



ISEL
INSTITUTO SUPERIOR DE
ENGENHARIA DE LISBOA

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

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

1	Introduction	2
1.1	Motivation	2
1.2	Objectives	2
1.3	Specifications	2
1.4	Report structure	3
2	Technologies	4
3	Architecture	5
3.1	Backend	6
3.2	Frontend	6
4	Data Model	7
5	Server implementation	9
5.1	Web API	10
5.1.1	Authentication	10
5.1.2	Private data management	10
5.1.3	Authorization	11
5.2	Service	11
5.3	DAL	12
6	Client implementation	13
6.1	API access	13
6.2	Client side error handling	14
6.3	Authentication and Authorization	14
6.4	Code structure	15
6.5	Internationalization	15
7	User interface and Functionalities	17
7.1	Homepage	17
7.2	Sign In and Sign Up	18

7.3	Overview	20
7.4	Members	20
7.4.1	Profile	20
7.4.2	All users and All companies	22
7.5	Sports	24
7.5.1	My Sports	24
7.5.2	All Sports	25
7.6	Quotas	25
7.6.1	My Quotas	25
7.6.2	All Quotas	26
7.6.3	Management Quotas	27
7.7	Events	28
7.7.1	My Events	28
7.7.2	All Events	29
7.7.3	Event Page	30
7.8	Candidates	31
7.9	Responsive Design	32
8	Membership Card	34
9	Pagination and Filtering	36
9.1	Pagination	36
9.1.1	Client Side	36
9.1.2	Server Side	37
9.2	Filtering	37
9.2.1	Client Side	37
9.2.2	Server Side	37
10	Working Methodology	38
10.1	Unit Tests	38
10.2	Integration Tests	38
11	Conclusion	39
	References	41

List of Figures

3.1	General Architecture	5
4.1	EA model	7
5.1	API structure	9
5.2	Hashing passwords plus salt	11
6.1	Redux scheme	13
7.1	Homepage	17
7.2	Sign In page	18
7.3	Sign Up page	19
7.4	Application message page	19
7.5	Individual user profile page	21
7.6	Corporate user profile page	21
7.7	All users page	22
7.8	User creation modal	23
7.9	Member's sports page	24
7.10	Sports page	25
7.11	My Quotas Page	26
7.12	All Quotas Page	26
7.13	Quota Creation Modal	27
7.14	Quota Management Page	27
7.15	My Events Page	28
7.16	My Events Page	29
7.17	My Events Page	30
7.18	Candidates page	31
7.19	Profile in a laptop resolution	32
7.20	Profile in a phone and tablet resolution	33
8.1	Membership card example	34
8.2	Member validation scheme	35

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 Objectives

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

Technologies

This chapter describes the technologies used in this application.

The server side was developed in NodeJs [2] and uses the node-postgres [3] module for database access as well as the Express framework [4] for HTTP [5] requests.

A PostgreSQL [6] 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 [7].

The client side was built with React [8], a framework that enables for the building of high-performance responsive apps. The MUI library [9] 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 [10] 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..

Chapter 3

Architecture

The architecture of the developed system is described in this chapter, with its components illustrated in figure 3.1.

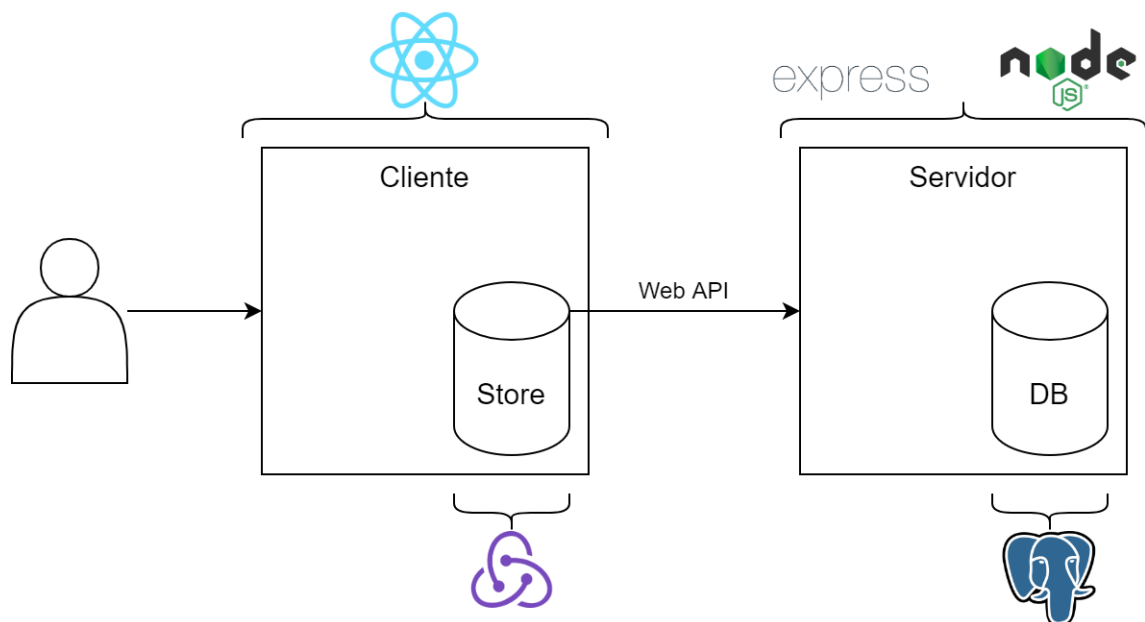


Figure 3.1: General Architecture

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 Operation System or device.

3.1 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.

3.2 Frontend

This web application is an SPA (Single Page Application) [11], 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.

The 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.

Chapter 4

Data Model

The following chapter presents the databases graphical representation in its Entity Association [12] format, describing the existing entities and the relations between them.

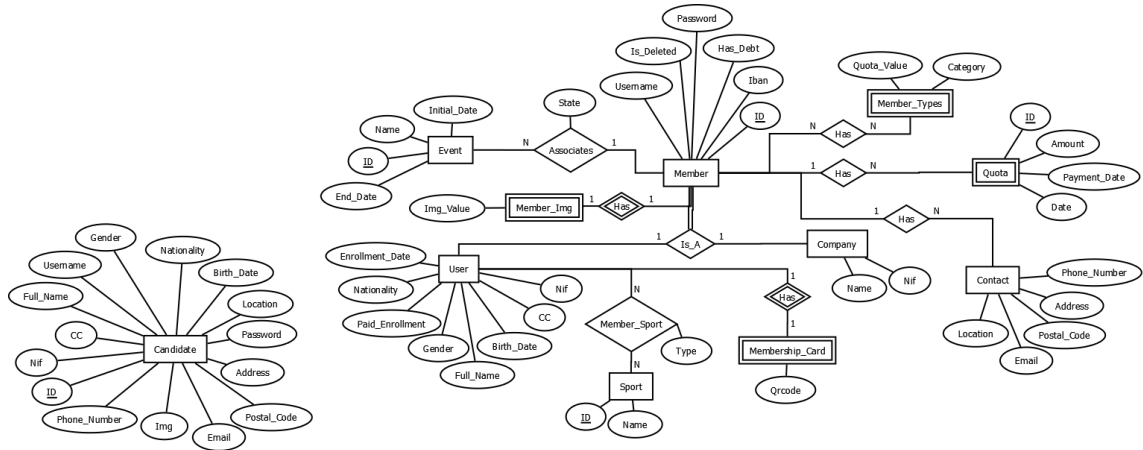


Figure 4.1: EA model

As shown in the figure 4.1, 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 5

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;

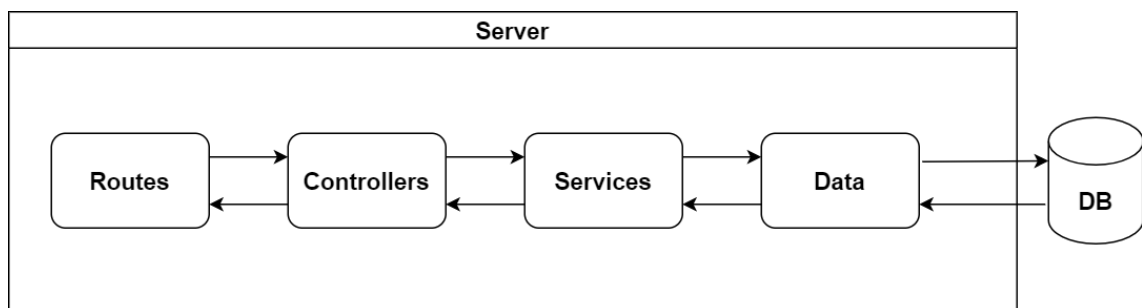


Figure 5.1: API structure

5.1 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.

5.1.1 Authentication

Authentication is done by using the passportJS [13] middleware, a NodeJs authentication middleware, which is based on a local strategy [14] with username and password and validates the authentication attempt by populating the session cookie [15].

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.

5.1.2 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 [16] is used, which does a hash of the entered password, together with the salt [17], 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 5.2 is an schematic example from an article of StackAbuse [18] :

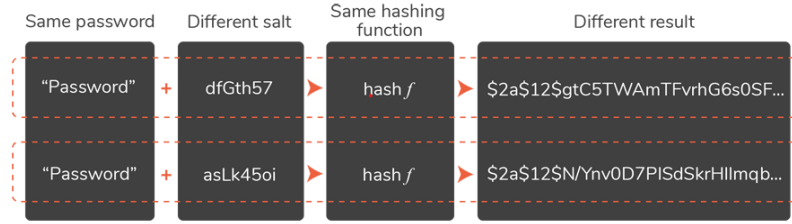


Figure 5.2: Hashing passwords plus salt

5.1.3 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.

5.2 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 [19] 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.

5.3 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 6

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.

6.1 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.

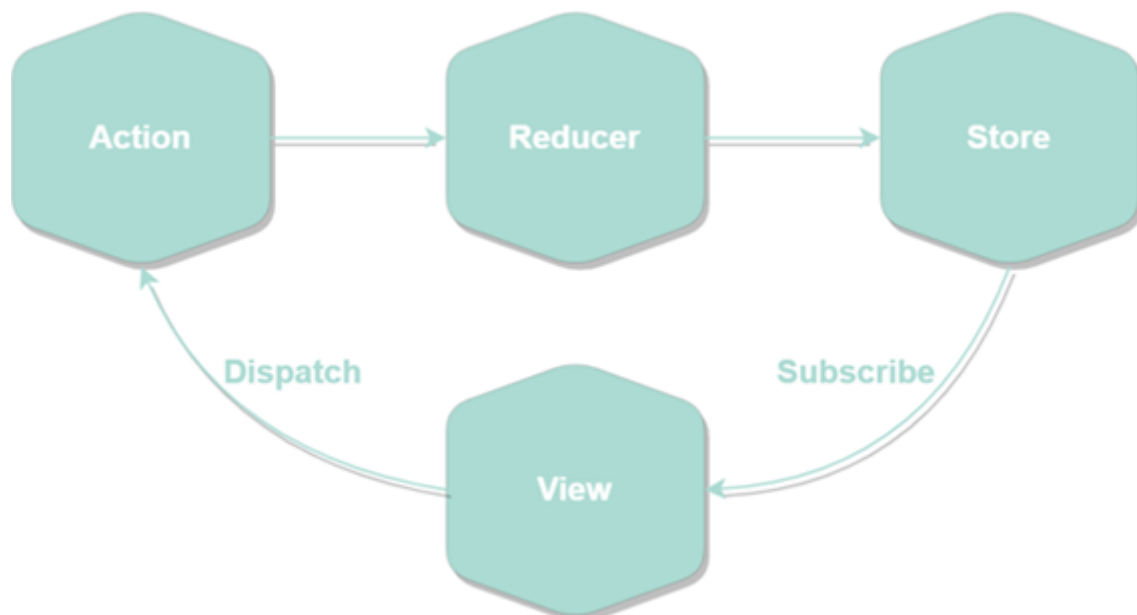


Figure 6.1: Redux scheme

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.

6.2 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.

6.3 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;

6.4 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 [20] of some pages;
- Components - Different react components
- Hooks - The different hooks [21] 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 Home-Header 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

6.5 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` [22] 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 7

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.

7.1 Homepage

The application's Homepage is represented by the figure 7.1. 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.

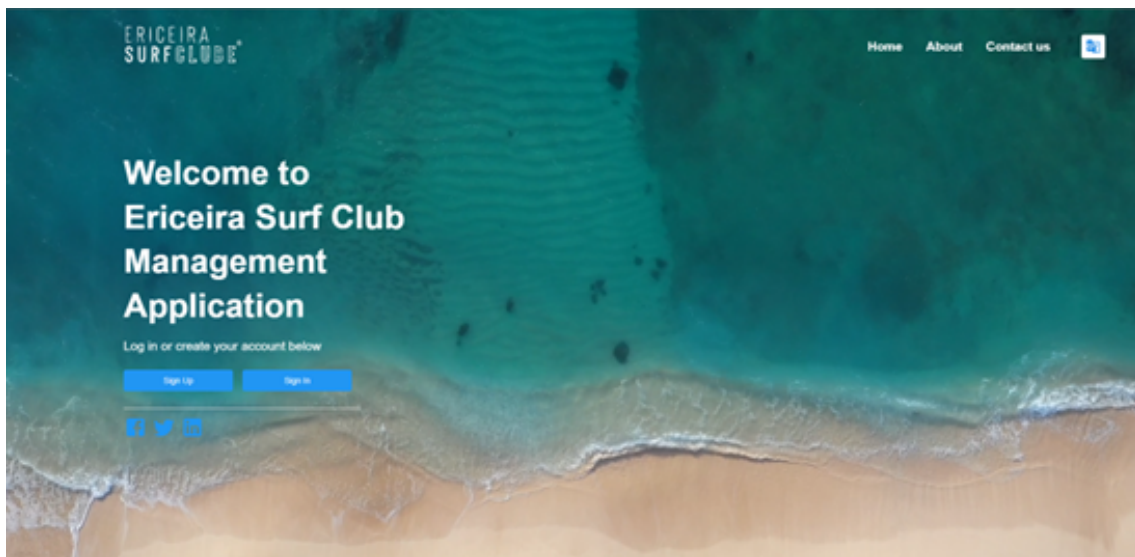


Figure 7.1: Homepage

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.

7.2 Sign In and Sign Up

It is possible to check the aspect of the page for a user to login in the following figure 7.2. 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 7.2: 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 form is shown in the following figure 7.3:

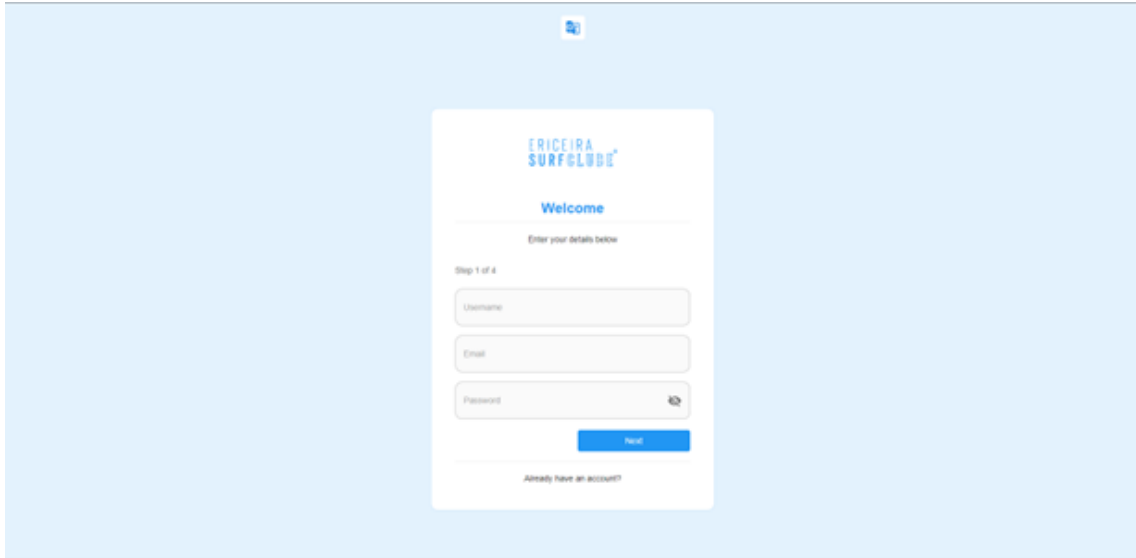
The image shows a web browser window with a light blue background. At the top center, there is a small logo. Below it, a white rectangular box contains the text "ERICEIRA SURFCLUBE" in blue. Underneath the logo, the word "Welcome" is displayed in blue. Below "Welcome", the text "Enter your details below" is shown in a smaller font. Further down, "Step 1 of 4" is indicated. There are three input fields: "Username", "Email", and "Password". The "Password" field has a small eye icon to its right. Below these fields is a blue button labeled "Next". At the bottom of the white box, the text "Already have an account?" is visible.

Figure 7.3: 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.

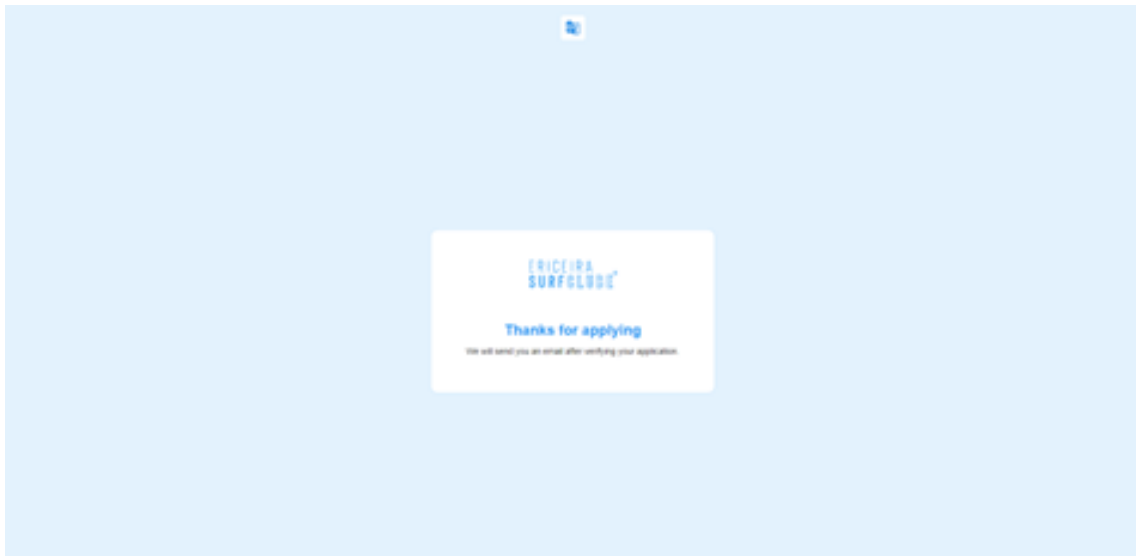
The image shows a web browser window with a light blue background. At the top center, there is a small logo. Below it, a white rectangular box contains the text "ERICEIRA SURFCLUBE" in blue. Underneath the logo, the text "Thanks for applying" is displayed in blue. Below "Thanks for applying", the text "We will send you an email after verifying your application." is shown in a smaller font.

Figure 7.4: Application message page

7.3 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 [ref], along with the NavBar [mui ref] 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]:

7.4 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.

7.4.1 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.

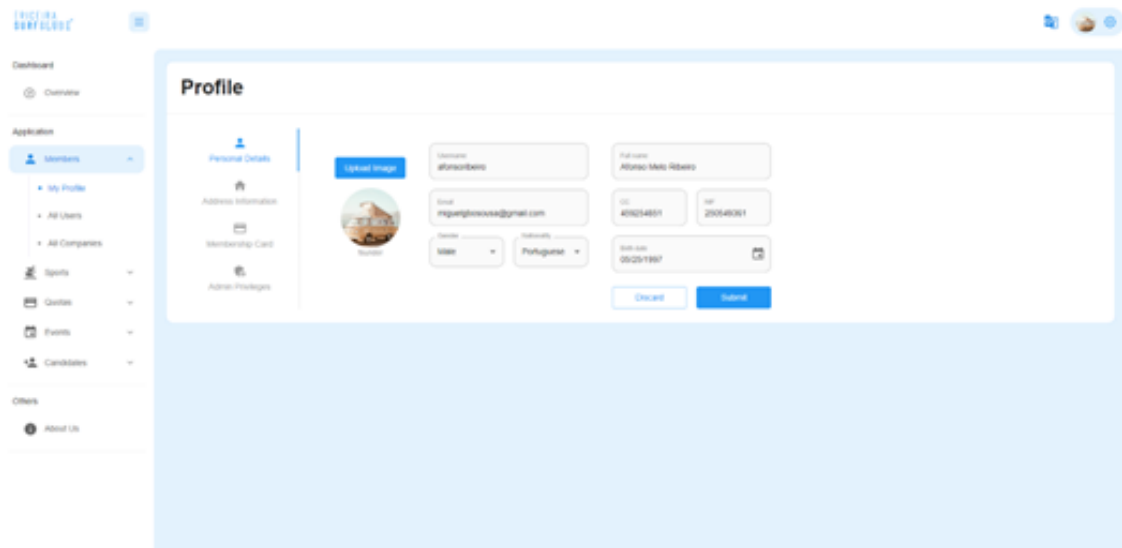


Figure 7.5: Individual user profile page

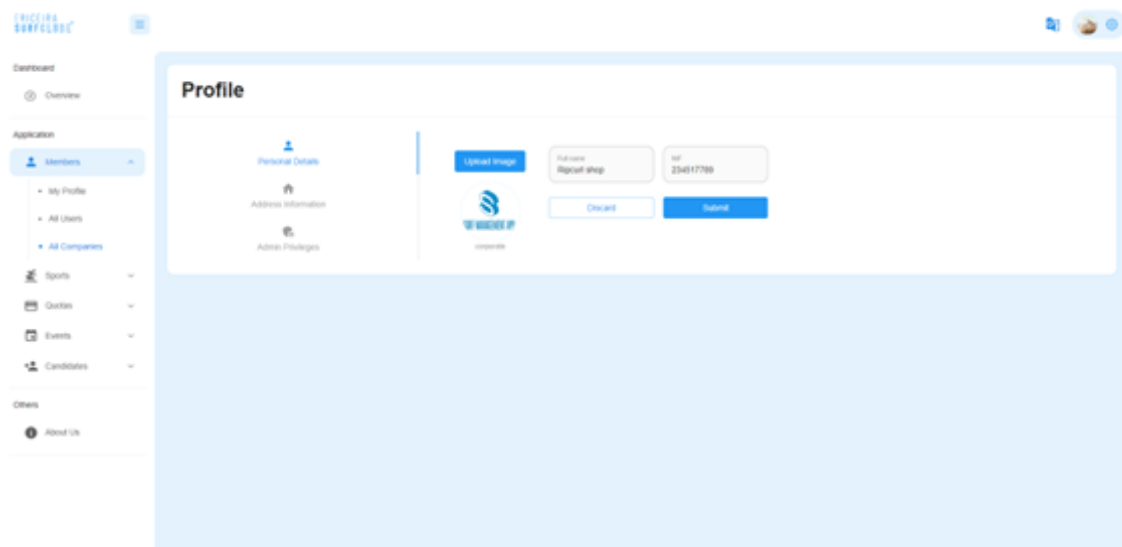


Figure 7.6: Corporate user profile page

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.

7.4.2 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 9.

ID	Username	Full name	Email	Gender	Nationality	Birth date	Member type	Has photo	Has document	Is active	Is owner
1	John	John Doe	john@doe.com	Male	Portuguese	2000-08-15	Member	X	✓	X	X
2	maria	Maria da Silva Costa	maria@costa.com	Female	Portuguese	2001-08-15	Member	X	✓	X	X
3	John	John Doe	john@doe.com	Male	Portuguese	2001-08-15	Member	X	✓	X	X
4	maria	Maria	maria@doe.com	Female	Portuguese	2001-08-15	Member	X	✓	X	X
5	maria	Maria da Silva Costa	maria@costa.com	Female	Portuguese	2001-08-15	Member	X	✓	X	X

Figure 7.7: All users page

Aside from data visualization, this page allows you to create a user by displaying a Modal [ref], figure 7.8, 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.

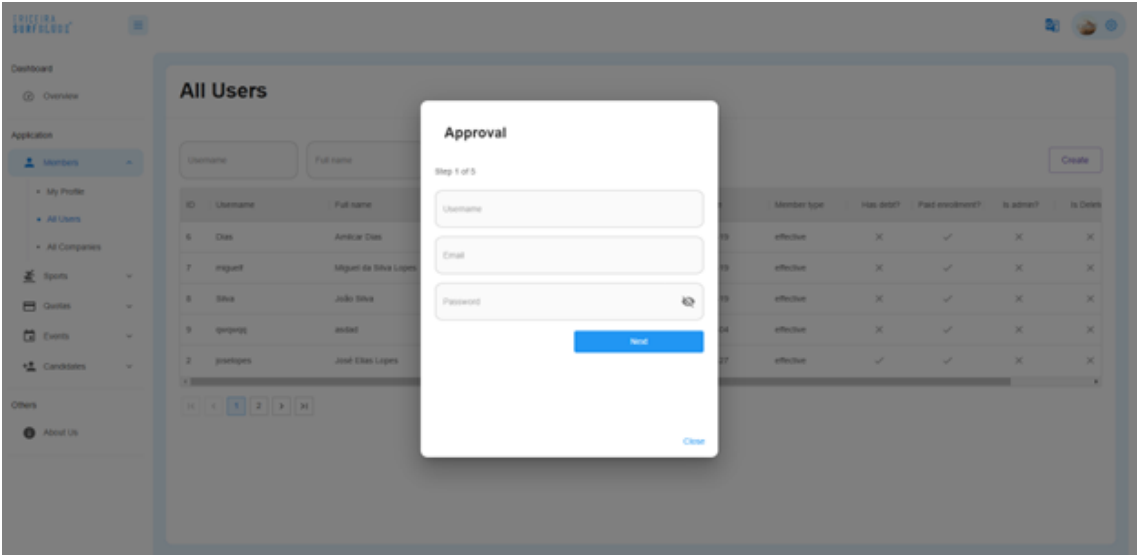


Figure 7.8: User creation modal

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.

7.5 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.

7.5.1 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 7.9:

Name	Type	Federation Number	Federation ID	Federation Name	Years Practiced	Status
Running	Running	1001	10	Federacion Portuguesa de Atletismo	2015-2020	Y
Handball	Handball	1002	10	Federacion Portuguesa de Handball	2015-2020-2021	Y
Basketball	Basketball	1003	10	Federacion Portuguesa de Basketball	2015-2020	Y
Football	Football	1004	10	Federacion Portuguesa de Futebol	2015-2020	Y
Hockey	Hockey	1005	10	Federacion Portuguesa de Hockey	2015-2020	Y

Figure 7.9: Member's sports page

7.5.2 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 7.10.

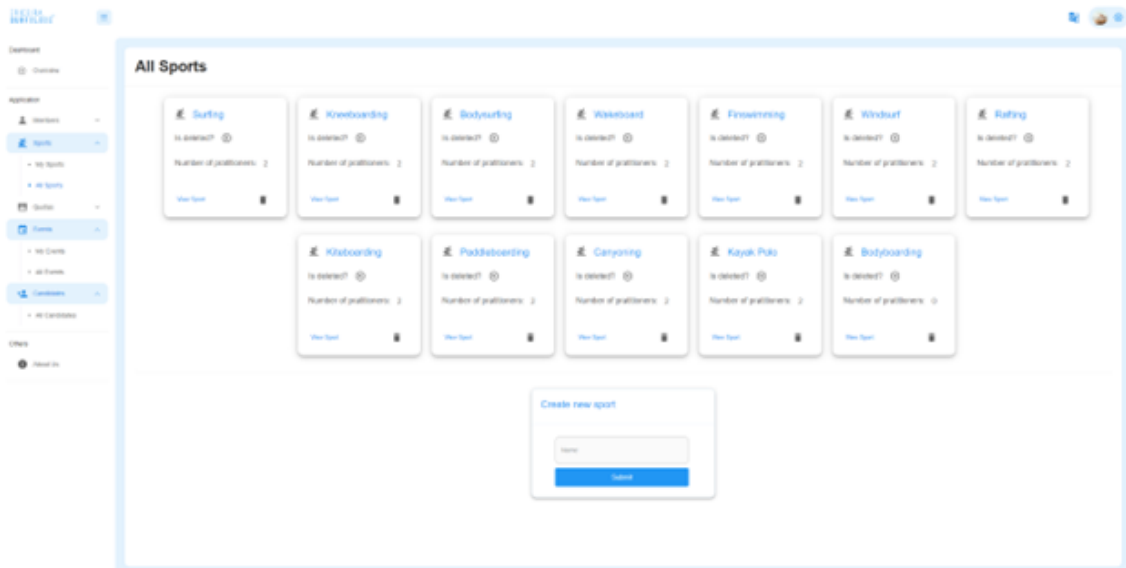


Figure 7.10: Sports page

7.6 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.

7.6.1 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 quotas is still for paying.

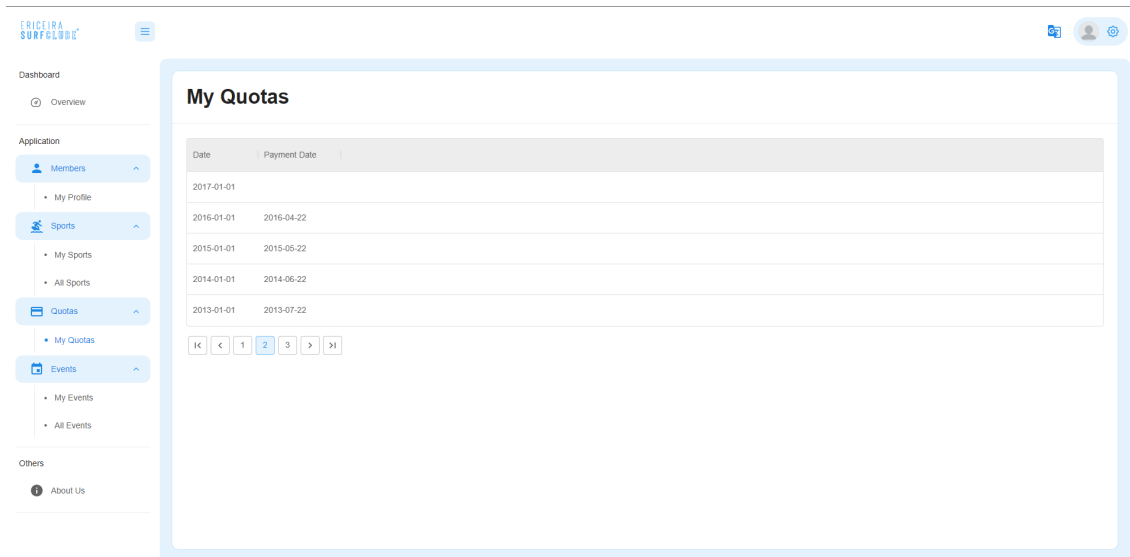


Figure 7.11: My Quotas Page

7.6.2 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.

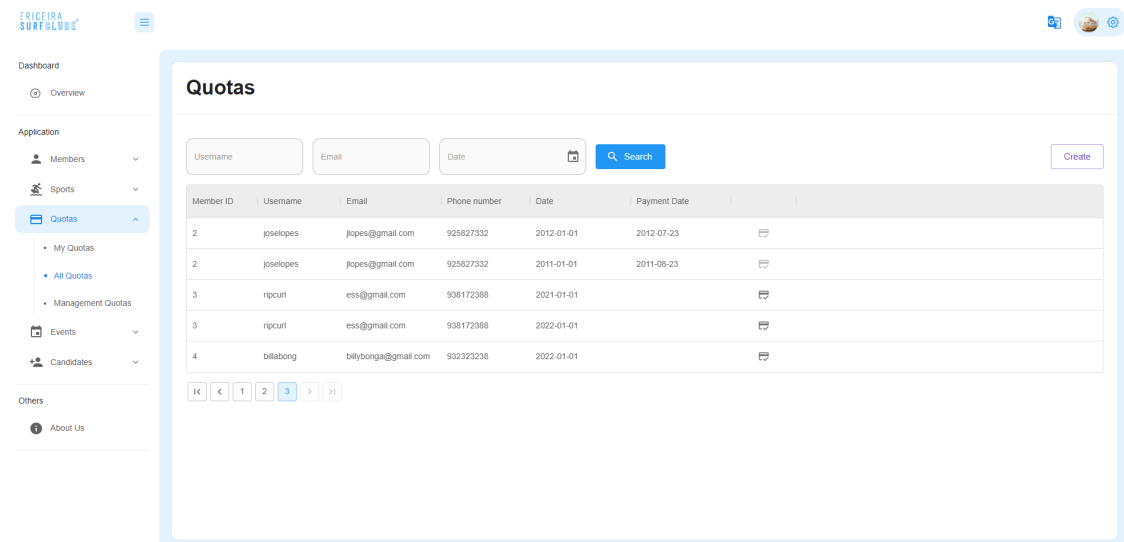


Figure 7.12: All Quotas Page

Aside from data visualization, this page allows you to create a new quota by displaying a Modal [ref], 7.13

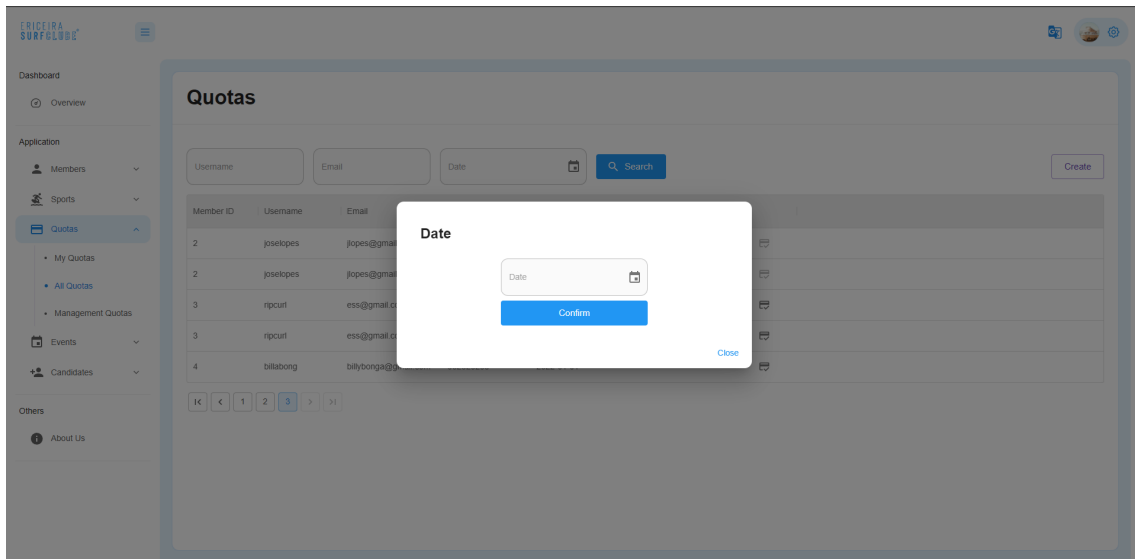


Figure 7.13: Quota Creation Modal

7.6.3 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.

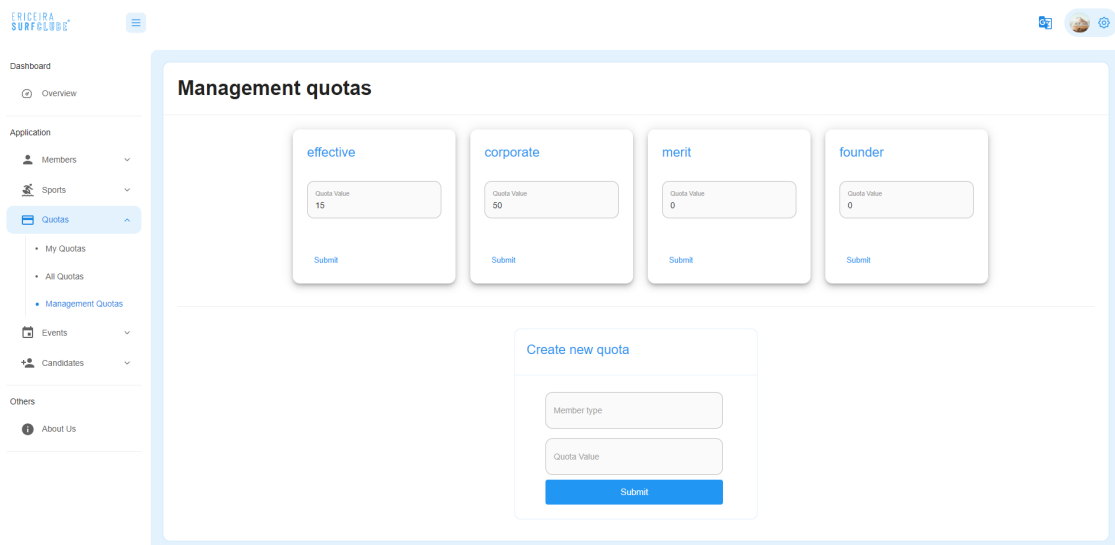


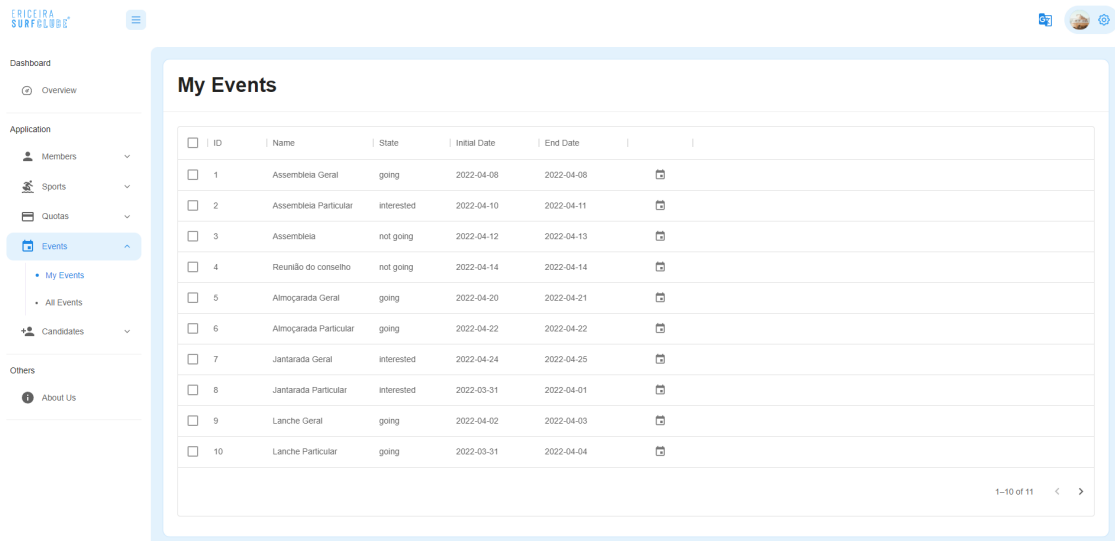
Figure 7.14: Quota Management Page

7.7 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.

7.7.1 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.



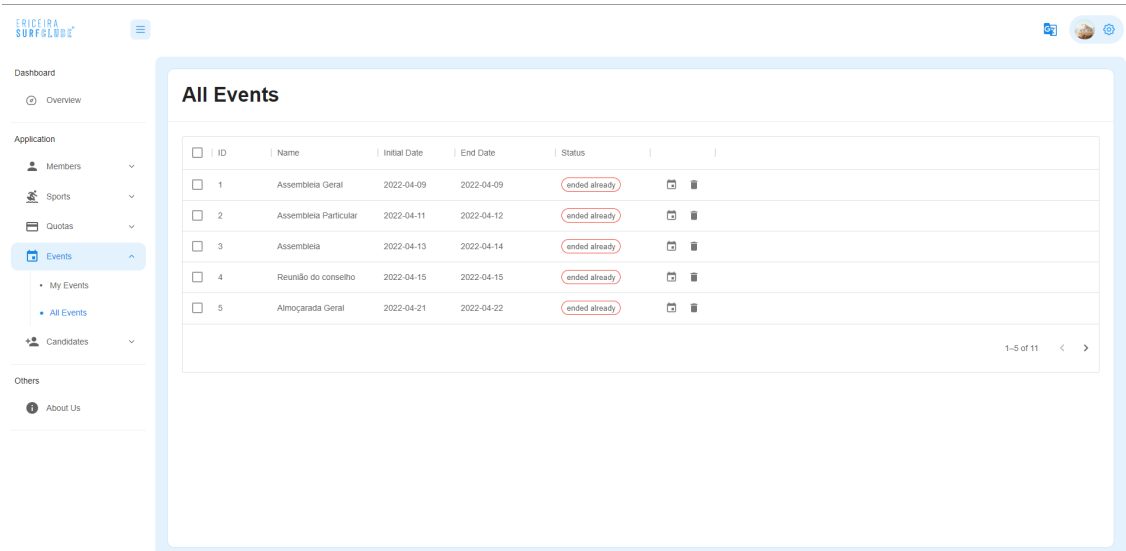
<input type="checkbox"/>	ID	Name	State	Initial Date	End Date	
<input type="checkbox"/>	1	Assembleia Geral	going	2022-04-08	2022-04-08	
<input type="checkbox"/>	2	Assembleia Particular	interested	2022-04-10	2022-04-11	
<input type="checkbox"/>	3	Assembleia	not going	2022-04-12	2022-04-13	
<input type="checkbox"/>	4	Reunião do conselho	not going	2022-04-14	2022-04-14	
<input type="checkbox"/>	5	Almoçada Geral	going	2022-04-20	2022-04-21	
<input type="checkbox"/>	6	Almoçada Particular	going	2022-04-22	2022-04-22	
<input type="checkbox"/>	7	Jantarada Geral	interested	2022-04-24	2022-04-25	
<input type="checkbox"/>	8	Jantarada Particular	interested	2022-03-31	2022-04-01	
<input type="checkbox"/>	9	Lanche Geral	going	2022-04-02	2022-04-03	
<input type="checkbox"/>	10	Lanche Particular	going	2022-03-31	2022-04-04	











1-10 of 11 < >

Figure 7.15: My Events Page

7.7.2 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.



<input type="checkbox"/>	ID	Name	Initial Date	End Date	Status	
<input type="checkbox"/>	1	Assembleia Geral	2022-04-09	2022-04-09	ended already	 
<input type="checkbox"/>	2	Assembleia Particular	2022-04-11	2022-04-12	ended already	 
<input type="checkbox"/>	3	Assembleia	2022-04-13	2022-04-14	ended already	 
<input type="checkbox"/>	4	Reunião do conselho	2022-04-15	2022-04-15	ended already	 
<input type="checkbox"/>	5	Almoçada Geral	2022-04-21	2022-04-22	ended already	 

1-5 of 11 < >

Figure 7.16: 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.

7.7.3 Event Page

This page shows all the information about a specif 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 displays the 'Assembleia Particular' event page within the 'Elections Surfer' application. The interface includes a sidebar with navigation options like Dashboard, Application, and Others. The main content area is divided into sections for Event Information, a List of participants, and summary statistics.

Event Information

Initial Date: 2022-04-11 End Date: 2022-04-12

List

<input type="checkbox"/>	Name	state	
<input type="checkbox"/>	joselope	going	
<input type="checkbox"/>	afonsoibe...	interested	

1-2 of 2 < >

Summary: People going: 1 People not going: 0 People interested: 1

Figure 7.17: My Events Page

7.8 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 form referred to in the application process at point 7.2. The previous-mentioned page can only be accessed by users who are administrators.

This information may be verified using the following figure 7.18:

ID	Licenseplate	Full name	Email	Phone number	Gender	Birth date	Location	Address	Postal code	ZIP	Age
1	Daria	Daria Daria	daria@gmail.com	912345678	Other	1999-05-15	Spain	Spain-De-Madrid-470	28714-045	137170202	21
2	Roger	Roger Roger	roger@gmail.com	911444333	Other	1999-01-01	Spain	Spain-De-Madrid-480	28001-276	135504000	22
3	Pedro	Pedro Pedro	pedro@gmail.com	911234567	Male	1997-05-27	Spain	Spain-De-Madrid-471	28001-276	135504000	22
4	Maria	Maria Maria	maria@gmail.com	911234567	Female	1997-04-01	Spain	Spain-De-Madrid-480	28001-276	135504000	22
5	Ingrid	Ingrid Ingrid	ingrid@gmail.com	911234567	Other	1999-02-10	Spain	Spain-De-Madrid-480	28001-276	135504000	21

Figure 7.18: Candidates page

7.9 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 7.19 and 7.20 shown below, the elements are arranged differently depending on the screen resolution, furthermore as can be seen by the Tabs [ref] component of MUI [ref], their orientation changes to suit better. to the screen where it is inserted.

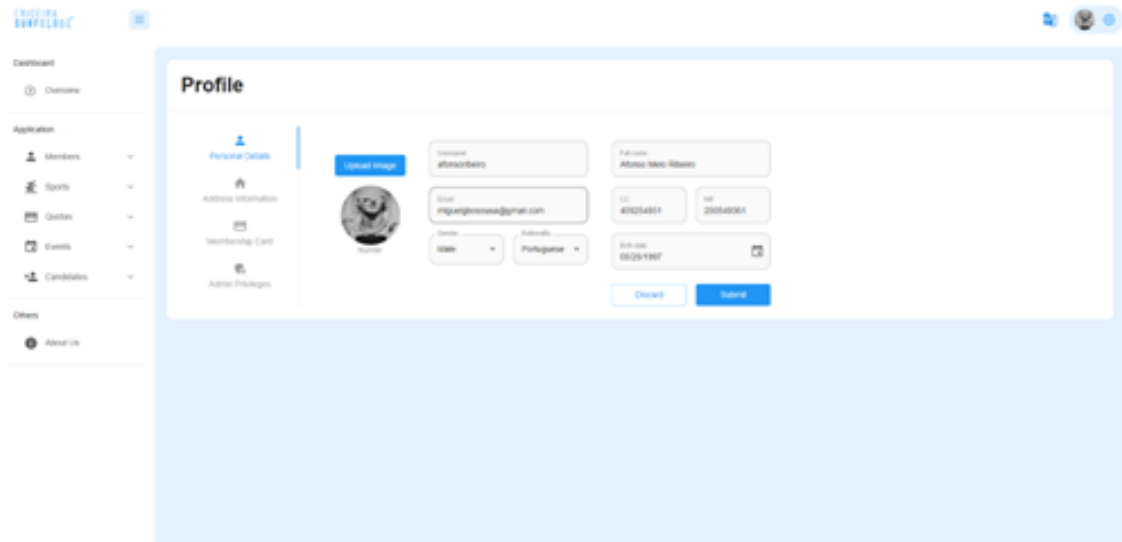


Figure 7.19: Profile in a laptop resolution

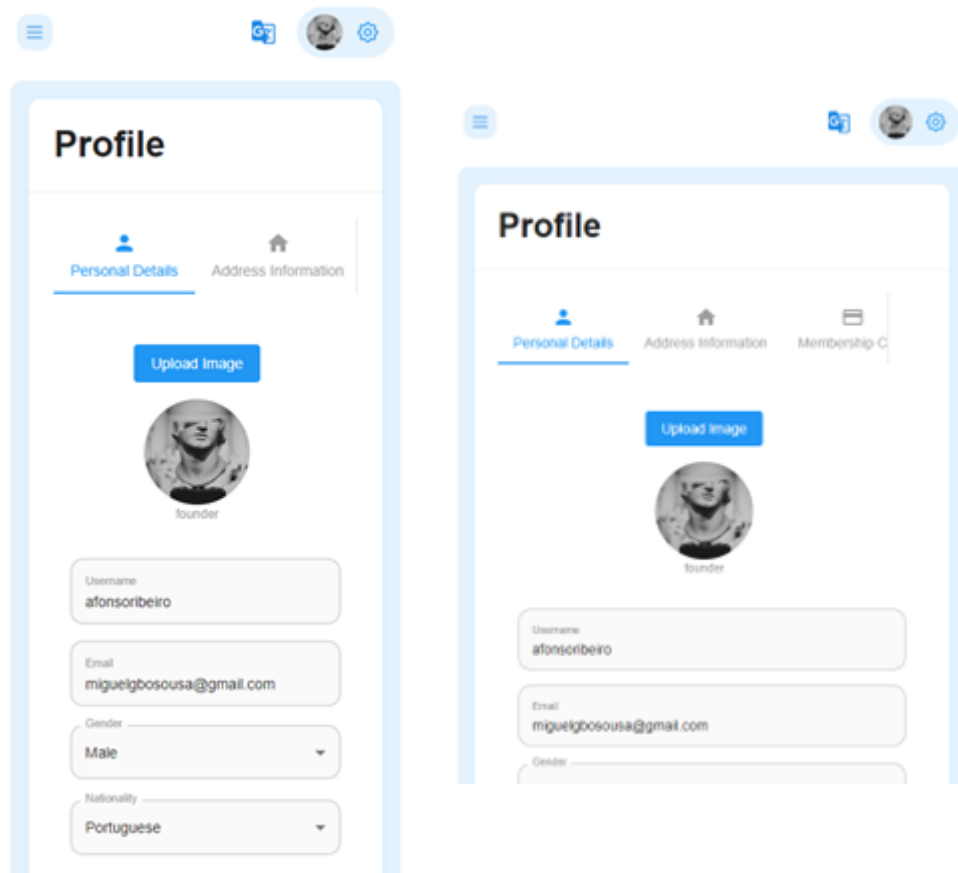


Figure 7.20: Profile in a phone and tablet resolution

Chapter 8

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 8.1: 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.

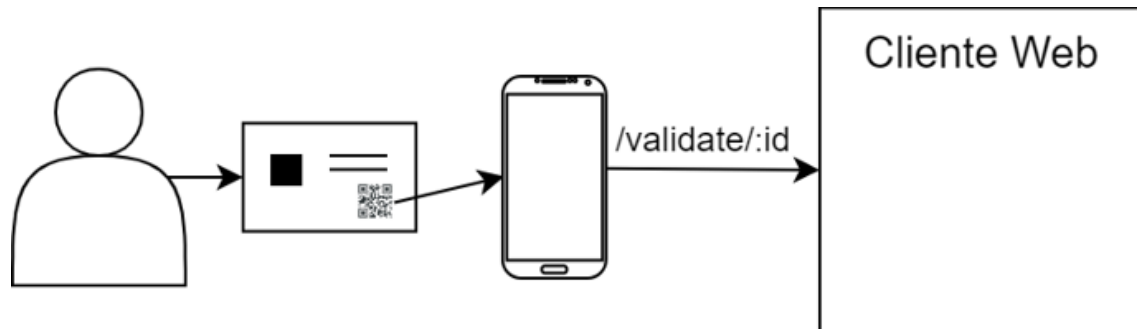


Figure 8.2: Member validation scheme

Chapter 9

Pagination and Filtering

9.1 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.

9.1.1 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.

9.1.2 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.

9.2 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.

9.2.1 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.

9.2.2 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 10

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.

10.1 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 [23] was used to make this tests.

10.2 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 [24] was the library used to make the integration tests.

Chapter 11

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] NodeJs. <https://nodejs.org/>.
- [3] node-postgres. <https://www.npmjs.com/package/pg>.
- [4] Express framework. <https://expressjs.com/>.
- [5] HTTP. <https://developer.mozilla.org/en-US/docs/Web/HTTP>.
- [6] PostGreSQL. <https://www.postgresql.org/>.
- [7] Heroku. <https://www.heroku.com/>.
- [8] React. <https://reactjs.org/>.
- [9] MUI library. <https://mui.com/>.
- [10] Redux. <https://redux.js.org/>.
- [11] Single Page Application. https://en.wikipedia.org/wiki/Single-page_application.
- [12] Entity Association. https://en.wikipedia.org/wiki/Entity%E2%80%9393relationship_model.
- [13] passportJS. <https://www.passportjs.org/>.
- [14] Local Strategy. <https://www.passportjs.org/packages/passport-local/>.
- [15] Cookie. https://en.wikipedia.org/wiki/HTTP_cookie.
- [16] BCrypt. <https://www.npmjs.com/package/bcrypt>.
- [17] salt. [https://en.wikipedia.org/wiki/Salt_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography)).
- [18] Hashing, StackAbuse. <https://stackabuse.com/hashing-passwords-in-python-with-bcrypt/>.
- [19] DAL. https://en.wikipedia.org/wiki/Data_access_layer.

- [20] Css. <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.
- [21] React hooks. <https://reactjs.org/docs/hooks-intro.html>.
- [22] react-i18next. <https://react.i18next.com/>.
- [23] Jest. <https://jestjs.io/>.
- [24] Supertest. <https://www.npmjs.com/package/supertest>.