Vehicle Monitoring Solution

# Solution Overview

The customer has a number of connected vehicles that belongs to a number of customers. They have a need to be able to view the status of the connection among these vehicles on a monitoring display.

The vehicles send the status of the connection one time per minute. The status can be compared with a ping (network trace); no request from the vehicle means no connection.

# Requirements

1. Web GUI (Single Page Application Framework/Platform).

* An overview of all vehicles should be visible on one page (full-screen display), together with their status.
* It should be able to filter, to only show vehicles for a specific customer.
* It should be able to filter, to only show vehicles that have a specific status.

1. Random simulation to vehicles status sending.
2. If database design will consume a lot of time, use data in-memory representation.
3. Unit Testing.
4. .NET Core

# The Solution Scenarios

The vehicle update status scenario

1. the vehicle consume the Gateway api[UpdateVehicleStatus] and pass its VIN as parameter
2. the gateway put the time stamp with current time of the ping then push a message in RabbitMQ queue contain {vin, ping time}
3. the RabbitMQ push the message to the vehicle status management service which update the vehicle last ping time , last ping time used to specify the vehicle status **Connected**: if the difference between the last ping time and the dash board query time greater than the no of ticks, as the above example 60 seconds, else the vehicle status will be **disconnected**

The customer vehicle dashboard

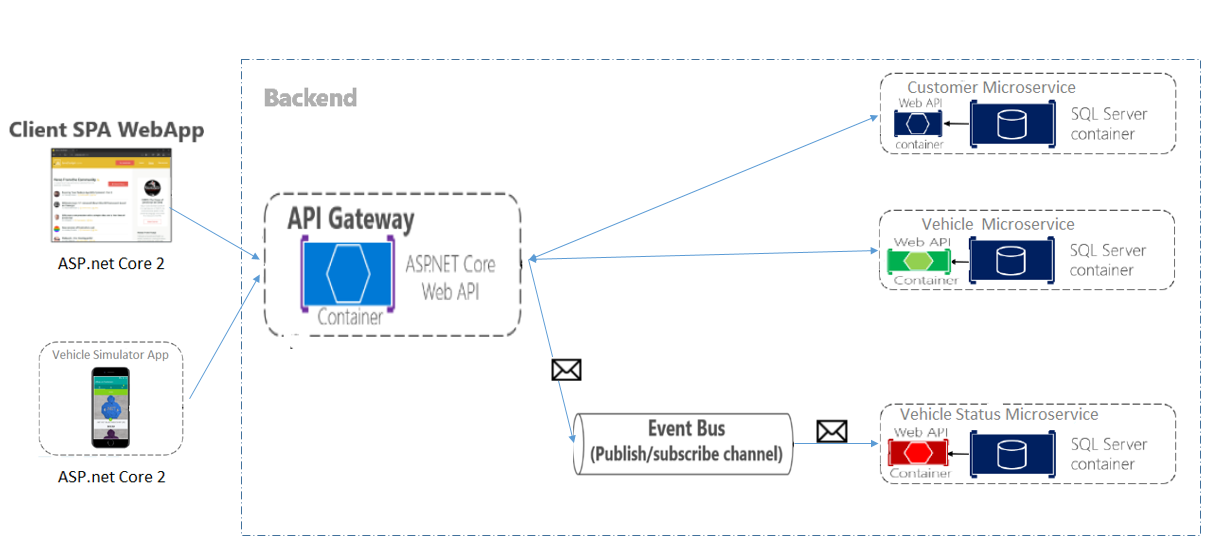
1. the customer select the status from status list which is filled via static values{connected=true/disconnected=false/ALL }
2. the customer select the customer from list of customers names filled by consuming the gateway api " GetCustomersLookup" or may select ALL
3. SPA web app consume the gateway api" GetCustomerVehicles" with the selected search criteria {customerid,status}
4. The gateway consume the customer micro service API " GetCustomer"via customer id and if customer id not sent get all customers
5. The gateway make aggregations by looping on the customers and for each customer consume the vehicle micro service " GetCustomerVehicles" by passing customer id and status which select the vehicles via these criteria's noting that the status specified as follow
   1. **Connected**: if the difference between the last ping time and the dash board query time greater than the no of ticks, as the above example 60 seconds, else the vehicle status will be **disconnected**

# Solution Architecture

The solution architecture is based on Micro services architecture and DDD "data domain driven" and using a single custom API Gateway service facing multiple and different client apps

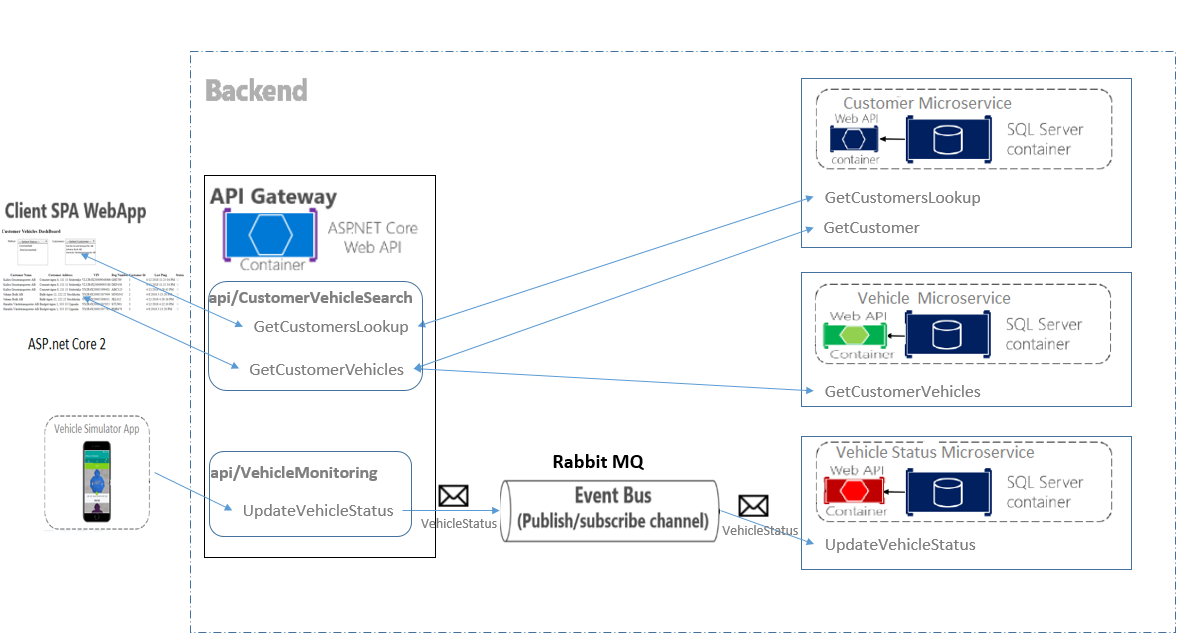
**Note:** for larger scale it's recommended to split the API Gateway in multiple services or multiple smaller API Gateways

The API Gateway implemented as a custom Web API service using aps.net core running as a container and used in aggregations of the solution entities and act as interface layer for the solution entities and operations, solution entities are consumed via backend services only as shown in the figures below.

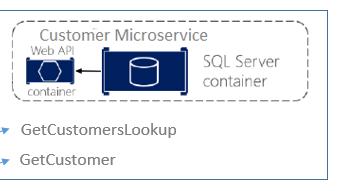
**Note:** we can use this architecture in the cloud by using Azure API management

## The solution entities

The solution entities are customer, vehicle and vehicle status management and each one is handled via a micro service developed via asp.ne core as follow and explained in the below figure



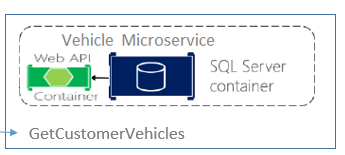
### Customer micro service



Responsible for the customer operations

* Get Customer lookup : used to search the customer entity and return the customers lookup used in customer vehicles dashboard in {id,name}format
* Get Customer : used to search the customers via customer id and if you did not send the customer id it will return all customers

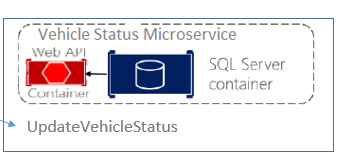
### Vehicle micro service



Responsible for the vehicle operations

* Get Customer Vehicle : used to search the vehicle entity via customer id and status criteria's and if you didn't send the customer id it will be excluded the from the search criteria and return vehicles for all customers and the same for the status if you didn't send status it will be excluded from search criteria and return vehicles with all status{connected/disconnected}

### Vehicle status management service



Responsible for update the vehicle last ping time as the last ping time will be used to specify the vehicle status

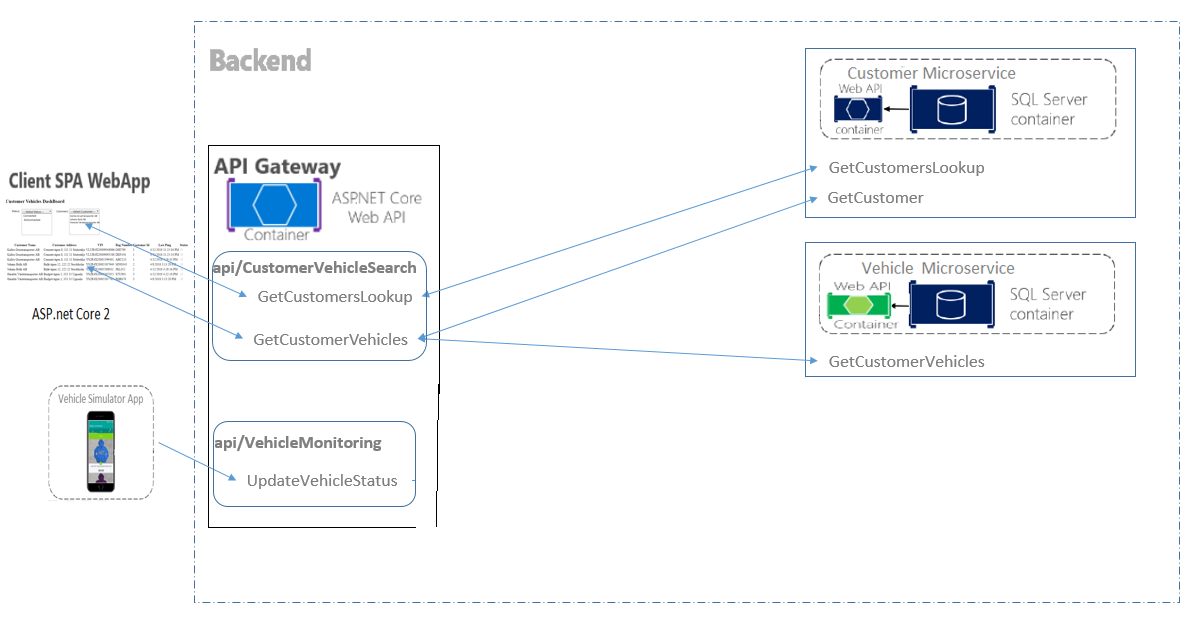
**Connected**: if the difference between the last ping time and the dash board query time greater than the no of ticks, as the above example 60 seconds, else the vehicle status will be **disconnected**

**Note**: the ticks no value is configurable in the service api settings

## The Communication types

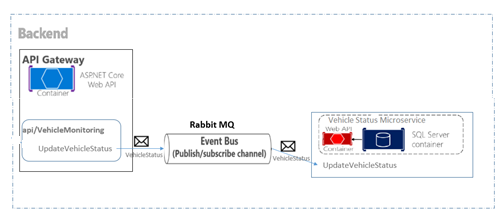
The solution is based on using two types of communications

### Rest based on HTTP Synchronous protocol.

REST based on HTTP Synchronous protocol The client sends a json request and waits for a response in json format from the Rest API. This method used via

1. Gateway to consume the customer micro services
2. Gateway to consume the vehicle micro services
3. Client spa web app to consume the gateway api to build the dashboard "search criteria and vehicle grid"
4. Vehicle simulator web app to consume the gateway api to ping its status

### Asynchronous protocol



Based on **publish/subscribe** , **Event-driven architecture** via using

**AMQP** (a protocol supported by many operating systems and cloud environments) use asynchronous messages. The client code or message sender usually does not wait for a response. It just sends the message as when sending a message to a **RabbitMQ** queue .

This method is used via

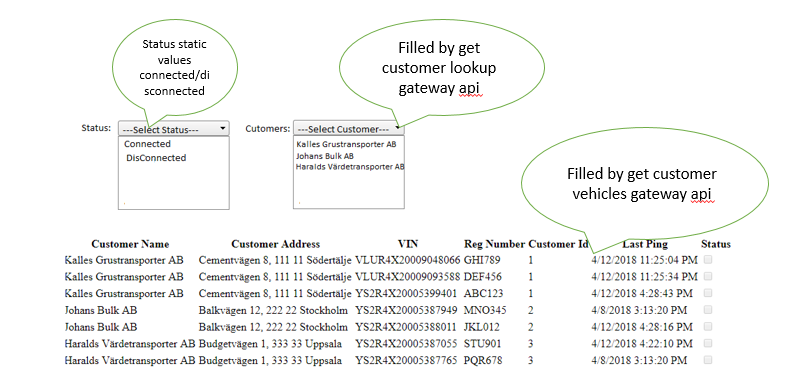
1. Gateway, as publisher, send the ping time of the vehicle to RabbitMQ queue and RabbitMQ deliver it to the vehicle status management micro service ,as subscriber.

**Note**: current solution support single receiver but can be extended to support multiple receivers

## The client apps

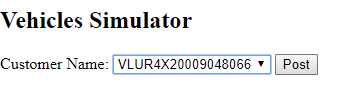
### Single page application SPA Web App

SPA web App is developed via asp.net core provide a dashboard for monitoring the customers vehicles



### Vehicle simulator

It's an asp.net core Web App consume UpdateVehicleStatus in gateway api via sending the VIN of the vehicle



# Used technologies

## Microsoft ASP.net Core 2

Used in development

* Client SPA web app
* Client vehicle simulator
* Gateway API
* Customer API
* Vehicle API
* Status management API

As .net core provide the below features

* Cross-platform implementation.
* The solution based on microservices.
* The solution is using Docker containers.
* To provide high-performance and scalable solution.
* To provide side-by-side .NET versions per application.

## Microsoft Entity Framework Core 2

Used in development the repository/unit-of-work pattern to provide

* Isolate the database code and adapt with different types of DBMS
* Aggregation functions between the solution entities As the solution based on DDD "data domain driven"
* Used with Moq in unit testing
* Hide the complex T-SQL

## RabbitMQ 3.7

To provide the main functionality required for pub/sub arch

## SQL Server 2016

Used as data base for the solution entities

**Note**: as the solution based on DDD and the micro services could have independent data stores like customer micro service need only customer table and vehicle need only vehicle table we can replace it by **SQL Express** or **SQLite**

## Swagger

To provide the below

* interactive API documentation
* **Ability for other products to automatically consume and integrate your APIs**.

## Docker

to create, deploy, and run applications by using containers and provide a cross platform deployment for the solution as a part of CD/CD

## Moq in unit testing

Provide automated unit testing cases as a part of CD/CD

# The Cloud implementation

The solution is based on the .net core and micro services architecture so it can be easily to extend and used the cloud

### Azure

we can use

* the Azure API management
* Azure SQL DB as a service

### AWS

we can use

* AWS Lambda to publish all the solution components
* Use AWS RDS for DB using SQL server or any RDBMS