Sales App Documentation

> See the problem description!

How to use the Sales App

1. Once you have the project in your PC, execute program.
2. **Welcome Page**:
    - Use the `Settings` button of this page to be able to set up how you want to log the operations that are performed in the application. Feel free to choose the targets you like the most and see what happens when you click many times on each button.
    - When you are ready, hit run to start enjoying.
3. In the following pages, the stores you have will be loaded via QR codes previously generated. Don’t worry if you don’t have any QR code, you can create new stores very easily.
4. The app is designed to help you with the logistics needed to deliver all your products. With this application, raising new orders and planning the route to deliver is easier than ever.
5. *Remember.*
    a. You can go back to see your logs.
    b. The QR codes are stored in your desktop folder.

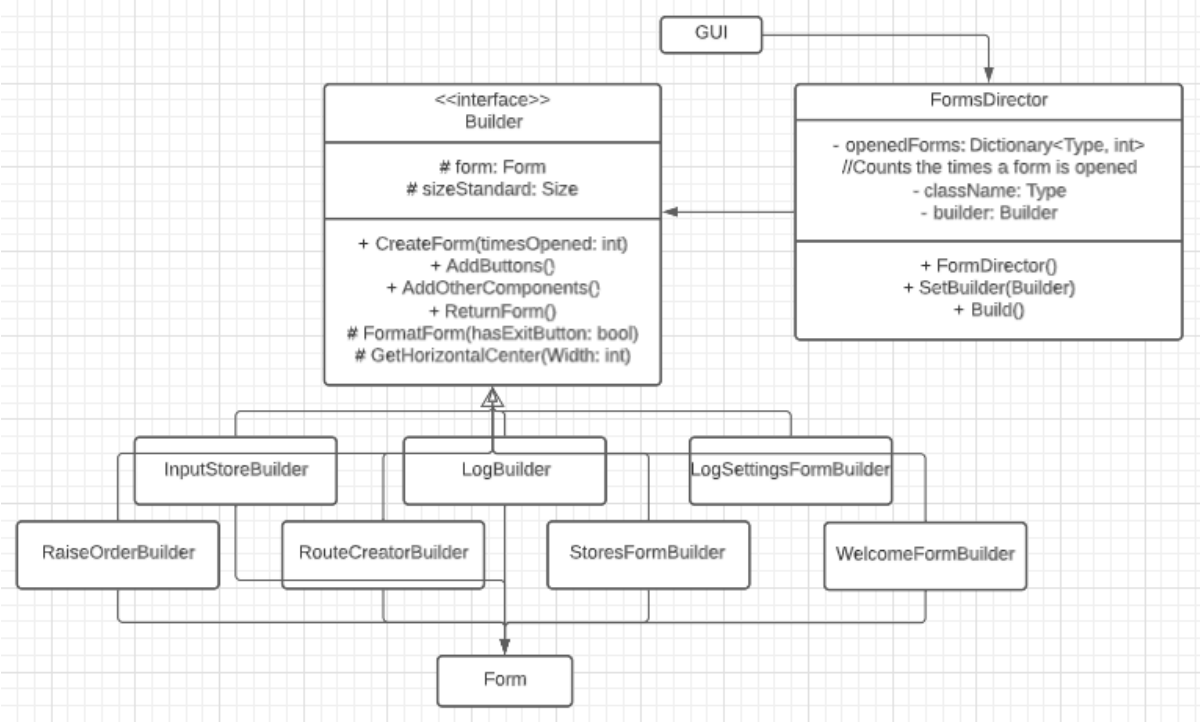
Design Pattern

The patterns implemented in this project are the following ones:

Creational

💡 Builder

This pattern was used to generate and setup every form, the client of this pattern was the class `GUI` and it was in charged of managing the execution of every page.



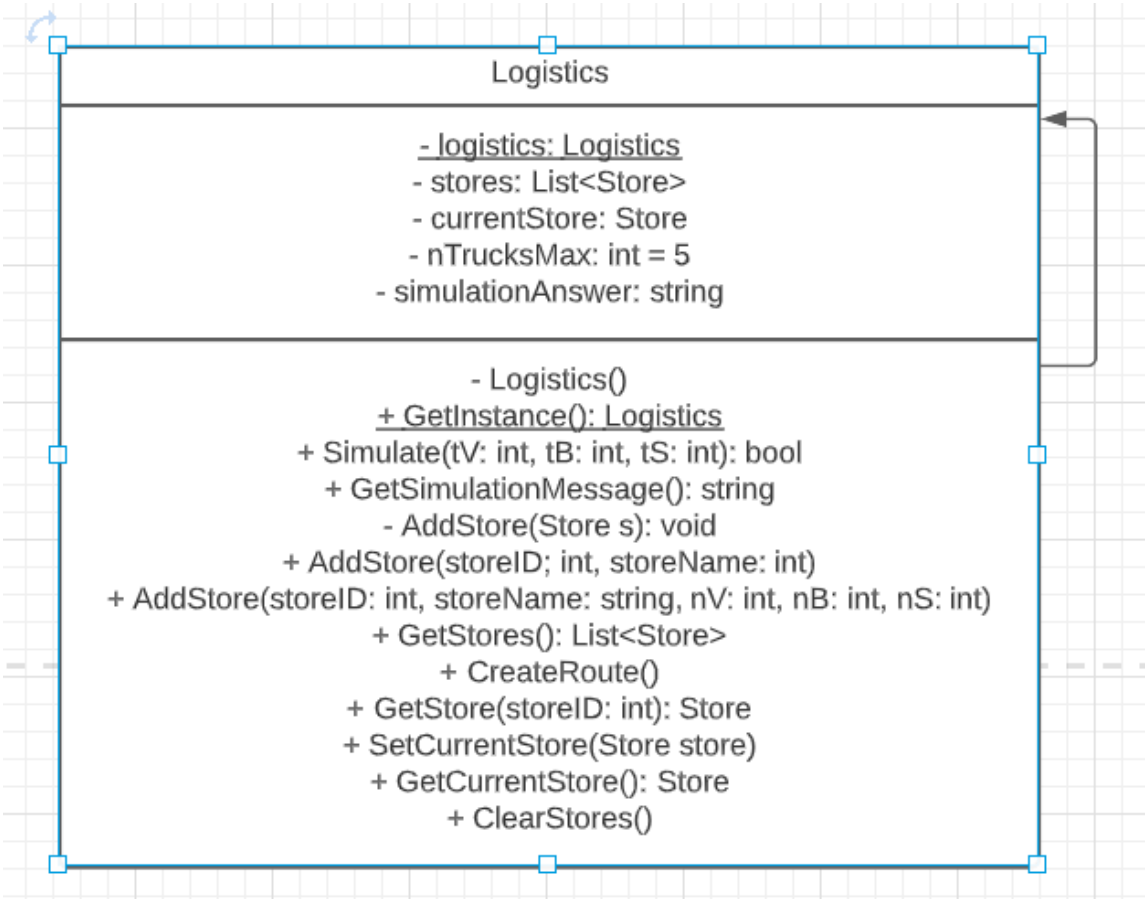
💡 Singleton

Two singletons were implemented in the solution:

- Logistics

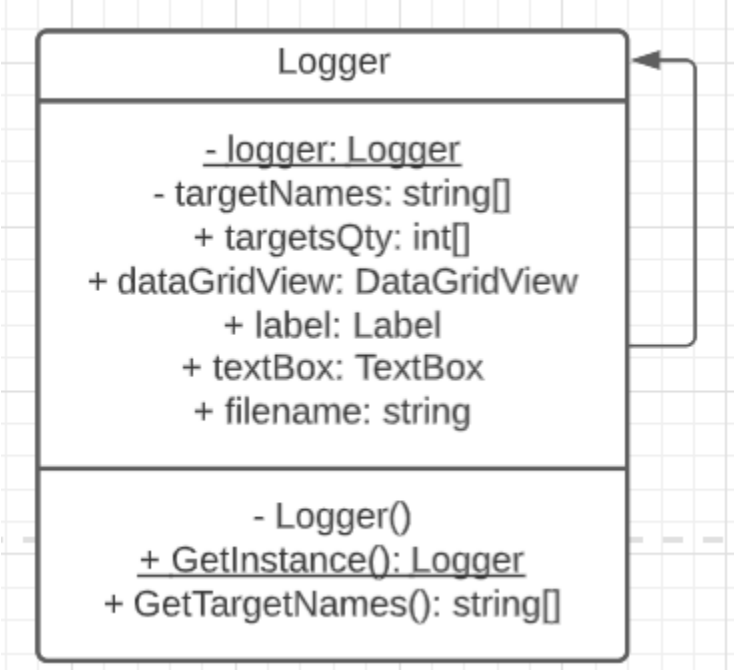
  A singleton was useful in this case because it is a class able to deal with lazy initialization, and a single instance with global access was going to be needed.

This class is responsible for all the logic related to `Store` management, simulation and creation of routes.



- Logger

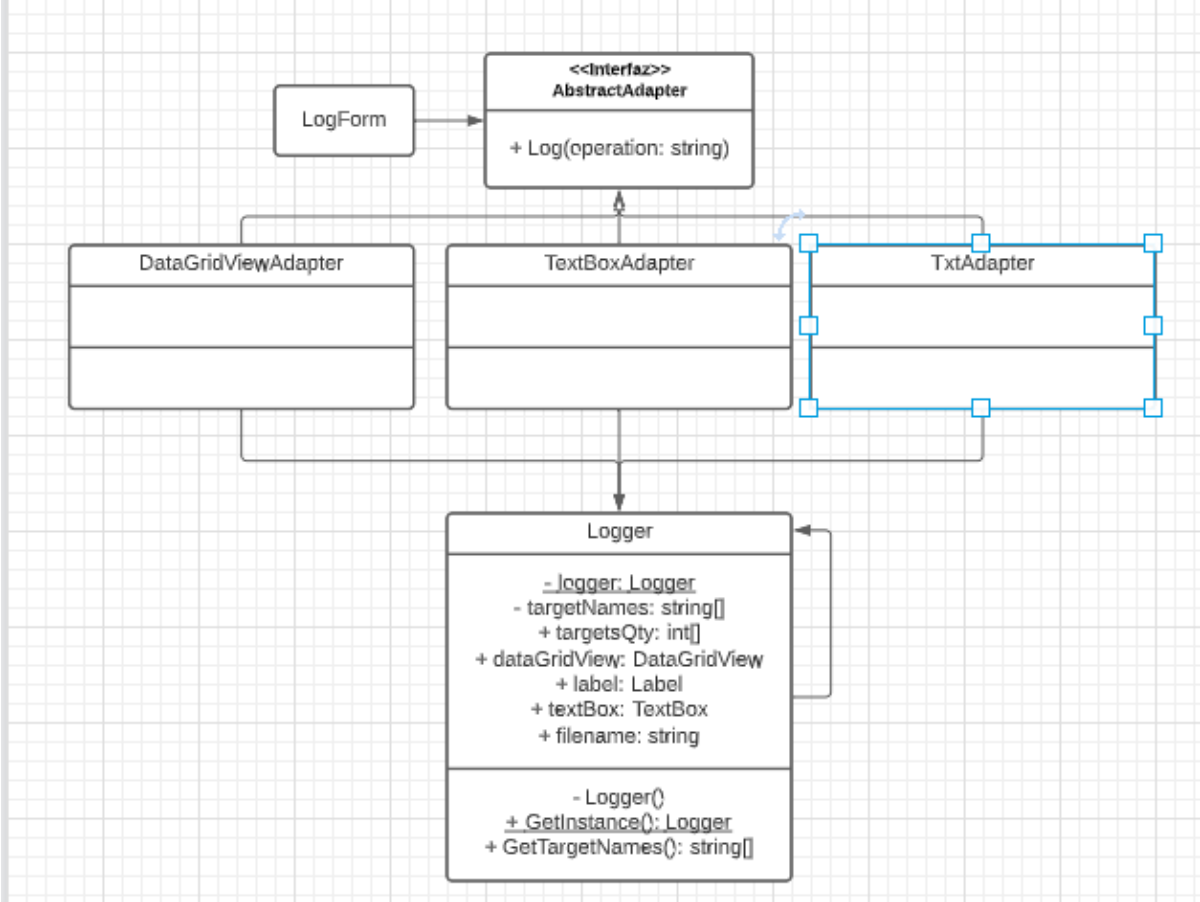
  This class is in charged of writing the operations made during the execution of the program and contains useful information. It also provides access to the containers that are going to have the log information.



Structural

> 💡 Adapter

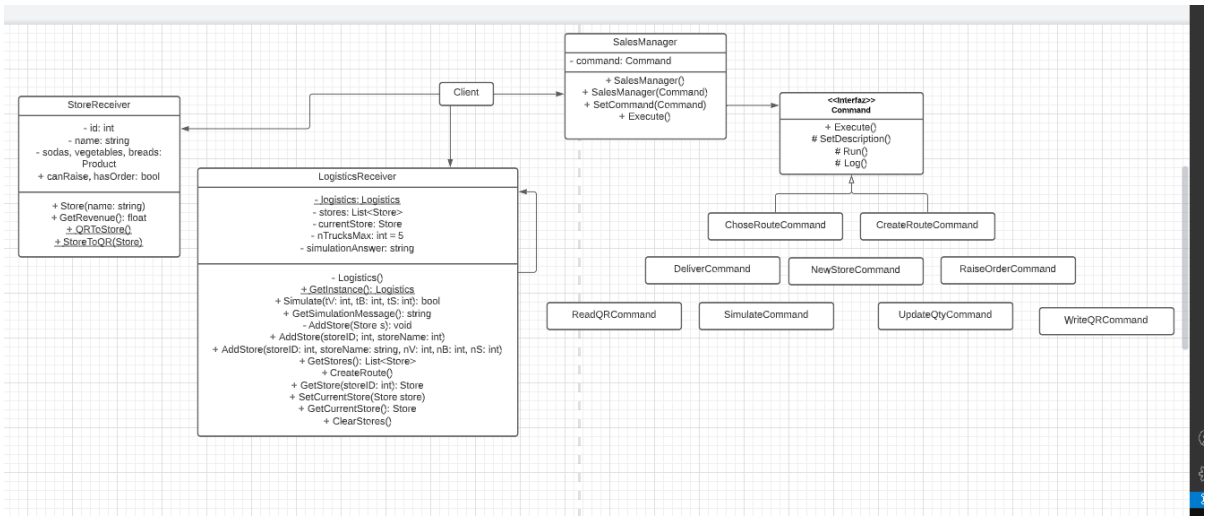
This class works in collaboration with the `Logger` class, because that Singleton plays the role of the adaptee, containing the necessary targets for the logging information, and the adapter classes provide the specific implementation needed to register into different containers.



Behavioral

💡 Command

Command is the pattern capable to connect all the requests between senders and receivers, like disctint forms and the Logisticts class, and had the super important job of logging all the operations that are made through it. This pattern also helps to decouple requests, which made the development phase of this project way easier. Lots of Commands where needed but registering the operations wasn’t a problem because Command provided support for it, and it helped to have more control over the requests and allowed to parametrize objects, which helped to reuse code.



Luis Eduardo Robles Jiménez