
# Hubbl, a gym bookings manager

Miquel de Domingo i Giralt

# 1 Introduction

On November 2019 the first cases of a new virus were coming into light. With similar symptomps as a regular flu, nobody would have imagined that the humanity would be at the edge of collapsing. The weeks went by and within few months, most countries of the world were completely shut down. It was not until the summer that people was able to go outside, again. Restaurants, hotels and any business that relied on the clients for their survival, had to face many restrictions imposed by the governors. One of such business where the gyms. The owners had to limit the total capacity of their installations and classes. Most companies had to find a new system which helped them handle appointents, since nearly any of them had an appointment system integrated in their respective applications.

# 2 Project goals

The goal of this final degree project is to develop a full stack application, which includes:

- Designing the database.
- Implementing an API in order to interact with the database.
- Briefly designing the user interface.
- Implement the required front end applications, web based, in order to access the system.

Fundamentally, the hubbl application will allow the creation of gym zones, which are explained in the next sections, which will allow the clients to create the needed appointments.
As a future implementation yet not being a priority, the system will also have:

- An analytics page which would provide information about the statistics of the gym.
- A subscription system for the clients.

# 3 Functionalities

## 3.1 Product description

The product will provide a rich interface for the gym owners and their coworkers in order to manage their gym. Managers will create virtual gyms, each of one with different constrains, which will be forwarded to their gym zones. Each gym zone will be of a certain type, for example, a cardio zone, a free-weight zone or even a powerlifting zone. Such and more characteristics will be determined by the workers or who is responsible for the creation of the zones. Once determined the capabilities of each zone, the clients will have the possibility to make their reservations.
This software is going to be interesting since it is mainly focused on the managing of appointments. There exist multiple manager systems, some are gym-focused yet there are few that provide an exclusive focus to managing the appointments. It is interesting to have an application that is that specific because most of the companies already have their client management system and other tools. The COVID has changed many things extremely fast and some gymnasiums have not been able to adapt fast enough. That is because using other management tools would mean to change the entire managing software of the company, in order words, starting from zero again. In the future, however, it would be nice to provide a client management system aswell, including subscriptions and so on.

## 3.2 User needs: Owner or worker

Such persona needs entire access to the management system, and it is the main user of it. For this persona, the following needs can be defined:

- *Ability to create virtual gyms and gym zones*. It is the main purpose of the application. It has to provide all the tools that are required to have an above average managing system.

- *Modify virtual gyms and gym zones*. Constraints may change during time. Overall capacity may increase, more cardio machines may be added, and, with these changes, the gym zones will have to adapt to such real life modifications. There has to exist an interface that provides an easy and simple tool to control it.
- *Ability to manage clients*. The income of the gym is based on the amount of clients they have, and such clients have to be subscribed into the system. However, the application is not a client management system. It purely focuses on the fact of appointments, yet the client still has to be registered in the application.
- *Ability to manage appointments*. The workers have to be able to create, modify or delete the appointments at any time. This process has to be fast and simple, since it is more than usual to have cancellations, clients without reservation, and many more.
- *Ability to manage guided events*. Some gym zones will be marked as a guided class zone. Therefore, a schedule will have to be set up for such zone in order to let the user know that.

Furthermore, two user profiles have to be differentiated: the *owner* and the *worker*. The owner needs more control than the worker and some additional needs are[1]:

- *Ability to manage workers*. The owner has to be able to add, remove or update their workers. There must exist an interface that allows such control.
- *Ability to manage the privileges of the workers*. In large gymnasium franchises, there may exist different types of workers, each with different tasks. For instance, some may only be able to manage guided sessions, others will manage the client subscriptions and so on. Therefore, a privilege system is required to define a hierarchy in the company.
- *Ability to manage gym trainers*. It is important to know what trainers will be responsible for each guided class. For instance, some clients may prefer some trainers than others, when choosing a guided class.

### 3.3 User needs: Client

The client has little interaction with the system. However, a management application for clients would be pointless if clients could not book their sessions. That is why it will exist another platform to provide access to the client. Such client, will have the following needs:

- *Ability to create, modify and delete appointments*. The clients will make the most use of the reservation system. There has to exist a simple and intuitive interface that allows the clients to interact with the system.
- *Ability to visualise guided events*. In case the client prefers booking a place for a guided class, it should be able to visualise all the possible events for a day or week, for a zone.

## 4 Development methodology

To structure and organise the project, working packages have been used in order to ensure a temporised and structured development. Following such structure has been extremely useful to simplify how are tasks managed and temporised. In order to organise the tasks, an *AGILE* approach has been used, even though the project has been done all by myself. Each subgroup of the *development* branch has been considered as an *epic* and each working package as a *story*, which has as many tasks as required. This approach has been used for each project and has made the development of the project easier to manage.
In order to simplify the process of *epic*, *story* and *task* creation, the Jira software from Atlassian has been used. It has been extremely useful as it can be integrated with many tools, one being GitHub, where the code is hosted. Using these elements, I have been able to automate the process of starting and completing tasks, alongside of the Jira automation. The following diagram, describes the workflow of the tasks inside Jira:

[1] Some workers may also have access to such capabilities, depending on the permission they have recieved by the owner.

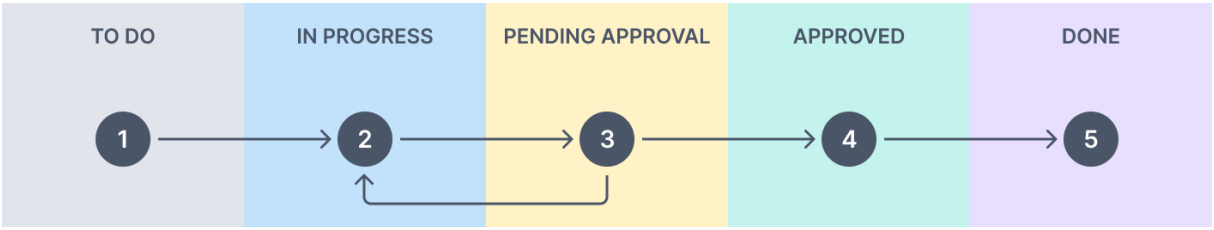


**Figure 1:** 5 steps of the Jira workflow

All the tasks would have to go from step 1 to step 5 in order to be considered as finished. Each step and the automation is as follows:

1. All the tasks that have been selected for the sprint start at the first step, at the *to-do* column.
2. For each task, a new git branch has to be created with the name of the Jira task tag, for instance `HBL-185`[2]. When the branch is created and pushed to GitHub, Jira will move the task automatically to the *in progress* column.
3. While in progress, each commit made is tracked in Jira. Additionally, Jira also supports *smart commits* which are words prefixed with a hash (#). With the smart commits, task properties can be changed. In this case, only the `#time` has been used which increments the amount of time spent in each task. Tracking time has been useful to know more precisely how much time has been spent overall.
4. Once the task is finished, a new pull request is made which triggers the continuous integration workflows set up at the GitHub repository. These workflows can also be watched by Jira and automate operations in function of the workflow result. If the task does not pass one of the workflows, the task is moved back to the *in progress* tab. Alternatively, if the checks pass, it is moved to the *approved* column.
5. Finally, in order to be considered *done*, the pull request of the task in question has to be merged. The merge will trigger another automated process which moves the task from *approved* to *done*.

In order to stick to the process as much as possible, tasks moved to *done*, should not be moved back. Instead, a new *bug* type task was created which would start the process again.
In relation to the environment process, the project has been built using TypeScript, from the server to the different UI applications. The environment has been created using a monorepo build tool called Nx, which simplifies the monorepo management for JavaScript projects.

## 4.1 Final product

Here are some images of the core application:

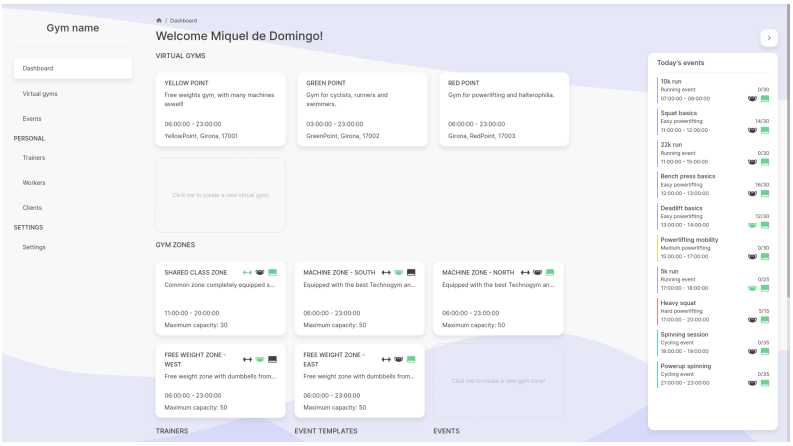


**Figure 2:** Dashboard of the core application from owner's view

[2]The `HBL` prefix is assigned when a new Jira project is created.

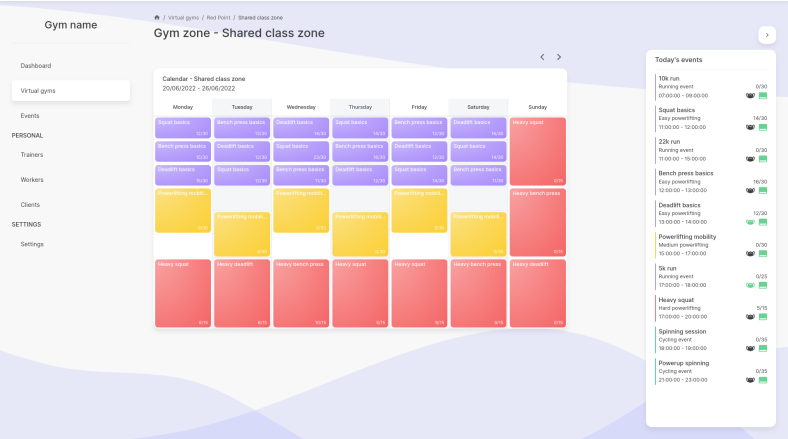


**Figure 3:** Calendar of a gym zone from owner's view

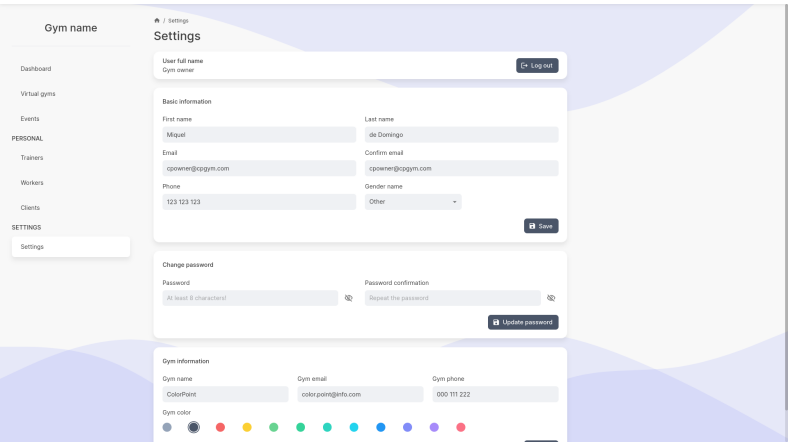


**Figure 4:** User settings from owner's view

# 5 Conclusions

The goal of the project was to provide a tool that helped both gym owners to manage the appointments their clients made, durign the COVID restrictions. This objective has been accomplished with the Hubbl web application, which allows users to manage appointments in a gym environment. Nevertheless, it is hard to define an app as completely finished, as there may exist a bug that the developers have not seen, or the clients will require new features which will have to be implemented.

It is important to note that one of the objectives of the application was to provide an elegant and sofisticated user experience, at least better than my experience as a user of a similar software. Therefore, the Hubbl application not only provides value with its functionality, but also with the interface and use experience it ships with.

Furthermore, the Hubbl application tries to cover as many aspects as possible, while simplifying the usage of the application. However, there are many other things to improve and to add as new features. Nonetheless, the version that is provided has more than what the app is expected to do. Due to the lack of time, inexpertise in some fields or tools used, and the fact that I have only been developing the application, it has not been possible to develop all that was expected for the wanted deadlines.

The project has required me to use pieces of knowledge acquired in the degree. At the same time, I have been pushed outside my comfort zone, dedicating time to reading documentation pages and similar.

To sum up, the project has provided a sense of how difficult and complex can it be from brainstorming the idea of the application, to finding the proper architecture to later develop it.