произошедшей ошибке.

Помимо базового набора, веб-сервис должен поддерживать следующие операции над объектами коллекции:

- Удалить все объекты, значение поля description которого эквивалентно заданному.
- Удалить один (любой) объект, значение поля genre которого эквивалентно заданному.
- Вернуть количество объектов, значение поля bestAlbum которых больше заданного.

Эти операции должны размещаться на отдельных URL.

Второй веб-сервис должен располагаться на URL ***/grammy***, и реализовывать ряд дополнительных операций, связанных с вызовом API первого сервиса:

- ***/band/band-id/singles/add*** : добавить новый сингл указанной группе
- ***/band/band-id/participants/add*** : добавить в группу нового участника

Эти операции также должны размещаться на отдельных URL.

Для разработанной спецификации необходимо сгенерировать интерактивную веб-документацию с помощью Swagger UI. Документация должна содержать описание всех REST API обоих сервисов с текстовым описанием функциональности каждой операции. Созданную веб-документацию необходимо развернуть на сервере ***helios***.

2 Выполнение

Для выполнения была выбрана 3.1.1 версия OpenAPI спецификации.

2.1 MusicBand Service

music.yaml

```yaml
openapi: 3.1.1

info:
  title: Musicband Service Endpoints - OpenAPI 3.1.1

  description: |-
    **OpenAPI** specification for **Laboratory work №1** of **Service-oriented architecture** at **ITMO-University**.

  contact:
    name: Roman Kobelev & Balin Artem
    url: https://github.com/ta4ilka69/Service-oriented-architecture

  license:
    name: The MIT License
    url: https://mit-license.org/

  version: 0.0.1

servers:
  - url: 'https://localhost:5252'
    description: Helios

paths:
  /music-bands:
```

```yaml
    get:
      summary: Get list of musicbands.
      description: Get list of musicbands with possible filtering, sorting and pagination.

      parameters:
        - name: sort
          in: query
          description: Fields to sort by. Prefix with '-' for descending order. If field is used multiple times (both with and without prefix), the last occurrence takes precedence.
          required: false
          schema:
            type: array
            items:
              $ref: '#/components/schemas/SortValues'

        - name: page
          in: query
          description: Page number. If provided, size must also be provided. Otherwise, page is ignored.
          required: false
          schema:
            type: integer
            format: int32
            minimum: 1
            default: 1

        - name: size
          in: query
          description: Number of items per page. If page is provided, size must also be provided. Otherwise, size is ignored.
          required: false
          schema:
            type: integer
            format: int32
            minimum: 1
            default: 10

        - name: filter
          in: query
          description: Filter conditions in the format field(≥|⟸|>|◁|≠|=|^=|$=|@=)value. "^=" is for prefix matching, "$=" is for suffix matching, and "@=" is for substring matching. If a non-numeric field is used with a numeric operator or date format is invalid, an error is returned.
          required: false
          schema:
            type: array
            items:
              $ref: '#/components/schemas/FilterCondition'

      responses:
        '200':
          description: A list of musicbands.
          content:
            application/xml:
```

```yaml
              schema:
                type: array
                xml:
                  name: musicBands
                  wrapped: true
                items:
                  $ref: '#/components/schemas/MusicBandAllSchema'

        '400':
          $ref: '#/components/responses/BadRequestInvalidQueryParameters'
        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

    post:
      summary: Create a new musicband.
      description: Create a new musicband with the provided details.

      requestBody:
        description: Musicband to add.
        required: true
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/MusicBand'

      responses:
        '201':
          description: Musicband created successfully.
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/MusicBandAllSchema'

        '400':
          $ref: '#/components/responses/BadRequestInvalidInput'

        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

  /music-bands/{id}:
    get:
      summary: Get a musicband by ID.
      description: Retrieve a single musicband by its unique ID.

```

```yaml
    parameters:
      - name: id
        in: path
        description: The unique identifier of the musicband.
        required: true
        schema:
          $ref: '#/components/schemas/MusicBandId'

    responses:
      '200':
        description: A single musicband.
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/MusicBandAllSchema'

      '400':
        $ref: '#/components/responses/BadRequestInvalidID'

      '404':
        $ref: '#/components/responses/MusicBandNotFound'
      '422':
        $ref: '#/components/responses/InvalidIdFormat'

      '500':
        $ref: '#/components/responses/InternalServerError'
      default:
        description: Internal Server Error
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/Error'

  put:
    summary: Update an existing musicband (with all fields).
    description: Update an existing musicband by its ID. All fields must be
  ↪ provided. Missing fields will be set to null.

    parameters:
      - name: id
        in: path
        description: The unique identifier of the musicband to update.
        required: true
        schema:
          $ref: '#/components/schemas/MusicBandId'

    requestBody:
      description: Updated musicband details.
      required: true
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/MusicBand'

    responses:
      '200':
```

```yaml
          description: Musicband updated successfully.
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/MusicBandAllSchema'

        '400':
          $ref: '#/components/responses/BadRequestInvalidInput'

        '404':
          $ref: '#/components/responses/MusicBandNotFound'
        '422':
          $ref: '#/components/responses/InvalidIdFormat'
        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'
    delete:
      summary: Delete a musicband by ID.
      description: Delete a single musicband by its unique ID.

      parameters:
        - name: id
          in: path
          description: The unique identifier of the musicband to delete.
          required: true
          schema:
            $ref: '#/components/schemas/MusicBandId'

      responses:
        '204':
          description: Musicband deleted successfully. No content returned.

        '400':
          $ref: '#/components/responses/BadRequestInvalidID'

        '404':
          $ref: '#/components/responses/MusicBandNotFound'
        '422':
          $ref: '#/components/responses/InvalidIdFormat'
        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

    patch:
      summary: Partially update a musicband by ID.
```

```yaml
    description: Update one or more fields of an existing musicband by its ID. Only the provided fields will be updated; other fields will remain unchanged.

    parameters:
      - name: id
        in: path
        description: The unique identifier of the musicband to update.
        required: true
        schema:
          $ref: '#/components/schemas/MusicBandId'
    requestBody:
      description: Fields to update in the musicband.
      required: true
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/MusicBandPatch'
    responses:
      '200':
        description: Musicband updated successfully.
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/MusicBandAllSchema'

      '400':
        $ref: '#/components/responses/BadRequestInvalidInput'

      '404':
        $ref: '#/components/responses/MusicBandNotFound'
      '422':
        $ref: '#/components/responses/InvalidIdFormat'
      '500':
        $ref: '#/components/responses/InternalServerError'
      default:
        description: Internal Server Error
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/Error'

  /music-bands/all-with-description:
    delete:
      summary: Delete all musicbands with a specific description.
      description: Delete all musicbands whose description matches the provided value. If nothing is deleted, it means success too.

      parameters:
        - name: description
          in: query
          description: The description to filter musicbands for deletion.
          required: true
          schema:
            $ref: '#/components/schemas/MusicBandDescription'

      responses:
```

```
        '204':
          description: Musicbands deleted successfully. No content returned.

        '400':
          description: Bad Request - Missing or invalid description parameter.
          $ref: '#/components/responses/BadRequestInvalidQueryParameters'

        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

  /music-bands/one-with-genre:
    delete:
      summary: Delete one musicband with a specific genre.
      description: Delete one random musicband whose genre matches the provided
    ↪ value. Input is case-sencetive. If nothing is deleted, it means success too.

      parameters:
        - name: genre
          in: query
          description: The genre to filter musicbands for deletion.
          required: true
          schema:
            $ref: '#/components/schemas/MusicBandGenre'

      responses:
        '204':
          description: Musicband deleted successfully. No content returned.

        '400':
          description: Bad Request - Missing or invalid genre parameter.
          $ref: '#/components/responses/BadRequestInvalidQueryParameters'

        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

  /music-bands/count-best-album:
    get:
      summary: Count musicbands with best album greater than provided.
      description: Get the count of musicbands whose best album has more tracks
    ↪ than the specified number. If equals, uses alphabetical comparison of the
    ↪ names.

      parameters:
        - name: albumName
```

```yaml
          in: query
          required: true
          description: The name of the album to compare against.
          schema:
            type: string
            minLength: 1
        - name: albumTracks
          in: query
          required: true
          description: The number of tracks in the album to compare against.
          schema:
            type: integer
            format: int64
            minimum: 1

      responses:
        '200':
          description: Count of musicbands with best album tracks greater than the
    ↪ specified number.
          content:
            application/xml:
              schema:
                xml:
                  name: count
                type: integer
                format: int64
                example: 5

        '400':
          description: Bad Request - Missing or invalid tracks parameter.
          $ref: '#/components/responses/BadRequestInvalidQueryParameters'

        '500':
          $ref: '#/components/responses/InternalServerError'
        default:
          description: Internal Server Error
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Error'

components:
  schemas:
    Album:
      type: object
      xml:
        name: album
      properties:
        name:
          type: string
          minLength: 1
        tracks:
          type: integer
          format: int64
          minimum: 1
      required:
```

```yaml
        - name
        - tracks

    Coordinates:
      type: object
      properties:
        x:
          type: number
          format: float
          exclusiveMinimum: -975
        y:
          type: integer
          format: int64
          minimum: 1
      required:
        - x
        - y

    MusicBand:
      type: object
      xml:
        name: musicBand
      properties:
        name:
          type: string
          minLength: 1
        coordinates:
          $ref: '#/components/schemas/Coordinates'
        numberOfParticipants:
          type: integer
          format: int32
          minimum: 1
        albumsCount:
          type: integer
          format: int32
          minimum: 1
        description:
          $ref: '#/components/schemas/MusicBandDescription'
        genre:
          $ref: '#/components/schemas/MusicBandGenre'
        bestAlbum:
          $ref: '#/components/schemas/Album'
          nullable: true
      required:
        - name
        - coordinates
        - numberOfParticipants
        - albumsCount
        - description
        - genre
    MusicBandAllSchema:
      allOf:
        - $ref: '#/components/schemas/MusicBand'
        - type: object
          required:
            - id
```

```yaml
        properties:
          id:
            type: integer
            format: int32
            minimum: 1
          creationDate:
            type: string
            format: date
      xml:
        name: musicBandAllSchema
    MusicBandPatch:
      type: object
      xml:
        name: musicBand
      properties:
        name:
          type: string
          minLength: 1
        coordinates:
          $ref: '#/components/schemas/Coordinates'
        numberOfParticipants:
          type: integer
          format: int32
          minimum: 1
        albumsCount:
          type: integer
          format: int32
          minimum: 1
        description:
          $ref: '#/components/schemas/MusicBandDescription'
        genre:
          $ref: '#/components/schemas/MusicBandGenre'
        bestAlbum:
          $ref: '#/components/schemas/Album'
          nullable: true

    Error:
      type: object
      required:
        - code
        - message
      xml:
        name: error
      properties:
        code:
          type: integer
          format: int32
          example: 500
        message:
          type: string
          example: Internal Server Error

    InternalServerError:
      allOf:
        - $ref: '#/components/schemas/Error'
        - type: object
```

```
          properties:
            code:
              example: 500
            message:
              example: Internal Server Error
    NotFoundError:
      allOf:
        - $ref: '#/components/schemas/Error'
        - type: object
          properties:
            code:
              example: 404
            message:
              example: Not Found
    BadRequestError:
      allOf:
        - $ref: '#/components/schemas/Error'
        - type: object
          properties:
            code:
              example: 400
            message:
              example: Bad Request
    UnprocessableEntityError:
      allOf:
        - $ref: '#/components/schemas/Error'
        - type: object
          properties:
            code:
              example: 422
            message:
              example: Unprocessable Entity

    FilterCondition:
      type: string
      pattern: '^((name|creationDate|description|genre|bestAlbum\.name)
    ↪ (⩾|◁>|⟛≠|=|^=|$=|@=).+)|(coordinates\.x(⩾|⟛>|◁≠|=)-?\d+(\.\d+)?|(
    ↪ id|coordinates\.y|numberOfParticipants|albumsCount|bestAlbum\.tracks)
    ↪ (⩾|⟛>|◁≠|=)-?\d+)$'

    SortValues:
      type: string
      pattern: '^(id|-id|name|-name|coordinates\.x|-coordinates\.x|coordinates\.y|-
    ↪ coordinates\.y|creationDate|-creationDate|numberOfParticipants|-
    ↪ numberOfParticipants|albumsCount|-albumsCount|description|-description|genre
    ↪ |-genre|bestAlbum\.name|-bestAlbum\.name|bestAlbum\.tracks|-bestAlbum\.tracks
    ↪ )$'

    PatchOption:
      type: string
      xml:
        name: field
      pattern: '^((name|creationDate|description|genre|bestAlbum\.name)=.+)|(
    ↪ coordinates\.x=-?\d+(\.\d+)?|(id|coordinates\.y|numberOfParticipants|
    ↪ albumsCount|bestAlbum\.tracks)=-?\d+)$'

```

```yaml
    MusicBandId:
      type: integer
      format: int32
      minimum: 1

    MusicBandDescription:
      type: string
      minLength: 0

    MusicBandGenre:
      type: string
      enum:
        - PSYCHEDELIC_CLOUD_RAP
        - SOUL
        - POP

  responses:
    InternalServerError:
      description: Internal Server Error
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/InternalServerError'

    MusicBandNotFound:
      description: Not Found - Musicband not found.
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/NotFoundError'

    BadRequestInvalidID:
      description: Bad Request - Invalid ID supplied.
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/BadRequestError'
    BadRequestInvalidQueryParameters:
      description: Bad Request - Invalid Query Parameters.
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/BadRequestError'

    BadRequestInvalidInput:
      description: Bad Request - Invalid input.
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/BadRequestError'

    InvalidIdFormat:
      description: Parameter 'id' must be a positive integer.
      content:
        application/xml:
          schema:
```

```
616              $ref: '#/components/schemas/UnprocessableEntityError'
```

2.2 Grammy Service

grammy.yaml

```yaml
openapi: 3.1.1
info:
  title: Grammy Service Endpoints - OpenAPI 3.1.1
  description: |-
    **OpenAPI** specification for **Laboratory work №1** of **Service-oriented
    ↪ architecture** at **ITMO-University**.

  contact:
    name: Roman Kobelev & Balin Artem
    url: https://github.com/ta4ilka69/Service-oriented-architecture

  license:
    name: The MIT License
    url: https://mit-license.org/

  version: 0.0.1

servers:
  - url: 'https://localhost:5252/api/v1/grammy'
    description: Helios

paths:
  /band/{band-id}/singles/add:
    post:
      summary: Add a single to the band
      description: Add a single to the band. Accepts a string in body

      requestBody:
        required: true
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/SingleSchema'

      responses:
        '201':
          description: Created. Returns created object
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Single'

        '400':
          description: Bad Request - Invalid name parameter.
          $ref: './music.yaml#/components/responses/BadRequestInvalidInput'
        '404':
          $ref: './music.yaml#/components/responses/MusicBandNotFound'
        '422':
          $ref: './music.yaml#/components/responses/InvalidIdFormat'
        '500':
```

```yaml
          $ref: './music.yaml#/components/responses/InternalServerError'
        '503':
          $ref: '#/components/responses/ServiceUnavailable'
        default:
          description: Returns error code and message in case of error
          content:
            application/xml:
              schema:
                $ref: './music.yaml#/components/schemas/Error'

      parameters:
        - name: band-id
          in: path
          description: ID of the band to add a single to
          required: true
          schema:
            $ref: './music.yaml#/components/schemas/MusicBandId'

  /band/{band-id}/participants/add:
    post:
      summary: Add a participant to the band
      description: Add a participant to the band

      responses:
        '201':
          description: Created. Returns new object
          content:
            application/xml:
              schema:
                $ref: './music.yaml#/components/schemas/MusicBandAllSchema'
        '400':
          description: Bad Request - Invalid name parameter.
          $ref: './music.yaml#/components/responses/BadRequestInvalidInput'
        '404':
          $ref: './music.yaml#/components/responses/MusicBandNotFound'
        '422':
          $ref: './music.yaml#/components/responses/InvalidIdFormat'
        '500':
          $ref: './music.yaml#/components/responses/InternalServerError'
        '503':
          $ref: '#/components/responses/ServiceUnavailable'
        default:
          description: Returns error code and message in case of error
          content:
            application/xml:
              schema:
                $ref: './music.yaml#/components/schemas/Error'

      parameters:
        - name: band-id
          in: path
          description: ID of the band to add a participant to
          required: true
          schema:
            $ref: './music.yaml#/components/schemas/MusicBandId'

```

```yaml
      requestBody:
        required: true
        content:
          application/xml:
            schema:
              $ref: '#/components/schemas/ParticipantSchema'


components:
  schemas:
    Single:
      allOf:
        - $ref: './music.yaml#/components/schemas/Album'
      xml:
        name: single
    SingleSchema:
      type: object
      properties:
        name:
          type: string
          xml:
            name: name
      required:
        - name
    ParticipantSchema:
      type: object
      properties:
        name:
          description: Имя нового участника.
          type: string
          minLength: 1
      required:
        - name
    ServiceUnavailableError:
      allOf:
        - $ref: './music.yaml#/components/schemas/Error'
        - type: object
          properties:
            code:
              example: 503
            message:
              example: Service Unavailable
  responses:
    ServiceUnavailable:
      description: Service Unavailable
      content:
        application/xml:
          schema:
            $ref: '#/components/schemas/ServiceUnavailableError'
```

3 Вывод

В ходе выполнения лабораторной работы было расписаны спецификации двух сервисов в виде **yaml** файлов. Разобрались с синтаксисом языка для написания спецификаций.