Hubbl, a gym bookings manager

Miquel de Domingo i Giralt

April 7, 2022

Some dedications to the people I care for.

Contents

List of Figures

List of Tables

1. Introduction, motivation, purpose and project goals

1.1 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.

1.2 Motivation

As a front-end engenieer and considering myself someone who really enjoys designing and later developing user interfices, I realised that I had a great oportunity in front of me in order to test myself. At my current job in Additio, a company which develops applications for the teachers, I have had the opportunity to start learning more about UI/UX. Nevertheless, I had to attach to creatain rules, in order to respect the design system of the application. Now, however, I had the opportunity of being responsible for each part of the process of developing an application: from designing the database and impementing the API, to designing the UI and implementing such design. Without constraints, I would have try and explore as many technologies and systems as I felt like.

Furthermore, as a regular user of my gym's booking application, I had few issues with it. Even though I have only experienced one side of the application, as a client, I believe that some parts could be improved, as in anything. One of such parts are both the user interface and the user experience of it.

1.3 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.

2. Methodology

2.1 Introduction

In this chapter, it will be briefly explained what project management will be utilised and the development methodology. Both decisions are crucial and will have effect in how the project is planned and evolves.

2.2 Project management: *PMBOK*

The methodology used to develop the overall project has been the *PMBOK* which has been explained in one of the subjects of the degree. Such methodology ha become a standard in the project management world. Since it is considered as the book of books when it comes to project management, it is definitely useful to be used in any project, which, most likely, will require some sort of management.

The PMBOK methodology is explained in the *Project Management Body of Knowledge* book, in which such standards and guidelines are explained more in depth. The book is produced and updated by the *Project Management Institute* (*PMI*) and it currently has 6 editions, the latest one released in 2017.

The procedure is based in five process groups, which are:

1. **Initiating**: the initiating processes are those which are performed in order to define the project or an upcoming phase of an existing project, and obtain permission to execute the project or phase.
2. **Planning**: the planning processes are those which are required to clearly define the scope, the objectives and the course of action of the project.
3. **Execution**: the planning processes are those which are performed to complete the work that has been defined in the previous processes in order to satisfy the project specifications.
4. **Monitoring and controlling**: the processes involved in this group, are required to track, regulate and review the progress and performance of the project. It identifies areas in which the plan has to change and initiates the corresponding changes.
5. **Closing**: the closing processes are those which are performed in order to finalize all activities across the above process groups so the project or phase can be closed.

2.3 Development methodology

To structure and organise the project, working packages[1] 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 temporized. In order to organise the tasks, an *AGILE*

[1] See Working packages for more information.

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. The GitHub integration provides a lot of utilities, yet one of the most important is the fact that it tracks your branches, commits and pull requests. Using the three 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:

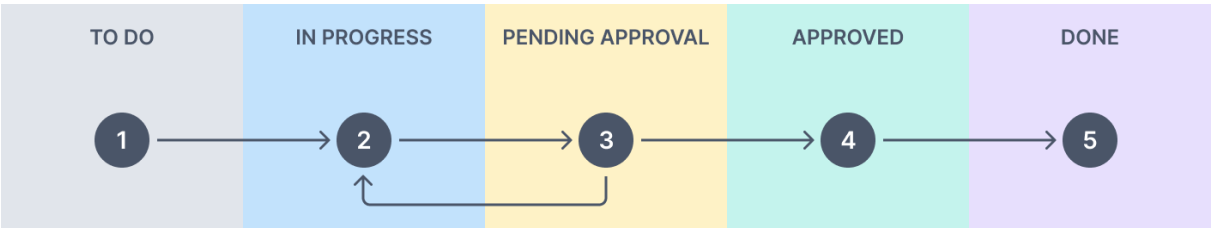

Figure 2.1: 5 steps of the Jira workflow

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*.

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

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.

3. Requirements specifications

3.1 Introduction

This chapter will cover the system requirements specification (*SRS*) for the software being developed. The goal of this chapter is to establish the basis for what will be developed, taking into account user needs, functional and non-functional requirements.

3.1.1 Purpose

The application's objective is to provide a simple, scalable and powerful platform to handle gymnasium appointments. Gym owners will be able to create their gym zones and modify the constraints such as total capacity, available hours, closed days as needed. On the other side, gym clients will be able to book an appointment to the gyms they have access to.

Therefore, in a single application, gym owners and their workers will be able to control everything that happens inside the gym; while at the same time being able to apply changes and modifications as wanted.

3.1.2 Definitions

There are some definitions to be explained before moving to the next section:

- *Gym*. The gym will represent the company overall, not the infrastructure. Such infrastructure is called *virtual gym*.
- *Virtual gym*. A virtual gym will be the representation of a gymnasium in the application. A *virtual gym* can have multiple *gym zones*.
- *Gym zone*. A gym zone will be the places where clients will book appointments.
- *User*. The *user* is the main person who uses the software. In this case, it is the owner and their workers.
- *Template events*. Even though the class concept is explained in further sections, it is important to make a distinction between a *class* and *class template*, *template events* or *template events* (as they are named in the database). On the one hand, events will be assigned to a gym zone and scheduled. On the other hand, *template events* will be used to schedule and link to a zone.
- *Archiving* vs *Deleting*. The purpose of archiving anything in the database means it will be stored but not visible. In order to edit it again, the archived element has to be *recovered*. On the other hand, a deleted element will permanently be deleted from the database.

3.2 Overall description

3.2.1 Introduction

This section provides a general explanation about the application, how it is intended to work, the different *personas* to whom it is focused and a brief introduction to the software's functionalities.

3.2.2 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.

3.2.3 User needs

After having briefly defined the application, two *personas* can be identified: the owner or worker of the gym, and the client.

In the following sections, a brief explanation will be given about the functional requirements, yet such requirements will be explained more in depth in later sections.

3.2.3.1 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.

3.2.3.2 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.

3.3 Product functionalities

The goal of this section is to explain how would the basic flow of the application be. From the owner arranging virtual gyms, the events and assigning the trainers; to the client being able to manage their appointments. By exemplifying the application behavior, it will be easier to determine the functional requirements of the software.

3.3.1 Gym

When registering, the owner will be required to enter the gym information and will be able to customise other information such as the gym contact data. Such settings will be common to all clients and workers, as they will be part of such gym.

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

3.3.2 Virtual gym and gym zones

The interface has to be intuitive both for the user and the client, while at the same time providing as much utility as possible. Before being able to create appointments, the client needs a place to create such appointments. Therefore, the first mission of the owner is to define each zone. A gym company may have one or more virtual gyms. As stated before, a virtual gym is the representation of the gym infrastructure in the system. Each virtual gym will have different characteristics and information as: the gym location, the total capacity, the opening and closing hours and much more. This virtual gyms can be modified as they may change time to time. If ever needed, the fact of deleting the different virtual gyms will also be possible.

Once the virtual gym has been set up, it can have gym zones. Each gym zone will also have its characteristics, most of them similar to the virtual gym ones. However, there is a characteristic that is important to mention: the *type* of the gym zone. Each zone is required a type since some zones may be suitable for events and some not. Such types will be predefined, however the user will have the ability to create their own types. Nonetheless, there will still be the need of distinction between class zones and non-class zones.

3.3.3 Events, trainers and workers

After having defined the structure of the gym, the owner or the workers of the gym can create and schedule events for the different gym zones. Events can not be overlapped in the same zone, yet some events can be scheduled at the same time in different zones. Furthermore, the same class can be done in multiples zones, with different trainers.

In order to keep such consistency, the gym owners will create *template events*, which will have specific characteristics. Each template class will be unique, and it will be used to create *events*. In short, the client will create an appointment to a class, which will be a scheduled template class. Nonetheless, appointments will also be required for non-class gym zones, as it is needed to know how many users will access such zone in a certain time. With the explained relation, owners and workers will not have the need to create a class with all the details every single time, rather simply scheduling already created template events.

Additionally, a trainer manager system is required. However, a trainer it is a worker as well, only it does not interact with the application. Hence, in the same management view, the owner, and if any worker is allowed to, will be able to create trainers and workers.

3.3.4 Clients

Before continuing, the system still lacks of the main income of the gymnasium: the client. As detailed before, the goal of this application is not to provide a client management system, though it could be a functionality to be implemented in the future. Therefore, the client will also be related to the gym entity.

There has to be, definitely, a small client management. However, this management does not need to know about all the information it is kept about the user in the gym. It will only need:

- Personal information such as the name, last name and so on. It should include an email to log in to the software.

That is it. From there, the client will have access to another view of the application in which they will be able to make the reservations.

3.3.5 Creating appointments

When the client has been able to connect to the platform, they will be able to visualise the virtual gyms of the gym they belong to. There, the client will see the availability of the different gym zones, the events offered in that zone and other relevant information. From there, they will create their appointments which will be persisted in the system.

After the appointment has been made, the client will be able to modify it and remove it, without the need of any explanation.

3.4 Functional requirements

3.4.1 Introduction

In this section, the above described requirements will be explained more in depth. The functional requirements are defined as the core functionalities of the system. All requirements exposed in the section are considered essentials and the system would be considered incomplete if it did not satisfy such requisites.

The requisites will be exposed with their code name and a number (as *FR-1*), their priority and a brief description. Three levels of priority have been defined:

1. **High priority (3)**. Such requisites must be fulfilled by the application in order to provide a proper user experience.
2. **Medium priority (2)**. Such requisites should be fulfilled by the application in order to provide an above average user experience.
3. **Low priority (1)**. Such requisites would provide an excellent user experience, yet they are not mandatory.

Furthermore, the overall application will be separated in three different front end applications:

1. **Landing app**. Application which contains the landing application of the product.
2. **Core app**. Main application of the product, where the user does most of the work.
3. **Appointments app**. Application used by the gym client in order to book their appointments.

3.4.2 Landing app - Product information

The application must have a landing page in which the potential user can find the different features offered by the application.

- **FR-1 (3)**. The user needs to know the different features of the product in a single page.
- **FR-2 (3)**. The user needs to be able to be redirected to the register and login page, from the landing page. Such page is the core page.
- **FR-3 (2)**. The user needs to be able to find a contact page in the landing page and ask for the desired information by providing personal information.
- **FR-4 (3)**. The user needs to find information about the data privacy. No personal data and information is intended to be sold, nor shared.

3.4.3 Core app - User registration

The user, in this case the owner, has to be able to register. In the beginning, the application will be completely free, so the gym owners will have complete access only by registering. A future implementation would be to provide additional features only to the users with a subscription.

- **FR-5 (3)**. The user has to provide personal information in order to register.
- **FR-6 (3)**. The user has to provide information about the gym in order to register.
- **FR-7 (3)**. The user has to provide an email and a password to access the application (log in).
- **FR-8 (3)**. The user needs to be able to see how their data is kept (personal data privacy).
- **FR-9 (1)**. The user should see the information for the different premium and free plans.
- **FR-10 (1)**. The user could have a free trial for the premium features.

3.4.4 Core app - Home page

In the home page view, the user will manage their virtual gyms and gym zones. Furthermore, it will be able to create, update and remove the template events which will be then linked to gym zones schedule.

3.4.4.1 Virtual gyms

The user will enter the home page and will see the list of their virtual gyms. If no virtual gym has been created, they will be asked if they want to create a virtual gym (which will be optional). From the same view, they will be able to create, modify and archive the different virtual gyms, aside from accessing the gym zones of that virtual gym.

- **FR-11 (3)**. The view has to display a list or a grid with the different virtual gyms of the user.
- **FR-12 (3)**. Each virtual gym item has to provide an option to visualise and update all the information of that virtual gym. This means changing both the virtual gym *metadata*, and the virtual gym constraints.
- **FR-13 (3)**. The user has to access the virtual gym view by clicking on a virtual gym.
- **FR-14 (3)**. The user has to access the gym zones view by clicking on a virtual gym.
- **FR-15 (2)**. The user has to be able to search for a virtual gym by its name.
- **FR-16 (1)**. Each virtual gym item has to provide an option to archive that virtual gym (and, consequentially, all the gym zones).
- **FR-17 (1)**. There virtual gym list has to provide an option to programmatically sort the virtual gyms (e.g. by ascending or descending capacity).
- **FR-18 (1)**. There virtual gym list has to provide an option to manually sort the virtual gyms.

3.4.4.2 Gym zones

Once a virtual gym has been created and the user has clicked on the card representing it, the user will be redirected to another view in which the zones of the gym will be displayed. Similarly, as in the virtual gyms view, if the user has not created any gym zone, they will be prompted to do so, if wanted.

- **FR-19 (3)**. The view has to display a list or a grid with the different gym zones of the selected virtual gym.
- **FR-20 (3)**. Each gym zone has to provide an option to visualise and update all the information of that gym zone. This means changing both the gym zone *metadata*, and the gym zone constraints.
- **FR-21 (3)**. The view has to display which zones are class type and wich not.
- **FR-22 (2)**. The user has to see the gym zone view by clicking on it.
- **FR-23 (1)**. Each gym zone has to provide an option to archive the zone.

3.4.4.3 Class gym zone schedule

When the user has created a class gym zone, that zone will have a schedule (or a calendar). Inside this view, the user will be able to see the schedule or calendar from the selected gym zone.

- **FR-24 (3)**. The view has to provide an option to visualise the current schedule of events.
- **FR-25 (3)**. From the calendar, the user has to be able to create a guided session by linking a template class to it. Such *scheduled event* must include: a start and end time, the trainer which will be the guide, a maxiumum capacity and other information.
- **FR-26 (3)**. From the calendar, the user has to be able to create an event.
- **FR-27 (2)**. From the calendar, the user has to be able to see future scheduled events.
- **FR-28 (1)**. From the calendar, the user has to be able to see past scheduled events.
- **FR-29 (1)**. The user has to be able to create guided sessions which are repeated when wanted, from the chosen interval.
- **FR-30 (1)**. After creating a new class on a gym zone, the application has to *recommend* a trainer for that class.

3.4.5 Core app - Events page

Even though this view is simple, it is important to separate the schedule view of a gym zone from the template events. The current view will only provide the necessary tools to create, edit, delete or archive the different event templates.

- **FR-31 (3)**. The user has to be able to see the gym list of event types.

- **FR-32 (3)**. There has to be an option in the menu which allows the user to create, update and delete event types.
- **FR-33 (3)**. The user has to be able to see the gym list of template events.
- **FR-34 (3)**. There has to be an option in the menu which allows the user to create, update and delete template events.
- **FR-35 (2)**. The user has to be able to archive the event types.
- **FR-36 (1)**. The user has to be able to archive template events.

3.4.6 Core app - Workers page

This view has to allow the user to manage the workers of the gym enterprise and set the different permissions.

- **FR-37 (3)**. The user has to be able to create a new worker, providing personal information and credentials for such worker to user the app. Furthermore, it has to allow the user to set the permissions for that worker.
- **FR-38 (3)**. The same view has to provide an option to update the permissions and the personal information of the worker.
- **FR-39 (3)**. The view has to provide a table to list all the workers.
- **FR-40 (1)**. The user has to be able to set the working hours of each worker.
- **FR-41 (1)**. The view has to provide an input to search the by name, last name or other characteristics the different workers.

3.4.7 Core app - Trainers page

This view has to allow the user to manage the trainers of the gym enterprise.

- **FR-42 (3)**. The user has to be able to create a new trainer, providing personal information and credentials for such worker to user the app.
- **FR-43 (3)**. The same view has to provide an option to update the trainer personal information.
- **FR-44 (3)**. The view has to provide a table list all the trainers.
- **FR-45 (1)**. The user has to be able to set the working hours of each trainer.
- **FR-46 (1)**. The view has to provide an input to search the by name, last name or other characteristics the different trainers.

3.4.8 Core app - Clients page

This view has to allow the user to manage the clients of the gym enterprise.

- **FR-47 (3)**. The user has to be able to create a new client, providing personal information and credentials for such worker to user the app. Alternatively, the client can register himself (**FR-65**) with a gym code.
- **FR-48 (3)**. The same view has to provide an option to update the client personal information.
- **FR-49 (3)**. The view has to provide a table list all the clients.
- **FR-50 (1)**. The view has to provide an input to search the by name, last name or other characteristics the different clients.

3.4.9 Core app - Settings page

This view has to provide enough settings to customizer the behavior of the behavior and the gym information.

- **FR-51 (3)**. The view has to allow the user to log out.
- **FR-52 (3)**. As an owner, the settings menu has to allow the user to change the gym name and gym characteristics.
- **FR-53 (3)**. As an owner or worker, the settings page has to allow the user to change their personal information.
- **FR-54 (3)**. As an owner or worker, the settings page has to allow the user to change their password.
- **FR-55 (1)**. The settings menu has to provide an option to change the application theme (light and dark) and persist it.
- **FR-56 (1)**. The settings menu has to provide an option to customise the main colours of the user interface.

3.4.10 Core app - Analysis page

The goal of this view is to provide some metrics to *what is happening at the gym*. However, **as it is not the main purpose of the application**, all the requirements from each will be marked with *low priority*.

- **FR-57 (1)**. The view has to provide a select option to choose what virtual gym wants to be analised.
- **FR-58 (1)**. After having selected a virtual gym, the view has to display in which intervals the virtual gym is most crowded at the current date.
- **FR-59 (1)**. After having selected a virtual gym, the view has to display in which intervals the virtual gym is most crowded during an interval.

- **FR-61 (1)**. After having selected a virtual gym, the view has to display what events will have more clients during an interval.
- **FR-62 (1)**. After having selected a virtual gym, the view has to display what events will have more clients during at the current date.
- **FR-63 (1)**. After having selected a virtual gym, the view has to provide a filter to check what gym zones are more crowded during an interval.
- **FR-64 (1)**. After having selected a virtual gym, the view has to provide a filter to check what gym zones are more crowded at the current date.

3.4.11 Client app

The client application will reuse most of the content from the Core application. However, as it is a client, all the functionalities of modifying and deleting will no appear, as the client does not have access to such views.

- **FR-65 (3)**. The client has to provide personal information and a gym code in order to register.
- **FR-66 (3)**. The client has to provide an email and password to access the application (log in).
- **FR-67 (3)**. The home page has to display a list or a grid with the different virtual gyms of the gym they are subscribed.
- **FR-68 (3)**. On clicking any virtual gym, the home view has to display the gym zones of the selected virtual gym.
- **FR-69 (3)**. The calendar should display the scheduled events, week by week.
- **FR-70 (2)**. On clicking a scheduled event, the client will be able to book a place, if the event is not full.
- **FR-71 (3)**. If the user clicks on a non-class gym zone, a form has to be shown to choose the interval in which they want to make the appointment. This includes the date of the appointment, the duration and the starting hour.
- **FR-72 (3)**. On selecting a date and a duration, the server has to return what available starting hours there are. If there is no capacity for such selections, the user will not be able to create the appointment.

3.4.12 Client app - Settings page

The personal information, which is nearly the settings page for the client, has to display most of their information. Additionally, it can show some user stats *for the geeks*.

- **FR-73 (3)**. The view has to allow the user to log out.
- **FR-74 (3)**. The view has to allow the client to change their personal information.
- **FR-75 (3)**. The view has to allow the client to change their password.

- **FR-76 (3)**. The view has to display a list with all the upcoming appointments of the user.
- **FR-77 (1)**. If wanted, the user has to be able to see a list with all the past appointments.
- **FR-78 (1)**. The settings menu has to provide an option to change the application theme (light and dark) and persist it.
- **FR-79 (1)**. The view has to display an analytics table to see which days have they accessed a virtual gym.

3.5 Non-functional requirements

A non-functional requirement is a requirement that defines system attributes such as security, reliability, maintainability, scalability and usability. Therefore, such requisites do not describe information to keep nor functionalities to implement, rather characteristics of such functionalities. In the system, the following non-functional requirements can be defined:

- **NFR-1 (3)**. The system has to provide a secure authentication method.
- **NFR-2 (2)**. The application has to provide different authentication methods in order to differentiate the gym owner, the gym worker and the gym client.
- **NFR-3 (2)**. The landing and client web applications must provide a responsive interface, so that the application can be seen both in small and large screens.
- **NFR-4 (1)**. The core application has to provide a responsive interface up to a point. In smaller screens such as phones, some functionalities would not be easy to implement nor to use.
- **NFR-5 (2)**. The three web applications have to provide an accessible and semantic HTML structure.

3.6 Requirements dependency matrix

Due to the large quantity of requirements that have been determined for the development of the project, and due to the low relationship between them, the dependencies will be shown individually, in order to avoid displaying a nearly empty table. Using the following notation, the result is more consise and can be understood better.

Each dependency will use the following format: **[FR-1, FR-2, FR-3] → [FR-4, FR-5]**. It states that the functional requirements 1, 2 and 3 depend on the 4 and the 5.

[FR-11, FR-13, FR-14, FR-16, FR-17, FR-18] → [FR-12]. In order to visualise, edit or delete a virtual gym, such virtual gym has had to be created previously.

[FR-19, FR-20, FR-21, FR-22, FR-23] → [FR-12]. The gym zones are required to exist inside a virtual gym. If such virtual gym has not been created, the user can not visualise, edit or delete the gym zone.

[FR-19, FR-21, FR-22, FR-23] → [FR-20]. In order to visualise, edit or delete a gym zone, such gym zone has had to be created previously.

[FR-24, FR-25, FR-26, FR-27, FR-28, FR-29, FR-30] → [FR-20]. A calendar is required to exist inside a gym zone. If such gym zone has not been created, the user can not visualise, edit or delete the events of the calendar.

[FR-31, FR-35] → [FR-32]. In order to visualise, edit or delete an event type, such event type has had to be created previously.

[FR-33, FR-36] → [FR-36]. In order to visualise, edit or delete an event template, such event template has had to be created previously.

[FR-38, FR-39, FR-40, FR-41] → [FR-27]. In order to visualise, edit or delete a worker, such worker has had to be created previously.

[FR-43, FR-44, FR-45, FR-46] → [FR-42]. In order to visualise, edit or delete a trainer, such trainer has had to be created previously.

[FR-48, FR-49, FR-50, FR-51] → [FR-47]. In order to visualise, edit or delete a client, such client has had to be created or registered previously.

[FR-66, FR-67, FR-68, FR-69, FR-70, FR-71, FR-72] → [FR-65]. In order to interact with the application, the client must be registered previously.

4. Planning

This chapter will cover the planning of the project. After having defined the requirements and the matrix of dependencies of such requirements, working packages will be defined, which will ensure an organized development using tasks.

4.1 Working packages

Each's table corresponds to one of the working packages. Since the *Development* branch of the working packages tree the most extensive, its tables are explained after the *Testing* branch, and so are the package numbers.

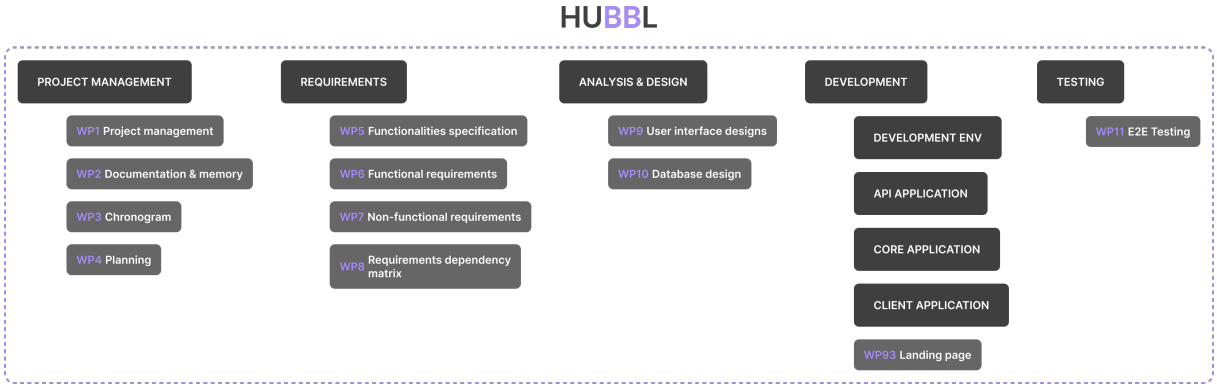

Figure 4.1: Structure of the working packages at the root level

The image above shows how the working packages have been organized. Each column represents a module, which are:

- Project management.
- Requirements.
- Analysis and design.
- Development.
- Testing.

Since the development branch is very extensive, each project has been extracted in smaller working packages.

4.1.1 Project management

The *project management* branch contains all the working packages that are related any task that involves the specification of how the project will be managed in order to be successful.

Package name	**WP1**: Project management
Description	Working package description.
Estimated time	8h
Tasks	**T1**: Project document writing. **T2**: Project document review.
Results	Project document to be submitted to the jury.

Table 4.1: Package one's table - Project management

Package name	**WP2**: Documentation and memory
Description	Wording of some chapters of the memory.
Estimated time	8h
Tasks	**T1**: Wording of the project introduction. **T2**: Wording of the requirements. **T3**: Wording of the analysis and design. **T4**: Wording of the studies and decisions. **T5**: Wording of the project development.
Results	Project document to be submitted to the jury.

Table 4.2: Package two's table - Documentation and memory

Package name	**WP3**: Chronogram
Description	Define the working chronogram for the project development.
Estimated time	4h
Tasks	**T1**: Temporal planning of the working packages.
Results	Gantt's project diagram.

Table 4.3: Package three's table - Chronogram

Package name	**WP4**: Planning
Description	Structure the working packages.
Estimated time	4h
Tasks	**T1**: Planning of the working packages.
Results	Chapter 6 of the memory document.

Table 4.4: Package four's table - Planning

Package name	**WP5**: Functionalities specification
Description	Analysis of the functionalities of the application.
Estimated time	8h
Tasks	**T1**: Users identification. **T2**: Define user needs (owner/worker and client). **T3**: Define product functionalities
Results	Sections 5.2 and 5.3 of the memory document.

Table 4.5: Package five's table - Functionalities specification

4.1.2 Requirements

The *requirements* branch contains all the tasks that are related with the analysis and definition of all the requirements and functionalities of each application.

Package name	**WP6**: Functional requirements
Description	Analysis and definition of the functional requirements of the application.
Estimated time	12h
Tasks	**T1**: Use case diagram. **T2**: Functional requirements' specification for the *core* application. **T3**: Functional requirements' specification for the *client* application. **T4**: Functional requirements' specification for the *landing* application. **T5**: Functional requirements review and make changes if needed.
Results	Section 5.4 of the memory document.

Table 4.6: Package six's table - Functional requirements

Package name	**WP7**: Non-functional requirements
Description	Analysis and definition of the non-functional requirements of the application.
Estimated time	4h
Tasks	**T1**: Initial non-functional requirements' specification. **T2**: Non-functional requirements review and make changes if needed.
Results	Section 5.5 of the memory document.

Table 4.7: Package seven's table - Non-functional requirements

Package name	**WP8**: Requirements dependency matrix
Description	Definition of the requirements' dependency matrix.
Estimated time	4h
Tasks	**T1**: Matrix dependency specification once the functional and non-functional requirements have been specified.
Results	Section 5.6 of the memory document.

Table 4.8: Package eight's table - Requirements dependency matrix

4.1.3 Analysis and design

The *analysis and design* branch contains all the working packages that are related with the design and analysis of the system interface and architecture.

Package name	**WP9**: User interface designs
Description	Definition of the requirements' dependency matrix.
Estimated time	96h
Tasks	**T1**: Design system specification. **T2**: Prototype most of the user interfaces. **T3**: Design review. Ensure it satisfies the requirements previously specified.
Results	User interface prototypes which provide a vague view of the application's look and feel.

Table 4.9: Package nine's table - User interfaces designs

Package name	**WP10**: Database design
Description	Definition of the application's database.
Estimated time	4h
Tasks	**T1**: Design of the applications' database structure. **T2**: Schema and structure review. **T3**: Analysis of the database structure using PostgreSQL.
Results	Database structure in which the application's data will be persisted.

Table 4.10: Package ten - Database design

4.1.4 Testing

The *testing* branch contains the working package that is related with the testing of the application.

Package name	**WP11**: E2e testing
Description	End-to-end testing implementation for the applications.
Estimated time	4h
Tasks	**T1**: *API* aplication e2e testing. **T2**: *Core* application e2e testing. **T3**: *Client* application e2e testing. **T4**: *Landing* application e2e testing.
Results	End-to-end tests which ensure the correct behaviour of each application, and that any futher new features and updates do not break the application.

Table 4.11: Package eleven's table - E2e testing

4.1.5 Development

The *development* branch is probably the most extensive, as it is the most important part of the project. The branch consists of 73 working packages in total. Nevertheless, most of the packages follow the same structure. For instance, as it is seen in the Api application, each web service package follows the same structure: a working package to create an entity, another one to update it, a third one to delete it, and a final one to fetch the entity. The structure is the same for each web service, which easily increases the number of working packages.

4.1.5.1 Development env

The first group that is in the *development* branch is the *development env* (or *development environment*). This part is as important as any other application, since it will define the structure of the project. Furthermore, the continuous integration has to be set up. Using the Nx build system, such feature becomes extremely simple and scalable. One of the options it offers is to run a command to as many projects as wanted. Not only so, that you can run the commands to the affected projects. Such feature reduces even more the CI execution time.

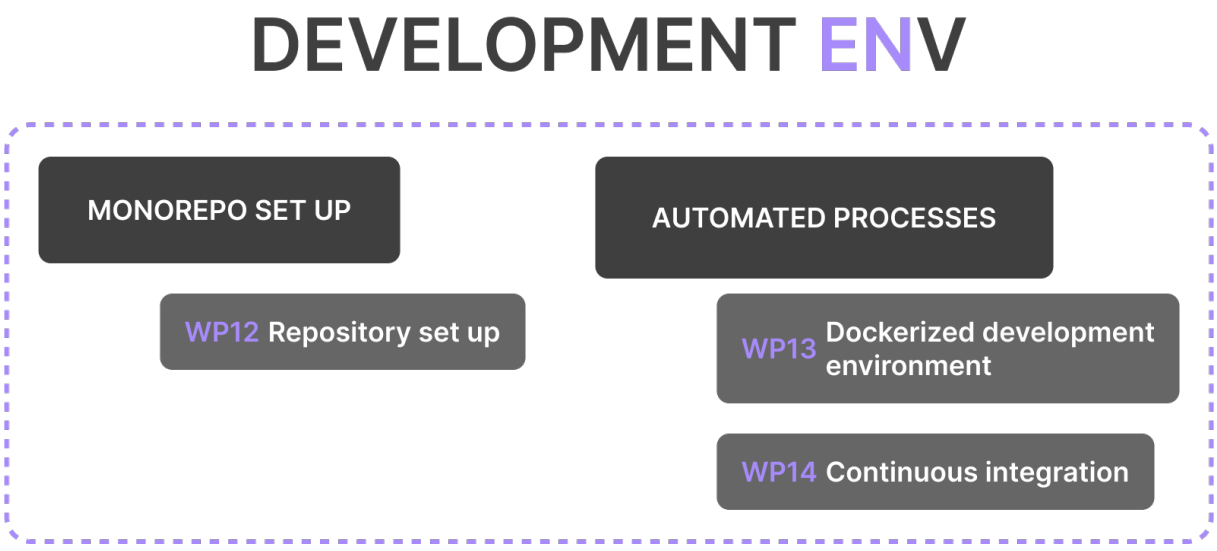

Figure 4.2: Api application working packages diagram

The *development environment* group is composed of the following working packages:

Package name	**WP12**: Repository set up
Description	Prepare the Nx monorepo.
Estimated time	2h
Tasks	**T1**: Generate an `nx-workspace` ready to develop.
Results	Initial structure of the monorepo.

Table 4.12: Package twelve's table - Repository set up

Package name	**WP13**: Dockerized development environment.
Description	Dockerize the database, the test database and the API application.
Estimated time	2h
Tasks	**T1**: Dockerize the main database and the test database. **T2**: Dockerize the API application using a NodeJS image.
Results	The dockerization of the databases and the REST API.

Table 4.13: Package thirteen's table - Dockerized development environment

Package name	**WP14**: Continuous integration.
Description	Set up Github Actions for CI.
Estimated time	8h
Tasks	**T1**: Set up CI for the *API* application. **T2**: Set up CI for the *core* application. **T3**: Set up CI for the *client* application. **T4**: Set up CI for the *landing* application. **T5**: Set up CI for the monrepo libraries.
Results	An automated process for testing the code developed.

Table 4.14: Package fourteen's table - Continuous integration

4.1.5.2 Api application

The working package groups of the *api* application have been designed so that each web service of the application is divided in multiple working packages with tasks that are very specific and simple.

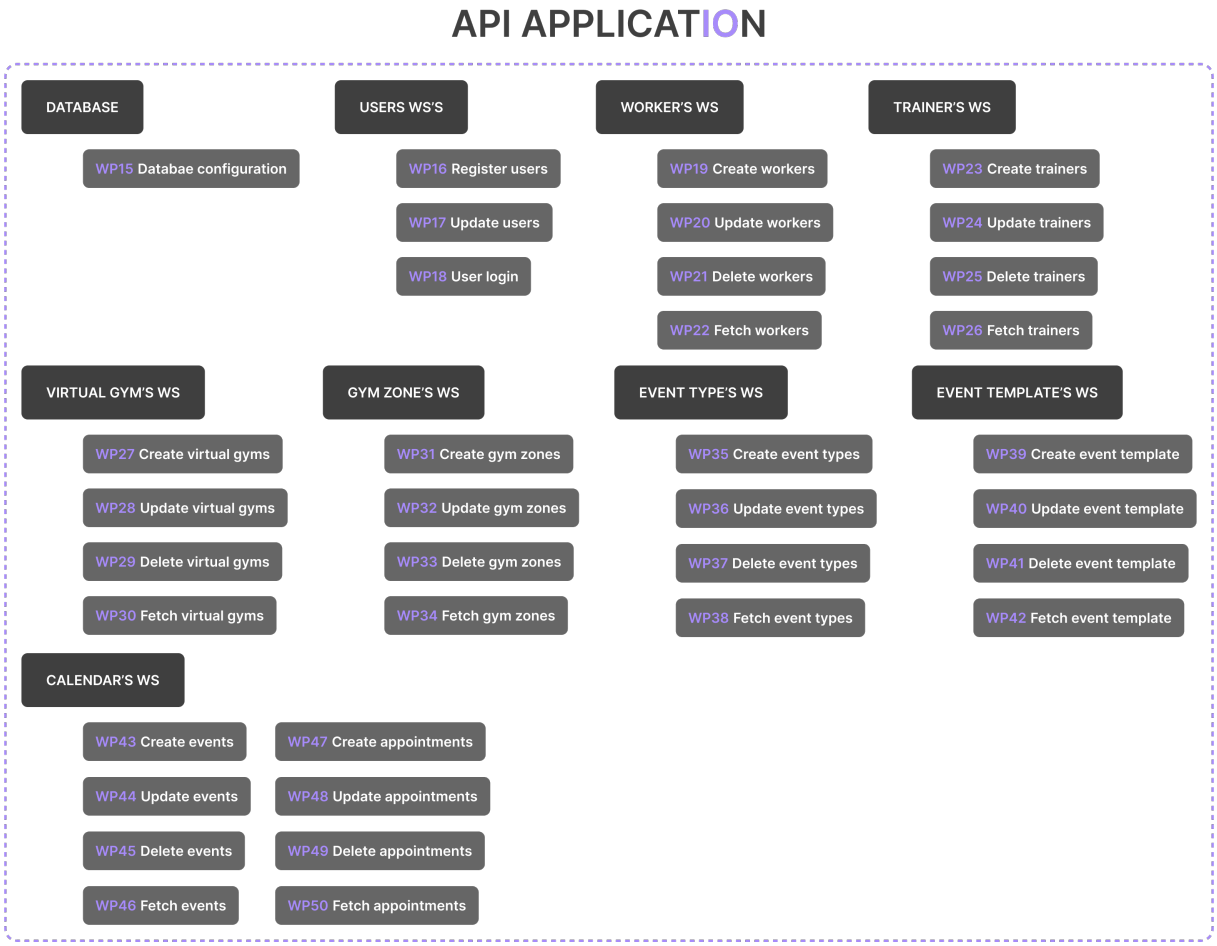

Figure 4.3: Api application working packages diagram

The *api* group is composed of the following working packages:

Package name	**WP15**: Database configuration.
Description	Configure the application so that is connected the database.
Estimated time	1h
Tasks	**T1**: Configure the API with the dockerized database.
Results	Connection of the api with the database.

Table 4.15: Package fifteen's table - Database configuration

Package name	**WP16**: Register users
Description	Allow the users to register.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the owner register endpoint. **T4**: Create the client register endpoint.
Results	Creation of a user.

Table 4.16: Package sixteen's table - Register users

Package name	**WP17**: Update users
Description	Allow the users to update their information.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the user update endpoint.
Results	Updation of user's information

Table 4.17: Package seventeen's table - Update users

Package name	**WP18**: User login
Description	Allow the users to log in to the application.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the session cookie validation endpoint. **T4**: Create the login endpoint.
Results	Endpoints to login and validate old user sessions.

Table 4.18: Package eighteen's table - User login

Package name	**WP19**: Create workers
Description	Allow the creation of workers.
Estimated time	6h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the workers create endpoint.
Results	Creation of workers.

Table 4.19: Package nineteen's table - Create workers

Package name	**WP20**: Update workers
Description	Allow the update of workers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the update workers endpoint.
Results	Updation of workers.

Table 4.20: Package twenty's table - Update workers

Package name	**WP21**: Delete workers
Description	Allow the deletion of workers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the delete workers endpoint.
Results	Deletion of workers.

Table 4.21: Package twenty-one's table - Delete workers

Package name	**WP22**: Fetch workers
Description	Allow the fetch of workers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the fetch workers endpoint.
Results	Fetching of workers.

Table 4.22: Package twenty-two's table - Fetch workers

Package name	**WP23**: Create trainers
Description	Allow the creation of trainers.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the create trainers endpoint.
Results	Creation of trainers.

Table 4.23: Package twenty-three's table - Create trainers

Package name	**WP24**: Update trainers
Description	Allow the update of trainers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the update trainers endpoint.
Results	Updation of trainers.

Table 4.24: Package twenty-four's table - Update trainers

Package name	**WP25**: Delete trainers
Description	Allow the deletion of trainers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the delete trainers endpoint.
Results	Deletion of trainers.

Table 4.25: Package twenty-five's table - Delete trainers

Package name	**WP26**: Fetch trainers
Description	Allow the fetch of trainers.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the fetch trainers endpoint.
Results	Fetching of trainers.

Table 4.26: Package twenty-six's table - Fetch trainers

Package name	**WP27**: Create virtual gyms
Description	Allow the creation of virtual gyms.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the virtual gym creation endpoint.
Results	Creation of virtual gyms.

Table 4.27: Package twenty-seven's table - Create virtual gyms

Package name	**WP28**: Update virtual gyms
Description	Allow the update of virtual gyms.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the virtual gym update endpoint.
Results	Updation of virtual gyms.

Table 4.28: Package twenty-eight's table - Update virtual gyms

Package name	**WP29**: Delete virtual gyms
Description	Allow the deletion of virtual gyms.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the virtual gym deletion endpoint.
Results	Deletion of virtual gyms.

Table 4.29: Package twenty-nine's table - Delete virtual gyms

Package name	**WP30**: Fetch virtual gyms
Description	Allow the fetch of virtual gyms.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the virtual gym fetch endpoint.
Results	Fetching of virtual gyms.

Table 4.30: Package thirty's table - Delete virtual gyms

Package name	**WP31**: Create gym zones
Description	Allow the creation of gym zones.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the gym zone create endpoint.
Results	Creation of gym zones.

Table 4.31: Package thirty-one's table - Create gym zones

Package name	**WP32**: Update gym zones
Description	Allow the update of gym zones.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the gym zone update endpoint.
Results	Updation of gym zones.

Table 4.32: Package thirty-two's table - Update gym zones

Package name	**WP33**: Delete gym zones
Description	Allow the deletion of gym zones.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the gym zone delete endpoint.
Results	Deletion of gym zones.

Table 4.33: Package thirty-three's table - Delete gym zones

Package name	**WP34**: Fetch gym zones
Description	Allow the fetching of gym zones.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the gym zone fetch endpoint.
Results	Fetching of gym zones.

Table 4.34: Package thirty-four's table - Fetch gym zones

Package name	**WP35**: Create event types
Description	Allow the creation of event types.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event types create endpoint.
Results	Creation of event types.

Table 4.35: Package thirty-five's table - Create event types

Package name	**WP36**: Update event types
Description	Allow the update of event types.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event types update endpoint.
Results	Updation of event types.

Table 4.36: Package thirty-six's table - Update event types

Package name	**WP37**: Delete event types
Description	Allow the deletion of event types.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event types delete endpoint.
Results	Updation of event types.

Table 4.37: Package thirty-seven's table - Delete event types

Package name	**WP38**: Fetch event types
Description	Allow the fetch of event types.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event types fetch endpoint.
Results	Fetching of event types.

Table 4.38: Package thirty-eight's table - Fetch event types

Package name	**WP39**: Create event templates
Description	Allow the creation of event templates.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event templates create endpoint.
Results	Creation of event templates.

Table 4.39: Package thirty-nine's table - Create event templates

Package name	**WP40**: Update event templates
Description	Allow the update of event templates.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event templates update endpoint.
Results	Updation of event templates.

Table 4.40: Package forty's table - Update event templates

Package name	**WP41**: Delete event templates
Description	Allow the deletion of event templates.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event templates delete endpoint.
Results	Deletion of event templates.

Table 4.41: Package forty-one's table - Delete event templates

Package name	**WP42**: Fetch event templates
Description	Allow the fetch of event templates.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the event templates fetch endpoint.
Results	Fetching of event templates.

Table 4.42: Package forty-two's table - Fetch event templates

Package name	**WP43**: Create events
Description	Allow the creation of events.
Estimated time	8h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the events fetch endpoint.
Results	Creation of events.

Table 4.43: Package forty-three's table - Create events

Package name	**WP44**: Update events
Description	Allow the update of events.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the events update endpoint.
Results	Updation of events.

Table 4.44: Package forty-four's table - Update events

Package name	**WP45**: Delete events
Description	Allow the deletion of events.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the events delete endpoint.
Results	Deletion of events.

Table 4.45: Package forty-five's table - Delete events

Package name	**WP46**: Fetch events
Description	Allow the fetching of events.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the events fetch endpoint.
Results	Fetching of events.

Table 4.46: Package forty-six's table - Fetch events

Package name	**WP47**: Create appointments
Description	Allow the fetching of appointments. It is important to create an algorithm that ensures that a gym zone does not have more appointments than its capacity.
Estimated time	16h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T3**: Create the available hours endpoint. **T4**: Create the appointments create endpoint.
Results	Creation of appointments.

Table 4.47: Package forty-seven's table - Create appointments

Package name	**WP48**: Update appointments
Description	Allow the update of appointments. It should only allow to cancel the appointment.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T4**: Create the appointments update endpoint.
Results	Updation of appointments.

Table 4.48: Package forty-eight's table - Update appointments

Package name	**WP49**: Delete appointments
Description	Allow the deletion of appointments. It should only allow to cancel the appointment.
Estimated time	4h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T4**: Create the appointments delete endpoint.
Results	Deletion of appointments.

Table 4.49: Package forty-nine's table - Delete appointments

Package name	**WP50**: Fetch appointments
Description	Allow the fetch of appointments.
Estimated time	6h
Tasks	**T1**: Create the services required. **T2**: Create the controllers required. **T4**: Create the appointments fetch endpoint.
Results	Fetching of appointments.

Table 4.50: Package fifty's table - Fetch appointments

4.1.5.3 Core application

The *core* group of working packages follows a similar approach as the one described in Development.

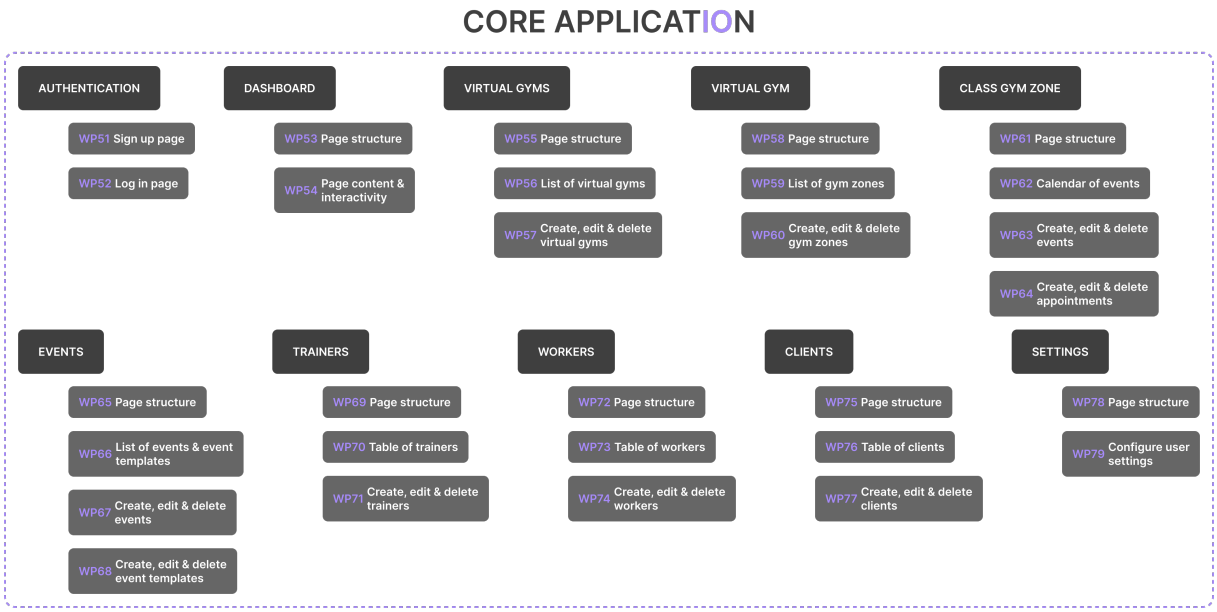

Figure 4.4: Core application working packages diagram

The *core* group is composed of the following working packages:

Package name	**WP51**: Sign up page
Description	Develop the sign up page
Estimated time	16h
Tasks	**T1**: Set up the page structure. **T2**: Sign up as an owner.
Results	Allow the users to register to the system.

Table 4.51: Package fifty-one's table - Sign up page

Package name	**WP52**: Log in page
Description	Develop the log in page.
Estimated time	16h
Tasks	**T1**: Set up the page structure. **T2**: Log in as an owner. **T3**: Log in as a worker.
Results	Allow the user to log in to the system.

Table 4.52: Package fifty-two's table - Log in page

Package name	**WP53**: Page structure (Dashboard)
Description	Apply the design of the dashboard page.
Estimated time	4h
Tasks	**T1**: Apply the design of the dashboard page.
Results	The scaffolding of the dashboard page.

Table 4.53: Package fifty-three's table - Page structure (Dashboard)

Package name	**WP54**: Page content and interactivity (Dashboard)
Description	Add the interactivity to the dashboard page.
Estimated time	12h
Tasks	**T1**: Add virtual gyms' interactivity. **T2**: Add gym zones' interactivity. **T3**: Add trainers' interactivity. **T4**: Add event templates' interactivity. **T5**: Add events' interactivity.
Results	User experience of the dasboard page.

Table 4.54: Package fifty-four's table - Page content and interactivity

Package name	**WP55**: Page structure (Virtual gyms)
Description	Apply the design of the virtual gyms page.
Estimated time	4h
Tasks	**T1**: Apply the design of the virtual gyms page.
Results	The scaffolding of the virtual gyms page.

Table 4.55: Package fifty-five's table - Page structure (Virtual gyms)

Package name	**WP56**: List of virtual gyms (Virtual gyms)
Description	Display the list of virtual gyms.
Estimated time	4h
Tasks	**T1**: List the virtual gyms of the gym. **T2**: List the gym zones inside each virtual gym.
Results	The page of virtual gyms.

Table 4.56: Package fifty-six's table - List of virtual gyms (Virtual gyms)

Package name	**WP57**: Create, edit and delete virtual gyms (Virtual gyms)
Description	Create the modal to create, edit and delete virtual gyms.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete virtual gyms.
Results	The modal to interact with virtual gyms.

Table 4.57: Package fifty-seven's table - List of virtual gyms (Virtual gyms)

Package name	**WP58**: Page structure (Virtual gym)
Description	Apply the design of the virtual gym page.
Estimated time	4h
Tasks	**T1**: Apply the design of the virtual gym page.
Results	The scaffolding of the virtual gym page.

Table 4.58: Package fitfty-eight's table - Page structure (Virtual gym)

Package name	**WP59**: List of gym zones (Virtual gym)
Description	List the gym zones of a virtual gym.
Estimated time	8h
Tasks	**T1**: List all class gym zones. **T2**: List all non-class gym zones.
Results	The page of a virtual gym.

Table 4.59: Package fifty-nine's table - List of gym zones (Virtual gym)

Package name	**WP60**: Create, edit and delete gym zones (Virtual gym)
Description	Create the modal to create, edit and delete gym zones.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete gym zones.
Results	The modal to interact with gym zones.

Table 4.60: Package sixty's table - Create, edit and delete gym zones (Virtual gym)

Package name	**WP61**: Page structure (Class gym zone)
Description	Apply the design of the class gym zone page.
Estimated time	4h
Tasks	**T1**: Apply the design of the class gym zone page.
Results	The scaffolding of the class gym zone page.

Table 4.61: Package sixty-one's table - Page structure (Class gym zone)

Package name	**WP62**: Calendar of events (Class gym zone)
Description	Display the calendar of events of the gym zone.
Estimated time	8h
Tasks	**T1**: Design the calendar of events. **T2**: Add interactivity to the calendar.
Results	The calendar of events for the gym zone.

Table 4.62: Package sixty-two's table - Calendar of events (Class gym zone)

Package name	**WP63**: Create, edit and delete events (Class gym zone)
Description	Modal to create, edit and delete events.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete events.
Results	The modal to interact with events.

Table 4.63: Package sixty-three's table - Create, edit and delete events (Class gym zone)

Package name	**WP64**: Create, edit and delete appointments (Class gym zone)
Description	Modal to create, edit and delete appointments.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete appointments.
Results	The modal to interact with appointments.

Table 4.64: Package sixty-four's table - Create, edit and delete appointments (Class gym zone)

Package name	**WP65**: Page structure (Events)
Description	Apply the design of the events page.
Estimated time	4h
Tasks	**T1**: Apply the design of the events page.
Results	The scaffolding of the events page.

Table 4.65: Package sixty-five's table - Page structure (Events)

Package name	**WP66**: List of events and event templates (Events)
Description	Display the list of events and event templates.
Estimated time	6h
Tasks	**T1**: List the events. **T2**: List the event templates
Results	The events page.

Table 4.66: Package sixty-six's table - List of events and event tempaltes (Events)

Package name	**WP67**: Create, edit and delete events (Events)
Description	Modal to create, edit and delete events.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete events.
Results	The modal to interact with events.

Table 4.67: Package sixty-seven's table - Create, edit and delete events (Events)

Package name	**WP68**: Create, edit and delete events templates (Events)
Description	Modal to create, edit and delete events templates.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete events templates.
Results	The modal to interact with events templates.

Table 4.68: Package sixty-eight's table - Create, edit and delete events templates (Events)

Package name	**WP69**: Page structure (Trainers)
Description	Apply the design of the trainers page.
Estimated time	4h
Tasks	**T1**: Apply the design of the trainers page.
Results	The scaffolding of the trainers page.

Table 4.69: Package sixty-nine's table - Page structure (Trainers)

Package name	**WP70**: Table of trainers (Trainers)
Description	Table of trainers.
Estimated time	4h
Tasks	**T1**: Display the table of trainers.
Results	The table of trainers.

Table 4.70: Package seventy's table - Table of trainers (Trainers)

Package name	**WP71**: Create, edit and delete trainers (Trainers)
Description	Modal to create, edit and delete trainers.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete trainers.
Results	The modal to interact with trainers.

Table 4.71: Package seventy-one's table - Table of trainers (Trainers)

Package name	**WP72**: Page structure (Workers)
Description	Apply the design of the workers page.
Estimated time	4h
Tasks	**T1**: Apply the design of the workers page.
Results	The scaffolding of the workers page.

Table 4.72: Package seventy-two's table - Page structure (Workers)

Package name	**WP73**: Table of workers (Workers)
Description	Table of workers.
Estimated time	4h
Tasks	**T1**: Display the table of workers.
Results	The table of workers.

Table 4.73: Package seventy-three's table - Table of workers (Workers)

Package name	**WP74**: Create, edit and delete workers (Workers)
Description	Modal to create, edit and delete workers.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete workers.
Results	The modal to interact with workers.

Table 4.74: Package seventy-four's table - Table of workers (Workers)

Package name	**WP75**: Page structure (Trainers)
Description	Apply the design of the trainers page.
Estimated time	4h
Tasks	**T1**: Apply the design of the trainers page.
Results	The scaffolding of the trainers page.

Table 4.75: Package seventy-five's table - Page structure (Trainers)

Package name	**WP76**: Table of trainers (Trainers)
Description	Table of trainers.
Estimated time	4h
Tasks	**T1**: Display the table of trainers.
Results	The table of trainers.

Table 4.76: Package seventy-six's table - Table of trainers (Trainers)

Package name	**WP77**: Create, edit and delete trainers (Trainers)
Description	Modal to create, edit and delete trainers.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete trainers.
Results	The modal to interact with trainers.

Table 4.77: Package seventy-seven's table - Table of trainers (Trainers)

Package name	**WP78**: Page structure (Settings)
Description	Apply the design of the settings page.
Estimated time	4h
Tasks	**T1**: Apply the design of the settings page.
Results	The scaffolding of the settings page.

Table 4.78: Package seventy-eight's table - Page structure (Settings)

Package name	**WP79**: Configure user settings (Settings)
Description	Set up the form to update user settings.
Estimated time	8h
Tasks	**T1**: Log out the user. **T2**: Configure the basic fields of the user. **T3**: Update the user password. **T4**: Update the gym information as an owner.
Results	The settings page.

Table 4.79: Package seventy-nine's table - Configure user settings (Settings)

4.1.5.4 Client application

The *landing* group of working packages follows a similar approach as the one described in Development.

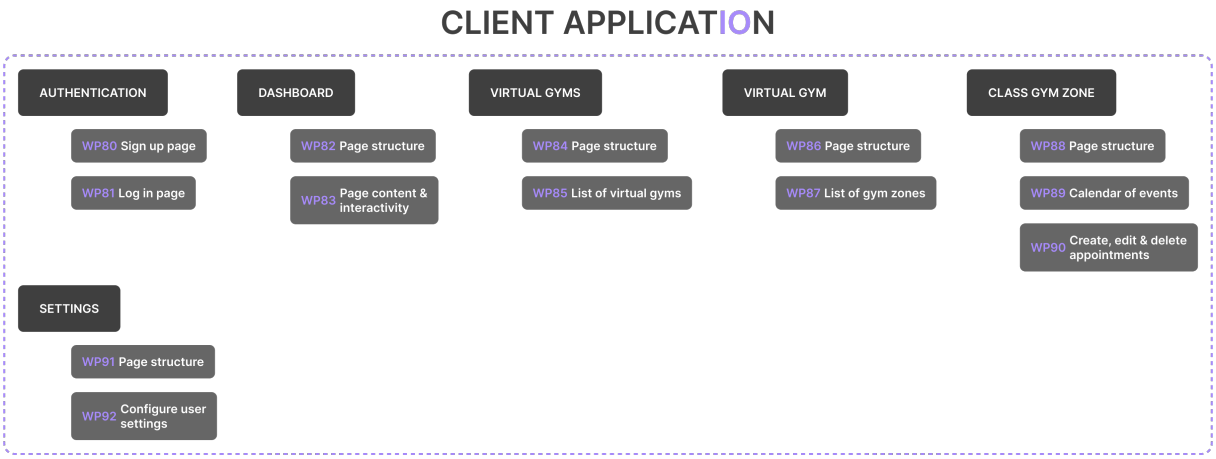

Figure 4.5: Client application working packages diagram

The *client* group is composed of the following working packages:

Package name	**WP80**: Sign up page
Description	Develop the sign up page
Estimated time	8h
Tasks	**T1**: Set up the page structure. **T2**: Sign up as a client.
Results	Allow the clients to register to the system.

Table 4.80: Package eighty's table - Sign up page

Package name	**WP81**: Log in page
Description	Develop the log in page.
Estimated time	8h
Tasks	**T1**: Set up the page structure. **T2**: Log in as a client.
Results	Allow the user to log in to the system.

Table 4.81: Package eighty-one's table - Log in page

Package name	**WP82**: Page structure (Dashboard)
Description	Apply the design of the dashboard page.
Estimated time	4h
Tasks	**T1**: Apply the design of the dashboard page.
Results	The scaffolding of the dashboard page.

Table 4.82: Package eighty-two's table - Page structure (Dashboard)

Package name	**WP83**: Page content and interactivity (Dashboard)
Description	Add the interactivity to the dashboard page.
Estimated time	6h
Tasks	**T1**: Add virtual gyms' interactivity. **T2**: Add gym zones' interactivity.
Results	User experience of the dasboard page.

Table 4.83: Package eighty-three's table - Page content and interactivity

Package name	**WP84**: Page structure (Virtual gyms)
Description	Apply the design of the virtual gyms page. Should reuse the core's page.
Estimated time	2h
Tasks	**T1**: Apply the design of the virtual gyms page.
Results	The scaffolding of the virtual gyms page.

Table 4.84: Package eighty-fourt's table - Page structure (Virtual gyms)

Package name	**WP85**: List of virtual gyms (Virtual gyms)
Description	Display the list of virtual gyms.
Estimated time	2h
Tasks	**T1**: List the virtual gyms of the gym. **T2**: List the gym zones inside each virtual gym.
Results	The page of virtual gyms.

Table 4.85: Package eighty-five's table - List of virtual gyms (Virtual gyms)

Package name	**WP86**: Page structure (Virtual gym)
Description	Apply the design of the virtual gym page. Should reuse the core's page.
Estimated time	2h
Tasks	**T1**: Apply the design of the virtual gym page.
Results	The scaffolding of the virtual gym page.

Table 4.86: Package eighty-six's table - Page structure (Virtual gym)

Package name	**WP87**: List of gym zones (Virtual gym)
Description	List the gym zones of a virtual gym.
Estimated time	2h
Tasks	**T1**: List all class gym zones. **T2**: List all non-class gym zones.
Results	The page of a virtual gym.

Table 4.87: Package eighty-seven's table - List of gym zones (Virtual gym)

Package name	**WP88**: Page structure (Class gym zone)
Description	Apply the design of the class gym zone page. Should reuse the core's page.
Estimated time	2h
Tasks	**T1**: Apply the design of the class gym zone page.
Results	The scaffolding of the class gym zone page.

Table 4.88: Package eighty-eight's table - Page structure (Class gym zone)

Package name	**WP89**: Calendar of events (Class gym zone)
Description	Display the calendar of events of the gym zone.
Estimated time	4h
Tasks	**T1**: Design the calendar of events. **T2**: Add interactivity to the calendar.
Results	The calendar of events for the gym zone.

Table 4.89: Package ninety's table - Calendar of events (Class gym zone)

Package name	**WP90**: Create, edit and delete appointments (Class gym zone)
Description	Modal to create, edit and delete appointments.
Estimated time	8h
Tasks	**T1**: Modal to create, edit and delete appointments.
Results	The modal to interact with appointments.

Table 4.90: Package ninety's table - Create, edit and delete appointments (Class gym zone)

Package name	**WP91**: Page structure (Settings)
Description	Apply the design of the settings page. Should reuse the core's page.
Estimated time	2h
Tasks	**T1**: Apply the design of the settings page.
Results	The scaffolding of the settings page.

Table 4.91: Package ninety-one's table - Page structure (Settings)

Package name	**WP92**: Configure user settings (Settings)
Description	Set up the form to update client settings.
Estimated time	4h
Tasks	**T1**: Log out the client. **T2**: Configure the basic fields of the client. **T3**: Update the client password.
Results	The settings page.

Table 4.92: Package ninety-two's table - Configure user settings (Settings)

4.1.5.5 Landing application

The landing application is a simple application that displays information about the system and acces to the main application.

Package name	**WP93**: Design the landing application
Description	Set up the form to update client settings.
Estimated time	16h
Tasks	**T1**: Display basic information about the system. **T2**: Display information about the data privacy. **T3**: Display a contact form.
Results	The landing application.

Table 4.93: Package ninety-three's table - Landing page

4.2 Traceability matrix

Due to the large quantity of requirements and working packageds that have been determined for the development of the project, and due to the low relationship between them, the traceability matrix will be shown individually, in order to avoid displaying a nearly empty table. Using the following notation, the result is more consise and can be understood better.

Each dependency will use the following format: [FR-1] → [WP-1, WP-2, WP-3]. It states that the functional requirements 1 will be covered in the working package 1, 2 and 3.

- [FR-1, FR-2, FR-3, FR-4] → [WP-93]
- [FR-5, FR-6] → [WP-51]
- [FR-7] → [WP-52]
- [FR-8, FR-9, FR-10] → [WP-93]
- [FR-11, FR-13, FR-14, FR-15, FR-17] → [WP-56, WP-28, WP-29, WP-30]
- [FR-12, FR-16] → [WP-57, WP-27, WP-28, WP-29]
- [FR-19, FR-21, FR-22] → [WP-59, WP-34]
- [FR-20, FR-23] → [WP-60, WP-31, WP-32, WP-33]
- [FR-24, FR-27, FR-28] → [WP-62, WP-43, WP-44, WP-45, WP-47, WP-48, WP-49]
- [FR-25, FR-26, FR-29, FR-30] → [WP-63, WP-34, WP-46, WP-50]
- [FR-31, FR-33] → [WP-66, WP-38, WP-42, WP-46]
- [FR-32, FR-35] → [WP-67, WP-35, WP-36, WP-37]
- [FR-34, FR-36] → [WP-68, WP-39, WP-40, WP-41]
- [FR-37, FR-38, FR-40] → [WP-74, WP-19, WP-20, WP-21]
- [FR-39, FR-41] → [WP-73, WP-22]
- [FR-42, FR-43, FR-45] → [WP-70, WP-23, WP-24, WP-25]
- [FR-44, FR-46] → [WP-71, WP-26]
- [FR-47, FR-48] → [WP-77]
- [FR-49, FR-50] → [WP-76]
- [FR-51, FR-52, FR-53, FR-54, FR-55, FR-56] → [WP-79, WP-17]
- [FR-65] → [WP-80, WP-16]
- [FR-66] → [WP-81, WP-17]
- [FR-67, FR-68] → [WP-85, WP-30]
- [FR-69] → [WP-89, WP-46]
- [FR-70, FR-71, FR-72] → [WP-90, WP-43, WP-44, WP-45, WP-47, WP-48, WP-49]
- [FR-73, FR-74, FR-75, FR-76, FR-77, FR-78] → [WP-92, WP-17]
- [NFR-1] → [WP-16, WP-18]
- [NFR-2] → [WP-52]

- [NFR-3] → [WP-93]
- [NFR-4] → [WP-53, WP-55, WP-58, WP-61, WP-65, WP-69, WP-72, WP-75, WP-78]
- [NFR-5] → [WP-53, WP-55, WP-58, WP-61, WP-65, WP-69, WP-72, WP-75, WP-78, WP-82, WP-84, WP-86, WP-88, WP-91, WP-93]

4.3 User interfaces

The design of the user interfaces has been a priority before diving into the front end development. Such interfices, provide a preliminar view of what show the application look and feel. The following figures, are the designs of the user interfaces that have been designed using the Figma software[1].
Designing every possible state of the application is far from achievable with such little time. However, since the core and client application will share much of their user interface, only the core views have been designed. When developing the client application, most of the components have been reused, modifying they behaviour as needed, since the client interactions are limited (for instance, the client sees the same dashboard page, yet the button to add a virtual gym is hidden). Needless to say, some views do not correspond exactly as they are in the application, mainly because some features have been modified or added as required.
Finally, the following aspects have been considered when designing the views:

- The views should also be displayed as an owner used. This is due to the fact that the only differences between the views of an owner, a worker or a client, will be minimal. Some things will be hidden or not allowed, which has little effects to the final design.
- Each element that can be created (virtual gyms, gym zones, trainers and so on) must have their corresponding dialog or view in which such items are created, edited or deleated.
    - In case a dialog is used, it has to contain how it looks when: it is being created, it is being edited, and it is being loaded, for each of the previous states.

With these requirements, there are views modals that contain 4 images, each one for the described states.

[1] The figma application is a free software which can be found in: www.figma.com

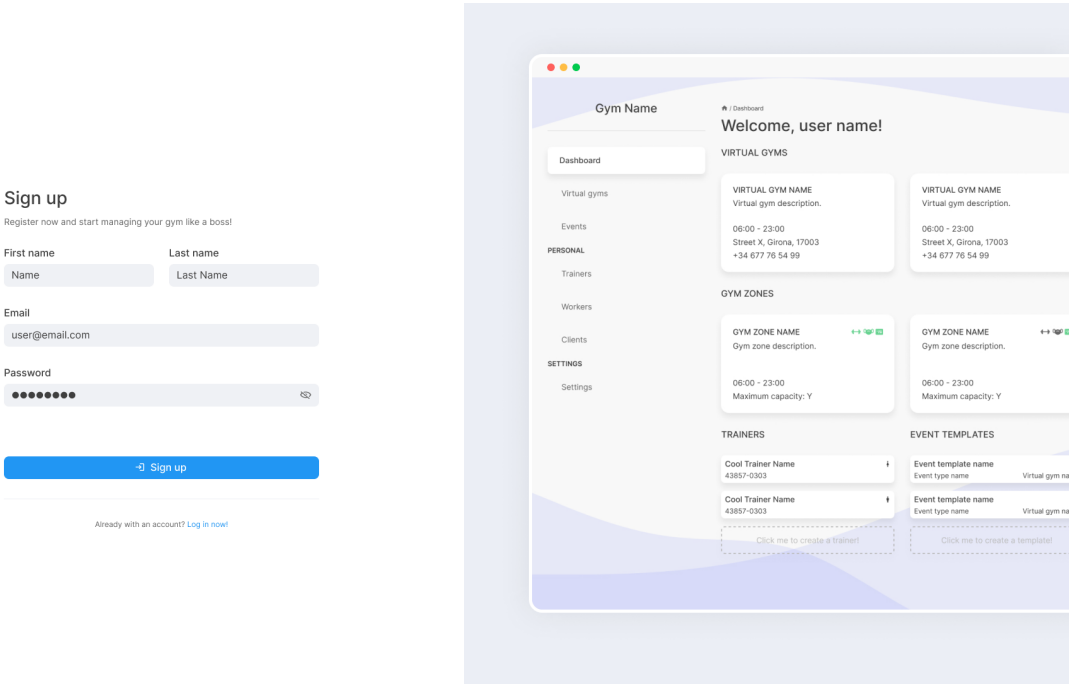

Figure 4.6: First step of the sign up page

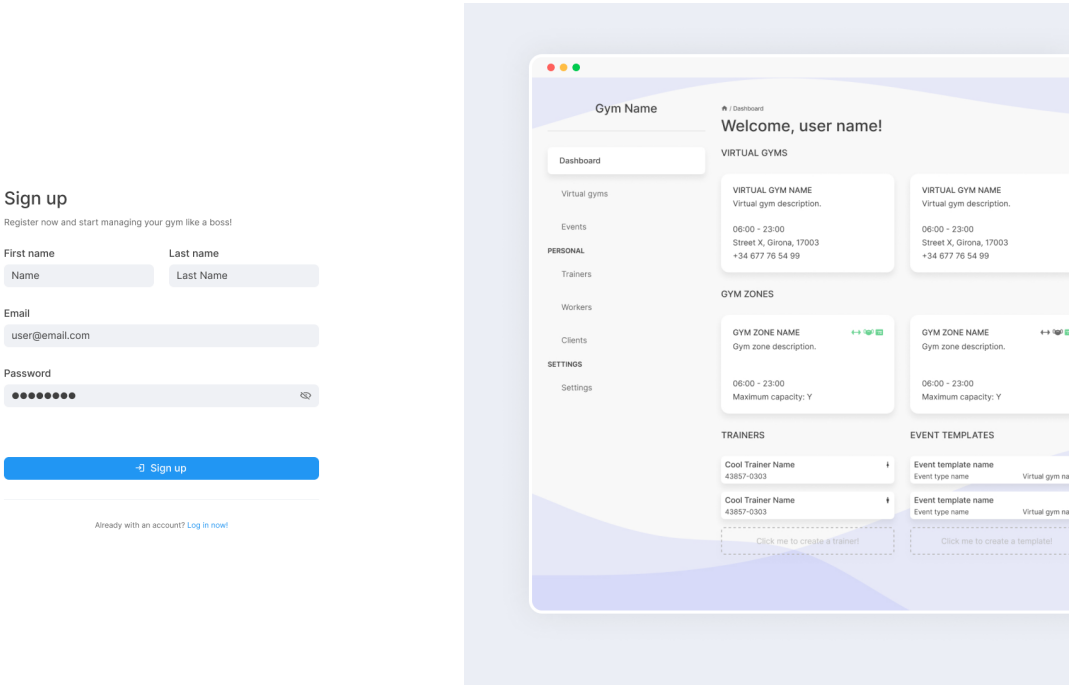

Figure 4.7: Second step of the sign up page

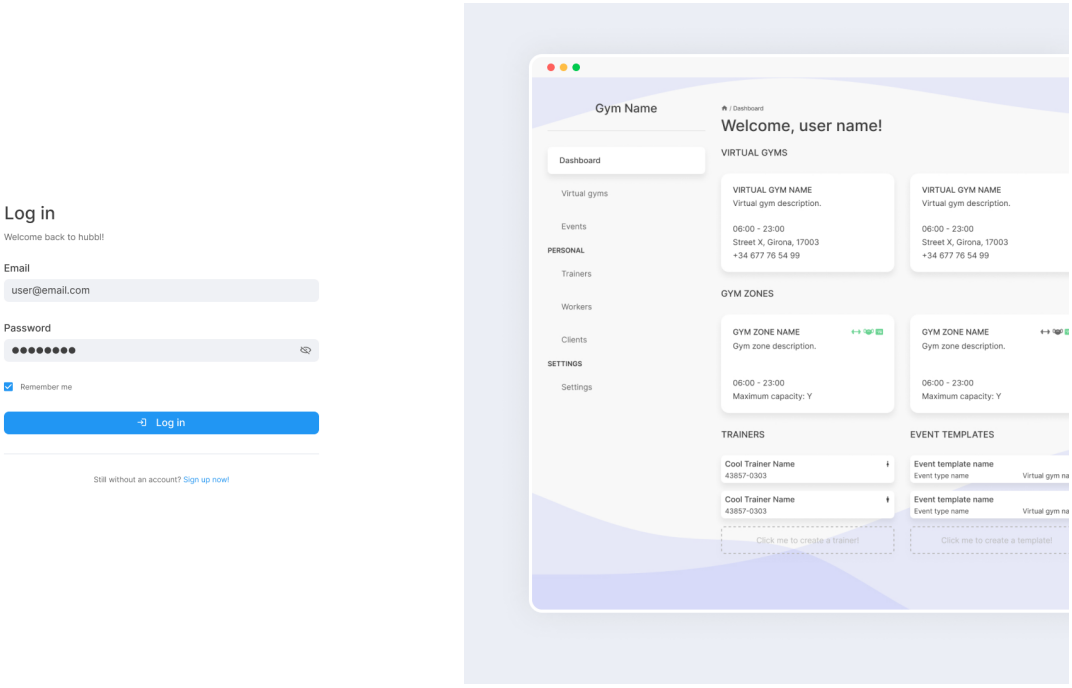

Figure 4.8: View of the login page

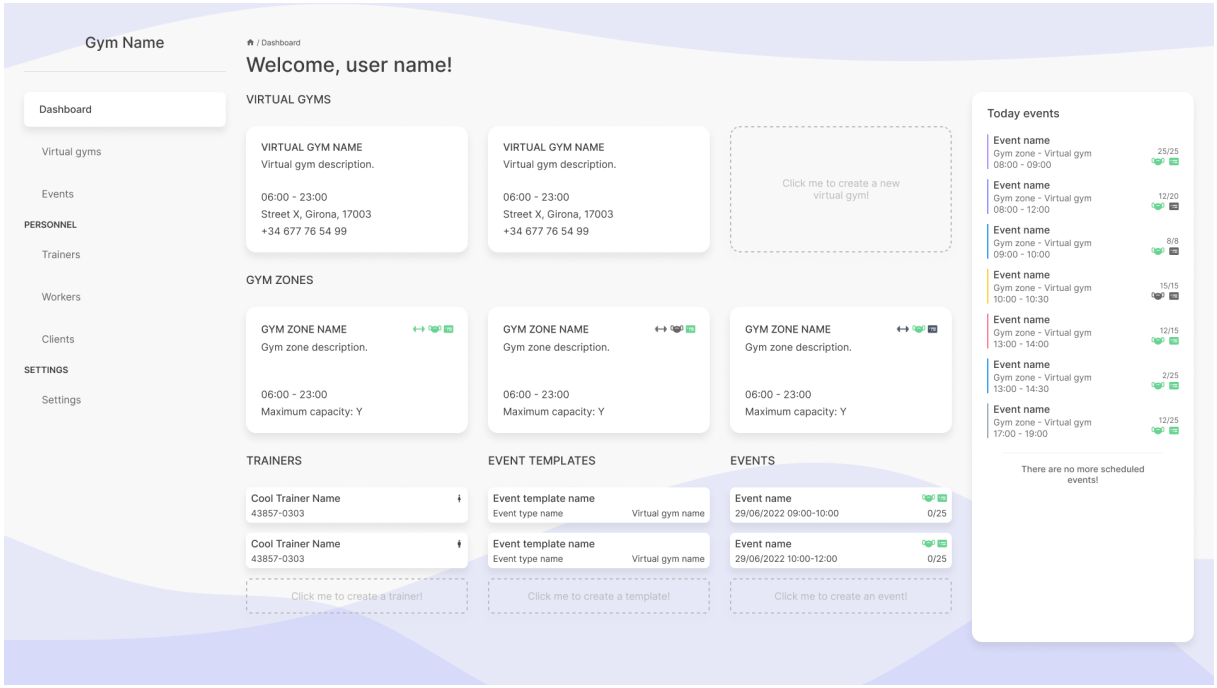

Figure 4.9: Dashboard page, displaying a summary of the gym's information

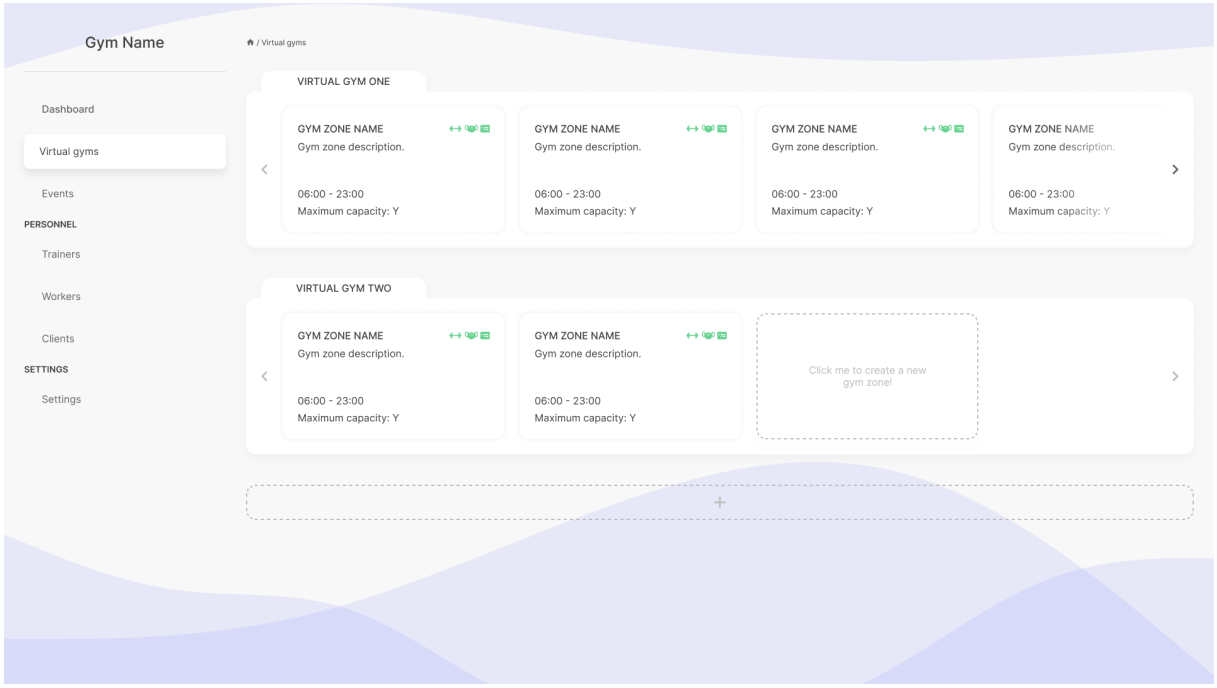

Figure 4.10: Virtual gym's page, which is accessed using the left navigation bar

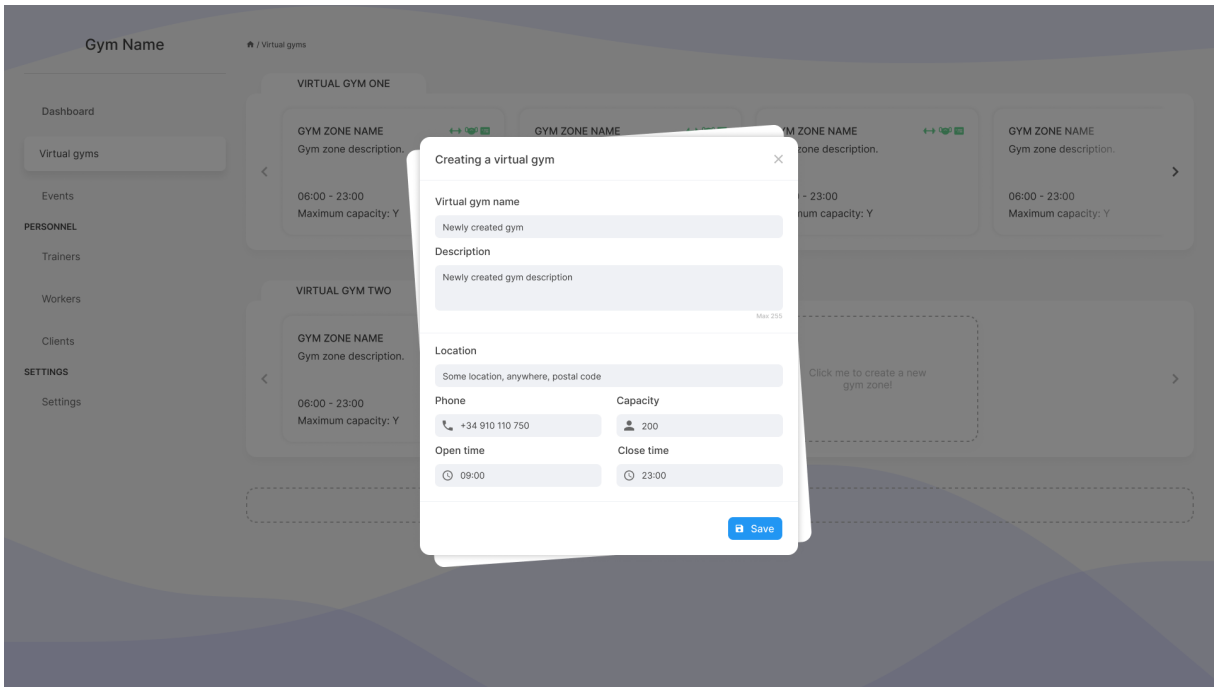

Figure 4.11: Virtual gym dialog (create state)

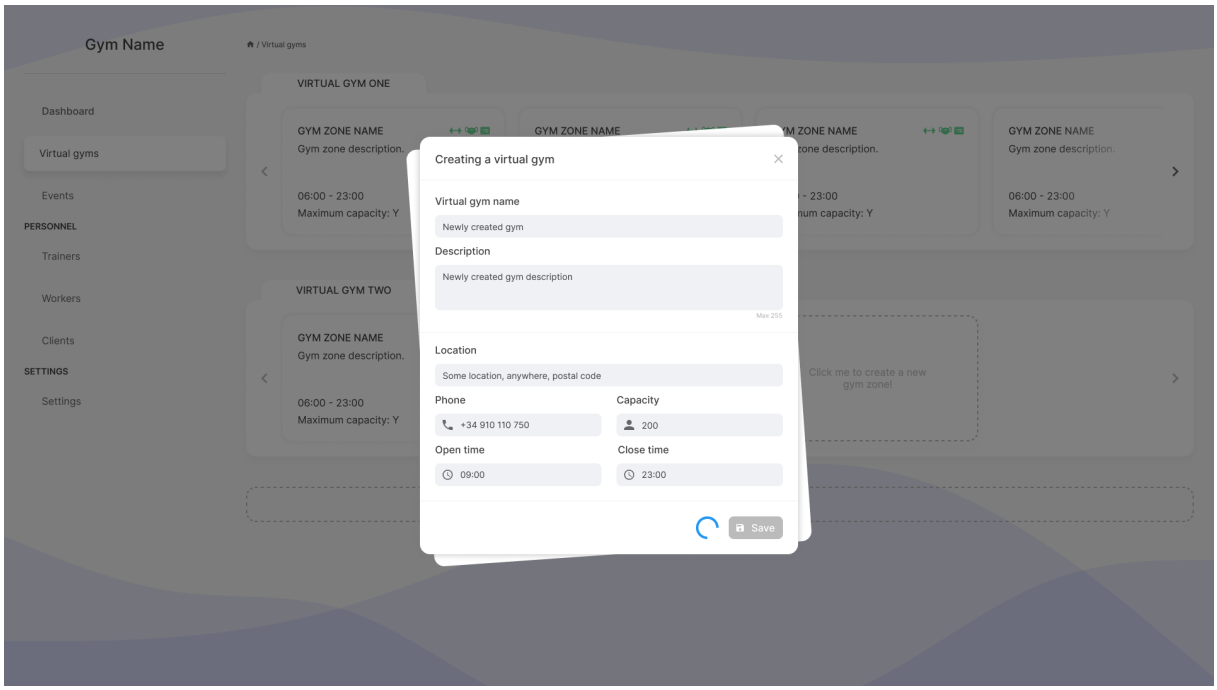

Figure 4.12: Virtual gym dialog (create-loading state)

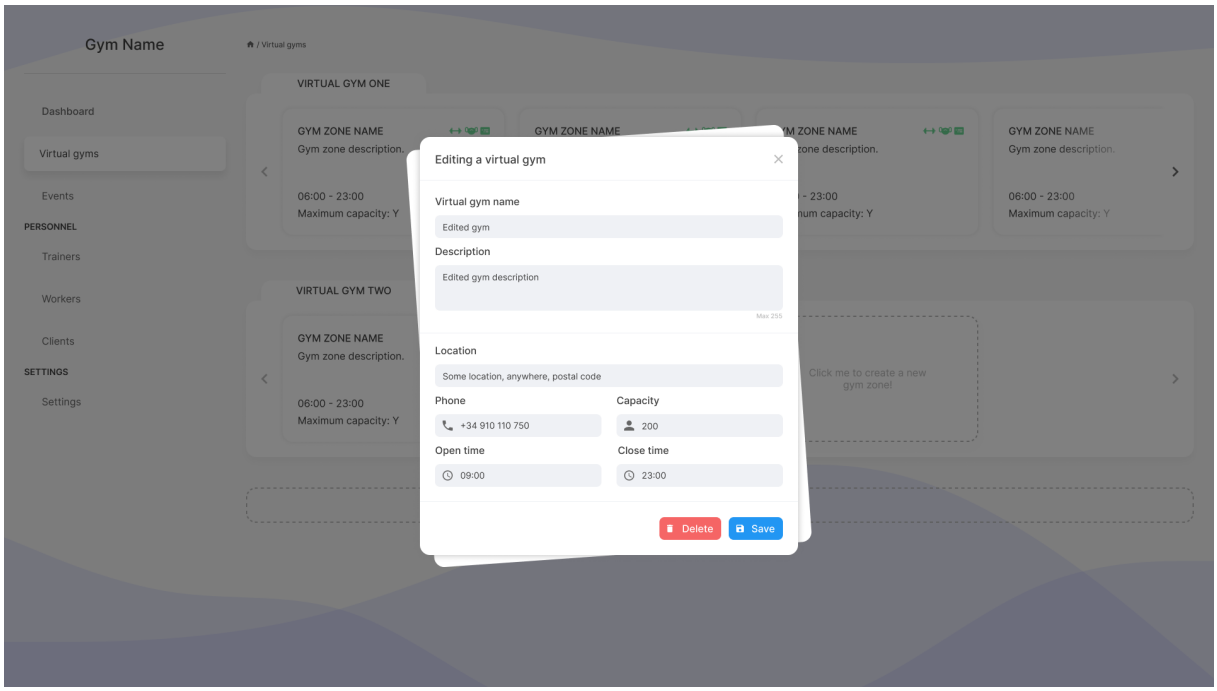

Figure 4.13: Virtual gym dialog (edit state)

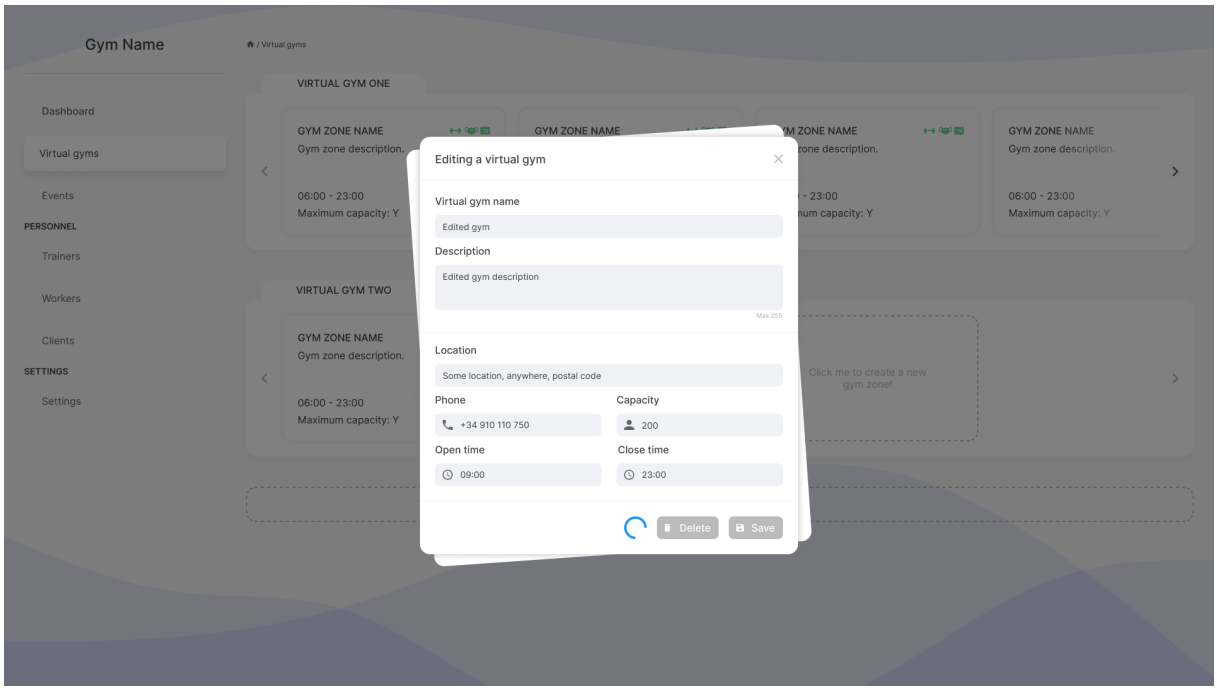

Figure 4.14: Virtual gym dialog (edit-loading state)

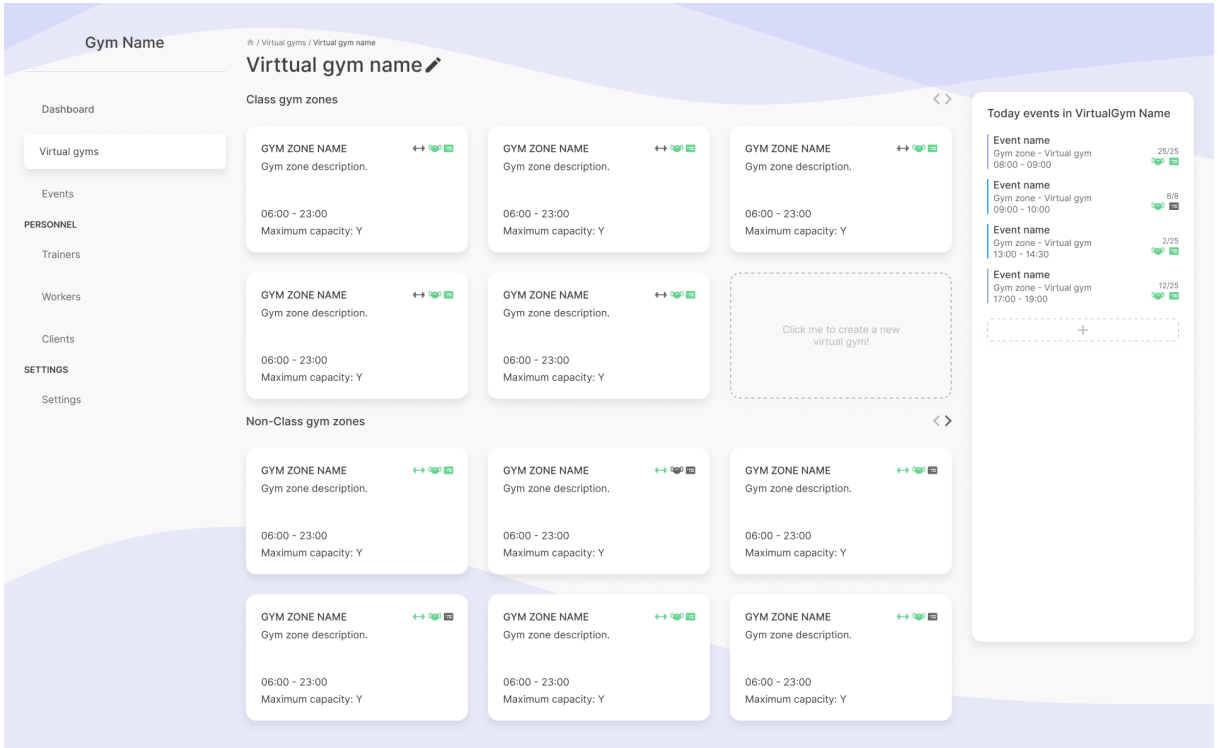

Figure 4.15: Single virtual gym view, accessed by clicking on a virtual gym

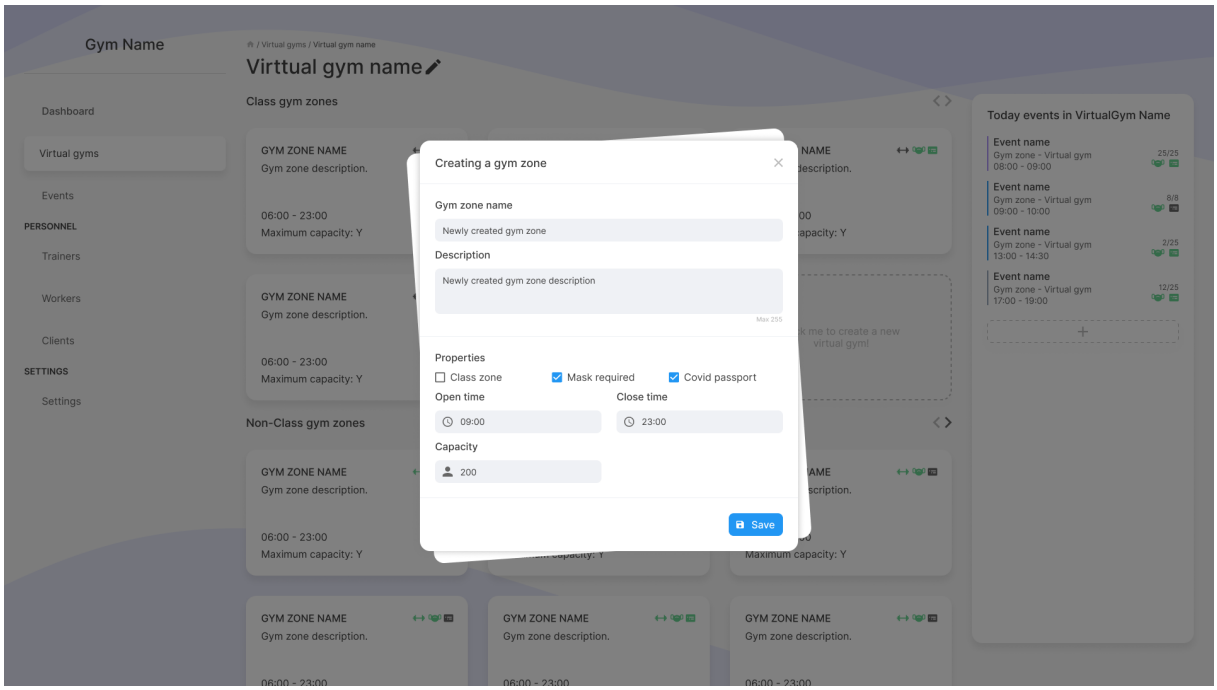

Figure 4.16: Gym zone dialog (create state)

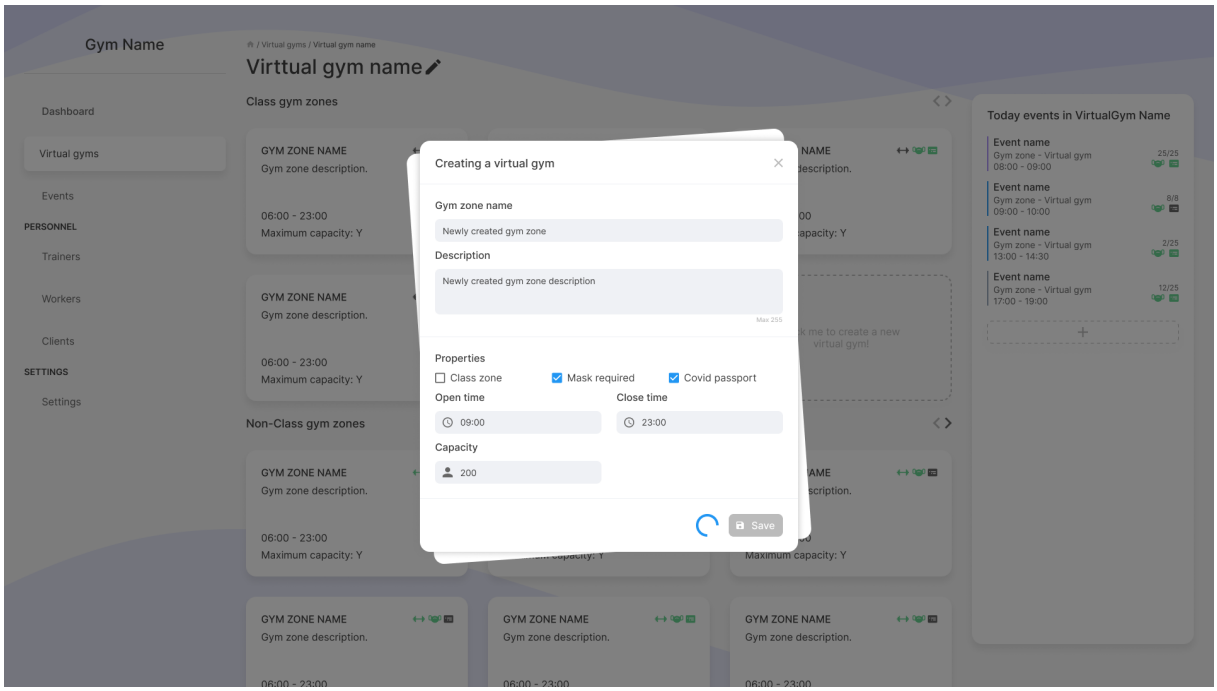

Figure 4.17: Gym zone dialog (create-loading state)

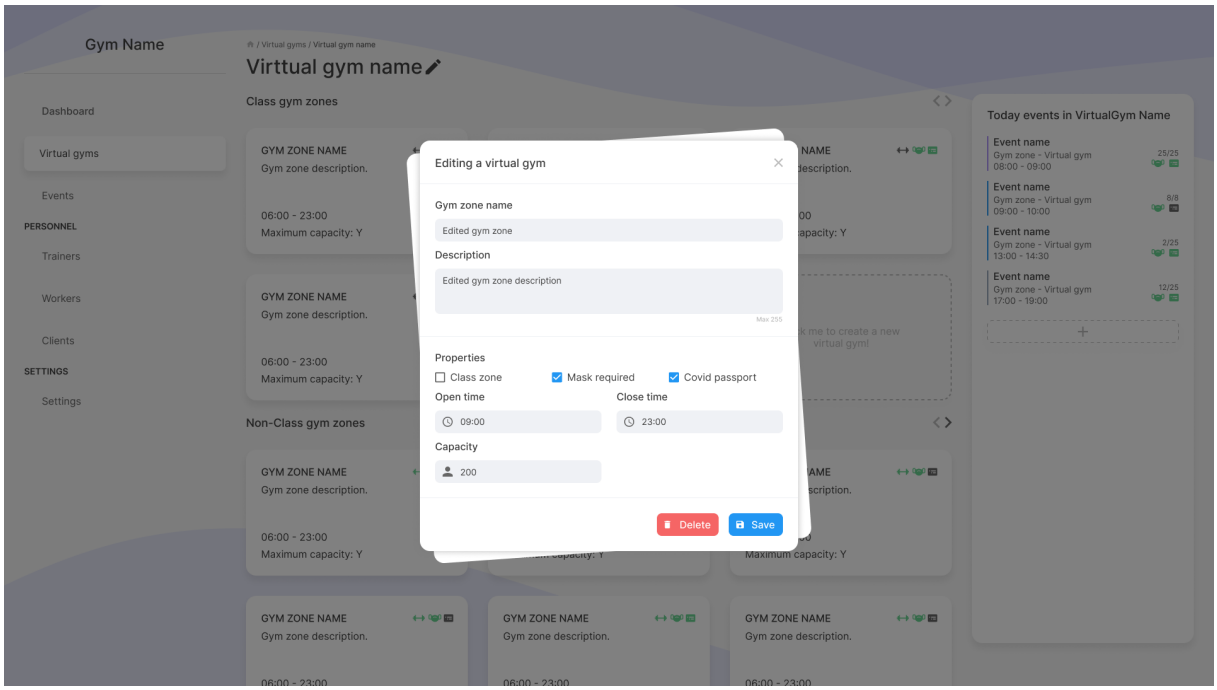

Figure 4.18: Gym zone dialog (edit state)

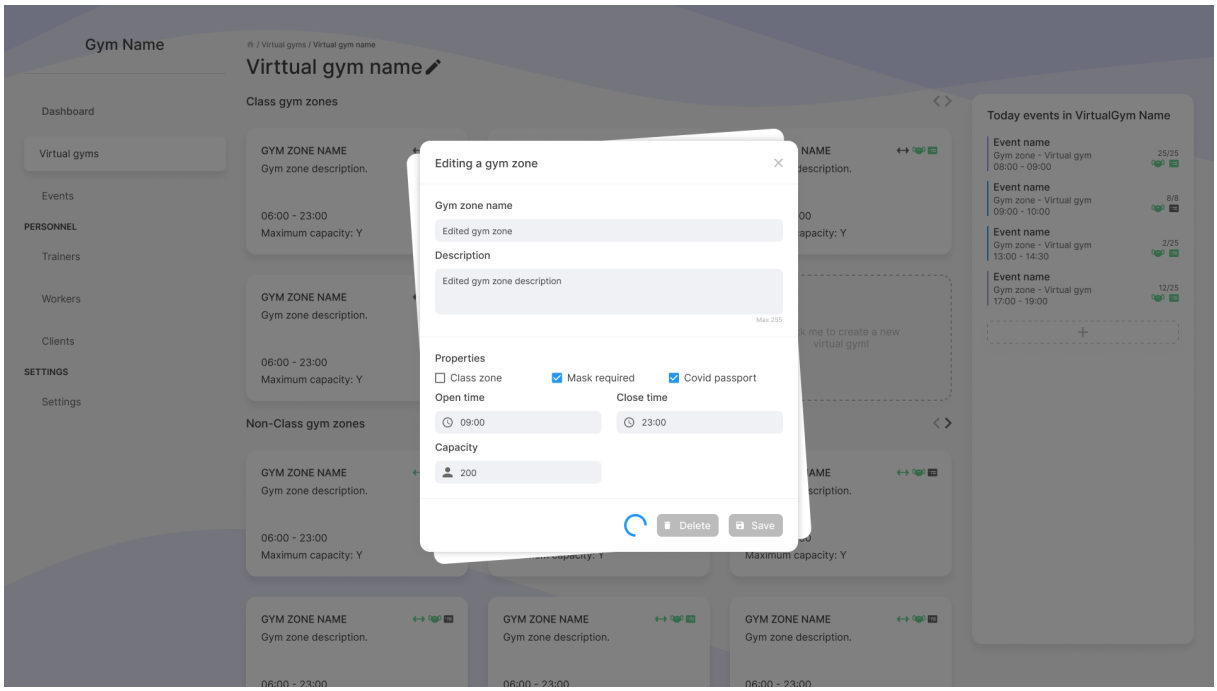

Figure 4.19: Gym zone dialog (edit-loading state)

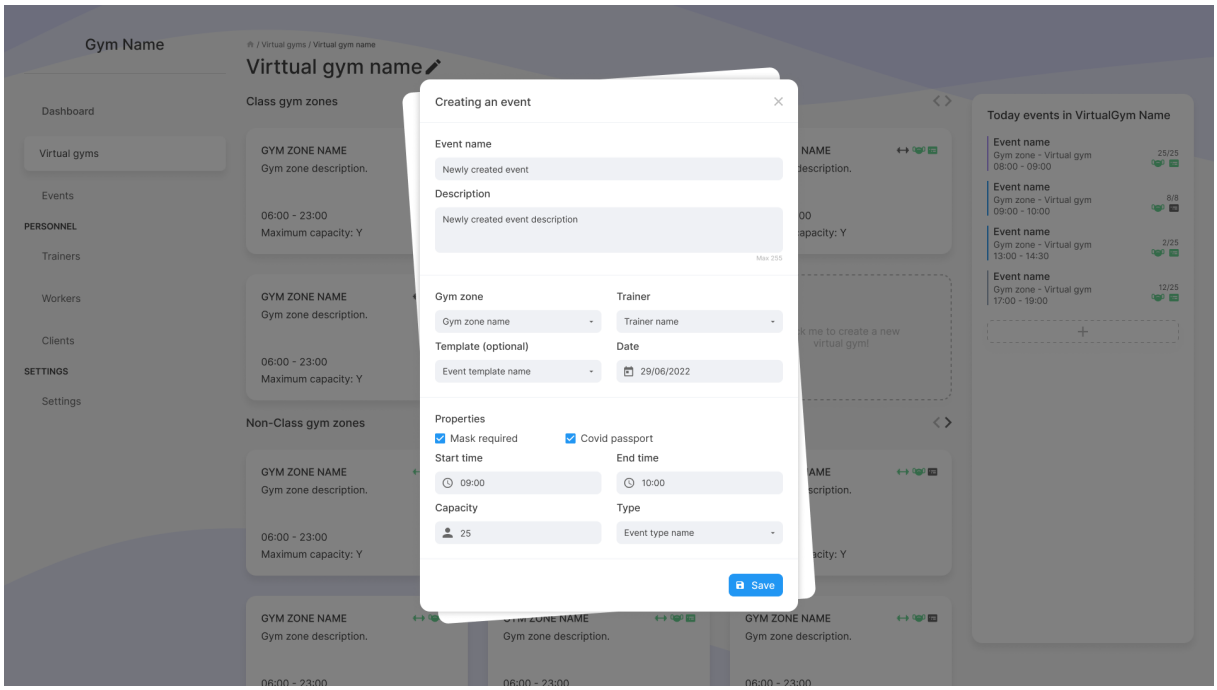

Figure 4.20: Event dialog (create state)

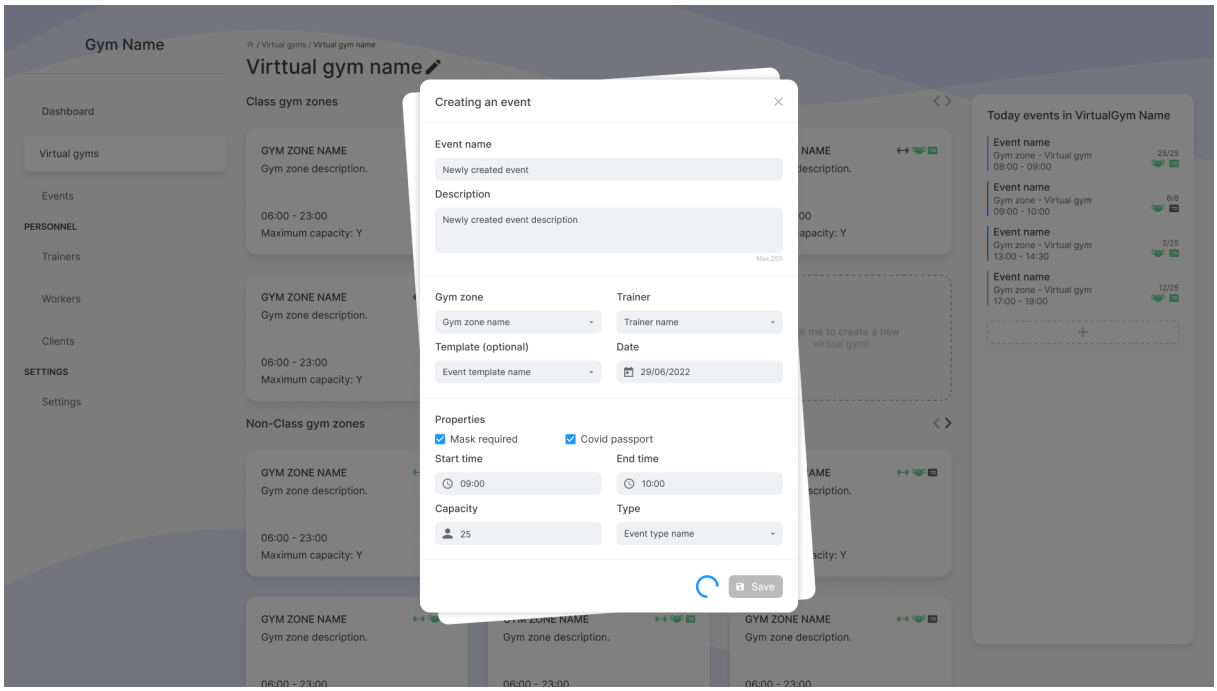

Figure 4.21: Event dialog (create-loading state)

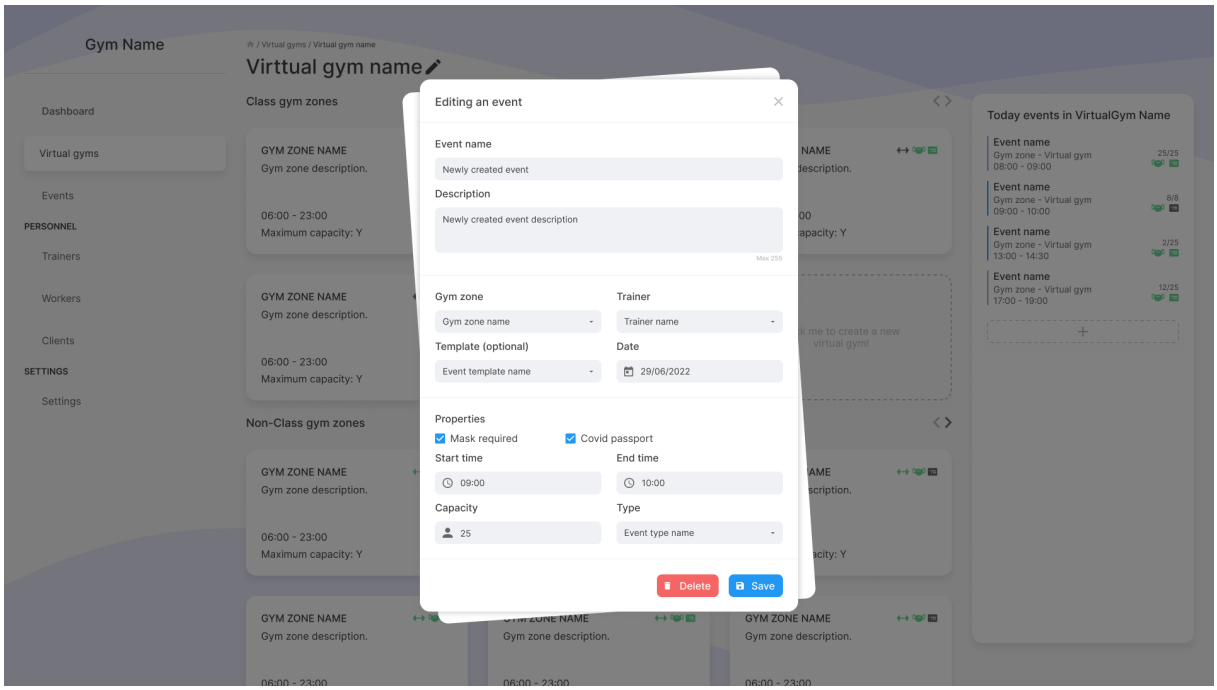

Figure 4.22: Event dialog (edit state)

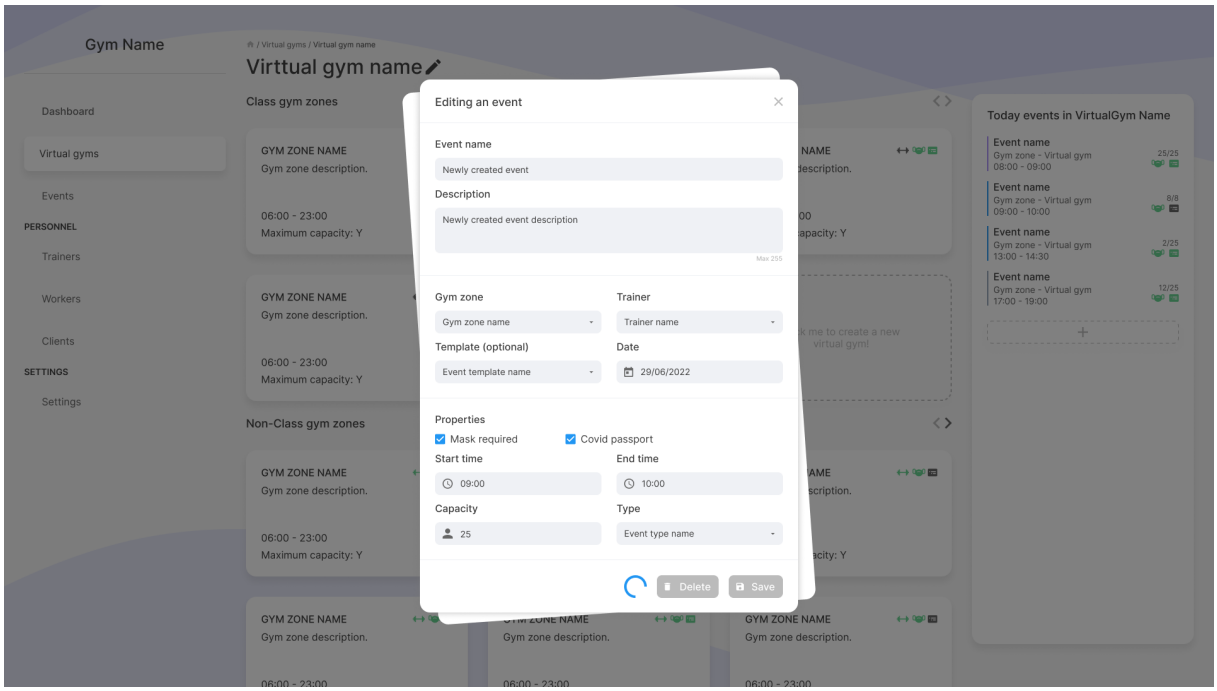

Figure 4.23: Event dialog (edit-loading state)

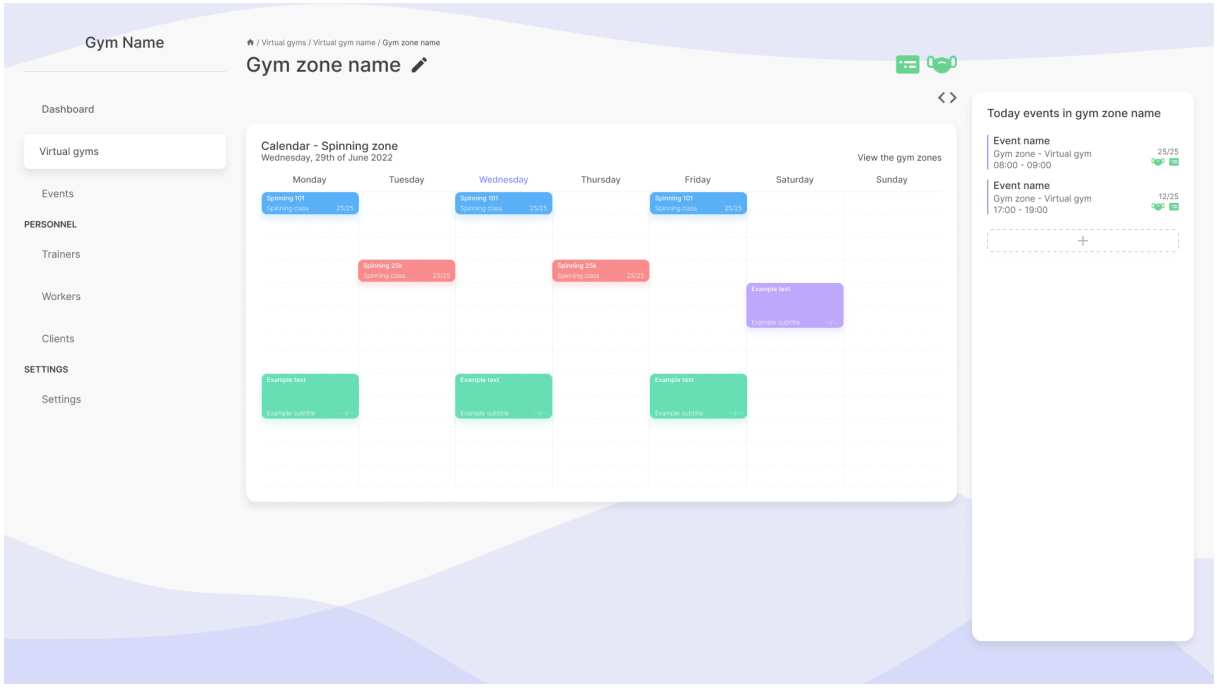

Figure 4.24: Class gym zone page, accessed by clicking on any class-type gym zone

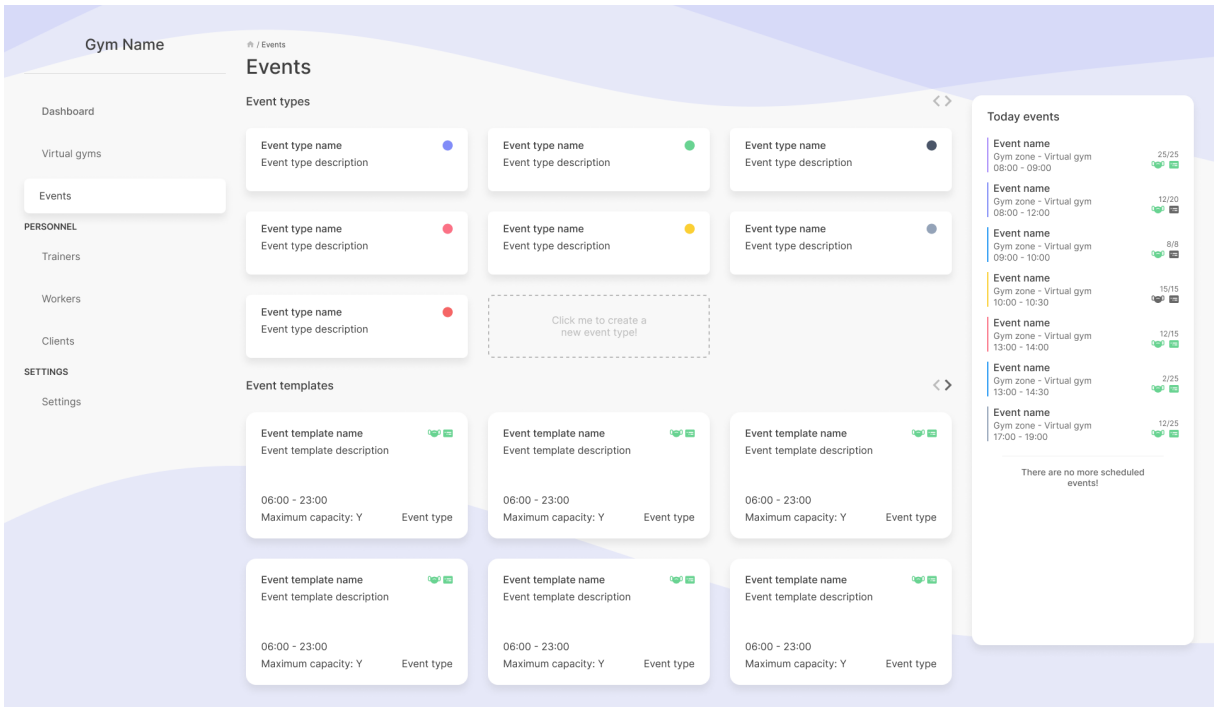

Figure 4.25: Event's page, which is accessed using the left navigation bar

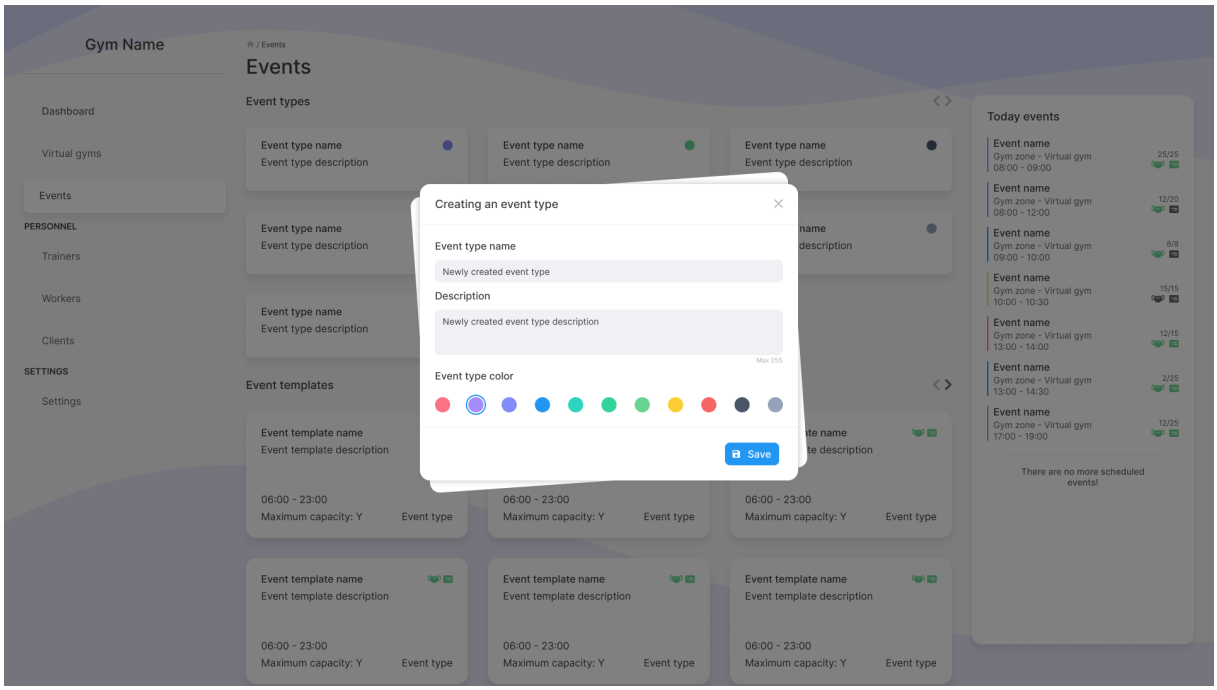

Figure 4.26: Event type dialog (create state)

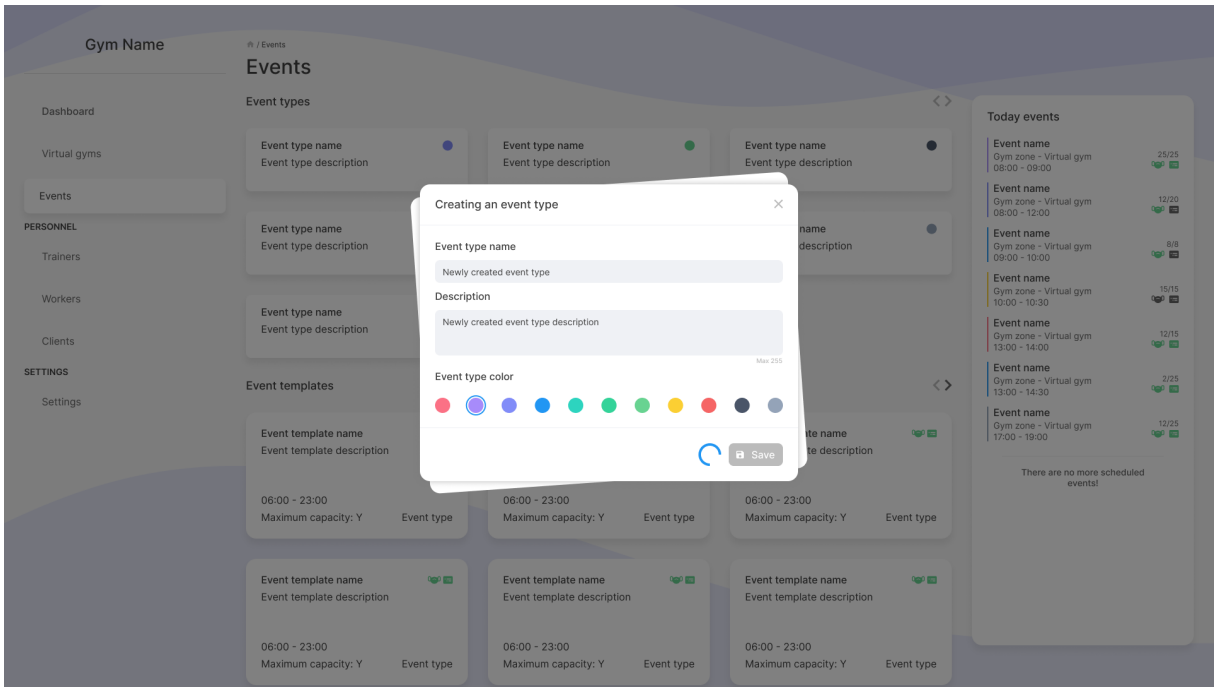

Figure 4.27: Event type dialog (create-loading state)

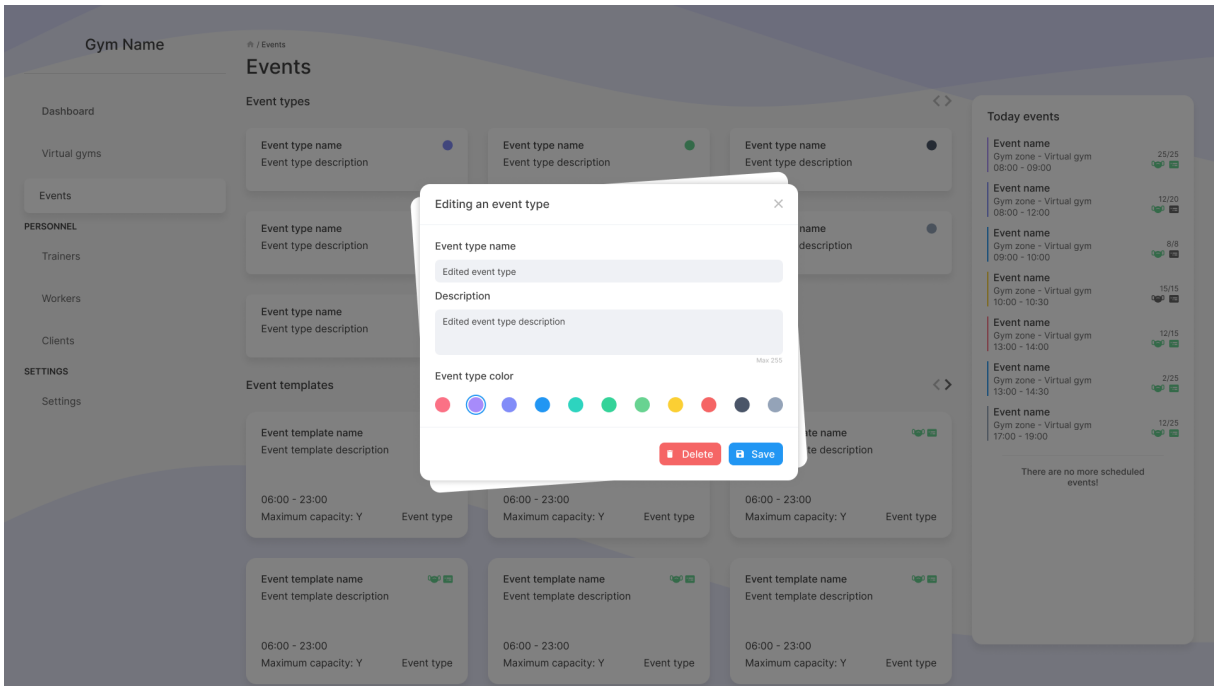

Figure 4.28: Event type dialog (edit state)

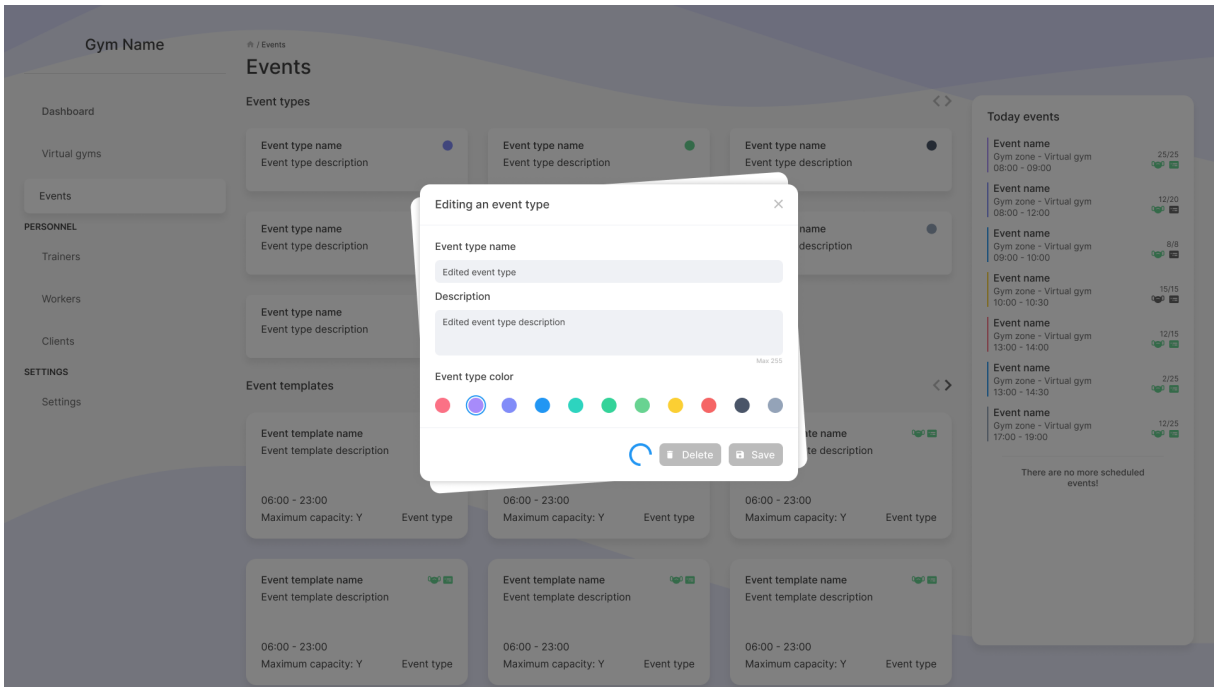

Figure 4.29: Event type dialog (edit-loading state)

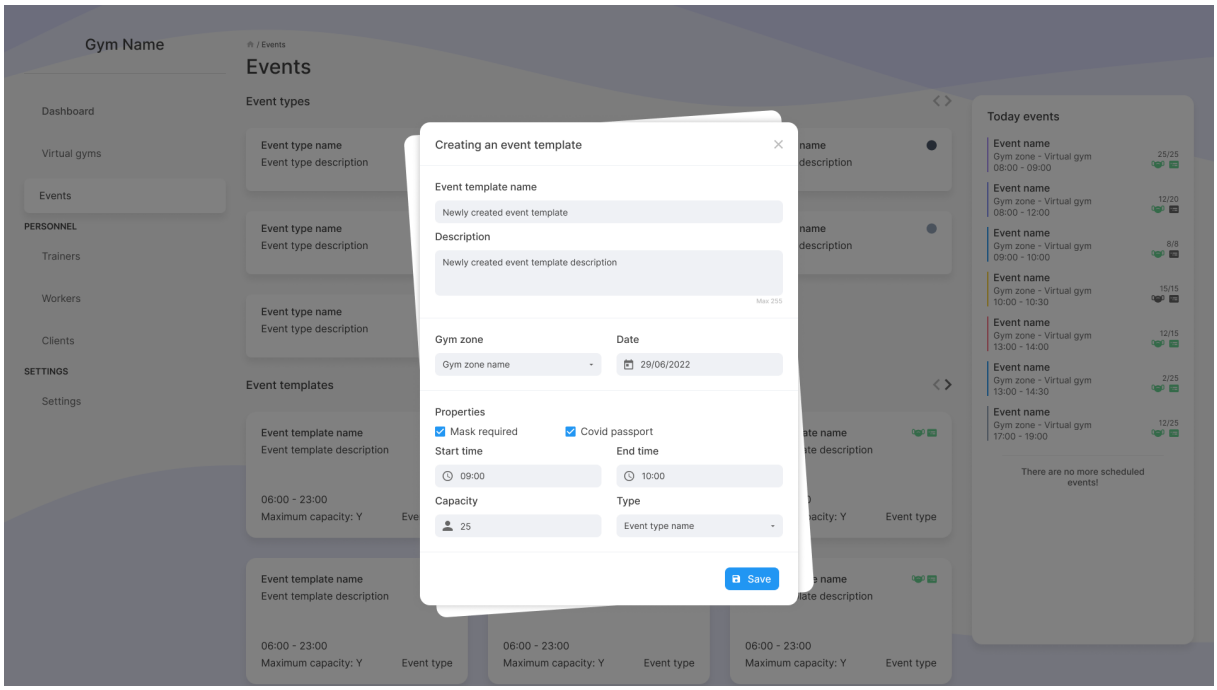

Figure 4.30: Event template dialog (create state)

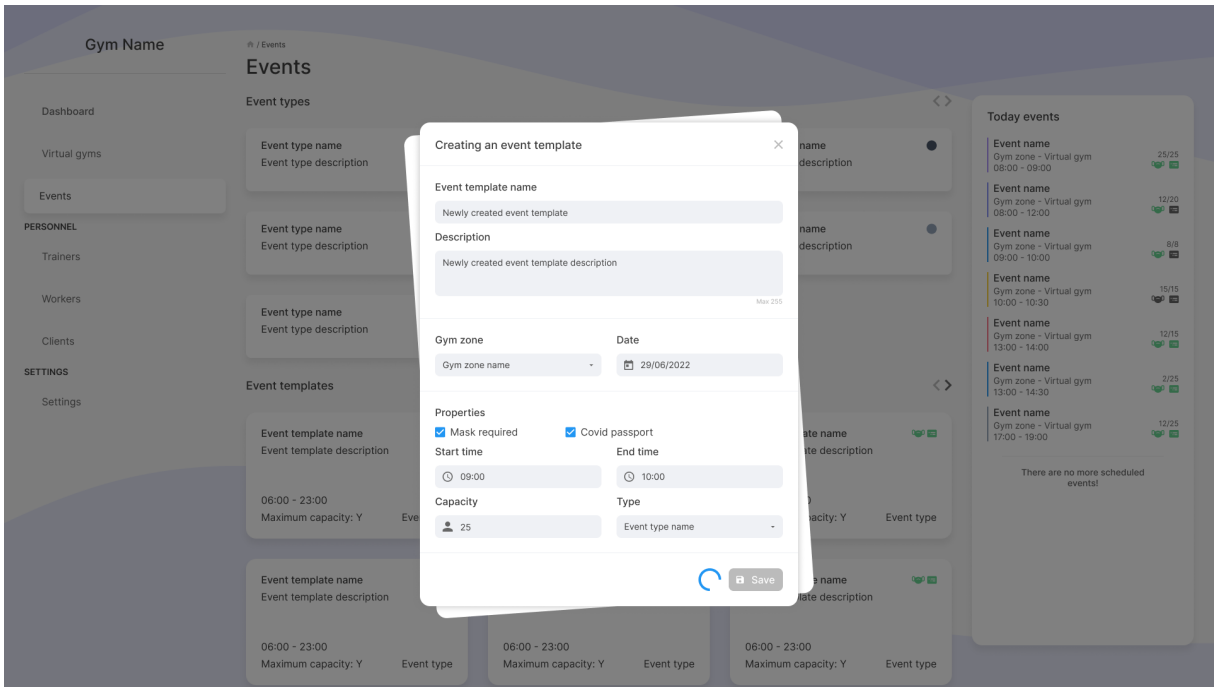

Figure 4.31: Event template dialog (create-loading state)

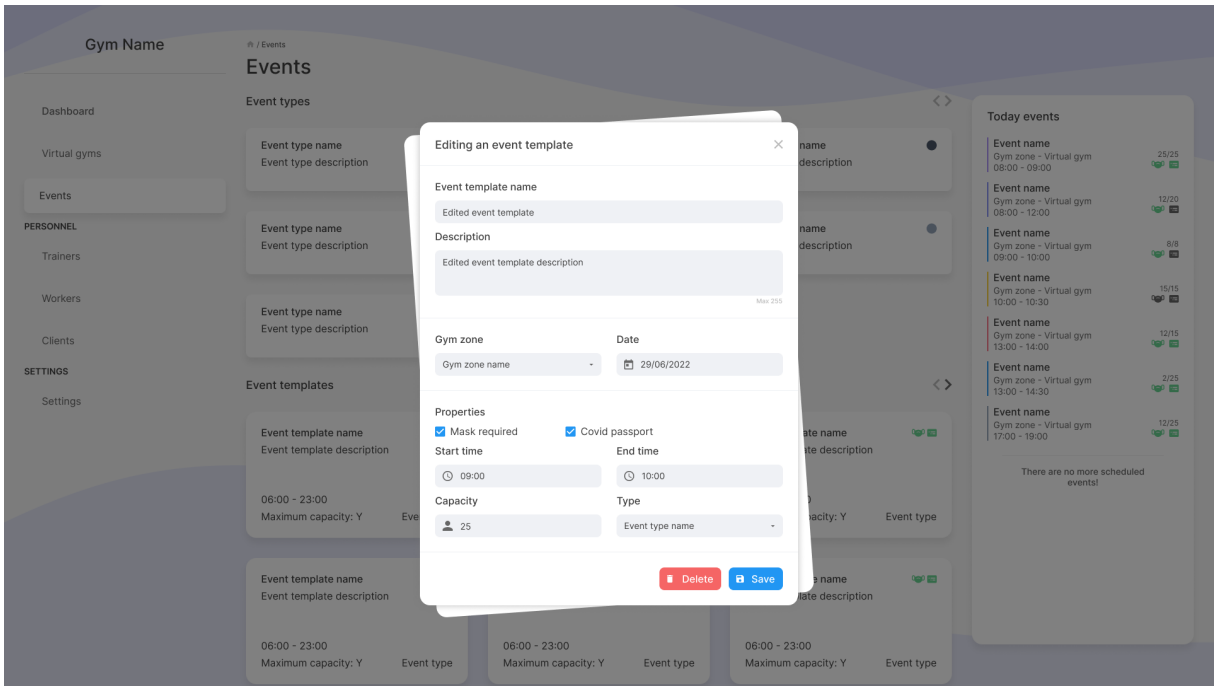

Figure 4.32: Event template dialog (edit state)

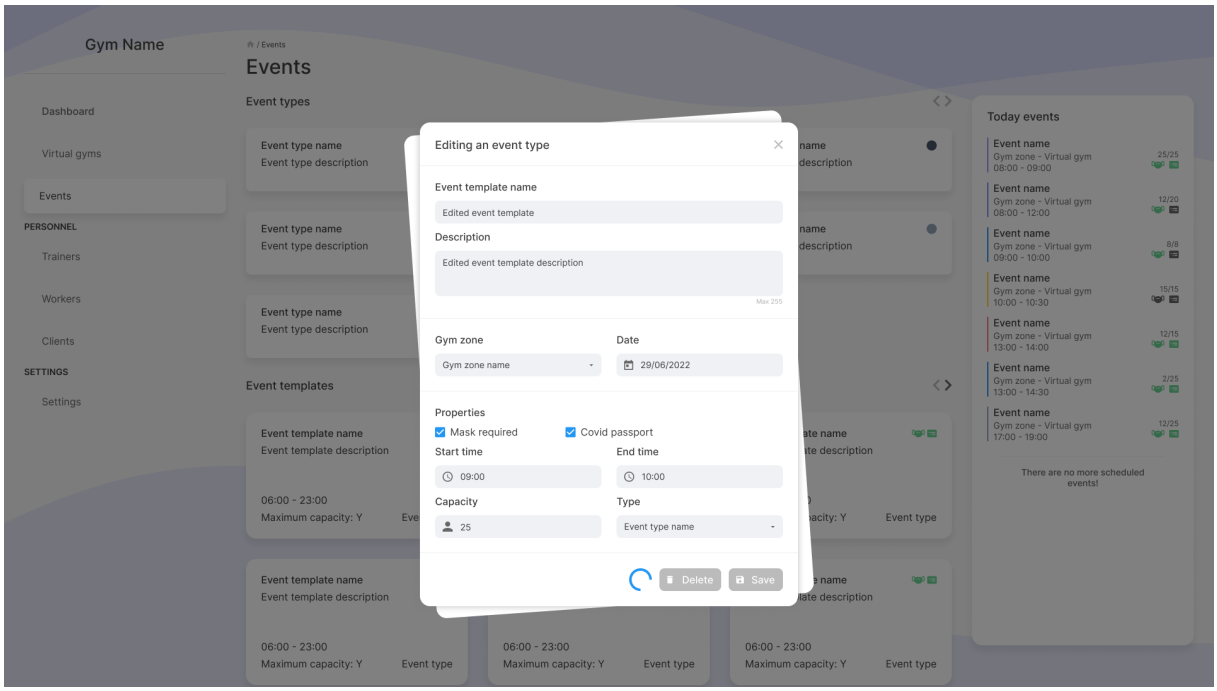

Figure 4.33: Event template dialog (edit-loading state)

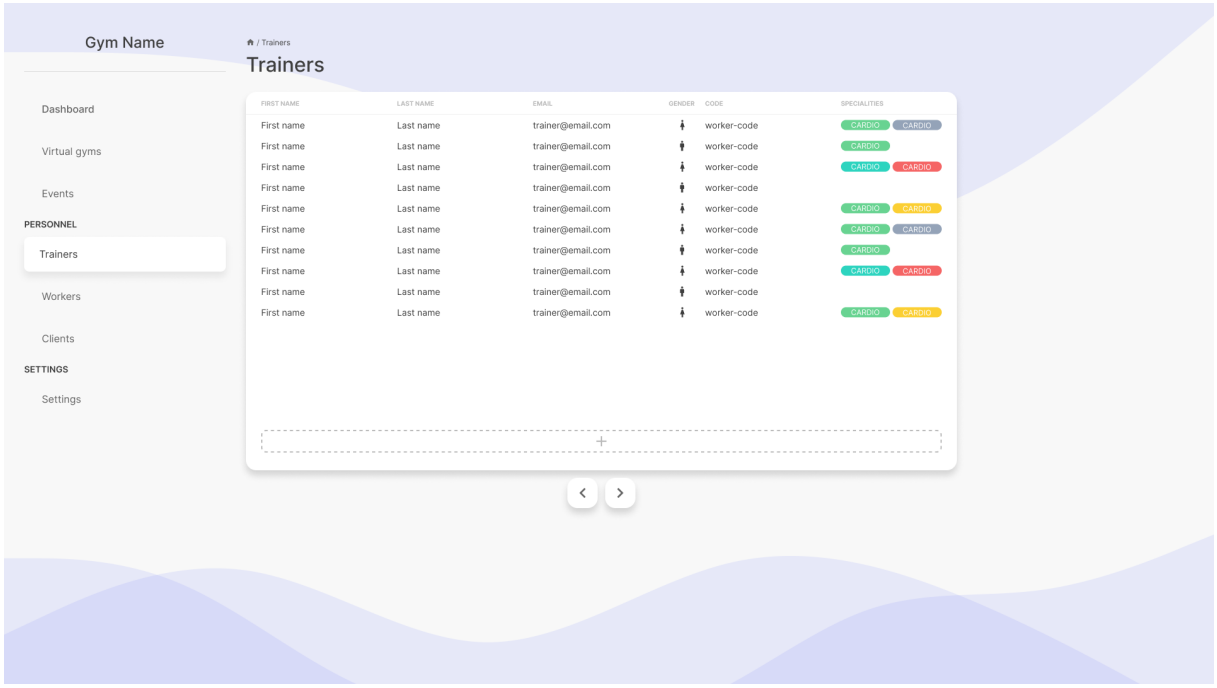

Figure 4.34: Trainer's page, which is accessed using the left navigation bar

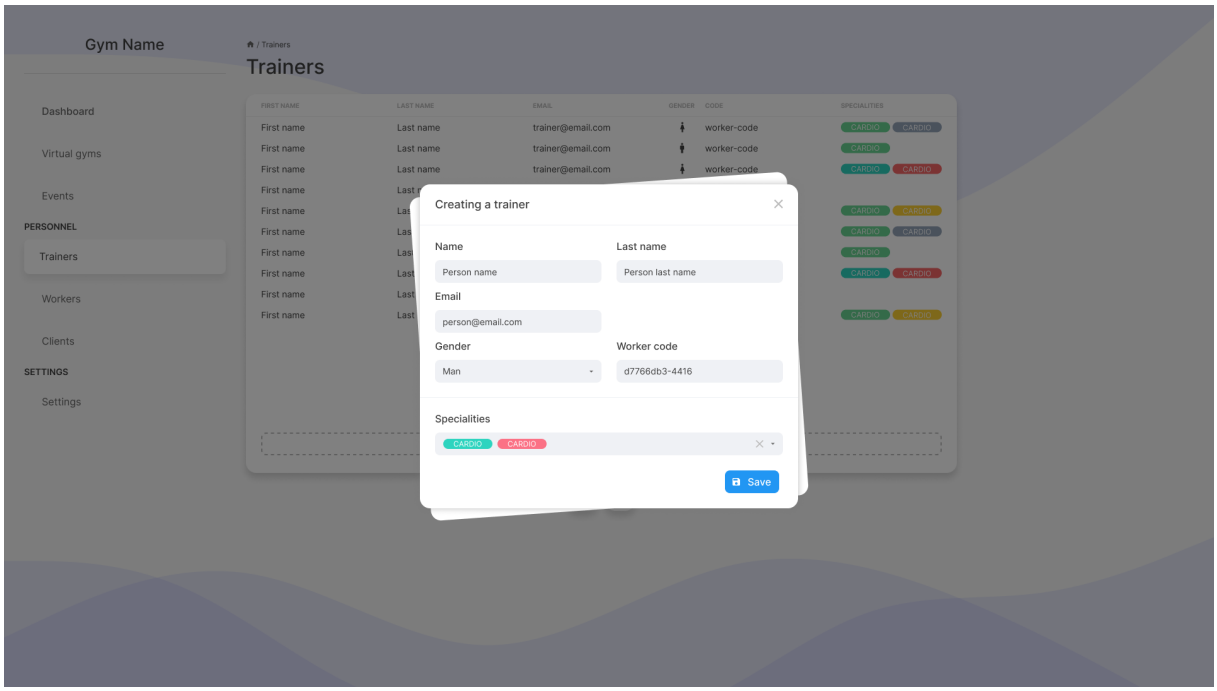

Figure 4.35: Trainer dialog (create state)

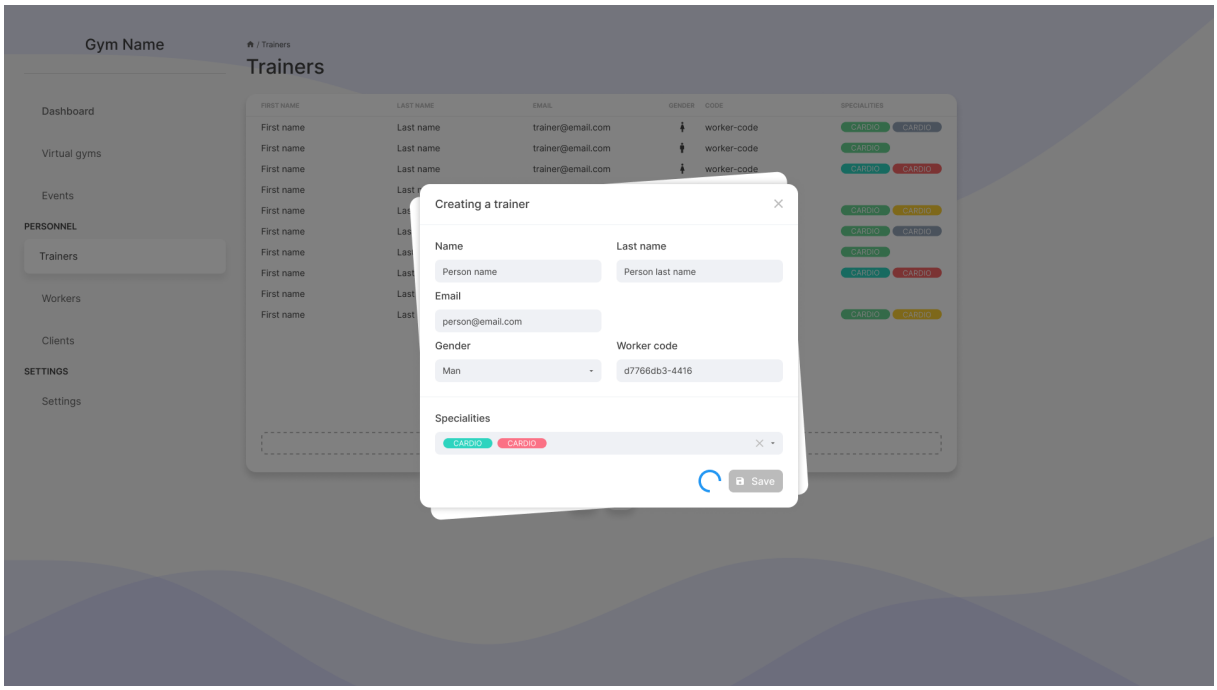

Figure 4.36: Trainer dialog (create-edit state)

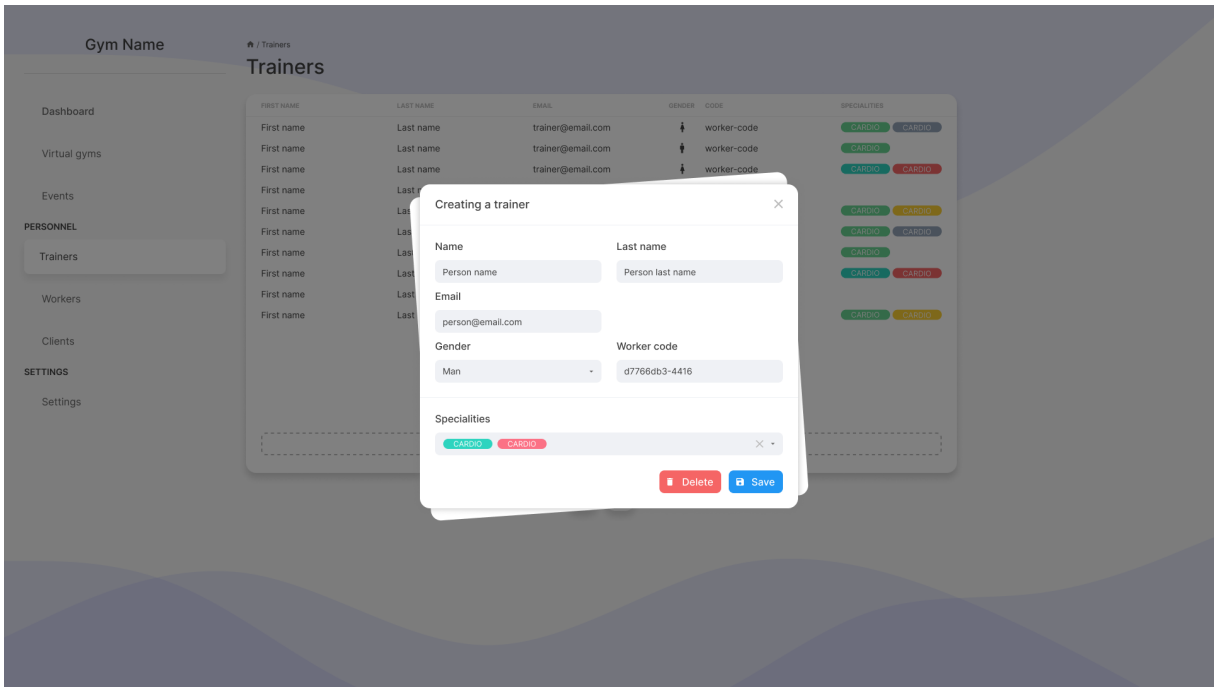

Figure 4.37: Trainer dialog (edit state)

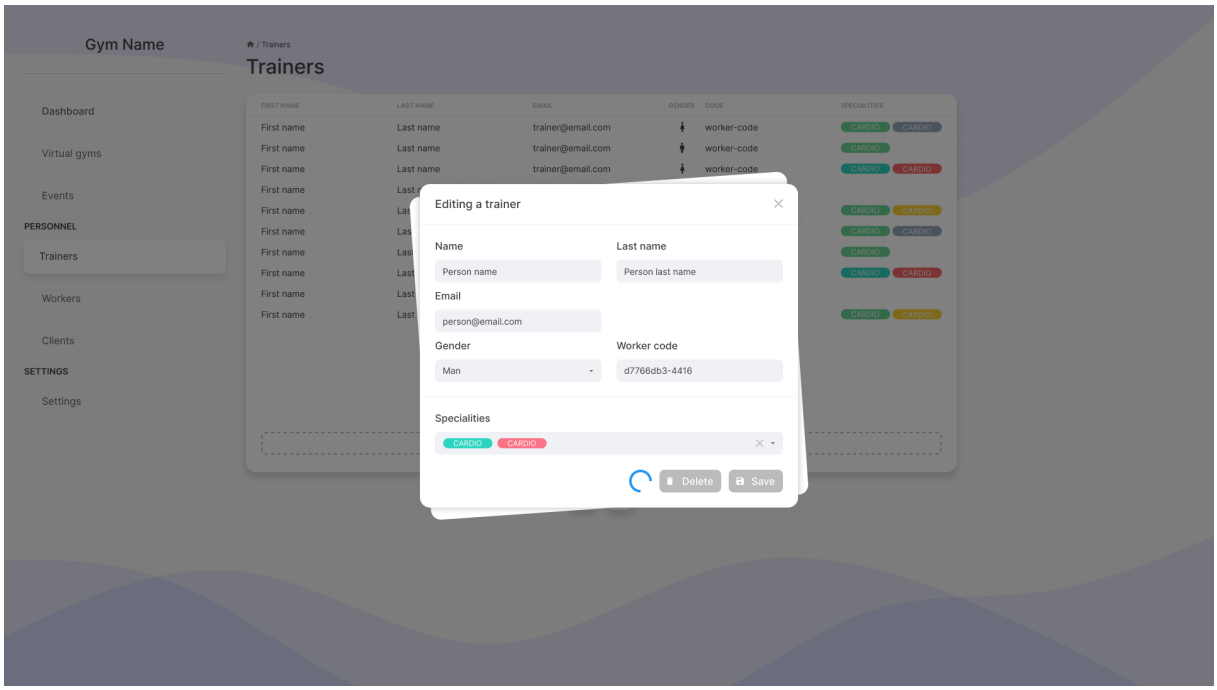

Figure 4.38: Trainer dialog (edit-loading state)

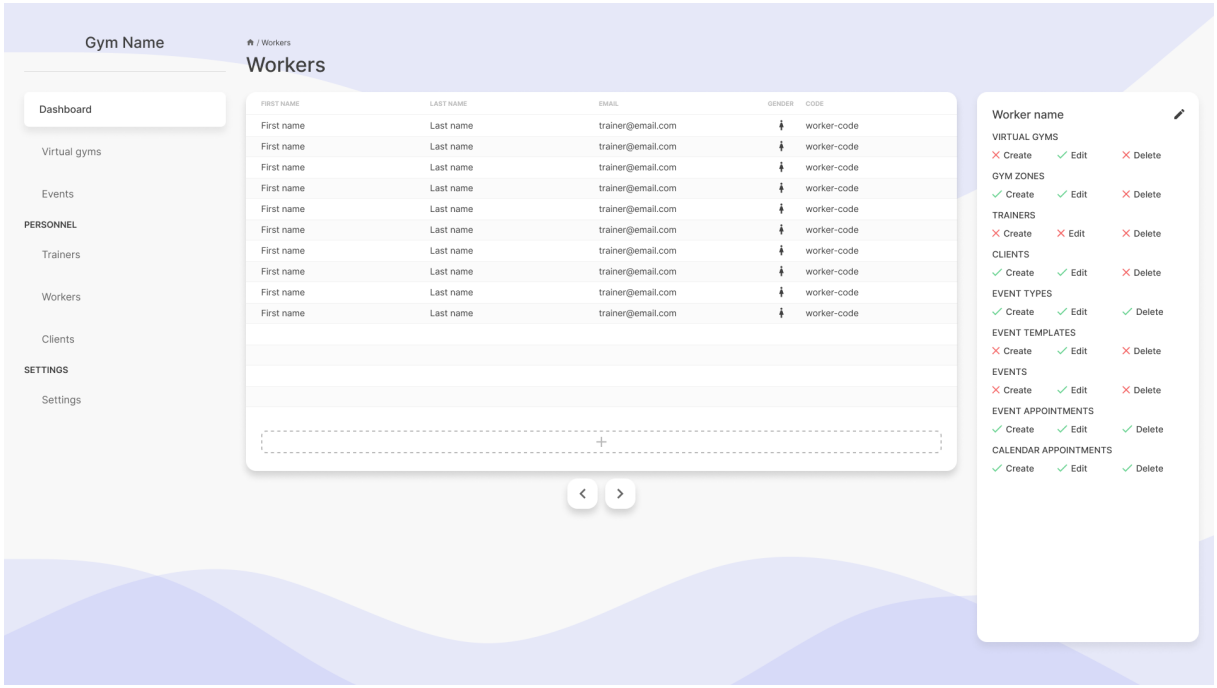

Figure 4.39: Worker's page, which is accessed using the left navigation bar

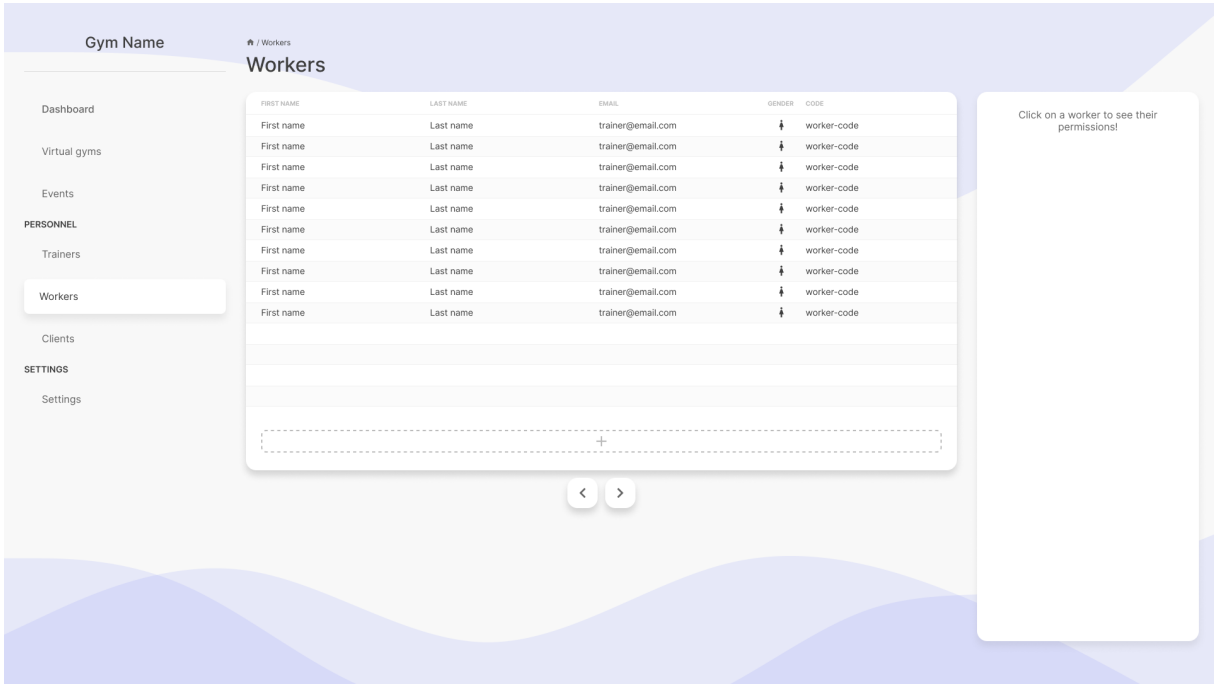

Figure 4.40: Same as previous, with a selected worker

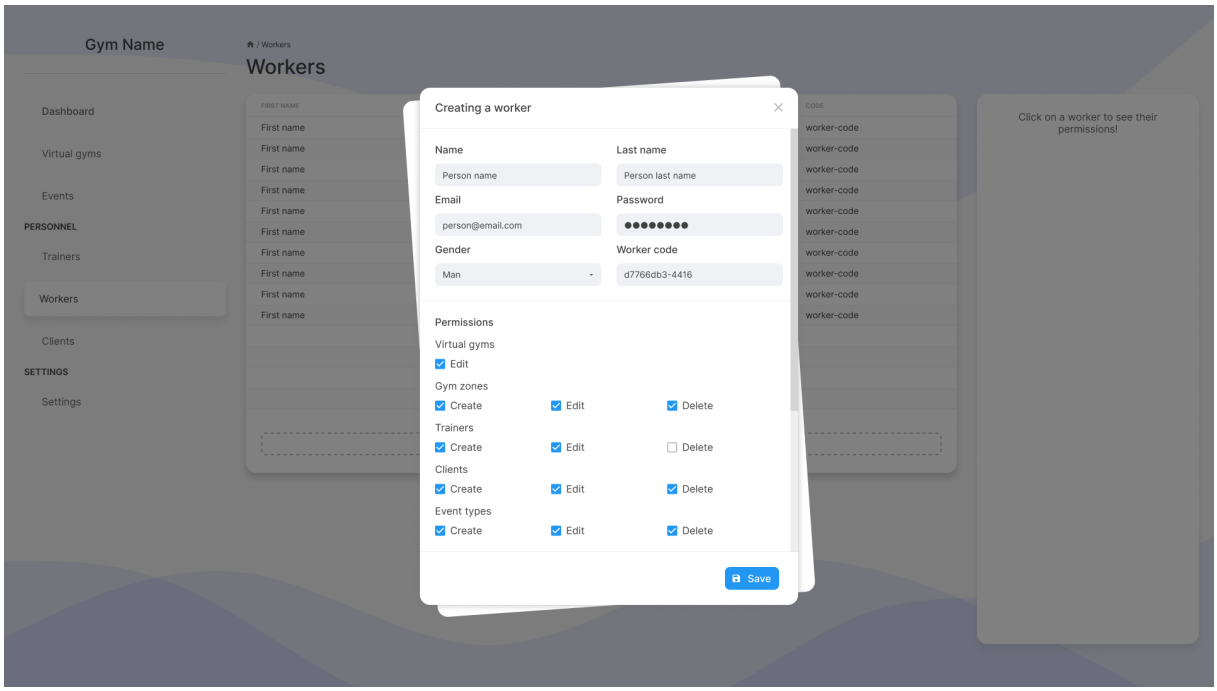

Figure 4.41: Worker dialog (create state)

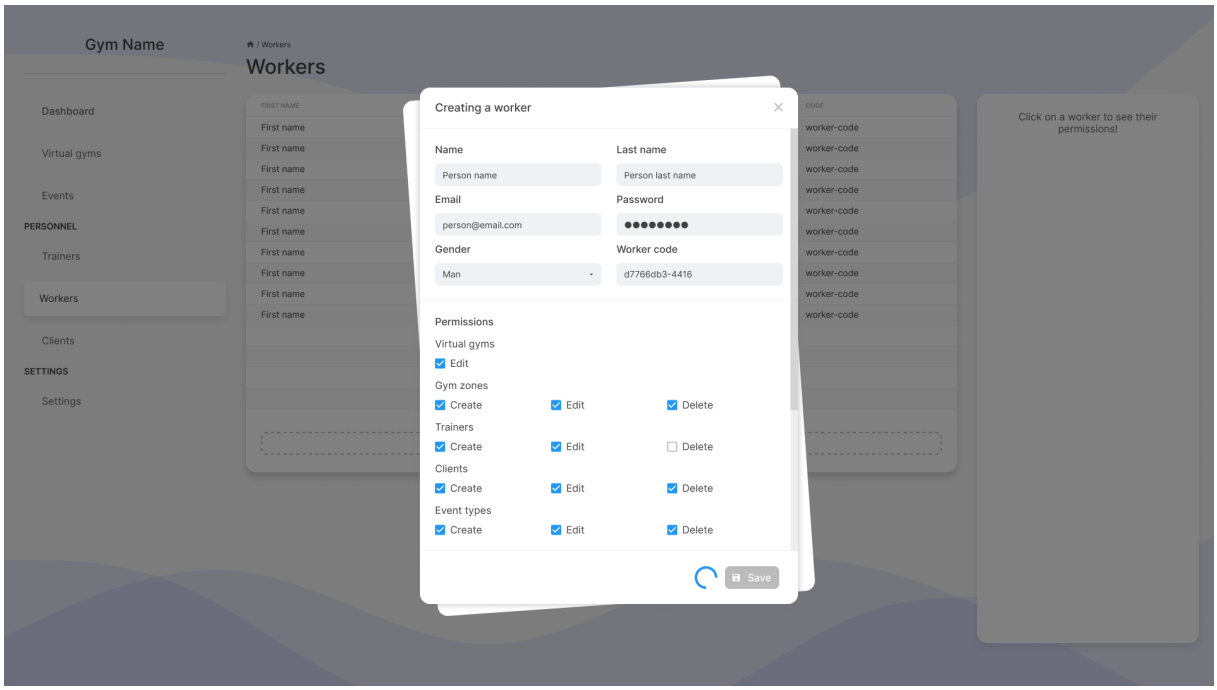

Figure 4.42: Worker dialog (create-loading state)

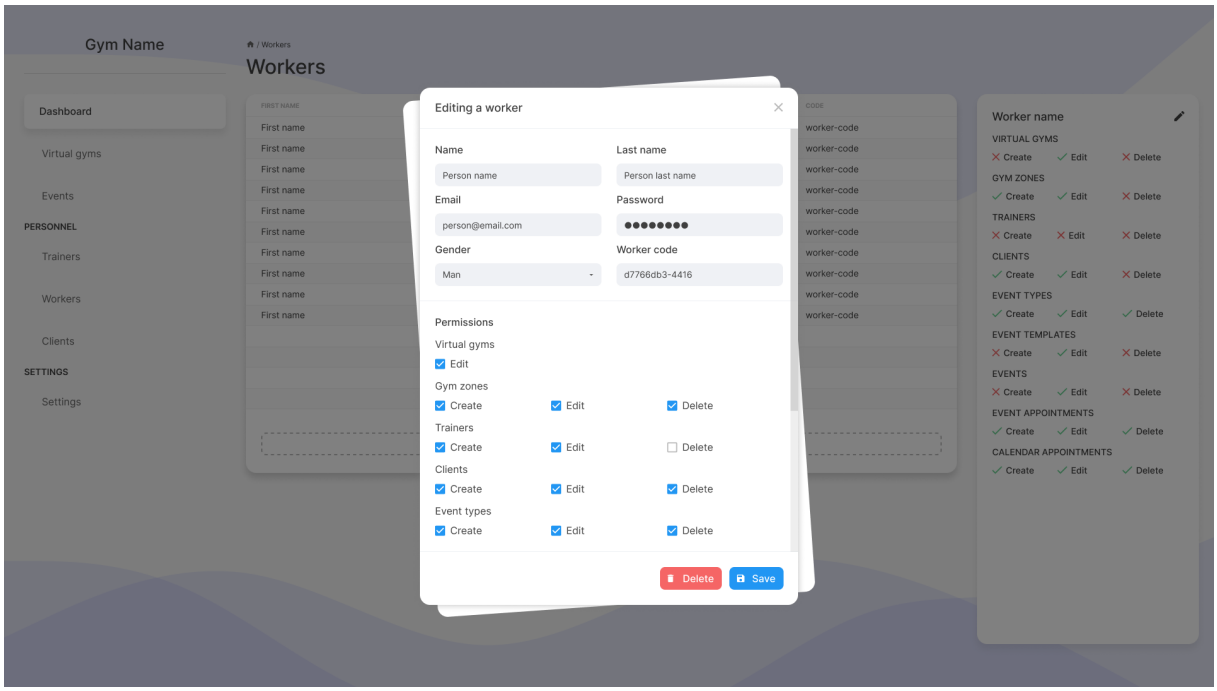

Figure 4.43: Worker dialog (edit state)

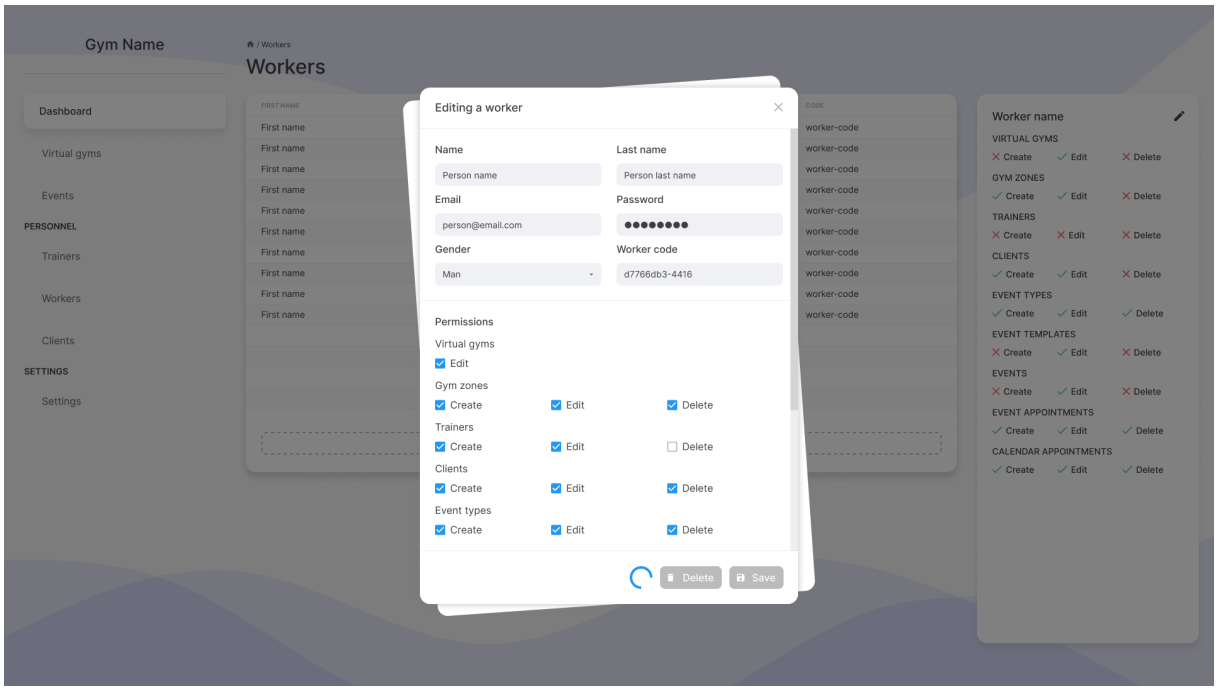

Figure 4.44: Worker dialog (edit-loading state)

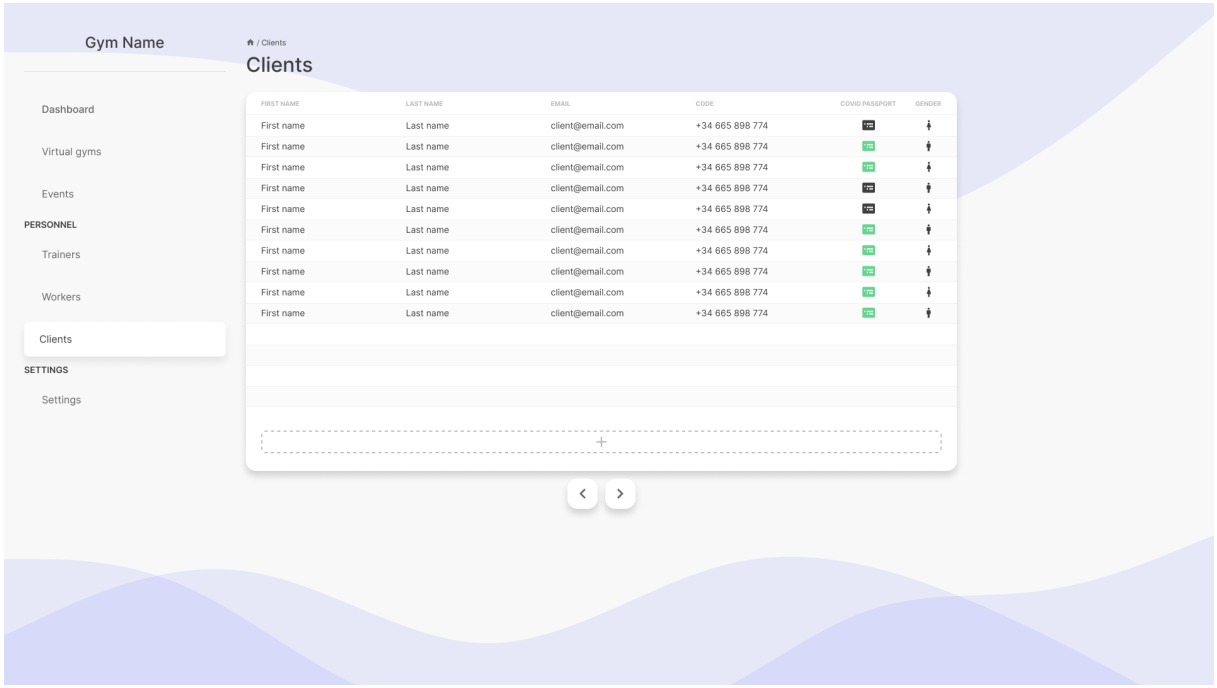

Figure 4.45: Client's page, which is accessed using the left navigation bar

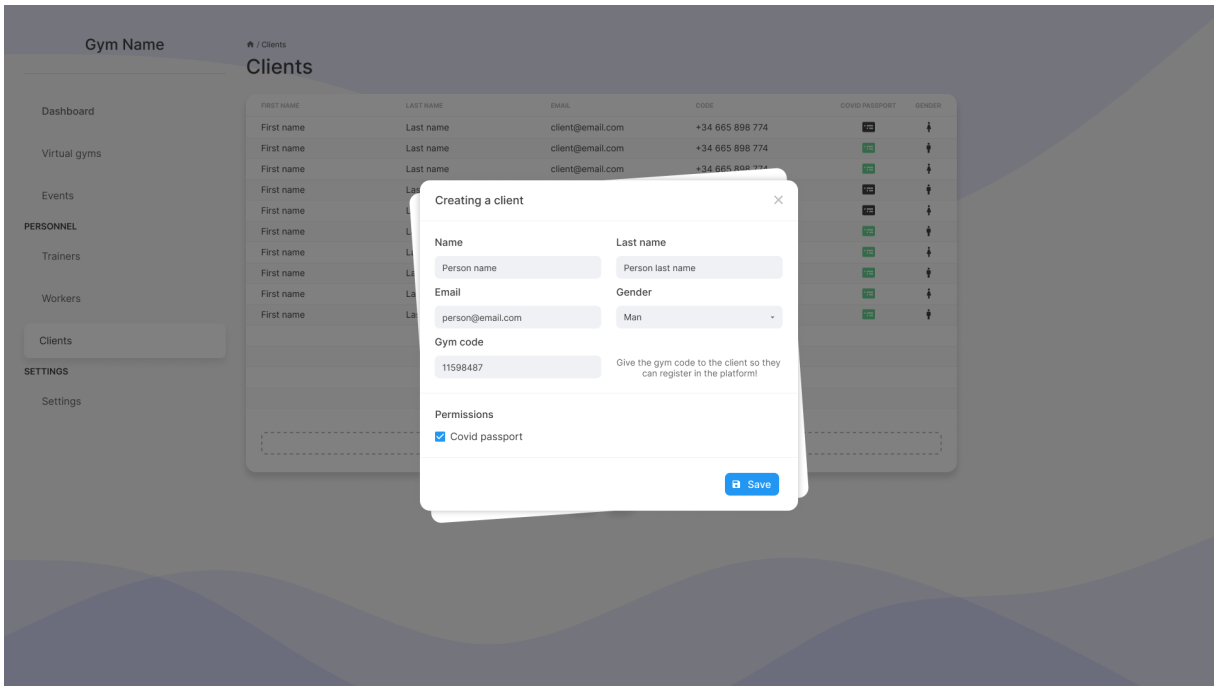

Figure 4.46: Client dialog (create state)

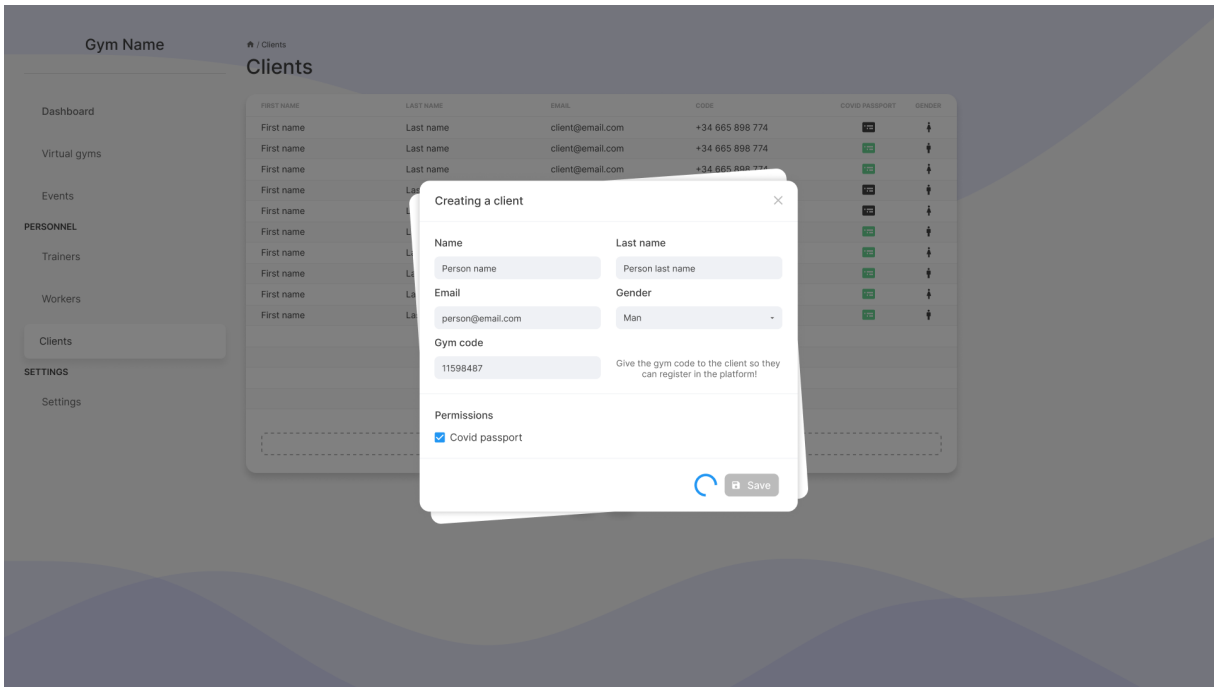

Figure 4.47: Client dialog (create-loading state)

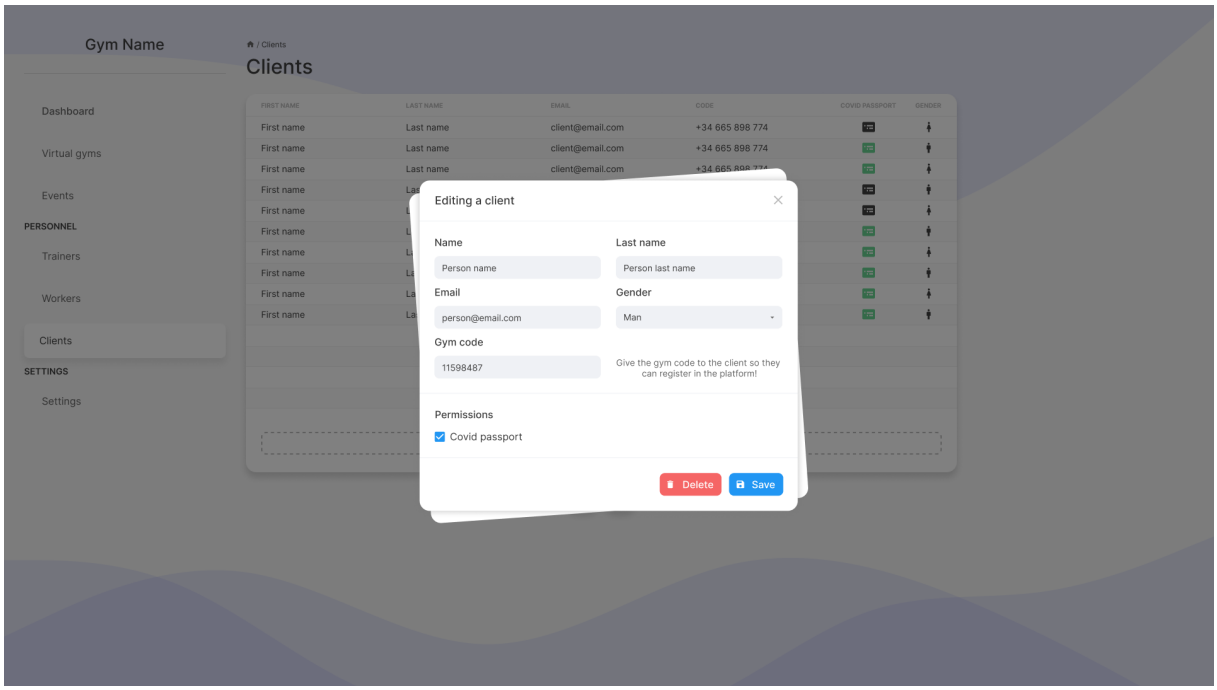

Figure 4.48: Client dialog (edit state)

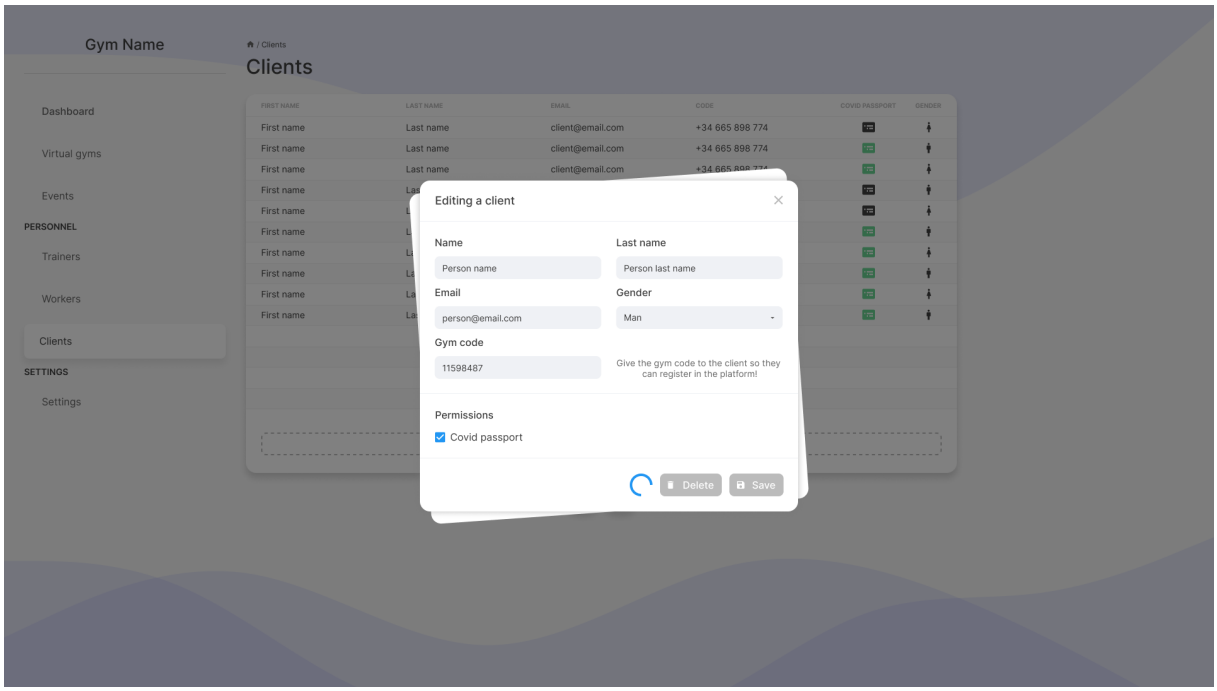

Figure 4.49: Client dialog (edit-loading state)

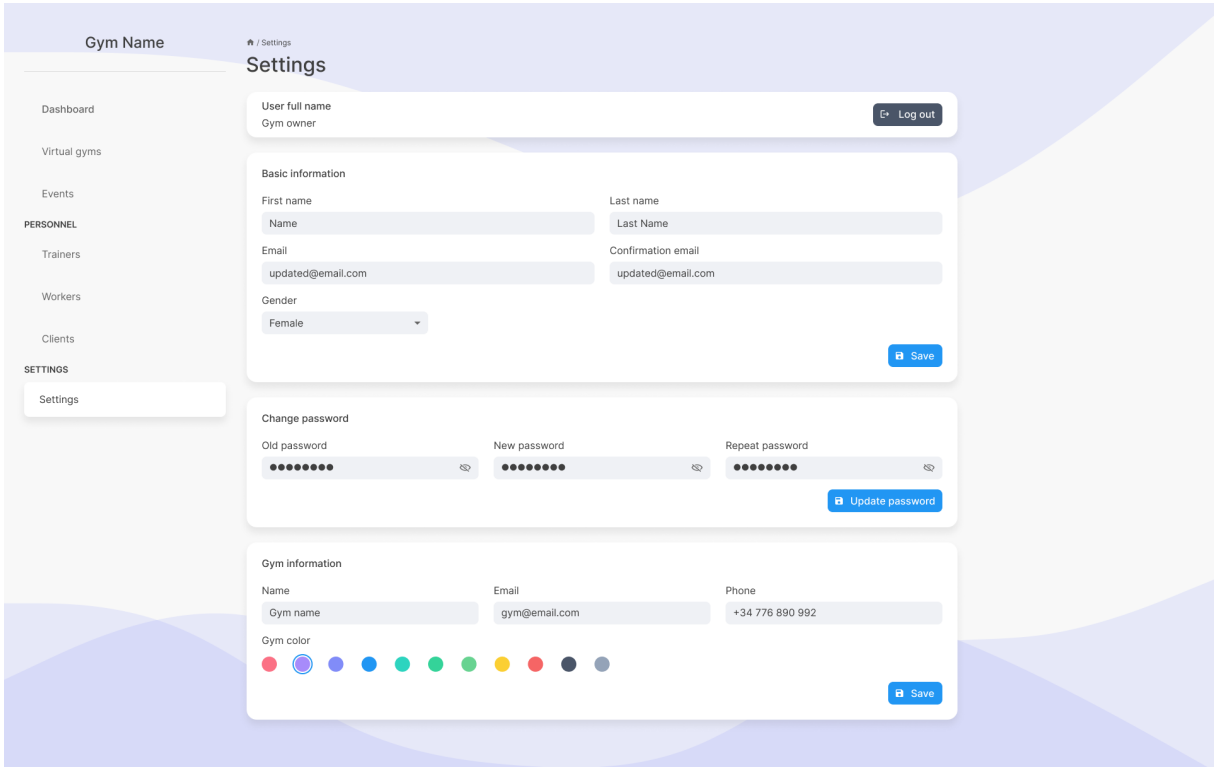

Figure 4.50: Settings page, from the owner's view

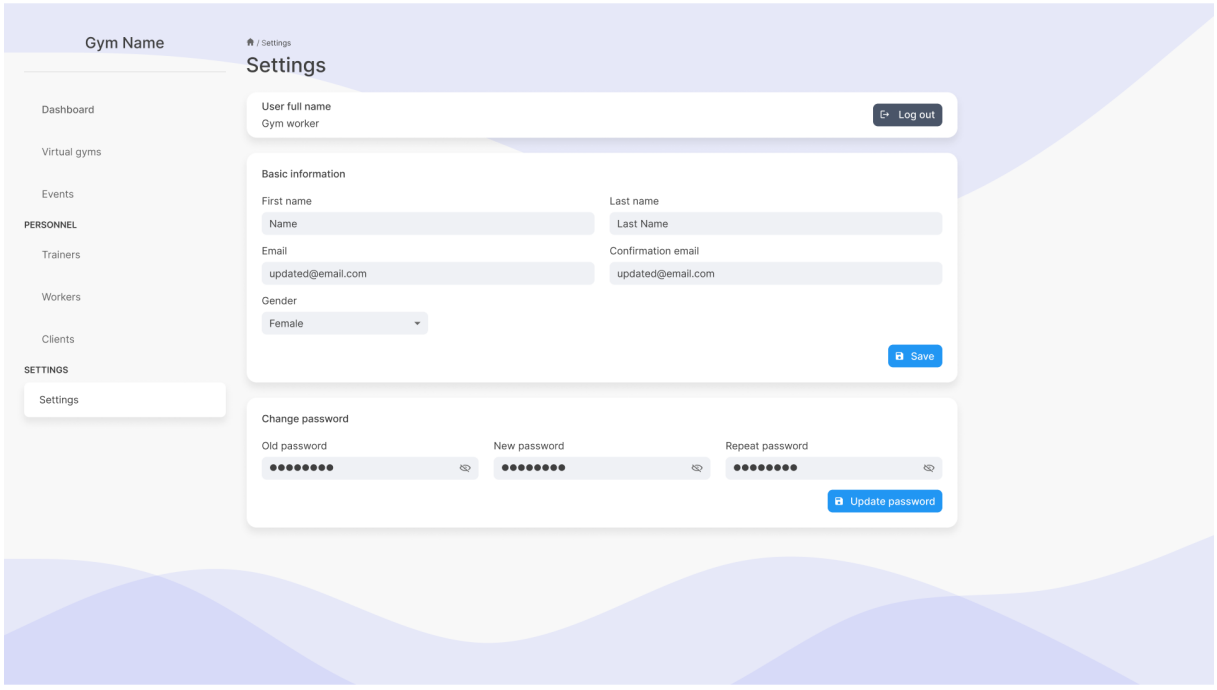

Figure 4.51: Settings page, from the worker's view

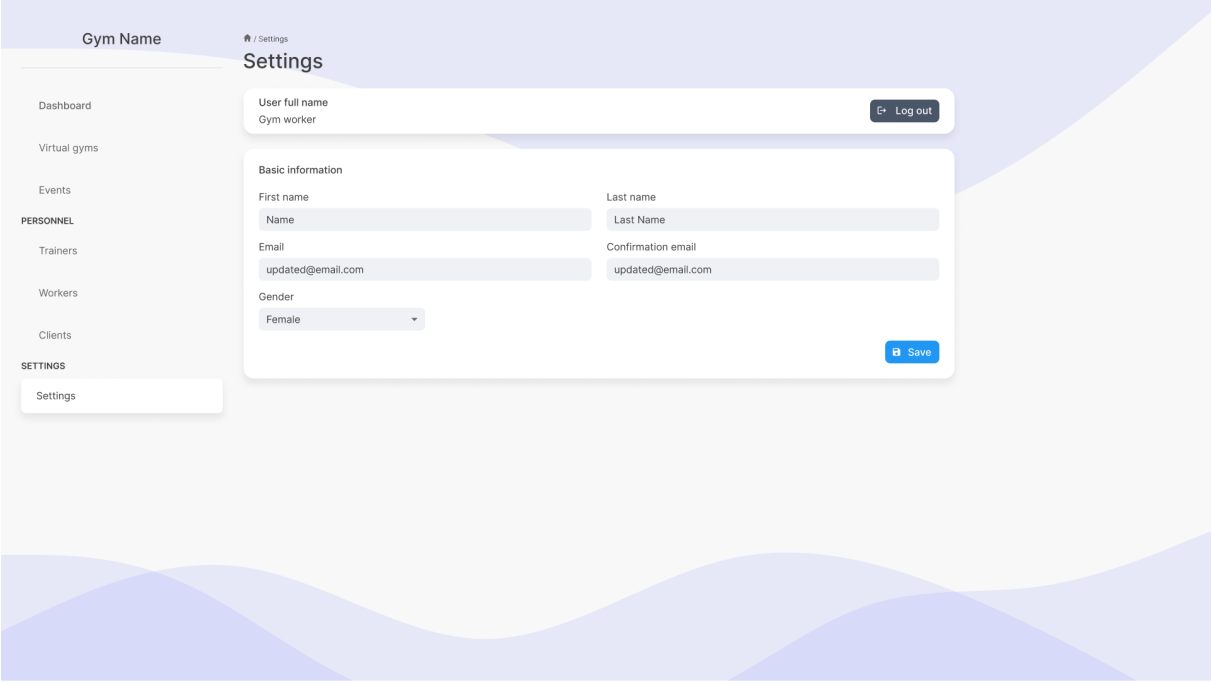

Figure 4.52: Settings page, from the client's view

4.4 Roadmap

Once that the tasks required to develop the project have been determined and specified, the time management can be easily structured, using a Gantt's diagram. Such diagram provides an estimated visualisation of the projecte development: visualising the start and finish of each part of the project.

Nonetheless, it is still an aproximation. Many things may happen during the development of the project, for instance, a web service that was not expected when plannig the project, yet now is necessary. The tasks that have not been planned, since were difficult to predict, may affect negatively the project schedule, distorting the roadmap. On the contrary, it could happen that some task was overestimated, or may not even be necessary. Such tasks will also affect the roadmap.

4.4.1 Gantt's diagram

Since the number of packages is very large, the diagram has been structure in their parent packages. Therefore, the roadmap only contains the workig packages groups that have been explained in the *Working packages* section. Furthermore, a brief explanation is required, as the diagram does not talk by itself.

First, the *Project management* package has a square in the roadmap everytime a part of the project is started. That is because the documentation has to be continually updated, with the new changes in order to keep consistency between the documentation and the project. Secondly, the *Analysis & Design* group has been divided in two parts. The first part involves the design of the database. This design is a bare initial implementation, as it may change while the project is being developed. The second part is the design of the user interfaces. An important amount of views will be designed, and it would speed up the process if the API

application is already implemented, since knowing what data the server will provide, the UI is easier to design. That is why the second part happens after the development of the API application. Third, at the same time the database design is started, the *Development env* group of tasks will take place. The reason behind this decision is due to the fact that the development environment and the database design are related. Fourth, the client side applications will be developed one after the other. The main group, the *Core application* will take most of the amount of time, since it is the most complex. Furthermore, one of the goals of the project is the creation of libraries which allow the reuse of code within the applications. Such abstraction will accelerate and simplify the development of the client and landing applications. Last but not least, there's one group that does not appear in the diagram. Such grup is the *Testing* group. The reason why the group has not been added to the roadmap is beacause the testing of the application it is something that has to be done while the application is being developed. As explained previously, the application has been developed using a TDD methodology, therefore, testing is mandatory.

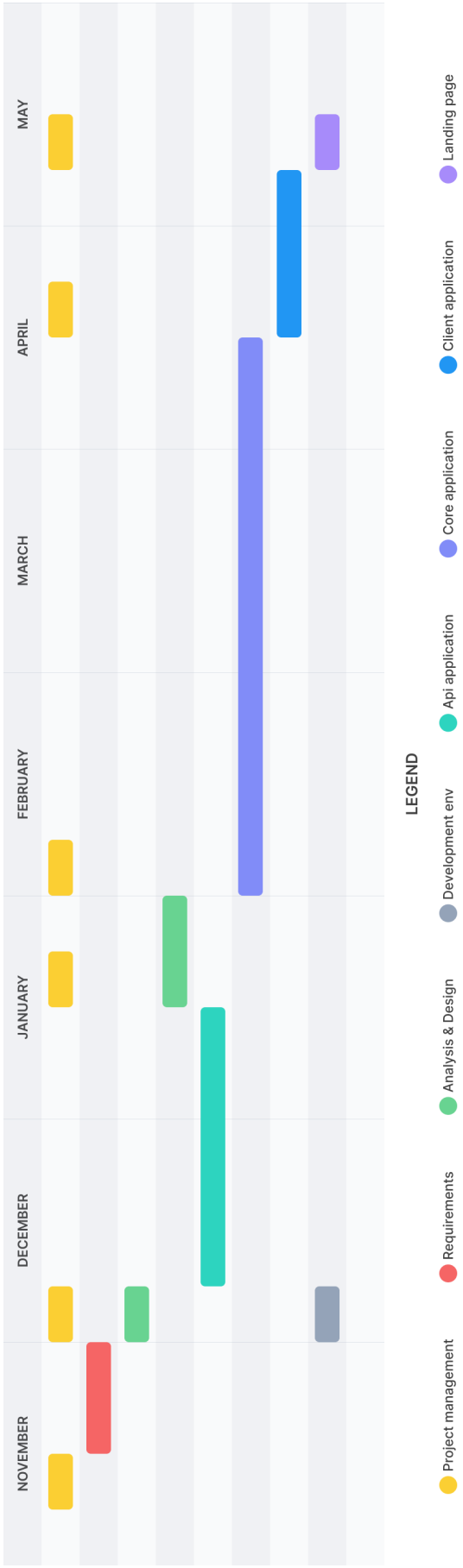

Figure 4.53: Chronogram of the development process