About Myself

# My career & technologies

More than seventeen years working as a web developer and architecting .Net solutions, always making sure that SOLID principles are applied using C#, Entity Framework, Web API, SQL Server, JavaScript and others cutting-edge technologies. Considering quality and performance, I am used to apply DDD, Tests and being an Agile enthusiast.

# Companies Sumary

* FENABRAVE / FIESP
* HOSPITAL ALBERT EINSTEIN
* BANCO REAL / CAIXA ECONOMICA
* PEPSI
* SKY
* AZUL AIRLINES
* SERASA EXPERIAN
* MARISA
* AVON
* FARFETCH

# Career Growth TimeLine

The Challenge

There is a legacy scooter rental system that needs to be scalable to support the business growth.

### Challenge architectural statements:

* Implement micro-services architecture
* Guarantee that all communication between the services be resilient
* Use container
* Must be scalable to support the business network initiatives

# Technical statements

### Main Problems Definition:

* Hard to maintain the code and to develop new features.
* Too much effort and expensive costs to create & scale Windows Virtual Machine (VM)
* The database is not supporting the data volume without lost performance
* Bad user experience
* Monolith does not allow scale the system per resource

### Proposal Solution:

The proposal includes to migrate the currently technology for:

|  |  |
| --- | --- |
| LEGACY | NEW |
| .net Framework 4.6.1 | Microsoft .net Core 3.1 |
| Monolith MVC on IIS | Micro-services using Docker & Kubertes |
| Razor | Angular 2+ |
| SQLite | Microsoft SQLServer Azure with Dapper |
| Virtual Machine Host | Windows Azure |
| Directly Controller Call | WebAPI & Service BUS Azure |

### Why use these technologies?

* Microsoft .net Core
* Improved Performance
* Cross Platform – Able to run easy in cloud environments
* Open Source community support
* Support built in Dependency Injection
* Dapper
  + It is a micro ORM that makes the performance higher
* Docker
  + Docker enables more efficient use of system resources. ...
  + Docker enables faster software delivery cycles. ...
  + Docker enables application portability. ...
  + Docker shines for micro-services architecture.
* Kubernetes
  + is a portable, extensible, open-source platform for managing containerized workloads and services. It has a large, rapidly growing ecosystem, turning the application widely available.
* Angular
  + It is a modern UI, attends the performance needs
  + Uses Type Script, a typed JavaScript easy to maintain the code
* Microsoft SQL Server Azure
  + There are many options as Oracle, MySQL, etc
  + SQLServer is a Microsoft database on azure and it attends the performance needs
* Windows Azure
  + There are many cloud platforms to choose as AWS, Google Cloud, etc.
  + The cloud environment allows us to manage our resource easy, paying just when It is in use.
* Service BUS Azure
  + There is an excellent management of dead letters and panel
  + It is a queue to communicate between the domains
  + Maintain each domain isolated
  + Guarantee that the micro-services down, it will wait to process the information later

### Way to be followed in architecture:

* Domain Driven Design (DDD)

It is a group of techniques to make your code near from the business language, broken it in specific domains, making possible to identify boundaries and micro-services

* Domain Events

Some action will have a reaction to do something related from it

* Test Driven Development (TDD)

Guarantee the application test coverage before developing a new feature

* Azure Service Bus as queue

In case of unavailable service, the queue will guarantee the communication, keeping the micro-services working independently

The Azure Service Bus has a dead letter and retry schemas in case of exception on service before Pop.

* Azure Functions as Job

In this case will be executed in time to time using CRON expression

### What is the main benefits of this approach:

* Easy to scale to support an elevate web traffic and data
* Easier to increase new features on code
* Multiplatform and scalable system
* Increase of web performance and user experience
* Lower cost using the pay-per-use on cloud
* Easy to identify and build isolated services

# Business Scenario

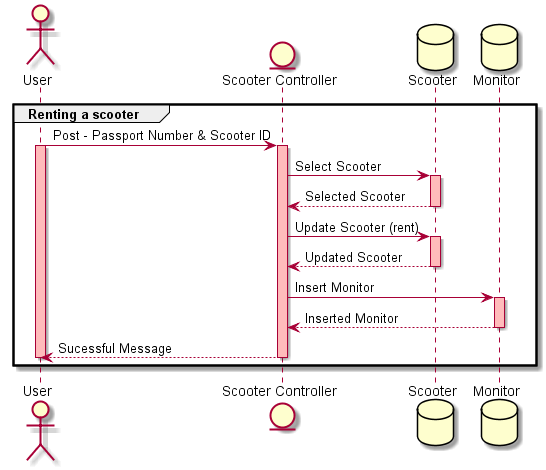
*\* The current business rules will be maintained*

### Current Problems Definition:

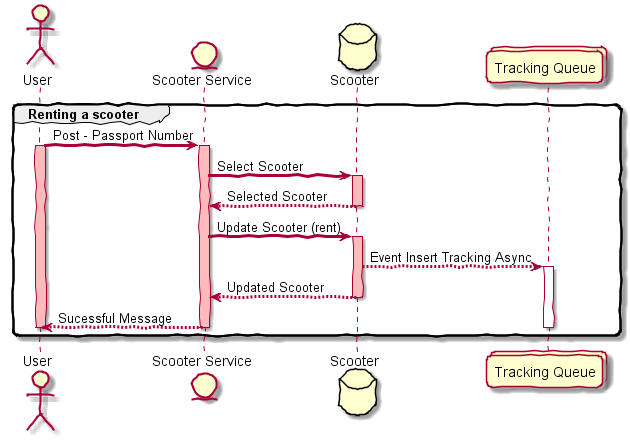
* It is expensive to the company apply new features.
* The company realize that is too expensive and needs a specialist to scale the current Virtual Machine schema.
* The database is going down and degrading performance for each day.
* There are a lot of user calls that don’t know how to use the system.
* They realized that the most required and slower functionality is about monitoring scooter

### Renting Scooters

Legacy

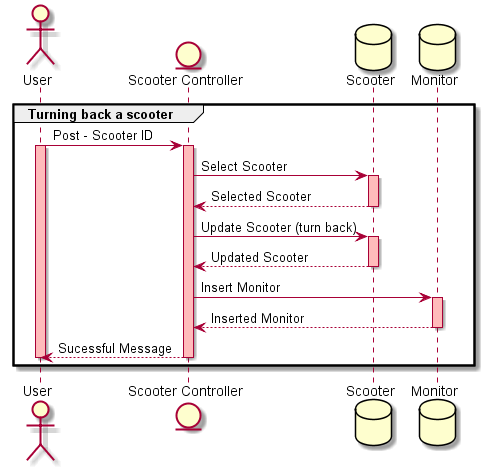


Proposal

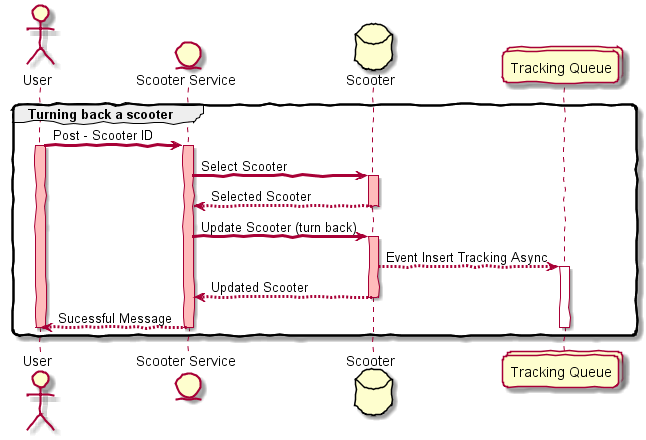


### Turning Back Scooters

Legacy

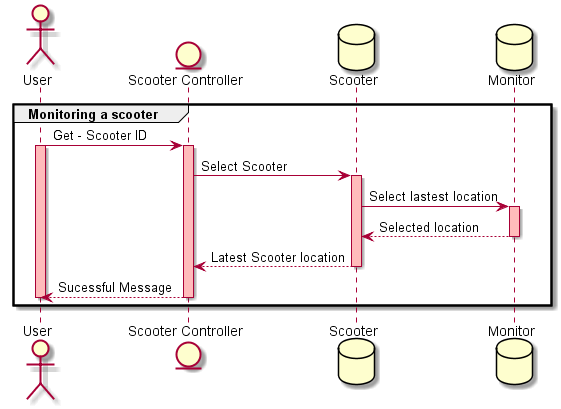
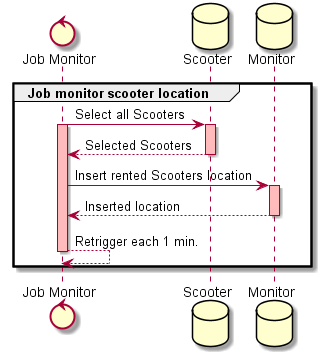


Proposal

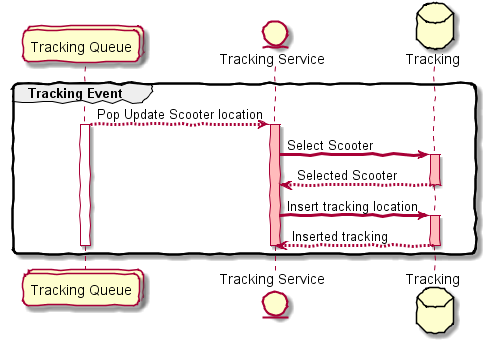
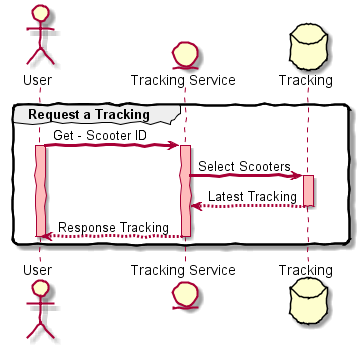
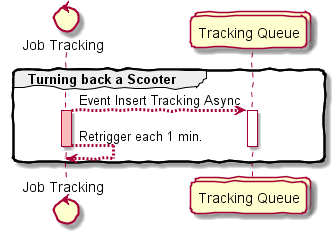


### Tracking scooters

Legacy



Proposal



# Technical Scenario

* Each domain will work independently
* The relation between the domains will be the queue, working asynchronous as well its invokes

### Identified Domains

These identified domain will become micro-services making possible to scale it on the cloud

* Scooter
* Tracking

# Proof of Concepts

### Proposal

<https://github.com/andreraica/ScooterRental>

### Legacy

<https://github.com/andreraica/ScooterRental/tree/master/legacy>