TaxiOperator API

User Guide

Table of Contents

[1. Introduction: 3](#_Toc74310908)

[2. Requirements: 3](#_Toc74310909)

[3. Configuration: 3](#_Toc74310910)

[3.1 Database bak file: 3](#_Toc74310911)

[3.2 Edit Appsettings.json File: 3](#_Toc74310912)

[4. Running the application: 3](#_Toc74310913)

[5. Features: 3](#_Toc74310914)

[5.1 Data Controller: 3](#_Toc74310915)

[5.2 User Controller: 4](#_Toc74310916)

[6. Import/Export postman collection: 4](#_Toc74310917)

[6.1 Import: 4](#_Toc74310918)

[6.2 Export: 4](#_Toc74310919)

[7. Automated tests: 5](#_Toc74310920)

# Introduction:

TaxiOperator API is used to manage and view the all operations that are done on taxi model and to make authorization and authentication for users using JWT tokens in .net core framework.

# Requirements:

* .Net core version 3.1.
* SQL server instance 16.
* Visual studio 2019 or any other IDE that supports the mentioned .net core version
* Postman

# Configuration:

## Database bak file:

Start by restoring the bak file in the solution to your database.

## Edit Appsettings.json File:

After opening the application in your favorite IDE, go to **appsettings.json** file and make sure to update the **DefaultConnection** field to suits your database configurations.

# Running the application:

In case of using visual studio 2019 the application can run directly by pressing F5 to run in debugging mode or (ctrl + F5) to run without debugging mode. Also as a second way open a new Cmd and change directory to the location of the application **Ex: Cd D:\WorkBanch\taxiOperator**. And then run **dotnet run** command in the Cmd.

# Features:

This API contains actions in the DataController to make CRUD operations on customer table in the database.

## Data Controller:

To Get List of customers, select GET as request type and use this URL: <https://localhost:44319/Data>

To Delete Customer, Select DELETE as request type and use this URL: [https://localhost:44319/Data/{{Id}}](https://localhost:44319/Data/%7b%7bId%7d%7d)

To save Customer data, select POST as request type and use this URL. Don’t forget to add the customer object in the request body: https://localhost:44319/Data

## User Controller:

Use this URL to authenticate user and get required token:

Use username = Admin

Password = Admin

[PostRequest] : <https://localhost:44319/Account/login>

**NOTE**:

In case of using postman make sure to add the following script in Tests sections of the request to make sure to add the token to your current environment:

const user = pm.response.json();

pm.test("Has properties", function () {

pm.expect(user).to.have.property('username');

pm.expect(user).to.have.property('token');

});

if (pm.test("Has properties")) {

pm.globals.set('token', user.token);

}

And to add Authorization field to request Header and giving it value: Bearer {{Token}}

# Import/Export postman collection:

A JSON formatted file is found in Documentation and backup folder in the solution that contains the required http requests, and this file can be imported to postman

## Import:

To import a collection to postman, Click on the file tab and then click import.

## Export:

To Export the requests collection from postman, Right click on selected collection and go to the export command, then click the recommended version to export. The exported data will be a file of JSON extension

# Automated tests:

TaxiOperator solution contains a project to unit testing.

|  |  |
| --- | --- |
| Method Name | Description |
| GetCustomerList\_WhenCalled\_ReturnsOkResult() | This testing method is used to check if OkObjectResult is returned when getting list of customers from database using the data controller. |
| GetById\_ExistingIdPassed\_ReturnsOkResult() | This method is used to check if OkObjectResult is returned when getting a customer with existed Id. |
| GetById\_ExistingIdPassed\_ReturnsRightItem() | This method checks if the returned customer item has the same id with the Id parameter passed to the DataController action. |
| SaveCustomer\_InvalidObjectPassed\_ReturnsBadRequest() | This method checks if BadRequestObjectResult is returned when sending customer object to be saved that doesn’t have a required and not null field Id. |
| SaveCustomer\_ValidObjectPassed\_ReturnsOkResult() | This method checks if OkResult is returned when saving a customer object |
| DeleteCustomer\_ExistingIdPassed\_ReturnsOkResult() | This method checks if OkResult is returned when deleting a customer with specific Id |

Those automated tests can be extended by taking into consideration all possible returned responses from the actions. In addition some tests can be added to make sure of authorization and also role authorization for the users. And also, this unit testing solution can contain many classes in which each class will represent the testing methods for different API versions in case of API versioning