UNIVERSITY of INFORMATION TECHNOLOGY and MANAGEMENT in Rzeszow, POLAND

FACULTY OF APPLIED INFORMATION TECHNOLOGY

Field Of Study: Information Technology

Specialty: Programming 3LID-A-P/2022-ASL02

Project

Course: PROJECT_Technologie webowe (Web Technologies)

Yerzhanov Adil W68671

Books API Documentation

Introduction:

Welcome to the Books API documentation. This API is designed to manage a library's book database, allowing users to perform various operations like adding, updating, searching, and deleting books. This API is mostly directed to optimize the system of searching, editing, and working with the data of existing books in the library. It uses all CRUD methods and JWT Tokens to ensure access to the data. It can be easily changed to work with the data of a real library since the books are represented as simple JSON objects.

Table of Contents

1.CRUD Methods :

1.1 Get Books <a name="get-books"></a>

Endpoint: GET /books

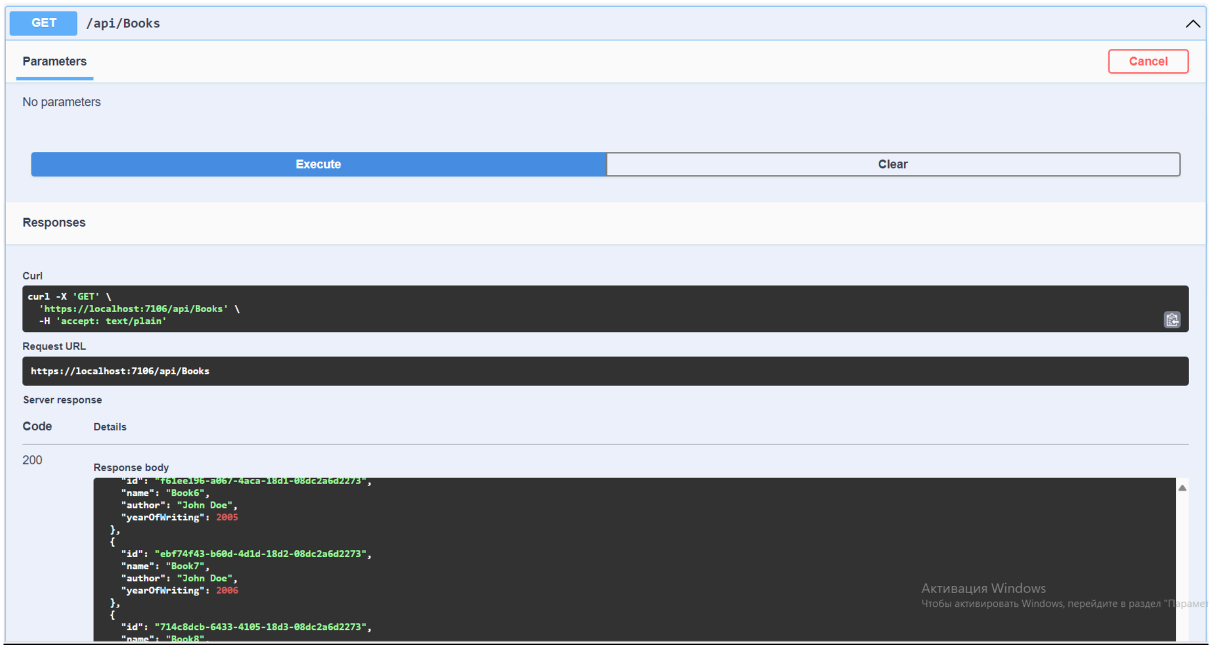
Description: Retrieve a list of all books in the library.

Authentication: Not required

Response:

- 200 OK: Returns a list of books.

Example Request:



Curl

```
curl -X 'GET' \
  'https://localhost:7106/api/Books' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7106/api/Books
```

Server response

Code Details

200 Response body

```
    "id": "f61ee196-a067-4aca-18d1-08dc2a6d2273",
    "name": "Book6",
    "author": "John Doe",
    "yearOfWriting": 2005
  },
  {
    "id": "ebf74f43-b60d-4d1d-18d2-08dc2a6d2273",
    "name": "Book7",
    "author": "John Doe",
    "yearOfWriting": 2006
  },
  {
    "id": "714c8dcb-6433-4105-18d3-08dc2a6d2273",
    "name": "Book8",
```

2.2 Add Books <a name="add-books"></a>

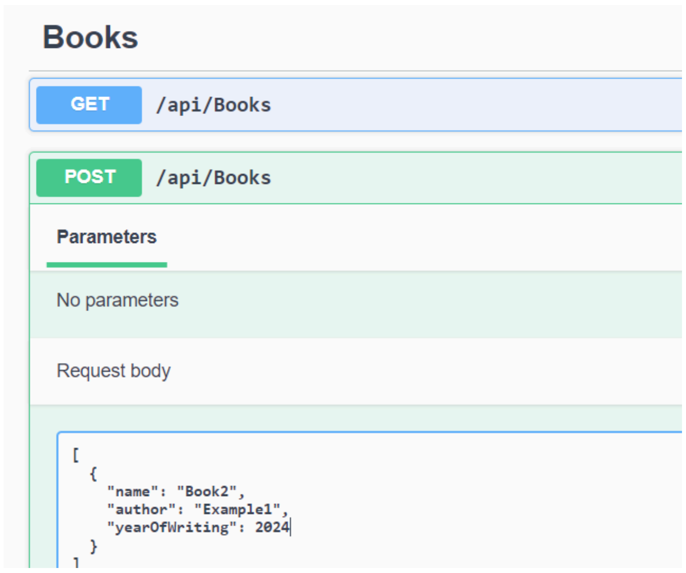
Endpoint: POST /books

Description: Add one or more books to the library.

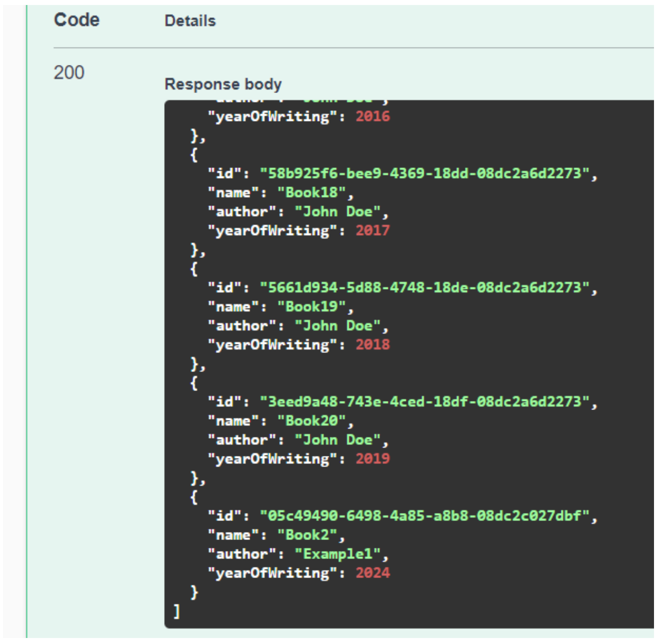
Authentication: Not Required

Response:

- 200 OK: Returns the updated list of books.



```
[
  {
    "name": "Book2",
    "author": "Example1",
    "yearOfWriting": 2024
  }
]
```



```
    "yearOfWriting": 2016
  },
  {
    "id": "58b925f6-bee9-4369-18dd-08dc2a6d2273",
    "name": "Book18",
    "author": "John Doe",
    "yearOfWriting": 2017
  },
  {
    "id": "5661d934-5d88-4748-18de-08dc2a6d2273",
    "name": "Book19",
    "author": "John Doe",
    "yearOfWriting": 2018
  },
  {
    "id": "3eed9a48-743e-4ced-18df-08dc2a6d2273",
    "name": "Book20",
    "author": "John Doe",
    "yearOfWriting": 2019
  },
  {
    "id": "05c49490-6498-4a85-a8b8-08dc2c027dbf",
    "name": "Book2",
    "author": "Example1",
    "yearOfWriting": 2024
  }
]
```

2.3 Update Book <a name="update-book"></a>

Endpoint: PUT /books/{id}

Description: Update information for a specific book.

Authentication: Not Required

Request Parameters:

- {id}: Book ID (Guid)

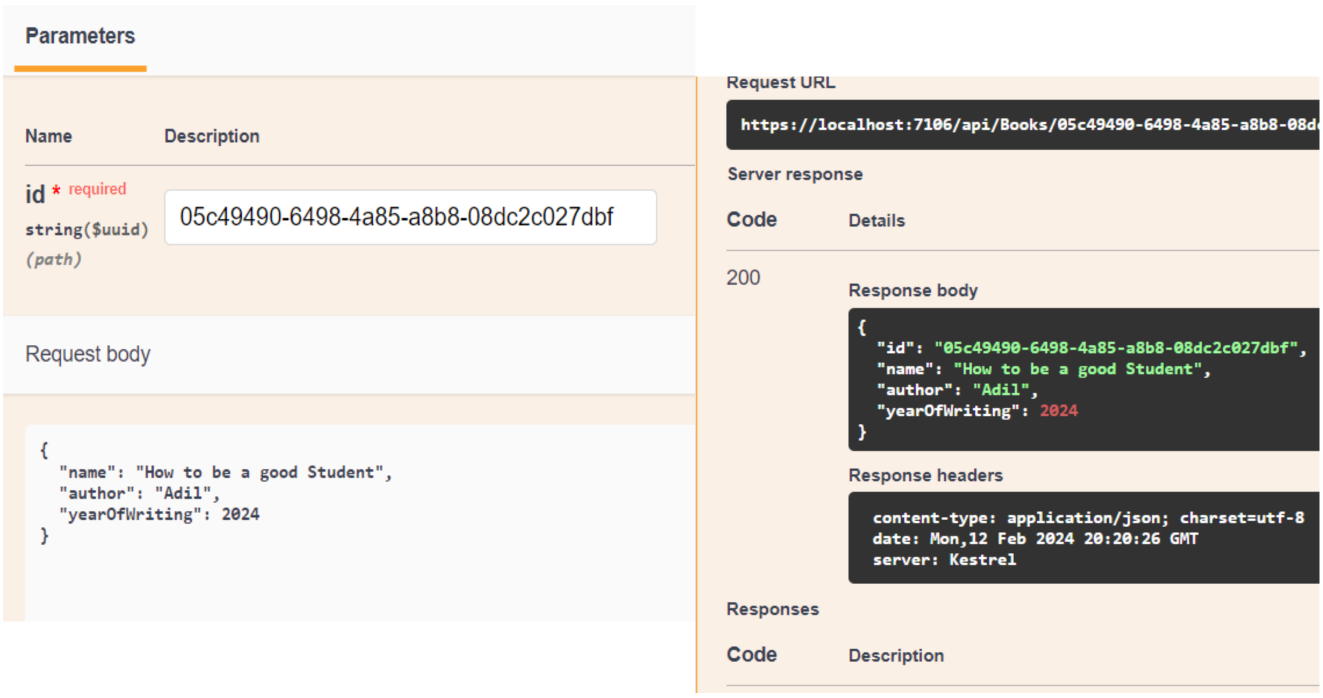
Request Body:

- UpdateBookRequest object

Response:

- 200 OK: Returns the updated book.

Example Request:



05c49490-6498-4a85-a8b8-08dc2c027dbf

```
{
  "name": "How to be a good Student",
  "author": "Adil",
  "yearOfWriting": 2024
}
```

```
https://localhost:7106/api/Books/05c49490-6498-4a85-a8b8-08d
```

200

```
{
  "id": "05c49490-6498-4a85-a8b8-08dc2c027dbf",
  "name": "How to be a good Student",
  "author": "Adil",
  "yearOfWriting": 2024
}
```

```
content-type: application/json; charset=utf-8
date: Mon,12 Feb 2024 20:20:26 GMT
server: Kestrel
```

2.4 Get Book by ID <a name="get-book-by-id"></a>

Endpoint: GET /books/{id}

Description: Retrieve details for a specific book using its ID.

Authentication: Not required

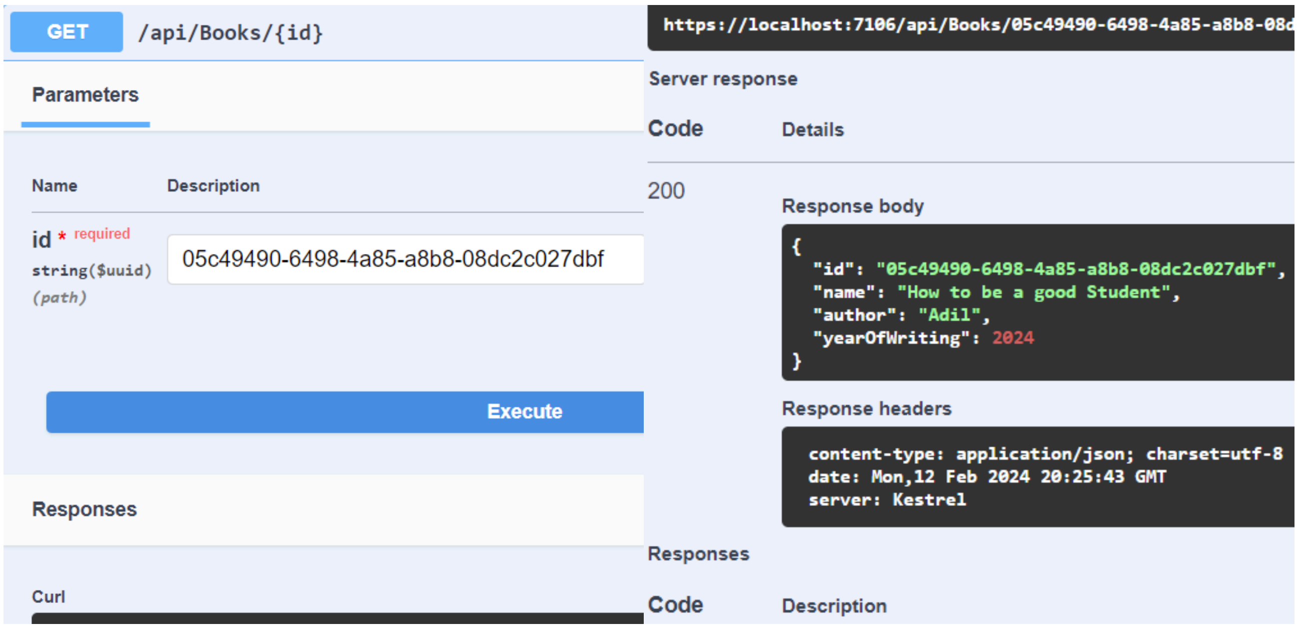
Request Parameters:

- {id}: Book ID (Guid)

Response:

- 200 OK: Returns the book details.

Example Request:


GET /api/Books/{id}

Parameters

Name Description

id * required string($uuid) (path) 05c49490-6498-4a85-a8b8-08dc2c027dbf

Execute

Responses

Curl

```
https://localhost:7106/api/Books/05c49490-6498-4a85-a8b8-08d
```

Server response

Code Details

200

Response body

```
{
  "id": "05c49490-6498-4a85-a8b8-08dc2c027dbf",
  "name": "How to be a good Student",
  "author": "Adil",
  "yearOfWriting": 2024
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,12 Feb 2024 20:25:43 GMT
server: Kestrel
```

Responses

Code Description

2.5 Delete Book <a name="delete-book"></a>

Endpoint: DELETE /books/{id}

Description: Delete a specific book from the library.

Authentication: Required (JWT)

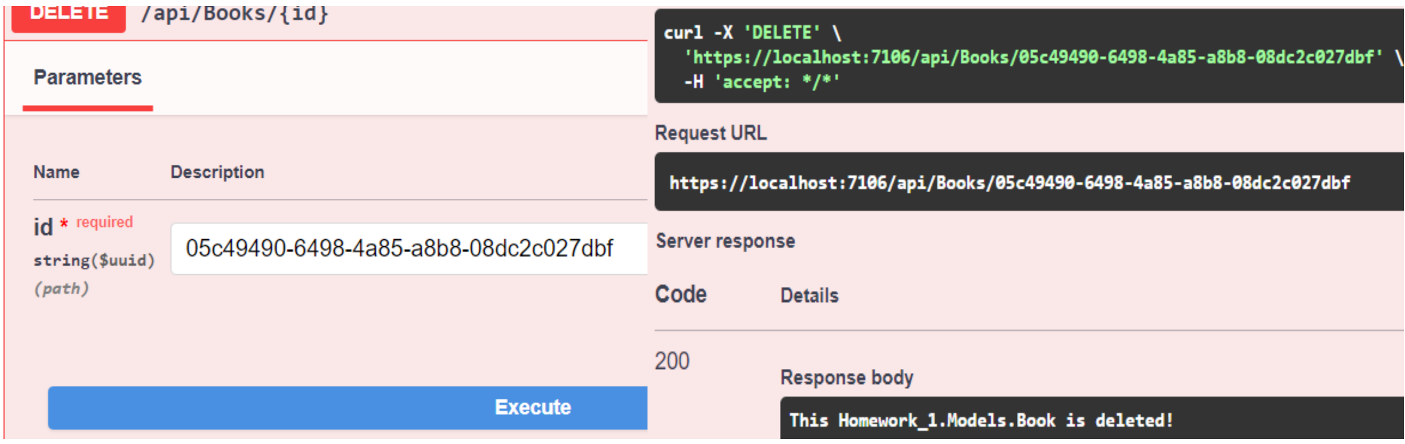
Request Parameters:

- {id}: Book ID (Guid)

Response:

- 200 OK: Returns a confirmation message.

Example Request:


DELETE /api/Books/{id}

Parameters

Name Description

id * required string($uuid) (path) 05c49490-6498-4a85-a8b8-08dc2c027dbf

Execute

```
curl -X 'DELETE' \
  'https://localhost:7106/api/Books/05c49490-6498-4a85-a8b8-08dc2c027dbf' \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7106/api/Books/05c49490-6498-4a85-a8b8-08dc2c027dbf
```

Server response

Code Details

200

Response body

```
This Homework_1.Models.Book is deleted!
```

2.6 Search Books <a name="search-books"></a>

Endpoint: GET /books/search

Description: Search for books based on author and/or year.

Authentication: Not required

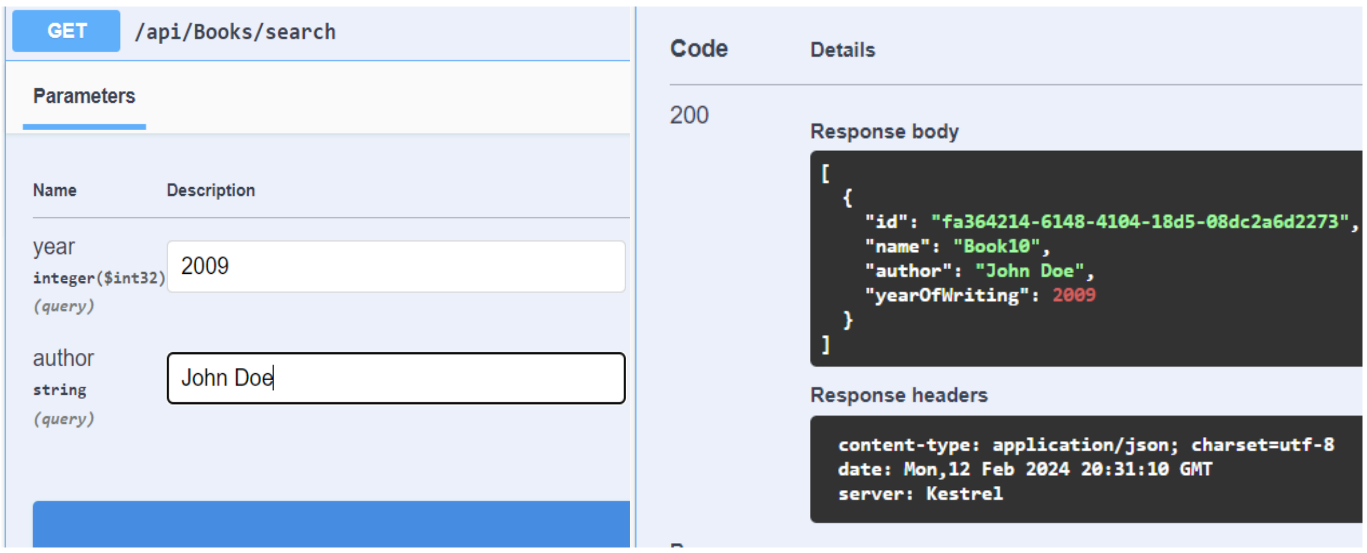
Request Parameters:

- year (optional): Filter books by the year of writing.
- author (optional): Filter books by author.

Response:

- 200 OK: Returns a list of matching books.

Example Request:



Name	Description
year integer($int32) (query)	2009
author string (query)	John Doe

200

Response body

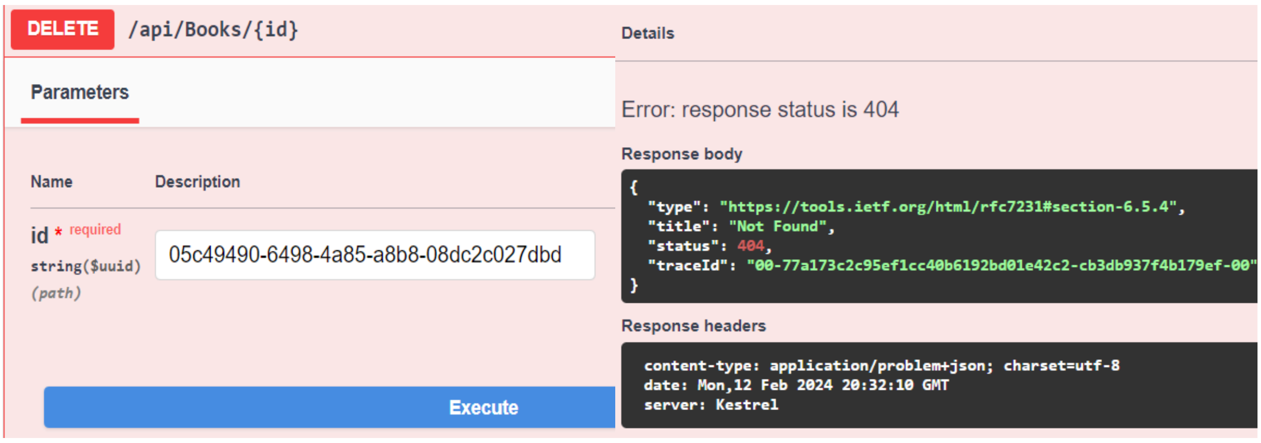
```
[
  {
    "id": "fa364214-6148-4104-18d5-08dc2a6d2273",
    "name": "Book10",
    "author": "John Doe",
    "yearOfWriting": 2009
  }
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,12 Feb 2024 20:31:10 GMT
server: Kestrel
```

3. Error Handling <a name="error-handling"></a>

In case of errors, the API will return appropriate HTTP status codes along with error messages. Refer to the specific endpoint documentation for details on possible error responses. In this example not correct ID was inserted to Delete method:



05c49490-6498-4a85-a8b8-08dc2c027dbd

Error: response status is 404

Response body

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
  "title": "Not Found",
  "status": 404,
  "traceId": "00-77a173c2c95ef1cc40b6192bd01e42c2-cb3db937f4b179ef-00"
}
```

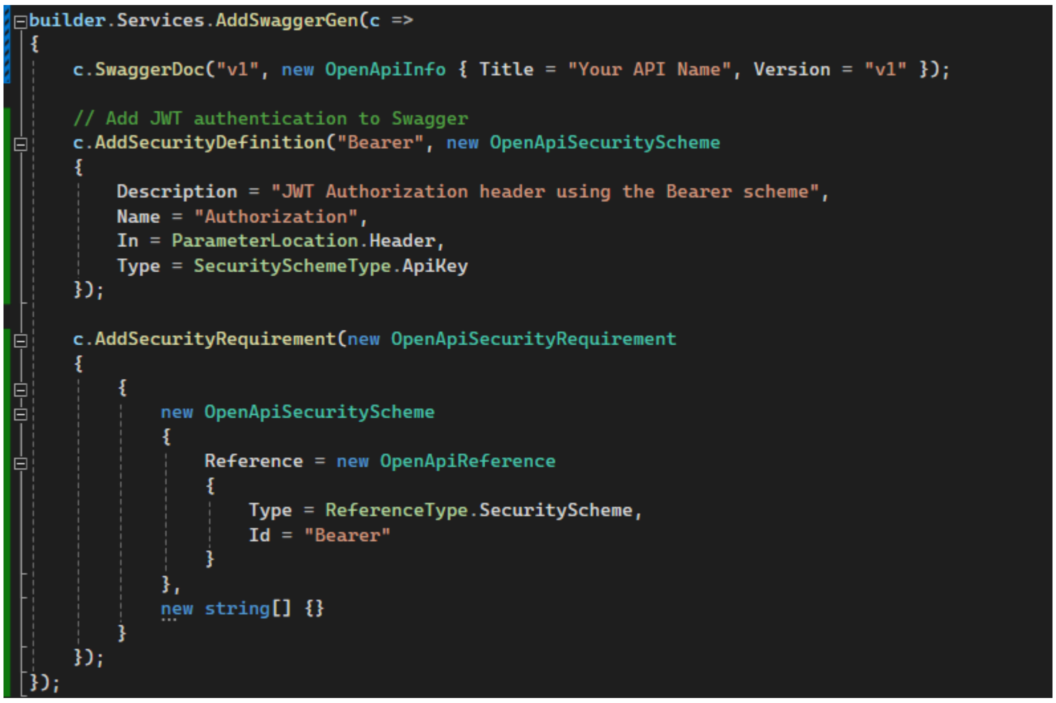
Response headers

```
content-type: application/problem+json; charset=utf-8
date: Mon,12 Feb 2024 20:32:10 GMT
server: Kestrel
```

5. Security <a name="security"></a>

The API uses JWT for authentication, ensuring secure access to authorized users. In this project the methods can change their security access if the Admin would like to do that, but currently all methods can be used without passing the JWT token.

6. Swagger Documentation

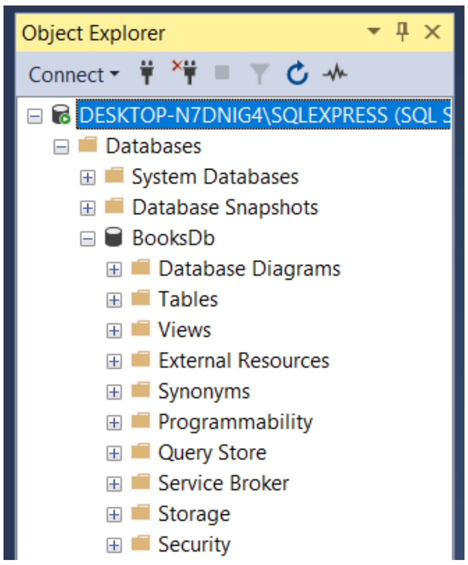


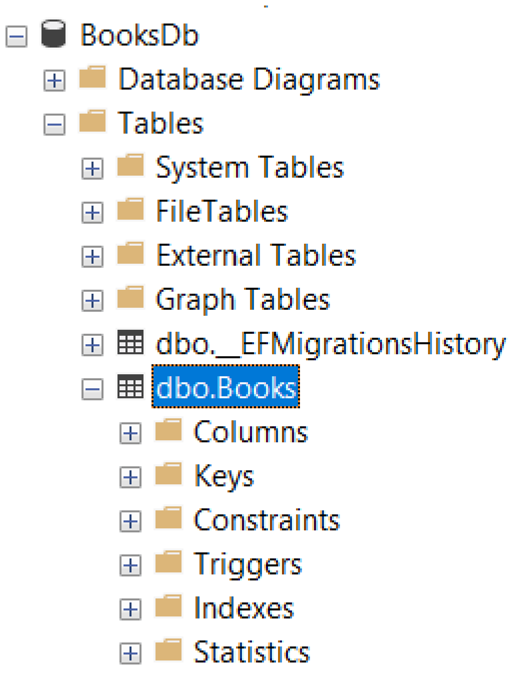
```
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "Your API Name", Version = "v1" });

    // Add JWT authentication to Swagger
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Description = "JWT Authorization header using the Bearer scheme",
        Name = "Authorization",
        In = ParameterLocation.Header,
        Type = SecuritySchemeType.ApiKey
    });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            new string[] {}
        }
    });
});
```

The code's documentation is made on Swagger for better user navigation on each method.

7.The connection to the database:

The connection to the data is made on MS SQL server as a relational table of the data. As it was written before, the process of adding the new JSON objects is pretty easy since the PUT method can take more than one JSON object and give for each object its unique ID.


Object Explorer
Connect
DESKTOP-N7DNIG4\SQLEXPRESS (SQL S
Databases
System Databases
Database Snapshots
BooksDb
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security


BooksDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.__EFMigrationsHistory
dbo.Books
Columns
Keys
Constraints
Triggers
Indexes
Statistics