

UNIVERSIDADE DE LISBOA  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE INFORMÁTICA



## **Registration Platform For Federations And Gymnastics Clubs**

Diogo de Oliveira Lopes

**Mestrado em Engenharia Informática**

Versão Pública

Trabalho de Projeto orientado por:  
Prof. Doutor Nuno Cruz Garcia



## Resumo

Os clubes e federações de ginástica necessitam de inscrever os seus membros e equipas em competições e gerir estes membros e equipas de uma forma mais digitalizada. O objectivo deste projeto é criar uma solução para este problema.

Este projeto foi desenvolvido em contexto de estágio na empresa Acro Companion, uma startup criada em 2016 na Bélgica, que desenvolve software para ginástica. O estágio teve início a 13 de setembro de 2021 e terminou a 13 de junho de 2022, tendo durado um total de nove meses. Depois disso, o projeto continuou a ser melhorado, já em contexto de trabalho na empresa, mas sem que este fosse o único foco.

O presente documento visa descrever o contexto e as tecnologias associadas a este projeto, Membership and Registrations, como também demonstrar o que foi feito durante os nove meses de estágio e os dois meses que se seguiram, em termos de planeamento e desenvolvimento, bem como propor melhorias que poderão ser implementadas no futuro.

Durante o desenvolvimento do projeto, várias tecnologias foram utilizadas. O projeto foi escrito em Typescript, seguindo o paradigma da programação reativa. Enquadrando-se o projeto numa aplicação web, esta tem uma GUI (Graphical User Interface), que permite aos utilizadores interagirem com o programa. A GUI necessita de reagir a eventos imprevisíveis. O paradigma da programação reativa permite que o programador não se preocupe com a ordem na qual os eventos ocorrem. Especificamente, a biblioteca RxJS do Angular foi utilizada para este propósito. Angular é a framework de desenvolvimento web que a Acro Companion utiliza para desenvolver a sua aplicação web. A Cloud Firestore foi utilizada como base de dados NoSQL, e a Cloud Storage para armazenar ficheiros. Para testes, o Playwright e o emulador do Firestore, bem como o Jest foram utilizados. Algumas outras tecnologias que foram úteis para o projeto foram o npm para gerir pacotes, o Azure DevOps para continuous integration, continuous testing e continuous delivery. O Chrome DevTools foi usado para debugging, e o Adobe XD para criar os protótipos para a GUI.

Existem duas partes importantes neste projeto, a parte relacionada com a gestão de membros e equipas, Membership, que já existia inicialmente, tendo sido expandida. E a parte relacionada com a inscrição desses mesmos membros e equipas em competições nacionais ou internacionais de ginástica, Registrations. Esta última, não existia inicialmente e foi totalmente criada a partir do zero, tendo sido o maior foco do projeto, sendo o objetivo criar uma solução que fosse genérica, de modo a poder ser utilizada pelo maior número de clientes da Acro Companion possível e de modo, também, a poder ser facilmente expandida com novas funcionalidades no futuro. Este projeto foi desenvolvido na aplicação web da empresa, que é uma aplicação do tipo serverless.

Numa fase inicial do projeto, tentou-se perceber o ambiente envolvente do mesmo. Quais os conceitos envolvidos e como as coisas funcionavam. Começou-se por definir qual era o domínio da aplicação.

Para isso, um modelo de domínio foi criado, detalhando todos os conceitos associados ao projeto e como estes conceitos se relacionam entre si: Clubes, Federações e Competições que têm Gestores capazes de criar Inscrições dos Ginastas e Equipas dos Clubes nas Competições. Depois, registaram-se quais os casos de uso já implementados.

Em relação ao planeamento que foi feito para o desenvolvimento do projeto, começou-se por definir quais seriam os casos de uso a implementar.

Do lado dos clubes e federações, os gestores necessitariam de procurar por competições que estivessem abertas a inscrições, inscrever ginastas, equipas e juizes nessas competições, editar uma inscrição previamente criada e cancelar uma inscrição. Já durante a fase de implementação, dois casos de uso foram adicionados. Os gestores de clubes ou federações deveriam também ser capazes de editar o nível em que um ginasta iria competir numa dada competição, o email do treinador responsável por esse ginasta na competição, e o tipo de exercícios a serem realizados na competição. Esses mesmos gestores deveriam também conseguir editar o nível em que uma equipa iria competir, bem como o email do treinador responsável pela mesma.

Do lado das competições, os gestores necessitariam de abrir uma competição para receber inscrições, aprovar as inscrições recebidas, rejeitar as inscrições recebidas, importar ginastas, equipas, juizes e exercícios da solução Registrations para a parte da aplicação responsável pela administração da competição e, finalmente, poder ver essa informação que foi importada listada.

Seguiu-se a definição do modelo de dados, tanto da parte relacionada com Membership, já existente, mas cujo modelo foi expandido, bem como da parte relacionada com Registrations, para a qual o modelo foi definido de raiz. Em relação à Membership, definiram-se algumas propriedades de modo a possibilitar a criação de equipas de vários tipos de modalidades (inicialmente apenas se conseguiam criar equipas para a modalidade de acrobática) e definiram-se também propriedades específicas a ginastas e juizes. Em relação à Registrations, definiu-se que seria necessário criar uma nova coleção na base de dados para guardar os documentos referentes às inscrições. Definiram-se também as propriedades que seriam armazenadas para cada uma das inscrições e acrescentaram-se propriedades aos ginastas e equipas que seriam úteis durante o processo de inscrição dos mesmos. Por fim acrescentaram-se também algumas propriedades às competições, de modo a que estas pudessem abrir o processo de inscrição.

Depois, definiu-se um diagrama de classes, onde se detalhou como o código seria organizado de modo a implementar o projeto. Neste processo, definiu-se a necessidade de criar novos componentes, principalmente relacionados com a parte referente às inscrições. Assim, definiu-se a necessidade de criar componentes para listar as inscrições tanto do lado das competições como do lado dos clubes e federações, de adicionar um componente para criar, editar ou ver uma inscrição em específico. Definiram-se também alguns componentes auxiliares. Foram também definidos os serviços que seriam utilizados por esses componentes, nomeadamente, um serviço relacionado com as inscrições em geral e um serviço que seria unicamente responsável pela inscrição a ser manipulada num dado momento. Por fim, foram criados alguns serviços que assumiram responsabilidades previamente atribuídas a outros serviços de modo incorreto.

De seguida, definiram-se os vários estados de uma inscrição, 'rascunho', 'a aplicar', 'aprovada', 'rejeitada', 'alterações requisitadas pelo aplicante', 'cancelamento requisitado' e 'cancelada'. Definiu-se também o modo como uma inscrição poderia avançar de um estado para outro.

A última fase do planeamento foi a criação de protótipos para a GUI.

Depois da fase de planeamento, passou-se para a fase de implementação. Este documento descreve a implementação de cada um dos novos casos de uso definidos e o modo como a programação reativa foi utilizada. É também feita a comparação dos protótipos definidos para a GUI com o resultado final. Apresenta-se ainda a implementação de outras partes da aplicação, não diretamente relacionadas com os casos de uso definidos. Nomeadamente, apresentam-se os melhoramentos feitos à parte de Membership, um componente que permite a seleção de níveis de ginástica válidos para uma dada modalidade e país, os testes feitos para a criação de membros e para um dos Observables de um serviço, o modo como se configura o diálogo para a criação, edição ou visualização de uma inscrição, e um serviço onde a programação reativa foi fortemente utilizada.

Por fim, é feita uma análise ao impacto do projeto, analisa-se o que correu menos bem e sugerem-se múltiplas melhorias a ser implementadas no futuro.

**Palavras-chave:** Angular, RxJS, Gymnastics, Serverless, Firebase



## **Abstract**

Gymnastics clubs and federations need to register their members and teams in competitions and handle memberships in a more digitalized way. The goal of this project is to create a solution for this problem in Angular, following the reactive programming paradigm, and using a serverless architecture. This document will give a good comprehensive overview of the conceptual world of the organization of gymnastics events, and explain the architecture, and technologies used by the web application where the Membership and Registrations solution will be included. The solution will have to take into account functionality, security, and maintainability requirements. The solution's development will be presented, from the planning phase, where the use cases were defined as well as the organization of the code, to the final implementation of those use cases. Finally a reflection on the impact of this new solution on gymnastics and what can still be improved will be presented.

**Keywords:** Angular, RxJS, Gymnastics, Serverless, Firebase





# Contents

<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Context . . . . .	1
1.2 Motivation . . . . .	2
1.2.1 The problem and its importance . . . . .	2
1.3 Goals and contributions . . . . .	3
1.3.1 The solution . . . . .	3
1.4 Structure of the document . . . . .	4
<b>2 Related Work</b>	<b>5</b>
2.1 Acro Companion’s previous registrations solution . . . . .	5
2.2 Other gymnastics solutions . . . . .	5
2.3 Other registrations solutions . . . . .	5
<b>3 Technologies and Background</b>	<b>9</b>
3.1 Technologies . . . . .	9
3.1.1 Frontend Technologies . . . . .	9
3.1.2 DevOps Technologies . . . . .	9
3.1.3 Cloud Technologies . . . . .	9
3.1.4 NPM . . . . .	9
3.1.5 Chrome DevTools . . . . .	10
3.1.6 Typescript . . . . .	10
3.1.7 Angular . . . . .	10
3.1.8 RxJS . . . . .	11
3.1.9 Firebase . . . . .	11
3.1.10 Azure DevOps . . . . .	11
3.1.11 Playwright . . . . .	12
3.1.12 Jest . . . . .	12
3.1.13 Git . . . . .	12
3.1.14 Adobe XD . . . . .	12
3.2 Background . . . . .	13
3.2.1 Previous Operation . . . . .	13

3.2.2	Architecture . . . . .	13
3.2.3	Domain and implemented use cases . . . . .	14
<b>4</b>	<b>Registrations Platform</b>	<b>19</b>
4.1	Planning . . . . .	19
4.1.1	New Use Cases . . . . .	19
4.1.2	Data Model . . . . .	21
4.1.3	Registrations Status Flow . . . . .	21
4.1.4	GUI Prototypes . . . . .	22
4.2	Methodology . . . . .	22
4.2.1	Functional requirements . . . . .	22
4.2.2	Non-functional requirements . . . . .	23
4.3	Implementation . . . . .	23
4.3.1	Searching for competitions open for registrations . . . . .	24
4.3.2	Registering Gymnasts, Teams, and Judges in a competition . . . . .	26
4.3.3	Editing Gymnast’s level, coach email, or apparatuses for registration . . . . .	28
4.3.4	Editing Team’s level, or coach email for registration . . . . .	32
4.3.5	Editing a registration . . . . .	32
4.3.6	Cancelling a registration . . . . .	33
4.3.7	Setting a competition open for registrations . . . . .	35
4.3.8	Approving a registration . . . . .	36
4.3.9	Rejecting a registration . . . . .	39
4.3.10	Membership . . . . .	41
4.3.11	Level Selector . . . . .	43
4.3.12	Testing . . . . .	45
<b>5</b>	<b>Conclusions and Forthcoming Work</b>	<b>47</b>
<b>A</b>	<b>Time management gantt chart.</b>	<b>51</b>
	<b>Bibliography</b>	<b>55</b>

# List of Figures

3.2	Domain model. . . . .	14
3.1	Web application's architecture. . . . .	15
3.3	Already implemented use cases for competition's managers. . . . .	16
3.4	Already implemented use cases for applying organization's managers. . . . .	16
3.5	Original dialog for creating a member. . . . .	17
3.6	Original dialog for creating a team. . . . .	17
3.7	Original view of the members. . . . .	18
4.1	MVP's use cases for applying organizations. . . . .	20
4.2	MVP's use cases for competitions. . . . .	20
4.3	Stepper on the GUI. . . . .	23
4.4	Implemented stepper. . . . .	23
4.5	Registrations' view prototype. . . . .	24
4.6	Registrations' view implementation. . . . .	24
4.7	GUI Prototype for Competition Selection . . . . .	25
4.8	Implementation of Competition Selection . . . . .	25
4.9	GUI Prototype for member and team selection. . . . .	26
4.10	Gymnast selection implementation. . . . .	26
4.11	Team selection implementation. . . . .	27
4.12	Judge selection implementation. . . . .	27
4.13	Registration summary. . . . .	28
4.14	Gymnast customization. . . . .	29
4.15	Gymnast customization with a member selected. . . . .	29
4.16	Level selection in gymnast customization. . . . .	30
4.17	Gymnast customization with multiple members selected. . . . .	30
4.18	Level and coach selection for multiple members in gymnasts customization. . . . .	31
4.19	Gymnasts customization for discipline with apparatus selection. . . . .	31
4.20	Gymnasts customization for discipline with apparatuses selected. . . . .	32
4.21	Selected competition details. . . . .	33
4.22	GUI prototype for cancelling a registration through Registrations' view. . . . .	34
4.23	Implementation for cancelling a registration through Registrations' view. . . . .	34
4.24	Implementation for cancelling a registration through the dialog's view. . . . .	35
4.25	GUI prototype for opening a competition for registrations. . . . .	35

4.26	Implementation for opening a competition for registrations. . . . .	36
4.27	Approving multiple registrations. . . . .	36
4.28	Confirmation dialog for approving all approvable registrations. . . . .	37
4.29	Approving all approvable registrations. . . . .	37
4.30	Applying organization details. . . . .	38
4.31	Selected gymnasts from the competition side. . . . .	38
4.32	Customized gymnasts from the competition side. . . . .	39
4.33	Registration summary from the competition side. . . . .	39
4.34	Rejecting multiple registrations. . . . .	40
4.35	Rejecting changes for a registration. . . . .	40
4.36	Definition of member properties. . . . .	41
4.37	Definition of gymnast-specific properties. . . . .	41
4.38	Acrobatic team, with category 'Male Pair'. . . . .	42
4.39	Acrobatic team, with category 'Women Group'. . . . .	42
4.40	Dialog to select gymnasts for team. . . . .	43
4.41	Rhythmic team, with category 'Group 8'. . . . .	43
4.42	Level options. . . . .	44
5.1	AcroCompanion's end-to-end application. . . . .	47
A.1	Time management gantt chart. . . . .	52



# Chapter 1

## Introduction

### 1.1 Context

As part of the final year of the Master's Degree in Informatics Engineering at the Faculty of Sciences of the University of Lisbon, an effort was put into developing a project for the company Acro Companion as an intern. The project development started on the 13th of September and ended on the 13th of June, spanning the whole nine months of the internship. After that, the project continued being improved as a developer officially working for Acro Companion, however, it was no longer the sole focus of development as the internship was over.

This document aims to describe the context and technologies related to this project, Membership and Registrations, as well as demonstrate what was done during the nine months of the internship plus the following two months during which the project was improved, in terms of planning and development, and what can still be improved in the future.

Acro Companion is a startup created in 2016 in Ghent, Belgium, that developed a web application for gymnastics. Before the beginning of the project, the web application already allowed its users to create sheets for exercise creation and correction, manage competitions, and judge and score exercises in a fraction of the time. For example, according to Acro Companion's founder, exercise creation used to take around 40 minutes and was reduced to 4 minutes. It also provided live stream integration. Acro Companion's team incorporates members from different nationalities, all working remotely from home.

Regarding time management, as discussed, the project was developed for nine months in the context of an internship and then for two more months in a work context, but no longer being the sole focus of development. It will continue to be expanded in the future, but this document focuses on what was done until the time of this writing.

Annex [1] shows a Gantt chart displaying the work that was done during the total of 11 months of development. The x-axis is scaled weekly, with every tick representing a Monday. The first day of each month is represented by a black vertical line, and the end of the 9 months of internship are represented by a green vertical line. The y-axis contains the main tasks that were performed during the project's course.

In the beginning, the development rhythm was slow, and by December it started to gain some speed. The Registrations implementation phase started in February, already after the presentation of the preliminary report. Before that, the work performed was mainly related to Membership and planning Registrations. From February onwards, the multiple steps to get to a functional Registrations solution were implemented.

During all these months, other smaller tasks were performed, like code reviews and minor bug fixing. Also, after implementing certain functionalities, tests for those functionalities were added to the respective test suites, which is not always represented in the chart.

## 1.2 Motivation

This project was selected as it is a web development problem that involves thinking of how to integrate a new functionality with an already existing working application. Not only that, interesting technologies would be used, like Angular. Then the cloud was to be used for the database and file storage. Finally, the reactive programming paradigm was to be followed. All these aspects of the project were a big match with the author's interests.

### 1.2.1 The problem and its importance

Acro Companion wanted to develop a membership and registration solution that would allow clubs and federations to register their members and teams for national and international competitions, as well as manage their memberships. Clubs and federations will be called "applying organizations" from here onwards, as they are the organizations that apply for competitions.

To achieve this result, Acro Companion's web application had to be expanded. Not only that, the new Membership and Registrations solution would have to work in synergy with the existing competition management platform.

As mentioned, this is a web development problem. To be able to solve it, and implement the solution, one had to, first, learn the startup's workflow, practices, and technologies used, to then apply the principles learned and the knowledge gathered during its academic path, to expand Acro Companion's web application with these new functionalities.

For Acro Companion, Membership and Registrations is a core functionality that had to be implemented. It was essential for their product as it was the last step for creating a complete solution for gymnastics. Acro Companion already offered a way for users to create exercises and correct them, manage national and international competitions, judge, and score them. Even more, it provided, as already mentioned, live stream integration. With Membership and Registrations, Acro Companion will, on top of what it already offered, allow applying organizations to register their members and teams for competitions in a way that is more convenient for them. Not only that, the competitions' data will be associated with the members that participated in the competition, which gives Acro Companion the possibility to use that data in a creative way and offer even more functionalities in the future.

A complete solution adds tremendous value to all kinds of users (coaches, applying organizations' managers, and competition's organizers) since their workflows can be performed on a single site: Acro Companion's web application. Not having to hop from one platform to another is more convenient and saves time. Rather than using one platform for managing memberships, a different platform for scoring exercises, and yet another platform for managing a competition, and then having to export all the data and integrate everything externally, Acro Companion's clients will now be able to use a single platform to do all that, without having to integrate the data manually.

A client that already used and liked Acro Companion's solution for creating exercises and needs a

way to manage its memberships and register its members and teams in competitions will very likely use Acro Companion's new solution as well, which further accelerates Acro Companion's growth and recognition. Therefore, a complete solution adds tremendous value to Acro Companion as well.

## 1.3 Goals and contributions

There were two major goals of this project:

- to create a Minimum Viable Product (MVP) for Registrations that uses the data from Membership, allowing applying organizations' managers to register members and teams in competitions. The specifics of what exactly should be possible to do within Registrations will be presented later in section 4.1.1.
- to extend Membership in a way, that it can be used together with Registrations. This was a more abstract goal, as it would depend on what would be necessary for Registrations to work.

The main technological contributions of this project for Acro Companion are listed below:

- Creation of a new collection in the database and respective access rules to support Registrations, using Cloud Firestore;
- Creation of new Angular Services to support all the use cases defined for Registrations, using Angular and RxJS;
- Creation of new Angular Components and respective templates to support all the use cases defined for Registrations, using Angular and RxJS.

### 1.3.1 The solution

The technologies that were used for this project's development were the technologies already used by Acro Companion.

This project's implementation was written in Typescript, following the reactive programming paradigm. As a web application, Acro Companion's product has a GUI (Graphical User Interface), which is how the users interact with the program. The GUI needs to react to unpredictable events. As such, the application's control flow is driven by those events. The reactive programming paradigm allows the programmer to not worry about the order in which those events happen or the computation dependencies that are triggered by them, as it provides abstractions that handle these problems [2]. Specifically, the RxJS library from Angular was used for this purpose.

Angular is the web development framework that Acro Companion uses to develop the web application. When it comes to persistence, Cloud Firestore was used as a NoSQL database, and Cloud Storage to store files. For testing, Playwright and Firestore's emulators, as well as Jest were used. Some other tools that were useful for this project were npm for library management, Azure DevOps for continuous integration, testing, and delivery, and Chrome DevTools for debugging. Adobe XD was used to create prototypes for the GUI.

When it comes to the development methodology, an Agile approach was followed.



## 1.4 Structure of the document

The rest of this document is organized as follows:

- In chapter 2, a previous solution for registrations that was developed by Acro Companion will be presented. Finally, an analysis of how registrations are done in other areas will be performed.
- In chapter 3, the technologies and general background will be presented. The purpose of frontend, DevOps, and cloud technologies will be discussed, as well as the technologies used by Acro Companion. Acro Companion's background will be described. Namely, how the operation of the application looked like before this project's implementation, how Acro Companion's web application architecture is organized, what the domain of the application is, and what use cases were already implemented at the start.
- In chapter 4, the project's implementation will be presented. What was done when it comes to planning the new solution for registrations: the definition of new use cases, the definition of the data model, flow between the different statuses of a registration, and GUI prototypes. And finally, the implementation of the defined new use cases, and also the implementation of other things not directly related to those use cases.
- In chapter 5, a global overview of what was implemented and what still has to be done to close the circle for Acro Companion's goal of having an end-to-end solution will be analyzed. Multiple improvements to the current registrations solution will be suggested. Lastly, some final thoughts on this project's impact will be shared.

# Chapter 2

## Related Work

### 2.1 Acro Companion's previous registrations solution

Acro Companion has already developed a registrations solution in the past, as something that a client had requested for a specific competition. However, that solution had to be quickly created, and it was hardcoded for that competition in particular. As it would be too complex to adapt the old solution to a generic one, and there were a few things that Acro Companion wanted to do differently, it was decided that it would be better to start from scratch with a simpler, generic solution, that could later be expanded.

Despite being somewhat hardcoded, looking at the previous solution was important for understanding what had to be done. However, this previous solution was only explored from a user's point of view and not from a developer's point of view. The reason for this is that it was important to get a feeling of what should be possible to do, without the interference of how it had been implemented. This is important because the new solution was to be implemented from zero.

In the new Registrations solution, the applying organization's managers will be the ones creating the registrations, instead of Acro Companion, which makes this new solution more scalable.

### 2.2 Other gymnastics solutions

Worldwide there are multiple solutions that handle memberships, one example is Gymfed, a Belgian gymnastics federation, which has its own membership solution. There are also multiple solutions for scoring, for example, SmartScoring is a company that also offers a scoring solution for gymnastics. However, these solutions are not integrated with other services to provide a full solution for gymnastics.

### 2.3 Other registrations solutions

Eventsport is a Portuguese company specializing in athletics events organization. One of its services is Registration Management [3].

The process to create a registration in an event using Eventsport is the following:

- On the home page, the user selects the event in which he intends to register;
- Once the event is selected, the user is redirected to the event's page, in which the following information is displayed:

- A brief event description;
  - Regulation information;
  - Possibly a video to promote the event;
  - Location;
  - Sponsors.
- The user clicks a button to register himself in the event;
  - Once clicked, the user is redirected to a registration page, where he has to enter the following data:
    - Personal Info:
      - \* Name;
      - \* Gender;
      - \* ID card number;
      - \* Birthdate;
      - \* T-shirt size;
      - \* Whether the user is a federated athlete or not.
    - Team Info:
      - \* Name.
    - Other Information:
      - \* Email;
      - \* Phone number;
      - \* Place of residence;
      - \* Country;
      - \* Address.
  - The user then consents to the data policy, accepts the regulation for the event, and defines a password for his personal area for the specific event he is registering to;
  - If there are multiple races/other events inside the same event, the user selects the one he intends to participate in;
  - The user can repeat this process multiple times to add more participants (without having to redefine the password);
  - The last step is to select the payment method and finish the registration.

Some key differences between Eventsport's registrations service and the solution that Acro Companion intends to develop are:

- Acro Companion's registrations will be done per club or federation. There won't be the possibility for the users to register individually;

- Acro Companion doesn't have a personal area per competition. Instead the applying organization's manager logs in the application once, and can register to multiple competitions once logged in;
- Acro Companion will use a membership solution for member/team creation. This means that the manager will enter the data for each member/team in Membership. And then can register those members/teams to multiple competitions by simply selecting them, rather than having to enter their information each time a registration is being created;
- Acro Companion's solution won't involve payments, at least initially. Payments will be handled between the competition organizers and the participants without Acro Companion's involvement.

The following table highlights a few points where Acro Companion's solution is better:

	Eventsport	Acro Companion
Password definition for personal area	per event	once
Member/team information saved for future registrations	no	yes
Necessity to enter the same data more than once	yes	no
Time consuming	yes	no
Input validation	partial	total
Possibility to edit registration after creation	no	yes



# Chapter 3

## Technologies and Background

### 3.1 Technologies

#### 3.1.1 Frontend Technologies

There are multiple frontend technologies, like HTML, CSS, and JavaScript, for example. They are used to develop the part of an application that a user will interact with directly. HTML defines the structure, CSS defines the visual appearance, and JavaScript defines the behavior of a website.

The frontend technologies used in this project are Typescript and Angular, presented later in this chapter.

#### 3.1.2 DevOps Technologies

DevOps Technologies help organizations delivering applications and services more rapidly than traditional organizations that use traditional software development and infrastructure management processes [4]. These technologies allow formerly isolated roles to coordinate and collaborate to produce better, more reliable products. They allow organizations to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster [5].

The DevOps technology used in this project is Azure DevOps, presented later in this chapter.

#### 3.1.3 Cloud Technologies

Cloud technologies allow organizations to use scalable IT resources over the internet, on demand. These organizations no longer need to own and manage their infrastructure with physical data centers and servers. Instead they can access computing power, storage, and databases from a cloud provider on a pay-as-you-go model and save lots of time [6] [7].

The Cloud technology used in this project is Firebase, presented later in this chapter.

In the following sections, the specific technologies that Acro Companion uses will be presented.

#### 3.1.4 NPM

Node Package Manager, as the name suggests, is the package manager for Node.js. It organizes the modules for Node.js to find and manages any dependency conflicts that might occur. Created in 2009 as an open-source project, npm aimed to help JavaScript developers to share package modules of code

among them. These packages can be found in a public collection, npm Registry. They are published and installed through the npm command-line client [8][9].

### 3.1.5 Chrome DevTools

Chrome DevTools is a set of tools that are directly built into the Google Chrome browser, used by web developers to evaluate and debug their applications. These tools allow developers to, among many other things, view logged messages, run Javascript code, inspect network activity (see if resources are being uploaded or downloaded, and look into the properties of those resources), analyze the application's performance, and evaluate memory usage [10].

### 3.1.6 Typescript

Typescript is a programming language, and at the same time, a static type checker for Javascript, which means that it detects type errors in Javascript code without running it [11].

Created by Microsoft, its purpose is to address the problems that Javascript has that become overwhelming when it is used in large applications [12]. Javascript was originally intended to write small programs that handled simple interactions in web pages, and not having types in the code would not be a problem as ease of reading was important, and the complexity of the code was low. But today, Javascript is used to create complex applications, and adding types makes the development of these applications easier and manageable [13].

Typescript is a superset of Javascript, extending it with classes, interfaces, a module system, lambda expressions, and a static type system, therefore all programs written in Javascript are also valid Typescript programs [12][14]. Typescript infers the types for variables, therefore detecting incorrect behavior in the code, lowering the chances of having bugs. However, it is not able to infer all types for all expressions, so it allows developers to define their own types as well. In practice, Typescript code is transpiled to Javascript [12][14].

### 3.1.7 Angular

Angular is a popular web development framework that is maintained by Google. It is based on Typescript and allows web developers to create scalable applications [15].

Angular web applications' base units are Typescript classes with a `@Component()` decorator, the components. These components have an HTML template associated with them, and also a style definition file. Components facilitate structuring the application and offer good encapsulation [15].

Angular allows developers to insert values from a component into its template dynamically using Property Binding and Interpolation, and also to respond to events detected in the DOM (Document Object Model), using Event Binding. Besides, developers can also modify the structure of the DOM in a dynamic way, using Directives [15].

Angular takes care of instantiating the dependencies of a class with a design pattern called Dependency Injection. Dependency Injection increases the modularity of a web application [15][16].

While components contain the functionality that is directly related to the correspondent view, other types of specific functionality are the responsibility of another type of classes, called services. Services

can be injected into components, providing their functionality to them. This kind of separation increases reusability (of the services) and modularity [17].

A variety of first-party libraries are available for developers to use in their applications as well as tools that help developers during the application's development and maintenance [15].

Acro Companion decided to use Angular, due to its good integration with Firebase, a technology discussed below. Both Angular and Firebase are maintained by Google, and the `angularfire` library makes a good link between the two.

### 3.1.8 RxJS

RxJS is a reactive programming library for JavaScript. It helps developers write asynchronous and event-based programs. Its main type is `Observable`, an invocable collection of future values or events. It also provides other types that interact with `Observable` and operators like `map`, `reduce`, and `filter`, which when combined help to propagate and react to changes [18].

Acro Companion decided to use RxJS as it makes the process of building event-driven applications easier. In Acro Companion's founder opinion, it also allows developers to create applications with less bugs, which will also be easier to pinpoint. It is also the library used for reactive programming in Angular.

### 3.1.9 Firebase

Firebase is a platform managed by Google. It is used by client-side developers to develop mobile and web applications [19]. Firebase offers a set of useful tools, such as:

- Cloud Storage - to store user-generated content, like photos and videos [20].
- Authentication - that provides an end-to-end identity solution, helping developers integrate a secure authentication system in their apps [20].
- Cloud Firestore - a cloud-hosted NoSQL database, to store, synchronize, and query data at a global scale [21].
- Realtime Database - a cloud-hosted NoSQL database, to store and synchronize data between users in real-time, allowing users to collaborate and access their data from different devices [20].
- Cloud Functions - single-purpose JavaScript functions that can be triggered by other Firebase products, like Authentication, Cloud Firestore, and Realtime Database. These functions are executed in a Node.js environment. Computing resources are automatically scaled to match the usage patterns of the application. Can also be used to keep the application logic private on the server-side [20].

Acro Companion decided to use Firebase, due to its good integration between all the tools mentioned. Firebase is also a technology that is easy to start with, as everything works out of the box. Finally, the Realtime Database was really useful for Acro Companion's scoring system.

### 3.1.10 Azure DevOps

Azure DevOps is a planning, collaboration, development, and deployment platform used by teams of developers and project managers to build applications [22]. Azure DevOps offers multiple services:



- Azure Repos - a set of version control tools, it allows developers to use Git repositories, or alternatively, Team Foundation Version Control, Microsoft's version control system [23].
- Azure Pipelines - combines continuous integration (CI), continuous delivery (CD), and continuous testing (CT) to automatically build, test, and ship software projects [24].
- Azure Boards - a set of tools with native support for Agile, Scrum, and Kanban processes, that are used to manage code projects [25].

Acro Companion decided to use Azure DevOps as the founder was looking for something that had a ticketing solution and disliked the alternatives. Azure DevOps also offers agents to run the pipeline locally.

### 3.1.11 Playwright

Playwright is a Node.js library used for end-to-end testing in web applications. It allows developers to run tests across different browsers, execute tests in parallel, and capture videos and screenshots when tests fail for a better understanding of what did not work. It is also possible to record actions in the browser to auto-generate tests [26] [27].

Acro Companion started using Cypress initially, but the founder disliked the technology and switched to Playwright once it came out.

### 3.1.12 Jest

Jest is a JavaScript testing framework that works out of the box, requiring very little configuration. Tests can be run in parallel, which makes the testing process quicker. Previously failed tests are run first, which further contributes to making the testing process as fast as possible [28].

The main factor that led to Acro Companion using Jest was its speed.

### 3.1.13 Git

Git is a distributed version control system that allows developers to have multiple independent branches to develop new functionalities that can later be merged into the main branch. Developers can work together on those branches, pushing changes to a shared repository, and solving merge conflicts if needed. Previous versions of the code can be recovered if necessary [29] [30].

### 3.1.14 Adobe XD

Adobe XD is a tool used to create realistic prototypes for multiple purposes, like brand design and web design. It allows designers to collaborate even if they are on different platforms. Some of its very useful features are wireframing, which allows the designer to define how the user would flow from one page to the next, and the ability to share links to designs, allowing other people to view and comment on those designs [31].

## 3.2 Background

### 3.2.1 Previous Operation

In Acro Companion's application, organizations or, in other words, clubs, federations, and competitions are created by the developers on request, as the process would be too complex for the user, and the chances of making mistakes are reasonably high. Once created, Acro Companion adds the responsible people for the organization as managers of that organization. Managers can edit the organization details at their convenience through the web application's GUI. The communication between club, federation, and competition's managers happens outside of Acro Companion.

When this project started, Registrations was not an actual feature of Acro Companion's web application. However, the competition's managers were able to import spreadsheets containing the participating gymnasts, teams, and the exercises that were going to be performed in that competition. Once imported, the details could be edited in the GUI. Gymnasts, teams, exercises, and judges could also be created manually in the GUI, instead of doing the import. The way it was done before was not the most convenient, as the data for the gymnasts and teams would have to be entered in the application for each competition separately, consuming a lot of time with redundant work. The new solution aims to be an easier alternative.

There are multiple different disciplines in gymnastics that Acro Companion currently supports in competitions, for example: Acrobatic, Rhythmic, Trampoline, and Artistic. However, the new registrations solution aims to be as generic as possible, since Acro Companion intends to widen the scope of its application to all gymnastics disciplines.

### 3.2.2 Architecture

In figure 3.1, is represented the architecture of Acro Companion's web application. In blue are the technologies that are used by developers, but not by the end-user.

As already mentioned, the code is written in Typescript using the Angular framework. Angular was used to develop the frontend, and for defining the views.

Acro Companion's web application is a serverless application, meaning that the developers in Acro Companion can focus on writing business logic, and do not have to worry about resource provisioning, monitoring, maintenance, scalability, and fault-tolerance when it comes to the backend, because those operational concerns are handled by the cloud provider, in this case, Google Cloud [32]. More precisely, Acro Companion uses the Function-as-a-Service (FaaS) model. Developers write simple stateless functions, and Google Cloud runs them, scaling as needed and handling faults [32]. The angularfire library interacts with the firebase SDK, which is responsible for communicating with the cloud. This is what connects the frontend and the backend.

As figure 3.1 shows, different technologies in the cloud are employed. These technologies, and their purpose, are described in section 3.1.9.

Acro Companion's developers use Chrome DevTools for debugging the web application on the browser, Playwright for end-to-end testing, Jest for unit and integration testing, and Firebase Emulator to avoid manipulating production data while on development. Git is the version control system that is used, and npm is the package manager for installing dependencies. When it comes to DevOps, Acro

Companion resorts to Azure DevOps: Azure boards for creating tickets that are assigned to different developers and for monitoring the work progress, Azure Repos for containing the code repository that developers interact with through git, and Azure Pipelines for Continuous Integration (CI), Continuous Delivery (CD) and Continuous Testing (CT). These technologies were described in section 3.1.

When it comes to the database, Acro Companion uses Cloud Firestore which is a NoSQL database. NoSQL gives the startup lots of flexibility on the data structures used and allows its developers to iterate quicker in the development process since it is not as strict as SQL. NoSQL databases are easier to scale as well, which is also an advantage for Acro Companion. Cloud Firestore is integrated into Firebase, a complete cloud platform that Acro Companion wanted to use.

In the next subsection, the domain model is discussed.

### 3.2.3 Domain and implemented use cases

Figure 3.2 presents the domain model with the concepts that are closely related to Membership and Registrations. Competitions, Clubs, and Federations have one or more managers that are responsible for those Organizations. A Federation is a parent Organization for multiple Clubs, meaning that there are multiple Clubs that belong to the same Federation. A Competition can have as a parent Organization, a Club or a Federation, but it does not have to. Both Clubs and Federations have Teams and Members, which can belong to different member types. One or more of the Club/Federation's Managers is responsible for creating the Registrations for a Competition. Each Registration has either one Club or one Federation that registers multiple Gymnasts, Teams, and Judges in one Competition. Finally, Competitions, Clubs, and Federations have multiple Registrations associated with them.

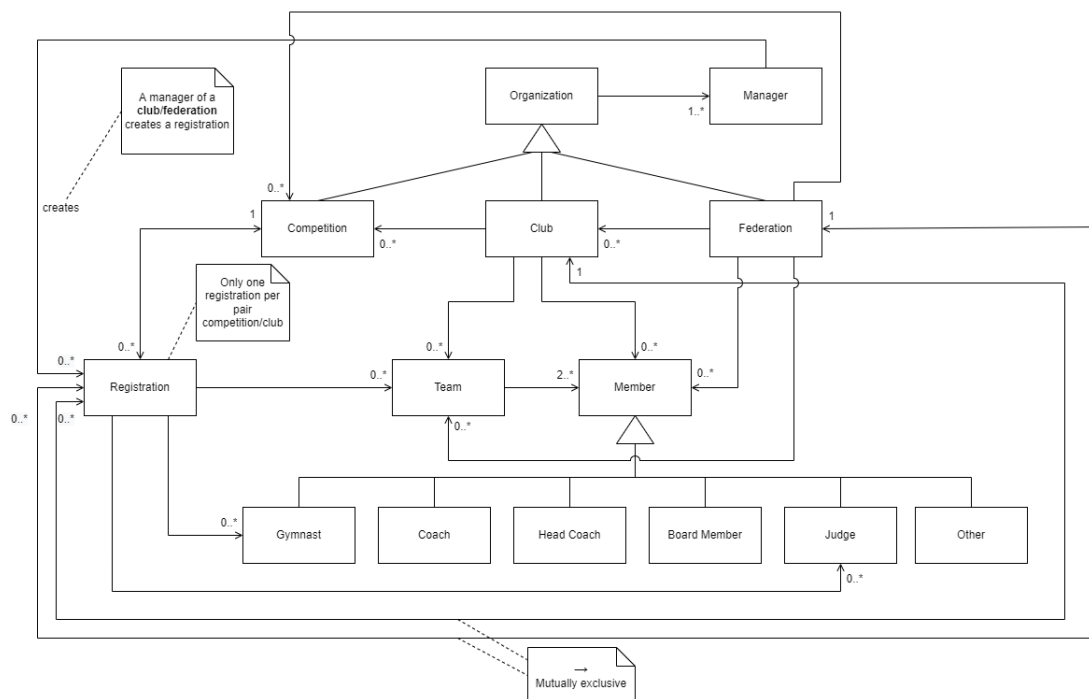


Figure 3.2: Domain model.

When it comes to functionality, multiple use cases were already implemented before the start of this project, as figures 3.3 and 3.4 show.

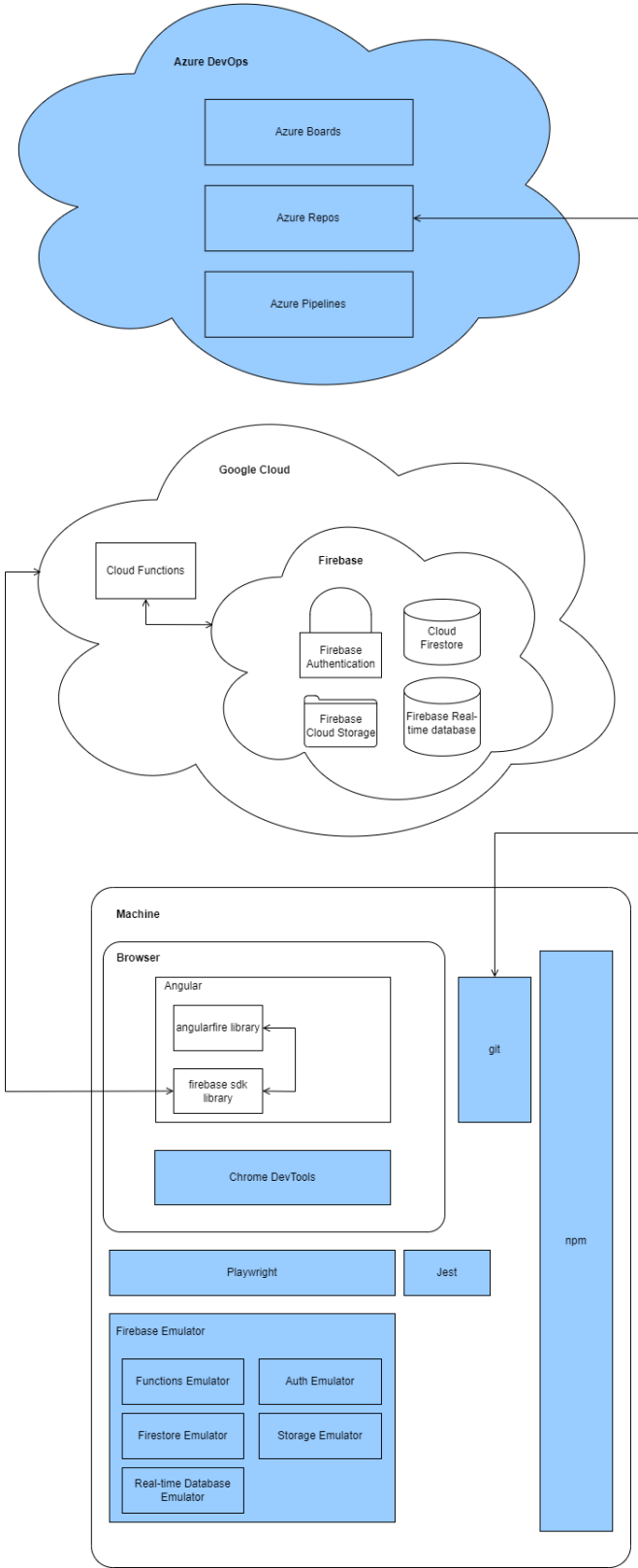


Figure 3.1: Web application's architecture.

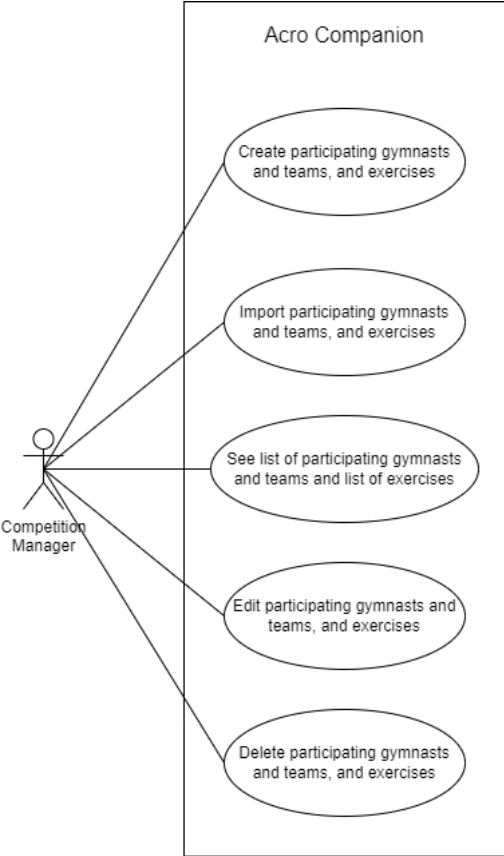


Figure 3.3: Already implemented use cases for competition’s managers.

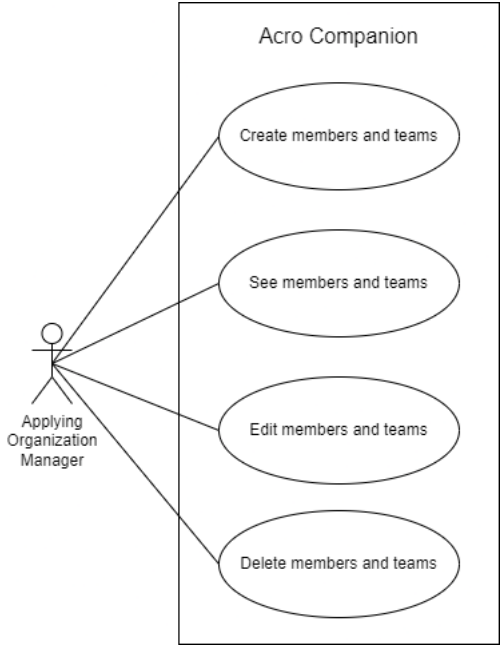



Figure 3.4: Already implemented use cases for applying organization’s managers.

A competition’s manager could create gymnasts and teams that would be participating in the competition, as well as the exercises to be performed. The manager could also import all this information by filling in a spreadsheet with the associated properties.

Create a member

Member



First Name \*  
Mario

Last Name \*  
Creed

Email \*  
mario.creed@gmail.com

Telephone number

Gender \*  
Male

Discipline(s) \*  
Acrobatic, Trampoline

Postal Code

City

Street

House Number

Birth Date \*  
10/1/2003  
Format (DD/MM/YYYY)

Member status \*  
Active

Select Member Type(s)

Gymnast

Coach

Head Coach

Board Member

Other

Judge

Delete Member Cancel Save


Figure 3.5: Original dialog for creating a member.

Create a team


Team Name \*  
Matt and Oliver

Category \*  
Mp

Level \*  
12-18




Top



Oliver Davidson  
Birth date: 10/01/2000

Base 1



Matt Davidson  
Birth date: 10/01/1999

Delete Team Cancel Save

Figure 3.6: Original dialog for creating a team.

After creating or importing the information, the manager could see the list of participating gymnasts or teams, as well as the list of exercises, by navigating to the respective tabs "Gymnasts", "Teams", and "Exercises".

For each row containing a gymnast, a team, or an exercise, the manager would be able to edit the information for each column, therefore editing the respective gymnast, team, or exercise.

Finally, by selecting one or more rows, the manager could delete the respective instances.

An applying organization's manager could create, as well as edit or delete members and teams for its organization. Figure 3.5 shows the dialog for creating or editing members, and figure 3.6 shows the dialog for creating or editing teams.

In the "Members" tab, the manager would be able to see the members of the applying organization and, in the "Teams" tab, the teams, as displayed in figure 3.7.

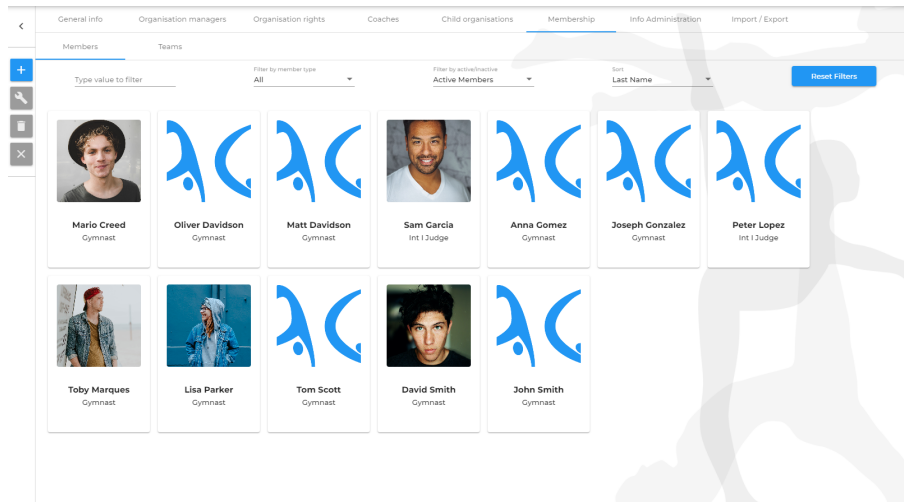


Figure 3.7: Original view of the members.

As previously mentioned, the creation of competitions, and organizations in general is actually done by Acro Companion, for that reason, it is not represented as a use case.

## Chapter 4

# Registrations Platform

### 4.1 Planning

Before the start of the implementation of the new Membership and Registrations solution, there was some planning that was done. One very important first step was understanding what Acro Companion desired to achieve. For that purpose, the communication between team members was done mostly through a shared google docs file, where questions and answers were logged. Small meetings were also scheduled whenever necessary.

Regarding this project, the first questions asked were more domain-related and not related as much to implementation, since first was important to understand the concepts and how they were related.

#### 4.1.1 New Use Cases

One of the first things to be discussed were the use cases that defined the Minimum Viable Product (MVP) for the new Registrations solution.

In figure 4.1, are represented the use cases for a manager of an applying organization. A manager should be able to search for competitions open for registrations, register Gymnasts, Teams, and Judges in a competition, edit a registration, and cancel a registration. The two use cases in red were not originally in the plan but were considered necessary during the implementation phase.



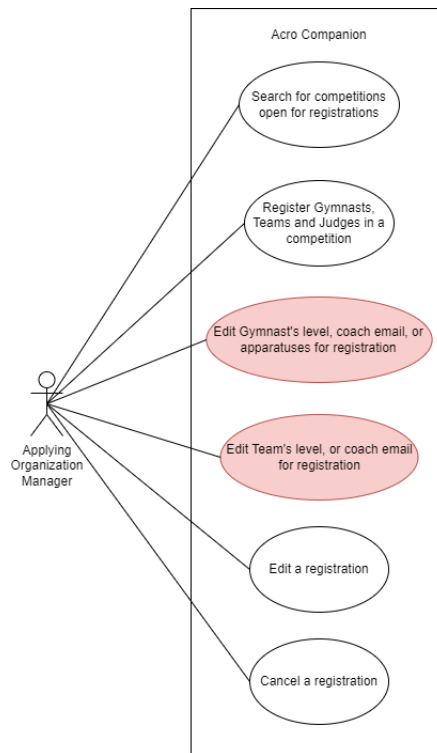


Figure 4.1: MVP's use cases for applying organizations.

In figure 4.2, are represented the use cases for a manager of a competition. A manager should be able to set a competition open for registrations, approve or reject a registration, import the participating gymnasts, teams and judges, and see them listed, as well as the exercises.

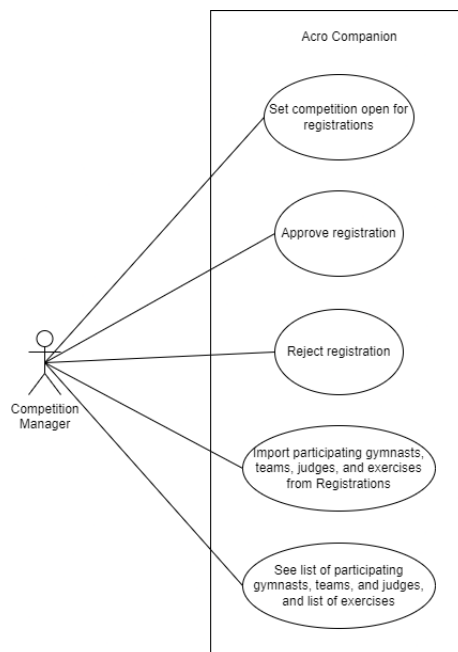


Figure 4.2: MVP's use cases for competitions.

There are a few important points to emphasize here:

- From the applying organization's side, the use cases regarding Membership were already imple-

mented, as mentioned in section 3.2.3. The use cases defined for the applying organization's manager in this section refer to Registrations.

- From the competition's side, the new use cases for importing and seeing the participants and the exercises do not replace the previous similar use cases. This import should be possible to do from Registrations, and consequently, the manager should be able to see the imported data in the same place as before.

### 4.1.2 Data Model

With the use cases defined, the next phases were the design of the solution, and then, the implementation.

Regarding Membership, the model had to be expanded. In particular, gymnast-specific properties had to be defined, as well as a property specific for judges. These new properties are the ones that the Swiss Gymnastics Federation uses in its current solution. Since the Swiss Gymnastics Federation is the main client that Acro Companion has for the new Membership and Registrations solution, and since the properties indicated by its representatives were considered generic enough and easy to adapt that they would apply to future clients as well, it was decided that these were good properties to start with.

Initially, the team creation process was very specific for the Acrobatic discipline. Acro Companion intends to make this process generic to all gymnastics disciplines that involve teams, in the future. As the number of members in a team varies, even for a single discipline, a few extra properties were temporarily added to allow the addition of up to eight members to a team. This is bad design, and in the future, the Team's model will have an array of tuples containing the member and its respective role in the team. The role is currently hardcoded in the property name ('top', and 'base'), and is still very specific to Acrobatics. In the meantime, this temporary solution already allows the creation of teams for non-acrobatic disciplines.

There are also properties for storing previous values for gymnasts, teams, and judges. These properties were only added in the implementation phase, as they were necessary for a competition's manager to be able to reject changes to a registration, rolling the registration back to its previous state.

When a Gymnast participates in a Competition, he might compete in a different level than the level that he usually competes in. He might also have a different coach responsible for him for that specific Competition. The same applies to Teams.

The terms "exercise type" and "apparatus" will be used as synonyms.

When it comes to Registrations, there are two parts involved. Clubs and federations have to be able to create and edit registrations. However, the competition's managers also have to access it so that they can see and approve or reject the submitted registrations.

In section 4.3, the implementation of each use case will be presented, as well as the properties and methods that are used for each of those use cases.

### 4.1.3 Registrations Status Flow

One of the most important steps during the design of a solution for Registrations was understanding how a registration would evolve from the moment it is first created until the moment it is approved.

When a registration is created, it can be saved as a draft, or immediately sent in for approval. Registrations can be saved as drafts, for example, if the applying organization's manager has not yet decided on who to include in the registration. Registrations with status 'draft' will not be visible for the competition's manager, while registrations that were sent in will.

#### 4.1.4 GUI Prototypes

After defining both the data model, and the class diagram, and understanding how a registration would evolve from one status to another, one relevant aspect that still had to be defined was how all of this information was going to be displayed on the GUI, how would the user experience the Registrations solution. For this purpose, some mockups were created with Adobe XD. The mockups were created as a way for developers to be able to visualize the Registrations solution, to discuss how the information should be presented, and how should the user flow from one step to the next. All of this, prior to the implementation phase. The goal of these mockups was never to create the final design for Registrations, but to facilitate the discussion about how the solution would be implemented, therefore they were kept as simple as possible in order to allow a rapid transition into the implementation phase. That does not mean that the design of the solution was completely ignored, but rather that the focus was on how to integrate the different functionalities. Acro Companion, usually implements a new feature without thinking too much about its final design. Once the feature is completely implemented, Acro Companion has a designer that is responsible for improving the UI.

With the GUI prototypes created, the implementation phase started. The technologies described in section 3.1 were used, and tests were developed during the process to ensure that everything worked as expected.

## 4.2 Methodology

### 4.2.1 Functional requirements

The functional requirements in this project were the same as the use cases already presented:

- It should be possible for an applying organization's manager to:
  - Search for competitions open for registrations.
  - Register Gymnasts, Teams, and Judges in a competition.
  - Edit a Gymnast's level, coach email, or apparatuses for a registration.
  - Edit a Team's level, or coach email for a registration.
  - Edit a registration.
  - Cancel a registration.
- It should be possible for a competition's manager to:
  - Set a competition open for registrations.
  - Approve a registration.
  - Reject a registration.

- Import participating gymnasts, teams, judges, and exercises from Registrations.
- See list of participating gymnasts, teams, and judges, and list of exercises.

## 4.2.2 Non-functional requirements

During the planning phase, some non-functional requirements were also discussed, however, this discussion was very informal, and no strict requirements were established. The reason for this is that the application is serverless, as discussed, therefore a lot of the areas of concern where these requirements would have been defined are actually handled by the cloud provider.

Nevertheless, the following requirements were kept in mind during the implementation phase:

- Membership and Registrations should work on mobile devices;
- Users should never see a frozen screen. The screen should always have an indication that something is happening, for example, a loading animation;
- The queries to the database should be reduced to a minimum. If some information was already fetched, it should not be fetched again;
- The queries to the database should be limited to the least scope possible;
- Only the correct users should have access to documents in the database. Database access rules should be written for every database collection.

In annex [1] is displayed everything that was done during the project. Both the functional and non-functional requirements were an important guide for all the tasks performed.

## 4.3 Implementation

In this section, the implementation of each use case will be presented.

Before presenting each use case's implementation, there is a small, yet important modification that was made in comparison to what was originally planned. When comparing the dialog for creating and editing registrations that was designed for the GUI prototypes and the one that was then implemented, there are significantly more steps in the latter one, as displayed in figures 4.3 and 4.4.



Figure 4.3: Stepper on the GUI.



Figure 4.4: Implemented stepper.

It was decided that the selection of members and teams should be done in separate steps so that there was not too much information being displayed on the screen at the same time. Not only that but gymnasts and judges should be selected separately as well.

Then, as mentioned in section 4.1.1, there were two use cases that were considered necessary during the implementation phase, which will be discussed later, that are handled in the steps "Customize Gymnasts" and "Customize Teams".

Finally, it was considered relevant to present a summary at the end of the registration creation process, so that the manager could review the information introduced in one single step, without having to navigate back and forth, as now there were more steps.

The first use cases to be presented are performed by the applying organization's manager and require that the manager opens a dialog to edit or create a registration.

Figure 4.5 shows the GUI prototype for the Registrations' view.

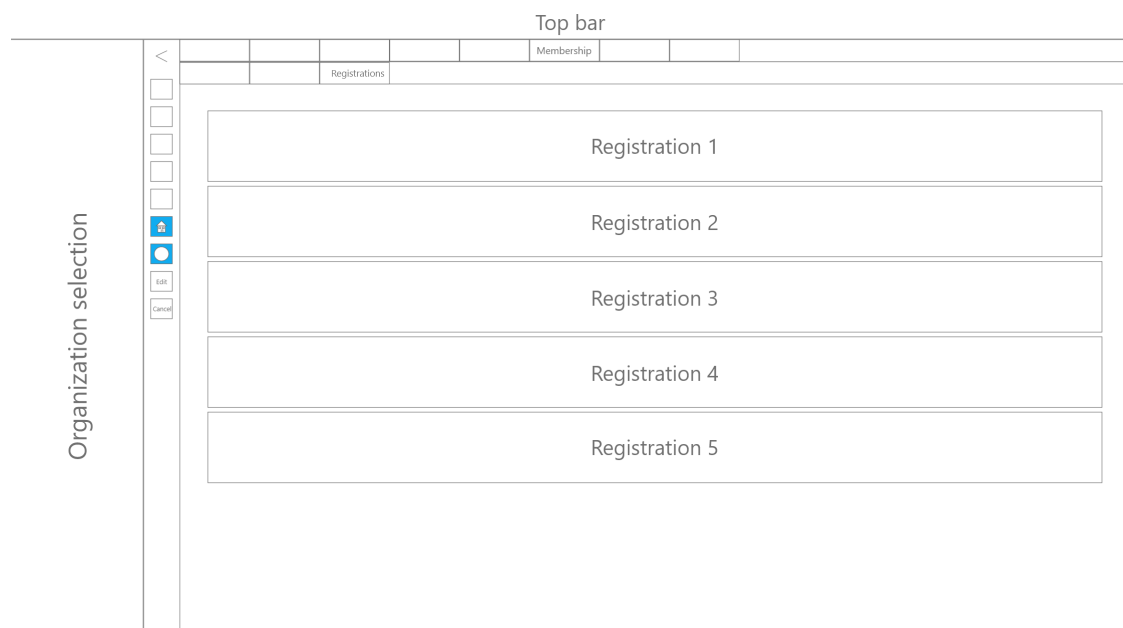


Figure 4.5: Registrations' view prototype.

Figure 4.6 shows the implementation for the Registrations' view.

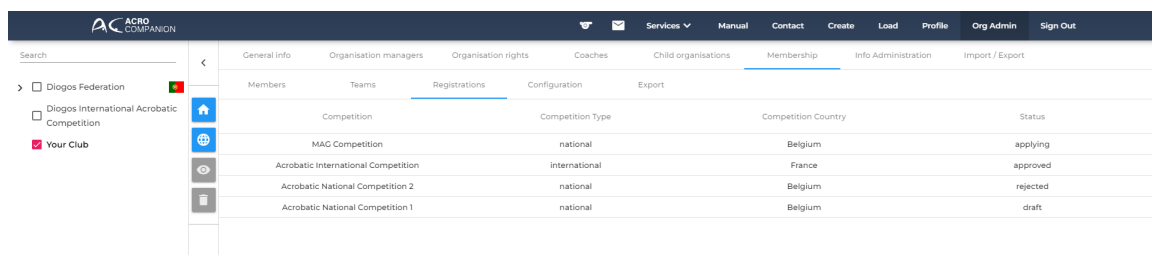


Figure 4.6: Registrations' view implementation.

### 4.3.1 Searching for competitions open for registrations

The first step when creating a new registration, is to select the competition the manager wants to register to. For that purpose, he will need to search for competitions that are open for registrations. In figures 4.7 and 4.8, are presented the GUI prototype for competition selection, and the final implementation, respectively.

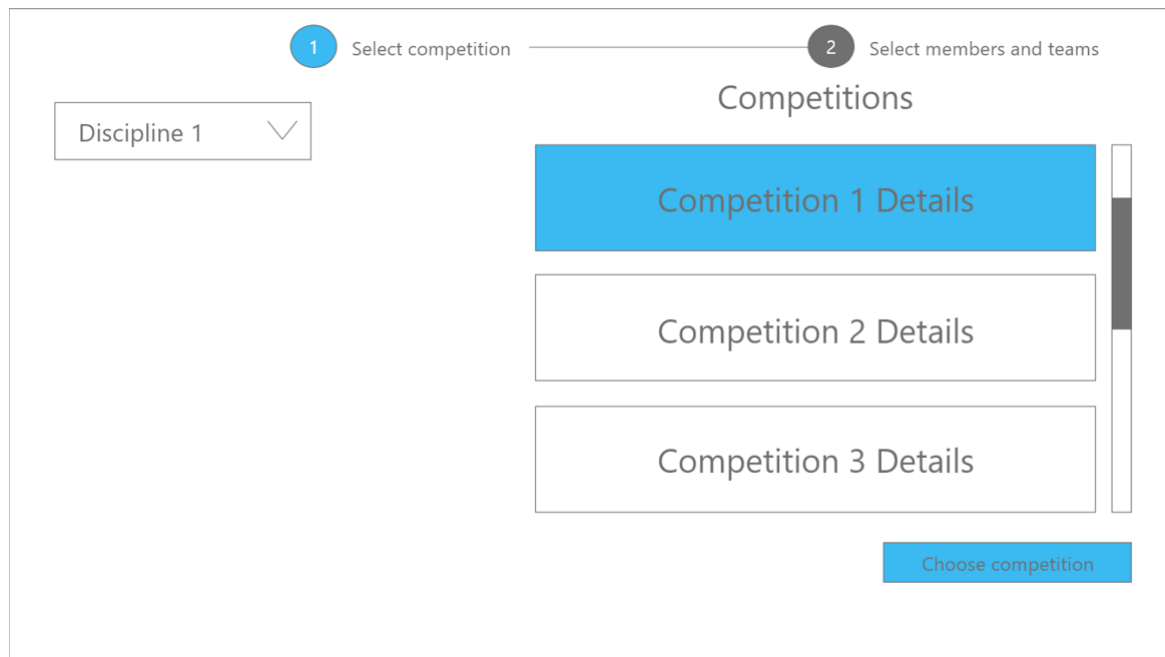


Figure 4.7: GUI Prototype for Competition Selection

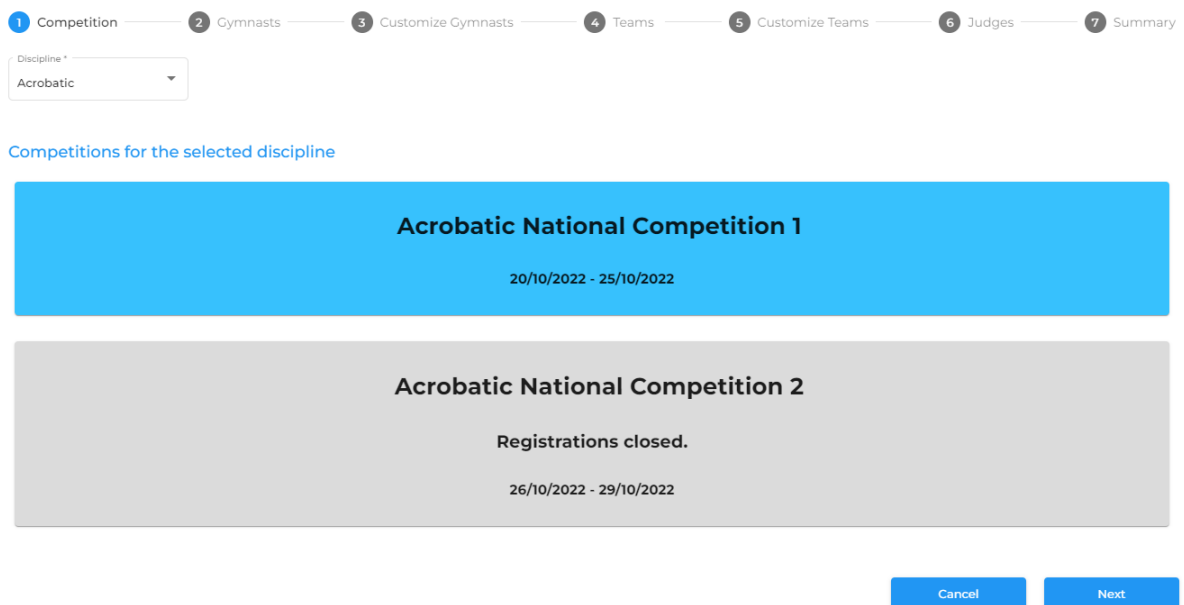


Figure 4.8: Implementation of Competition Selection

A competition will not be selectable in only two scenarios:

- the competition is not open for registrations yet;
- the applying organization is already registered in the competition.

### 4.3.2 Registering Gymnasts, Teams, and Judges in a competition

The following steps are the selection of the members, teams, and judges. As it was mentioned at the beginning of this section, this is done in multiple steps. In figures 4.9, 4.10, 4.11 and 4.12, are presented the GUI prototypes for members, teams, and judges selection, and the final implementations.

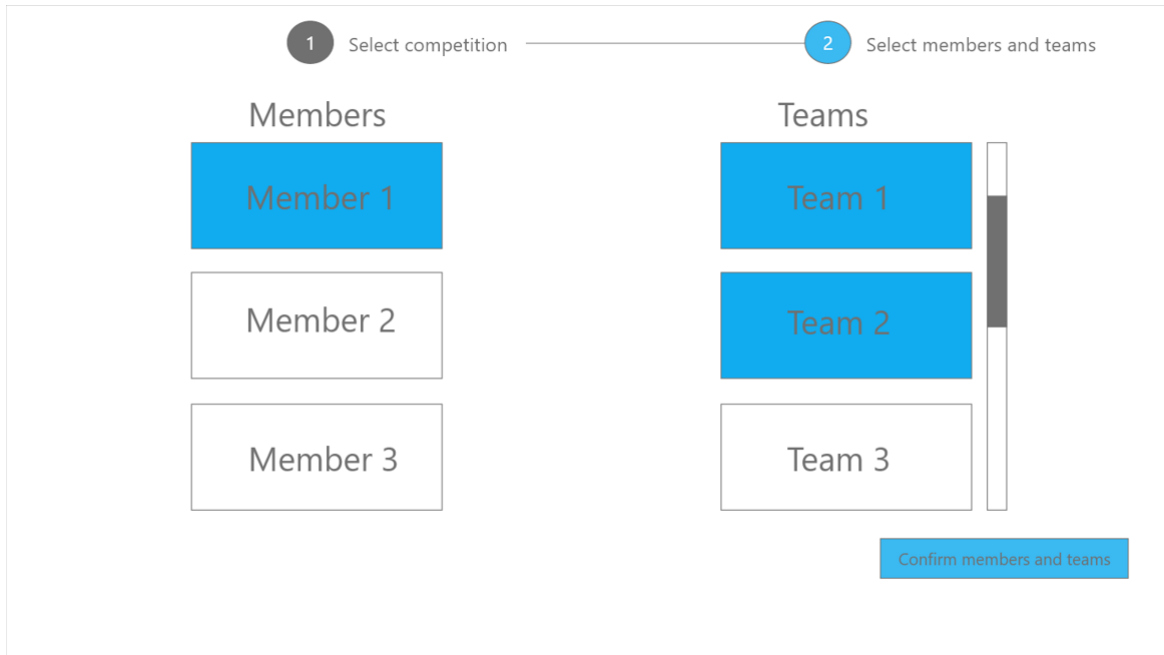


Figure 4.9: GUI Prototype for member and team selection.

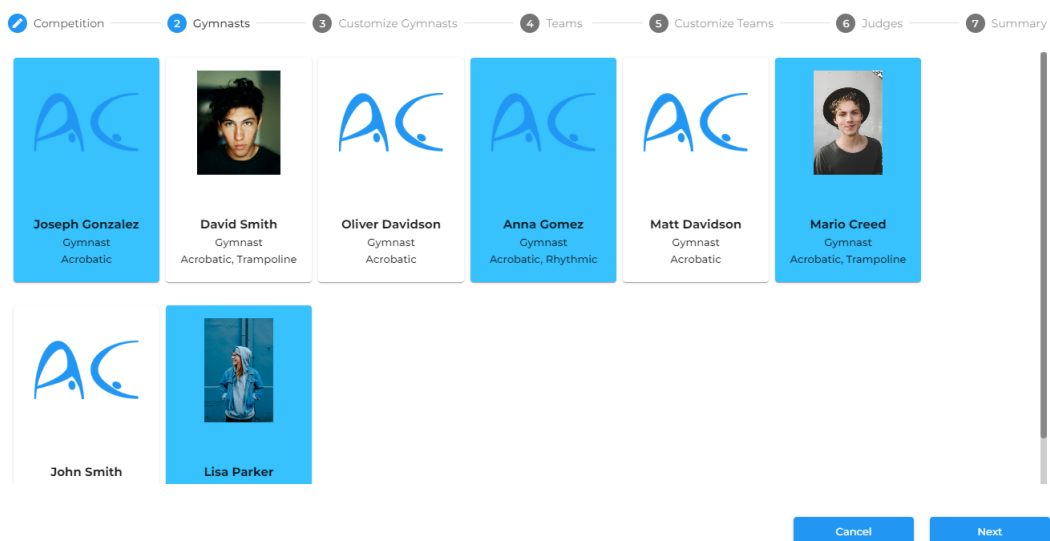


Figure 4.10: Gymnast selection implementation.

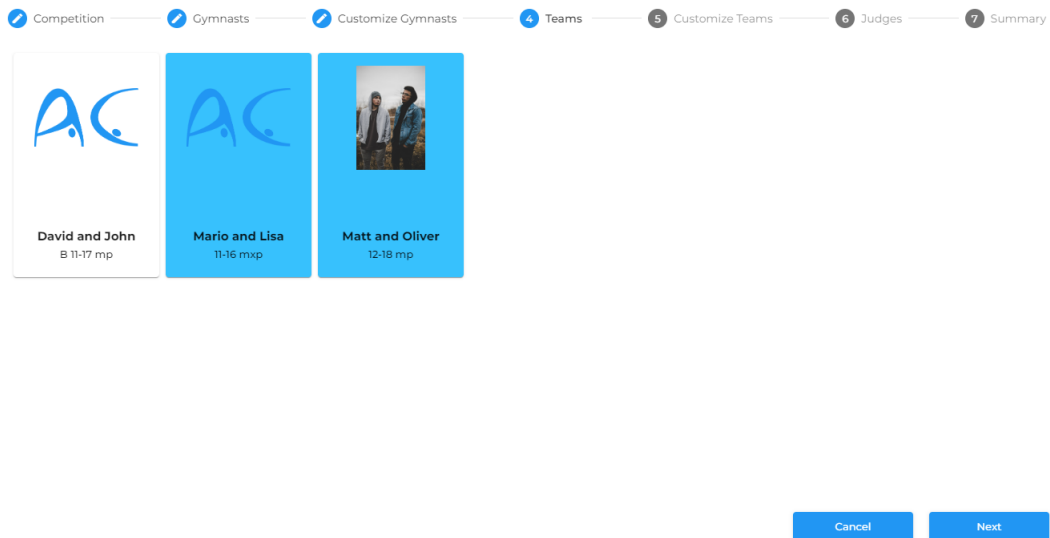


Figure 4.11: Team selection implementation.

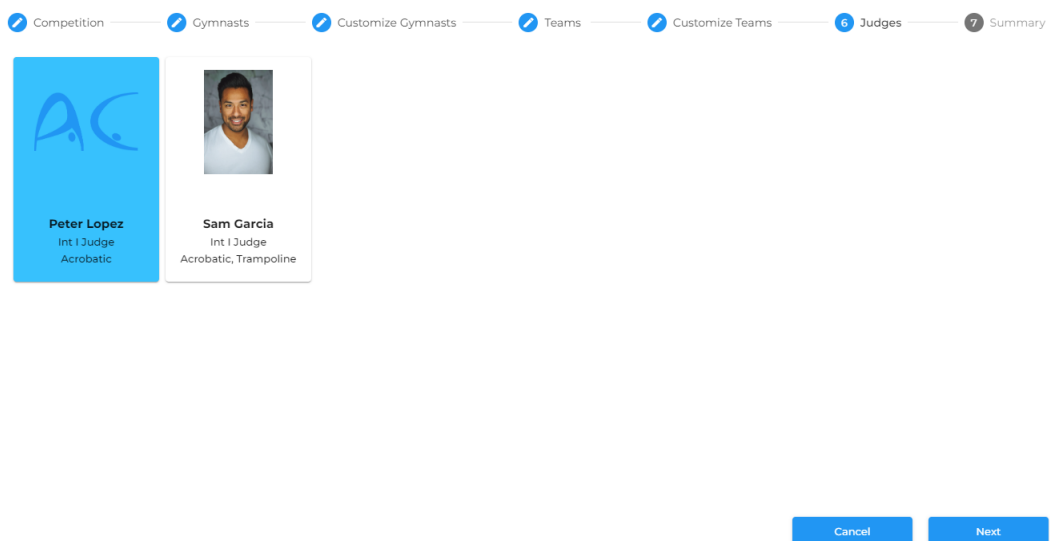


Figure 4.12: Judge selection implementation.

Committing to a competition, means that the manager has not only selected a competition, but also moved to the next step in the dialog. From that point onwards, the manager will not be able to edit the registration's competition, and if he wants to register into another competition, he will have to create a new registration.

The eligible gymnasts, teams, and judges are then listed on the GUI, in the respective steps, for the manager to select the ones he intends to register in the competition.

The manager has to be the manager of the applying organization, which means that from the competition side, the members and teams displayed are not clickable. Also, the registration has to be editable, cancelled registrations for example are no longer editable.



Once the user gets to the final step, the "Summary" step, as displayed in figure 4.13, the selected competition, members, and teams are displayed.

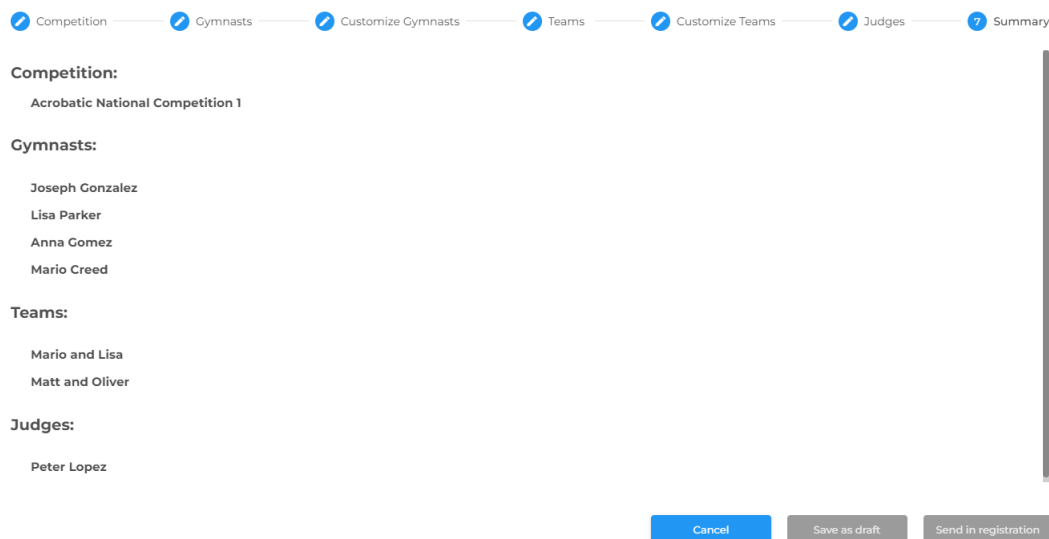


Figure 4.13: Registration summary.

### 4.3.3 Editing Gymnast's level, coach email, or apparatuses for registration

As discussed in section 4.1.2, gymnast's and teams' levels and coaches might have to be customized for a specific competition. For competitions of certain disciplines, the exercises that a gymnast will perform have to be selected in the registration creation process.

In this section the customization of gymnasts will be presented.

Figure 4.14 shows the step for customizing the gymnasts after selecting them. For each gymnast, the name (e.g. Joseph Gonzalez), level (e.g. International rules-11-16), and coach email (e.g. josephs.coach@gmail.com) is displayed.

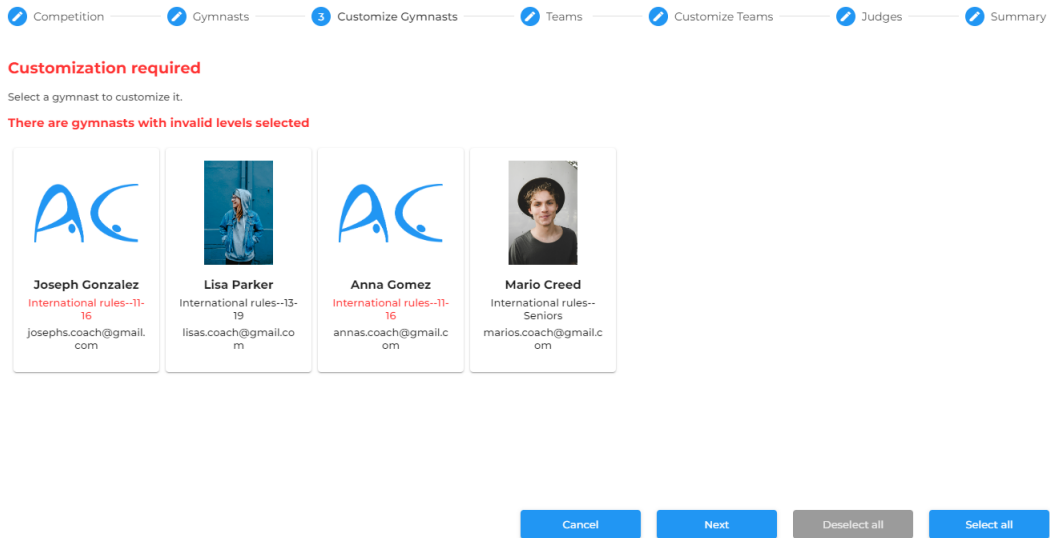


Figure 4.14: Gymnast customization.

At the top there is a main warning stating that customization is required, then a message indicating that a gymnast has to be selected so that it can be customized, and finally a more specific error message stating that gymnasts with invalid levels for the selected competition were selected.

When a gymnast is selected, as displayed in figure 4.15, dropdowns for selecting a level and a coach email for that gymnast appear. The coach email can be changed if the gymnast will have another coach responsible for him for the competition. However, the level has to be changed, as the current level associated to the gymnast is invalid for the selected competition. By clicking the level dropdown, the manager can select a level that is valid for the selected competition. This dropdown will be presented in more detail later.

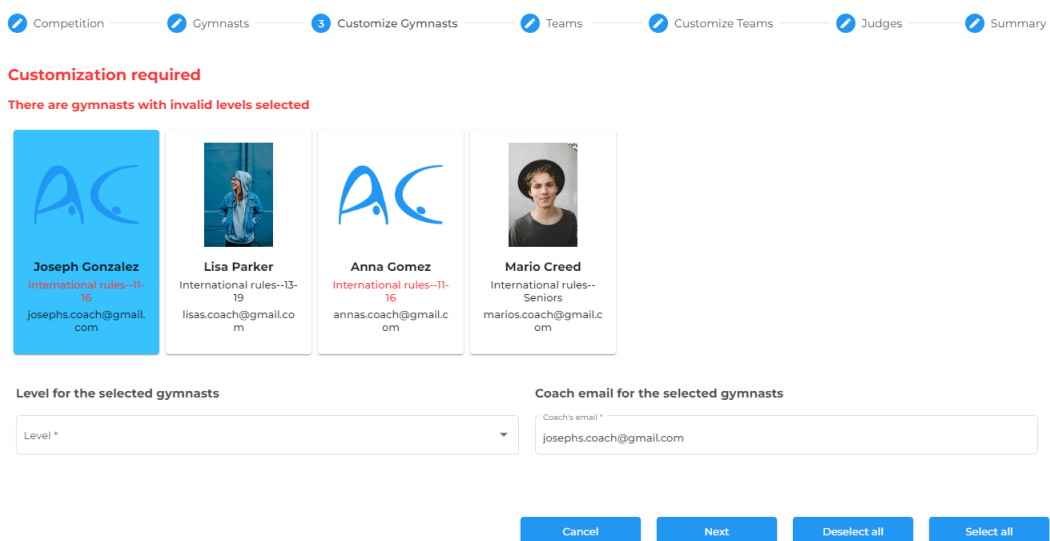


Figure 4.15: Gymnast customization with a member selected.

With a new valid level selected, as displayed in figure 4.16, the gymnast's card no longer displays the level in red.

Competition Gymnasts **3** Customize Gymnasts Teams Customize Teams Judges Summary

**Customization required**  
There are gymnasts with invalid levels selected

<p><b>Joseph Gonzalez</b> International rules--13-19 josephs.coach@gmail.com</p>	<p><b>Lisa Parker</b> International rules--13-19 lisas.coach@gmail.com</p>	<p><b>Anna Gomez</b> International rules--11-16 annas.coach@gmail.com</p>	<p><b>Mario Creed</b> International rules--Seniors marios.coach@gmail.com</p>
--	--	---	---

Level for the selected gymnasts:

Coach email for the selected gymnasts:

Cancel Next Deselect all Select all

Figure 4.16: Level selection in gymnast customization.

Multiple gymnasts can be selected and their levels changed at the same time. By selecting two gymnasts, as displayed in figure 4.17, the dropdowns for the level and coach will not show a value, unless the selected members have the exact same value for that dropdown.

Competition Gymnasts **3** Customize Gymnasts Teams Customize Teams Judges Summary

**Customization required**  
There are gymnasts with invalid levels selected

<p><b>Joseph Gonzalez</b> International rules--13-19 josephs.coach@gmail.com</p>	<p><b>Lisa Parker</b> International rules--13-19 lisas.coach@gmail.com</p>	<p><b>Anna Gomez</b> International rules--11-16 annas.coach@gmail.com</p>	<p><b>Mario Creed</b> International rules--Seniors marios.coach@gmail.com</p>
--	--	---	---

Not all selected gymnasts have the same level, please select one level for them:

Not all selected gymnasts have the same coach email, please write the coach's email for them:

Cancel Next Deselect all Select all

Figure 4.17: Gymnast customization with multiple members selected.

Once a value is selected in one of the dropdowns (or both), that value will be displayed in the gymnast's cards, as displayed in figure 4.18.

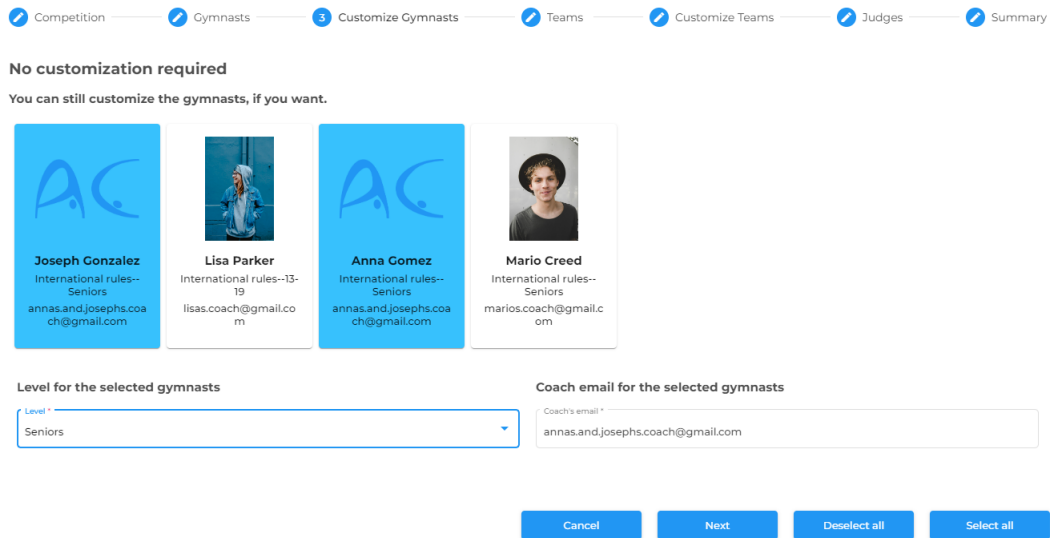


Figure 4.18: Level and coach selection for multiple members in gymnasts customization.

With all invalid levels replaced by valid ones, at the top of the dialog, a "No customization required" message is displayed, and the previous error message is no longer shown.

The way the coach's email is assigned to the gymnasts is very similar.

Finally, for some disciplines, the apparatuses that will be performed have to be selected. Those disciplines are "Women Artistic Gymnastics" and "Men Artistic Gymnastics". Figure 4.19, shows the same gymnast customization step, if the selected competition's discipline was "Men Artistic Gymnastics".

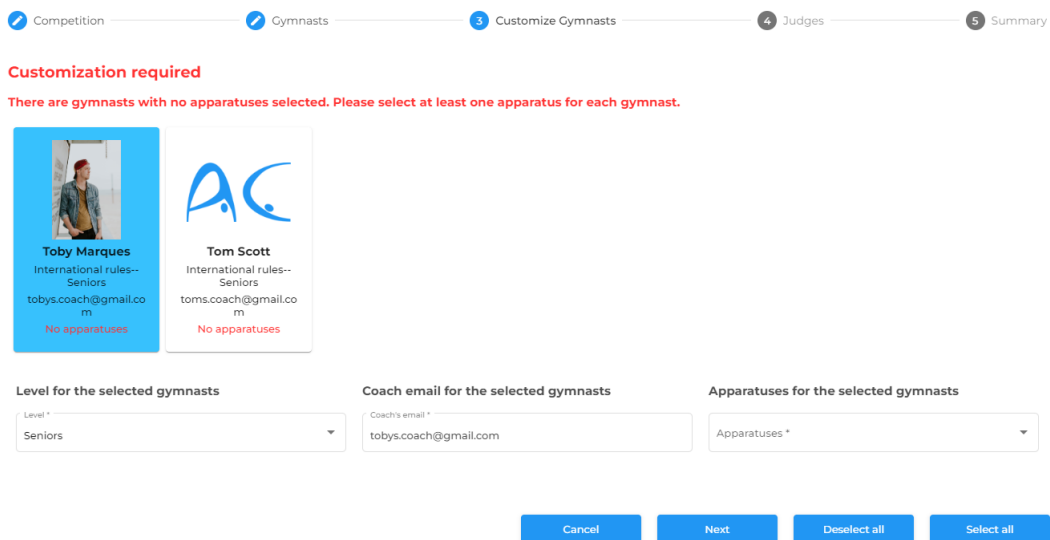


Figure 4.19: Gymnasts customization for discipline with apparatus selection.

In this case, all selected gymnasts have valid levels, however customization is still required since no apparatuses have been selected for the gymnasts, as the messages displayed at the top of the dialog and on each card indicate.

The logic for the apparatuses selection is identical to the logic for the selection of a level or a coach email.

Once the apparatuses have been selected, they are displayed in the gymnast's cards, and no further customization is required assuming the levels were valid, as displayed in figure 4.20.

Progress bar: 1 Competition, 2 Gymnasts, 3 Customize Gymnasts, 4 Judges, 5 Summary

**No customization required**  
You can still customize the gymnasts, if you want.

**Toby Marques**  
International rules--  
Seniors  
tobys.coach@gmail.co  
m  
floor,rings

**Tom Scott**  
International rules--  
Seniors  
toms.coach@gmail.co  
m  
pommel\_horse,vault,p  
arallel\_bars

Level for the selected gymnasts: Seniors

Coach's email for the selected gymnasts: toms.coach@gmail.com

Apparatuses for the selected gymnasts: pommel\_horse, vault, parallel\_bars

Buttons: Cancel, Next, Deselect all, Select all

Figure 4.20: Gymnasts customization for discipline with apparatuses selected.

#### 4.3.4 Editing Team's level, or coach email for registration

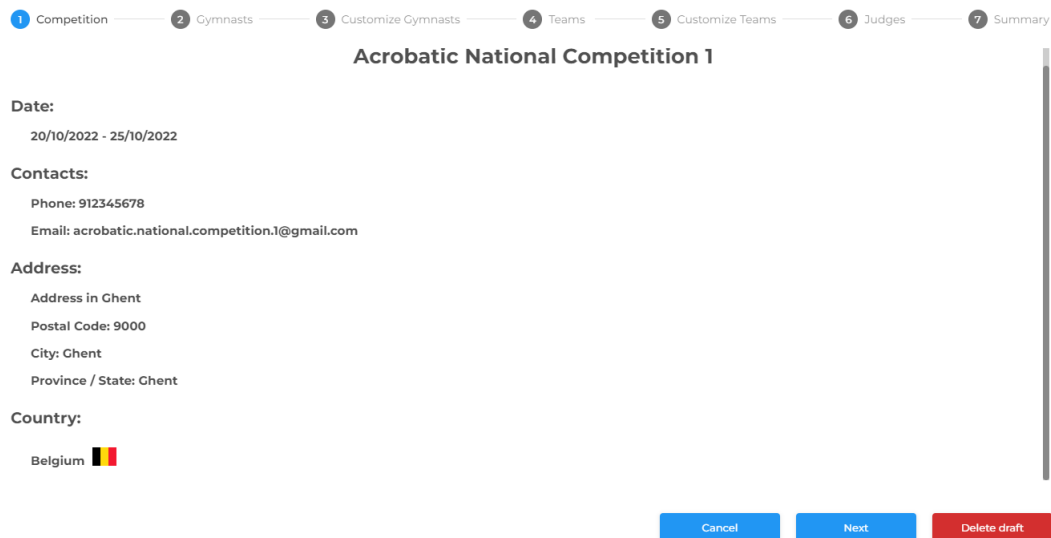
The process for customizing a team's level or coach email is identical to the process of customizing a gymnast's level or coach email.

When customizing a team, no apparatuses have to be selected, since the only disciplines that require apparatuses selection do not involve teams (in figure 4.20, the stepper does not include a step for selecting teams or for customizing selected teams).

#### 4.3.5 Editing a registration

When the applying organization's manager opens the dialog to edit the registration, he can see the details of the competition that was selected, as well as the selected gymnasts, teams, and judges.

Figure 4.21, shows the competitions details.



1 Competition — 2 Gymnasts — 3 Customize Gymnasts — 4 Teams — 5 Customize Teams — 6 Judges — 7 Summary

### Acrobatic National Competition 1

**Date:**  
20/10/2022 - 25/10/2022

**Contacts:**  
Phone: 912345678  
Email: acrobatic.national.competition.1@gmail.com

**Address:**  
Address in Ghent  
Postal Code: 9000  
City: Ghent  
Province / State: Ghent

**Country:**  
Belgium 🇧🇪

Cancel Next Delete draft

Figure 4.21: Selected competition details.

The members and teams are displayed as being selected just like they were when the registration was being created.

As now the competition is being edited, the manager is considered to have committed to a competition as soon as the dialog is opened, as the competition cannot be changed.

To edit the registration, the manager can select and deselect members and teams, and customize them differently. The behavior is the same that was described in the previous sections.

One extra step that happens when editing a registration is that the previous gymnasts, teams, and judges are stored when the registration object is created, so that if the competition manager rejects these changes, the registration can go back to the previously selected members and teams.

### 4.3.6 Cancelling a registration

There are two ways an applying organization's manager can cancel a registration, as mentioned in section 4.1.4.

A manager can select one or more registrations and click the button with the trash can icon in the Registrations' view. Figure 4.22 shows the GUI prototype for cancelling a registration through Registrations' view, while figure 4.23 shows the real implementation.

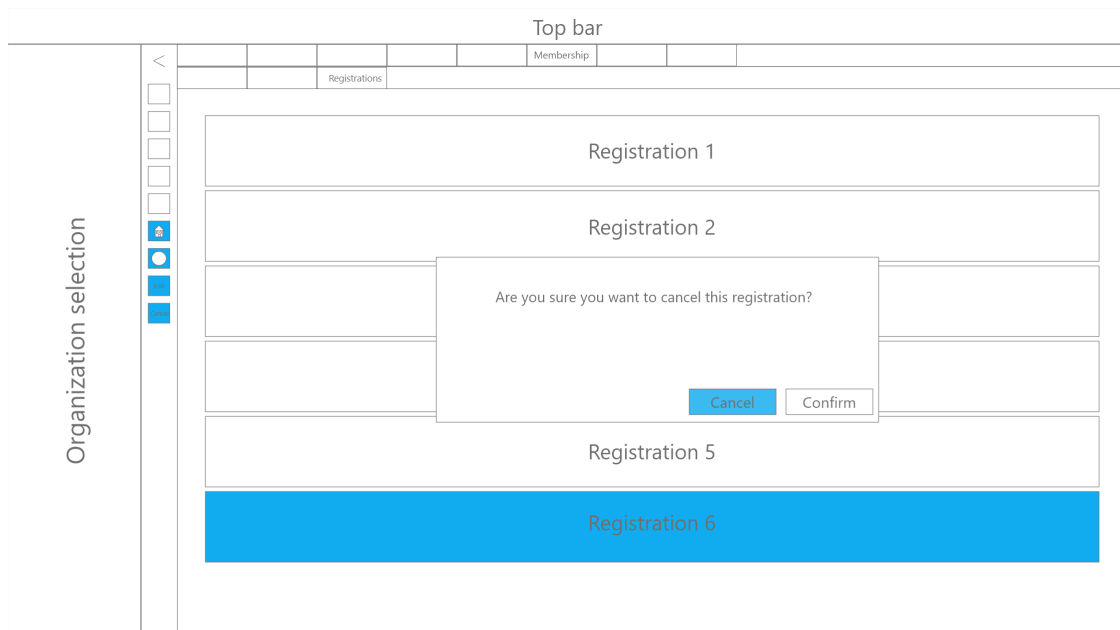


Figure 4.22: GUI prototype for cancelling a registration through Registrations' view.

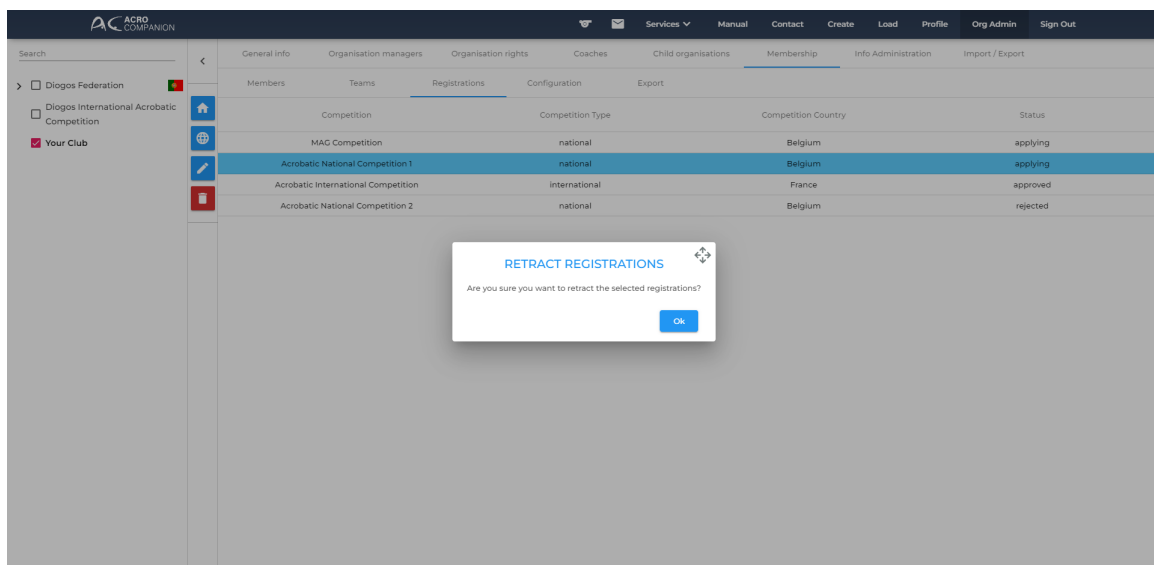


Figure 4.23: Implementation for cancelling a registration through Registrations' view.

The manager can cancel the action by clicking the escape key.

Another way a manager can cancel a registration is through the dialog for editing a registration. Figure 4.24 shows the dialog that pops up when the 'Retract registration' button is clicked.

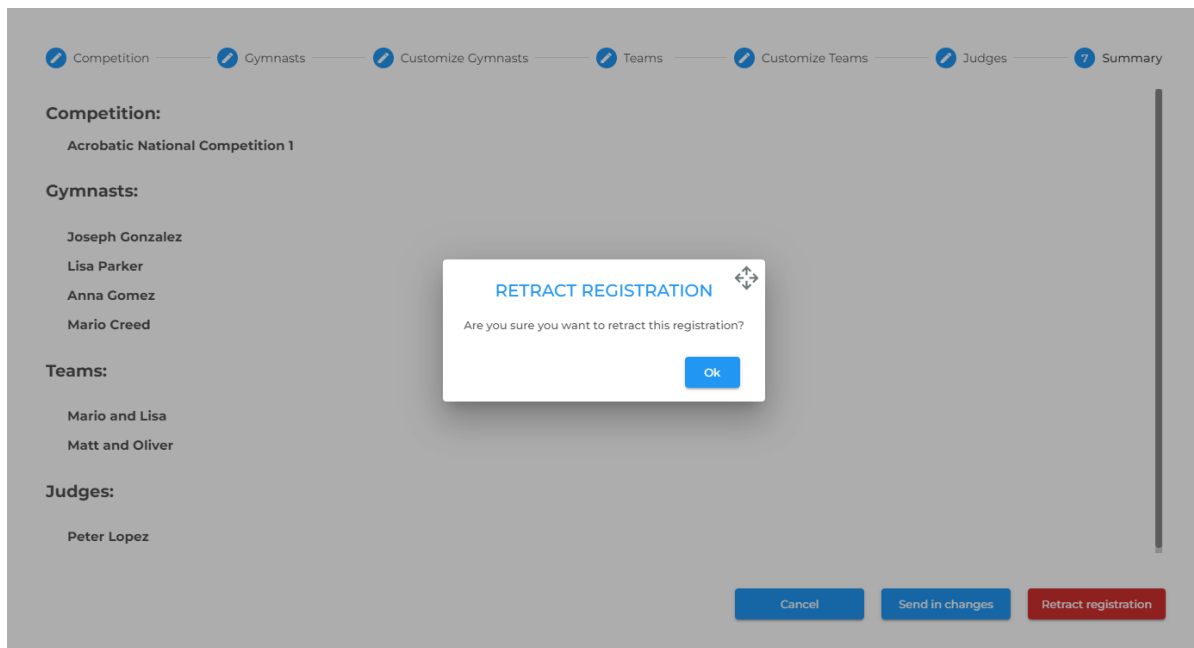


Figure 4.24: Implementation for cancelling a registration through the dialog's view.

When the manager confirms that he wants to cancel the registration clicking the "Ok" button, the registration is cancelled.

The following use cases are performed by the competition's manager.

### 4.3.7 Setting a competition open for registrations

Figure 4.25 shows the GUI prototype for opening a competition for registrations.



Figure 4.25: GUI prototype for opening a competition for registrations.

Figure 4.26 shows the implementation for opening a competition for registrations.



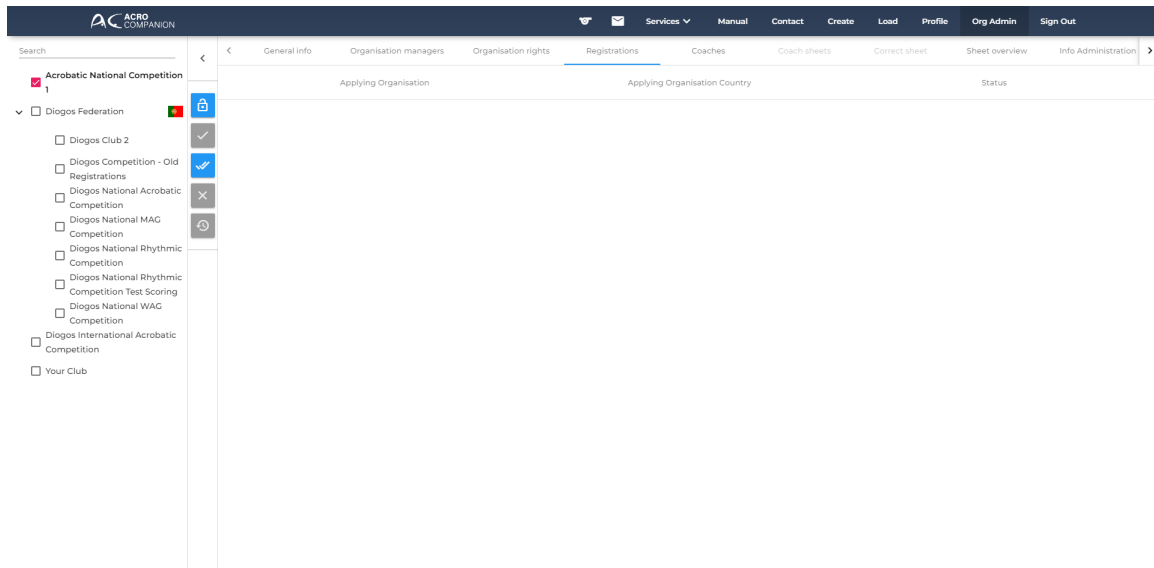


Figure 4.26: Implementation for opening a competition for registrations.

A small difference between the prototype and the implementation is that while on the prototype the button reflects the current status of the registration, in the implementation it reflects the action that will be performed if the button is clicked, which was considered more intuitive.

### 4.3.8 Approving a registration

There are two ways a competition's manager can approve a registration.

The first one is by selecting the registrations he intends to approve and clicking the button with a single 'done' icon. Once the button is clicked, a dialog pops up, as displayed in figure 4.27.

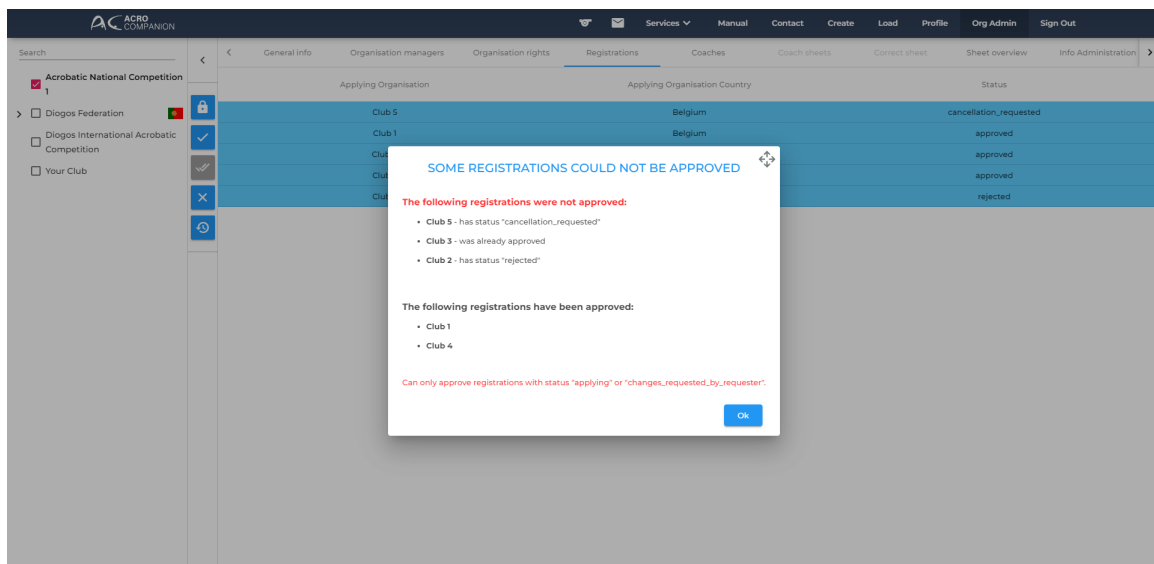


Figure 4.27: Approving multiple registrations.

If the manager wants to approve all the approvable registrations, he can simply click the button with two 'done' icons, without having to select any registration. By clicking the button, as figure 4.28 shows, a confirmation dialog pops up.

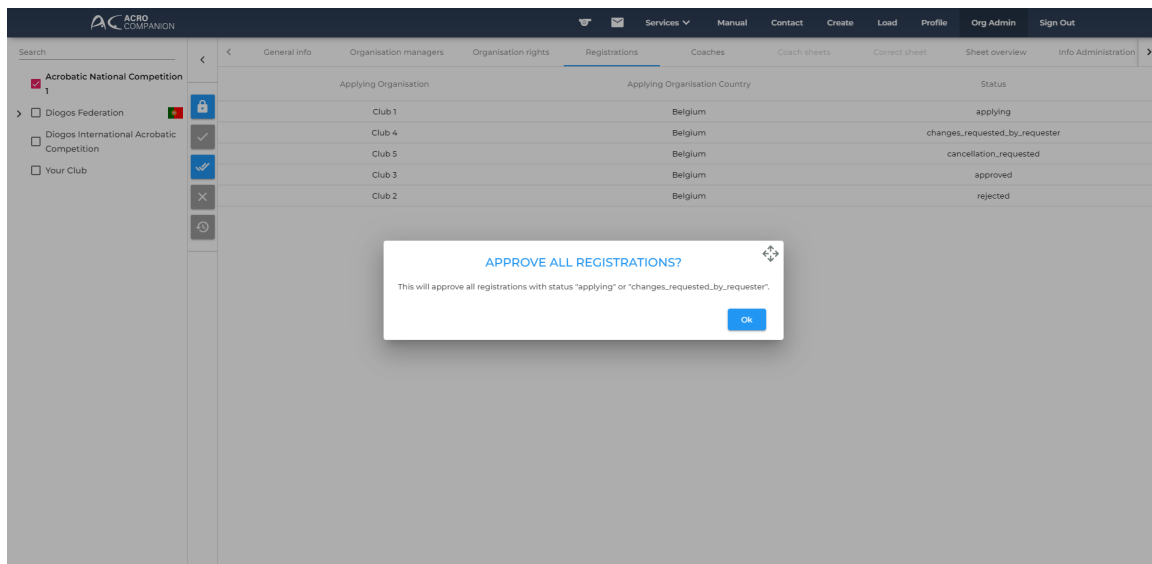


Figure 4.28: Confirmation dialog for approving all approvable registrations.

Once the manager clicks "Ok", the dialog in figure 4.29 pops up, showing the registrations that were approved. This dialog will never show registrations that could not be approved, as only approvable registrations are approved when the button with two 'done' icons is clicked.

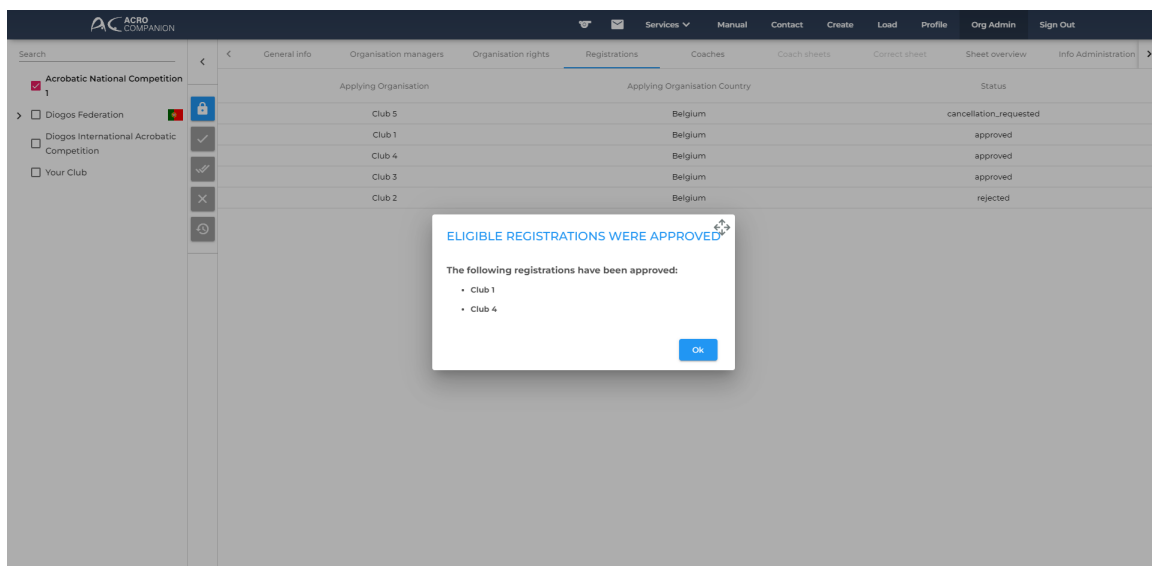


Figure 4.29: Approving all approvable registrations.

The second way a manager can approve a registration is from the dialog for viewing a registration. From the competition side, the first step of the dialog shows details about the applying organization, instead of showing the details of the competition, as displayed in figure 4.30.

1 Applying organisation — 2 Gymnasts — 3 Customize Gymnasts — 4 Teams — 5 Customize Teams — 6 Judges — 7 Summary

**Club 1**

**Contacts:**  
Phone: 912345678  
Email: club1@gmail.com

**Address:**  
De Veurs Comberg 54  
Postal Code: 5550  
City: Vresse-sur-semois  
Province / State: Namur





**Country:**  
Belgium 🇧🇪

Cancel Next

Figure 4.30: Applying organization details.

The other steps, show the selected gymnasts, teams, and judges, in a read-only mode, as displayed in figure 4.31.

1 Applying organisation — 2 Gymnasts — 3 Customize Gymnasts — 4 Teams — 5 Customize Teams — 6 Judges — 7 Summary

 <b>Joseph Gonzalez</b> Gymnast Acrobatic	 <b>Lisa Parker</b> Gymnast Acrobatic	 <b>Anna Gomez</b> Gymnast Acrobatic, Rhythmic	 <b>Mario Creed</b> Gymnast Acrobatic, Trampoline
---	---	--	---

Cancel Next

Figure 4.31: Selected gymnasts from the competition side.

The "Customize Gymnasts" and "Customize Teams" steps show the gymnasts and teams with their customized properties, as displayed in figure 4.32.

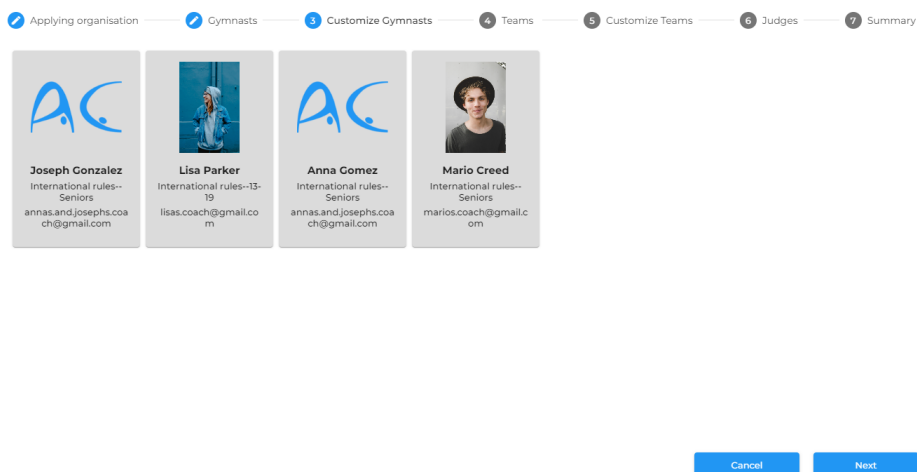


Figure 4.32: Customized gymnasts from the competition side.

The last step displays a summary of the registration, as shown in figure 4.33.

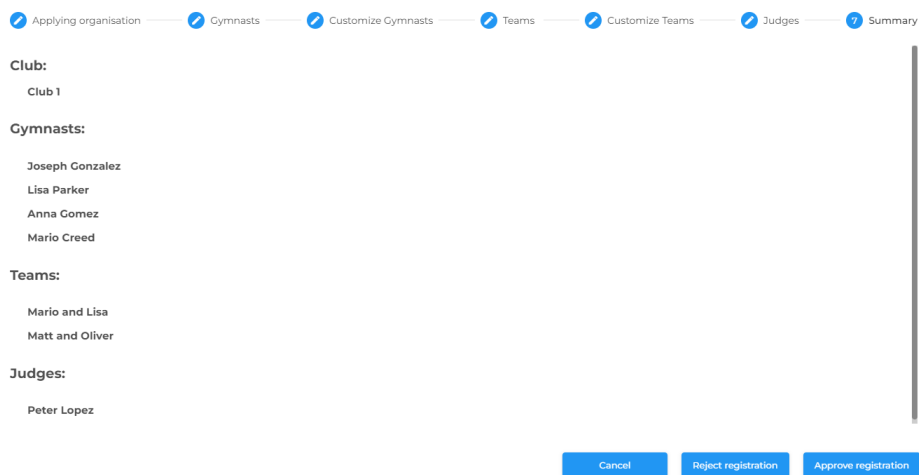


Figure 4.33: Registration summary from the competition side.

A manager can click the "Approve registration" button to approve the registration. By clicking the button, the registration is immediately approved without displaying any confirmation dialog.

### 4.3.9 Rejecting a registration

Rejecting a registration is a similar process to approving a registration. However, there is no button for rejecting all rejectable registrations, as this is not a common operation to perform.

To reject registrations, the manager selects the registrations he intends to reject and clicks the button with a cross. Once the button is clicked, a dialog pops up, as displayed in figure 4.34.

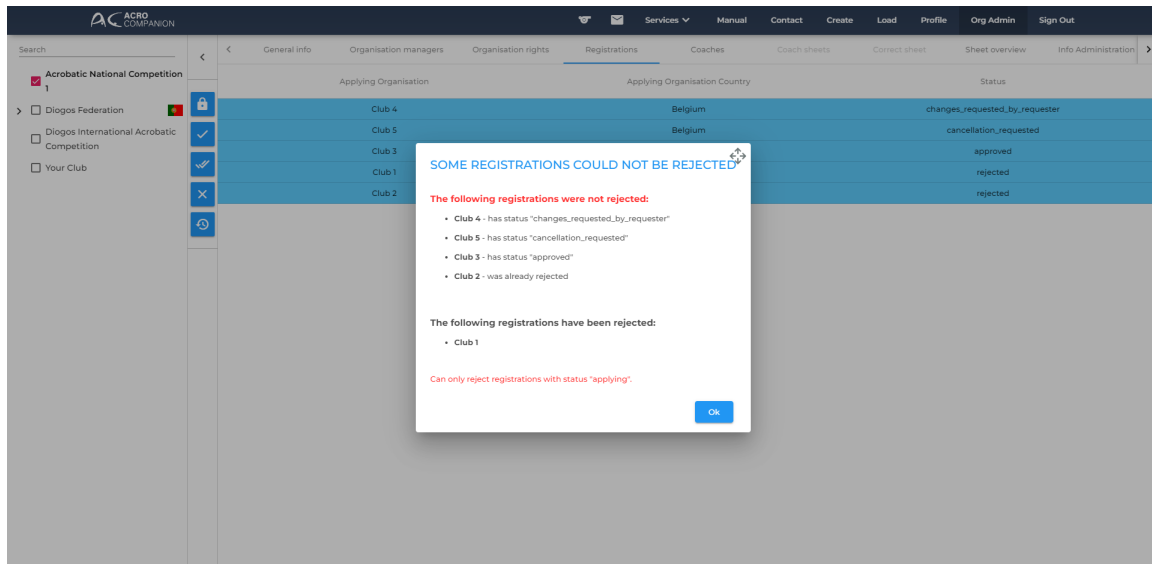


Figure 4.34: Rejecting multiple registrations.

To reject a registration through the dialog's view, the manager clicks the "Reject registration" button displayed in figure 4.33, which will immediately reject the registration without displaying any confirmation dialog.

To reject the changes of a registration, there are also two ways. One through the Registrations' view and another through the dialog's view, and they work the same way as for approving or rejecting registrations. In Registrations' view, the button for rejecting the changes has an icon of a clock with a counter-clockwise arrow, which can be seen in figure 4.34. In the dialog's view the button for rejecting the changes can be seen in figure 4.35.

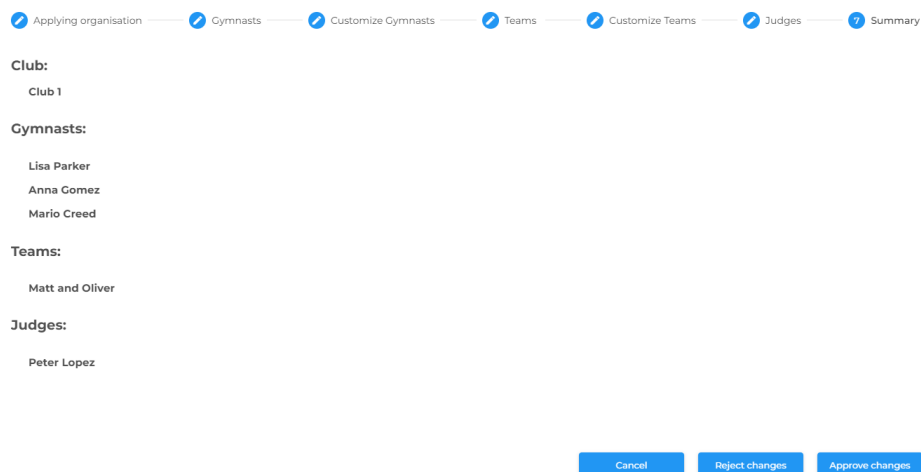


Figure 4.35: Rejecting changes for a registration.

With the use cases' implementation presented, there were other challenges not directly related to the defined use cases, but that are also interesting, which will now be presented.

### 4.3.10 Membership

When it comes to Membership, one of the things that were done was adding an extra step to the dialog, to allow the definition of gymnast-specific properties.

The step for creating a member is displayed in figure 4.36.

Figure 4.36: Definition of member properties.

The new dialog's step is displayed in figure 4.37.

Figure 4.37: Definition of gymnast-specific properties.

Initially, it was only possible to create teams for the Acrobatic discipline. Depending on the selected category, the number of gymnasts to assign to the team would vary as well as the gender for each gymnast. This was extended to support other disciplines. A difference between Acrobatic and the other disciplines is that in Acrobatic there are different roles, as figure 4.38 shows (at the top of the member's cards there are two roles "top" and "base 1").

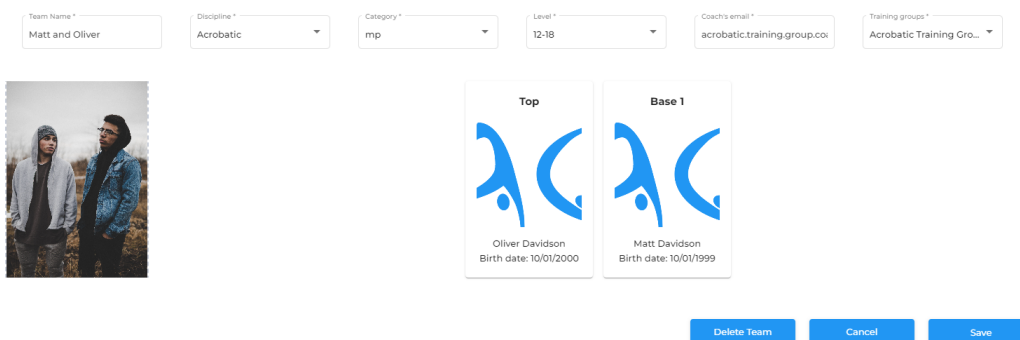


Figure 4.38: Acrobatic team, with category 'Male Pair'.

For the category "Male pair", the team can only have two male gymnasts. But as displayed in figure 4.39 if the category was changed to "Women Group", then the team would have to be composed by three female gymnasts.

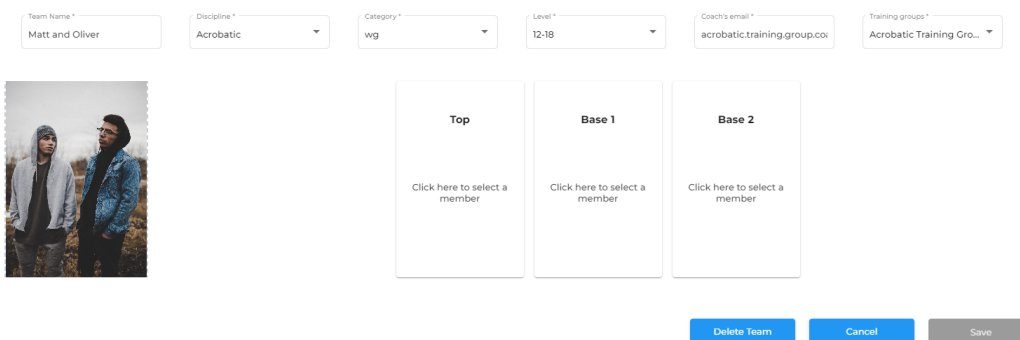


Figure 4.39: Acrobatic team, with category 'Women Group'.

By clicking the member cards, a dialog pops up, as shown in figure 4.40, displaying the possible gymnasts to select for that card filtered by gender. Only for acrobatics the gender can be different between the different cards, because of the roles. It is common for example for the bases to be males and the top a female.

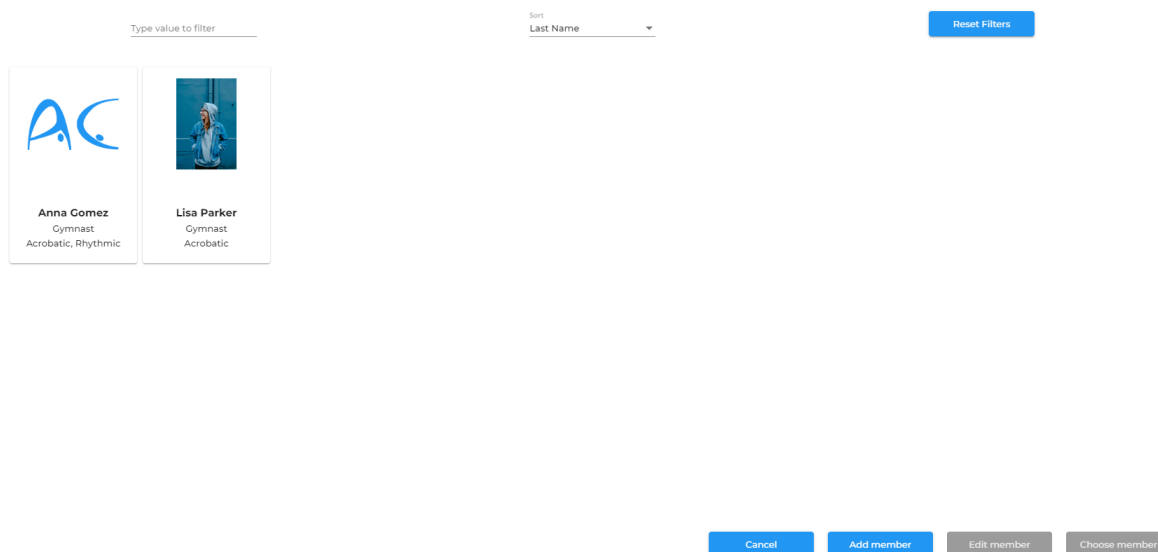


Figure 4.40: Dialog to select gymnasts for team.

With the new temporary properties, it is possible to create a team for the discipline Rhythmic, for example, and depending on the selected category, that team can have up to 8 female gymnasts, none of which with a specific role, as figure 4.41 shows.

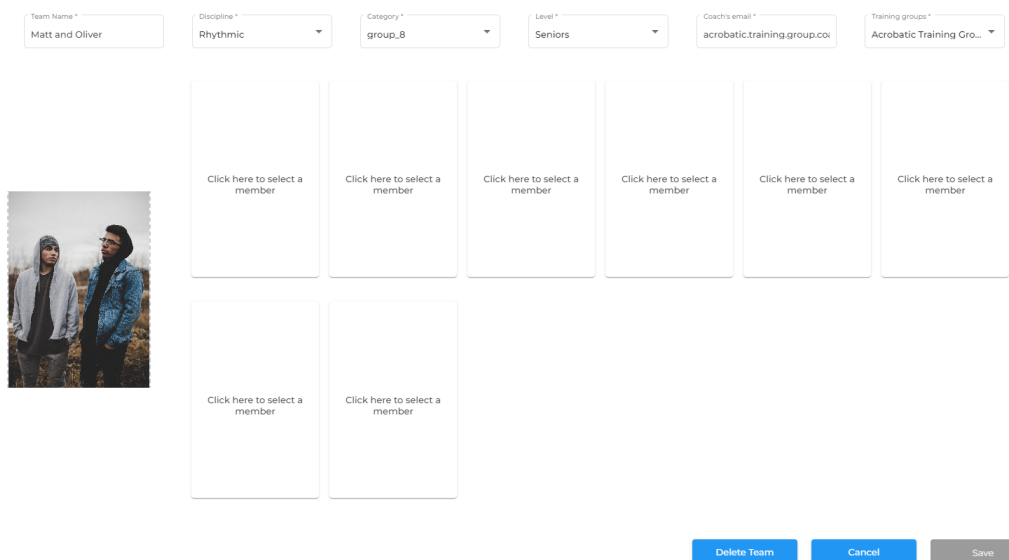


Figure 4.41: Rhythmic team, with category 'Group 8'.

### 4.3.11 Level Selector

When presenting the use case "Editing Gymnast's level, coach email, or apparatuses for registration", in section 4.3.3, it was mentioned that the dropdown for selecting a level would be presented later.

When creating a member, a level has to be assigned to each selected discipline, as was already mentioned. When creating a team, a discipline is also selected, and a level has to be assigned to the team as well. Finally, when creating a registration, the manager might have to customize the gymnasts



or teams' level according to the selected competition. In all these cases, the valid levels for selection will depend on a discipline and a country, respectively:

- the discipline selected for the gymnast and the country of the gymnast's organization;
- the discipline selected for the team and the country of the team's organization;
- the discipline and country of the selected competition.

A third factor that is taken into account are the enabled levels. For example, a competition might restrict the levels allowed for participation, even though they are valid for the competition's discipline and country. In this case, these levels will be greyed out.

Figure 4.42, shows the options for the level selection when customizing a gymnast's level during the registration creation process.

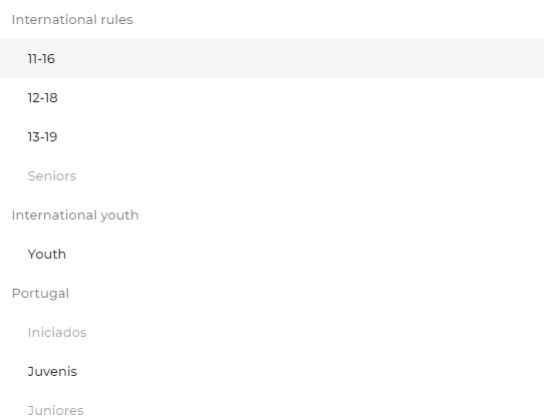


Figure 4.42: Level options.

The gymnast belongs to a club from Belgium, and has a level that is valid only in Belgium. However, the selected competition takes place in Portugal.

There are three categories of levels that are allowed: "International rules", and "International youth", which are always available independently of the selected country, and a country-specific category, "Portugal".

During the course of the project, there were also smaller tasks that were performed, for example fixing bugs as functionalities were being developed (a lot of times revealed by the test suites that were written), participating in code reviews and creating a manual for Membership and Registrations which can be accessed in:

- [https://www.acro-companion.com/introduction/manage\\_your\\_members/top](https://www.acro-companion.com/introduction/manage_your_members/top)
- [https://www.acro-companion.com/introduction/manage\\_your\\_teams/top](https://www.acro-companion.com/introduction/manage_your_teams/top)
- [https://www.acro-companion.com/introduction/register\\_to\\_a\\_competition/top](https://www.acro-companion.com/introduction/register_to_a_competition/top)

One example of a bug that was identified thanks to the test suite developed for member creation is the following: gymnasts have some properties that other members do not have, which are defined when the gymnast is created. A member can stop being a gymnast and start being a judge for example. For that to be done, the user only has to change the member type by unchecking the "Gymnast" checkbox and checking the "Judge" checkbox. But something that has to be done by the software is deleting the gymnast properties from the database document when the member is no longer a gymnast. This was not being taken into account before. One of the tests in the test suite verified that the database instances did not have any properties defined that did not make sense for the member's type, as mentioned in the next section 4.3.12, and therefore, identified this bug.

### 4.3.12 Testing

During the project, as new features were added, tests for those features were created as well. Membership did not have any tests in the beginning, and Registrations did not exist.

Playwright was used for end-to-end testing. With Playwright, it is possible to launch a browser's instance and simulate user input, indicating the expected result for each action.

Test suites for managing members, teams, and registrations were created, as well as for the level selector.

Before each test suite was created, a list of different tests was discussed amongst AcroCompanion's developers, which try to come up with the simplest list of tests that covers all the functionalities involved.

More recently Acro Companion has started to develop more unit tests focused on asserting that the Observables behave as expected. Unit tests are simpler to develop as they are focused on a much smaller scope. For that same reason, the bugs are much easier to fix, as the developers know where to search for its origin (the unit of code being tested). Finally, they are faster to run, allowing for a quicker development process, as the units being tested are usually small, and no GUI has to be loaded.

Not only the developers, but also Acro Companion's clients tested the new solution for Membership and Registrations. Up until the time of this writing, there were virtually no bugs due to all the planning and testing done beforehand. There was however feedback that will be implemented in the future:

- It should be possible to import members for Membership, rather than creating them manually.
- It should be possible to add any member to a registration, and give it a role. For example, it should be possible to add a coach to a registration and give it a role of 'Medical Doctor'.
- The summary should use cards rather than text in order to improve readability.
- The creation of a registration with a team for which not all gymnasts have been selected should not be allowed. Furthermore, in the summary there should be a warning to help the user understand why the registration cannot be created.
- In the customization steps, it should be possible to select an exercise for a gymnast/team for all the disciplines, rather than just for Artistic.
- The competition's managers should be able to cancel registrations. If an applying organization's manager sends an email saying that the club/federation is no longer going to the competition,

but does not cancel the registration in Acro Companion's application, the competition's manager should be able to do it.

## Chapter 5

# Conclusions and Forthcoming Work

Membership and Registrations can still be vastly expanded. The first step would be to implement the use cases that were defined at the beginning of the project, but ended up not being implemented. Those use cases are "Import participating gymnasts, teams, judges, and exercises from Registrations" and "See list of participating gymnasts, teams, and judges, and list of exercises".

In practice, Membership and Registrations is working as expected, with the exception that it has not been integrated with the rest of the application. As mentioned earlier, Acro Companion aspires to become an end-to-end solution for gymnastics, with Registrations being the piece that was missing. The piece has been created now and just has to be connected to the application. Figure 5.1 shows the different main functionalities that are present in Acro Companion's application.

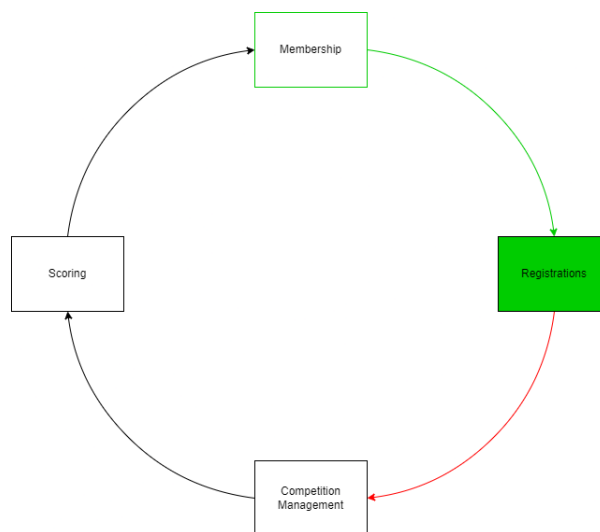


Figure 5.1: AcroCompanion's end-to-end application.

The applying organizations can now create their members and teams in Membership. This functionality is represented with a green border, meaning that it already existed, and was improved during the course of the project. In black is represented what already existed in the application prior to the start of the project.

Membership's members and teams are then registered into competitions through Registrations. The green arrow represents the fact that the members and teams from Membership are accessible in Registrations, which did not exist before the start of the project, since Registrations was not a functionality

yet.

Then Registrations has a green background, to represent the fact that it was the focus of the project and that it was built from zero. The created registrations hold all the information necessary, that will have to be imported into Competition Management, which is what the competition's managers use to prepare and control the competition. For example, the starting order by which the exercises will be performed is defined here.

The information that the registrations hold includes the applying organization, the participating members and teams (with the respective levels and coach's emails), and the apparatuses that will be performed (when the competition's discipline forces their definition). This is the information that will be imported into Competition Management. As this information cannot be imported yet, the arrow connecting Registrations and Competition Management is red.

Competition Management would then use this information to prepare the competition. Managers would be able to see it in Competition Management (use case "See list of participating gymnasts, teams, and judges, and list of exercises"), and as discussed, prepare the starting order. During the Competition, the starting order dictates which gymnast or team is going to perform, and then Scoring is used to score the exercises. Finally, these scores are associated to the gymnasts from Membership.

From the users' perspective, there are also multiple things that can be improved.

When a competition's manager has to approve the changes made to a registration, there is, currently, no way for that manager to see which gymnasts, teams, and judges were previously selected, which makes it much more difficult to understand what changes were made. The previous members and teams are actually already stored in the registration object, and therefore, it should not be too difficult to develop a way for those managers to compare the previous state of the registration with the new state that still has to be approved, helping them making a decision.

From the applying organization's side, the managers currently do not have an easy way to see if the changes to a registration were approved or rejected, since in both scenarios, as was already discussed, the status of the registration will be 'approved'. It would, therefore, be interesting to implement a notification system that would notify the applying organization's manager that the registration's changes have been approved/rejected, and, if rejected, that the competition's manager could provide the reason for the rejection.

It would also be interesting to allow the competition's manager to request that the applying organization's manager modifies a certain aspect of the registration. This would be useful if the competition's manager accidentally approved a registration, that should not be approved, for example.

Finally, currently, the names of the statuses are hardcoded in the GUI. Each status will, in the future, have a more readable label to be displayed. One step further, each label can be translated to the language of the user seeing it.

Membership and Registrations will be used in a competition for the first time in November, in Germany. Until then, the two missing use cases will have to be implemented. After that, as Membership and Registrations gets used by more and more users, a lot of feedback is expected, and it will be the focus during the following months to implement that feedback.

A complete solution for gymnastics benefits Acro Companion. As the company's software gets used by more and more clients that have the main tools for managing gymnastics now available in a

single place, Acro Companion is likely to get more and more contracts to collaborate with federations in multiple competitions, which in practice means it will grow its revenue. But gymnastics competitions and gymnastics as a whole will greatly benefit from this solution as well. This solution will allow the transition from working with spreadsheets to having a functional tool dedicated to gymnastics, which is significantly faster, more practical, and intuitive to work with.



## **Appendix A**

### **Time management gantt chart.**



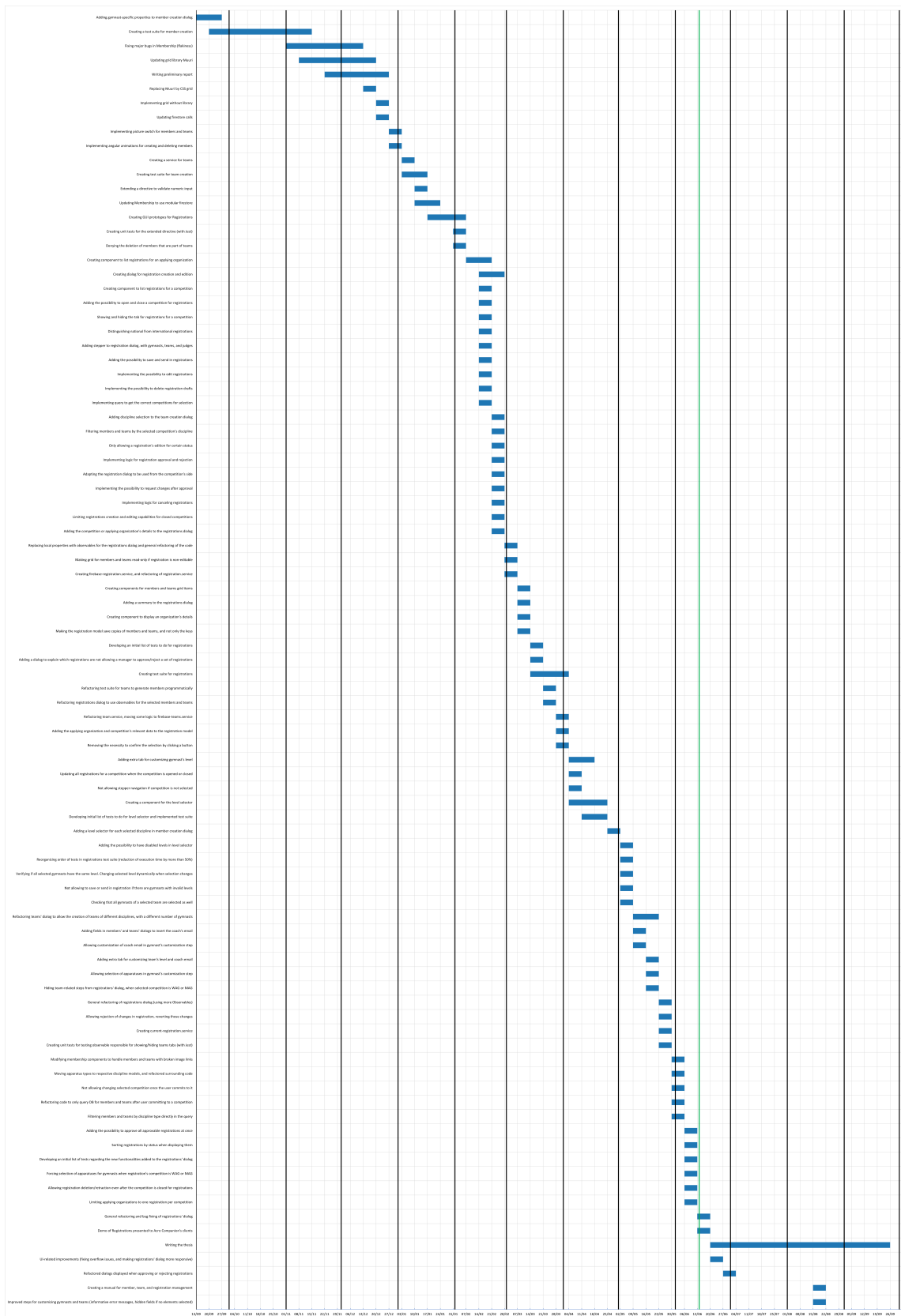


Figure A.1: Time management gantt chart.

# Bibliography

- [1] Time management gantt chart. <https://1drv.ms/u/s!Ajv1R6hR2vQdh8hBlv4Ap4VumSSPEQ?e=7wm60L>.
- [2] Engineer Bainomugisha, Andoni Lombide Carreton, Tom van Cutsem, Stijn Mostinckx, and Wolfgang de Meuter. A survey on reactive programming. *ACM Comput. Surv.*, 45(4), aug 2013.
- [3] Eventsport. Eventsport, quem somos. <https://eventsport.pt/quem-somos/>. Accessed: 2022-09-20.
- [4] Amazon. What is devops. <https://aws.amazon.com/devops/what-is-devops/>. Accessed: 2022-09-1.
- [5] Microsoft. What is devops. <https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-devops/>. Accessed: 2022-09-1.
- [6] Amazon. What is cloud computing. <https://aws.amazon.com/what-is-cloud-computing/>. Accessed: 2022-09-1.
- [7] Google. What is cloud computing. <https://cloud.google.com/learn/what-is-cloud-computing>. Accessed: 2022-09-1.
- [8] NPM. About npm. <https://www.npmjs.com/about>. Accessed: 2021-12-21.
- [9] NPM. npm. <https://docs.npmjs.com/cli/v8/commands/npm>. Accessed: 2021-12-21.
- [10] Google. Chrome devtools. <https://developer.chrome.com/docs/devtools>. Accessed: 2021-12-22.
- [11] Microsoft. Typescript for the new programmer. <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html>. Accessed: 2021-12-14.
- [12] Gavin Bierman, Martín Abadi, and Mads Torgersen. Understanding typescript. In Richard Jones, editor, *ECOOP 2014 – Object-Oriented Programming*, pages 257–281, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [13] Microsoft. Why create typescript. <https://www.typescriptlang.org/why-create-typescript>. Accessed: 2021-12-14.

- [14] Microsoft. Typescript for javascript programmers. <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>. Accessed: 2021-12-14.
- [15] Google. What is angular? <https://angular.io/guide/what-is-angular>. Accessed: 2021-12-14.
- [16] Google. Dependency injection in angular. <https://angular.io/guide/dependency-injection>. Accessed: 2021-12-14.
- [17] Google. Introduction to services and dependency injection. <https://angular.io/guide/architecture-services>. Accessed: 2021-12-14.
- [18] Rxjs introduction. <https://rxjs.dev/guide/overview>. Accessed: 2021-12-21.
- [19] Google. Firebase & google cloud. <https://firebase.google.com/firebase-and-gcp>. Accessed: 2021-12-21.
- [20] Google. Firebase products. <https://firebase.google.com/products-build>. Accessed: 2021-12-21.
- [21] Google. Cloud firestore. <https://firebase.google.com/products/firestore>. Accessed: 2021-12-21.
- [22] Microsoft. What is azure devops? <https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>. Accessed: 2021-12-24.
- [23] Microsoft. What is azure repos? <https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops>. Accessed: 2021-12-24.
- [24] Microsoft. What is azure pipelines? <https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops>. Accessed: 2021-12-24.
- [25] Microsoft. What is azure boards? <https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops>. Accessed: 2021-12-24.
- [26] Microsoft. Playwright. <https://playwright.dev/>. Accessed: 2021-12-24.
- [27] Microsoft. Getting started. <https://playwright.dev/docs/intro>. Accessed: 2021-12-24.
- [28] Meta. Jest. <https://jestjs.io/>. Accessed: 2022-09-17.
- [29] Git. <https://git-scm.com/>. Accessed: 2022-09-17.
- [30] About git. <https://git-scm.com/about>. Accessed: 2022-09-17.

- 
- [31] Adobe. What is adobe xd used for. <https://www.adobe.com/products/xd/learn/get-started/what-is-adobe-xd-used-for.html>. Accessed: 2022-09-17.
- [32] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, and Philippe Suter. *Serverless Computing: Current Trends and Open Problems*, pages 1–20. Springer Singapore, Singapore, 2017.
- [33] Rxjs merge operator. <https://v6.rxjs.dev/api/index/function/merge>. Accessed: 2022-09-1.