Software development project
Final report

Tardif Félix, De Pret Rose de Carlsberg Sébastien, Léonard Matthias,
Krasowski Dawid, Foly Marius, Drewnowski Bartlomiej & Ventura Andrea

June 2021

Bovélo

Table of Contents

1 Technical specifications

1.1 Objectives

The objective of this project is to create an optimal IT solution for a startup that specializes in selling bicycles in order to improve its operation and help it organize its business. To do this, we were asked to focus on the following three issues:

- The elaboration of the production planning, this planning is realized every week based on that week bicycle orders by the production manager.
- Stock management, despite the will to keep a set of common parts between the different models, it happens regularly that the production is stopped because of a lack of parts. Bovélo would like us to help the production manager to manage the stock according to the orders and the planned production so that this does not happen anymore while optimizing the quantities in stock. It happens that parts are damaged, so a minimum stock of parts must be provided in addition. The application must also be able to store the generated parts supplier orders.
- Sales because the demand is very high and the representatives are unable to give a provisional delivery date, which greatly hinders the bike stores in their orders. Bovélo would like us to create an application that allows representatives to estimate a manufacturing date based on the demand expressed by the bike store. We also want to have the cost price of each model. This application must be simple and ergonomic because the representative must be able to handle it very easily during the discussion with the bike store manager.

The addition of a new type of bike, for example electric, must of course be easily integrated.

It is imposed to us to work according to the AGILE methodology in three distinct iterations, which are defined as follow:

1. The management of the bike catalog and customer orders.
2. The elaboration of a production schedule of the bikes to be assembled.
3. The management of the stock of spare parts and their orders from independent suppliers.

As it is a group project, we divided each tasks among us, and put the results in common at the end of each iteration. We were also required to use Azure DevOps to allow a follow-up of the teachers and a GitHub repository to easily follow the progress of the project.

The imposed programming language is CSharp, and the database is MySQL. We were asked to respect the coding conventions, good practice rules and SOLID object oriented design principle.

1.2 GitHub link

https://github.com/20325-ecam/Projet_Bovelo

1.3 Glossary

Here is a list of the common term used throughout this report and their definition.

-Bovélo

A company which orders bicycles components to assemble bicycles and sell them. Hired us to create a software solution for their business.

-Mechanic

A person working for Bovélo assembling bicycle with spare parts.

-Production Manager

Person working for Bovélo supervising the planning and assembly of new bicycles.

-Representative

A person working for Bovélo selling assembled bicycles to other stores to distribute them to consumers.

-Client

A business buying bikes from Bovélo to re-sell them to consumers.

-Provider

Provides the spare parts that Bovélo uses to assemble its bicycles.

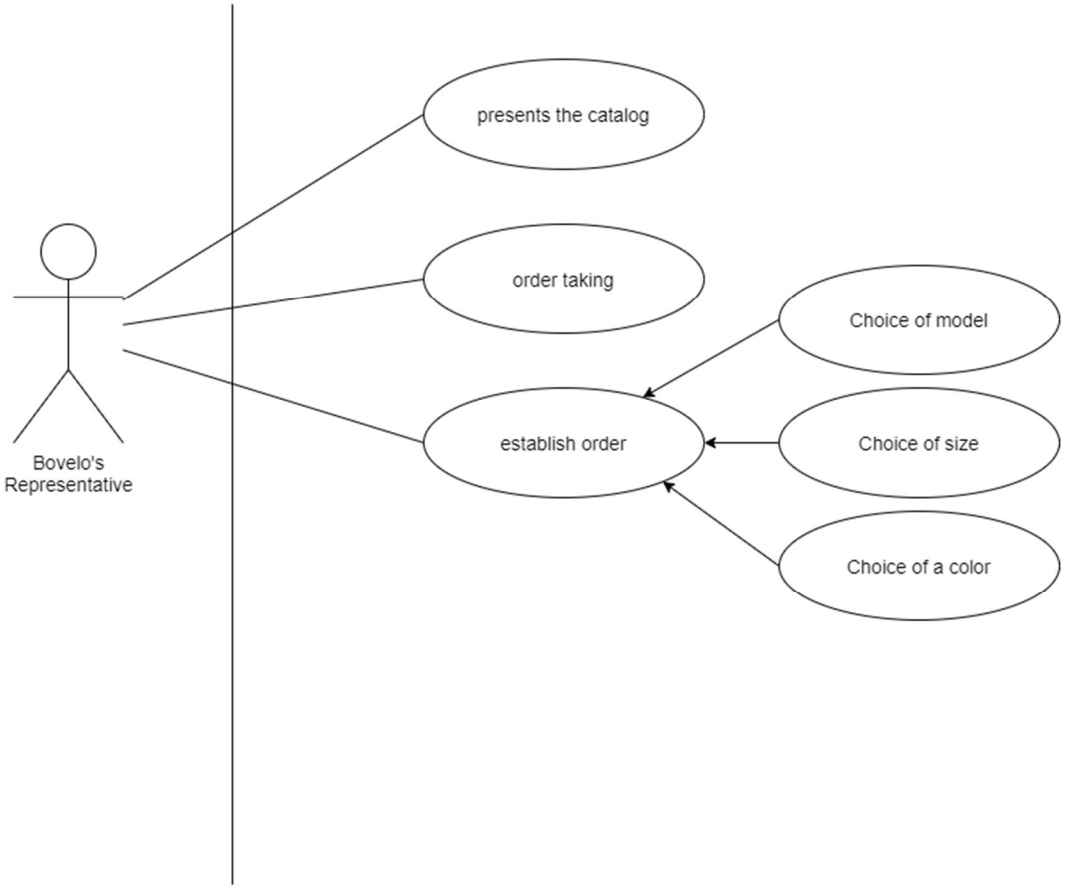
-Bicycle variation

There are three basics types of bicycles sold by Bovélo, but each one of these can come in different color and size. Each color and size combination correspond to a single variation.

1.4 Use case diagram

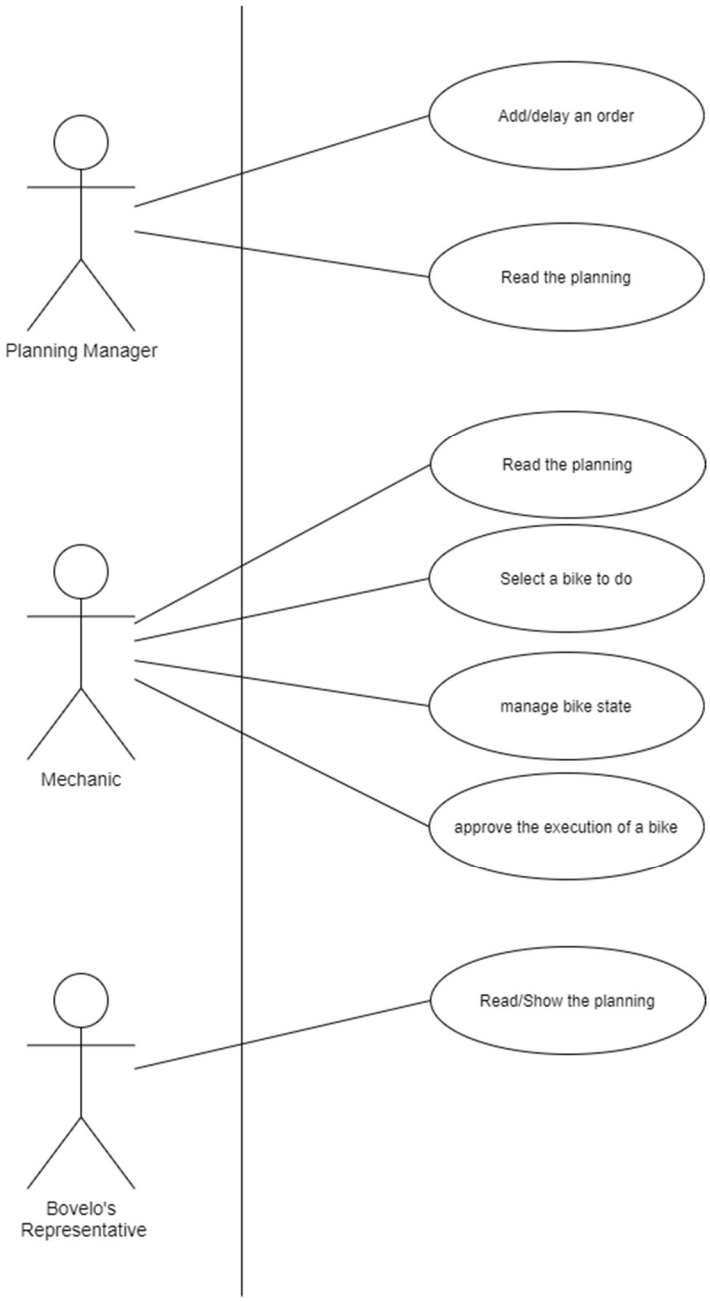
First Iteration:

The software in the first iteration is intended for only one type of actor: the Bovelo's representative who firstly wish to present the catalog to the client and take the order of the client. He must be able to select the model of the bike, choose the size and the color in order to establish an order.



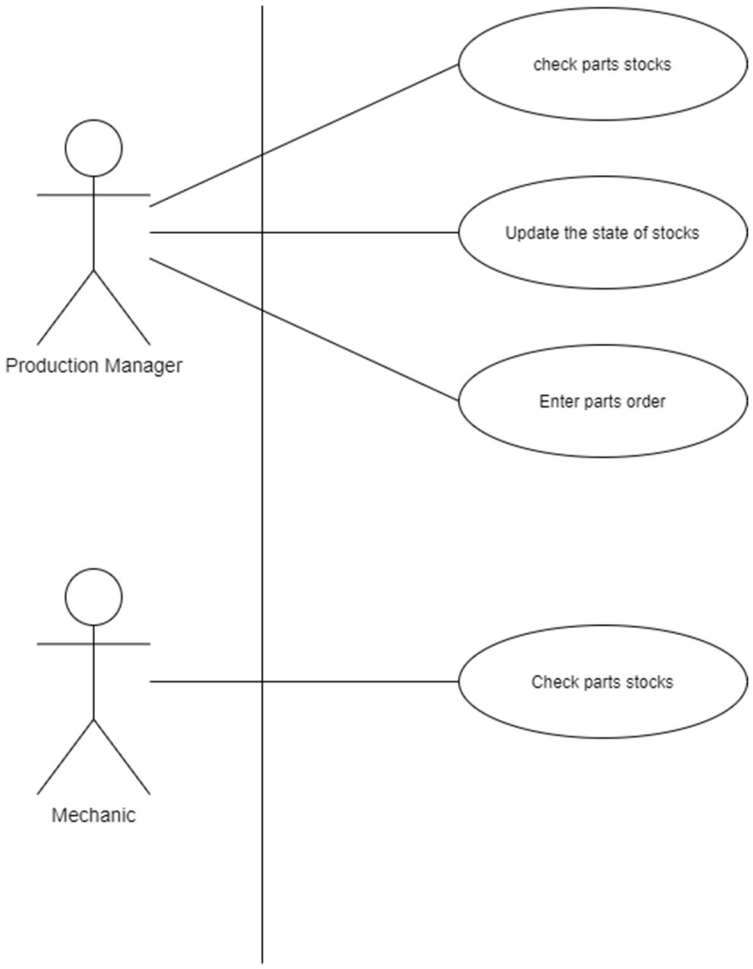
Second Iteration:

For the second iteration there are three actors using the software : The Planning Manager, Mechanic and the Bovelo's Representative. All of them must be able to read the planning. The Planning Manager has the right to edit the planning to add and delay any order. Every Mechanics have the possibility to select a bike to do and manage its state of production. Once the bike is finished a Mechanic can approve the execution of the bike.



Third Iteration:

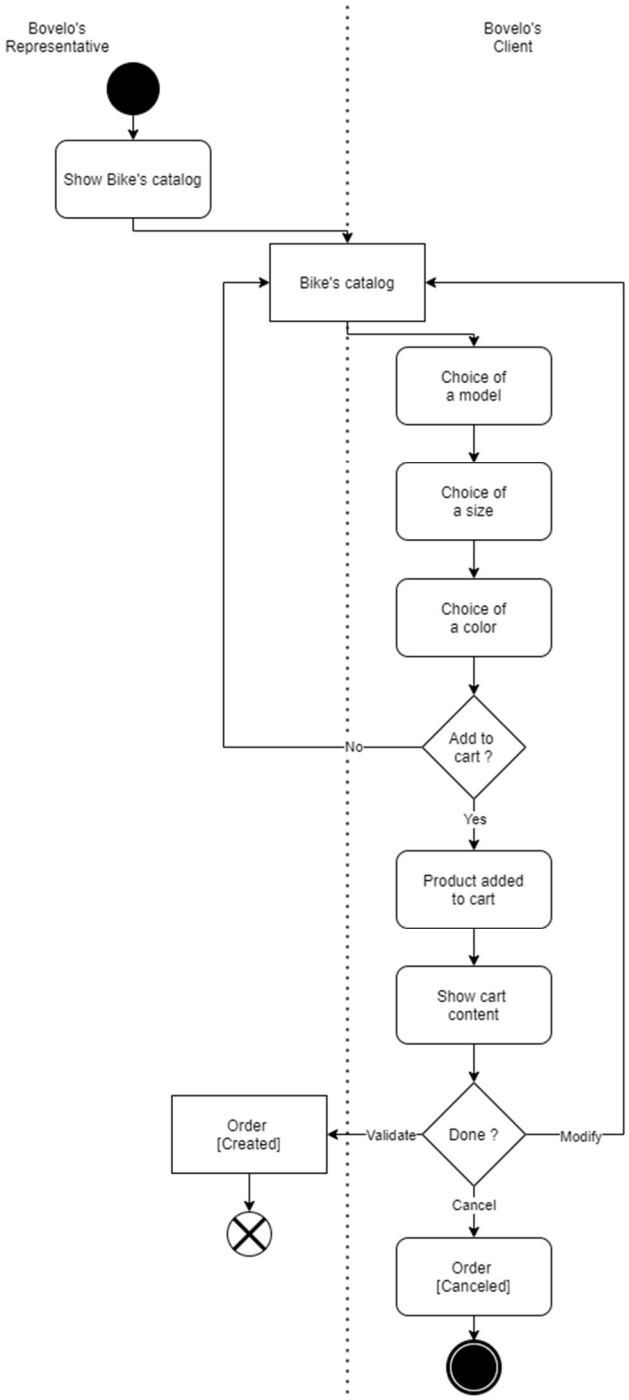
For the last iteration two users will operate the software: the Mechanic, to start assembling a bike, need to check the parts stocks. The production Manager will also check the parts stock. He must be able to enter a new part order if the balance is lacking for a particular part. Finally, He can modify the state of stocks if there are broken parts or to update the minimum stock required.



1.5 Activity diagram

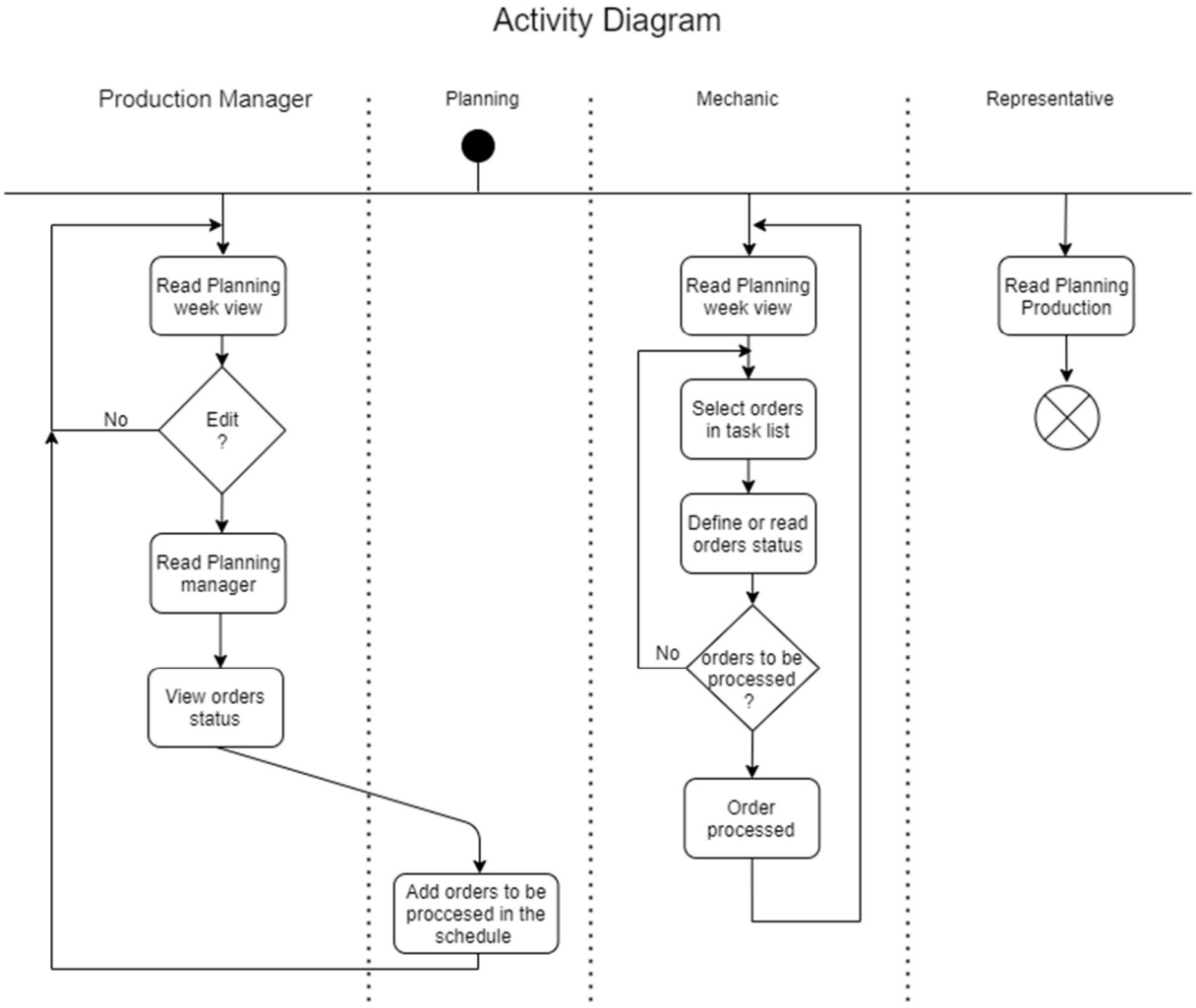
First Iteration:

For the first iteration, we have two actors namely the representative and the customer. Here, the representative can show the bovélo catalog to the customer. However, the representative is the person who validates the customer's order. As for the customer, he has the ability to order one or more bikes on the Bovélo app but he can also cancel his order. To order a bike, the customer has the possibility to choose the model, size and color of the bike.



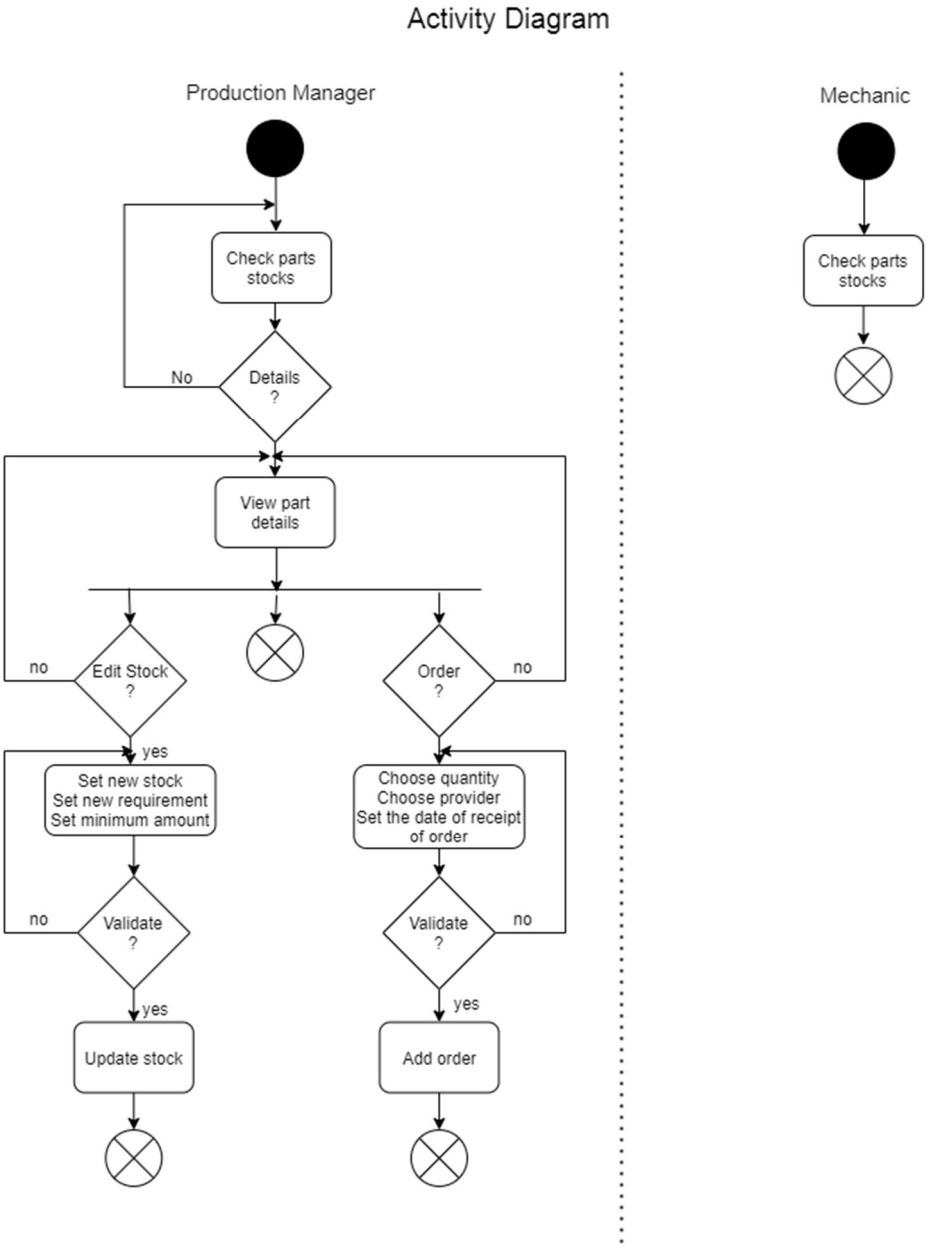
Second Iteration:

The second iteration is to implement a production schedule based on customer orders. Here, we have three actors: the production manager, the mechanic and the representative. First of all, the representative has reading rights on the production schedule in order to be able to inform customers about the delivery of the orders made. Then, the production manager can view the weekly schedule, modify the production schedule by switching orders and see the progress of the orders. Finally, the mechanic defines the status of an orders according to the schedule established by the production manager.



Third Iteration:

The third iteration allows users to see and manage the stock of parts needed for bikes. We have two actors who are the production manager and the mechanic. The mechanic has the rights to check parts in the stock. As for the production manager, he also has the rights to check parts in the stock then to modify the stock and place new orders for missing parts.



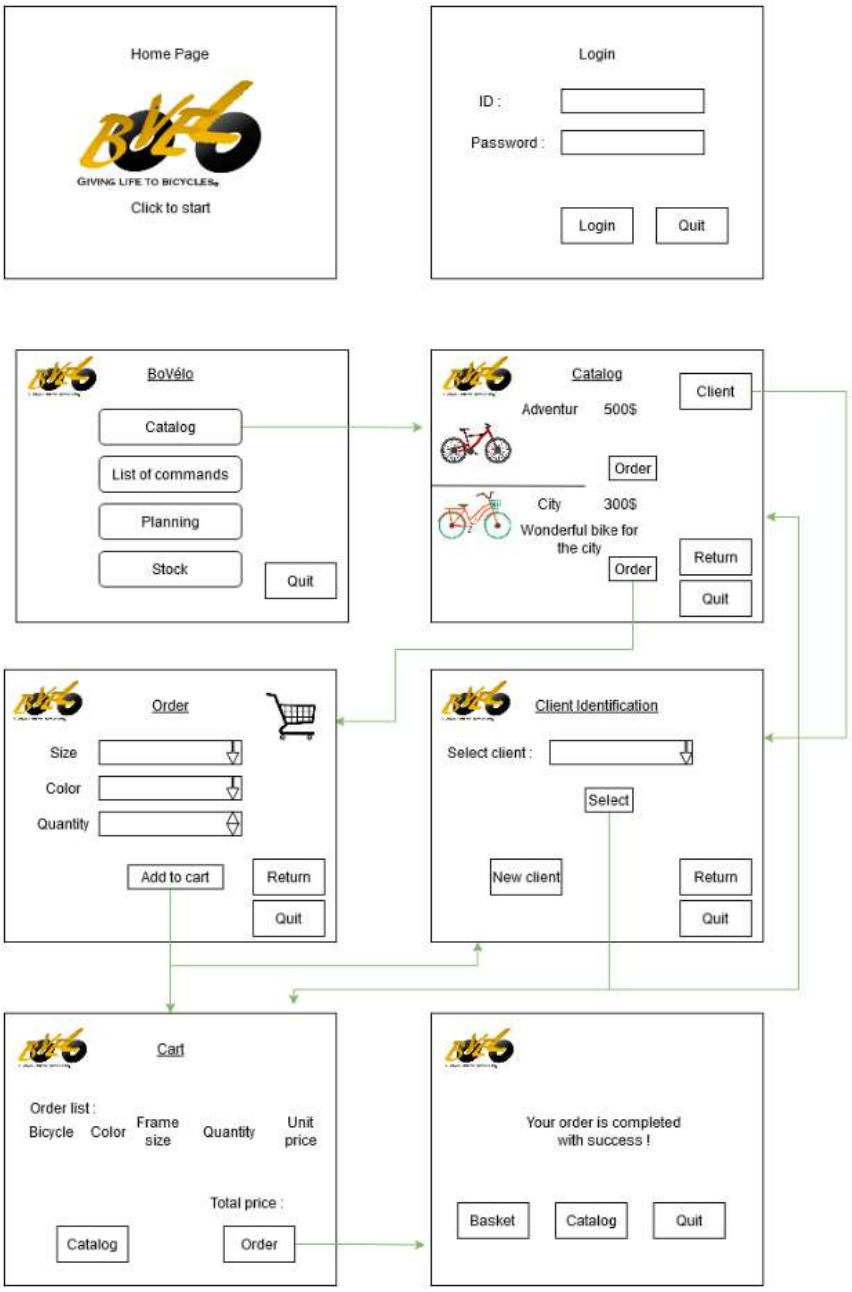
2 Design specifications

2.1 Graphical interface guide

2.1.1 Mockups

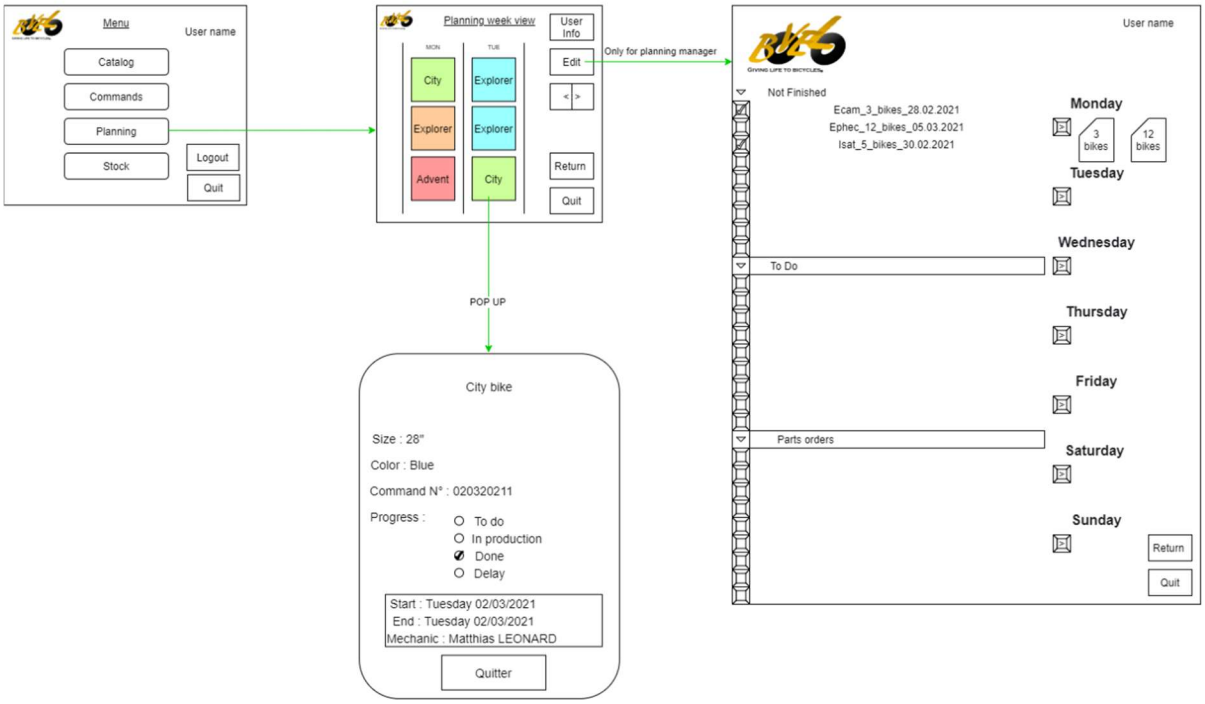
First Iteration:

For the first iteration we wanted to allow users to login as either a mechanic, representative or manager based on their login ID. The first idea was to display the different menu only for the users with the right role. The representative had access to the catalog, in the catalog the user can log the client or create a new one. If the client wants to pass an order of a bike he must choose several settings such as the size, colour and quantity. If the client is already identified a recap of the cart is displayed otherwise, he has to be identified.



Second Iteration:

For the second iteration we focus on the planning management, we imagined a planning view with a grid compound of the days in columns and the bikes in row. We also imagined a colour code for progress, red for delay, green for the bike already done, and blue for the ones still to do and orange for bikes in production. On the Planning view we can change the progress of a bike just by clicking on the bike. The planning manager is also allowed to edit the planning. In the end this mockup proved to be a bit difficult to implement so we resorted to a simpler option whish you'll see in the user interface part of this report.



Third Iteration:

The goal of the third iteration was to create a stock part manager. We decided to display a long list of all bike's parts, with the columns giving the main information of each part, whish show the number of parts required, ordered, and available in stock. Through this information we are able to calculate a balance which describe the risk of a stock to be empty.

Each row which is a bike's part has a details button, this button open a windows with detailed information of that part, the minimum required, the number we need and also the amount ordered to external parts providers.

In this window we can add a part order just by clicking on the bouton "add new", we select the provider, the date of shipping and the quantity.

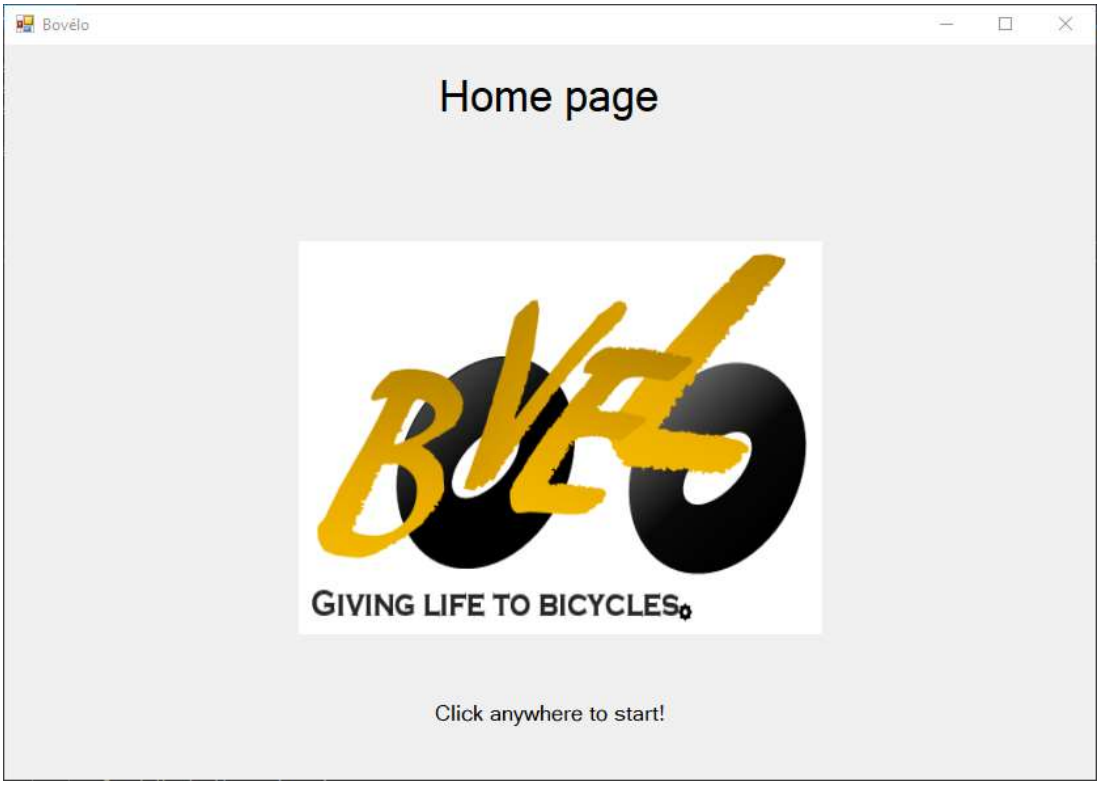
There is a data table which is a recap of all the current part order with the quantity ordered and the date of the shipping. When a order has been received we can confirm the reception, this action add the quantity ordered to the stock.

It is possible to manually edit the amount of parts currently in stock and the minimum required number of parts that should always be in stock if the user wish to do so.

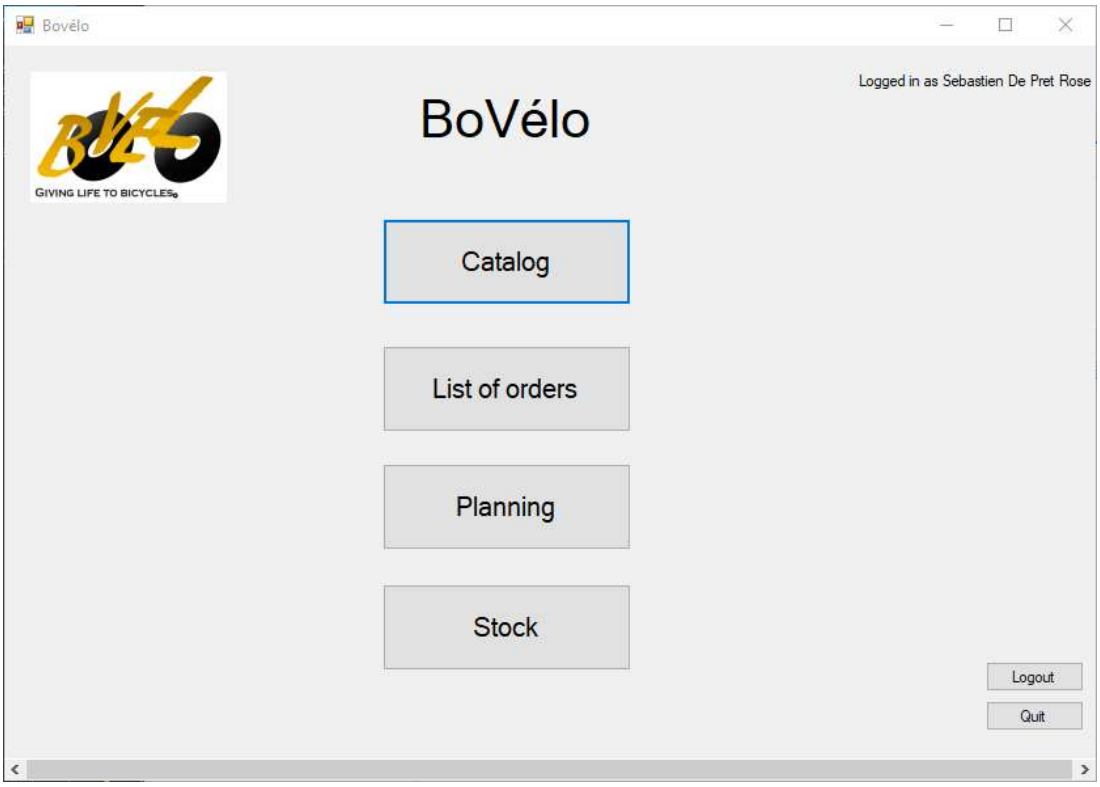
2.1.2 User interface

In this part of the report are presented a few screen captures of the main windows of the application, the Home Page, Main Menu, and all four Sub-Menus.

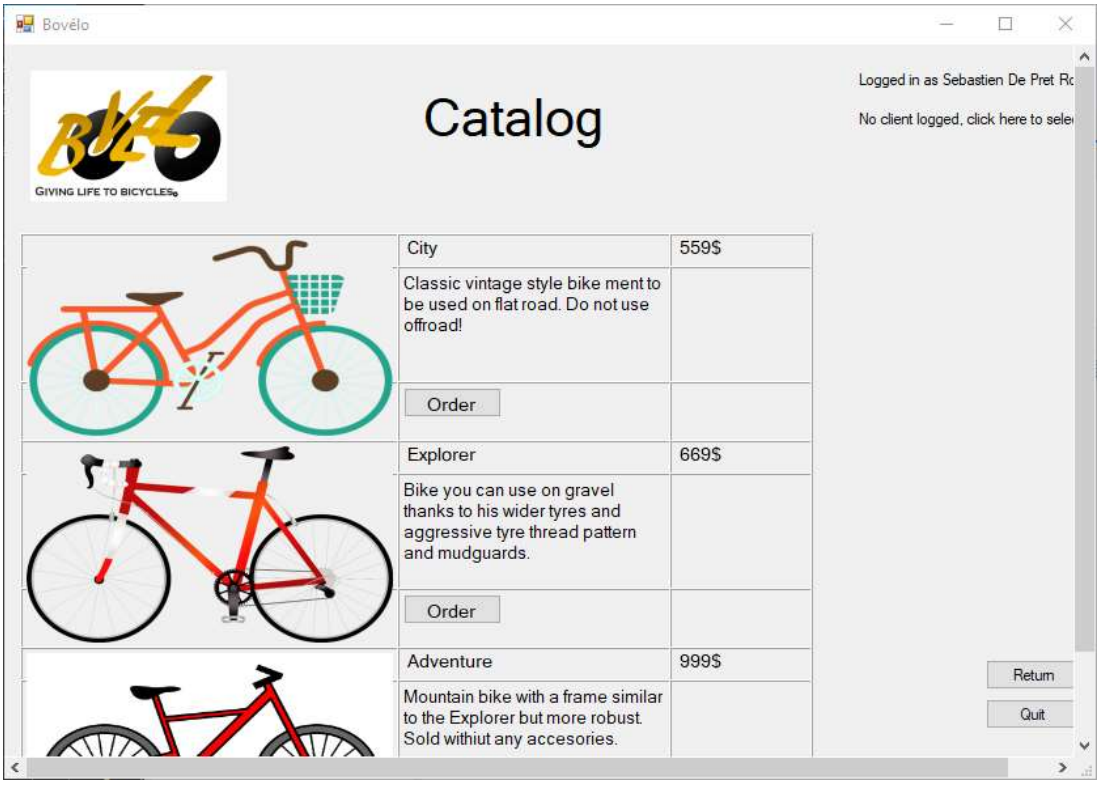
This first screen is the Home Page, the window you are greeted with when launching the application. The user can click anywhere on the screen to login and start using the app.



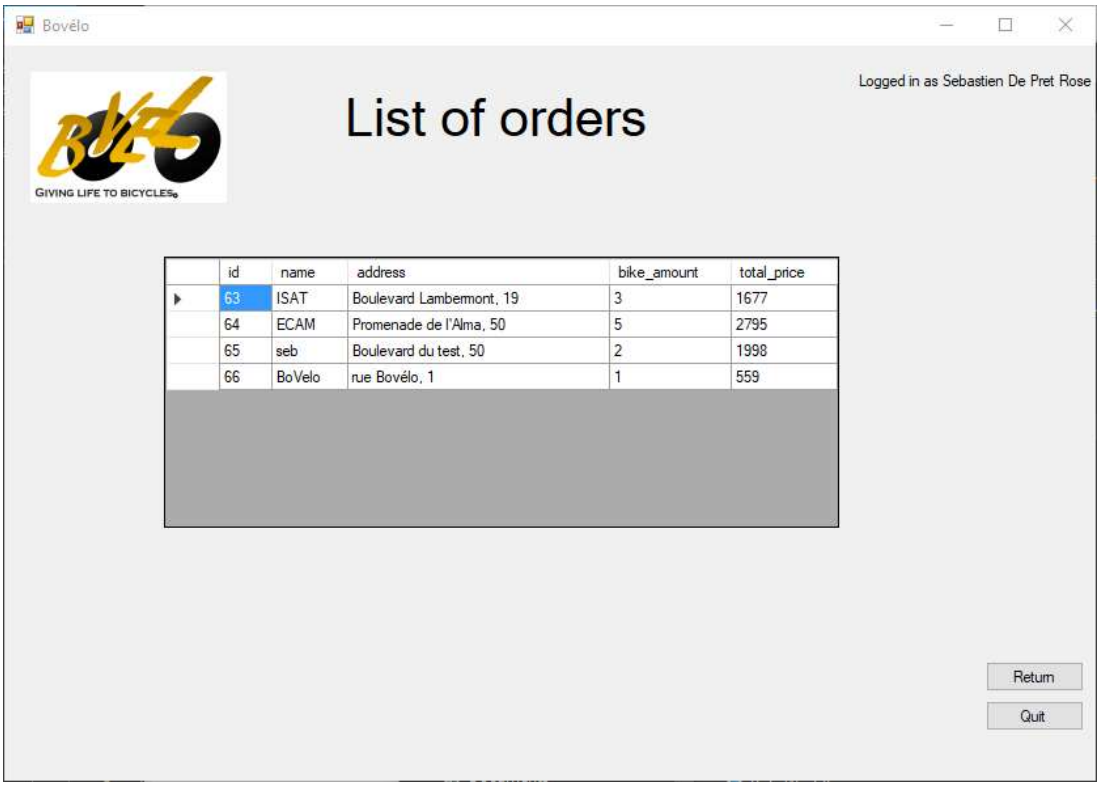
After logging in, the user is taken to the Main Menu, where he can choose one of the four Sub-Menus to access different parts of the application. The user can also logout or simply quit the app.



The catalog is used mainly by a Bovélo representative to show all the bicycle available for purchase to a client. After selecting the type of bike wanted, the client can further choose a color, size and eventually order several bicycles.

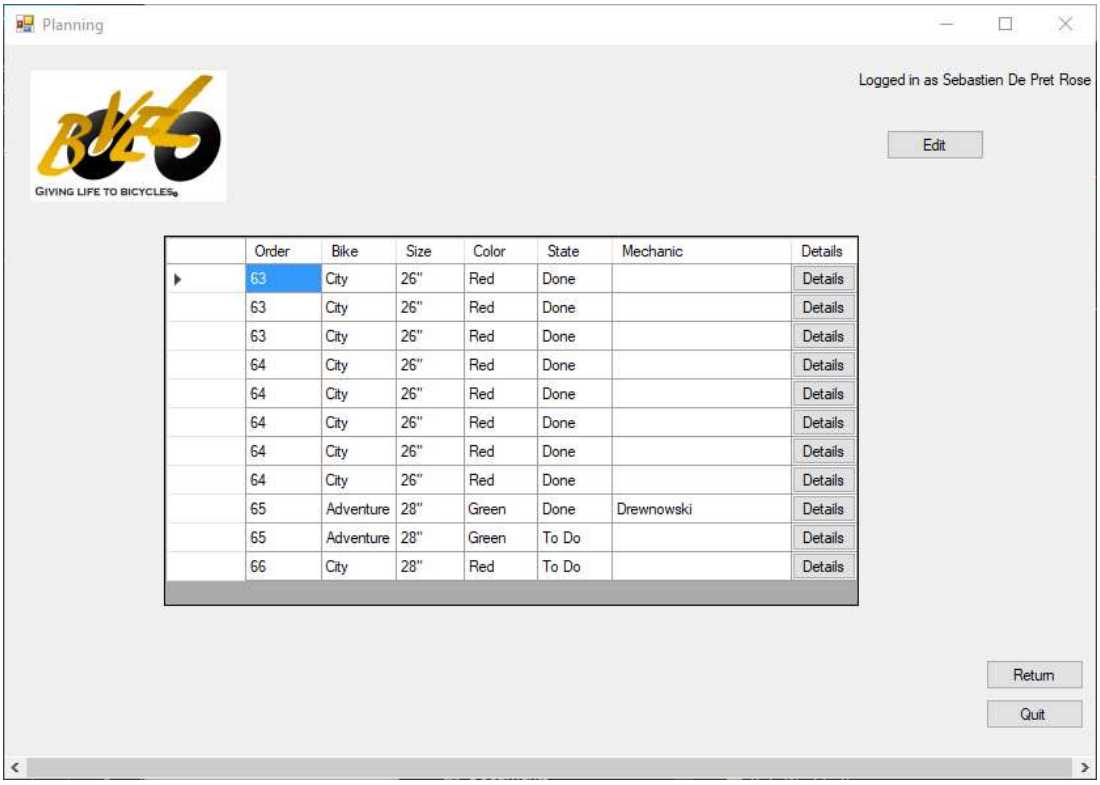


The list of orders lets the user see how many order are to be processed, along with the client information, the quantity of bicycles ordered and the total price of the order.



id	name	address	bike_amount	total_price
63	ISAT	Boulevard Lambermont, 19	3	1677
64	ECAM	Promenade de l'Alma, 50	5	2795
65	seb	Boulevard du test, 50	2	1998
66	BoVelo	rue Bovélo, 1	1	559

The planning window is used by Bovélo's mechanics to let them see at a glance the bike that needs to be assembled, with the relevant bike information and their state of production. The production manager can edit the priority of a task by reordering them with the *Edit* button.

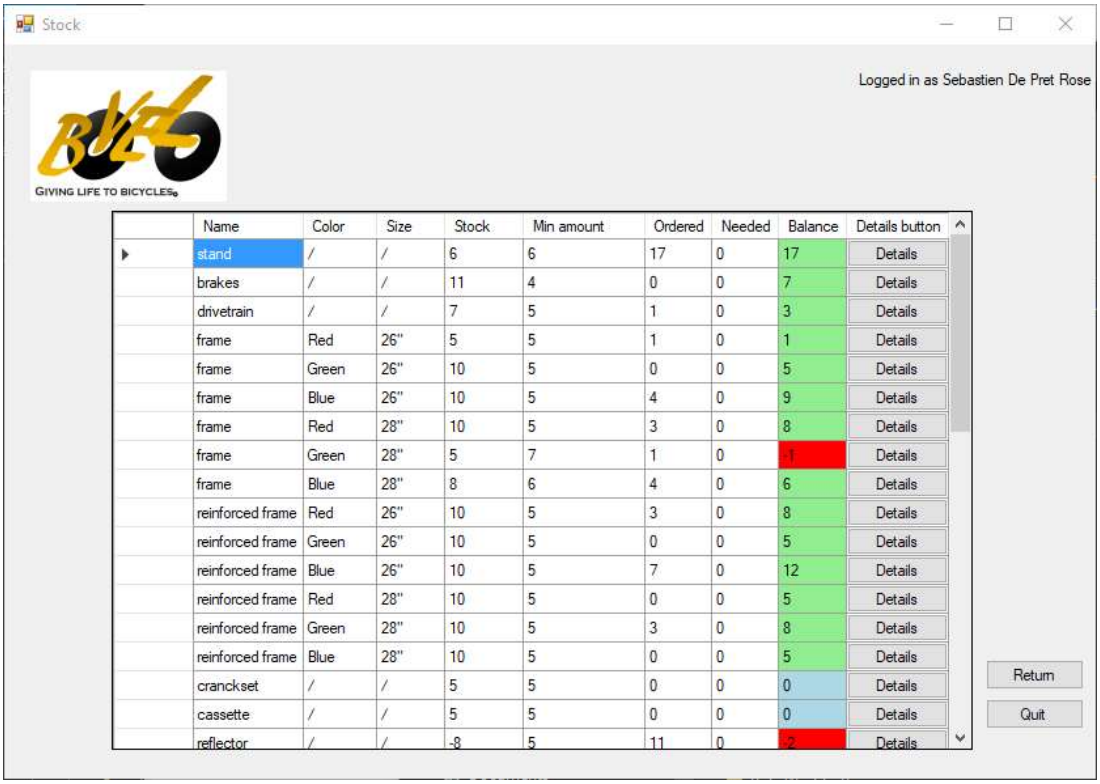


Order	Bike	Size	Color	State	Mechanic	Details
63	City	26"	Red	Done		Details
63	City	26"	Red	Done		Details
63	City	26"	Red	Done		Details
64	City	26"	Red	Done		Details
64	City	26"	Red	Done		Details
64	City	26"	Red	Done		Details
64	City	26"	Red	Done		Details
64	City	26"	Red	Done		Details
65	Adventure	28"	Green	Done	Drewnowski	Details
65	Adventure	28"	Green	To Do		Details
66	City	28"	Red	To Do		Details

Finally, the stock window lets the user know the state of Bovélo's stock for each parts used to assemble a bicycle. Its name, color and size are displayed as well as the amount currently in stock, the minimum amount required in stock at all time, the quantity ordered to external parts provider and the amount needed for bicycle that are to be assembled. The balance column display the current balance of each parts and give the user information on whether he should urgently order more part or not. The balance is calculated with the simple formula:

Balance = (Stock + Ordered) - (Minimum amount + Needed)

A Details button allows the user to easily access more details of the part and edit the information manually.



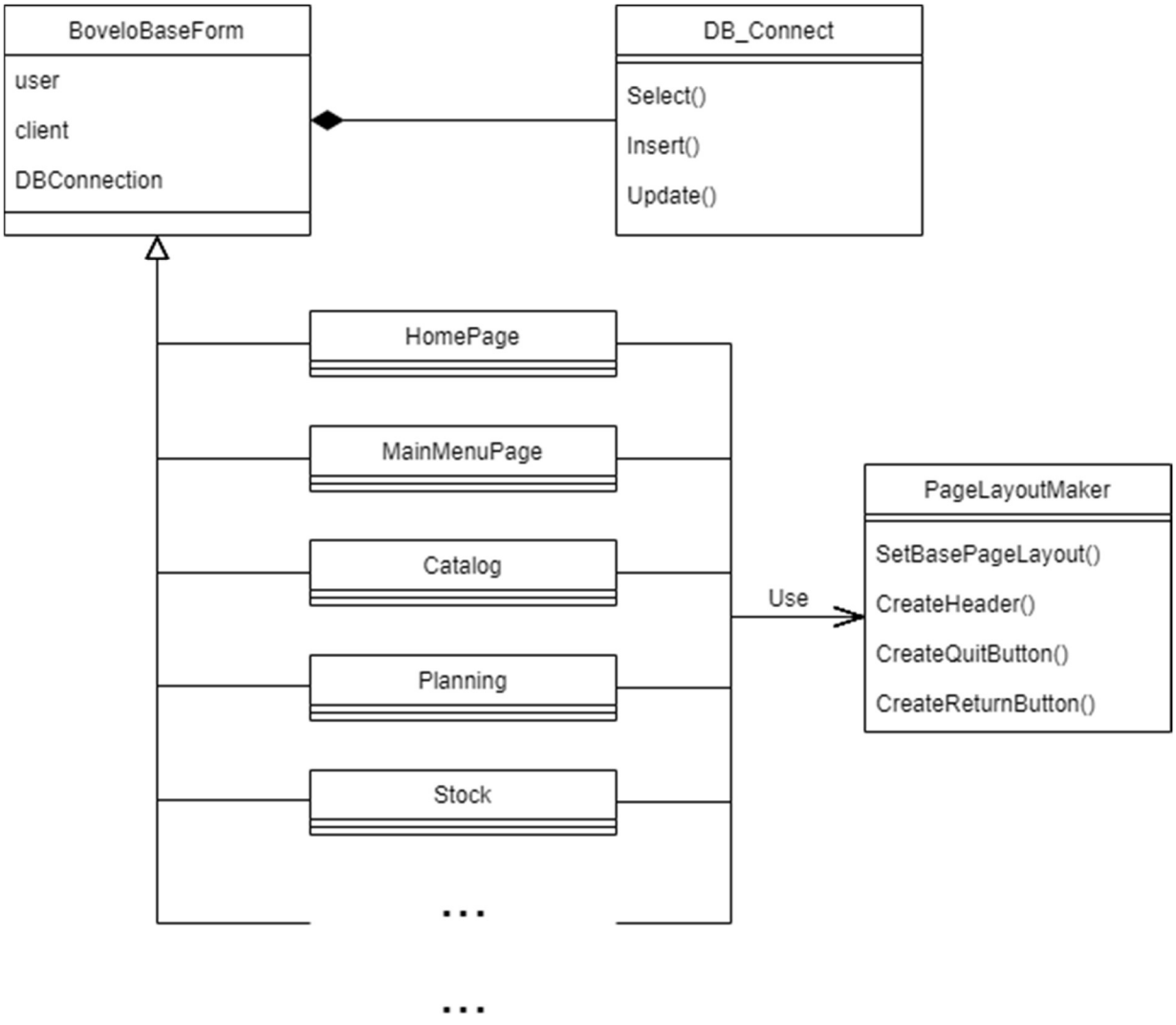
Name	Color	Size	Stock	Min amount	Ordered	Needed	Balance	Details button
stand	/	/	6	6	17	0	17	Details
brakes	/	/	11	4	0	0	7	Details
drivetrain	/	/	7	5	1	0	3	Details
frame	Red	26"	5	5	1	0	1	Details
frame	Green	26"	10	5	0	0	5	Details
frame	Blue	26"	10	5	4	0	9	Details
frame	Red	28"	10	5	3	0	8	Details
frame	Green	28"	5	7	1	0	-1	Details
frame	Blue	28"	8	6	4	0	6	Details
reinforced frame	Red	26"	10	5	3	0	8	Details
reinforced frame	Green	26"	10	5	0	0	5	Details
reinforced frame	Blue	26"	10	5	7	0	12	Details
reinforced frame	Red	28"	10	5	0	0	5	Details
reinforced frame	Green	28"	10	5	3	0	8	Details
reinforced frame	Blue	28"	10	5	0	0	5	Details
crancket	/	/	5	5	0	0	0	Details
cassette	/	/	5	5	0	0	0	Details
reflector	/	/	-8	5	11	0	-2	Details

2.2 Software architecture

2.2.1 Windows form class diagram

In order to create all the different windows accessed by the user without repeating a lot of code for each page, we created a mother Form class, *BoveloBaseForm*, containing the user and client field, as well as the implementation of the DB_Connect class, used to send queries to the database. All the Forms displayed in the application inherit from this class.

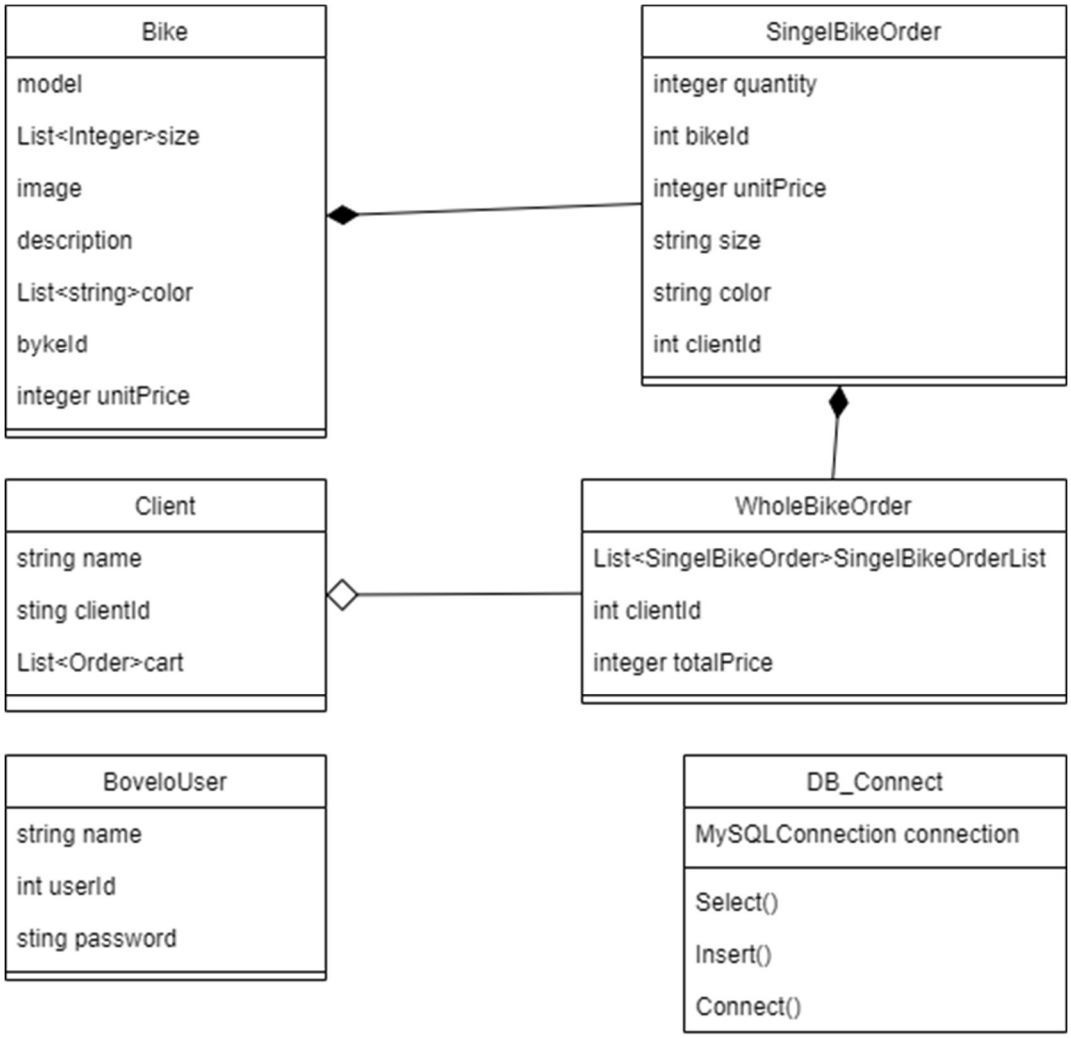
An other class used to reduce the amount of code is the static *PageLayoutMaker* class, which is used by all Forms to streamline the layout of each components of the page, such as the size of the page, the header and its content, the return or quit button, etc…



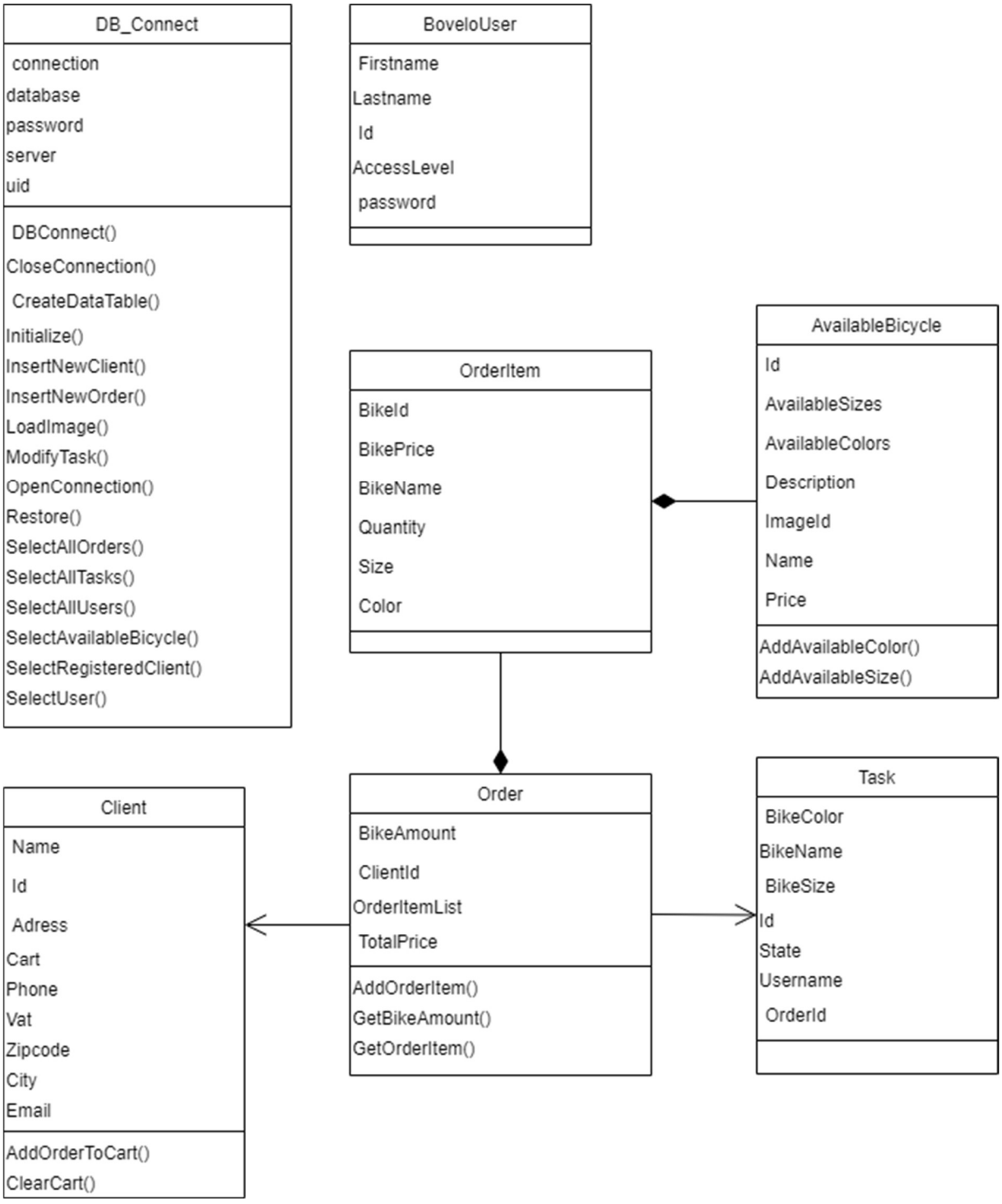
2.2.2 Database objects class diagram

In order to easely select objects from the database into the software, and vice-versa (inserting new objects from the software into the database), we have created several classes which are very similar to the tables found in the Database. The class *DB_Connect* is used for all the different SQL queries linking the software and the database. Each iteration saw the addition of new classes.

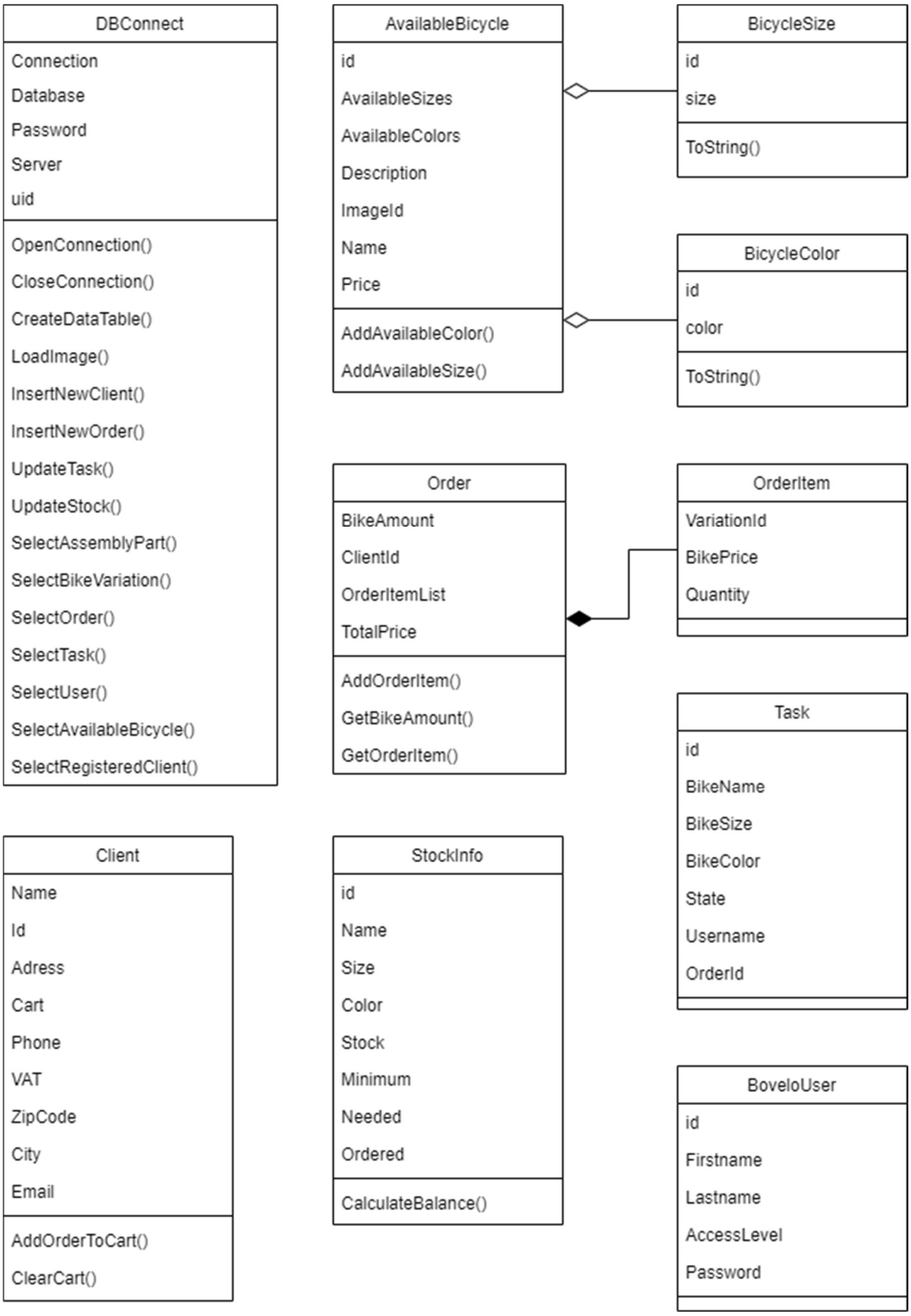
First Iteration:



Second Iteration:



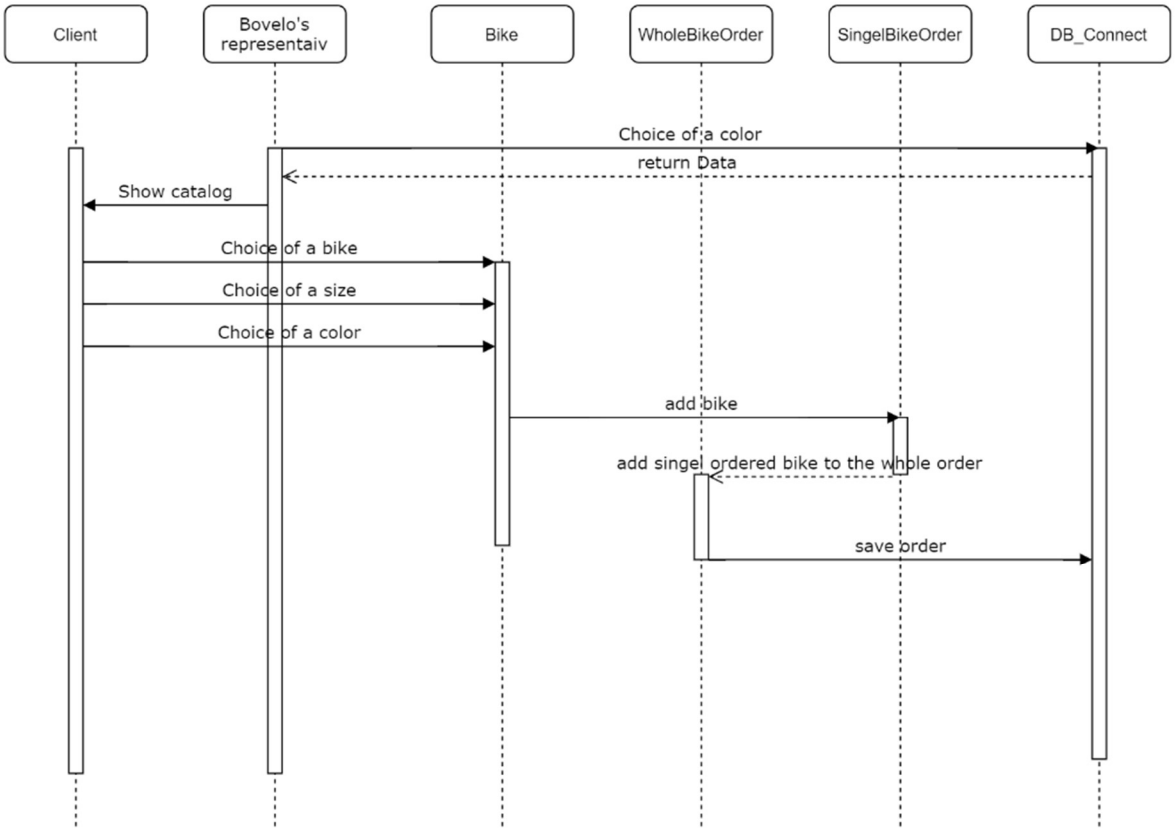
Third Iteration:



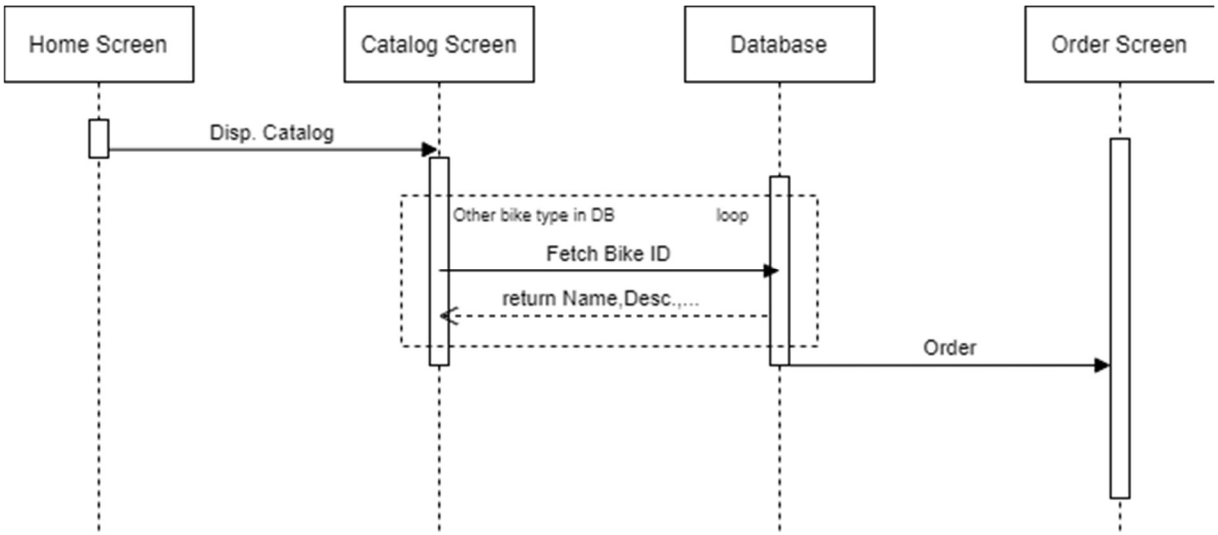
2.2.3 Sequence diagram

First Iteration:

This first diagram shows the sequence of a Bovélo representative showing the available bicycle catalog to a client, who proceeds to choose a model, color and size. Each bicycle chosen is a *SingleBikeOrder*, and all the bicycle the client wish to buy forms the *WholeBikeOrder*. Once the order is accepted by the client, the order is saved in the database.



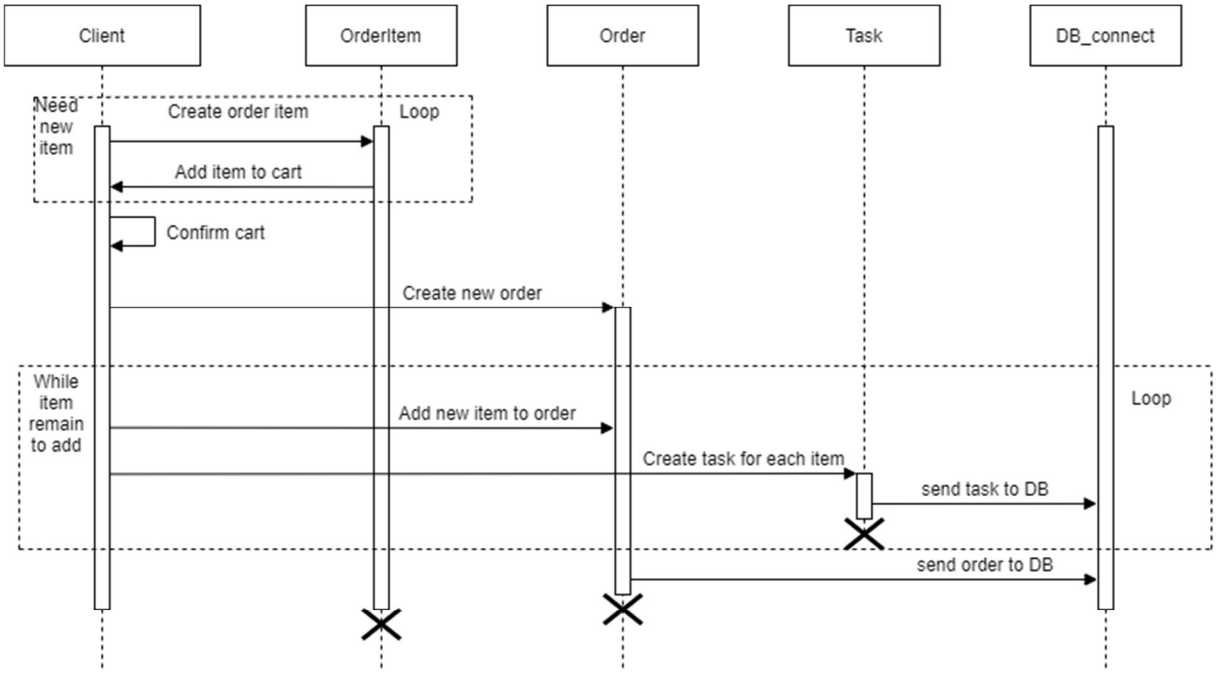
This second diagram shows the step taken to display all orders in the order screen.

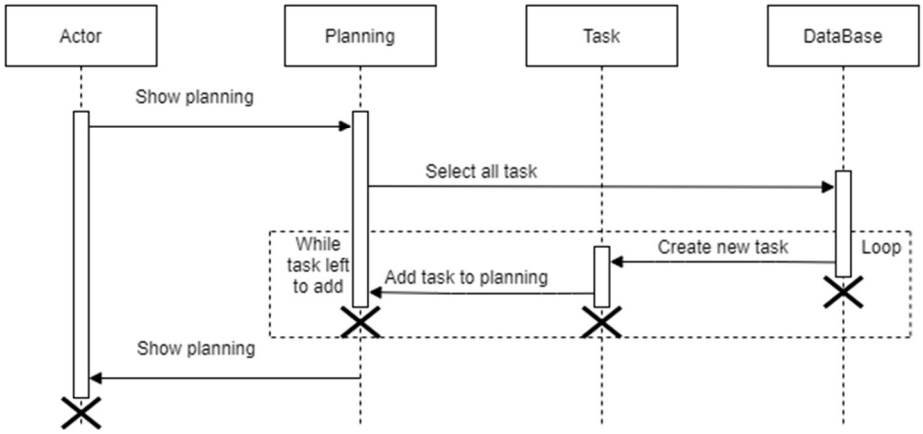


Second Iteration:

The first sequence diagram shows the steps taken in the program when the client pass a new order, selecting several bicycle it wish to purchase (orderItem), which are then grouped as part of a single, whole Order, as well as the creation of the tasks for the planning manager and their insertion in the database.

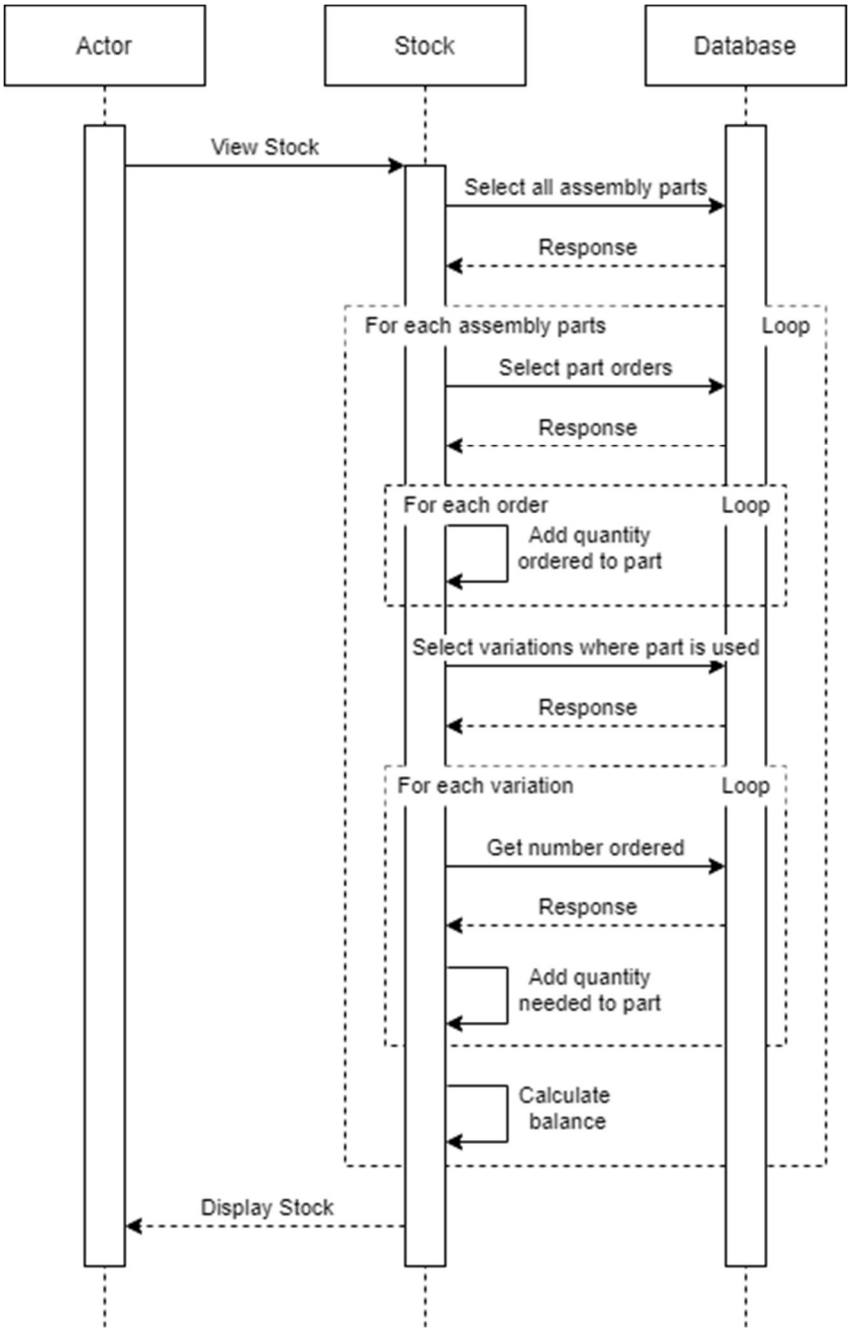
The second diagrams is the sequence the program go trough when an actor wish to see the current planning.





Third Iteration:

This sequence diagram shows the steps taken in order to display the catalog of parts (the stock) to the user. As can be seen, several for loop are imbricated, with many queries sent to the database, which explain why the stocks takes a longer time to load (a few seconds). Fixing this issue might be a target for future improvements.

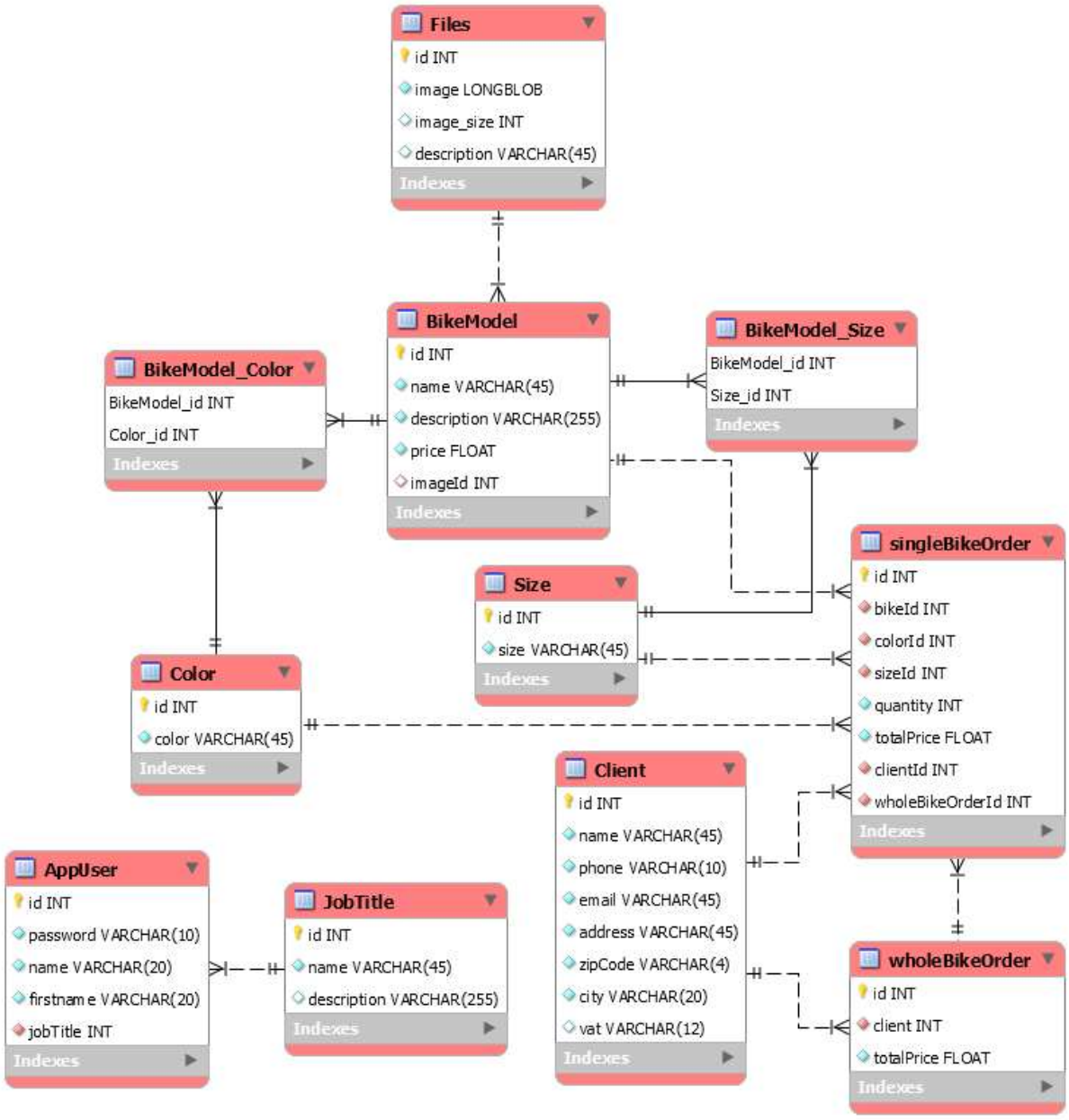


2.3 Database

2.3.1 Entity-relationship diagram

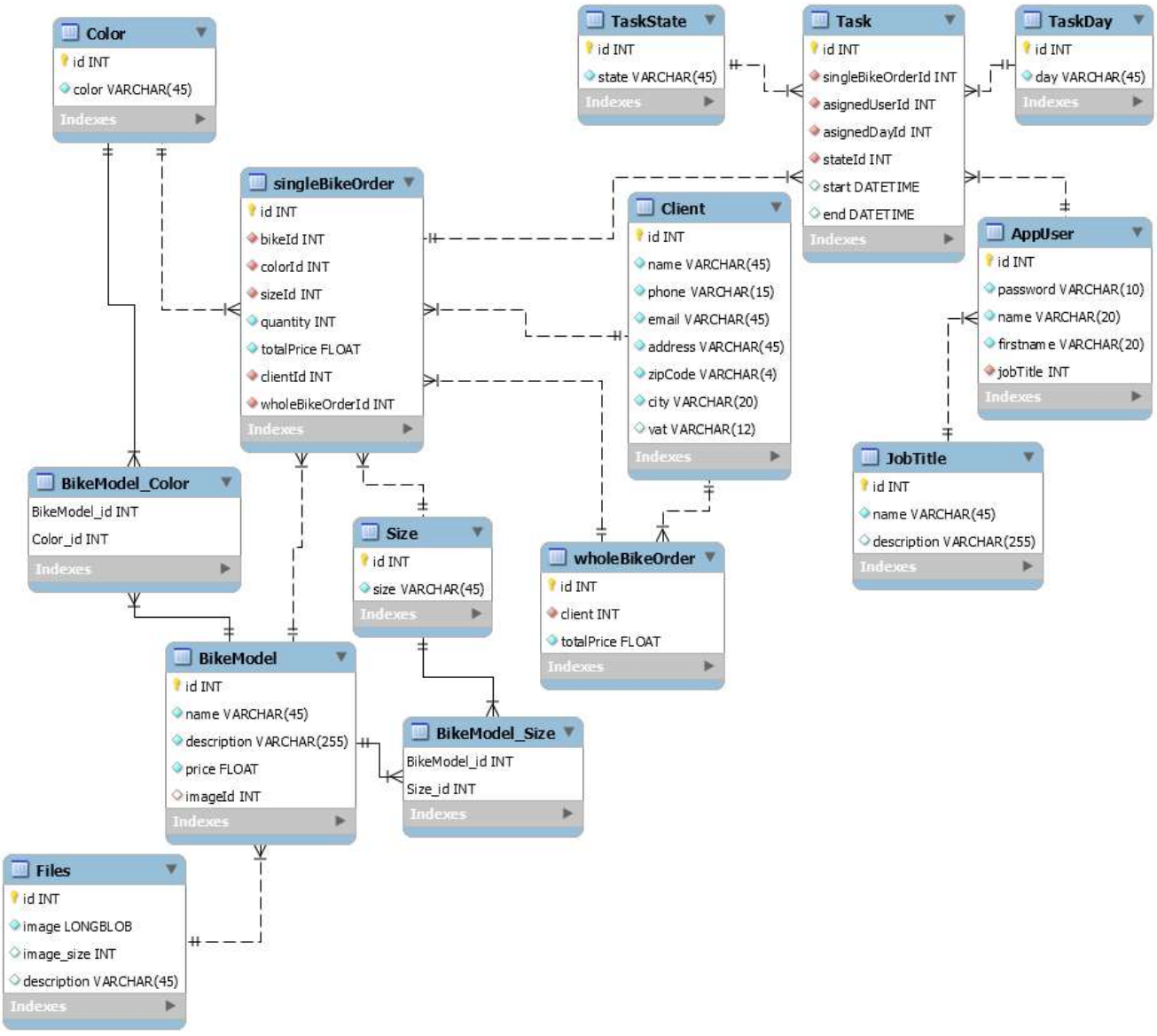
First Iteration:

This diagram represents the structure of our database. In the first iteration, we had to manage bike catalog and orders. To achieve that we must specify the available models, sizes, colors by creating several tables (Color, Size, BikeModel). Since those informations are correlated we used linking tables (BikeModel_Color and BikeModel_Size).
We also made a table called Files which contains images of the bikes in a long blob type file.
To make an order, we needed client infos (Client) and we made a table called singleBikeOrder which contains one bike configuration ordered with the selected model and information about it (model, size, color, price, …). WholeBikeOrder is the final order and contains all the configured bikes, the final price and a link to the client.
The table AppUser stores users credentials, unique ID and a link to their job function (JobTitle).



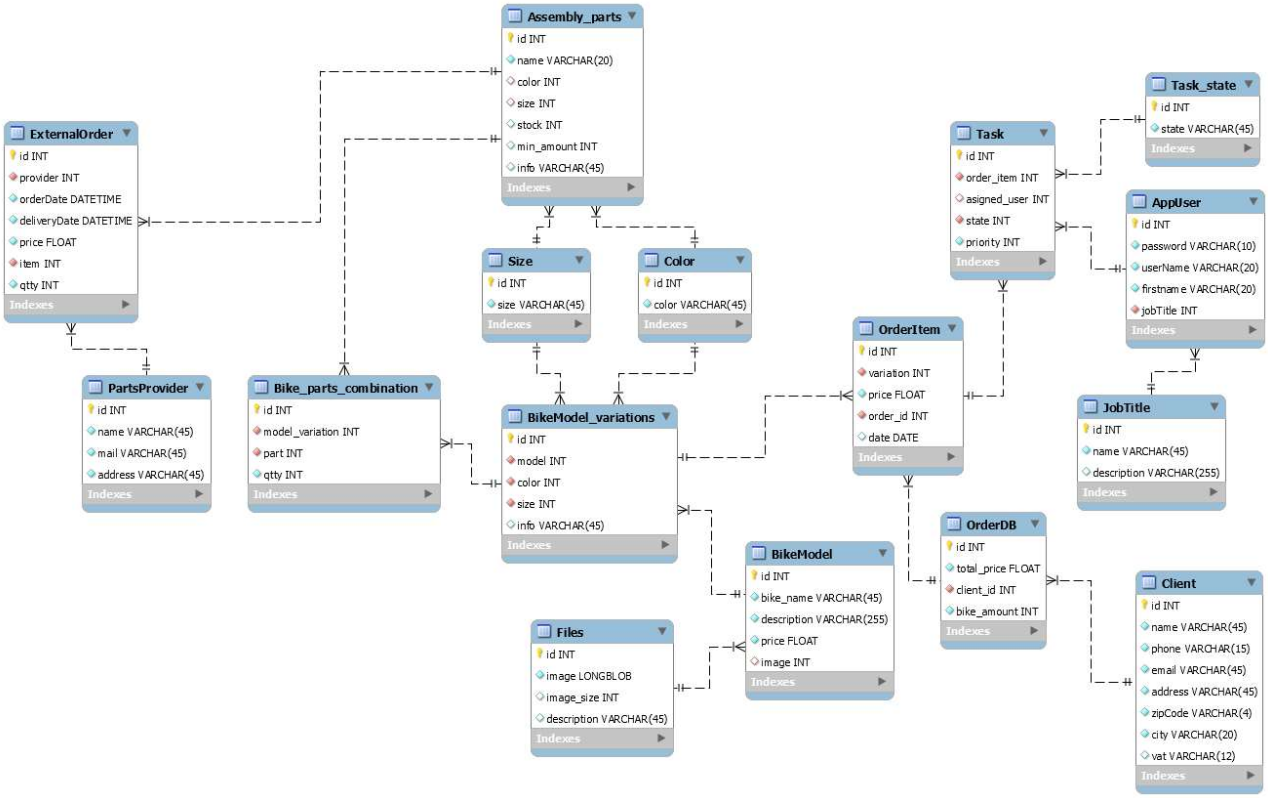
Second Iteration:

Our goal for the second deliverable was the implementation of a production planning. We added a table called TaskDay to track the day of the week when you create a task and TaskState with the valid states of a bike assembly. Task table is linked to those two tables and also contains diverse informations like the mechanic who will assemble the bike and a start and end date.



Third Iteration:

In this final version, we added stock management, including external orders for bike parts. The table Assembly_parts has all the pars used to assemble any bike model variation. We needed a link table between all the parts and a configured bike, Bike_parts_combination. The second goal of the last deliverable was to add any order placed by the manager for the parts stock. We decided to add a table with all providers (PartsProvider) and a table with a link to bike parts and other details about the order (ExternalOrder).



In our opinion, the database respects the ACID principle and it is easy to add a new color, size, model or any other element. We tried our best to implement the database normalization as we've seen this year.

3 Conclusion

3.1 Objectives achieved

Most of the objectives set from the start have been achieved, namely:

1. The elaboration of the production planning, which allow the production manager to see the states of the bicycles to be assembled and allows him to easely reorder them.
2. The creation of a stock management window, wich keep tracks of the quantity of parts currently in stock, those needed for assembly and those ordered. It's also really easy to edit the stock in case of a misscount.
3. The creation of an ergonomic bicycle catalog allowing the representative to show any clients the bikes available with all relevant information, wich can easely be ordered in any quantity.

Altought we managed to add the date at wich new parts ordered where expected to be delivered, we weren't able to use this information to show the expected time a new bicycle would take to be assembled.

3.2 Future improvements

Completing the last and final objective, whish is to give the client an estimate for the date his order will be completed.

Improving the way queries sent to the database are handled, to optimise the program and reduce the time taken to load some pages of the application (especially the stock window).

Another possible optimizations could be to implement a new way to edit our bike catalog. At this moment, we use MySQLworkbench to edit the available sizes, colors and models and add images of the bikes. It would be nice to do it directly from our software.

3.3 Teamwork

As a whole, the group cohesion was very good, we didn't have any internal conflict among us regarding any issue we faced during the project, and managed to always keep civil our discussion of the best course of action to take when starting a new part of the development. If there was one inconvenience to note, it would be the large size of our group. We all felt like seven people working on a project that would have been easely done with five person was a bit much. Still, we managed for each iteration to find something to do for everyone, with no one left behind having no goal, altough some had a bit less to do than other.

As per the tool used for our collaboration, we each had our task clearly defined on Azure DevOps at the very start of each iteration, with no major deadline issue, and merging our work on the GitHub repository proved to be very easy, altough we had some issue at first linking our commit with the corresponding Azure DevOps tasks.