Project Resolution process

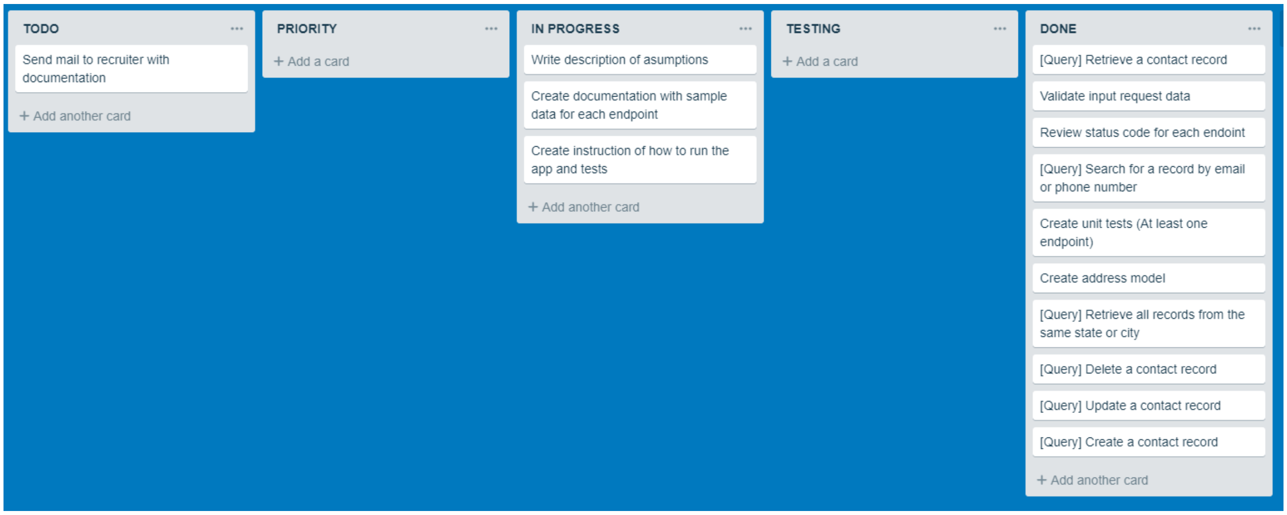
Hello! I'm Max! How are you doing up there in Chicago? Let me introduce you to this document where I will to briefly describe how I developed the project, what I assumed to do so and how to run it, along with its unit test. Let's get to it!

Technologies, tools and Work organization

- Frameworks:
  - ASP.NET Core 2.2
  - Entity Framework Core 2.2
- Database:
  - Microsoft SQL 2017.
- IDEs:
  - Microsoft Visual Studio
  - Visual Studio Code

- Version Control:
  - Git
  - Host: Github
- Endpoint testing: Postman.

To organize myself I created a *Trello* board, followed the workflow and got to it!



TODO
- Send mail to recruiter with documentation

PRIORITY

IN PROGRESS
- Write description of assumptions
- Create documentation with sample data for each endpoint
- Create instruction of how to run the app and tests

TESTING

DONE
- [Query] Retrieve a contact record
- Validate input request data
- Review status code for each endoint
- [Query] Search for a record by email or phone number
- Create unit tests (At least one endpoint)
- Create address model
- [Query] Retrieve all records from the same state or city
- [Query] Delete a contact record
- [Query] Update a contact record
- [Query] Create a contact record

As for version control, I used **Github** and worked with two main Branches: *master* and *dev*. In the last one I commited and pushed developed and tested code, and after a final test of the entire application, merge into *master*.

Assumptions

Keeping in mind the main **entity<Contact>**, I've assumed this app is developing to save and store people related to the user, Contact also has two object properties, each one with a 1:1 class relations: <**Address>** and <**ContactPhone>**.

- **Contact fields.**

Field Name	Required	Format	Validation
Id	**Yes**. Auto generated Unique identifier, Key	long	
Name	Yes	String	Max length (40)
Company	No	String	Max length (40)
Profile Image	No	String, receiving a imageURL or a base64 string	-
Email	Yes. AlternateKey	String	In ContactValidator
Birthday	Yes	DateTime	
ContactPhone	**Yes**.	ContactPhone class	
Address	Yes	Address class	

- **Address fields**

Field Name	Required	Format	Validation
Id	**Yes**. Auto generated Unique identifier, Key	long	-
AddressLine1	Yes	String	Max length (60)
AddressLine2	No	String	Max length (40)

City	No	String	Max length (40)
State	Yes	String	Max length (40)
Contact (owner)	Yes	Contact class	

- **ContactPhone fields**

Field Name	Required	Format	Validation
ContactPhoneid	**Yes**. Auto generated Unique identifier, Key	String	-
PersonalPhone	Yes, American phone number.	String	In ContactValidator
WorkPhone	No, American phone number.	String	In ContactValidator
Contact (owner)	yes	String, receiving a imageURL or a base64 string	Max length (40)

Setup

Download .NET Core 2.2 SDK (Or just the runtime)

To **build and run** the project using the command line:

1. Open the Command Prompt
2. Download the project repository with the following command: **"git clone** https://github.com/maximojgonzalezc/SCodeChallengeWebAPI.git"
3. Step into the solution directory folder **"cd SCodeChallengeWebAPI"**
4. Restore nuget packages with **"dotnet restore"**
5. Step into the project src **"cd SCodeChallengeWebAPI"**
6. Create the database **"dotnet ef migrations add FirstMigration"**
7. Apply the migration to the database to create the schema **"dotnet ef database update"**
8. Run the project **"dotnet run in the src directory."**

Point your browser to **http://localhost:5000**

Of course, you can also run it from either Visual Studio 2017 or Visual Studio Code with the IDE handling most of the steps above.

To run the **unit tests** follow all steps until 4, followed by:

1. Step into the project src **"cd SCodeChallengeWebAPITest"**
2. Run "dotnet test"

API Endpoints documentation:

https://documenter.getpostman.com/view/6990804/S17tQ7i7