**Console Application**

The console application was implemented by first mapping the objects of all responses api, then implementing the services that call the api and show the result in the console.

All the develop was done on the develop branch and then merged into master. Generally, this is not the way I follow for branch management, but the next:

When starting the development of a new feature, first you should create a new branch from develop and give it a name that explain the new feature or sprint or bug (as INC817423\_CloneInvoice or sprint2).

Once the feature has been developed, tested and committed by the developer, the branch sprint2 is merged with a pull request to develop branch and, generally, this start a devops pipeline that release the new code in dev environment.

If the dev test as successfully executed, then the develop branch is merged with another pull request to test (or QTA) branch, which again start a devops pipeline that release the new functionality in test environment.

Generally, now will be execute test from users or test manager to check that the functionality was developed correctly and there haven’t regression.

When all tests are passed, then test branch will merge to master branch and a manual release into environment production

**Automated Test**

As I had said during the first interview, I have never done automated test (I have only supported a colleague in a tool configuration to run automatic tests on native application because I develop also iOS and Android app). I read up on it and I made some simple tests using xUnit, that check if the service and functions do their task without errors. This pushed me to change the developed code, inserting a start class so it’s functions could be called in test. In addition, I set all the class as public so that they could always be called in the tests if necessary. Then I had to change the version of .net from 5.0 to 3.1 because the last version was not supported by xUnit.

I understand that is not the way to develop automatic test, which needed to be done in a more structured way and with more functionality. It will be one of the next topics I will try to learn.

**Code review**

About code review, my experience is limited. For work I use often, and for various languages, sonarqube or sonacloude but I have never had to configure them (because they are generally managed by the customers who wants to verify our work). So, I created an account on sonacloud and then connected it to my github repository but I wa blocked during configuration

Graphical user interface, text, application, email

Description automatically generated

Once configured, these systems are generally easy to use, and the way I write code has changed significantly since using them:

* I make more atomic functions that do one thing instead of one function that does everything
* I’ve reduced the cyclomatic complexity of my way to write code

I’m not able for now to show you the analysis of my code, but I hope you will appreciate the fact I tried.

**API document**

When I wrote the Api document, I tried to think of everything that could be needed. For this reason, I inserted also an api for Login because it there isn’t in the exercise.

For each API I have indicated description of behavior, endpoint, parameters and response (including any errors).

This is a first version, and several improvements can be made:

* Instead of separate parameters, generally I use an object that contains all of them in order to better structure the code and simplify (in my opinion) the work of FE developer.
* All the API, both get and others, currently only manage one entity at a time. It is possible request the weather for one city, but to get the weather for two separate cities or for multiple days of the same city, you need to make multiple calls. This is a big limitation that can be easily passed by accepting an array of objects as parameter, where each object contains all the information about a request, and the response return an array of objects and not a single instance. In this way it’s possible to ask for a single city or more cities without having to change the implementation either on the FE or BE side. You are more open to changes and a new specification can be incorporated without particular problems or regression