



## Surf Club Management Application

**Authors:** Bernardo Fragoso  
Gonçalo Albuquerque  
Miguel Sousa

**Advisors:** Filipe Freitas  
Miguel Pires, ESC

Project report carried out under the Project and Seminar  
Computer Science and Computer Engineering Bachelor's degree

May 2022

---

# INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

## Surf Club Management Application

47203 Bernardo Filipe Martins Fragoso

a47203@alunos.isel.pt

---

47265 Gonçalo Jorge Carvalheira de Albuquerque

a47265@alunos.isel.pt

---

47270 Miguel Gustavo Beirão de Oliveira e Sousa

a47270@alunos.isel.pt

---

Advisors: Filipe Freitas

ffreitas@cc.isel.ipl.pt

---

Miguel Pires, ESC

miguel.toscano.pires@gmail.com

---

Project report carried out under the Project and Seminar  
Computer Science and Computer Engineering Bachelor's degree

May 2022



# Abstract

Managing an association and its members can be a difficult effort, especially without the right tools, one of these examples is Ericeira Surf Club [1], that uses a simple excel sheet to manage their members.

This approach can be a handy option with a simple application, however as the association grows, it becomes a less feasible option due to lack of scalability. Besides making the maintenance of members a long task and inefficient, it also makes the implementation of new functionalities difficult.

In addition to replacing the excel sheet which only managed its members and their personal information, it is now feasible to acquire member engagement, thanks to the future availability of a system that will allow members to view their profile and receive club related notifications, such as event or payment reminders.

A management application enables the creation of a shared infrastructure for achieving compliance with business policies, resulting in improved results in terms of both goal execution and budget management.

One of the services that our project offers in addition to the application is a digital membership card, which allows a club member to obtain discounts at partner stores. Member verification is done through a QRcode present on the card, upon scanning the code the store representative is presented with a quick overview of the member's state.

The main aim of this project is to solve this problem creating a more modern environment, bringing more exposure to the company, and allowing a simple and easier management of their members.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Motivation . . . . .	2
1.2	Goals . . . . .	2
1.3	Specifications . . . . .	3
1.4	Report structure . . . . .	3
<b>2</b>	<b>Functionalities</b>	<b>4</b>
<b>3</b>	<b>Technologies</b>	<b>6</b>
<b>4</b>	<b>Architecture</b>	<b>7</b>
4.1	Backend . . . . .	8
4.2	Frontend . . . . .	8
<b>5</b>	<b>Data Model</b>	<b>9</b>
<b>6</b>	<b>Server implementation</b>	<b>11</b>
6.1	Web API . . . . .	12
6.1.1	Authentication . . . . .	12
6.1.2	Private data management . . . . .	12
6.1.3	Authorization . . . . .	13
6.2	Service . . . . .	13
6.3	DAL . . . . .	14
<b>7</b>	<b>Client implementation</b>	<b>15</b>
7.1	React . . . . .	15
7.1.1	Components . . . . .	15
7.2	Router . . . . .	16
7.3	API access . . . . .	16
7.4	Client side error handling . . . . .	17
7.5	Authentication and Authorization . . . . .	17
7.6	Code structure . . . . .	18

7.7	Internationalization . . . . .	19
<b>8</b>	<b>User interface and Functionalities</b>	<b>20</b>
8.1	Homepage . . . . .	20
8.2	Sign In and Sign Up . . . . .	21
8.3	Overview . . . . .	23
8.4	Members . . . . .	23
8.4.1	Profile . . . . .	23
8.4.2	All users and All companies . . . . .	25
8.5	Sports . . . . .	27
8.5.1	My Sports . . . . .	27
8.5.2	All Sports . . . . .	28
8.6	Quotas . . . . .	28
8.6.1	My Quotas . . . . .	28
8.6.2	All Quotas . . . . .	29
8.6.3	Management Quotas . . . . .	30
8.7	Events . . . . .	31
8.7.1	My Events . . . . .	31
8.7.2	All Events . . . . .	32
8.7.3	Event Page . . . . .	33
8.8	Candidates . . . . .	34
8.9	Responsive Design . . . . .	35
<b>9</b>	<b>Membership Card</b>	<b>37</b>
<b>10</b>	<b>Pagination and Filtering</b>	<b>39</b>
10.1	Pagination . . . . .	39
10.1.1	Client Side . . . . .	39
10.1.2	Server Side . . . . .	40
10.2	Filtering . . . . .	40
10.2.1	Client Side . . . . .	40
10.2.2	Server Side . . . . .	40
<b>11</b>	<b>Working Methodology</b>	<b>41</b>
11.1	Unit Tests . . . . .	41
11.2	Integration Tests . . . . .	41
<b>12</b>	<b>Conclusion</b>	<b>42</b>
<b>References</b>		<b>44</b>

# List of Figures

4.1	General Architecture . . . . .	7
5.1	EA model . . . . .	9
6.1	API structure . . . . .	11
6.2	Hashing passwords plus salt . . . . .	13
7.1	Component lifecycle [2] . . . . .	16
7.2	Redux scheme . . . . .	17
8.1	Homepage . . . . .	20
8.2	Sign In page . . . . .	21
8.3	Sign Up page . . . . .	22
8.4	Application message page . . . . .	22
8.5	Individual user profile page . . . . .	24
8.6	Corporate user profile page . . . . .	24
8.7	All users page . . . . .	25
8.8	User creation modal . . . . .	26
8.9	Member's sports page . . . . .	27
8.10	Sports page . . . . .	28
8.11	My Quotas Page . . . . .	29
8.12	All Quotas Page . . . . .	29
8.13	Quota Creation Modal . . . . .	30
8.14	Quota Management Page . . . . .	30
8.15	My Events Page . . . . .	31
8.16	My Events Page . . . . .	32
8.17	My Events Page . . . . .	33
8.18	Candidates page . . . . .	34
8.19	Profile in a laptop resolution . . . . .	35
8.20	Profile in a phone and tablet resolution . . . . .	36
9.1	Membership card example . . . . .	37

9.2	Member validation scheme . . . . .	38
9.3	Validation Page . . . . .	38

# **Chapter 1**

## **Introduction**

The following chapter introduces the motivation behind the project and also the objectives and specifications that are to be developed.

### **1.1 Motivation**

Managing an association and its members can be a difficult effort, especially without the right tools, one of these examples is Ericeira Surf Club, that uses an excel sheet to manage their members.

This approach is simple to get started with and may seem functional, however, as the associations starts to grow so will the excel sheet, making it harder to do the maintenance of members and also limiting the implementation of new functionalities.

Besides replacing the excel sheet which only managed its members and their personal information, there's now an interaction with the members, allowing them to view their profile and receive club related notifications, such as events or payment reminders.

### **1.2 Goals**

This projects objective is the development of a system that pretends to solve the problems listed in section 1.1 of this chapter, by building a management application with features that allow the management of the associations members, keeping track of the sports they practice, their quotas and also the events organized.

## **1.3 Specifications**

So that the project can reach the desired outcome, the definition of the minimum requirements was needed, thus becoming the following:

- Allow the creation of members(users and companies), quotas, events, sports and also club candidates. It is also needed to allow the update and deletion of the resources that are to be created.
- Associate athlete users with their sports.
- Create a digital membership card that permits companies to identify club members and offer discounts or promotions.
- Notify by email new events and candidates approval.
- Data analysis for statistic purposes.
- Contact administration through the application.

## **1.4 Report structure**

This report is divided into several chapters, that outline the technologies chosen for the project as well as the way its architecture is assembled.

The data model used and the server and clients implementation are also detailed to allow a better perception of the how the project was developed, along with documented features the project provides.

# Chapter 2

## Functionalities

The following chapter presents all functionalities that the application will be capable of executing. Visual examples can be found in chapter 8.

- **Signup:** Non-members who want to join the club can do so by filling out an application.
- **Login/Logout:** Members of the club are verified using the system's verification procedures.
- **CRUD**<sup>1</sup>:
  - **Candidates:** Ability to add, edit, view and delete candidates.
  - **Members:** Ability to add, edit, view and delete members.
  - **Sports:** Ability to add, edit, view and delete sports.
  - **Events:** Ability to add, edit, view and delete events.
  - **Attendance:** Ability to add, edit, view and delete attendances.
  - **Users Sports:** Ability to add, edit, view and delete the user sport association.
- **Create, update and read quotas:** Ability to add, edit and view quotas. New members will automatically have the current quota in debt.
- **Upload Club's current data to the system:** Ability to upload a file containing all the club's information to Surf Club Management Application, structuring the data according to the prerequisites.
- **Notification via email:** Besides communicating with administration through an email form in the application, members will get notification about events and quota payments. Candidates will receive an email if they were accepted.
- **Club statistics:** Ability to view all information about the current state of the club.

---

<sup>1</sup>CRUD: is an acronym to implement a persistent storage application: create, read, update and delete.

The system supports three types of roles: Administrator, Club member and Candidate.

- **Administrator:** Able to access the system's maximum capability and functionalities without restrictions;
- **Club member:** There are two types of club member, a User and a Company. A user is able to access all common information such as Events and Sports and also has the ability to view and edit their own personal information, as well as information related to sports, events or quotas. A Company can also access all common information and is able to view and edit their information as well as their attendance in Events.
- **Candidate:** Able to only access the homepage of the system and make his club application.

# Chapter 3

## Technologies

This chapter describes the technologies used in this application.

The server side was developed in NodeJs [3] and uses the node-postgres [4] module for database access as well as the Express framework [5] for HTTP [6] requests.

A PostgreSQL [7] relational database was utilized to store the data for this application. This Database Management System (DBMS) was chosen because of past expertise with it and the simplicity with which it can be hosted on the Heroku platform [8].

The client side was built with React [9], a framework that enables for the building of high-performance responsive apps. The MUI library [10] is also used in the client application, which makes it easier to develop a responsive and intuitive user interface.

To ensure well managed state of the frontend application the use of Redux Store [11] was required, since Redux simply provides a subscription mechanism which can be used by any other code. That said, it is most useful when combined with a declarative view implementation that can infer the UI updates from the state changes, such as React or one of the similar libraries available.

Heroku was chosen as the platform to launch the app since it has built-in support for PostgreSQL-based projects.

To have a visual example of where these technologies are being used, check this figure 4.1.

# Chapter 4

## Architecture

The architecture of the developed system is described in this chapter, with its components illustrated in figure 4.1.

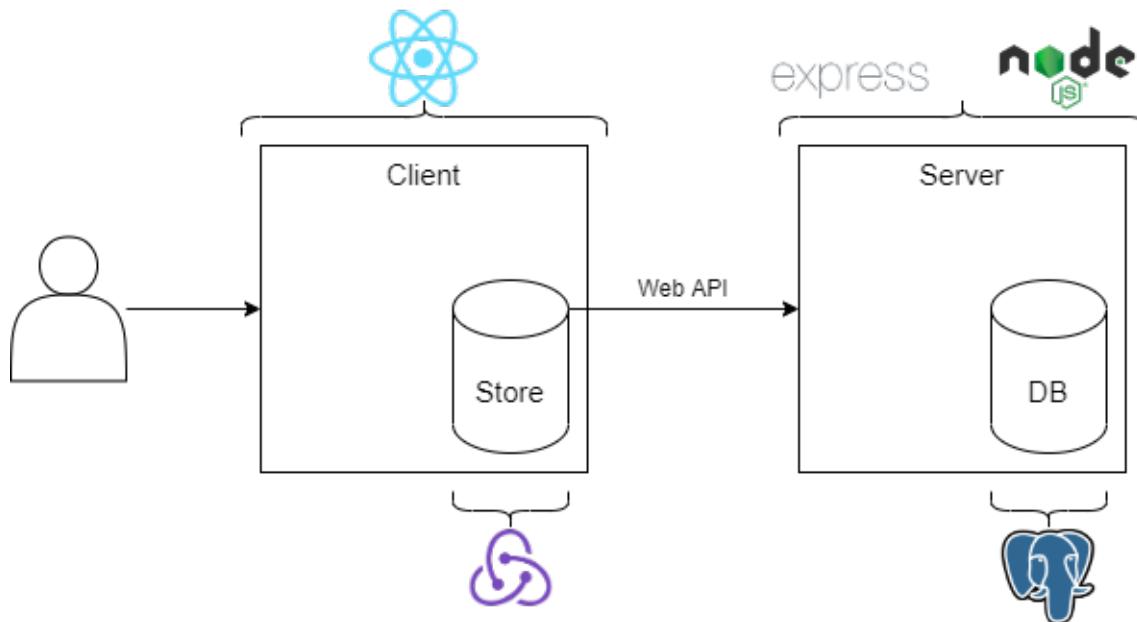


Figure 4.1: General Architecture

The system is divided into two applications. The server application is the major one, and it's in charge of implementing the system's logic, storing data, and providing an HTTP API that offers a set of capabilities to potential customers. The system's other application is a web application that serves as a client for the server application, allowing users to interact with it via a graphical interface.

This technique was adopted in order to accommodate the possibility of numerous types of user engagement, such as mobile or desktop applications. Clients do not need to maintain knowledge about the system or implement its logic in this way because these pieces are already available in the server application.

A web application was chosen for the creation of a client application because it is compatible with a simple browser, which is what most club members use. It still allows us with just one implementation to be compatible with any Operating System or device.

## 4.1 Backend

The server application is an application that exposes a web API. This API serves as the central point of contact for all clients, meaning that all clients (web, mobile, etc.) connect with the same server application.

Customers contact the different endpoints to access, change, or create the required resources, and here is also where all the system's logic is implemented.

The Javascript programming language [12] and the NodeJS execution environment were used to create this application. A PostgreSQL relational database was utilized to hold the data for this application.

This Database Management System was chosen because of past expertise with it and the simplicity with which it can be hosted on the Heroku platform.

## 4.2 Frontend

The web application consists of a frontend application that aims to design a User Interface (UI) to access the resources made available by the backend.

This web application is an SPA (Single Page Application) [13], which means it just has one page that changes its appearance numerous times over the application's lifespan.

A single-page app was chosen over a multi-page app because single-page apps are quicker since most of its resources are only loaded once. As mentioned briefly in the technologies chapter, the ReactJS library was utilized since it is now one of the most popular technologies for designing user interfaces.

To ensure well managed state of the frontend application the use of Redux Store was required, since Redux provides a subscription mechanism which can be used by any other code. That said, it is most useful when combined with a declarative view implementation that can infer the UI updates from the state changes, such as React or one of the similar libraries available.

The package module / web application consists of a front-end application that aims to design a User Interface (UI) to access the resources made available by the backend. All pages were designed using the Material-UI framework and on mobile devices to match design needs and industry standards.

# Chapter 5

## Data Model

The following chapter presents the databases graphical representation in its Entity Association [14] format, describing the existing entities and the relations between them.

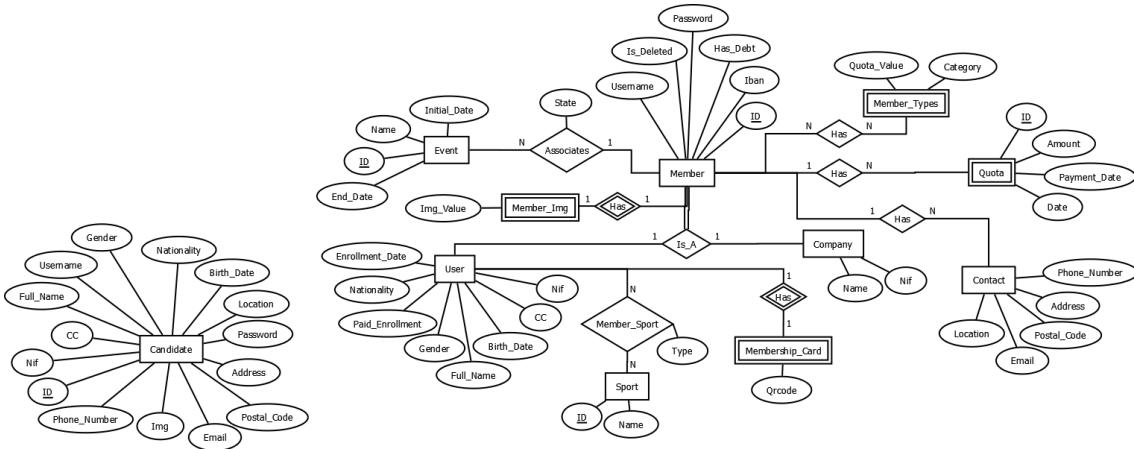


Figure 5.1: EA model

As shown in the figure 5.1, Member is the core entity, representing a member of the surf club, and this member is one of two types, a User or a Company. These Members also have associated a Member Type to help define what kind of role the member has within the organization.

The Quota entity allows to keep record of the organizations quotas and to identify which Members have paid their dues regularly.

As there are Events, which can represent a plethora of different affairs within the organization, there are also records of Attendance so that participating Members can be identified.

A Member, as it was referred before, can be a User or a Company, being that a Company is more restricted in terms of its uses since it is only a way to allow a company representative to access its own profile and validate Users by scanning their membership card. Contact saves the relevant information regarding the ways to contact a Member.

A User, is a broader type of Member representative of common members of the organization and preserves more relevant information in regards to their activities in the organization, it being related to which Sports they are associated with.

A Candidate is the state before it becomes a Member, saving all information needed for when the transitions happens.

# Chapter 6

## Server implementation

In this chapter it will be described, in greater detail, the server's implementation, each layer's mission and how they accomplish it.

The server is the block in charge of storing, maintaining, and providing all the information required by the client application. It manages all of this through three distinct layers:

- Web API, which is divided in two layers – Routes and Controllers:
  - Routes – establishes endpoints and authorization middleware;
  - Controllers - responsible for extracting the data from all requests;
- The Business, that contains all the logic of the API, is represented by one layer – Services;
- Data Access Layer, is responsible for accessing the database, verifying the existence of the resource and if it already exists – Data;

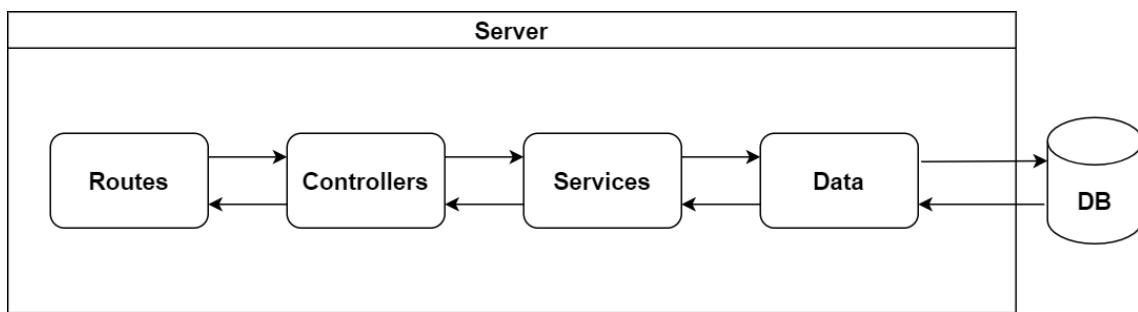


Figure 6.1: API structure

## 6.1 Web API

The interface layer will receive HTTP requests from the application's client side, process any arguments given, and then route them to the service layer.

When the service delivers the data that the client requested, it is the interface's responsibility to map this information into an HTTP response. Similarly, all errors that occur are mapped into an HTTP error response. Each module in charge of this logic was referred to as a controller.

Since the server uses the Express.js framework, the UI follows the tool's architecture. Express.js is a popular Node.js web framework. A router is used by the framework to implement the server's endpoints. In this application, the server module creates the Express' router, which is then provided to the module routes, where it registers all the paths.

### 6.1.1 Authentication

Authentication is done by using the passportJS [15] middleware, a NodeJs authentication middleware, which is based on a local strategy [16] with username and password and validates the authentication attempt by populating the session cookie [17].

This middleware encapsulates this functionality while delegating non-essential elements to the application, such as data access. This separation of responsibilities makes Passport incredibly straightforward to integrate, whether you're creating a new app or working on an existing one.

Authenticating requests is the responsibility of strategies, which they achieve by implementing an authentication mechanism. Authentication techniques provide how to encode a credential in a request, such as a password or a claim from an identity provider. They also lay out the steps for verifying that credential. The request is authenticated if the credential is properly checked.

### 6.1.2 Private data management

Storing clear passwords in a database is a serious mistake that jeopardizes the data's security and privacy.

To fix this problem, the BCrypt library [18] is used, which does a hash of the entered password, together with the salt [19], by the user. The purpose of using a salt in a password hash is to ensure that identical passwords do not have the same hash, as the same hash function applied to the same password produces the same output, making the hash predictable and vulnerable.

The salt is just a string that is generated in an asynchronous manner and appended to the password in clear text so that the hash may be performed afterwards.

The following figure 6.2 is an schematic example from an article of StackAbuse [20] :

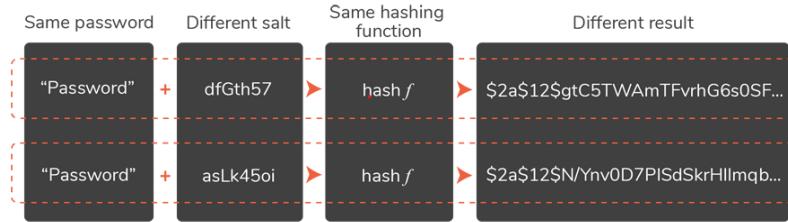


Figure 6.2: Hashing passwords plus salt

### 6.1.3 Authorization

Custom middlewares were created to handle authorization for each route of the API. These middlewares were necessary since administrators have access to all routes, members have access to common routes for all members as well as personal data routes, and anyone who is not a member has access to only the candidature and login route.

## 6.2 Service

The business layer is in charge of receiving the parameters of the request from the interface layer. The company must then guarantee that all business logic is followed. It will choose which DALs [21] to query in order to gather and send the information required to fulfill the client's request.

The service layer is responsible for translating that information into business known objects and finally passing them to the API layer after the DALs have returned the requested information from the database. The same holds true for any errors received from the data access layer, which must be mapped before being delivered to the interface layer.

A service was given to each module in charge of this logic.

### **6.3 DAL**

The data access layer is a layer that separates the database from the service layer. It must be done in such a way that the database and service layer are completely independent of one another. To do so, it queries the database and translates the answer into service-specific types. Its main goal is to offer methods for querying databases and returning results to the service layer.

Since querying the database can frequently lead to crucial syntax mistakes and even code repetition, the DAL makes use of a Schema module which all queries. Its main purpose is to increase the code's maintenance and provide a more comprehensive organization.

# Chapter 7

## Client implementation

The following chapter explains the implementation and thought process behind the client side, that is responsible for the visual element of the project.

### 7.1 React

React makes the creation of dynamic views easy due to its ability to send props to children components, which means that any modification to a prop triggers a re-render of the component that was modified. This effect is due to React using a virtual DOM, that compares the previous state of the components and updates only the ones affected.

#### 7.1.1 Components

A React component goes through different moments in its life cycle as shown in figure 7.1. The creation and render of the component enter the mounting process, in which the components constructor is called to define the needed props, afterwards the view designed by the user is rendered onto the DOM.

Unmounting is the process in which the component is removed from the DOM and destroyed and performs any sort of cleanup in regards the residual code or state.

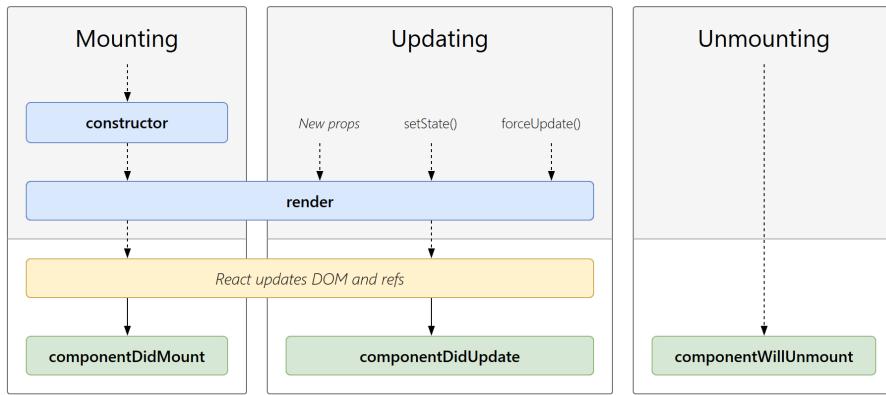


Figure 7.1: Component lifecycle [2]

## 7.2 Router

Since the client is a standalone application, disconnected from the web API, there's a necessity to create client side routing.

The definition of a route consists in associating a path with a component as well as defining a hierarchy and division.

This division is necessary to the different type of routes, specifically the dashboard views, login and candidature views, error views and home page views.

By defining groups of routes we can compose a specific layout that will be used for all children routes. This is especially useful to build the dashboard, and define that every children uses a sidebar for navigation and header. Besides defining the specific layout, it's also possible to do the authorization on the client side, which is done through the use of different components that check if there is an authenticated member or if the member has the necessary credentials.

## 7.3 API access

The web application serves as the web API client for our system, and as such there must be a connection between both components.

This bridge is maintained with the use of a redux store, that saves the global applicational state, in other words, the state of the API requests, in a way that allows each view to subscribe to the state and directly observe the state without the need of tools like `useState` to alter the step in which the request is at.

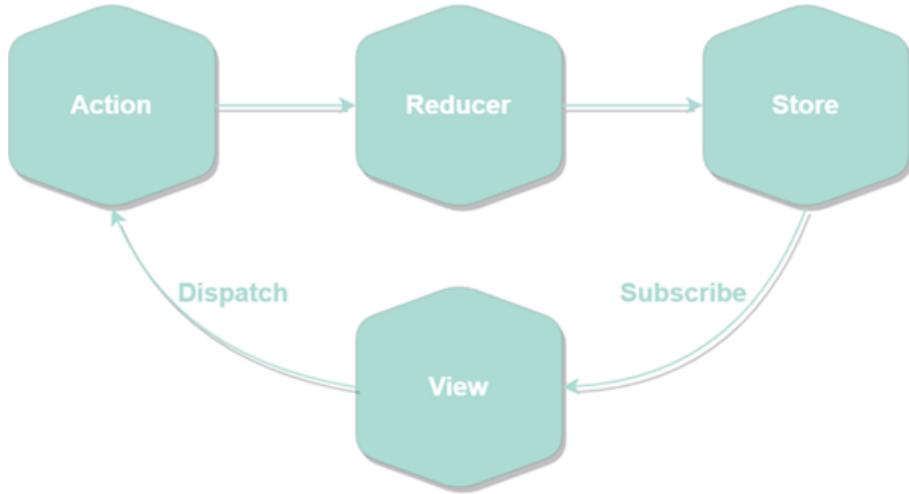


Figure 7.2: Redux scheme

As it was said before, the view subscribes to the chunk of state it will need, and this state may vary, however there are always 3 components: a loading, error, and value variables.

Afterwards, the view can dispatch an action, that in our case corresponds to an API request, also considered to be a process describer.

The action also executes several dispatches to its correspondent reducer, that based on the values received by the action, it updates the values of the state. These dispatches that the action does are referent to a request's lifecycle, in which it enters a loading state, and after receiving the response of the API it passes onto the state of success or failure depending on the response.

All these changes directly affect the centralized store, which in turn also affects the view that subscribed to the state.

## 7.4 Client side error handling

As it was mentioned above, an API request can produce an error, and this error is saved on the Redux store.

The view, since it subscribed to the state was notified of this change in value and renders a new component that describes the error to the user.

## 7.5 Authentication and Authorization

The authentication in the client also resorts to the use of Redux, in which if the credentials do match an existing account, then the necessary info about the member is saved in the session storage of the browser. This is due to the technique used in the login process, which is based in session authentication, saving cookies to validate the member.

The authorization merely prevents the client from doing requests to the web API in which it does not have access, this is done in two steps:

- The pages themselves do not appear in the user interface if the authenticated member does not have the necessary credentials;
- If the member manages to access these restricted pages, for example by entering the URL directly, they are redirected to an unauthorized page;

## 7.6 Code structure

The frontend folder contains everything for the client side and has as sub-folders:

- **Assets**, which is divided into:
  - **Data** - Contains files that are render in some pages like the application logo and the background video of the home page;
  - **Scss** - Contains the css [22] of some pages;
- **Components** - Different react components
- **Hooks** - The different hooks [23] used on the application, Hooks are functions that allow the use of state and other React features without writing a class.
- **Layout** - Layouts used in various pages, like the sidebar on the dashboard or the HomeHeader for the HomePage.
- **Locales** - Contains the different strings for both Portuguese and English used on the Internationalization
- **Menuitems** - Contains the definition of the different pages
- **Pages** - All pages of the client application
- **Routes** - The different routes of the client application
- **Store**, which is divided into:
  - **Actions** - Actions are plain JavaScript, you can think of an action as an event that describes something that happened in the application;
  - **Constants** - Type of the actions and reducers are being used at two different files. Constants help to import them and use them from a single page.;
  - **Reducers** - Reducers are functions that take the current state and an action as arguments and return a new state result.
- Themes - contains the different fonts and colors used throughout the pages

## 7.7 Internationalization

The initial idea was for the application to be entirely in English, although the staff from Ericeira Surf Club asked if it could be done in Portuguese, so the idea of internationalization came so that we could have both English and Portuguese.

For internationalization, the react-i18next [24] library was used. It's a popular internationalization library which uses components to render or re-render the translated content of our application once users request a change of language.

In the application we have a button that allows the user to choose only between Portuguese and English, however, adding a third language would be effortless due to its simplicity in implementation, consisting in just creating the strings with the value in the new language.

# Chapter 8

## User interface and Functionalities

The application's graphical user interface (UI) will be presented in this part. The responsive views below were created with the goal of delivering information with as few clicks as feasible.

### 8.1 Homepage

The application's Homepage is represented by the figure 8.1. This is the first page that a user sees when they open the application.

At this time, the user is encouraged to enter the application if he or she is already a member of the club, or to register/candidate if they choose to proceed with their club membership application.

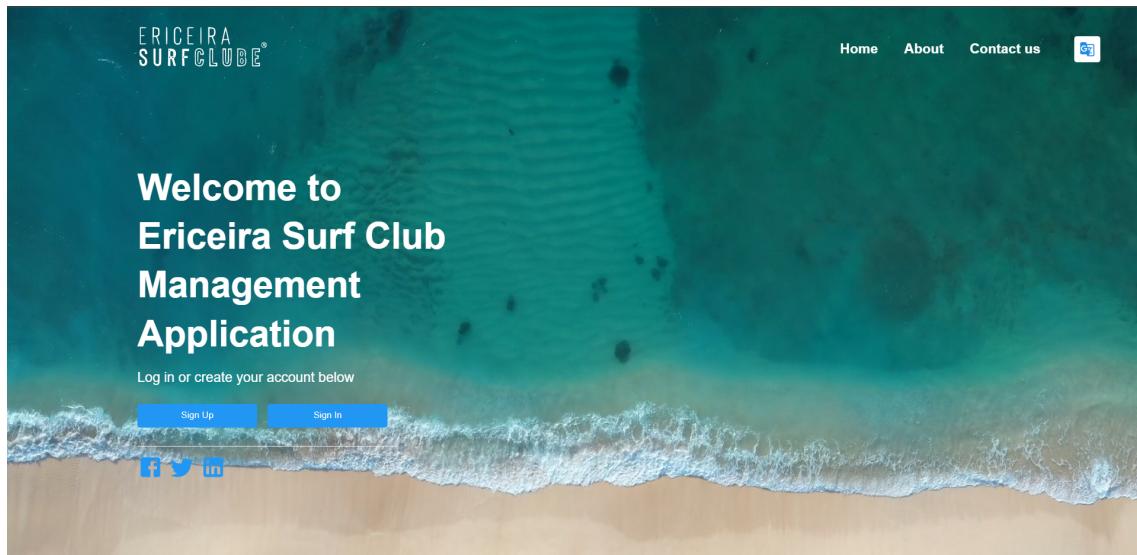


Figure 8.1: Homepage

It is important to note that the language displayed on this page, as well as all others in the application, may be changed.

It is also possible to learn more about the application in the About section, and users have the ability to communicate with the club's administration by email by filling out a form on the Contact Us page.

## 8.2 Sign In and Sign Up

It is possible to check the aspect of the page for a user to login in the following figure 8.2. It is also possible to be redirected to the application page if you do not yet belong to the club, or to the password recovery page.

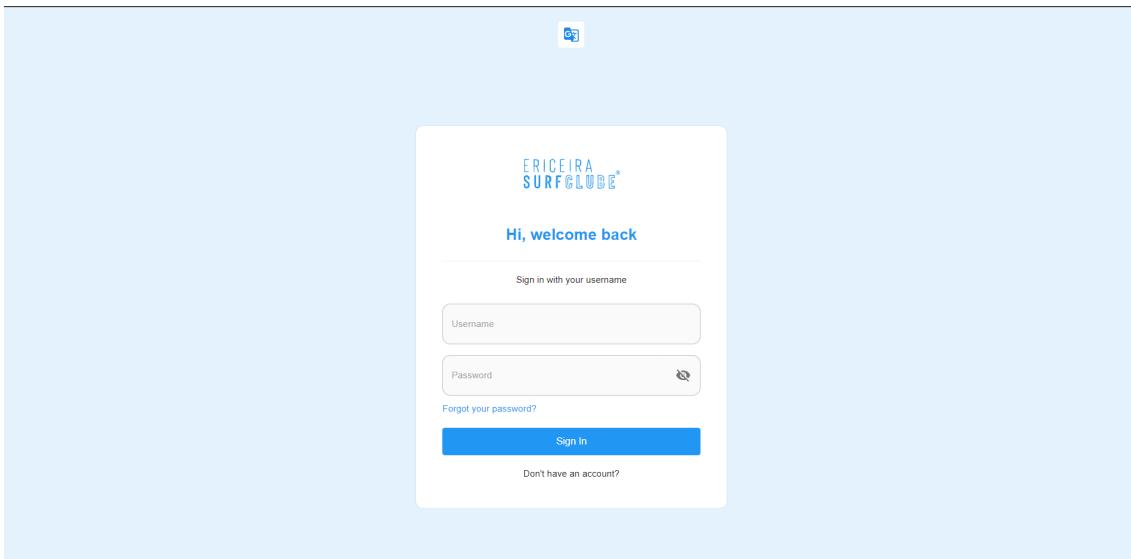


Figure 8.2: Sign In page

For the application procedure, a more comprehensive form must be filled out, including all pertinent information, so that an administrator may determine whether or not the candidate is eligible to join the club. This formula is shown in the following figure 8.3:

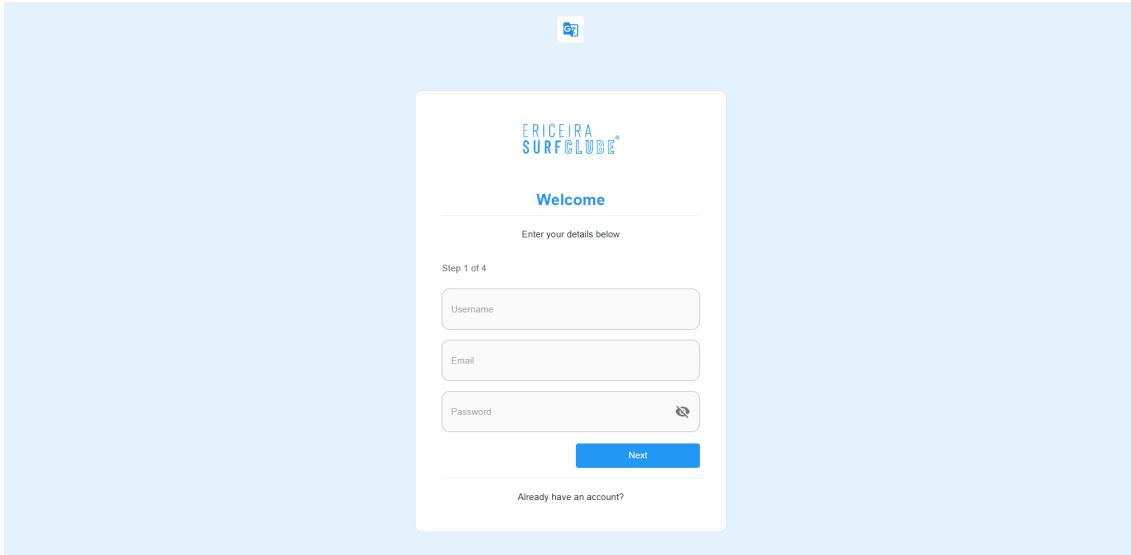


Figure 8.3: Sign Up page

Finally, once the user has completed all of the required fields, with the exception of the photograph, an email is sent to notify the user that if he or she is accepted into the club, they will receive an email notification.

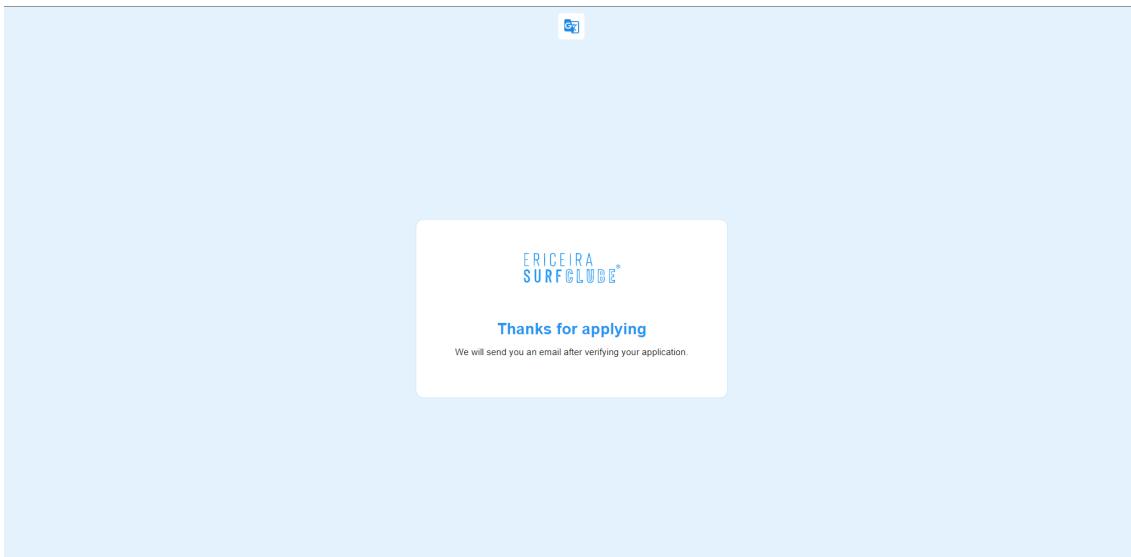


Figure 8.4: Application message page

## 8.3 Overview

This page's sole purpose is to give the user, in a glance, the ability to see a few statistics and some important information for either the administrators or the common users.

Apart from what is available on the overview page, it is also possible to access the application's other pages, such as those dedicated to members, sports, quotas, events, and, finally, candidates.

The overview page is shown in the above figure, along with the NavBar on the side, which has all of the application's options because the user who completed the login is the administrator.

However, if the person who logs in is not an administrator, their dashboard overview will have different contents, and the NavBar on the side will not provide access to pages that include general information about the application or information about other users. Continue with the demonstration in Figure [ref]:

## 8.4 Members

This section is going to describe how the data about the club members is displayed on the user interface and how it is managed.

### 8.4.1 Profile

The visualization of the member's profile page is shown in this section.

This page has two types of views, which are:

- individual user profile visualization;
- visualization of a user's profile that represents a company.

However, only the administrator has access to all of the members' profiles, and the rest of the members can only see their own. Aside from that, the only people who may make changes to the data in a given profile are the person who created it and the administrator.

All non-administrative members have the Tab of 'admin privileges' deactivated.

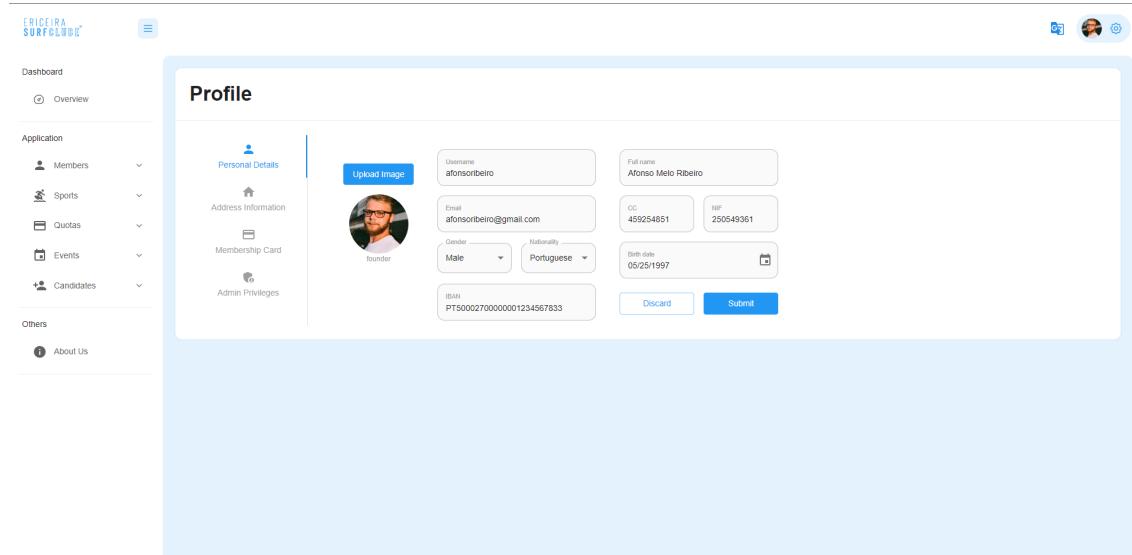


Figure 8.5: Individual user profile page

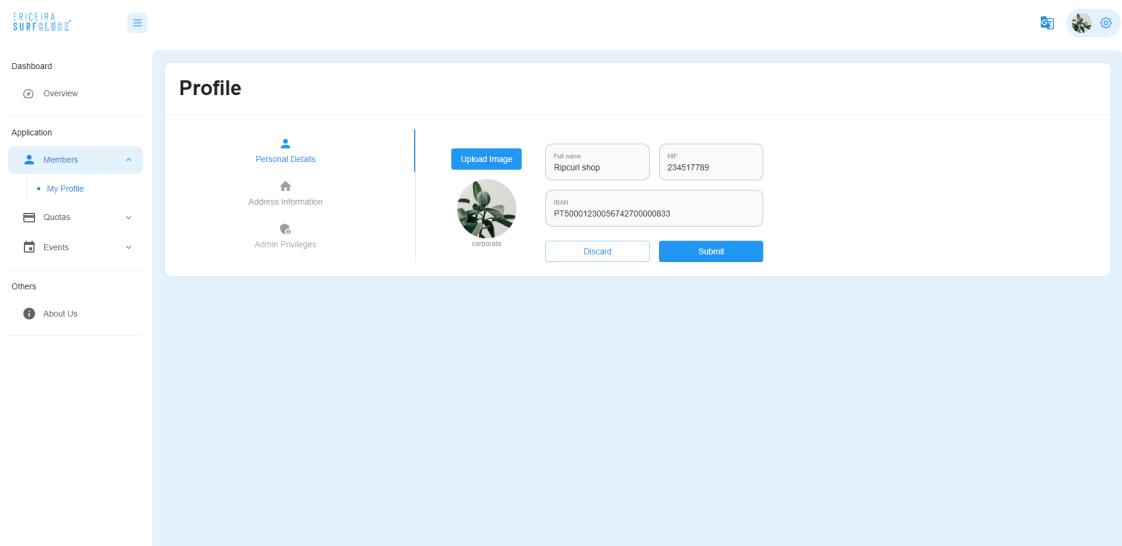


Figure 8.6: Corporate user profile page

As seen in the two figures above, there are differences between the two profile pages when the member is an organization or a person, particularly the Tab 'membership card,' because the corporate member does not have a membership card.

## 8.4.2 All users and All companies

The visualization page for all individual members allows you to see a user's most important data in a table format with the rest of the club's users. This page can only be accessed by an administrator.

It is possible to be redirected from this table to a user's personal profile, as well as to delete it. However, this deletion is subject to the soft delete condition, which means that the data remains in the database but with just a flag indicating that the user has been removed.

Given that visualizing tables might be difficult due to the large number of data points, the ability for users to filter by username, full name, and email has been included. This topic is covered in further depth in the pagination and filtering chapter 10.

ID	Username	Full name	Email	IBAN	Gender	Nationality	Birth date	Member type	Has debt?	Paid enrollment
2	jolopes	José Elias Lopes	jolopes@gmail.com	PT50001234270000000567833	Male	Portuguese	1989-04-27	effective	✓	✓
1	afonsoribeiro	Afonso Melo Ribeiro	afonsoribeiro@gmail.com	PT50002700000001234567833	Male	Portuguese	1997-05-25	founder	✗	✓
5	Silva	João Silva	Silva@gmail.com	PT50002700000001234567832	Male	Portuguese	2001-10-19	effective	✓	✓
6	Jorge	Américo Jorge	Jorge@gmail.com	PT50002700000001234567834	Male	Portuguese	2000-02-17	effective	✓	✗

Figure 8.7: All users page

Aside from data visualization, this page allows you to create a user by displaying a Modal [ref], figure 8.8, which is similar to a candidate application form. This feature is crucial for an administrator's manual addition if the member does not need to go through the application process.

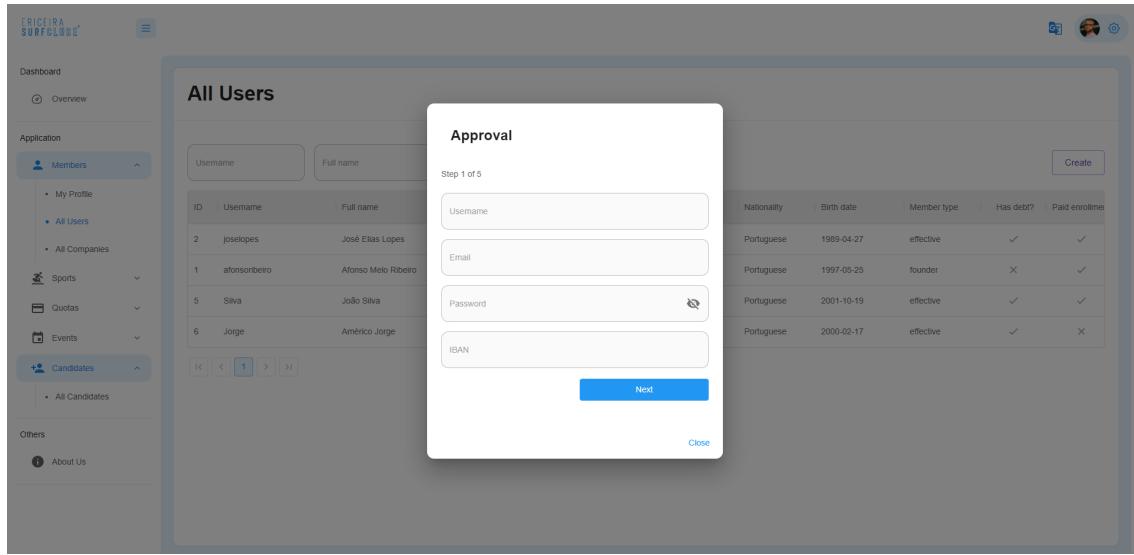


Figure 8.8: User creation modal

The page of all companies are identical to all users, with the exception of the tables' content and the creation form that has different to fill in.

## 8.5 Sports

This section is going to describe how the data about the sports of the club is displayed on the user interface and how it is managed.

### 8.5.1 My Sports

On this page, you may find information on various sports that the user is in, in a table format, along with the user's kind of position in relation to the sport, his federation number, the federation's number and name, the years he was federated, and if he still practices.

This information is shown in the following figure 8.9:

The screenshot shows the 'My Sports' page of the Freguesia Surfclub application. The left sidebar has sections for Dashboard, Overview, Application (Members, Sports, Quotas, Events, Candidates), and Others (About Us). The 'Sports' section is selected. The main area displays a table titled 'My Sports' with the following data:

Name	Type	Federation Number	Federation ID	Federation Name	Years Federated	Absent	Actions
Surfing	aprendice	4891	54	Federacao Portuguesa de Surf	2021,2022	X	
Kneeboarding	coach	4891	52	Federacao Portuguesa de Kneeboard	2019,2021,2022	X	
Bodysurfing	pratcioner	4891	51	Federacao Portuguesa de Bodysurf	2021,2022	X	
Wakeboard	aprendice	4891	50	Federacao Portuguesa de Wakeboard	2011,2012	X	
Finswimming	aprendice	4891	55	Federacao Portuguesa de Finswimm	2012,2013	X	

Pagination controls at the bottom show pages 1, 2, 3, >, and >>.

Figure 8.9: Member's sports page

## 8.5.2 All Sports

On the page dedicated to each sport, you may see all of the sports that the club offers for practice, which are shown using Cards[mui].

These Cards provide information on the total number of participants in the sport, the ability to remove them (soft delete) and be notified if they are removed, and, finally, the ability to be redirected to a specific page for the sport, as shown in Figure 8.10.

The screenshot shows a user interface for managing sports. On the left is a sidebar with sections for Dashboard, Application (Members, Spots, Quotes, Events, Candidates), and Others (About Us). The main area is titled 'All Sports' and displays a grid of cards for different sports. Each card includes a sport icon, the sport name, a 'Is deleted?' checkbox, the number of practitioners (all set to 2), and 'View Sport' and trash bin icons. Below the grid is a 'Create new sport' form with a 'Name' input field and a 'Submit' button.

Sport	Is deleted?	Number of practitioners	Action
Surfing	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Kneeling	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Bodysurfing	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Wakeboard	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Finswimming	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Windsurf	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Rafting	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Kiteboarding	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Paddleboarding	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Canyoning	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Kayak Polo	<input type="radio"/>	2	<a href="#">View Sport</a> <span style="color: red;">Delete</span>
Bodyboarding	<input type="radio"/>	0	<a href="#">View Sport</a> <span style="color: red;">Delete</span>

Figure 8.10: Sports page

## 8.6 Quotas

This section is going to describe how the data about the quotas of the members of the club are displayed on the user interface and how it is managed.

### 8.6.1 My Quotas

On this page the user will find information about his own quotas, namely the quota date and the payment date, which has a date if the quota was already paid or is empty if the quota is still for paying.

The screenshot shows a web application interface titled 'My Quotas'. On the left, there is a sidebar with a 'Dashboard' section and a 'Quotas' section under 'Application'. The 'Quotas' section has a 'My Quotas' option selected. The main content area is titled 'My Quotas' and contains a table with two columns: 'Date' and 'Payment Date'. The table lists five rows of data:

Date	Payment Date
2022-01-01	
2021-01-01	2021-06-22
2020-01-01	
2019-01-01	2019-02-23
2018-01-01	2018-02-23

At the bottom of the table, there is a navigation bar with icons for back, forward, and search.

Figure 8.11: My Quotas Page

### 8.6.2 All Quotas

This page is only accessible for admins, on this page the admins will have information about all the quotas of the club.

Each row represents a quota of an user and shows the username, email, phone number of the user as well as the date and payment date of the quota it also has a button so that the admin can signal that the quota was paid.

The screenshot shows a web application interface titled 'Quotas'. On the left, there is a sidebar with a 'Dashboard' section and a 'Quotas' section under 'Application'. The 'Quotas' section has a 'My Quotas' and 'All Quotas' option. The main content area is titled 'Quotas' and contains a table with several columns: Member ID, Username, Email, IBAN, Phone number, Date, and Payment Date. There is also a 'Create' button. The table lists multiple rows of data, each corresponding to a user quota. At the bottom of the table, there is a navigation bar with icons for back, forward, and search.

Figure 8.12: All Quotas Page

Aside from data visualization, this page allows you to create a new quota by displaying a Modal [ref], 8.13

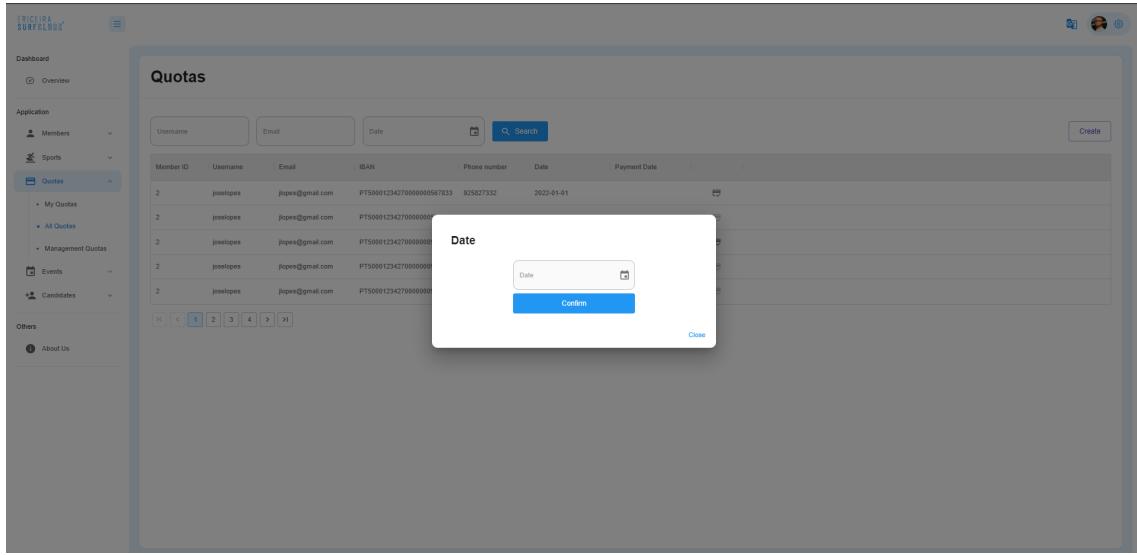


Figure 8.13: Quota Creation Modal

### 8.6.3 Management Quotas

This page will only be available for admins, and has the purpose of creating a new member type with a new quota value, and visualizing the existing ones.

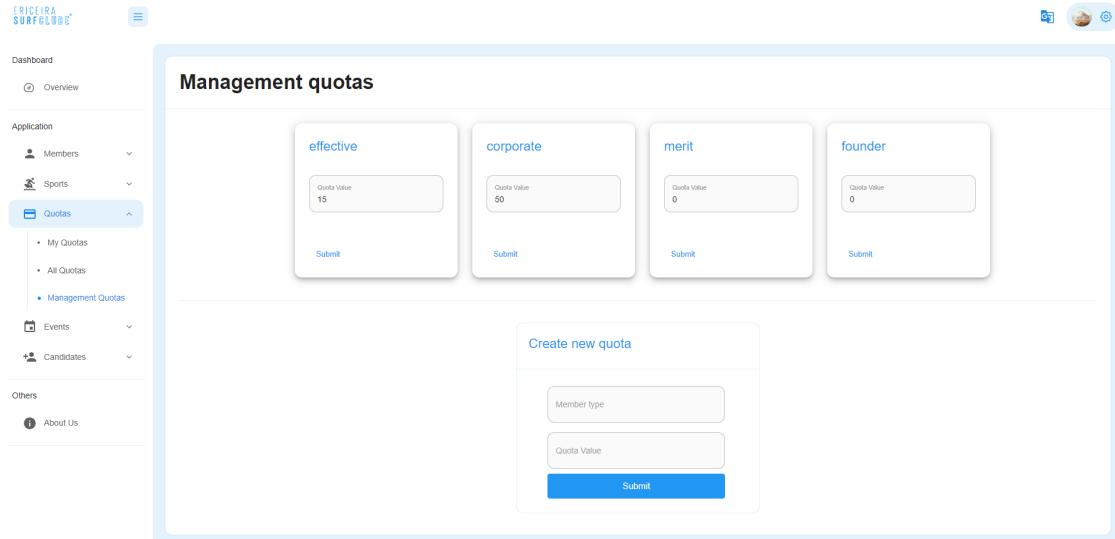


Figure 8.14: Quota Management Page

## 8.7 Events

This section is going to describe how the data about the events of the club are displayed on the user interface and how it is managed.

### 8.7.1 My Events

On this page the user can see the different events that interacted with, showing the state, that can be going, not going or interested , the initial date and end date as well as the name of the event.

ID	Name	State	Initial Date	End Date
1	Assembleia Geral	going	2022-04-09	2022-04-09
2	Assembleia Particular	interested	2022-04-11	2022-04-12
3	Assembleia	not going	2022-04-13	2022-04-14
4	Reunião do conselho	not going	2022-04-15	2022-04-15
5	Almopradaria Geral	going	2022-04-21	2022-04-22

Figure 8.15: My Events Page

## 8.7.2 All Events

This page is accessible for all the users and show all the events of the club, the ones that already happened , happening or even the ones that didn't started yet. Each event has a recycle bin icon so that a admin can delete the event.

ID	Name	Initial Date	End Date	Status
1	Assembleia Geral	2022-04-09	2022-04-09	ended already
2	Assembleia Particular	2022-04-11	2022-04-12	ended already
3	Assembleia	2022-04-13	2022-04-14	ended already
4	Reunião do conselho	2022-04-15	2022-04-15	ended already
5	Almopraga Geral	2022-04-21	2022-04-22	ended already

Figure 8.16: My Events Page

Both my event and all event pages have an icon to access an event page with a bit more information about the event.

### 8.7.3 Event Page

This page shows all the information about a specific event, end and start date, a list with all the users that reacted with the event and a count of people going, not going and interested

The screenshot shows the 'Assembleia Particular' event page within the Surfclub application. The left sidebar contains navigation links for Dashboard, Overview, Application (Members, Sports, Quotas, Events, Candidates), and Others (About Us). The 'Events' link under Application is currently selected. The main content area displays the event details: 'Initial Date: 2022-04-11' and 'End Date: 2022-04-12'. Below this is a table titled 'List' showing two entries:

Member ID	Name	Email	Phone number	state
1	afonsoribeiro	afonsoribeiro@gmail.com	962681730	interested
2	jlopes	jlopes@gmail.com	925627332	going

Below the table, there are three counts: 'People going: 1', 'People not going: 0', and 'People interested: 1'. The page header includes the Surfclub logo and a user profile icon.

Figure 8.17: My Events Page

## 8.8 Candidates

To sum up what the user interface contains, the page that corresponds to all candidates is a table that contains all of the information about them, information that is also available in the from referred to in the application process at point 8.2. The previous-mentioned page can only be accessed by users who are administrators.

This information may be verified using the following figure 8.18:

The screenshot shows the 'All Candidates' page within the 'Candidates' section of the application. The left sidebar includes sections for Dashboard, Overview, Application (Members, Sports, Quotas, Events), Candidates (All Candidates), and Others (About Us). The main content area is titled 'All Candidates' and features a table with 7 rows of data. The table columns are: ID, Username, Full name, Email, IBAN, Phone number, Gender, Birth date, Location, and Address. The data in the table is as follows:

ID	Username	Full name	Email	IBAN	Phone number	Gender	Birth date	Location	Address
1	miguefl	Miguel da Silva Lopes	miguefl@gmail.com	PT50002700000001234567831	967021559	Male	2000-05-19	Lisboa	Rua do Ouro n45
3	Bolha	Roberto Bolha	Bolha@gmail.com	PT50002700000001234567833	922221559	Other	1999-05-19	Porto	Rua da madeira n15
4	Dias	Amilcar Dias	Dias@gmail.com	PT50002700000001234567835	911121559	Other	2000-09-19	Braga	Rua do Ouro n45
6	Roger	Roger Dias	Roger@gmail.com	PT50002700000001234567836	911144559	Other	1999-01-11	Santarém	Rua da santa n45
7	Pedro	Tiago Pedro	Pedro@gmail.com	PT50002700000001234567838	911121999	Male	1987-05-27	Portimão	Rua da mão n11

Figure 8.18: Candidates page

## 8.9 Responsive Design

The necessity for the application to be responsive was one of the major criteria mentioned at the outset of the project.

Responsive web design is a technique for making online pages seem well on a wide range of devices and window or screen sizes. The Material UI library was used by the team to achieve this responsiveness, in the online application, several instances of this method are shown below.

As can be seen in the figures 8.19 and 8.20 shown below, the elements are arranged differently depending on the screen resolution, furthermore as can be seen by the Tabs [25] component of MUI, their orientation changes to suit better. to the screen where it is inserted.

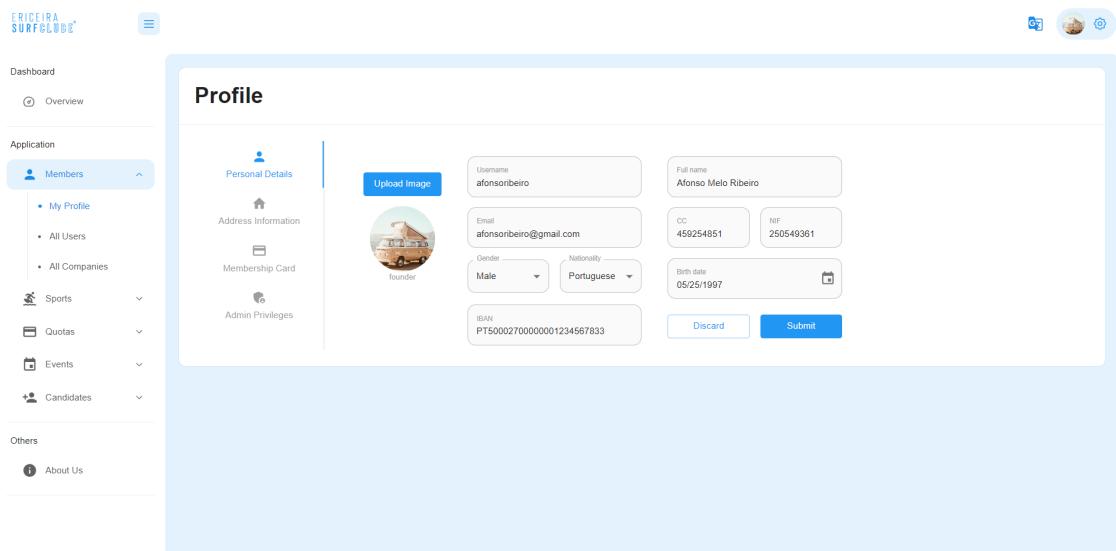


Figure 8.19: Profile in a laptop resolution

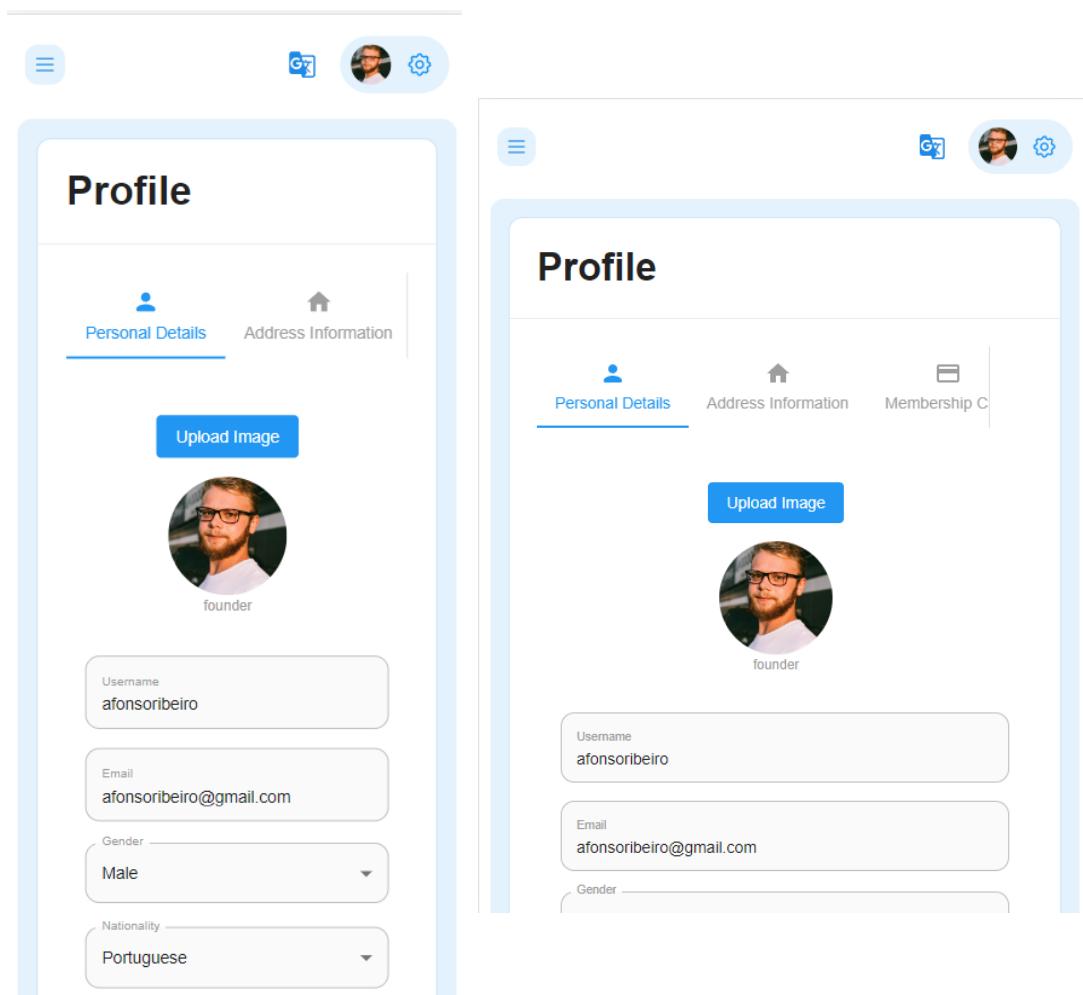


Figure 8.20: Profile in a phone and tablet resolution

# Chapter 9

## Membership Card

Ericeira surf club offers its members the possibility of discounts at partner stores, so it was important to have a way to identify its members, so the idea of a membership card came up.

For it to be sustainable and environmentally friendly, it was decided that it would be digital, the card will contain the member's name, role, club enrollment date and a photo of the same, it will also contain a QRCode so that partner companies can validate the member's information.



Figure 9.1: Membership card example

Ericeira surf club's partner companies will scan the QRCode present on the card and will be redirected to a page where they will have to log in and where they will be able to validate if the member in question is entitled to discounts.

Internally this verification is done by observing if the member has paid all quotas and the initial enrollment.

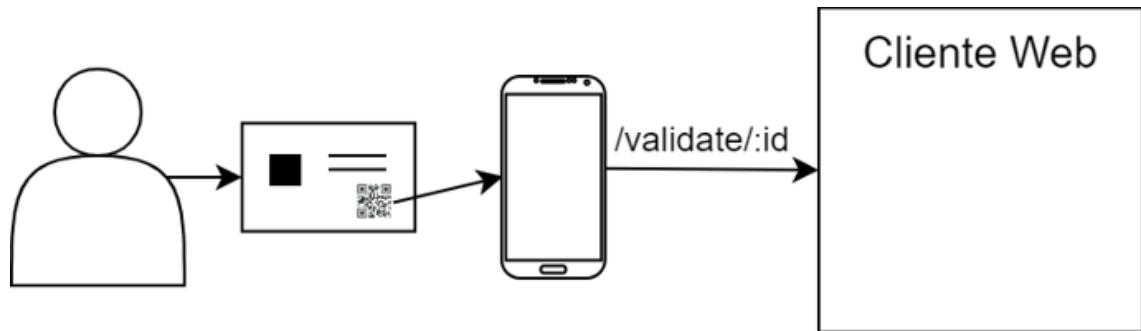


Figure 9.2: Member validation scheme

The next figure illustrate the page that a company will see after scanning the QRCode and login in if the member has is quotas paid.



Figure 9.3: Validation Page

# Chapter 10

## Pagination and Filtering

### 10.1 Pagination

Pagination is a mechanism for dividing material across multiple pages and showing a new piece of content each time the page number is modified. Pagination not only makes the user interface more intuitive, but it also saves time over the alternative strategy of fetching all the resources in the database.

This technique is employed across all pages that allow the listing of different items, such as listing all users, companies, candidates, quotas, sports, and sports related with the user.

The sections that follow go through how the technique was implemented on both sides of the application in detail.

#### 10.1.1 Client Side

The client side handles the state of the page number using the react hook useState to save the number of the page and the Material UI component Pagination to allow changing pages by altering the state.

This alteration provokes a trigger in the hook useEffect, that dispatches a new request to redux, making a new request to the web API sending the limit of rows that need to be fetched and the offset.

By consequence of the request, the page, that subscribed to the state that redux holds, is being modified as the state changes, and when that process is finished, the data has now been loaded into the state and is now rendered in the page.

### **10.1.2 Server Side**

The client side handles the state of the page number using the react hook useState to save the number of the page and the Material UI component Pagination to allow changing pages by altering the state.

This alteration provokes a trigger in the hook useEffect, that dispatches a new request to redux, making a new request to the web API sending the limit of rows that need to be fetched and the offset.

By consequence of the request, the page, that subscribed to the state that redux holds, is being modified as the state changes, and when that process is finished, the data has now been loaded into the state and is now rendered in the page.

## **10.2 Filtering**

Alongside pagination, filtration also helps reduce the load on the network by supplying the server with information that can outline and restrict which rows should be fetched, giving a more pertinent result based on what the application user needs to observe.

### **10.2.1 Client Side**

On the client side, state must be maintained regarding the filtering criteria, since every new request to the API needs these parameters.

This technique consists in a form, that holds different types of information relevant for filtering, and upon submission a new request is dispatched, and much like pagination, the hook useEffect is triggered, although this time the page is reset to its original value, it being 1.

### **10.2.2 Server Side**

Each endpoint that previously supported pagination can now also be supplied with query parameters relevant to each endpoint. These new parameters are used at the end of the pipeline, the data access layer, to build complex queries based on the existence of filtering parameters.

Since the filtering works alongside pagination, the previous aspect mentioned regarding pagination also apply to filtering, meaning the total number of rows that fit the criteria is also returned to facilitate the calculation of the number of pages needed.

# **Chapter 11**

## **Working Methodology**

This chapter is meant to explain the methodology and the steps taken to ensure the quality of work in this application. The chapter is divided in 2 halves: unit tests and integration tests.

### **11.1 Unit Tests**

During the realization of the project several tests were made to validate the code done, and to assist in finding bugs or errors that the group might have missed through development. The framework Jest [26] was used to make this tests.

### **11.2 Integration Tests**

The integration tests, verify the good behavior of the different routes and the server. The purpose of this tests is to expose defects in the interaction between different software modules when they are integrated. Supertest [27] was the library used to make the integration tests.

# **Chapter 12**

## **Conclusion**

Through the development of the project, the knowledge acquired along the course was applied and also new technologies that are used in a real world environment. Since this project was devised to cooperate with Ericeira Surf Club the group also had the opportunity to experience how a customer relationship proceeds in terms of development.

The major challenges that was faced during the development of the project was the use of React, as it was a library no member of the group had any experience with and it exposes a lot of different tools, such as hooks like useState and useEffect. Since a choice of separating the render of the page from the request to the API was made, the learning of React-Redux was crucial, and although it is simple to learn and implement, its implementation is extensive which meant a more difficult time on debugging errors.

Despite the difficulties the group faced, the project is progressing well and offers most functionalities, albeit some are to be slightly changed to fit Ericeira Surf Club better, and as positive feedback has been received from the organization we can affirm that the project is progressing well and although a bit behind schedule, it is well within the groups reach to finish in time.

# Bibliography

- [1] Ericeira WSR+10. <https://ericeirawsr10.com/ericeira-surf-clube/>.
- [2] React Lifecycle. <https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>.
- [3] NodeJs. <https://nodejs.org/>.
- [4] node-postgres. <https://www.npmjs.com/package/pg>.
- [5] Express framework. <https://expressjs.com/>.
- [6] HTTP. <https://developer.mozilla.org/en-US/docs/Web/HTTP>.
- [7] PostgreSQL. <https://www.postgresql.org/>.
- [8] Heroku. <https://www.heroku.com/>.
- [9] React. <https://reactjs.org/>.
- [10] MUI library. <https://mui.com/>.
- [11] Redux. <https://redux.js.org/>.
- [12] Javascript. <https://www.javascript.com/>.
- [13] Single Page Application. [https://en.wikipedia.org/wiki/Single-page\\_application](https://en.wikipedia.org/wiki/Single-page_application).
- [14] Entity Association. [https://en.wikipedia.org/wiki/Entity%E2%80%93relationship\\_model](https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model).
- [15] passportJS. <https://www.passportjs.org/>.
- [16] Local Strategy. <https://www.passportjs.org/packages/passport-local/>.
- [17] Cookie. [https://en.wikipedia.org/wiki/HTTP\\_cookie](https://en.wikipedia.org/wiki/HTTP_cookie).
- [18] BCrypt. <https://www.npmjs.com/package/bcrypt>.
- [19] salt. [https://en.wikipedia.org/wiki/Salt\\_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography)).

- [20] Hashing, StackAbuse. <https://stackabuse.com/hashing-passwords-in-python-with-bcrypt/>.
- [21] DAL. [https://en.wikipedia.org/wiki/Data\\_access\\_layer](https://en.wikipedia.org/wiki/Data_access_layer).
- [22] Css. <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.
- [23] React hooks. <https://reactjs.org/docs/hooks-intro.html>.
- [24] react-i18next. <https://react.i18next.com/>.
- [25] Tab. <https://mui.com/material-ui/react-tabs/>.
- [26] Jest. <https://jestjs.io/>.
- [27] Supertest. <https://www.npmjs.com/package/supertest>.