Engineer to Engineer Guide

Proekspert – MyIoTService Solution



Prepared by:

*Muhammad Zubair*

Table of Contents

Table of Contents ii

1. Introduction 1

2. Tools and Technology 1

2.1 Technologies 1

2.2 Tools 1

3. Folder Structure 1

4. IoTDevice 3

5. MyIoTService 4

5.1 ERD Diagram 4

6. Running the Project 4

References 5

# Introduction

MyIotService is Cloud solution that provide the end users to create accounts and register IoT devices and manage(Add, Update, Delete, etc.) them.

# Tools and Technology

MyIoTService has been developed using Asp.Net Core 3.1 and Microsoft SQL server for data storage. Following are the complete list of technologies and their purposes mentioned for the reviewer.

## Technologies

1. **IoTDevice**
   1. Bogus 34.0.2
   2. Microsoft Entity Framework Core 3.1.24
   3. Microsoft Entity Framework Core (Sqlite) 3.1.24
   4. Swashbuckle Asp.Net Core 6.3.1 (Swagger UI)
2. **MyIoTService**
   1. Asp.Net Core JwtBearer Authorization 3.1.24
   2. JWT Token Authorization 6.17.0
   3. Microsoft Entity Framework Core 3.1.24
   4. Microsoft Entity Framework Core (SqlServer) 3.1.24
   5. Swashbuckle Asp.Net Core 6.3.1 (Swagger UI)

## Tools

Following tools are required to run the application:

1. Microsoft Visual Studio (2022)
2. Microsoft SqlServer 2019

# Folder Structure

This section will explain the folder structure used throughout the development.

Text

Description automatically generated

1. **Controllers**

This folder contains the Web-API end points.

1. **Database**

This folder contains the database context file for entity framework. These files are responsible to create database based on entities defined in Entities folder.

1. **Entities**

This folder contains the database entities which needs to be stored in the database. These entities will never use for handling requests and responses. To return the data of entity, this data first needs to be stored in models then those models will be returned and vice-versa.

1. **Exceptions**

This folder contains the classes with the exception messages. No string data would return through APIs until unless It comes from a class object. We have to define every message in an object to avoid redundancies of test messages.

1. **Middleware/Helpers**

This folder contains the middleware and helper methods that will help to control the execution of in a sense of security and validation.

1. **Migrations**

This folder is auto-generated via PackageManagerConsole. This folder contains the migration of entities and predefined data which will be used to create database on execution.

1. **Models**

This folder contains the model classes in which API will receive data and the processed data will return. These classes can be multiple and can have any number of attributes according to need and separation of concerns.

1. **Repositories**

This folder contains the repository classes. All the business logic is written in these classes.

For Example, Authenticating the user, registering the user, registering the devices, fetching updated data of devices, etc.

1. **Services**

This folder is only present in MyIoTService Project. It contains the Device Integration Service. The DeviceIntegrationService Class is responsible to communicate the IoTDevice with MyIoTService.

# IoTDevice

* The purpose of this project is to simulate the IoT Device.
* There is an SQLite database that holds only one entity for storing the Device.
* The Bogus class is implemented to generate the fake data for Device Sensors. Only the following attributes are generated using faker.
  + InsideTemperature
  + OutsideTemperature
  + WaterTemperature
* There is one additional attribute that I stored in the Device entity which is DeviceStartTime. This attribute help to calculate the OperationTimeInSec and OperationTimeInHours(WorkingHour).
* Following attributes of the device can only be modified externally. Except for these attributes, all other attributes of the class are autogenerated and non-changeable.
  + HasOutsideTemperature
  + IsOperational
  + SilentMode
  + MachineIsBroken
* SqlLite database is used to store every device, because we just need to simulate an IoTdevice. We can also use the SqlServer but the best suitable database for small storage is SqlLite.
* The WebAPIs for IoTdevice are secured using API-KEY based authorization. Anyone can access these devices if they have a valid API-KEY. We just need to secure the devices. We can also apply CORS to limit access but I didn’t implement it because of the local URI. If I run it in my system I will have my localhost IP address and you may have different. So for the sake of ease. I just focused on functionality and understandability rather than security.

# MyIoTService

* This is the main project that contains user registration and IoT device management.
* There is an SQLServer database connection to store the data of EndUser and their IoTDevices.
* The entities are defined in such a way that there can be one or many EndUsers. There also can be one or many Devices. One user can add one or many devices. One device is associated to only one EndUser. The Erd Diagram is attached in section 5.1 of this document.
* Only Authorized users can access the resources of MyIoTService.
* There is an option for user to self-register into MyIoTService my providing FirstName, LastName, Username, and Password.
* APIs will automatically get the ContextUser and will store the user Id along with the device.
* APIs will only provide the User with the device data that belongs to that specific user. There is no other way to access or see other users’ devices.
* The API Endpoints are secured by implementing JWT Authorization. One user will provide the credentials to get a JWT. Which will help the user to get access to its resources. JWT token will expire after 7 days of issuance.

## Diagram Description automatically generatedERD Diagram

# Running the Project

Since these are WebApi projects so we need a proper setting for the deployment of these applications. But for the sake of ease, I provide you with a simple mechanism to run the project and check the functionality.

1. Open this solution file in two different visual studio instances.
2. First run the IoTDevice solution in one instance.
3. Second run the MyIoTService solution in another instance of visual studio.
   1. In AppSettings.Json file in this solution there will be a connectionstring for sqlserver.

"ConnectionStrings": {

"SqlServerConnectionString": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=MyIoTServiceDB;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

}

* 1. Change the Data Source and Initial Catalog according to your server configurations.

1. Make sure both projects have been started and the first page that will open in the browser will be the Swagger UI Page for both projects.
2. Since these APIs are secured with authorization, so, first click Authorize Button in IoTDevice Swagger page and provide the API-Key. API-Key will be written in the Authorize dialogue box, or you can take it AppSettings.Json file inside the IoTDevice Project.
3. Secondly, Authorize the MyIoTService. Execute the Authenticate Endpoint by providing the username and password, by default there is one user registered having “username”:”admin” and “password”:”admin”. You can use these credentials. In response you will get the user with the JWTToken. Copy the Token value and place in the textbox inside the Authorize dialogue box. This dialogue box will open when you click the Authorize Button.
4. After Authorizing the data you can play with the interface as you wish.
5. If a user registers a new device It will save in the IoTDevice as well as MyIoTService Database. Every operation you will apply will affect both the records simultaneously.
6. If a user registers a device. And wants to change the data. It will update on both sides.
7. If a user deletes it will delete from both sides.
8. If a user gets a device, it will give the user the locally stored data. If he wants to get the latest device data. Then the users need to run the fetch current data API. This API will update the local device data with the remote device data.

# References

**Git Repository:**

This repository is private and can be given access on demand. If you want to make it public or if you want to add a contributor to this repository. You can further communicate with me via email. Thanks

[cybercommando/Proekspert-MyIoTService (github.com)](https://github.com/cybercommando/Proekspert-MyIoTService)

git clone <https://github.com/cybercommando/Proekspert-MyIoTService.git>