OQPY

Coding sans: Petra Gregurović

Josip Paulik

Nikola Klemeš

Branimir Ričko

# Intro

This paper will briefly explain technical side of an application OQPY. It will explain how it's build, why is built like it's build, technical challenges, current state, planed features.

# High level overview

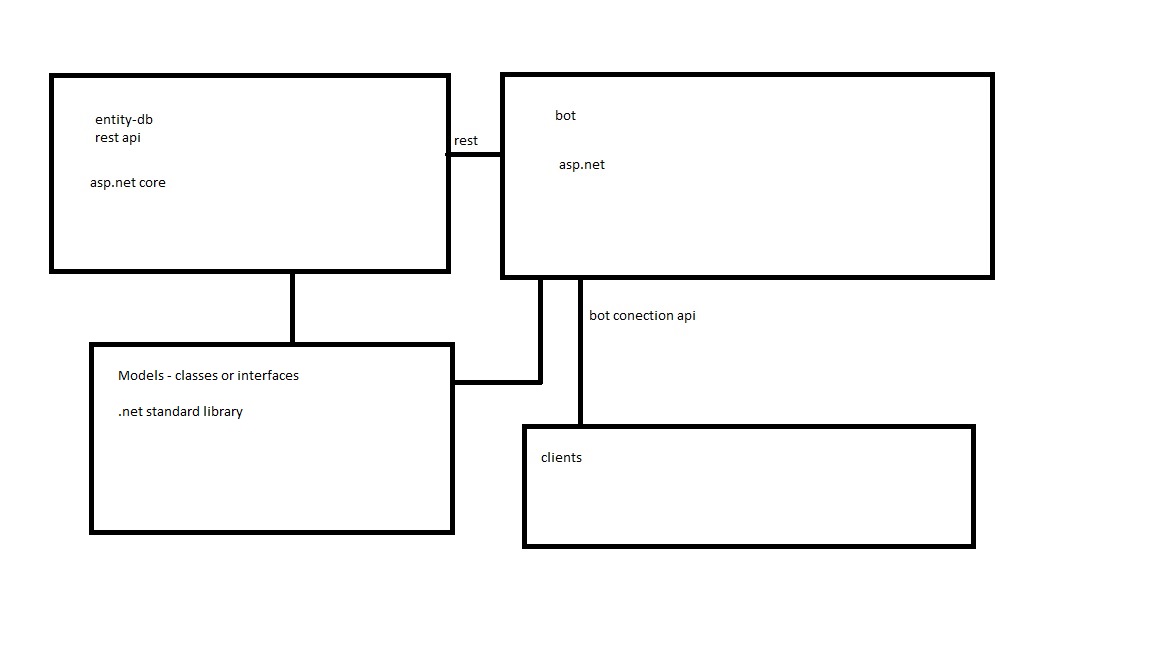


Image 1

In these there are two main projects:

* Database project with:
  + OQPYManager
  + Asp.net core 1.1
  + SQL db
  + Entity core framework
  + Rest api
* Bot project with:
  + OQPYBot
  + Asp.net (not core, because bot framework is not supported in .net core )

Later, in the projects life time we are planning to make an app for business owners and employees targeting all the major platforms for quicker access to our services. To achieve this goal, we'll be using Xamarin.Forms, because it's easy to create application that targets all the major platforms and it's free and open source these days.

For IDE, we are using Visual Studio 2017, because it's the best c# IDE and it's free for students of FER. 😊

For distribution we are using Azure, because it's free for our purposes and it's good enough for us.

# In depth analysis of each project

## OQPYManager

**Purpose**

OQPYManager is a project which serves a purpose of communication with a SQL database, communication with other services, these days only OQPYBot, but we are planning to add more.

## How it's done?

This project is written in [c#](https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx) using [.net core](https://www.microsoft.com/net/core) 1.1, [asp.net core](https://www.asp.net/core) 1.1 and [entity framework core](https://docs.microsoft.com/en-us/ef/core/) 1.1.

For communicating with database, we are using entity framework, because it allows us to write everything in c#, no need for sql scripts.

For communication with other services we are creating [RESTful](https://en.wikipedia.org/wiki/Representational_state_transfer) api, that will allow us to create numerous services that use our api, and late on we can even sell our api to other people to create their own applications using our api. To create RESTful api we are using asp.net core [Controllers](https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/adding-a-controller), because it makes it easy to create apis.

## Current state

This project is going well it’s about 70% done for our first planned version. Database and models are about 90% done, but we need more time to finish all the controllers and Restful api, which is about 40-50% done.

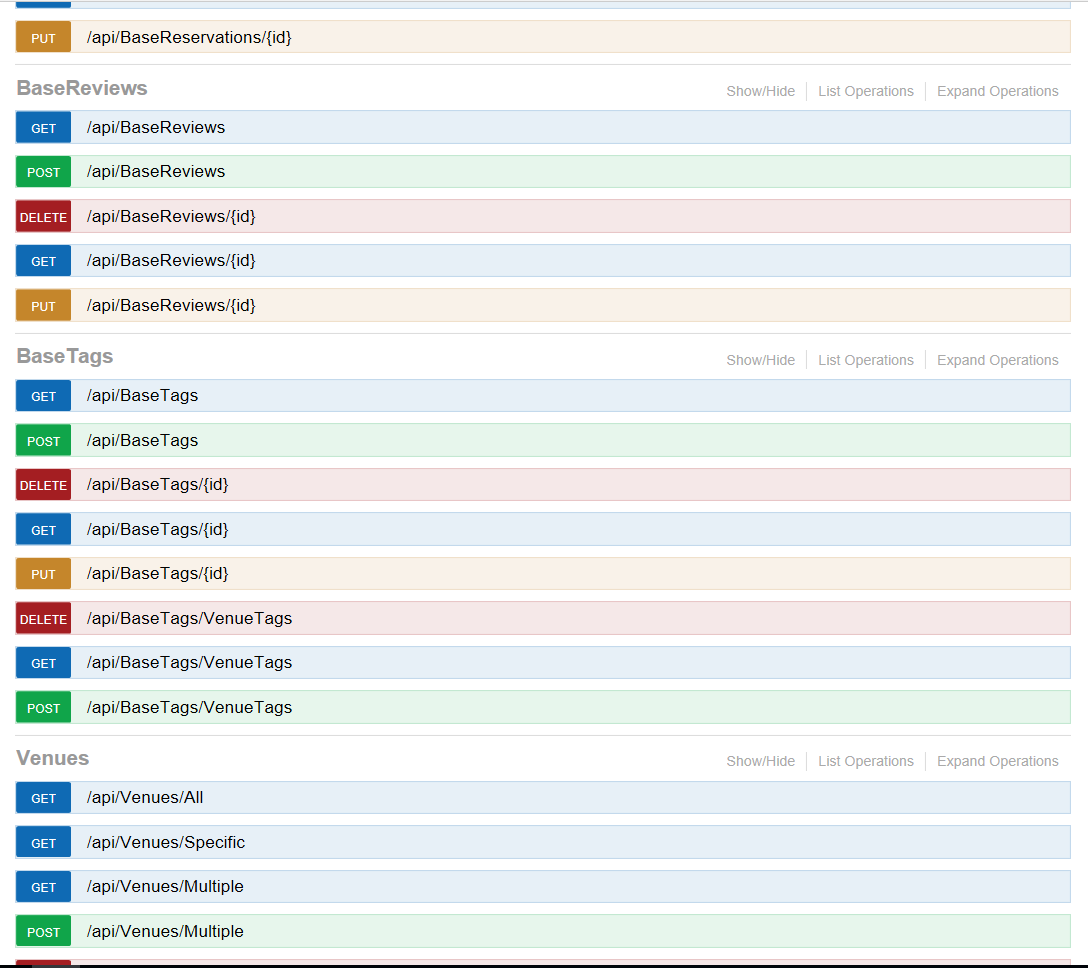


Image 2

Image 2 – some of the Restful apis, displayed using debug [middleware](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware) we are using – [swagger](http://swagger.io/) - [swashbuckle](https://github.com/domaindrivendev/Swashbuckle.AspNetCore) - [curl](https://curl.haxx.se/)

## OQPYBot

## Purpose

OQPYBot is a project which serves a purpose of communicating with the users on various channels (Facebook messenger, skype, slack, …) and it serves as an interface for the user. It communicates with OQPYManager to get and modify data in a OQPYManager database.

## How it’s done?

This project is written in c# using [.net framework 4.6](https://msdn.microsoft.com/en-us/library/w0x726c2(v=vs.110).aspx), [asp.net](https://www.asp.net/) 5.2.3, [bot framework](https://dev.botframework.com/), and [LUIS cognitive service](https://www.luis.ai/home/index).

Full .net framework is used because bot framework is built on top of full asp.net and asp.net is built in .net framework and that represents a challenge, because full .net is not portable and can only run on Windows so we can only run on Windows servers. It represents another challenge, because OQPYManager is built with .net core and that implies that when we write a library that needs to run on manager and on bot, we must use .net standard library, which is a union of subset of .net core and subset of full .net framework, and that implies that we are limited in the number of apis that we can use in building our libraries that need to run on bot and manager. Fix for this obstacle may come in Q2 this year when [.net standard 2.0](https://github.com/dotnet/standard/blob/master/docs/netstandard-20/README.md) ships which promises a lot more apis that can be used.

Full asp.net is used for the same reason the full .net framework is used.

Bot framework it quite a nifty little tool that allows us to easily make a conversational bot, that can pass turing test and the best part is, this service is free 😊

LUIS service is also quite nifty tool that helps us understand what the user wants to do, it’s a joy to use this service, it’s powerful and it’s intuitive.

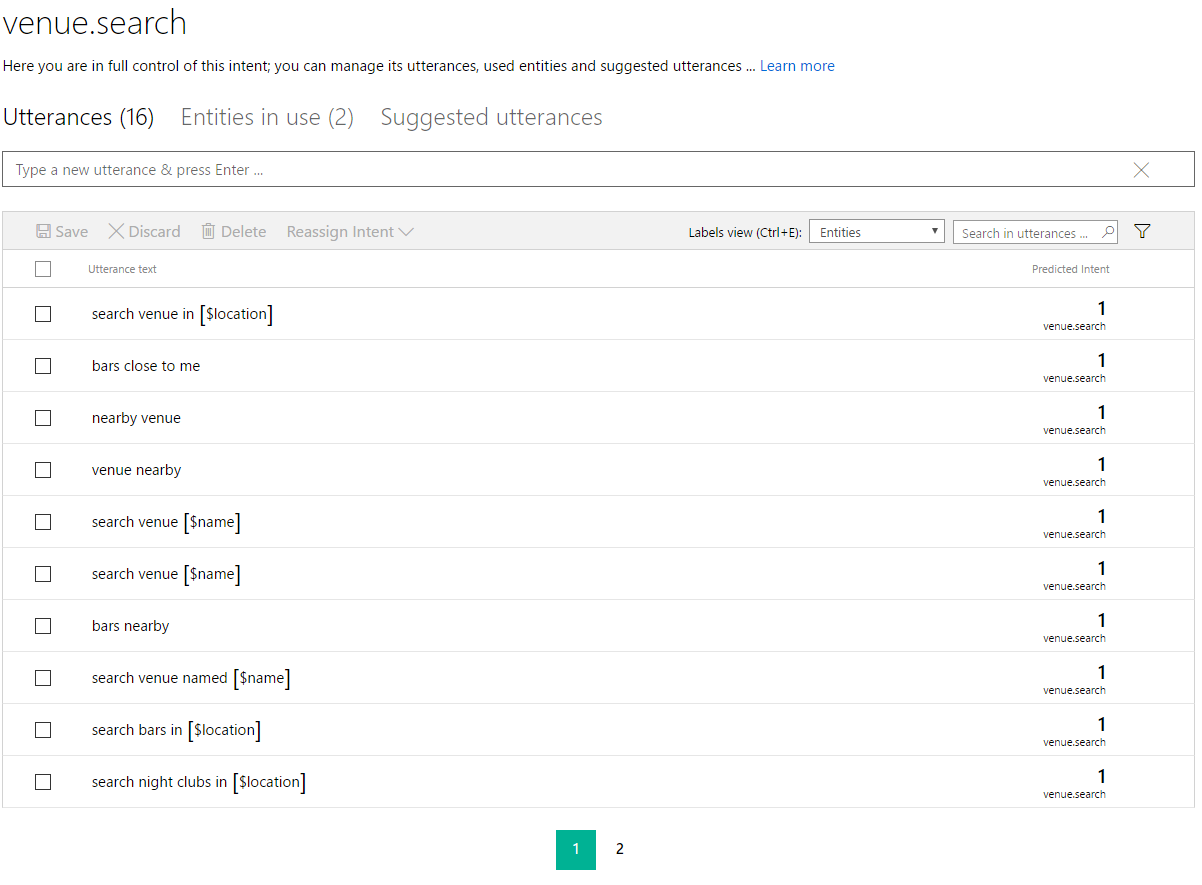


Image 3

Image 3 – training luis to recognize when the user wanted to find venue, when luis figures out that user wanted to search venues, block of code inside of the OQPYBot will be executed that will handle the request.

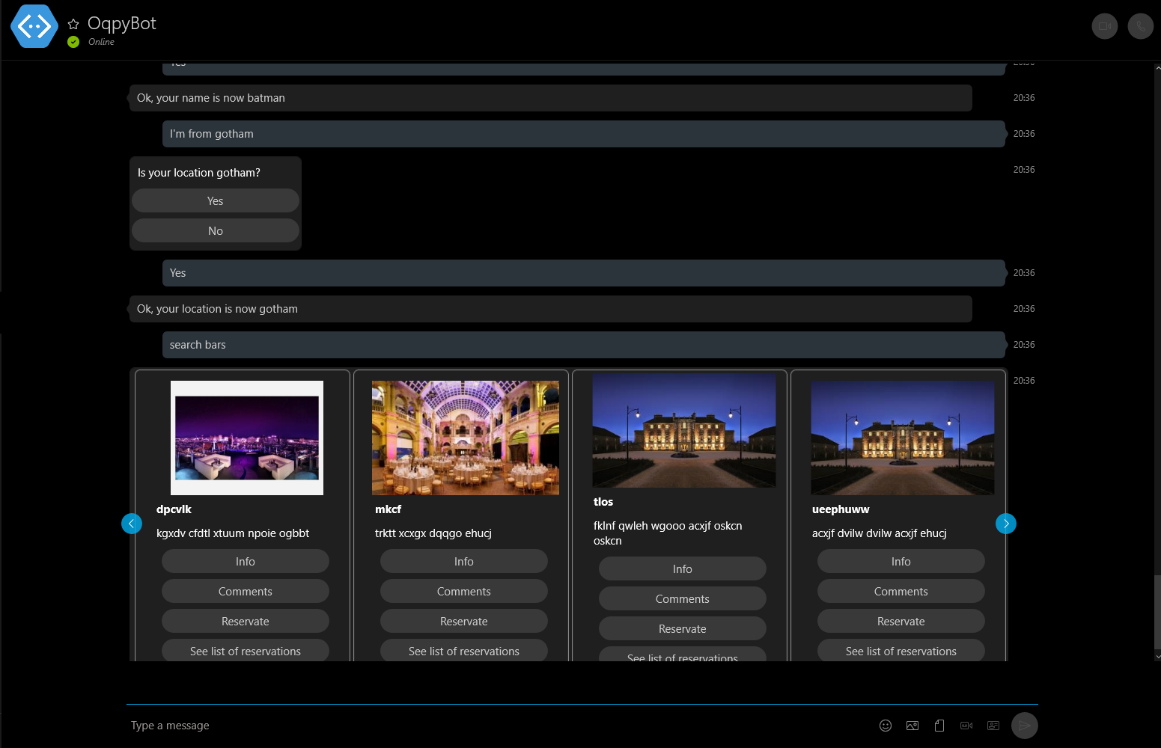


Image 4

Image 4 – Bot in action in skype client, with some test data, names of the venues are, obviously, not real, this was just part of one debug session.

## Current state

This project is a little bit behind OQPYManager project about 30-40% complete, because there is a lot of moving parts in this one and a lot of try and error was required, because we had to learn the whole bot framework and Luis api. They are nice but one has to take ones time to understand how it ticks and it ticks quite beautify after you spend some time with it.