

ALTEN Vehicle Monitor

Table of Content

[Sysetm Architecture 3](#_Toc533008523)

[Solution Design And Analysis 3](#_Toc533008524)

[How to Run the Solution 4](#_Toc533008525)

[Data in-memory representation 4](#_Toc533008526)

[Unit Testing and Integration Testing 4](#_Toc533008527)

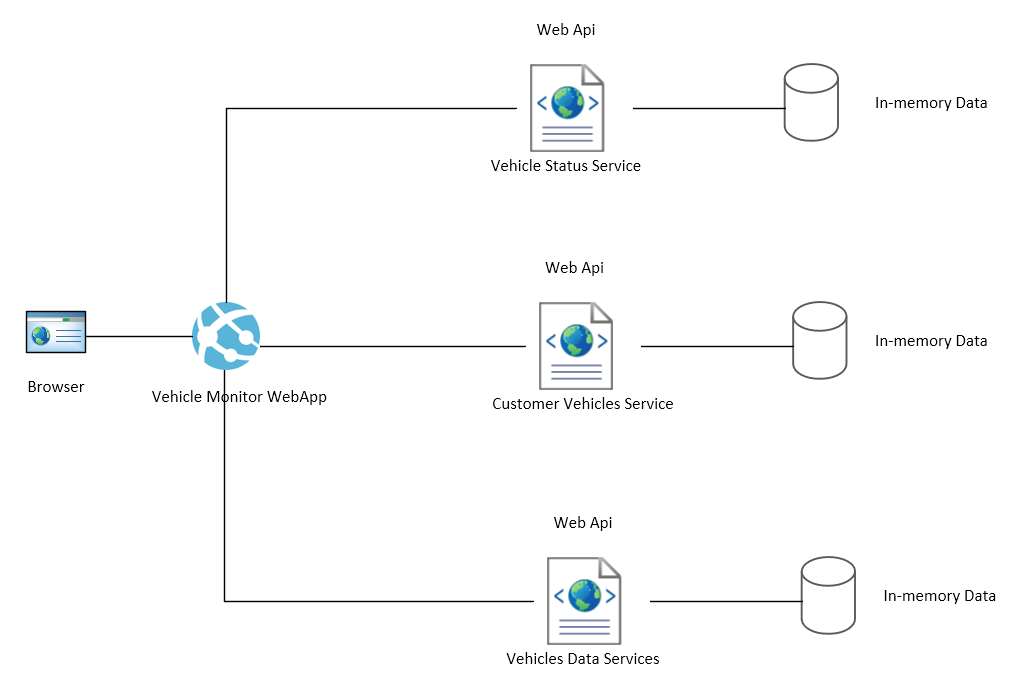
[Language Used 4](#_Toc533008528)

[Continuous Integration 4](#_Toc533008529)

[Microservices architecture 4](#_Toc533008530)

[Cloud Platform Used 4](#_Toc533008531)

# Sysetm Architecture



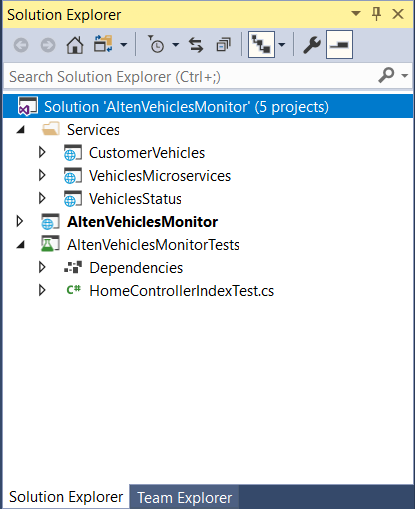
# Solution Design And Analysis

The solution has been designed with **.Net Core 2.1** and **Visual Studio 2017,** and consist of:

* **1 ASP.NET Core Web Application (MVC)**
  + AltenVehiclesMonitor**:** Represents the web front application to monitor all the vehicles data and status

1. One controller {*HomeController*}
2. One View {*Index*}
3. RestApiCaller: which call the microservices

* **3 ASP.NET Core Web Api**
  + CustomerVehicles: Represents the service responsible for retrieving specific customer’s vehiclesapi/customervehicles/{Customer Id}
  + VehiclesStatus: Represents the service responsible for retrieving Vehicle Status (Random Status every time)/api/vehiclestatus/{Vehicle Id}
  + VehiclesMicroservice: Represents the service responsible for retrieving all Vehicles with their customer dataapi/vehicles
* **1 xUnit Test Project**
  + AltenVehiclesMonitorTests: Responsible for running all test scenarios

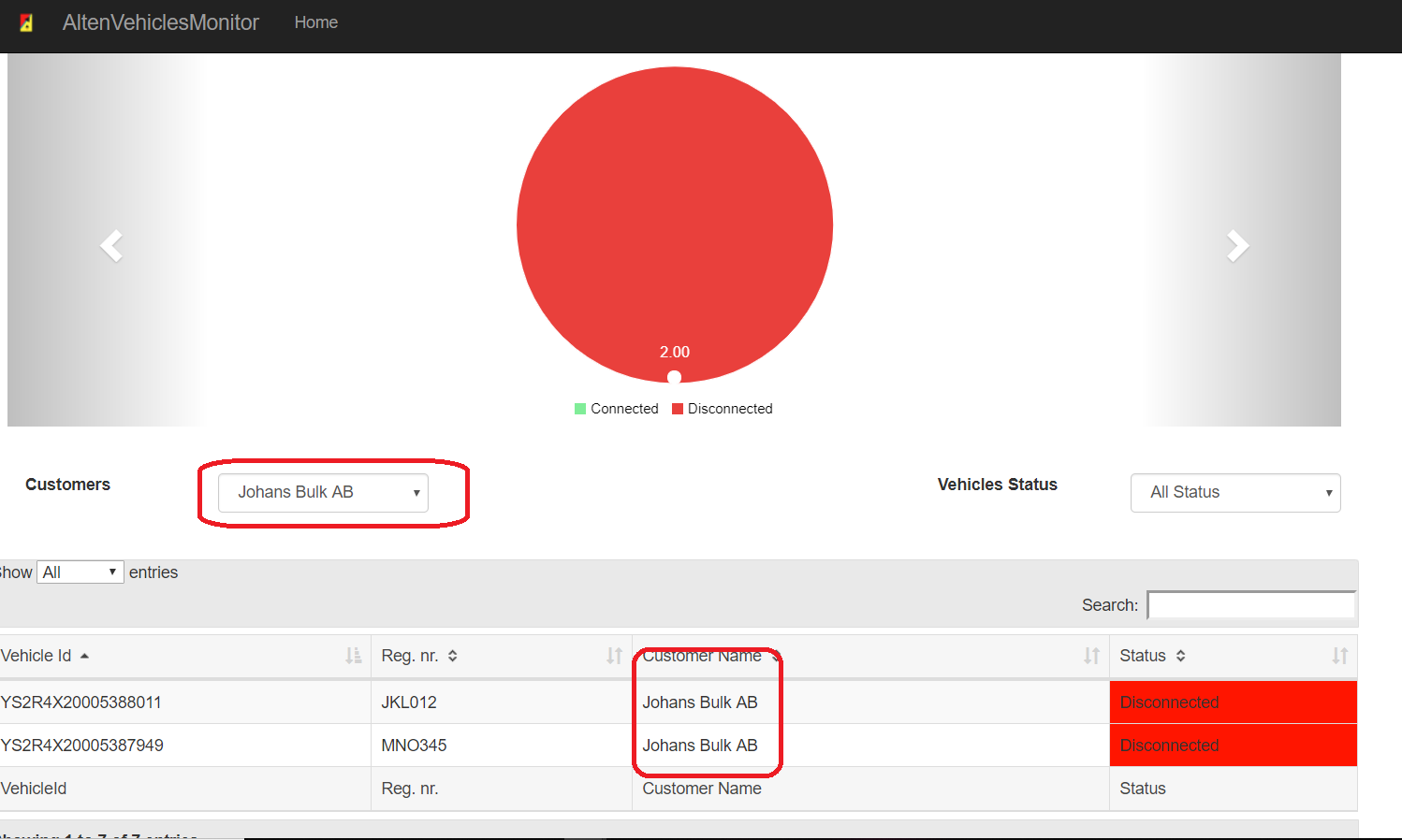


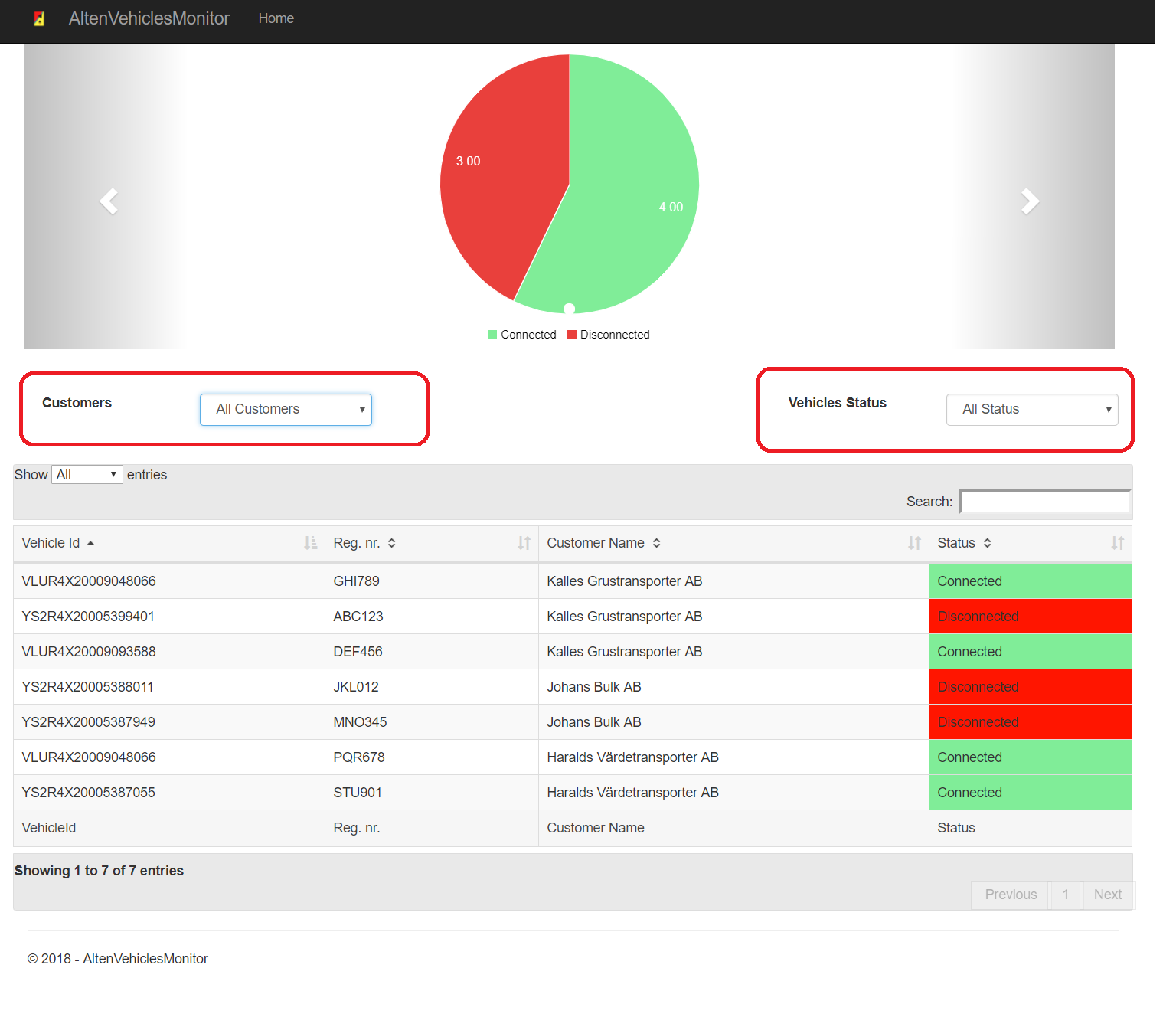
# How to Run the Solution

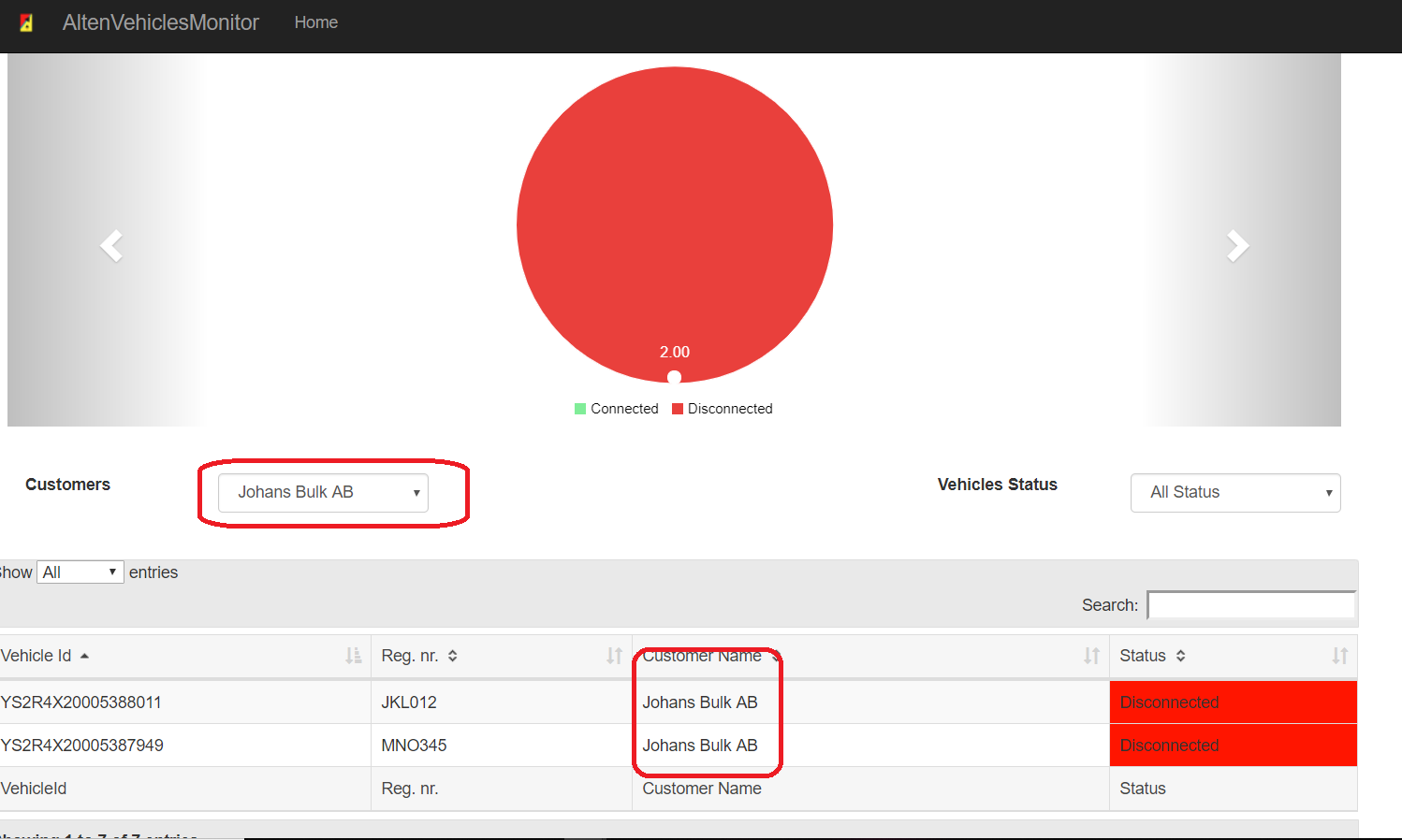
The Main Project is **AltenVehiclesMonitor** has one view page which displays all vehicles data and update their status every minute

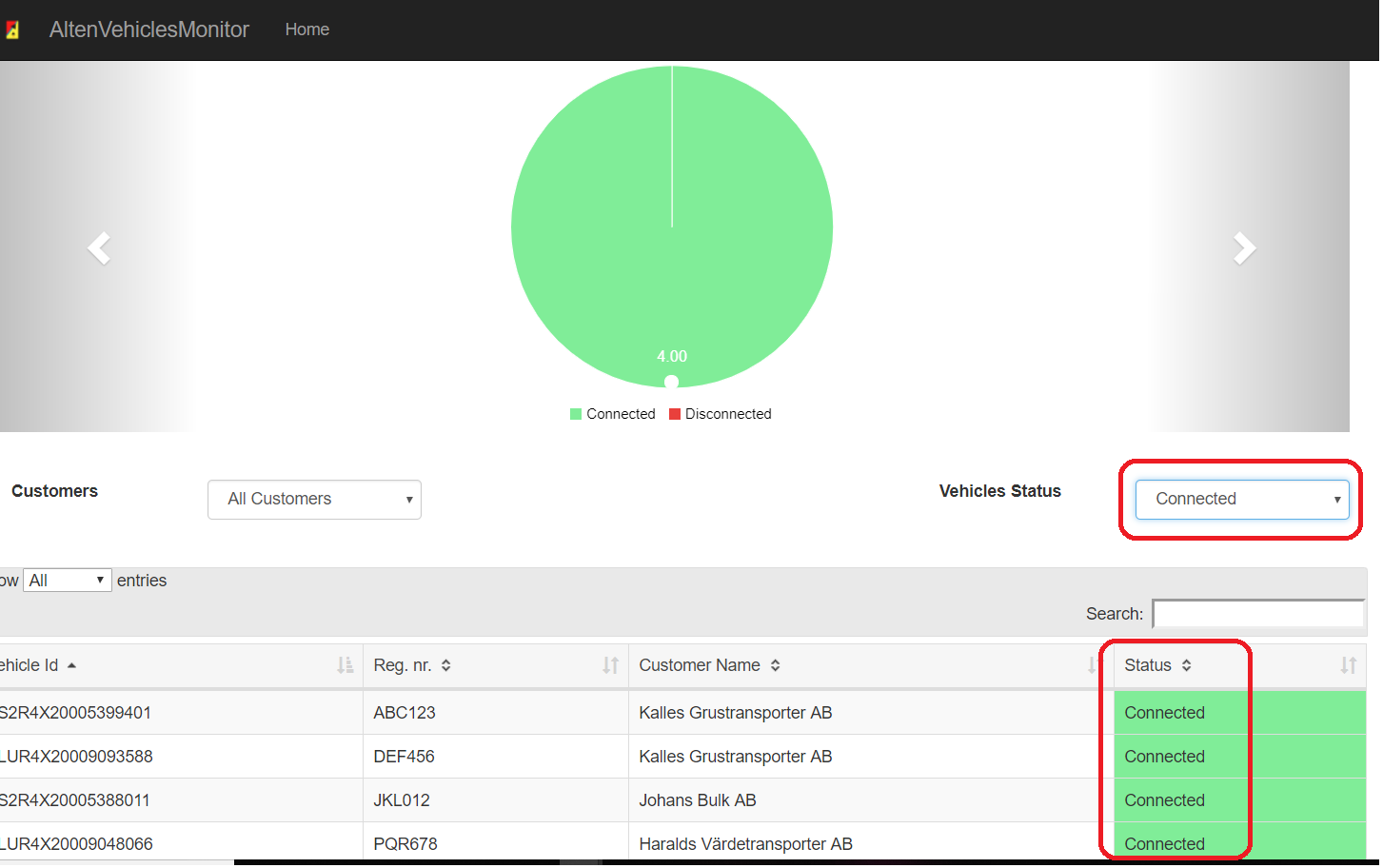
Just Run this project and it will open and displays the main dashboard for the vehicles data and status

You can filter by the vehicles either by customer or by vehicles status from the dropdown filters in the main page



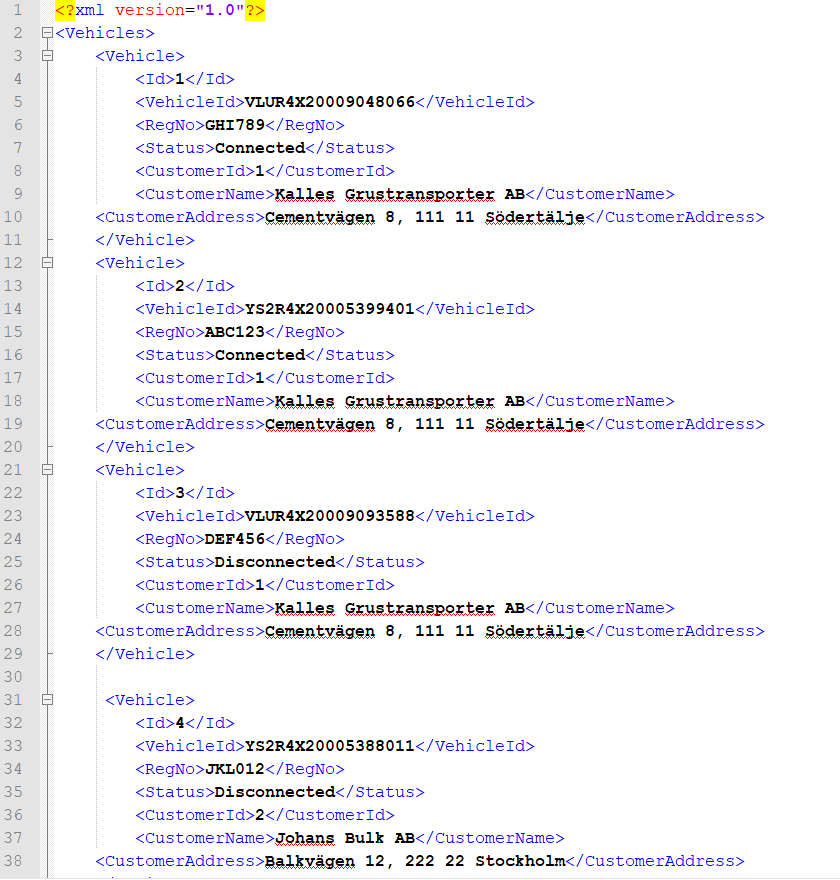


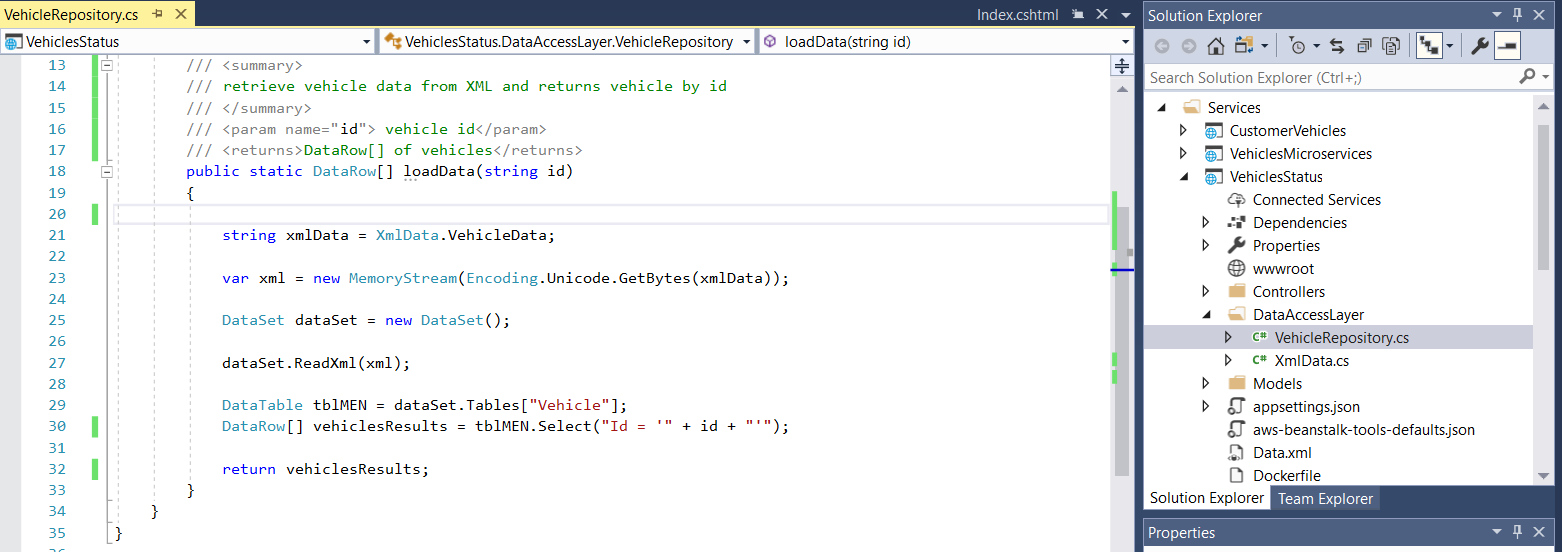




# Data in-memory representation

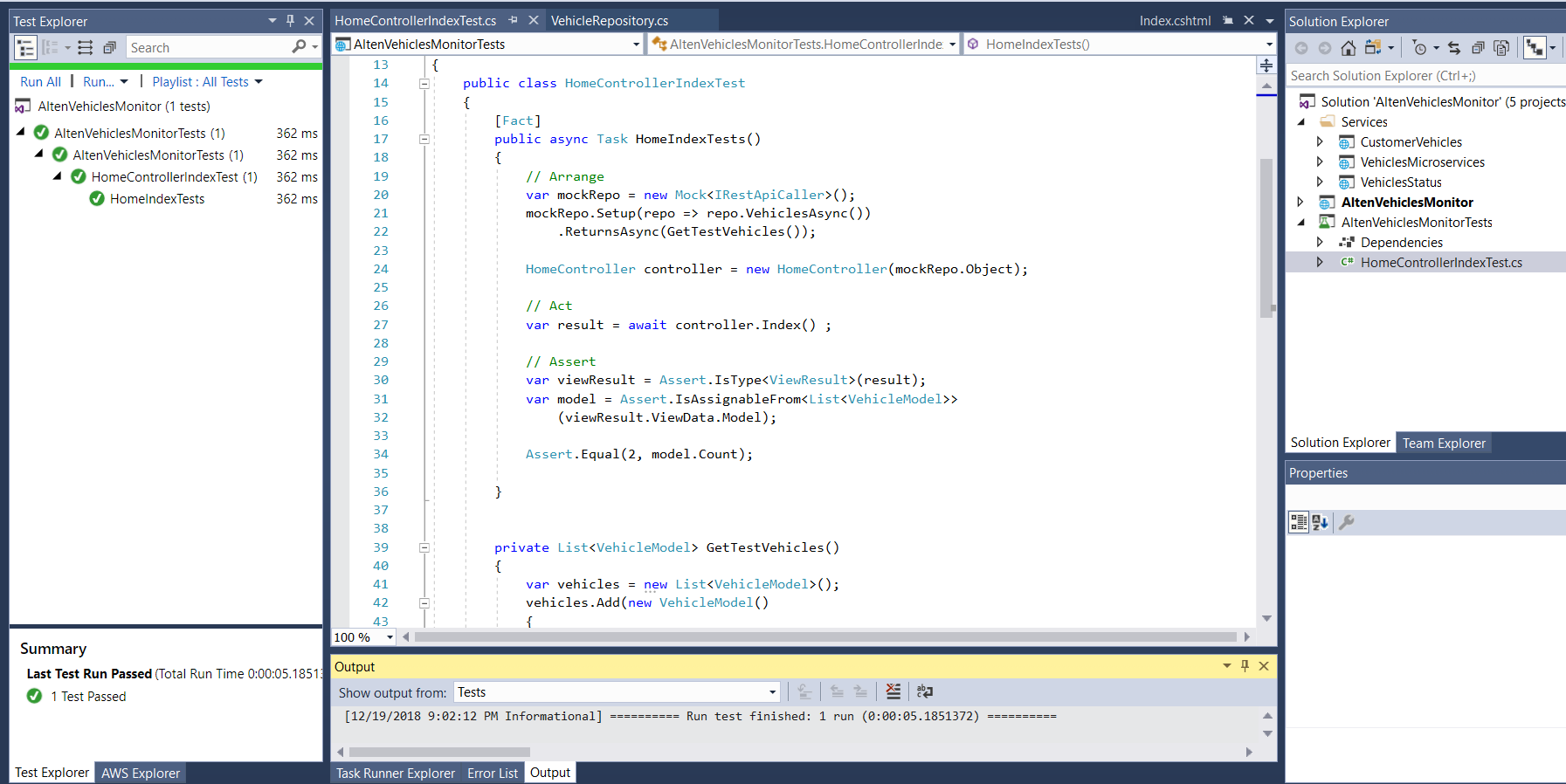
I have used data in-memory with ***XML*** to represent vehicles data and then converts it to normal ***Dataset***

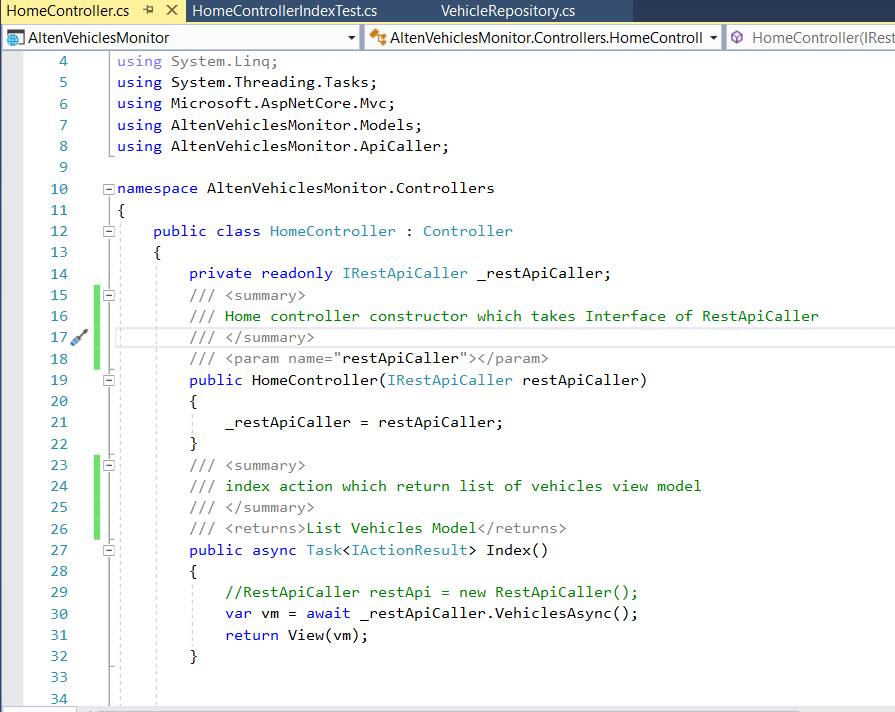




# Unit Testing and Integration Testing

I have used xUnit test project ***AltenVehiclesMonitorTests*** which Responsible for running all test scenarios

And by using ***Dependency Injection in HomeController***

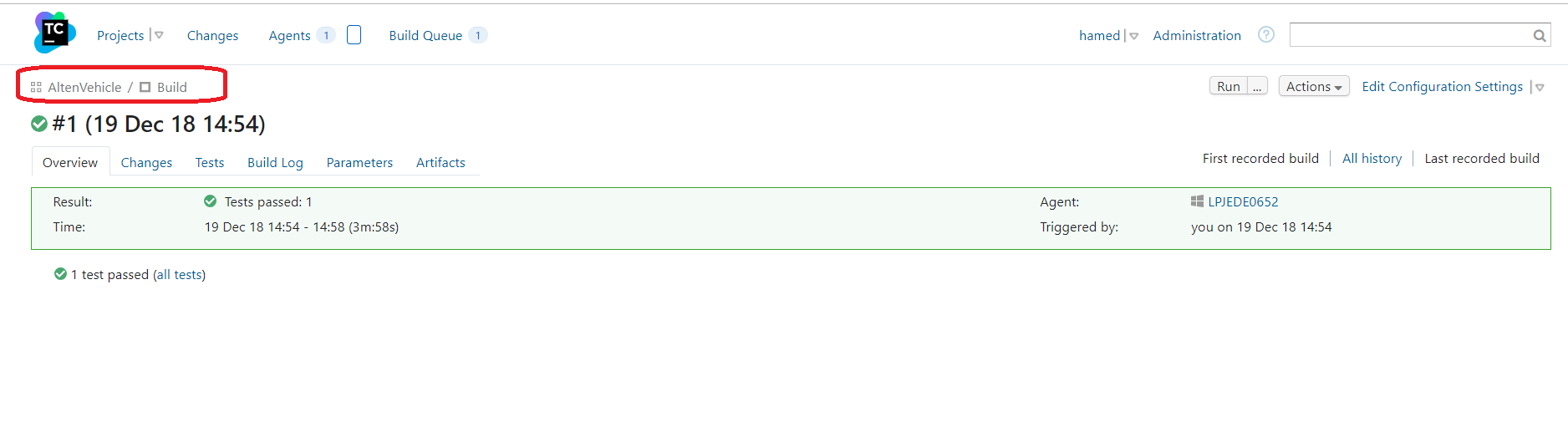


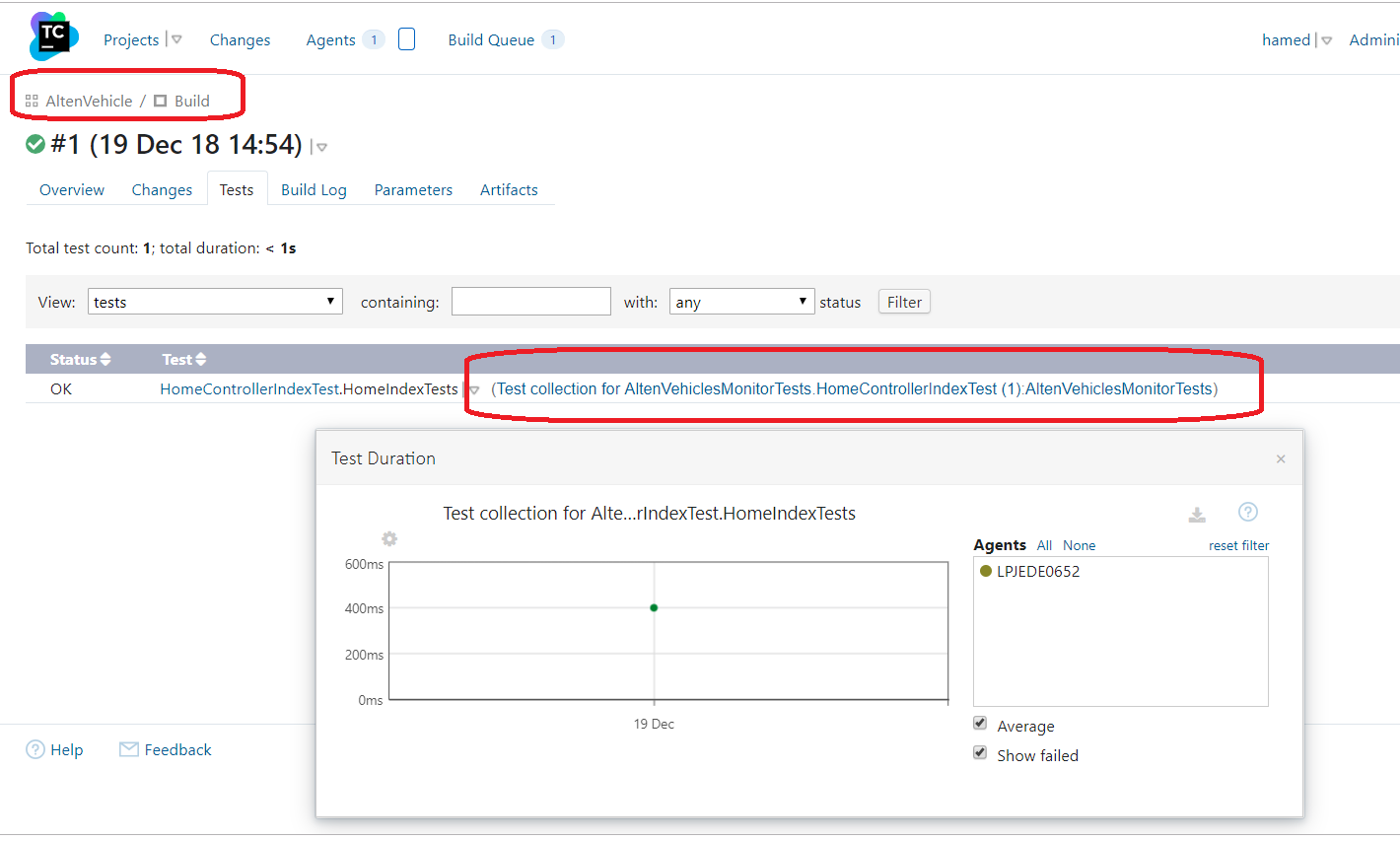
# Language Used

**I have used .Net Core 2.1** **with** **Visual Studio 2017**

# Continuous Integration

I have used ***TeamCity*** for CI and tests the project

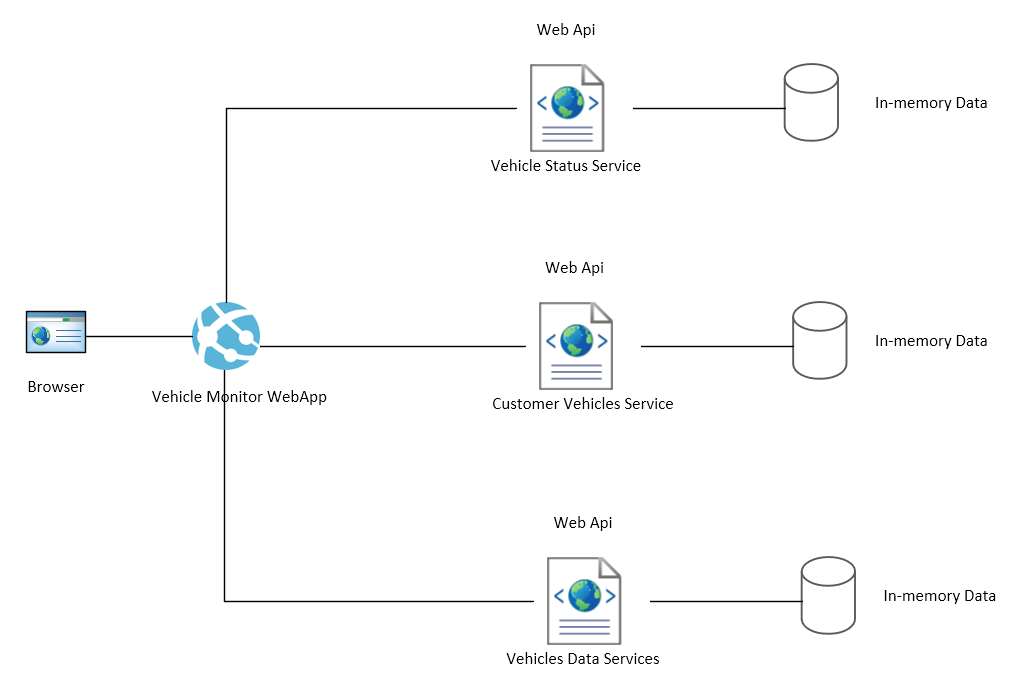




# Microservices architecture

I have used 3 loosely coupled Web Api as microservices each one has its own vehicles data

* **3 ASP.NET Core Web Api**
  + CustomerVehicles: Represents the service responsible for retrieving specific customer’s vehiclesapi/customervehicles/{Customer Id}
  + VehiclesStatus: Represents the service responsible for retrieving Vehicle Status (Random Status every time)/api/vehiclestatus/{Vehicle Id}
  + VehiclesMicroservice: Represents the service responsible for retrieving all Vehicles with their customer dataapi/vehicles



# Cloud Platform Used

i have used AWS as cloud platform for hosting the Web App and for the microservices

Main Web App:

<http://altenvehiclesmonitor-dev.us-east-1.elasticbeanstalk.com>

Vehicle Status Service:

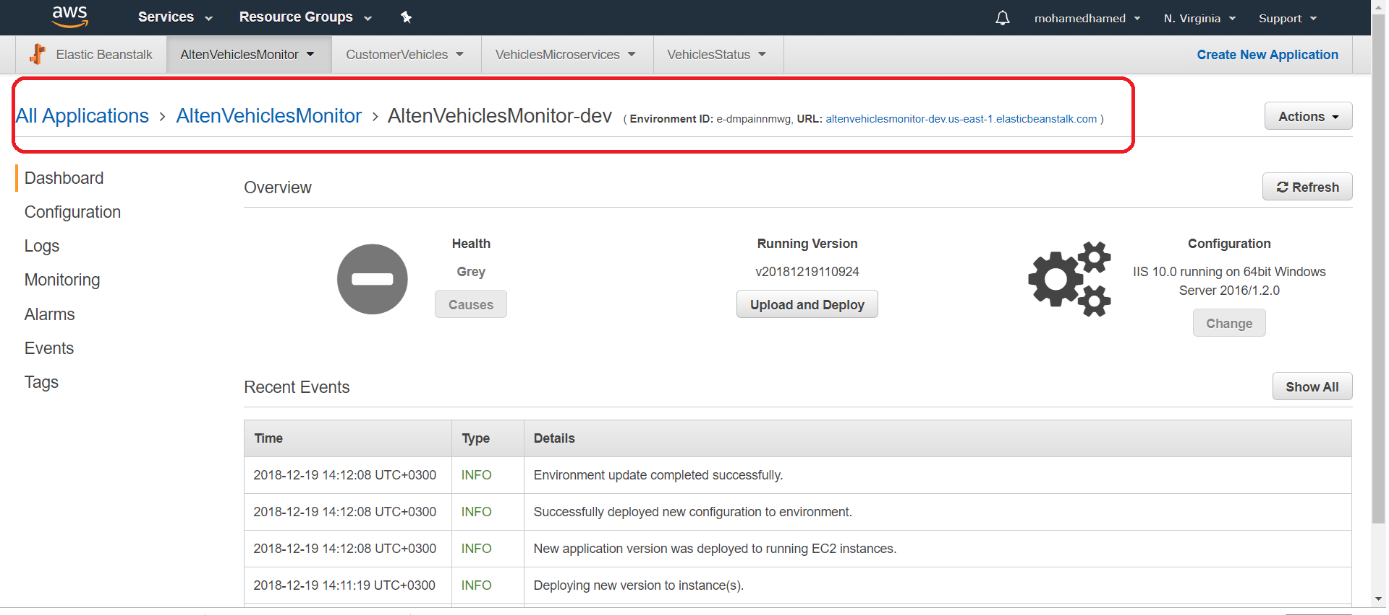
<http://vehiclestatus-dev.us-east-1.elasticbeanstalk.com/api/VehiclesStatus/2>

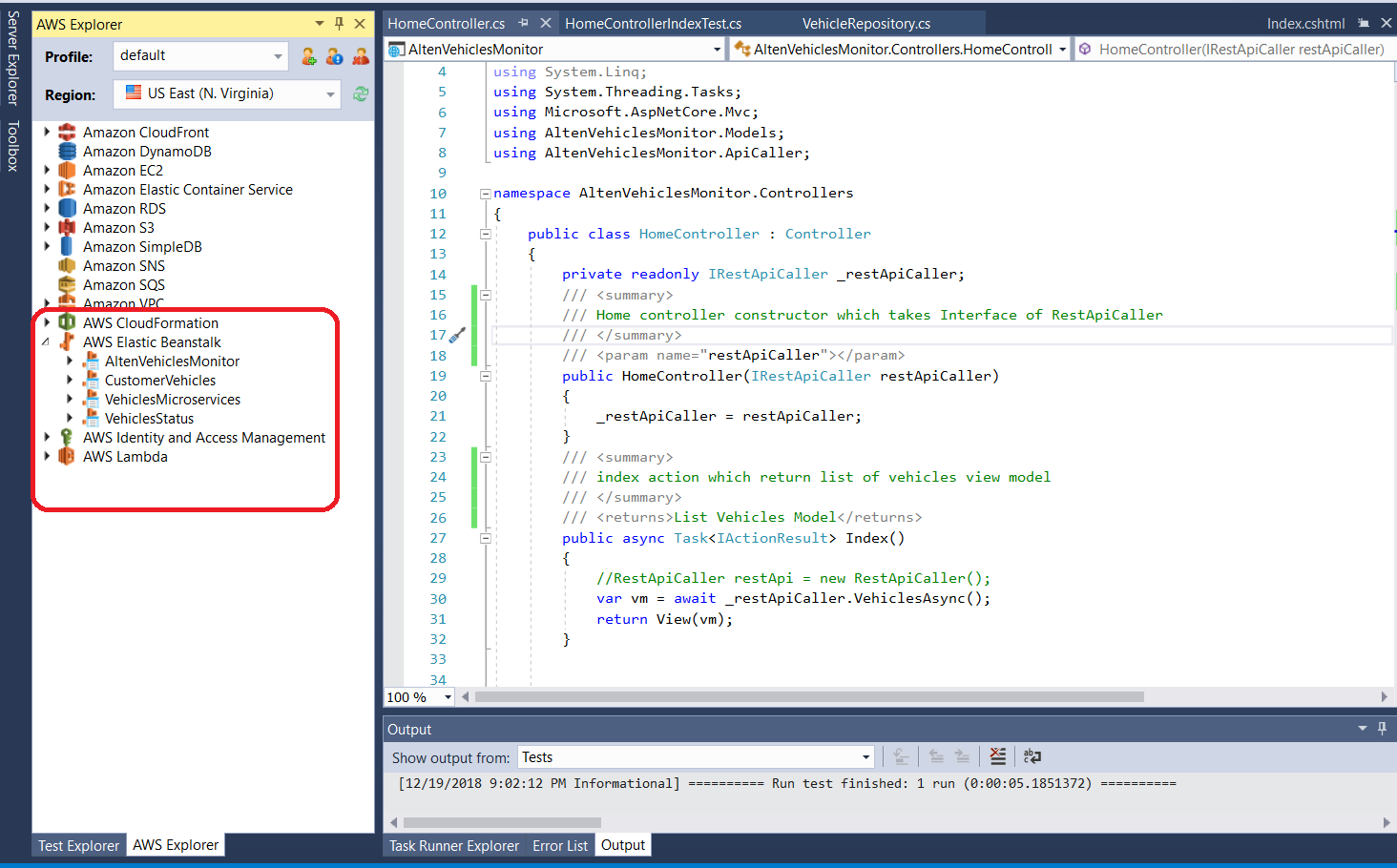
Customer Vehicles Service:

<http://customervehicles-dev.us-east-1.elasticbeanstalk.com/api/customervehicles/2>

Vehicles Microservice:

<http://vehiclesmicroservices-dev.us-east-1.elasticbeanstalk.com/api/vehicles>





# Serverless architecture

I believe that it is possible for this application to be in Serverless architecture by migrating all its services into FaaS like AWS Lambda Function or MS Azure Functions and by doing this we will decrease the cost as we only will pay by function calls