## Problem statement

People from Emag company need a new feature for their website. They need a new page in which there will be sold only reconditioned laptops in order to avoid confusion for their customers, because sometimes they do not read the description carefully to notice that they will not buy a brand new laptop, but a refurbished one. So, a page with only this type of laptops is perfect to solve this problem.

They gave us some details about how the new page should be organised. The application will be divided in two: the administrator mode, the customer mode.

The administrator will be able to: add a laptop in stock, to remove a laptop from the stock, to update a laptop from the stock, to see all the laptops from the stock.

The customer will be able to see only one laptop together with its specifications, price and how many laptops of that brand are on stock. If the customer wants to buy one, it will say yes, otherwise there will be shown the next laptop from the stock. If the customer has seen the entire list, it will be shown again until the customer wants to stop the process. When it is done, the total sum will be displayed.

## Solution

To accomplish this task, I began with the class named Laptop in which I included the constructors, getter and setter methods and also, the assigment operators that I will use later. To store all the refurbished laptops, I created a Repository class with the help of templates, because, this way helped me to avoid the repetition of code. For a better and safer functionality, I used a std::vector container, because of the multitude of functions, such as: std::find\_if( ), vector.size( ), vector.remove( ), etc.

The next step was to create the Controller that helped me a lot with the implementation and syncronization of the undo and redo containers. Besides the fact that the administrator was able to add/remove a laptop from the repository, he could also undo/redo these actions in case of a mistake.

Then, at some point, I had a task that required to filter the laptops by some properties. For example, I had to filter the laptops by their display size. So, I made an interface called “BaseFilter” in order to generalise the process. The classes that will inherit from BaseFilter interface( such as “LaptopsWithBaseFilter”) will give a proper filter according to the required property.

The application requirements did not require a database, but I used a persistence engine. Why did I use this? Because I needed a way in which I keep safe all the modifications applied to the stock. For example, we are in the Administrator mode, we open the program and we see that we have 2 laptops from Toshiba and one from Asus. How is this possible? Well, we have a txt file that stores a string which represents all the data from our repository. I made a function that is called every once we open the program and it converts our string into the objects that we need.

Now, we want to remove the Asus laptop and close the program. We also have a function which will convert the remained objects from our repository into a string and this string will be placed in the txt file that we have discussed above. I want to mention that, I did this kind of functions that work with both a CSV file and JSON file.

As a last step, for user interface I used Qt in order to give to the client a friendly and intuitive graphical user interface. With the help of this platform, I included some vertical/ orizontal/ grid layouts( in order to organise the text boxes, buttons, etc), a menu bar and some pop-up widgets.

To conclude, this project helped me to understand way much better the concepts of a class, the usefulness of an interface and of a template, to implement the loading data and saving data accordingly to each format( CSV, JSON) and to organise an user interface easier in Qt.