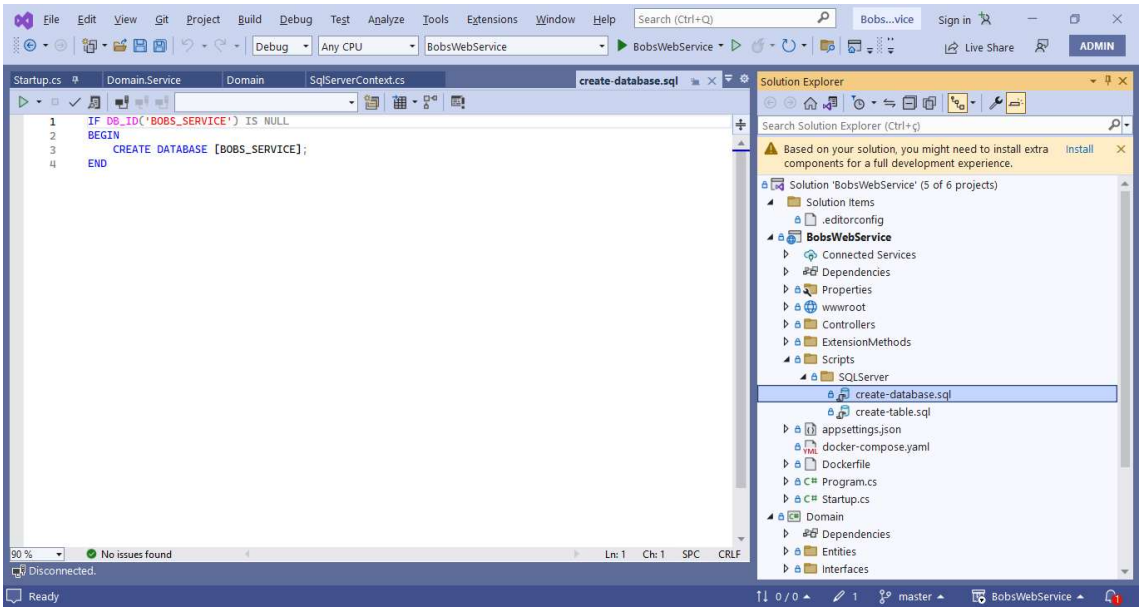
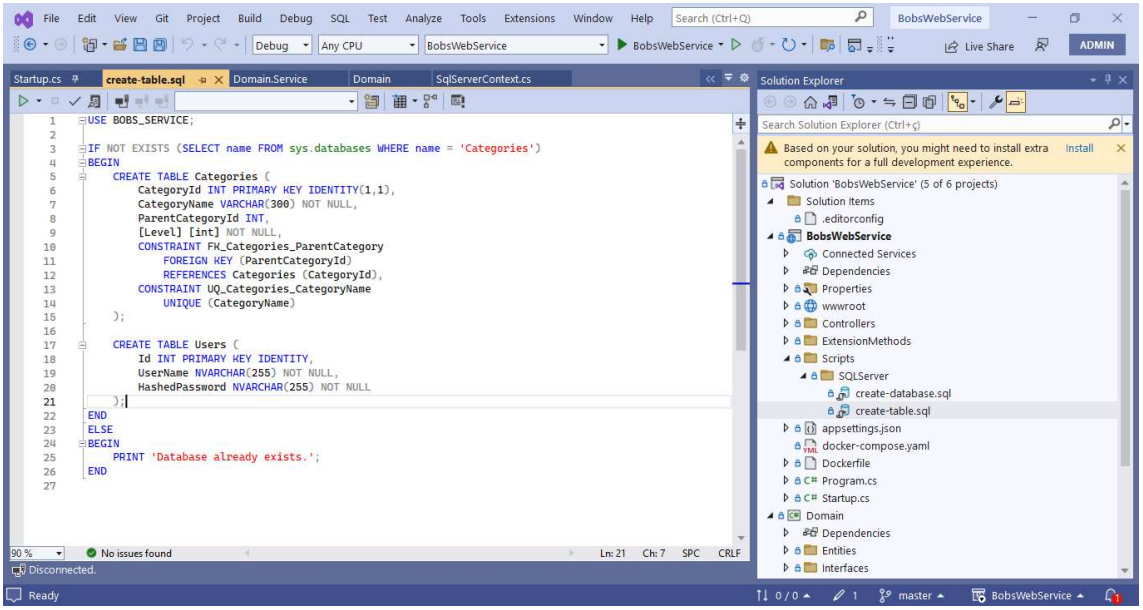
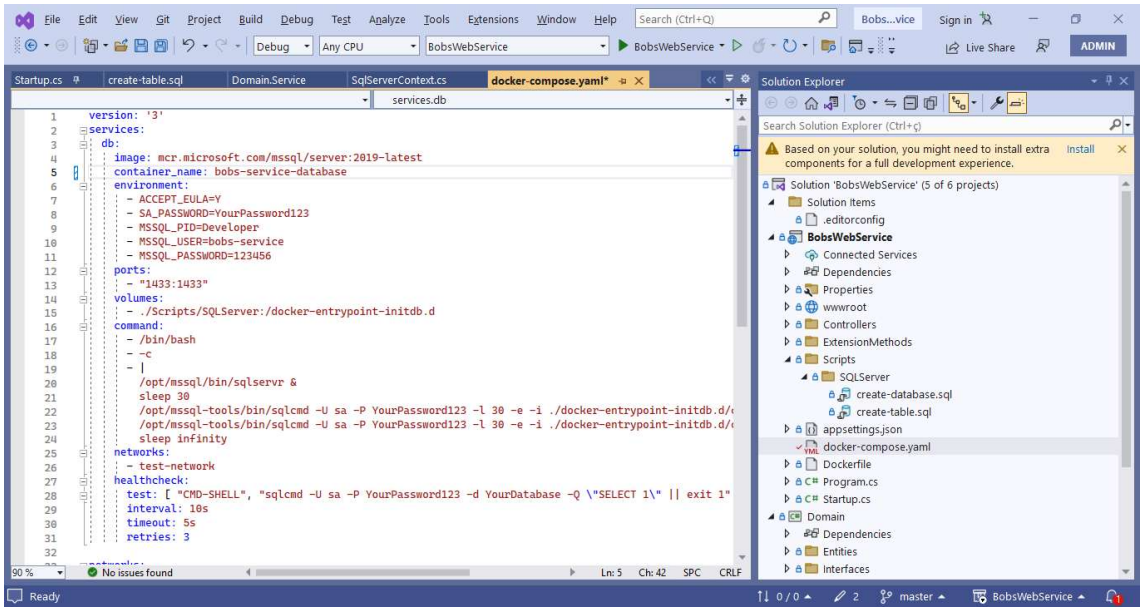
BobServiceWeb

To run the application, execute docker-compose up command in the BobsWebService folder. This will start the infrastructure creation, including the database and tables. The SQL script file for database creation is also located in the same folder, named " BobsWebService /Scripts/SQLServer".





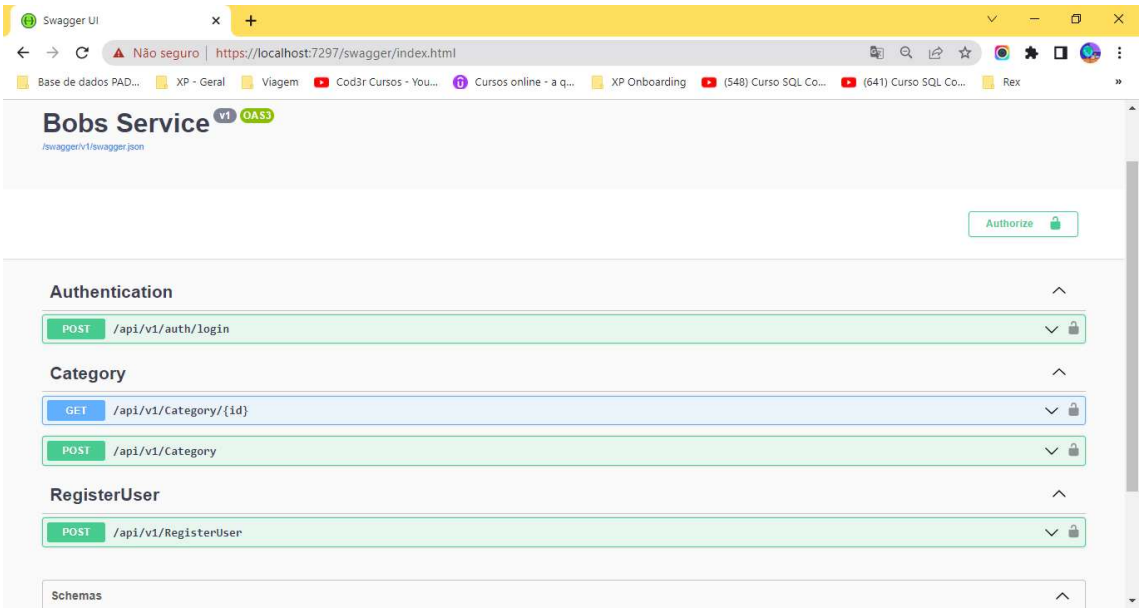


To run the project, you will need the following software installed on your machine:

.NET 6: Install the .NET 6 SDK, which includes the necessary tools and libraries for building and running .NET 6 applications. You can download it from the official Microsoft website.

Docker: Install Docker, which is a platform for containerization. Docker allows you to package and run applications in isolated environments called containers. You can download Docker from the official Docker website and follow the installation instructions specific to your operating system.

Docker Compose: Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to specify the services, networks, and volumes required by your application in a YAML file. Docker Compose is usually installed automatically when you install Docker on your machine.



AuthenticationController

The AuthenticationController class handles the authentication process for user login in the application.

Constructor

AuthenticationController(IUserAuthService userAuthService)

userAuthService: An instance of IUserAuthService used for user authentication.

Routes

POST api/v1/auth/login

Handles user login and authentication.

Method: POST

Route: api/v1/auth/login

Request Body

The request body should contain the following parameters:

Name: The username of the user.

Password: The password of the user.

Response

If the login request is successful and the user credentials are valid, the response will have a status code of 200 (OK) and the following JSON structure:

json

```
{
  "Status": true,
  "Message": "",
  "Token": "token-in-here"
}
```

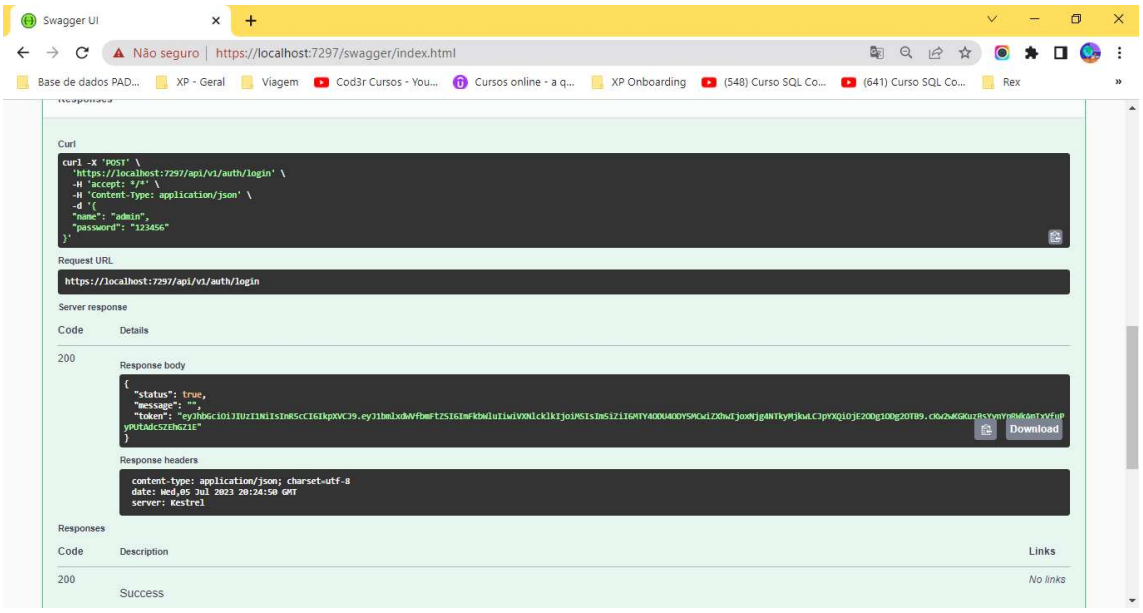
Status: A boolean value indicating the success status of the login process.

Message: An optional message related to the login process.

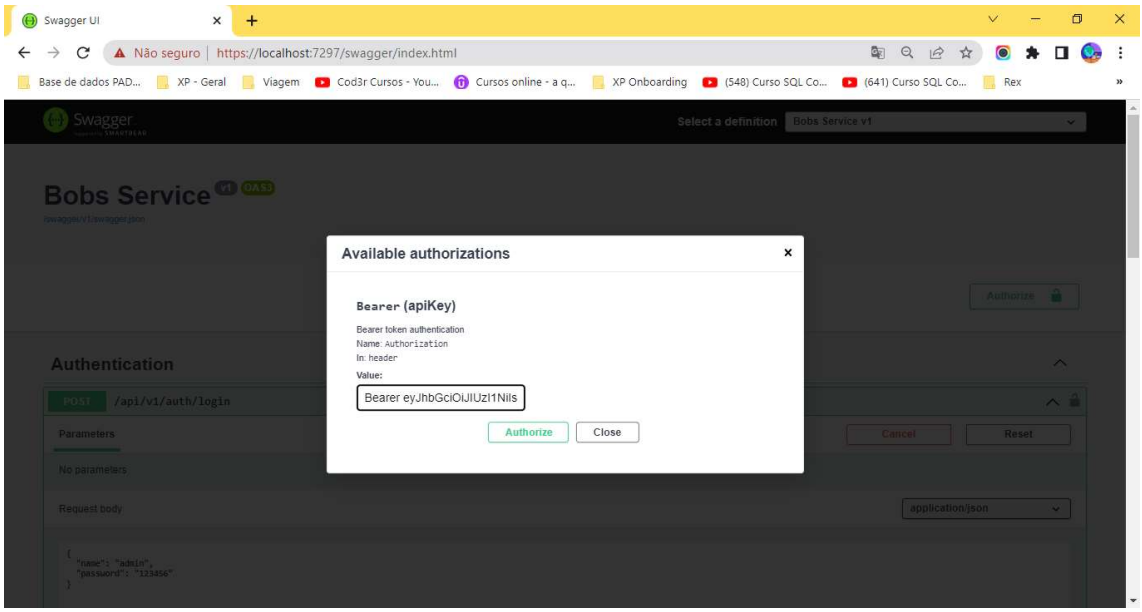
Token: A JWT token generated for the authenticated user.

If the login request fails due to invalid credentials or other errors, the response will have an appropriate status code and an error message will be returned in the response body.

Example authentication



When receiving the authentication payload, include the Bearer token as follows: "Bearer {your authentication token}".



Additional Details

The AuthenticationController class is responsible for handling user authentication for login requests.

The IUserAuthService dependency is injected into the constructor to perform user authentication and token generation.

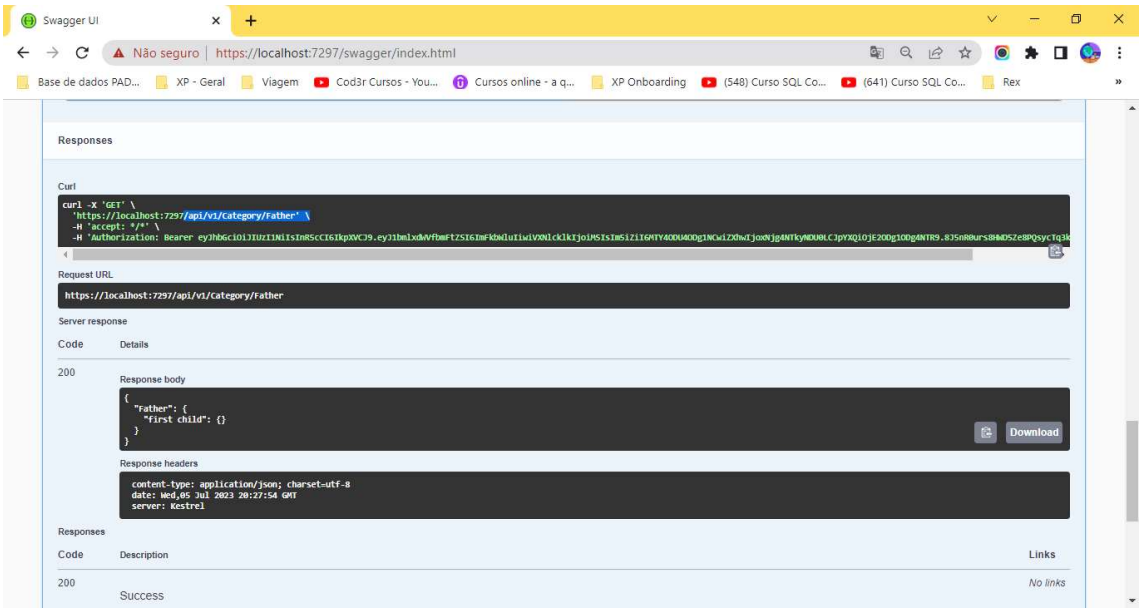
The Login method receives the login credentials from the request body and performs the authentication logic. If the provided credentials are valid, a token is generated using the authService and returned in the response.

The user's password is stored in a hashed form in the database. The validation of the password is done using the IsEqualPassword method, which compares the hashed password from the database with the hashed password generated from the user's input.

If an error occurs during the authentication process, an appropriate status code and error message are returned in the response.

Please note that this documentation is based on the code snippets you provided. If there are additional details or functionality that need to be documented, please let me know.

API for querying categories saved in the database by searching for the Parent name.



Contract of return

```
{
  "Father": {
    "first child": {}
  }
}
```

CategoryManagementService

The CategoryManagementService class is responsible for managing categories in the application.

Constructor

CategoryManagementService(SqlServerContext sqlServerContext, ILogger<CategoryManagementService> logger)

sqlServerContext: The SQL Server context used for database operations.

logger: The logger instance used for logging.

Methods

CreateCategory(Dictionary<string, string> categoryDictionary)

Creates categories based on a dictionary of parent and child category names.

categoryDictionary: A dictionary containing the parent and child category names.

GetCategory(string categoryName)

Retrieves a category and its child categories based on the category name.

categoryName: The name of the category.

Returns: A dictionary representing the category and its child categories.

Additional Details

The CreateCategory method takes a dictionary of category names, where the keys represent parent categories and the values represent child categories. It creates the categories in the database by creating CategoryEntity objects and updating the Categories table using the sqlServerContext.

The GetCategory method retrieves a category and its child categories from the database based on the provided category name. It returns a dictionary that represents the category hierarchy.

The CategoryManagementService class uses the SqlServerContext for database operations and the ILogger for logging.