Release 03

Release Summary

In this release, the focus was on the implementation of the function to calculate the cost of the flat and calculate the total payable for each owner.

The most important part was the implementation of the respective tests and the creation of the Test project where all the tests will be

Release Phases

Phase 1 - Planning

- In this release we start by interpreting the new statement and collecting all the information we might need.
- It was also at this stage that we owed tasks and their respective branches. Looking like this:
  - **feature@calculate-owner-flats-price** → Branch responsible for implementing the feature to calculate the total cost of flats for a specific owner
  - **feature@calculate-flat-price** → Branch responsible for implementing the feature to calculate flat cost
  - **test@calculate-owner-total-flats-price** → Branch responsible for implementing the tests to calculate the total cost of flats for a specific owner
  - **test@calculate-flat-price** → Branch responsible for implementing the tests to calculate flat cost

Choice of NUnit as Test library

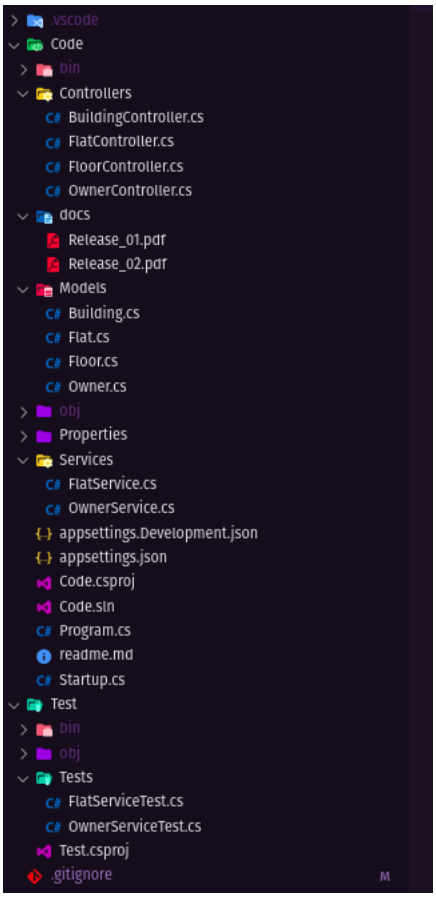
We decided to choose NUnit as we saw the following advantages:

- Library with the best written documentation (Microsoft support)
- Simple syntax
- Easy integration with Dotnet
- Good performance

Phase 2 - Execution

For the implementation we decided to isolate the new features in a **"Services"** folder, respectively the **"FlatService"** and the **"OwnerService"**.

Project Folder Structure



For the two projects to integrate we had to add the following content to Test.csproj of the Test Project.

```
<ItemGroup>
  <ProjectReference Include="..\Code\Code.csproj" />
</ItemGroup>
```

Demonstrations

FlatService implementation

```
using Code.Models;

namespace Code.Services
{
    4 references
    public class FlatService
    {
        2 references
        public float calculateFlatCost(Flat flat, float squareMeterCost){

            float flatCost = flat.SquareMeters * squareMeterCost;
            return flatCost;
        }
    }
}
```

FlatService Test implementation

```
namespace Test.Tests
{
    public class FlatServiceTest
    {
        FlatService flatService = new FlatService();

        [Test]
        public void itShouldBeAbleToCalculateFlatCost()
        {
            Flat flat = new Flat
            {
                Id = 1,
                SquareMeters = 20,
                NumberOfRooms = 2

            };

            float expectedValue = flatService.calculateFlatCost(flat, 100);

           // Console.Write("############ Expected Value: " + expectedValue);

            Assert.AreEqual(expectedValue,2000);



        }
    }
}
```

Phase 3 - Documentation

- For the construction of the project documentation, we used the Notion tool, a very practical application of notes in Markdown, with support for PDF export.
- Endpoint documentation with Swagger.
    - We decided to implement the swagger as it is an excellent way to test and document the built endpoints.
    - To access it you should:
        - Clone the repository

- Build and Run the project
- Access the http://localhost:5000 and you be redirected to Swagger page

Tools used

- The tools that were used for the development of the project were:
  - **VS Code with pack extensions for .Net CORE**
    - We chose VS Code because the operating system that most members of the group use is Linux, and the main IDE for .NET (Visual Studio) is not available for it.
  - **Dotnet CLI**
    - The use of the dotnet CLI was fundamental for the project's productivity, and with simple commands we can give the project **build** and **run**

  This development environment pleased us immensely, as we were able to implement the code and execute it very easily. Without a doubt a very good long-term environment.

Sources

Unit testing C# with NUnit and .NET Core - .NET

This tutorial takes you through an interactive experience building a sample solution step-by-step to learn unit testing concepts. If you prefer to follow the tutorial using a pre-built solution, view or download the sample code before you begin. For download instructions, see Samples and Tutorials. This article is about testing a .NET

https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit

NUnit.org

NUnit 3 was created by Charlie Poole, Rob Prouse, Simone Busoli, Neil Colvin and numerous community contributors. Earlier versions of NUnit were developed by Charlie Poole, James Newkirk, Alexei Vorontsov, Michael Two and Philip Craig.

https://nunit.org/

Remote Repositories

Github

GitHub - JoaoGomes5/DevOps2022G-JJD: 🚧 Project carried out for the Software Development and Operations subject to improve the practice of versio

🚧 Project carried out for the Software Development and Operations subject to improve the practice of version control and development in .NET Core 🐙 - GitHub - JoaoGom for the Software Development and Operations subject to improve the practice of version control and development in .NET Core 🐙

https://github.com/JoaoGomes5/DevOps2022G-JJD

Bit Bucket

Bitbucket

https://bitbucket.org/joaogomes13/devops2022gi/src/master/