MySportsFace

Marcos Sebastião Pereira Germano

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. João Manuel Brisson Lopes

Examination Committee
Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. João Manuel Brisson Lopes
Member of the Committee: Prof. João António Madeiras Pereira

November 2015
Dedicated to my friends and family who helped and supported me through the creation of the MySportsFace application.
Acknowledgments

Firstly, I would like to thank Instituto Superior Técnico for being a highly respectable university in terms of Engineering courses in Europe.

Secondly, I would also like to thank my dissertation supervisor, Prof. João Brisson Lopes, for guiding me through the development of the MySportsFace project and my friends and family for coming up with interesting suggestions and helping out with the testing of the application.

Last but not least, I would like to thank every other person from the general public who also collaborated by participating in the usability and satisfaction tests of the application.
Abstract

This master thesis focuses primarily on the creation of the MySportsFace application based on the social aspects of various sports, to facilitate communication and sharing of information or multimedia content between members of sports organizations or the public in general.

This application was developed to accommodate important aspects of a social network. All users have their own customizable profiles and can follow or be followed by other users and friends, in order to afterwards be able to see posts from each other through information feeds.

On top of this, every user is able to view and comment news articles provided by various sports organizations, as well as create content of their own, in the form of events, which others can view, attend and later use to upload multimedia content in the form of images and/or videos.

As part of the development of this application, a series of integration, usability and satisfaction tests was carried out to guarantee its full functionality and user experience.

Users were highly motivated to experiment this new application and even suggested improvements and changes during the testing phase.

Keywords: social networks, sports, sports organizations, multimedia, information
Resumo

Esta tese de mestrado centra-se na criação da aplicação MySportsFace com base nos aspectos sociais dos desportos, facilitando assim a comunicação e partilha de informação ou conteúdo multimédia entre membros de organizações desportivas ou o público em geral.

Esta aplicação foi desenvolvida com o objectivo de implementar funcionalidades relevantes para uma rede social. Os utilizadores possuem perfis customizáveis e podem seguir ou ser seguidos por outros utilizadores, de forma a visualizar os posts de cada um através de feeds de informação.

Para além disso, cada utilizador pode visualizar e comentar artigos informativos provenientes das organizações desportivas, assim como criar eventos pessoais, que outros poderão ver, aderir e sobre os quais poderão posteriormente carregar conteúdo multimédia, na forma de imagens ou vídeos.

Como parte integrante do desenvolvimento desta aplicação, foi levada a cabo uma série de testes de integração, usabilidade e satisfação com o fim de garantir a sua total funcionalidade e experiência de utilizador.

Os utilizadores sentiram-se bastante motivados a usar esta nova aplicação, tendo inclusive sugerido várias melhorias e alterações durante a fase de teste.

Palavras-chave: redes sociais, desportos, organizações desportivas, multimédia, informação
## Contents

Acknowledgments ................................................................. v
Abstract .............................................................................. vii
Resumo ................................................................................ ix
List of Tables ......................................................................... xv
List of Figures ......................................................................... xvii
Glossary ................................................................................. xix

1 Introduction ......................................................................... 1
   1.1 Social Networking and Sports ........................................... 1
   1.2 Merging Ideas ................................................................... 2
   1.3 Objectives ........................................................................ 3
      1.3.1 Application Scenarios ................................................ 4
   1.4 Thesis Outline ................................................................... 5

2 Sports Applications and Social Media ..................................... 7
   2.1 Website of Fédération Internationale de Ski (FIS) .............. 8
      2.1.1 Evaluating the Website of FIS ................................. 9
   2.2 NASCAR Website ......................................................... 10
      2.2.1 Evaluating the NASCAR Website ....................... 11
   2.3 SPORT TV Multiscreen Application .................................. 11
      2.3.1 Evaluating SPORT TV Multiscreen Application .......... 12
   2.4 BeSoccer Application ...................................................... 13
      2.4.1 Evaluating BeSoccer Application ............................ 13
   2.5 Website of Ski Clube de Portugal ..................................... 14
      2.5.1 Evaluating the website of Ski Clube de Portugal ...... 15
   2.6 Website of Federação de Desportos de Inverno de Portugal (FDI-PT) ......................................................... 15
      2.6.1 Evaluating the website of FDI-PT ............................ 16
   2.7 Website for Federação de Triatlo de Portugal (FTP) ........... 17
      2.7.1 Evaluating the website of FTP ............................... 18
   2.8 State-of-the-Art Summary ................................................. 19
3 Selecting the Technology and Tools

3.1 Microsoft ASP.NET ........................................... 22
3.2 Zend Framework ............................................. 22
3.3 Ruby-on-Rails vs. Python's Django .......................... 23
  3.3.1 Ruby-on-Rails Project Structure ......................... 24
  3.3.2 Ruby Gems ............................................... 25
  3.3.3 Assets .................................................. 26
3.4 Choosing a Database and Web Server ....................... 26
  3.4.1 Development Environment ............................... 26
  3.4.2 Production Environment ................................ 26
  3.4.3 Migrations ............................................. 27
3.5 Model-View-Controller as per Ruby-on-Rails .............. 27
  3.5.1 Models ................................................. 28
  3.5.2 Views .................................................. 28
  3.5.3 Controllers, Routing and Resources ..................... 28
3.6 HTML5, CSS3.0 and JavaScript ............................... 29
  3.6.1 Front-End Code ........................................ 29
  3.6.2 Scripting and Animations ............................... 29
  3.6.3 Styling the Website .................................... 30
3.7 Ruby Gems and Libraries .................................... 31
  3.7.1 Superfeedr Engine ...................................... 31
  3.7.2 JavaScript and CSS Code Optimization .................. 32
  3.7.3 Bootstrap ............................................... 33
  3.7.4 Font-Awesome-SASS .................................... 35
  3.7.5 Uglifier ............................................... 36
  3.7.6 jQuery-Rails, Turbolinks and jQuery-Turbolinks .... 36
  3.7.7 SQLite 3, PostGre Rails and Puma ...................... 36
  3.7.8 Faker .................................................. 37
  3.7.9 BCrypt ............................................... 37
  3.7.10 Uploads .............................................. 37
  3.7.11 Rails Gallery .......................................... 38
  3.7.12 Omniath .............................................. 38
  3.7.13 Thumbs Up ............................................ 39
  3.7.14 Other Minor Gems ..................................... 40
  3.7.15 Other JavaScript Libraries ............................ 40

4 The Architecture of MySportsFace .......................... 42

4.1 MySportsFace Requirements .................................. 42
4.2 Initial Architecture and Problems .......................... 42
### 4.2 Users
- 4.2.1 Users
- 4.2.2 News Articles and Information
- 4.2.3 Events and Registration
- 4.2.4 Payments
- 4.2.5 Summary of the Issues

### 4.3 Current Architecture
- 4.3.1 Users and Relationships
- 4.3.2 Organization Feeds and News Article Extraction
- 4.3.3 Community Events and Other Content
- 4.3.4 Article and Event Comments

### 5 The MySportsFace Application
- 5.1 Static Pages
- 5.1.1 About the Website
- 5.1.2 Frequently Asked Questions
- 5.2 Users and Profiles
- 5.2.1 Sessions and Logging In
- 5.3 Requirements for Supporting Organizations
- 5.3.1 Feeds and Content Constraints
- 5.3.2 Adding Other Organizations (and their RSS feeds)
- 5.4 Posts and Comments
- 5.5 Community Events
- 5.5.1 Image Uploading and Video Links
- 5.5.2 Event Administration
- 5.6 Voting System
- 5.7 Security Permissions and Content Availability
- 5.8 Deployment and Amazon Web Services Integration
- 5.8.1 Heroku
- 5.8.2 Amazon Simple Storage Service (S3)

### 6 Testing the Application
- 6.1 Unit Tests
- 6.1.1 Database Fixtures
- 6.1.2 Model Tests
- 6.1.3 Controller Tests
- 6.1.4 Integration Tests
- 6.2 Usability and Satisfaction Tests
- 6.2.1 Test-user Profile
- 6.2.2 Use-Case Scenarios and Tests Specification
- 6.2.3 Test Results
List of Tables

1.1 Number of monthly active users per social networking site. .......................... 2

2.1 Summary of all the criteria respected by each website or application reviewed. .... 19

6.1 Student's t-test data and results for the initial users when completing Task 3. ........ 72
6.2 Student's t-test data and results for the initial users when completing Task 5. ........ 72
6.3 Student's t-test data and results for the initial users when completing Task 9. ........ 72
6.4 Student's t-test data and results for the initial users when completing Task 10. .... 72
6.5 Student's t-test data and results comparing the initial and new users when completing Task 3. ................................................................. 74
6.6 Student's t-test data and results comparing the initial and new users when completing Task 5. ................................................................. 74
6.7 Student's t-test data and results comparing the initial and new users when completing Task 9. ................................................................. 74
6.8 Student's t-test data and results comparing the initial and new users when completing Task 10. ................................................................. 74

C.1 Time and error measurements for task 1. ................................................... 90
C.2 Time and error measurements for task 2. ................................................... 91
C.3 Time and error measurements for task 3. ................................................... 91
C.4 Time and error measurements for task 4. ................................................... 91
C.5 Time and error measurements for task 5. ................................................... 92
C.6 Time and error measurements for task 6. ................................................... 92
C.7 Time and error measurements for task 7. ................................................... 93
C.8 Time and error measurements for task 8. ................................................... 93
C.9 Time and error measurements for task 9. ................................................... 94
C.10 Time and error measurements for task 10. ............................................. 94
C.11 Time and error measurements for task 11. ............................................. 95
List of Figures

1.1 Process for asynchronously consuming RSS feeds from sports organizations. 3

2.1 Layout of the website for FIS-SKI. 8
2.2 Photo of an event shown in a gallery at FIS. 9
2.3 The navigation bar at the website of FIS. 9
2.4 Layout of the website for NASCAR. 10
2.5 Part of NASCAR’s website showing links to various event results. 11
2.6 Interface of the SPORT TV Multiscreen application showing a football match. 12
2.7 Interface of the BeSoccer application where users can leave comments. 13
2.8 Layout of the website of Ski Clube de Portugal. 14
2.9 Layout of the website of FDI-PT. 16
2.10 Screenshot of the website for Winter Sports Federation of Portugal showing an event advertisement image. 16
2.11 Layout of the website of Federação de Triatlo de Portugal. 17
2.12 Form required to become a non-federated athlete at FTP. 18

3.1 Typical collaboration of the MVC components. 21
3.2 Programming language distribution in a Ruby-on-Rails project. 25
3.3 Example of a command creating a new migration in Ruby-on-Rails. 27
3.4 Example of a resource specification in a Rails routing file. 29
3.5 List of included JavaScript files and code libraries for MySportsFace. 30
3.6 Superfeedr configuration in MySportsFace. 31
3.7 Example of the difference in code syntax between CSS and SASS. 32
3.8 Defining dynamic properties using traditional CSS code and SASS code. 32
3.9 Code excerpt written in CoffeeScript and its JavaScript counterpart. 34
3.10 Pagination in Rails using the “Will Paginate” gem. 35
3.11 Small set of icons from the Font-Awesome-SASS gem. 35
3.12 Example of a credentials provider specified for Omniauth. 38
3.13 Helper methods provided by the “Thumbs Up” gem. 40
3.14 A customized text-area after being replaced by the NicEdit library. 41

4.1 Initial architecture of MySportsFace. 43
4.2 UML class diagram of MySportsFace. .................................................. 46
5.1 Information shown in the MySportsFace "About" page. .......................... 50
5.2 Example of a user profile and the content displayed in it. ......................... 51
5.3 MySportsFace homepage, showing the various ways to log into the application. . 52
5.4 HTML form for adding new organizations to MySportsFace. ...................... 53
5.5 Recent Posts feed page on MySportsFace. ........................................... 54
5.6 Section of an event page showing its information. ................................. 55
5.7 Example of an event image gallery. .................................................... 56
5.8 Example of an event video slideshow. ................................................ 57
5.9 Example of an event participants list with administration options. ............. 58
5.10 Gallery image thumbnails with a "delete" option. .................................. 58
5.11 Three posts showing star icons for voting. ......................................... 58
5.12 Heroku application dashboard. ....................................................... 60
6.1 Excerpt of a model test code. .......................................................... 63
6.2 Excerpt of a controller test code. ..................................................... 64
6.3 Excerpt of a integration test code. .................................................... 65
6.4 Average time and errors for each task in the first testing phase. ............... 68
6.5 Average time and errors for each task in the second testing phase. .......... 69
6.6 Comparison of the average task errors between both testing phases. ........ 70
6.7 Comparison of the average task completion times between both testing phases. 70
6.8 Comparison of the average task completion times between both testing phases for the 10 initial users only. .................................................. 71
6.9 Comparison of the average task completion times between the 10 initial users and the new ones during the second testing phase. ......................... 73
6.10 Comparison of the average task completion times between the 10 initial users during the first testing phase with just the new ones during the second phase. .... 73
Glossary

API Application Programming Interface is the name given to a set of routines, protocols and tools for building software applications. 24, 25, 30, 31, 33, 39, 60, 75, 76

FDI-PT Federação de Desportos de Inverno de Portugal is the leading sports organization in Portugal which is in charge of all winter sport modalities. 15

FIS Fédération Internationale de Ski is a sports organization in charge of all worldwide ski events and other similar competitions. 8, 10

FTP Federação de Triatlo de Portugal is the main organization in Portugal which is in charge of Triathlons and other related sport modalities. 17

MVC Model-View-Controller is an architectural pattern used by web frameworks to build applications with graphical user interfaces. 21, 22, 27

RSS Really Simple Syndication or Rich Site Summary is a form of web feed used to subscribe for and publish information. 3, 45, 53

SIBS Sociedade Interbancária de Serviços is a Portuguese company which manages Multibanco and most payment services. 44, 45, 76
Chapter 1

Introduction

Since the early 2000s, some people around the world started creating their very own social networks and other spaces where users could publish content and socialize. Examples of these are MySpace, Facebook, hi5 and later Twitter and Google+, among many others.

With the very young concepts of social networking websites back then, any user from around the world could already form virtual relationships with a multitude of other users, using “friendship requests” (which is the case of an active mutual relationship, since both users have to agree on it) or by “following” other users (thus having a passive one-sided relationship). Besides this, users could also post content of various types and forms, such as videos, images and texts.

However, not until later, with the development of better mobile technology and applications in general, did the sports world start seeing its first web and mobile applications, which offered some features for browsing match and event results, standings and other information.

Recently, there is a wide variety of online sports applications and websites which provide a multitude of services to their users, such as the websites for well-known organizations like FIFA or FIS-SKI, which have daily batches of news articles with all kinds of information, together with other pages that contain event or match specific details.

1.1 Social Networking and Sports

There is an incredibly vast amount of users around the world who use social networking websites. For example, Facebook alone has over 1.3 billion monthly active users and other websites like LinkedIn, Twitter, Instagram or Google+ do not have as many users, but the numbers are still well over the 200 million mark. Figure 1.1 shows the number of users each social networking site had, as of March 2015, according to research by BrowserMedia, Socialnomics and MacWorld [1].

Even though there are many places where users can browse to gather information on the sports world and hot topics, there are still a lot of other tasks that users are able to perform but require additional external applications, such as is the creation of events.

On the one hand, for example, some organizations allow users to pay for their annual subscriptions
fees online while, on the other hand, part of them just put the results and standing of events online for every interested user to check.

Moreover, another problem is the fact that websites which offer the most wanted features and services are not user-friendly for beginners.

Therefore, it is urgent to address these issues, in order to provide every member of sports organizations with the necessary services regarding payments, signing up for events or even creating personal private events for friends and family and being able to share any multimedia content with other members, be it images or videos.

Thus, it would be important to solve all these issues at once by creating a web application which would provide users with the aforementioned services. Some of the services already exist in some applications so, in that case, the objective will be to learn from them instead, whenever possible. Among those existent services, the following ones should all be present within the same application:

- Registering at the application to have access to all the content
- Performing payments and online shopping
- Organizing and creating events, signing up for events and browsing results
- Sharing multimedia content with other users
- Browsing worldwide information regarding various organizations

### 1.2 Merging Ideas

Since people from all around the world spend a great deal of their time constantly checking for updates on social networking websites, and an even bigger number are fans of various sports and events, the

<table>
<thead>
<tr>
<th>Largest Social Networks in the World by Number of Users</th>
<th>Number of users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>1,374,000,000</td>
</tr>
<tr>
<td>QZone</td>
<td>635,000,000</td>
</tr>
<tr>
<td>Google+</td>
<td>347,000,000</td>
</tr>
<tr>
<td>LinkedIn</td>
<td>336,000,000</td>
</tr>
<tr>
<td>Instagram</td>
<td>302,000,000</td>
</tr>
<tr>
<td>Twitter</td>
<td>289,000,000</td>
</tr>
<tr>
<td>Tumblr</td>
<td>237,000,000</td>
</tr>
<tr>
<td>Sina Weibo</td>
<td>162,000,000</td>
</tr>
<tr>
<td>Snapchat</td>
<td>113,000,000</td>
</tr>
<tr>
<td>Pinterest</td>
<td>73,500,000</td>
</tr>
</tbody>
</table>
idea to create an application that would merge both concepts appeared. Thus, the application would serve as a social hub by containing user relationships, their profiles and favorite sports, as well as multiple sports organizations which provide informative articles on a daily or weekly basis.

Moreover, by adding these concepts together, the users become able to leave their remarks on any type of content they find while browsing the application, be it in the form of comments, "wall"/mood updates or even votes (such as the "Like" or "Thumbs Up/Down" systems provided by Facebook and Google’s Youtube, respectively).

In order to acquire the information necessary to provide it back to the users, the idea was to grab the RSS feeds from existing organizations, extract each entry and transform it into actual database objects. The flowchart for the concept of extracting the information is represented in Figure 1.1.

![Figure 1.1: Process for asynchronously consuming RSS feeds from sports organizations.](image)

Another interesting idea that came out at a later stage was the creation of custom events made by any user of the application. This would allow everyone to organize small or big size events and competitions of all kinds through the use of the event feature, imbued into the social network which offers the previously mentioned services.

Lastly, there was also one remaining idea which was to bring online shopping and payment methods into the application, but this was later discarded to future work, due to significant constraints, detailed in Section 4.2 of Chapter 4.

### 1.3 Objectives

After having all the main ideas and concepts sorted, the main objective of this work is to create a web application that provides users with a social networking experience, by creating virtual relationships and posting content, as well as direct access to various sports organizations worldwide and the information that they contain and update on a daily basis.

Additionally, the application will also allow users to organize customized events, so as to increase the socialization between members, outside the application. The customized events will also contain ways for users to share multimedia content, such as images or videos (taken during the events, for example). In the case of images, the users will be able to upload them directly from their devices and, in the case of videos, they will first have to upload them to a third-party website (such as Youtube or Vimeo), before linking them to the event page.

Later during the development of the application, multiple sets of usability and satisfaction tests were conducted to further guarantee that the users have the best experience when using the application and
that the results get better after each testing phase.

1.3.1 Application Scenarios

In order to have a better understanding of what the application needs to do, two scenarios were created and they present the various features the application should have, along with some ways of using those features.

Scenario I

Joe, who is an athlete member of the Winter Sports Federation and has actively practised ski for 10 years, wants to be up to date with all the information and news regarding the federation and the sport in general.

In order to be able to do that, he will use the newly created MySportsFace web application, which offers access to the news posts from most organizations worldwide.

Therefore, Joe has to either register a new account in the website or sign in using his Facebook or Google+ accounts. He chooses to go with the first option.

After that, he accesses the organization index page to find the Winter Sports Federation somewhere in the list. Once he has done so, he visits the federation page to view all the recent news articles posted and he also adds that organization to his favorites, so as to facilitate search next time.

One of the articles Joe viewed regarded the upcoming National Ski Championship. The article had links to the original post and to the registration forms. He then proceeds to follow the latter link in order to enroll in the competition.

Later on, Joe looks up for other articles regarding the results of previous events he attended and decides to leave his comments on a few, to share his appreciation and thoughts on the aftermath of the competition.

The next day, he finds out that his friend Michael has also registered at the MySportsFace website, while talking to him during breakfast, and thus both start “following” each other on the application. By doing so, they now both start receiving posts from each other through their recent posts feed.

Scenario II

Inspired by his father, Joe’s son, Phillip, also likes practising ski. However, there haven’t been any ski competitions for children organized by the Winter Sports Federation recently.

Since Phillip knows that his father started using the MySportsFace application, he decided to ask his father for help to check it out so as to see if they could find any organization that would hold a ski event in the near future. Sadly, they were not able to find any events, but Joe told his son that he could use the application to create a custom event for his friends and their parents.

Phillip was excited and helped his father by overlooking the creation of the event, which included filling in a title, a small summary, a description and a time and place for the event. Afterwards, Joe
finished the event creation by also filling in the schedule for the event.

In order to make Phillip's event registration official, his father created a new account in the application for him, which he used to browse the list of events and find the one they had just made, so as to sign up for it through the application.

Then, his registration became visible through the list of participants, just like his father's, who in this case was the creator of the event.

As the days passed, some of Phillip's friends asked their parents if they could attend the event and thus, the number of participants started building up.

Finally, the date of the event came and everything went well. Both Joe and his son took a lot of photos and even filmed a video of the event. Some of the parents of Phillip's friends also did the same.

During the next few days, everyone logged into MySportsFace to publish their photos and videos. Joe uploaded his video to Youtube and linked it to the application afterwards. Meanwhile, Phillip uploaded the photos through the application directly. By that time, various other users had already left their comments about the event, too.

1.4 Thesis Outline

In Chapter 2, a list of various sports websites and applications are going to be described and commented on. Also, a set of criteria is going to be defined in order to evaluate each of the chosen websites and applications, regarding their positive and negative aspects. At the end of the chapter, an overview of all the evaluated items is presented along with some final considerations.

In Chapter 3, the reasons for choosing certain programming tools and technology over other options are explained in detail. Additionally, the most important aspects of the chosen technologies and tools are also described thoroughly, in order to understand the application's code distribution and organization. Chapter 3 also presents and describes a complete list of all the external libraries and Ruby Gems used to build the MySportsFace application.

In Chapter 4, both the initial architecture of the MySportsFace application and the current on are described, with emphasis on each module the application contains as well as the downsides of the initial design.

Chapter 5 showcases the current version of the MySportsFace application, describing each and every part of the application, from user profiles, relationships, posts and comments, to news articles, custom events and sports organizations. Furthermore, the chapter also goes into detail on some of the application's constraints, the deployment process and a possible integration with Amazon Web Services.

In Chapter 6, the details of each testing phase conducted, both in terms of integration and unit testing in general, as well as the usability and satisfaction tests with the help of users, are described and the results commented on and compared.

Last but not least, in Chapter 7, the thesis ends with the overall conclusions on achievements and future work.
Chapter 2

Sports Applications and Social Media

In order to create an application with the most suitable features, other sports organization’s websites and applications were studied and evaluated against a set of requirements, and their detailed descriptions can be found in the sections below.

The MySportsFace application has various architectural and functional requirements which will translate into a number of criteria used to evaluated each of the social media websites and applications studied. The criteria used to evaluate each website and application was the following:

- Users are able to become registered members of the application or website and possess an editable profile;
- Provides a way of showing event results, standings and information in general;
- Users are able to sign up for events;
- Allows users to view, upload and share multimedia content (images and/or videos);
- Contains news articles and posts, where users may leave their comments.

Additionally, below is the list of websites and applications evaluated and used as reference for the MySportsFace application:

- Website for Fédération Internationale de Ski (FIS)
- Website for NASCAR
- SPORT TV Multiscreen Application
- BeSoccer Application
- Website for Ski Clube de Portugal
- Website for Federação de Desportos de Inverno de Portugal (FDI-PT)
- Website for Federação de Triatlo de Portugal (FTP)

In the end of this chapter, an overview of the qualities of each website and application will be put together in a table for a better analysis of the studied material and to extract a few more conclusions.
2.1 Website of Fédération Internationale de Ski (FIS)

Starting with the website of Fédération Internationale de Ski (FIS), one can notice at first glance that it holds a lot of information [2]. The layout of the website is presented in Figure 2.1, and shows various recent news articles, as well as links to photo galleries and other sections of the website.

![Website Layout](image.png)

**Figure 2.1: Layout of the website for FIS-SKI.**

The homepage is heavy to load up, mostly due to providing users with a lot of information on events or competitions and news related to skiing in general. In this website, the users can perform multiple tasks like, for example, checking the photo galleries for recent events (as shown in Figure 2.2) or, with the help of the navigation bar (represented in Figure 2.3), check results, calendars, standings and even statistics of all competitions worldwide, as long as they are ruled by the federation or its affiliates.

This website already respects some of the criteria defined earlier, more concretely, the one that specifies that users should be allowed to view results and standings of competitions and events and, to some degree, the criteria that specifies that users should also be able to view, upload and share content online, such as pictures and videos.

However, the gallery is not very intuitive when working with a personal computer without a touch-screen component, because there is no way for the user to realize and understand that, in order to move to the previous or next pictures, one must use the keyboard, and press either the Left or Right Arrows, which makes the task of looking at the galleries a lot more time consuming.

On the one hand, the navigation bar comes in handy, due to the fact that it contains multiple sources of information for every type of skiing competition and event, including results, standings, upcoming events and the rules for each type of event, among other things.

On the other hand, the contents of the navigation bar are not organized properly in order to facilitate the navigation of a new user throughout the website. There is an overload of options for each different
type of competition, which leads, once again, to a slow completion of the task of searching the website for the desired information.

Secondly, there is the case of a user who wants to search for some event or athlete in specific. Although the site offers this service, along with a lot of search options, it does not let the user freely organize the results shown, using their own criteria.

Thirdly and lastly, as a way to promote the federation itself, it also relies on the most famous current social hubs, such as Facebook, Twitter, Youtube and Google+, to broadcast news, events and share videos or pictures, online.

2.1.1 Evaluating the Website of FIS

A list of the positive and negative aspects of this website can be found below. The positive aspects are:

- Users can navigate through the website to find multimedia content and share it.
- Users can search for results, standings and information of any competition or event.
And the negative aspects are:

- Lack of usability of the photo gallery part of the website.
- There is no easily visible information regarding how to become a member, only a semi-hidden member section, for already active users.
- The events are view-only, making it impossible to sign up for them.

2.2 NASCAR Website

Regarding the NASCAR website [3], there are plenty of positive things going in its favor. First, in terms of usability, the content of the website loads fast enough so that users do not have to wait a long time. Figure 2.4 shows the frontpage of the website, which has a navigation bar to every other section of the website, as well as a list of the most recent news articles.

Figure 2.4: Layout of the website for NASCAR.

Secondly, it already provides the users with many of the aforementioned requirements, such as the capability of searching for results of events and competitions (as seen in Figure 2.5) or even overall standings and profiles for each racer. On the other hand, it does not possess any way of signing up for events and races, since those are reserved for the racers themselves.

Unlike FIS, NASCAR is not an organization that any user can be part of, but is instead a company that uses its website to broadcast all the recent news, events and results. Nevertheless, any person
can still become a registered user at the website, which enables the use of additional tools, such as commenting news articles.

Finally, the website does not provide ways for its users to perform some social actions, such as sharing multimedia content. It does, however, let registered users comment on any news post.

2.2.1 Evaluating the NASCAR Website

With this said, the NASCAR website has a few pros and cons. The **positive aspects** include the following:

- Fast loading up of the website content.
- Allows searching for results of events, standings and news articles.
- Provides an easily accessible way for users to perform registration at the website.
- Registered users can comment on news posts.

The most important **negative aspect** is:

- Users cannot upload or share videos and images to or from the website.

2.3 SPORT TV Multiscreen Application

As regards the SPORT TV Multiscreen Application [4], there are a few things that can be highlighted, in what refers to the previously defined criteria, such as the fact that the application focuses mostly on giving the users a way to be up-to-date with everything that is happening in the sports world. Therefore, the application provides a way for each user to view live sports games and news on the go, through any mobile platform or personal computer. It also allows users to read any overview of a past game, which is one of the goals of MySportsFace, regarding the possibility of viewing results and standings, among other information, through a collection of news articles (as shown in Figure 2.6).

Moreover, the application provides the users with a collection of videos belonging to various different sports, which contain past matches according to the type of sport selected. Although the application
fulfills some of the criteria requirements, regarding the availability of multimedia content, it does not provide a way for users to comment on any of the matches, currently being shown live or in the past.

Additionally, in order to access the features in this application, the user must subscribe to the SPORT TV channel.

With this said, the SPORT TV Multiscreen Application was created to keep users on par with all the broadcast sports matches around the world, as it has been mentioned before, and therefore, the application lacks features to support the remaining criteria defined earlier in this chapter, e.g. signing up for events and viewing news articles.

2.3.1 Evaluating SPORT TV Multiscreen Application

The list below summarizes the best and the worst aspects of the SPORT TV Web Application. The positive aspects are:

- The application provides a good way for users to search information about any sport it covers.
- The application also provides a way of viewing any videos uploaded to it.

And the negative aspects are:

- It does not possess a way for users to write comments on any kind of content.
- It lacks a way for users to upload or create their own content in the application.
- Users may not directly become registered members through the application, as they can only access it by subscribing to the SPORT TV channel beforehand.
2.4 BeSoccer Application

The BeSoccer application [5], similarly to the SPORT TV Multiscreen Application, is used to mainly show results and standings of teams in the world of football.

However, in this case, BeSoccer also introduces some other social features, like allowing users to write comments on each match (see Figure 2.7).

Figure 2.7: Interface of the BeSoccer application where users can leave comments, during an ongoing or past broadcast sports match.

The application also comes with an additional functionality for placing bets on who the winning teams will be for each game, which in some way is a good form of stimulating socialization through the use of fun, as long as the bets are purely fictional, awarding only virtual prizes in the application.

In addition to this, BeSoccer also shows in detail each player transfer that occurred at a certain time for a football club, which can be related to the criteria defined before that suggests websites and applications like these should provide a way of searching through the standings and results of an event.

On the other hand, there are other crucial aspects missing in the BeSoccer application, such as the capability of uploading images, videos or any other multimedia content and it also does not provide any information in the form of news articles on the sports covered by BeSoccer.

Lastly, BeSoccer also fulfills the criteria which demands that users should be able to register at the application, since it is a necessary step in order to write comments.

2.4.1 Evaluating BeSoccer Application

To conclude, the following aspects of the BeSoccer application were identified.

The **positive aspects** are:

- Well-structured sports match information, with all the lineups and scores.
- Allows users to leave comments during the occurrence of matches or afterwards.
- Users of the application must register before using it.

And the **negative aspects** are:

- The application does not provide news articles regarding football (in this case).
- It also lacks a mechanism for sharing and uploading multimedia content to and from the application.

### 2.5 Website of Ski Clube de Portugal

Ski Clube de Portugal [6] is an organization (club) affiliated with Federação de Desportos de Inverno de Portugal (FDI-PT) and is much smaller than Fédération Internationale de Ski (FIS). However, it holds some of the same objectives and goals, which means that its members have to annually pay their fees by performing a bank transfer to the specified account number presented in a Google Form provided by the club. Even so, the applicants to membership have to be approved by the club board before becoming federated members.

The website for Ski Clube de Portugal has a layout similar to the ones from FIS and NASCAR, including a series of news articles and then the usual navigation bar at the top, as seen in Figure 2.8.

![Figure 2.8: Layout of the website of Ski Clube de Portugal.](image)

Furthermore, the section with multiple news articles does not allow users to write their comments, which goes against one of the criteria defined earlier, in which each user should be able to view and leave comments on the news posts.
On the other hand, Ski Clube de Portugal also does not possess any way of checking for standings and results of events, another criteria that is not met.

To conclude, if the organization actually promoted or was in charge of some events, it should also have a way of letting users sign up online for them, which is not the case. Therefore, the only criteria fulfilled by Ski Clube de Portugal are: user registration and the availability of news articles.

2.5.1 Evaluating the website of Ski Clube de Portugal

To sum up, there are some good factors to follow from the "under development" construction of this website, but there are also many other things which do not respect the criteria defined before. The positive aspects include:

- The website provides a good way for users to enroll for certain activities, such as being part of the Ski School or just being registered members, by filling in Google forms.
- The website contains news articles that users can view.

And the negative aspects are:

- The website does not provide functionality for searching through results and standings of events.
- There is no way for users to leave comments regarding anything they have seen in the news or events.
- Users cannot share or upload any multimedia content.

2.6 Website of Federação de Desportos de Inverno de Portugal (FDI-PT)

Federação de Desportos de Inverno de Portugal (FDI-PT) [7] is an organization associated with a number of clubs, such as Ski Clube de Portugal that was mentioned in section 2.5.

This sports organization possesses a website that not only includes multiple sources of information and multimedia content available to the users, but also offers some other services, such as enrolling in specific events online or becoming affiliated members of the federation. Figure 2.9 shows the layout for the homepage of the website, which has a minimalistic style and can be scrolled down to find more information regarding recent and upcoming events. The colors of the website (shades of blue and green) unfortunately do not help users read the information which is mostly colored in a white font.

Both the enrollment and the affiliation processes are done by filling in forms (one for each process) on the organization's website, which require personal information to be filled in and an additional declaration of payment of the fee to enter the competition (in the case of enrolling for an event), the sports medical examination results, an up-to-date photo of the user and also the payment of the annual fee and the insurance policy (in the case of wanting to become a registered active member).
Additionally, users can browse the "Media" section of the website in order to find news articles and the images or videos attached to them. Users can also share any article or "Like" and "Tweet" images (as shown in Figure 2.10) through the social networks, as is the case of Facebook and Twitter.

However, users cannot write comments on the news articles or check up standings and results of sports events promoted by the federation.

2.6.1 Evaluating the website of FDI-PT

To conclude the topic on Federação de Desportos de Inverno de Portugal, one can mention that it provides a series of services which follow some of the previously defined criteria. The positive aspects are:

- Users can enroll for some events online.
- Users can become registered members.
• It allows users to share multimedia content.

The negative aspects of the website are:

• Users are unable to leave comments on the news articles.
• It does not allow uploading images or videos into the website.
• There is no way for users to check results and standings of events.

2.7 Website for Federação de Triatlo de Portugal (FTP)

The website of Federação de Triatlo de Portugal (FTP) has a very aesthetic and organized layout, as seen in Figure 2.11 [8]. It has various sections for all kinds of content, ranging from federation information, event results and news articles to multimedia content and registration forms for events and for requesting membership. Additionally, the website uses JavaScript libraries to add animations to elements of the HTML pages, making the website visually more pleasant.

The website contains a section which allows users to perform their registration as athletes. Figure 2.12 shows the required form to be filled to become a non-federated athlete. Once a user becomes a non-federated athlete, they can sign up for events.

Furthermore, the website contains a section which shows recent events held and upcoming ones. By clicking on the links to these events, users have access to the details of the events (date, time, place, etc.) and also regulation PDF documents when available.
The website also contains multiple news articles that users can view and share on most social networking websites, but does not let users write comments.

Lastly, there are also links to multimedia content pages, such as their Youtube channel for videos and their Facebook page for photos and images, that users may visit. On the other hand, users are not able to upload their own multimedia content to the website or other platforms.

2.7.1 Evaluating the website of FTP

The website of Federacação de Triatlo de Portugal is the one which provides users a close to ideal set of features for a sports web application. Its positive aspects are:

- Users are able to become non-federated athletes through the website.
- The website contains news articles and event information viewable by users.
- It also provides links to external multimedia content pages.
- Users can sign up for events if they become non-federated athletes.
And the **negative aspects** are:

- Users are unable to write comments on the news articles.
- Users cannot upload multimedia content anywhere in the website.

### 2.8 State-of-the-Art Summary

To summarize, while some of the investigated websites and/or applications fulfill some of the pre-requisites defined at the start of this chapter, such as the capability to sign up for events and competitions, the remaining ones focus more on the other social aspect of federations, with the capability of allowing users to browse content (news, images and videos) on the website or application and in some cases even letting users leave comments on the news articles. However, there is no website or application that fulfills all the criteria, the desired goal for the MySportsFace application. Nevertheless, it is important to notice that due to the nature of the organizational websites, some of the criteria that focus on user interaction with website content may never be available.

Table 2.1, shows a summary of all the criteria that each website and application follow (partially or fully), and the ones they do not.

The criteria have been represented by their numbers, which are also shown below.

1. Users are able to become registered members of the application or website and possess an editable profile;

2. Provides a way of showing event results, standings and information in general;

3. Users are able to sign up for events;

4. Allows users to view, upload and share multimedia content (images and/or videos);

5. Contains news articles and posts, which users may comment on.

<table>
<thead>
<tr>
<th>Websites/Apps / Criteria</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIS-SKI</td>
<td>O</td>
<td>✓</td>
<td>X</td>
<td>O</td>
<td>✓</td>
</tr>
<tr>
<td>NASCAR</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>O</td>
<td>✓</td>
</tr>
<tr>
<td>SPORT TV Multiscreen App</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>BeSoccer App</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>O</td>
</tr>
<tr>
<td>Ski Clube de Portugal</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>FDI-PT</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of all the criteria respected by each website or application reviewed.

**Legend:**

- ✓ - The application/website fully respects the criteria;
- O - The application/website partially respects the criteria;
- X - The application/website does not respect the criteria.
Chapter 3

Selecting the Technology and Tools

In order to achieve what the MySportsFace application is today, various programming languages and tools were needed. Thus, in this chapter, all the choices made are described and then compared to the other possibilities which were not adopted.

There are various frameworks for web development, such as Ruby-on-Rails, Python's Django, Microsoft’s ASP.NET, Zend, AngularJS and EmberJS. All of these adopt a Model-View-Controller (MVC) architecture. This architectural pattern is advantageous for building graphical user interfaces and web applications, because it divides any software into three different components:

- The **models** which contain information regarding the various states of the application;
- The **controllers** which send commands to update a model's state or to change the presentation of a model;
- The **views** generate output presentations to the user based on changes in the models;

![Diagram of MVC components]

Figure 3.1: Typical collaboration of the MVC components.

Figure 3.1 shows the collaboration of the components of a MVC architecture.

For this application specifically, Ruby-on-Rails was the chosen framework, so it will be described how this framework is organized and the benefits of using it over Python’s framework, Django, Microsoft’s
ASP.NET and Zend. AngularJS and EmberJS are left out of the comparison because they had very few open-source plug-ins and libraries at the start of the development of MySportsFace.

As it will be seen, Ruby-on-Rails was chosen not because the others had various downsides, but instead because Ruby-on-Rails comes with several added benefits and extra features that the rest lacked and a much larger community with open-source projects.

Furthermore, since the application also requires a working database and a web server, the choices available at the time are also going to be presented, and then explained why they differ depending on the environment being used, in regards to the development and production environments.

Lastly, this chapter will also present what the Model-View-Controller (MVC) architecture brings to Ruby-on-Rails, the application's routing and resources and the different coding languages used to attain different objectives.

### 3.1 Microsoft ASP.NET

There are various commercialized tools that could be used to implement a web application such as MySportsFace. One that is used nowadays by web developers is Microsoft ASP.NET [9], which is a web framework for building applications of any kind.

ASP.NET has various qualities and the most important ones are listed below:

- Supports code written in .NET (which is C# and thus similar to Java);
- Provides a Model-View-Controller architecture with WebForms;
- Already comes with a few AJAX script libraries;
- Supports client-side and server-side programming;
- Microsoft Visual Studio can be used as an editor software for developing ASP.NET applications, providing various quality of life features.

However, when creating a new project using ASP.NET, the developer will have to do most things manually, such as creating files and selecting the correct location for each one of them, in order for the application to work properly (which is not the case for Ruby-on-Rails or Django). Additionally, there are very few open-source projects and external code libraries that developers can add to applications, making it less desirable to use ASP.NET. Additionally, due to the lack of plug-ins, ASP.NET provides poor theme and template management.

Nevertheless, as will be seen in the next sections, there are other commercial solutions that are more popular and suitable for creating web applications.

### 3.2 Zend Framework

Similarly to the remaining frameworks, Zend Framework [10] is an object-oriented framework with a MVC architecture, based on the PHP coding language.
Zend Framework supports almost every type of database, such as: MySQL, SQLite, PostgreSQL, Oracle, IBM DB2 and Microsoft SQL Server.

To provide developers with a large number of built-in code libraries and extensions, Zend Framework can be used by Zend Server, which is a PHP distribution stack designed to increase performance and productivity. Some of these extensions include:

- Page and Data Caching
- Job Queues
- Code Tracing and Insight
- Clustering
- Web Server and Deployment Support
- Software Updates and Hotfixes

Unfortunately, Zend Framework does not provide tools for automated file generation and database management, like Ruby-on-Rails or Django. Also, there are few templates for building websites without having to start them from scratch.

However and lastly, Zend Server is a paid commercial tool, therefore discarding Zend Framework as the choice for building MySportsFace.

### 3.3 Ruby-on-Rails vs. Python’s Django

The two most used web frameworks for building applications (before the popularization of AngularJS and other Node.js frameworks) are Ruby-on-Rails [11] and Python’s Django [12].

Django encourages a rapid and clean development and pragmatic design, by helping developers take their ideas and turn them into applications as quickly as possible, since it is an averagely easy framework to work with, thus making it suitable for people under strict timelines.

Moreover, Django is also built in a way that makes its users less prone to create security issues in their applications and provides ways to counter many security hacks, such as Clickjacking, Cross-Site Scripting, Request Forgery and SQL Injection.

Django’s goal is to ease the creation of database-driven applications and emphasizes reusability and "pluggability" of components. Similarly to Ruby-on-Rails, the framework also comes with dynamic and automated file generation, depending on the needs of the developer.

Last but not least, Django is built around a “shared-nothing” architecture, meaning that extra hardware can be added at any level, be it for database servers, caching servers or web servers, therefore providing a scalable framework.

On the other hand, Ruby-on-Rails provides the same scalability, through the specification of what kinds of database and web servers the application will run with, the same security mechanisms and the
same versatility that Django has, by using some of the many open-source plug-ins and API's created by the Ruby on Rails community.

On top of this, it also adds a superior testing environment and a more simplistic routing table that is based on resources and the purposes they may have.

Another selling point of Ruby-on-Rails is its code syntax, that is ideal for developers who prefer pattern-matching characters, because it uses differing characters as analogues to keywords, unlike Django, which has a syntax that resembles the English language.

Lastly and most importantly, Ruby-on-Rails was chosen due to being a lot more beginner friendly, as its community provides developers with a very large number of tutorials, guides and plug-ins. Also, Ruby-on-Rails is adopts a Convention over Construction (CoC) philosophy, meaning that it has a very well structured layout with default templates that make it easy to create applications on the go and run them quickly (like having a web starter kit for developing applications).

Nevertheless, as a side note, Django is still often a better choice for building analytic and scientific programming systems. However, that is not the case of the MySportsFace application.

3.3.1 Ruby-on-Rails Project Structure

Since the choice to use Ruby-on-Rails was made, its file structure for projects and its components have to be described.

The root directory for any Ruby-on-Rails project has the following folders:

- **app**: Where the front-end code used to build the application is stored. This folder contains the model definitions, the controller specifications and the application views. Additionally, it also contains helper classes, which function as interfaces, making any defined functions available to the remaining code;

- **bin**: This folder only contains the three executable programs required for developing any project with Ruby-on-Rails. These programs are *rake*, *rails* and *bundle*, which will be described further in this chapter;

- **config**: In this folder, the programmer can configure how the application is launched, how each environment is defined and also provides support for multiple locales (multiple languages, internationalization). Last but not least, this folder also contains the routing table for the website pages and its resources;

- **db**: Where all the database migrations are stored and where the programmer can define a seed for the application, so as to populate the database with records, for the purpose of testing or initialization;

- **lib**: Contains code that is not specific to the domain of the application. This code is loaded every time the application is booted up;

- **log**: This folder is used to maintain a log of the application for debugging purposes;
- **public**: All of the custom pages for showing HTTP responses and the website “favicon” (icon shown in the browser tab) are stored in this folder;

- **test**: This is the folder provided by Ruby-on-Rails to create a testing environment. In here the programmer can specific any number of unit tests with the purpose of testing models, controllers and helpers, along with creating integration tests for everything added up together;

- **vendor/assets**: Lastly, this folder holds any type and form of third-party code or libraries, which are used by the project in some way, but not directly in the form of a Ruby gem.

Moreover, Ruby-on-Rails projects also have a file called **Gemfile** which is where the programmer defines all the Ruby gems to be used by the application and a folder within the **app** directory, which holds the application-specific assets. The purpose and effect of these gems will be described in Subsection 3.3.2 and the type of assets used is shown in Subsection 3.3.3.

Figure 3.2 shows a diagram of Ruby-on-Rails projects’ folders and the coding languages they usually contain.

![Diagram of Ruby-on-Rails projects' folders and coding languages](image)

**Figure 3.2: Programming language distribution in a Ruby-on-Rails project.**

### 3.3.2 Ruby Gems

As mentioned before, any Ruby-on-Rails project contains a varying number of Ruby gems, explicitly stated in the **Gemfile** file, which are projects created by other programmers in the form of Application Programming Interfaces (API's). These API's help build a complete web application without having to program every single aspect of an architecture and its requirements from ground zero. This also helps the programmer by being able to freely add or remove modules as needed, therefore, creating an extremely modular application that prevents code repetition.
The **Gemfile** file most of the time also contains information regarding which type of database and web servers are used (as they can easily be added or removed in the form of Ruby gems too).

An extensive description of all the gems used for the MySportsFace application and the reasons for each one of them being used, is presented in Section 3.7.

### 3.3.3 Assets

Aside from the Ruby gems, the web applications often require a variety of additional assets. These assets can be images, videos or code for scripting and styling the website. For this specific application, various images were created using Adobe Photoshop CS6, to add a logo to the website, background images, icons and sports organizations logos.

The assets regarding styling and scripting will be described at a later stage in Section 3.6.

### 3.4 Choosing a Database and Web Server

Ruby-on-Rails provides multiple coding environments, one for development, one for testing and one for production.

For each environment, the programmer may configure various aspects of the application, such as which type of database to run and what kind of web server should be used. These configurations are as simple as adding some conditions to the **Gemfile** file previously mentioned in this chapter. Therefore, in this subsection, a detailed explanation of the various kinds of databases and servers used will be presented and the reasons for using each of them in given environments.

#### 3.4.1 Development Environment

During the development of the application, the environment uses a non-scalable web server (WEBrick) and a database with reduced operations (SQLite [13]). The reason for this is that, locally, the website traffic is negligible, as there is only one user online at all times and the required database operations are sufficient by using a lighter database than a MySQL database.

This way, the development environment is easier to manipulate, but still keeps all the required functionality to make the application work as a whole.

#### 3.4.2 Production Environment

When moving towards a more crowded environment, where a very large amount of users may be viewing the website at the same time, the previous web server (WEBrick) is not viable anymore. Thus, in order to configure the application to be scalable, a better and faster web server, called Puma, had to be used. Puma is built for speed and concurrency, which makes it a solid choice for social networking applications [14]. In regards to the type of database used, it solely depended on where the application would be
deployed, which in this case was to **Heroku**, a cloud service company that holds applications or other projects in general and allows them to be executed within their virtual machines.

A better explanation of the deployment step is described in Chapter 5.8. Heroku has some constraints on the kinds of databases that can be used in their virtual machines, more specifically only allowing PostgreSQL.

However, the differences between MySQL, SQLite and PostgreSQL are not significant, since the migration from one type of database to another can automatically be done when deploying. Locally, it has been stated that the migrations are done in the form of a SQLite database whereas, during production, all the tables and records are migrated instantly to PostgreSQL. This is possible because Ruby-on-Rails uses Object Relational Mapping system called ActiveRecord (representing “M” in the MVC architecture), which parses the migrations written in Ruby and transforms them into code for the chosen type of database.

### 3.4.3 Migrations

Every application written in Ruby-on-Rails that wants to be modular and present dynamic content is required to have a set of models. Each model can be added by creating new migrations that add or change tables in the database. These migrations can be done by simply calling the `rails` command on the command line a few extra arguments after it.

In Figure 3.3, is an example of a rails command in the prompt line, creating a migration file:

```
>rails generate migration CreateUsers name:string email:string
```

![Figure 3.3: Example of a command creating a new migration in Ruby-on-Rails.](image)

In this case, a new table called "Users", with columns "name" and "email", is added to the database, but it is also possible to add or remove columns from an existing table. For that, instead of writing "Create<X>"", where X is the table name, the programmer writes "Add/Remove<Y>To/From<X>", where Y is the column name to be added or removed and X is the table name to add or remove from.

### 3.5 Model-View-Controller as per Ruby-on-Rails

As Ruby-on-Rails is a MVC and Object-Oriented programming language, the creation of content within any application developed with it is divided into three steps. First, the programmer has to create a model for the type of resource or class to be used. Then, in order for that model to be used and shown on a web page, it has to pass through a controller, where the programmer determines which pages have access to what type of content of the model. At the end, all that is left is for the developer to code the content and style of each web page, through the creation of as many views as necessary.
3.5.1 Models

A model in Ruby-on-Rails is somewhat an equivalent to a class in other Object-Oriented languages, such as Java or Python, except that it lets the programmer specify model constraints and restrictions, which can be used later on to validate each object of that model. Much like in Java, the model files may contain any number of methods which are used to get or set content of objects, along with other methods to inspect the objects.

On the other hand, unlike Java, the relations between models are specified inside each model file. For example, if a model "User" can have many instances of a model "Post", then inside the file for the model "User" there has to be a line saying "has_many :posts". Likewise, the "Post" model has to state that it belongs to a "User", by adding "belongs_to :user".

Last but not least, the models can also be programmed so as to specify dependencies for deleting objects, which means that when an object with dependencies is deleted, every other object depending on that one will also be deleted.

3.5.2 Views

The views are the HTML files used in Ruby-on-Rails to configure the content in each web page of an application, or part of a page (partial view).

As stated, these files are written in HTML, but they also have the characteristic of being able to contain Ruby code, used to extract or present information from the models and to make use of the helper methods.

In general, a Ruby-on-Rails application has a “main view”, which contains the exoskeleton of the web pages and inside that view, the programmer is allowed to render any other views as required. This effect is accomplished by adding a keyword "yield" to the code of the "main view". That will allow the Rails engine to recognize that some other view file should be shown in the place of that "yield", depending on the routing and resources defined in a routing table.

Additionally, various partial views may be created so as to simplify code that would be repeated otherwise. This way, the application becomes more flexible and manageable in terms of code changes and re-factoring.

Lastly, each view may also contain code to add extra scripting or styling (in Javascript or CSS) to the content of that specific page only.

3.5.3 Controllers, Routing and Resources

In order to talk about controllers in Ruby-on-Rails, the routing and resources system has to be described. Every Ruby-on-Rails project contains a "routes.rb" file which is used to specify the HTTP protocols (GET, POST, PATCH and DELETE) used for each web page and what kind of resource is allowed to be shown on the pages.

A resource can be used in various ways: creation, deletion, indexation and showcasing. These four different options to use a resource are the default ones in Ruby-on-Rails, despite the fact that additional
options may be added. More specifically, the create and delete options are generally used to create and delete instances of a model, the index option is used to show multiple instances of a model and, finally, the show option is used to show a single instance of a model.

Below is an example of how a resource is defined in the routing file:

```ruby
resources :events, only: [:index, :show, :create, :destroy]
```

Figure 3.4: Example of a resource specification in a Rails routing file.

Figure 3.4 shows a type of resource called “Event” specified to be available only in pages which show, create, delete or index events. If no restraints are defined, the resource will be available for any page that makes use of events, as long as the controller method responsible for the page provides the information.

Moreover, as stated before, the controllers will then contain methods for each one of the options described in the routing file and other custom ones, as well. Once the application boots up, the “routes.rb” file is loaded and a series of web page paths (links) become available to users.

These paths are also useful for the augmentation of views, as they can be passed unto various Rails helper methods, such as "link_to", so as to create links between pages dynamically, instead of hard typing the link to the pages.

### 3.6 HTML5, CSS3.0 and JavaScript

The code for developing the MySportsFace application views can be divided into three big categories: front-end code written in HTML and a small portion of Ruby, script code written in JavaScript and style code written in CSS. Therefore, this section will present a detailed description of the various types of code used and where they are used.

#### 3.6.1 Front-End Code

The front-end code represents everything that is written in HTML and Ruby, and is the part of the application which provides visible information to a user, in the form of web pages. The web pages not only contain HTML elements, but also contain information retrieved through Ruby methods, directly from what the page controller specified to become visible. Any variable defined through the page controller (variables are usually defined with a `@` right before its name) becomes available to the respective view and can be accessed by encapsulating the code between "<%" and "%>", still inside the HTML view file.

#### 3.6.2 Scripting and Animations

However, the static code alone is not enough to make a brand new site look polished and professional. For that matter, there are scripting languages such as JavaScript, which was used in this project, to add
dynamic information changes and "smoother" animations to the website. Ruby-on-Rails projects contain a main script file which can be configured to integrate multiple other script files as required, as well as it can load everything within a "tree" dependency.

For example, in this project, the main JavaScript file includes a series of other files, which are loaded in order, from top to bottom:

```
// = require jquery
// = require jquery_ujs
// = require bootstrap
// = require_tree .
```

Figure 3.5: List of included JavaScript files and code libraries for MySportsFace.

Figure 3.5 shows the order of included JavaScript libraries used in the project. First, all of the JQuery script codes for smoothing animations are included.

Afterwards, an additional package for adapting JQuery code ("jquery_ujs") is included, in order to add extra flexibility to HTTP requests from hyperlinks and to asynchronously submit form and hyperlink information with Ajax.

Then, all of the code from Twitter’s Bootstrap API is included to augment the interactivity of the website with pre-existing scripts.

Lastly, every remaining JavaScript file in the current directory is included, through the use of the "require_tree ." line.

### 3.6.3 Styling the Website

Last but not least, after everything has been coded and scripted, the information visible to a user on the website probably still looks plain and very unattractive. This problem is easy to overcome by styling the page elements with CSS code.

For this project in specific, Twitter’s Bootstrap base stylesheets were adopted, so as to have some baseline looks to work with. Afterwards, various classes within the sheets were changed in order to attain a newer, distinguishable style in the website. All the changes were made in a file called "custom.css.scss", which is loaded in the same way the scripting code was, through a main stylesheet file. This file loads Bootstrap’s stylesheets and then loads the custom-made code which overwrites the original.

To conclude this chapter on the programming environment for the MySportsFace application, it is also important to state that with CSS3.0, a few styling variables are able to perform animations which only JavaScript was able to previously perform, such as adding transitions to any style class element. This feature was used in the application to stylize interface links and buttons.
3.7 Ruby Gems and Libraries

This section presents a detailed description of each ruby gem and JavaScript library included in the application to support its requirements and to better develop the application.

3.7.1 Superfeedr Engine

In order to extract the necessary news articles and information from all the sports organizations supported in the MySportsFace application, a RSS feed consumer engine had to be used. In this case, the “Superfeedr Engine” gem provided an extremely fast and reliable way of grabbing website content and publishing it on the application [15].

After including the gem in the application, a configuration file was created, to specify the communication process and the web hook used.

```ruby
SuperfeedrEngine::Engine.feed_class = "Feed"
SuperfeedrEngine::Engine.base_path = "/superfeedr_engine/"
SuperfeedrEngine::Engine.host = "mysportsface.herokuapp.com"
#
#
SuperfeedrEngine::Engine.login = "Marcos"
#
#
SuperfeedrEngine::Engine.password = "f41a633f6de4d2395a650ea65e736409"
#
#
SuperfeedrEngine::Engine.scheme = "http"
#
#
SuperfeedrEngine::Engine.port = 80
```

Figure 3.6: Superfeedr configuration in MySportsFace.

Figure 3.6, shows the configuration file used in MySportsFace. The file includes a username and a token with subscribe, unsubscribe, retrieve and list values (called password, even though it is not sensible information), along with the Ruby model/class to be used as a feed store and the hostname which the Superfeedr engine will use to communicate with, whenever there is new content to be retrieved.

Therefore, the API provided by Superfeedr includes three helper methods that deal with RSS feeds, "subscribe", "unsubscribe" and "retrieve". The first one, as it states, links the application to a specific feed and allows it to receive asynchronous notifications of new content available, while the "unsubscribe" method does the opposite, stopping the income of notifications. Lastly, the "retrieve" method is used to get all the information through JSON once it is available.

This gem was the most popular method of retrieving RSS feed content without requiring polling and constant updates, as the content is retrieved asynchronously, when users make a GET request to access the sports organization’s pages.
3.7.2 JavaScript and CSS Code Optimization

Ruby-on-Rails allows for some code optimization, by adding CoffeeScript syntax to JavaScript files and Syntactically Awesome Style Sheets (SASS) syntax to the CSS files. These two additional libraries simplify the way scripts and stylesheets are written, when in comparison to the original syntax. Specially in the case of SASS files, which remove a lot of repeated and unnecessary code from the original stylesheets.

**SASS-Rails**

Syntactically Awesome Style Sheets (SASS) is a derivation from the original Cascading Style Sheets (CSS) code, and it brings up a series of quality improvements for styling elements and organizing the code itself [16].

![CSS code vs SASS code](image)

**Figure 3.7:** Example of the difference in code syntax between CSS and SASS, in terms of defining classes for elements and sub-elements.

Figure 3.7 shows the syntax differences between code written using the original CSS language and code written using SASS, in terms of defining classes and subclasses in a stylesheet. While in CSS a new set of rules has to be defined in order to manipulate HTML elements inside of a specific class, SASS-Rails simplifies the process by allowing programmers to define properties for those elements inside the code of the given superclass (element that they are contained in).

Moreover, dynamic element properties, such as ":hover", ":focus" and ":active" can also be directly mentioned within its element's code brackets, instead of requiring a whole new set of rules, as seen in Figure 3.8.

![CSS code vs SASS code](image)

**Figure 3.8:** Defining dynamic properties, such as ":hover", using traditional CSS code and SASS code.

Lastly, when using SASS, one can also specify variables with any type of content (numbers, colors,
strings, etc.) or even import code from multiple files by adding "@import <filepath>" anywhere in the CSS file, but it is a good coding practice to place these imports at the start of the file. This set of features avoids code repetition as well as it makes the code somewhat more modular and susceptible to changes.

**CoffeeScript-Rails**

CoffeeScript is used in the MySportsFace application for the same reason SASS is: to change the syntax of the programming languages and make code more manageable. In this case, CoffeeScript allows programmers to code JavaScript in a simpler way, by allowing certain conditions to be skipped and adding extra required JavaScript semantic to the code when compiling a CoffeeScript file. For example, a variable may not exist in a CoffeeScript file, but due to the nature of the JavaScript language, a user may mention any variable name anywhere and, if it appears right before a question mark (e.g. `balloon?`), then CoffeeScript interprets it as a "does it exist" statement and thus creates the necessary JavaScript syntax for the statement to work after compiling. Figure 3.9 shows an example of CoffeeScript code and the respective JavaScript code originated after its compilation, showing the previous mentioned functionality along other several examples and their respective JavaScript counterparts.

Overall, the CoffeeScript-Rails gem brings additional quality of life features towards coding, scripting in this case.

### 3.7.3 Bootstrap

Twitter’s Bootstrap API [17] has a big role in the MySportsFace application, as it provides a very big set of pre-styled CSS classes and elements, specially for HTML forms and buttons and, most importantly, the website layout in general. Bootstrap itself is divided into various separate Ruby gems, each providing extra functionality to existing Rails features or other Ruby gems, such as styling, forms and pagination.

**Bootstrap-SASS**

The "Bootstrap SASS" gem has the main goal of allowing the programmer to directly import all of the Bootstrap styling components into a SASS-powered application [18], which is the case of MySportsFace or any new Rails application since version 4.0.

**Bootstrap Forms**

Moreover, "Bootstrap Forms" is another gem which enables additional customization of HTML forms [19]. This gem wraps multiple Rails default form helper methods and allows extra customization in terms of form layouts, sizes, showing helper icons and even appending and prepending information to form fields. The wrapped helper methods also support everything that Rails did by default, on top of adding a few other options. Finally, any form created with the "Bootstrap Forms" gem can also be automatically be validated to check for errors and non-valid input.
Bootstrap Will Paginate

Last but not least, every application which is required to manage a lot of content and information needs a way to make it visually less heavy to the users, by adding content pagination.

The “Bootstrap Will Paginate” gem wraps the original “WillPaginate” ruby gem, which allows the programmer to specify a threshold to object instances to be shown in a page, and delegating the remainder to be shown in secondary pages, in order to prevent a single web page from having all the content and be too heavily loaded [20]. The wrapped pagination helper shown on pages is then changed to have a style appropriate to the remaining Bootstrap elements. The paginated content can also be organized in two or more different manners, such as following a DESC order (descending) or an ASC order (ascending),
for a specific model property (and its respective database table column).

```ruby
## perform a paginated query:
@posts = Post.paginate(:page => params[:page])

# or, use an explicit "per page" limit:
Post.paginate(:page => params[:page], :per_page => 30)

## render page links in the view:
<%= will_paginate @posts %>
```

Figure 3.10: Pagination in Rails using the "Will Paginate" gem.

Figure 3.10 shows the process of paginating a certain collection of objects through "controller" and "view" code, written in Ruby. The first two lines of code shown in the image represent code that one would insert into a "controller" file, for fetching the information with a query, while the last line of code is used in a "view" file, to show the pagination, either before or after the collection of objects.

### 3.7.4 Font-Awesome-SASS

Bootstrap on its own already provides a wide variety of icons to be used alongside the HTML elements of a web page, but certain icons are too edgy and did not really fit into the MySportsFace style after adapting Bootstrap’s original stylesheets. Therefore, the "Font-Awesome-SASS" gem was added to provide the application with an even larger number of icons [21].

![Icons from Font-Awesome-SASS](image)

Figure 3.11: Small set of icons from the Font-Awesome-SASS gem.

Figure 3.11 showcases a small number of the icons provided by the aforementioned gem. These icons can be added anywhere within the HTML views, by creating an element with the "<i> </i>" tag boundaries and adding the wanted class. The gem basically consists of a big picture which has pixel
coordinates tagged for each of the icons, so that when the user specifies a class for the tag, only the correct portion of the picture is shown.

3.7.5 Uglifier

The Uglifier gem is used to wrap JavaScript code with additional options that can change the way JavaScript is interpreted by default. It can also perform changes to the output code, by specifying a certain code indentation or allowing to "beautify" the code, along with a series of other options to normalize code representation after compilation. Furthermore, Uglifier also allows name mangling and code compression for better performance. A complete list of the options made available by Uglifier can be found on the author's repository at GitHub [22].

3.7.6 jQuery-Rails, Turbolinks and jQuery-Turbolinks

The “jQuery-Rails” gem is just a simple gem which adds the necessary JavaScript files to the application asset pipeline, so that JQuery code works when writing scripts [23]. Additionally, in order to make the website load faster, the experimental “Turbolinks” gem was initially included which kept the whole HTML body of a web page saved in memory, so that it could rapidly be swapped with that of another page for faster loading and reloading [24]. However, this process made it look like the web pages were not being loaded (from a browser perspective), while still swapping the content of the pages when the links are pressed.

Unfortunately, JQuery scripts are usually loaded when a page loads (through “jQuery.ready()”), but in the case of “Turbolinks”, the pages are loaded once and then never again, which breaks various scripts and stops them from working properly. Due to the nature of this issue, an additional experimental gem, “jQuery-Turbolinks” [25], was included to mitigate some of the problems related to other external JavaScript libraries included into the project, which are later detailed in this chapter.

As per stated in the “Turbolinks” github repository, not every script is immune to the issue even after including the aforementioned gem, which was the case for the MySportsFace application, in terms of a script used to create a dynamic object grid layout. This problem forced the later removal of the “Turbolinks” gem usage, which means that the website runs normally as of this moment, but can support “Turbolinks” if its author releases a stable “production” environment version of the gem for Rails 4.0.

3.7.7 SQLite 3, PostGre Rails and Puma

It has been previously mentioned that the MySportsFace application was tested in an environment different from that of its release. Thus, for the development environment, the “SQLite 3” gem was used, to serve as database for the application, while for the production environment and due to the constraints posed by the Heroku deployment website, the “PostGre Rails” gem was used instead. Both these gems perform their database storage duty in a very similar way and the differences are negligible when working with the Ruby-on-Rails framework, which does all the background work of supporting multiple database types.
In terms of web servers, and as stated in the previous chapter 3, the “Puma” gem was used to set up a fast, concurrent and scalable web server for the released version of MySportsFace. No gem was required to be specified during development, as Rails uses a default server, called WEBrick, which is suitable enough for testing with a small number of users.

### 3.7.8 Faker

The “Faker” gem [26] was very useful during internal application testing, as it allows the programmer to create fictional records and add them to the database. This gem provides helper methods that randomly generate user names, company names, addresses, emails and other type of data if necessary. Therefore, the “Faker” gem was exceptionally useful to fill up the database with a set of random users and sports organizations, in order to test profile layouts and organization information.

### 3.7.9 BCrypt

Every user that logs into the application is required to have an encrypted session and that is the job of the “BCrypt” gem [27]. It provides helper methods to create password hashes and a programmer can specify whether or not a model possesses a password by writing "has_secure_password". Afterwards, the user’s account can be safely checked upon logging in.

### 3.7.10 Uploads

A part of the MySportsFace application requires that users are able to upload their own images from their devices into the website. In order to achieve that requirement, three related gems were used, to enable file uploads, image transformation and cloud storage. Those three gems are “CarrierWave”, "MiniMagick" and "Fog", respectively.

#### Carrierwave

The “CarrierWave” gem enables file uploads in an extremely flexible way [28]. First, the programmer has to create an uploader file through a "rails generate uploader" command. Afterwards, all there is to be done is to specify the ruby model which will "mount" the file uploader and then, inside the desired HTML view, just create a form with a field created with the helper method "file_field". This set of instructions was used in the MySportsFace to create an image uploader that is shown in every user-customized event. This gem also works alongside the “Fog” gem, in order to store the files in a cloud-based service, such as Amazon S3.

#### MiniMagick

During the image uploading process, the “MiniMagick” gem can work with the file so as to perform a series of possible transformations on the image [29]. These changes include: re-sizing, cropping,
rotating, flipping and even composing multiple images. Any "MiniMagick" image has various attributes which can also be validated and checked to ensure that certain restrictions are met.

**Fog**

Last but not least, the "Fog" gem gives any Ruby-on-Rails application a cloud service library [30], which can be used to simplify the interaction between the file uploads and the servers where the content is stored. "Fog" is not a cloud storage server on its own, which means that another type of storage still has to be used, whether it is the file system on its own, a database, or an external cloud service like Amazon S3 and EC2 (Simple Storage Service and Elastic Compute Cloud). Dropbox, OneDrive and other cloud storage services were not considered because "Fog" does not support them and also because there are no other stable ruby gems that integrate Dropbox and OneDrive with Ruby-on-Rails applications.

For MySportsFace, the file system was used to ultimately store the images uploaded, among other necessary pre-compiled assets. Nevertheless, the application was tested with and is ready to use the Amazon Web Services (AWS) at any time, simply by adding the required ruby gem and configuring a series of necessary variables. The AWS setup is described in Section 5.8 of Chapter 5.

### 3.7.11 Rails Gallery

Once the images are uploaded and stored, they need to be showcased at the event's page. In order to find a way to stylishly show the images and group them up, the "Rails Gallery" gem was used [31]. This gem offers an easy way to create a variety of image galleries and is entirely written in JavaScript. The gem imports the necessary JS and CSS files to make the gallery work and, afterwards, all there is to do is choose the type of gallery to use and the element(s) to the HTML view file.

Regarding MySportsFace, the gallery used has a minimalistic style and groups images in groups of six. In the case the limit is passed, the remaining pictures are paginated and shown in other groups of up to six. The limit of six images per group is imposed due to layout space constraints, but also because it is a suitable amount of pictures for a user to see at a given moment.

### 3.7.12 Omniauth

Moving onward, another highly required feature these days is to log in to places using information from other social networks. Thus, rails has a gem called "Omniauth", which provides a "common authentication system" for multiple third-party credentials from websites such as Facebook, Google+, Twitter and even GitHub [32]. The usage of this gem implies the creation of yet another initializer file each type of credentials "provider" specified.

```ruby
Rails.application.config.middleware.use OmniAuth::Builder do
  provider :facebook, ENV['FACEBOOK_APP_ID'], ENV['FACEBOOK_SECRET']
end
```

Figure 3.12: Example of a credentials provider specified for Omniauth, in this case, Facebook.
Figure 3.12 shows a way to specify that the application may receive credentials from Facebook, by making sure that the application can use the Facebook API. For that, the programmer is required to add a new application at the Facebook developer section and make sure that they retrieve the respective "FACEBOOK_APP_ID" and "FACEBOOK_SECRET" variables. The latter one should never be made public as it would give access the application’s Facebook API to possible attackers.

**Omniauth-Facebook and Omniauth-Google-OAuth2**

"Omniauth" by itself only provides a framework for adding any number of authentication methods, but to integrate Facebook and Google+ authentication within the application, two additional gems ("Omniauth-Facebook" and "Omniauth-Google-OAuth2") were required, each adding the necessary helper methods to access the social network API's.

Additionally, it was necessary to add the respective callback paths/URLs to the routing file of the project, so that when a user clicks the "Log in with Facebook/Google" they get redirected to the correct page, and that the correct HTTP request is sent.

**DotEnv**

As seen in Figure 3.12, the variables were not directly stated in the code, but instead are being stored in two environment variables, controlled by the "DotEnv" gem [33]. This gem creates a virtual environment for Ruby-on-Rails that allows the developer to specify any number of environment variables containing sensitive and non-sensitive data, for usage in any other gems and Rails content. Therefore, and since the MySportsFace application allows users to log in with two different types of third-party credentials, the necessary Facebook and Google+ API access information is being stored in environment variables.

However, this gem is not necessary during the deployment of the application (for the production environment), because the Heroku platform already provides an environment for storing these variables, accessible and editable from a console. Consequently, the gem is set to be only enabled for local testing.

### 3.7.13 Thumbs Up

Most of the content displayed in the MySportsFace application is allowed to be voted on or, more specifically, "starred". To accomplish this feature, the "Thumbs Up" gem was added to the project and it provides a very simple and straightforward way to implement a voting system. All the programmer has to do is state which models/classes will act as "voters" or "voteable objects".

Afterwards, to vote on an object, there has to be an interface element with a link that performs a POST request with a +1 or -1 to the vote count of that object. This can be achieved by using the two first helper methods shown in Figure 3.13.

As seen in Figure 3.13, there are also two other methods available, which can perform any type of vote or also clear every vote from a specific voter.
3.7.14 Other Minor Gems

The following subsections mention two gems, “TZ Info Data” and “Rails12Factor” which bring minor improvements to the application.

TZ Info Data

The “TZ Info Data” gem contains the modules necessary for a Ruby-on-Rails to have information regarding every timezone. If the gem is installed, the application will use its data as source of timezone information. Otherwise, it will attempt to use the system zone info files instead. Regarding MySportsFace, this gem was used to automatically keep track of the user’s timezone and provide him with accurate times for filling in forms. No additional setup was required other than simply including the gem into the application.

Rails_12Factor

Every website requires some static assets to be served. A static asset is a file which is pre-compiled and able to be shown after deployment (in a production environment). For that purpose, the “Rails_12Factor” gem installs two additional gems: “rails serve static assets” and “rails stdout logging”.

The first one forces Ruby-on-Rails not to discard static assets upon deployment, which in the case of MySportsFace are images that do not change, such as the website logo, the background image pattern, various customized icons and also sports organization’s logos. Any other images added to the website after deployment are not static assets and, therefore, do not suffer from the problem of being discarded by Ruby-on-Rails.

The second gem, while not as important from a production point of view, forces Ruby-on-Rails to send log messages to the standard output, instead of a specific default file, making it easier to debug the application during development.

3.7.15 Other JavaScript Libraries

In addition to all the Ruby gems used, the application also required an external JavaScript library to make certain parts of the website fully operational, called “NicEdit” [34]. This library offered a way to create a customized text-area with a WYSIWYG (“What You See Is What You Get”) set of options.
**NicEdit**

The default HTML text-area fields in forms do not allow rich text editing, which is basically writing HTML code without the user knowing that he is in fact adding HTML elements to his text. Therefore, the "NicEdit" library offers a cross-platform, lightweight and inline content editor that allows a user to make changes to their text, such as making words **bold** or *italic*, changing font colors, sizes or types and even adding images or hyperlinks to it, along with a lot more text editing options allowed by HTML.

To use this library, the JavaScript code was downloaded and included in the project and then whenever a text-area appears on the HTML view files, a method is called, once the page is loaded, to substitute every text-area tag for a customized one with all the aforementioned editing options. The customized text-area can be seen below, in Figure 3.14.

![Customized Text Area](Image)

Figure 3.14: A customized text-area after being replaced by the NicEdit library, as seen in the MySportsFace website.
Chapter 4

The Architecture of MySportsFace

This chapter presents the main functional requirements of MySportsFace, followed by a description of the architecture of MySportsFace, not only during a first iteration of its requirements but also after the requirements were reviewed and a new architecture was developed.

4.1 MySportsFace Requirements

In order to know what is necessary to develop the MySportsFace application, a list of requirements (see Appendix B) was compiled from the descriptions of the scenarios presented in Chapter 1. The most important requirements regarding features of the application are:

- Needs to support user accounts and have an authentication system;
- The authentication system should support multiple providers (Facebook, Google+ and the base authentication);
- Must support any sports organization which provides content through RSS feeds;
- Must be able to extract news articles and other information from RSS feeds of sports organizations;
- Registered users need to be able to create their own customized events;
- Must support uploading of files into galleries (images in this case);
- The application should possess a user voting/rating system for the various types of content displayed in it;

4.2 Initial Architecture and Problems

The first MySportsFace architecture originally had a different objective in mind regarding the amount of sports organizations to be supported. The plan was to only support a single organization and provide
special attention to every service the organization offered to its members, as those services would be
the most important to have in the application.

Therefore, this chapter will focus mainly on describing each aspect of the original architecture, such
as the users, the sports organization, the information users could browse with the application and,
finally, the payment module, which was required to implement some features. An early sketch of the
architecture is shown in Figure 4.1.

![Initial architecture of MySportsFace showing all modules of the application.](image)

**Figure 4.1:** Initial architecture of MySportsFace showing all modules of the application.

### 4.2.1 Users

To start with, users of the application were only meant to be the people who were already members
of the supported organization, or those who intended to become affiliated. The reason for this was
that multiple services offered by the application would require users to be registered. These services
included signing up for events or performing payments, such as annual fees.

**User Profiles**

Moreover, users would be able to log in to the application using a Single-Factor Authentication model
(with a username/email and a password) and then, each user would have access to their own profile,
which would contain their personal account data, an history of all the events they had participated in and
also a list of all the paid transactions they had performed.
4.2.2 News Articles and Information

Furthermore, another important part of the architecture required the application to provide its users with daily information from the media, regarding events, the community and the organization itself. This information would have been extracted directly from the organization’s database and stored as a set of news articles in the application.

However, at first, there was no planned way to import the information, which would mean that polling the database was required, wasting a lot of computing resources. This problem was fixed in the new MySportsFace application, as described later in Chapters 3.7 and 5.

4.2.3 Events and Registration

In addition to news articles, the application would also have to offer information on event results and standings and allow the users to sign up for events promoted by the organization. To perform these enrollments, users would be required to perform some sort of payment, in most cases, which meant that a payment mechanism had to exist (see Section 4.2.4).

Users were also going to be allowed to create their own custom events and send invitations to other users to participate in them. This feature is present in the newer MySportsFace architecture and is one of the main tasks that users can perform now.

On the other hand, registration for the events usually requires forms to be filled in, and thus, the website would have to provide links to them or have them embedded into the event pages. This would cause various layout problems and was one of the main reasons that caused the form registration process to be scrapped from the second version of MySportsFace.

4.2.4 Payments

Last but not least, multiple sections of the first MySportsFace application required users to perform payments such as annual fees. This type of registration requires a formal process which includes uploading a photo of the user, proof of payment of fees and sports insurance as well as the results of a sports medical examination report.

However, in many cases, the organizations (in Portugal) do not provide an online way to perform registrations, therefore reducing the number of organizations that could possibly be integrated in the application. Additionally, the ones that did provide a method of registration, required specific forms to be filled in.

On top of all the problems mentioned, the application would also need to be connected directly to a PayPal or SIBS account, so that the money could be transferred and stored in a safe place, to be later transferred to the organization.
4.2.5 Summary of the Issues

All things considered, the initial MySportsFace architecture had the following flaws and misconceptions:

- It was only able to support a single sports organization;
- The application had to have access to the organization’s database to keep track of user accounts and transactions;
- It lacked a defined method to import the news articles and information;
- In order to perform payments, the application needed to be linked to a PayPal or SIBS account and there would have to be a contract with the supported organization so as to send the part of the earnings;
- Organizations often do not provide a uniform way to perform registrations (for both membership and events).

Consequently, this architecture was abandoned to give place to a new one designed to support several sports organizations, so as to provide users of the application with varied sources of information on different sports.

Additionally, with this new design, MySportsFace only needs to access the RSS feeds (Really Simple Syndication) present in the websites of each sports organization, so that information may be retrieved. RSS feeds were the method chosen to extract information because they are a simple and efficient way of accessing and subscribing to streams of online data.

Moreover, since several organizations do not allow online registration for events, the new architecture provides users with a feature to create and sign up for events.

Lastly, the new architecture of the application is also designed towards being a social network, thus including various social features like user relations, commenting and voting/rating.

4.3 Current Architecture

Due to the faulty architecture of the first MySportsFace application, mentioned in Section 4.2, a series of important changes were introduced, to promote a better social environment for its users, while maintaining the required information sections of, in this case, multiple sports organizations, instead of just one. Therefore, by having various organizations for different types of sports, a wider group of users can be attracted to use the application.

Figure 4.2 shows the UML class diagram of MySportsFace, where it can be seen that most relations revolve around the users and the events or organizations.

4.3.1 Users and Relationships

In the new architecture, since users cannot perform payments using the application, there was no need to add a transaction history list to their profile information. However, the rest of the profile still presents
the same information as before: event participation list, personal data and their post list.

Furthermore, users can form connections with other users, by following or being followed and they can also interact with the organizations and events by adding them to their list of favorites and participating in them, respectively. With this said, users can also create their own small posts ("Microposts"), similar to "tweets", to share any information they like with their connections.

Another new addition to the users on this architecture is the concept of "identities". Each "identity" represents an alias for a user and, in this case, users can have up to three different "identities", used to enable authentication on the application through social networks (Facebook and Google+), instead of just using a normal MySportsFace account.

4.3.2 Organization Feeds and News Article Extraction

The new architecture also attempts to increase the amount of available information by implementing the "multiple-organization" theme and for that, feeds and feed entries were added to the database models.

Figure 4.2 shows that each organization has a single feed, which is used to contain the URL to the RSS feed for the real organization. The entries of the feed each represent information that is transformed into news articles for the users to view and write comments on.

Unlike the old architecture, the application does not need to be connected to the database of the various organizations. Instead, the RSS feed is the only link needed to extract information. As mentioned in Chapter 3, the Superfeedr gem takes care of all the subscribing and article extraction work, which means that the database does not need to spot duplicate article entries, for example.
4.3.3 Community Events and Other Content

The concept of creating and signing up for events was completely revamped in this new architecture. It became a community tool for creating and sharing content instead of a way to participate in official sponsored events.

New community events can be added to the application by any user and the database model saves the necessary basic event data. External database relations give information regarding the number and list of participants, event comments, images and videos.

Additionally, each event also possesses an editable schedule where its owner can add or remove entries, representing the various segments of the event, such as opening and closing ceremonies, prize award ceremony, the various rounds, etc.

Lastly, regarding the images and videos uploaded to events and, therefore added to the database, the only pieces of data required to be saved in each model are file names for images and links for videos.

4.3.4 Article and Event Comments

Similarly to news articles, events may also be commented. However, there was the need to create a new class/model for event comments, because if the one created for news articles was used, the comments would be forced to belong simultaneously to an event and an article, which can never happen. This constraint is caused by the way developers are able to specify dependencies and relations between classes/models, using the ":belongs_to" and ":has_many" keywords.

In the case presented above, where the same class/model is used for event and article comments, the file would contain the ":belongs_to" keyword twice, thus forcing the comments to belong to both an article and an event.

Otherwise, with the creation of a different comment class/model, if two comments with the same body (text) are be placed one in an event and the other in a news article, they are still two separate comments in the database, providing the desired effect.
Chapter 5

The MySportsFace Application

In addition to the revamped architecture of MySportsFace, presented at the end of Chapter 4, the interface of the application also required a significant number of other changes, to turn the old application from a "solo-organization" type of service to a more broad social network with support for the "multi-organization" theme. Hence, the user's profile information became more accessible and fully public for any other user to see and, in order to promote the social interaction within the application, the custom-made events were implemented, where users are allowed to insert their own content at will, respecting a few guidelines.

Moreover, as a way to support the owner(s) of certain pieces of content, and to increase their popularity, the users are able to give stars (imbued in a special voting system, named "Starring") to any post, comment, news article and event.

Consequently, in this chapter, the new MySportsFace application and its components are going to be described, along with an overview of each part of the application as seen from the point of view of a user, with various interface image examples.

5.1 Static Pages

Every website needs to have a homepage along with a few other so called pseudo-"static pages", which offer information that rarely changes, usually in order to guide users through the website or offering help regarding the usage of its features.

More specifically, in the MySportsFace application, there are three different "static pages", one of which is a homepage, that represents the root of the website and provides the users with a warm welcome to it, along with links for logging in. The other two, described in the subsections below, are the "About" page, which gives a brief overview of the website and attempts to answer the "what?", "why?" and "how?" questions, and the "Help" section that, as the word says, contains a series of answers for the most frequently asked questions, as an attempt to guide the user through possible common problems encountered while using the MySportsFace website.
5.1.1 About the Website

As mentioned above, the "About" page on the website has information regarding its origins, the content it holds and what a user can do once signed in.

![Figure 5.1: Information shown in the MySportsFace "About" page.](image)

As can be seen in Figure 5.1 (a "screenshot" of part of the "About" page), the information contained in it shows a general description of MySportsFace and explains in which platform it was developed, along with a list of tasks that a registered user can perform while using the application. Furthermore, the page also contains a very brief guide to help a user get started with the using MySportsFace. Last but not least, in case a user requires additional information or is interested in any content they see through the website, the "About" page also provides a set of email addresses for contacting the author.

5.1.2 Frequently Asked Questions

The remaining "static page" is the "Help" page which contains a set of most Frequently Asked Questions (FAQ) which happened during the usability and satisfaction tests with users, along with a few other questions which were still significant enough to be added to the list, as they help develop a better usage of the website and provide additional guiding.

As such, the questions are divided into four big groups. There are Account-related questions, Organization-related questions, Event-related questions and then also a Miscellaneous questions group, for questions which are not related to a certain part of the website, in specific.

The Account-related questions provide information relative to the user's profiles, account creation and the hybrid "log in" system. The Organization-related questions explain why the users may encounter possible difficulties while trying to find news articles information and the reason why not every sports organization can be found within the MySportsFace application. Moreover, the Event-related questions provide guidance towards information shown in the events and some of the restrictions imposed on
sharing content. Additionally, the Miscellaneous questions offer help regarding various sections and features of the website.

Lastly, if certain important questions were missed, the users can also contact the author to get an answer to a question they have and, if the question is relevant enough, it may also be added to the FAQ list.

5.2 Users and Profiles

A social networking application is nothing without its users. Therefore, it is important to keep track of the MySportsFace users data and make sure their profiles show significant information.

Hence, every user possesses a list of all posts they have made since the creation of their account, as well as a list of events that the user is participating in. Besides, the user can also check which users they are following and which are following them through a set of two other links on their profile. These “Followers” and “Followees” help form a healthy network of users with the similarity of the “Friends” in Facebook or the “Followers/Followees” in Twitter.

Figure 5.2: Example of a user profile and the content displayed in it.

Figure 5.2 presents a "screenshot" of a user profile, showing the various hyperlinks to all kinds of personal information, along with the post list which includes a fast way to post additional content, located at the top of the profile page boundaries. On the left side of the figure, a small profile can also be seen, which is present in various other pages in the website, for a more direct access to profile information and other important users.
5.2.1 Sessions and Logging In

The creation of an account at MySportsFace can be done in three different ways: through using Facebook credentials, Google credentials or simply by using the website’s account creation form.

On top of this, whenever a user logs in for the first time using a specific method, the internal “log in” system will create a new identity for the user, merging it with other identities which share the same email address. This is a way to prevent multiple accounts from being created for the same person, as long as the email used to log in is the same in all three methods.

![MySportsFace homepage](image)

Figure 5.3: MySportsFace homepage, showing the various ways to log into the application.

Finally, the sessions used to log into the website are all encrypted and stored into a browser cookie, which is safe against external attacks to prevent other users from hacking into another person’s MySportsFace account, unless they explicitly give out the respective session cookie.

5.3 Requirements for Supporting Organizations

As the MySportsFace application is a sports-driven social network, it is required to support a multitude of sports organizations, to cater for everyone’s needs. However, only specific organizations were inserted as there were restrictions imposed to the form content is consumed to be shown in the application.

5.3.1 Feeds and Content Constraints

With this said, only organizations which contemplated websites with RSS feeds of some sort were added, as this was the main way to access their news articles information and extract information regarding past, current and future events, social media and other miscellaneous news.

In addition to this aforementioned restriction, the RSS feeds also required to have specific tags, representing the article’s title, summary and hopefully the entire content of the page. These tags, written
in XML, were used by the Superfeedr gem to add content to the application through the creation of “Feed Entries”. These entries were then transformed into “Articles” for the respective “Organization” and, when possible, the full HTML content of the page was shown (if the "content" tag was present through RSS), otherwise the “Article” contained a summary, also given by the RSS feed entry, and a link to the original post which by default contains the information. This is also a way to avoid copyright infractions, by explicitly stating the owner of the content and sharing a link to it.

5.3.2 Adding Other Organizations (and their RSS feeds)

Without further due, it is important to state that administrators of the website are allowed to add extra organizations and therefore also their RSS feeds. However, this is a delicate process, where a series of constraints need to be respected. These constraints are:

- The new organization needs to have a logo image to be shown in the website (with a resolution of 650x300 pixels, to fit properly), which implies that the administrator is required to have a suitable image before starting the process;

- It is also necessary to have a small text that briefly describes the organization.

- Lastly and most importantly, the administrator needs to have a functional link to the new organization’s RSS feed, in order to automatically extract news articles using the “Superfeedr” gem.

Thus, the process is prone to human error, which could cause discrepancies in the information shown between different organizations and as such, the feature of adding new organizations to MySportsFace should be done with extreme care and, hopefully with the necessary preparations. Figure 5.4 shows the require form fields to imbue a new sports organization into the application.
This process cannot be efficiently automated as the links to the organization logo images would have to be known beforehand. The same reasoning applies to the small description texts, which are handpicked and edited.

In any other case, the developer(s) in charge of the application can also manually add new organizations to the database by hard coding them into the "seeds.rb" file, which will populate the database with the respective organizations, when used in a "rake db:seed" command.

### 5.4 Posts and Comments

Another social component of the website is the ability that users have to add comments to the things they browse, such as events and news articles and to post mood updates, which can be seen through a Recent Posts feed by the user itself and any person following that user. This way, a user can keep track of their surroundings by knowing what each person they are following is posting. To help distinguish posts authored by the user and posts by someone else, the background colors of the posts are different (GREEN for the owner's posts and BLUE for the remaining). Below, in Figure 5.5 is represented the Recent Posts feed page, which substitutes the home page after a user logs into the application. Finally, posts may be deleted by their owner.

![Recent Posts feed page on MySportsFace, with the different post colors signaling the owner's posts.](image)

In regards to the comments section presented in both the event and article pages, users have access to a customized text-area accomplished with the NicEdit script, where they can edit their comments before posting them for the community to see.
5.5 Community Events

MySportsFace application also provides users a way to create their own sports events and fill them with content at their leisure. A user-made community event is required to have the following traits:

- a title shown both in the “tooltip” of the event and the page itself;
- a brief summary of the event, only to be shown in the “tooltip”;
- a detailed description of the event with any kinds of useful information, shown in the event page;
- and a date, time and location for the event.

![Event Information](image)

During the creation of an event, as the user is defining its location, there is a Google Maps script which is searching for the location within Google’s database and marking it on the map.

Figure 5.6 shows a section of the event page which contains the event information, along with the mentioned Google map which points to the location defined by the user. Still in Figure 5.6, can be seen the schedule of the event, which can be altered at any moment by the user which created the event.

Figure 5.6: Section of an event page showing its information.
Alongside these, the event page will show other kinds of dynamic data, such as the event schedule, the current participants and any image uploaded or video linked by them, as well as every comment the users post. As a way to make navigation through event pages smoother and faster, a set of “quick links” are shown at the top, which allows users to browse directly into one of the sections of the event, be it images, videos, participants, comments or information in general.

5.5.1 Image Uploading and Video Links

The events would not be the same without allowing users to add photos or videos to them. Therefore, using a gallery (shown in Figure 5.7), the users of MySportsFace can upload image files directly from their devices to the event web page, which will be made public to anyone visualizing the event. In order to try and mitigate the inclusion of “spam”, only users which have signed up for the event may upload files. The uploading process also imposes restrictions regarding file extensions used, only allowing ".jpg", ".jpeg", ".gif" and ".png" files to be uploaded, up to a limit of 4 Megabytes (MB) as well.

![Image Gallery Example](image1.jpg)

Figure 5.7: Example of an event image gallery.

Furthermore, using a "carousel" style slideshow (as seen in Figure 5.8), enabled by the usage of Twitter’s Bootstrap, all the videos linked to the event page will be able to be browsed. Similarly to the constraints applied to the images uploaded, there are only two certificated websites, YouTube and Vimeo, from which users who signed up for the event can link their videos. On the other hand, unlike images, which can directly be uploaded to the website, videos have to be previously uploaded to one of
the two mentioned websites. This methodology ensures that any other user not viewing MySportsFace may also look at the videos through other third-party websites.

![Figure 5.8: Example of an event video slideshow.](image)

Additionally, videos uploaded to an event page use the embedded code available from either YouTube and Vimeo, which means that the users viewing them may perform any tasks of voting on the video and even sharing it through other social networks, if and when available. As regards the moderation of video content at MySportsFace, the two chosen websites already process the videos for violations of their own Terms and Conditions, which prevents videos with undesired and foul content from being shown at MySportsFace.

### 5.5.2 Event Administration

Lastly, in order to manage and maintain an event, its creator and owner may assign other trustworthy users to be administrators of the event, and thus gain the power to decide whether certain content is suitable or not, by being able to remove undesired images or videos.

Figure 5.9 shows an example of an event participants list with the options given to an event creator and admin that allow them to remove and add administration rights to users, as necessary. Moreover, Figure 5.8, also shows a “Remove” button below the current video being shown in the slideshow, which is only available to event administrators, and lastly, Figure 5.10, shows a removal option next to the image thumbnails in the event image gallery.
5.6 Voting System

The content of MySportsFace needed some more interactivity, which is why a special voting system was added, which allows users to show their appreciation by "starring" posts, comments, events and news articles. This voting system works with the help of the “Thumbs Up” gem, and while it is possible to implement a positive/negative voting mechanism, the choice was made to only allow users to vote positively and only once. This means that a user may add or remove their vote from a specific piece of content as they like, but they are unable to add more than one vote or vote negatively, which would add negative scores to the respective contents.
By using this methodology, other users will not get frustrated or offended by the fact that the content they created or posted is highly unpopular or unlikable, because that will not be shown through the "Starring" system.

Figure 5.11 shows the three conditions in which posts can be, in regards to the starring system. The same can be applied to any event, article or comment.

5.7 Security Permissions and Content Availability

Last but not least, the security permissions of MySportsFace are of great relevance, as not every page and content should be available to a non-registered user.

Therefore, every user has access to the three main "static pages", since they show general website information, but the remaining sections of the website require the person browsing to be registered and logged in. One of the reasons for taking this approach is to maintain the stability of the website information, because various features on the pages extract profile information or require a logged in user to collect the information for forms.

Nevertheless, after a user is logged in, all the content is technically public and may be visited/seen by everyone in the community, but with the restriction to uploading and linking images and videos to events, mentioned in Section 5.5.1 of this chapter.

5.8 Deployment and Amazon Web Services Integration

Despite the fact that the project was first tested on a local machine, it still required to be deployed somewhere in the Internet for public usage. Therefore, the MySportsFace application was deployed to the Heroku cloud-based deployment service, which has a partnership with GitHub, making it easy to directly import the project from one website to another.

However, the Heroku platform has some file storage constraints, which had minor impact in the project, but still required an additional third-party storage system to solve, the Amazon Simple Storage Service (S3).

As such, this chapter describes the Heroku deployment service briefly, as well as its drawback and the solution found for the problem in question.

5.8.1 Heroku

Heroku is a deployment service based around developer productivity, which means that it strives to make the deployment process as simple and as fast as possible, to induce zero effort from the programmers.

On the one hand, at Heroku, they offer a free single "dyno" (which is a process where the virtual machine is located) to run any web application. In the case of the MySportsFace application, the "dyno" is running an instance of a "Puma" web server, which was explained in Section 3.7. Additional "dynos"
for web servers, databases or web workers can be added through the Heroku application dashboard to increase performance, if necessary (inducing monthly costs beyond the first “dyno”).

Figure 5.12: Heroku application dashboard, showing the various database, web server and web worker resources allocated.

Figure 5.12 shows the application dashboard at Heroku, which states the number of web servers and other workers being used, as well as the type of database used and, finally, the costs induced. Furthermore, the Heroku platform also keeps track of all the activity (i.e. application builds and changes), which can be linked to the repository at GitHub, too.

On the other hand, Heroku unfortunately possesses a volatile storage system, which cleans up any non pre-compiled asset uploaded to the web application, after a certain threshold of hours has passed. What this means is that the files uploaded will become inaccessible and therefore any link(s) to them will be void, therefore breaking the content existing in the application. This was mainly an issue regarding the MySportsFace gallery images, which were the only uploaded resource.

Nevertheless, the image content is still present in the database even after the volatile files are cleaned up, which means that this problem can be solved by using an external web storage service, such as Amazon S3 (see below), which was temporarily integrated into the application as a way to go around the Heroku constraint. Amazon S3 ended up being removed after confirmation that the integration with MySportsFace was successful, as it involved monthly payments to keep using their servers.

### 5.8.2 Amazon Simple Storage Service (S3)

In order to integrate Amazon S3 in MySportsFace, a series of steps were required [35]. First, an Amazon account was created and the secure keys were saved for usage in the application. These keys gave the application access to the Amazon Web Services API and were immediately imported into Heroku as environment variables (or using the “DotEnv” gem, in the case of local testing).

Afterwards, a script was included in the project (the jQuery-File-Upload), which enabled file uploading
to the Amazon file system servers. Once an image is uploaded to Amazon S3, it becomes slightly encrypted, with the attribution of a new name, but it is then available forever, until someone decides to delete it, unlike what happened with the volatile file system at Heroku.

Moreover, the script used to upload files also provided a progress bar, so that the user uploading it could know when the file was fully saved in the Amazon S3 service, ready to be shown in the application’s gallery.

On another note, the Amazon Web Services have a “Free Tier” of usage, which includes some very little limits in terms of file storage and uploads. A user under the “AWS Free Tier” type of account is allowed to make only up to two thousand (2,000) PUT requests and up to twenty thousand (20,000) GET requests, along with a 5 GB storage limit.

Lastly, the Amazon S3 servers work on top of EC2 (Elastic Cloud Computing) server instances, also part of the Amazon Web Services, and these also have usage limits, in terms of computing hours and the type of servers used, to avoid incurring monthly payments.

Therefore, to conclude the topic in AWS integration, the S3 service was added to the MySportsFace application for a little while, to make sure it was possible to integrate and use it alongside a Heroku deployed application, but was later removed, keeping only the volatile file system working, which did not influence the results of the usability and satisfaction tests ran afterwards, as the users only require the material to be ready at the time of testing, at least for an academical application such as MySportsFace.
Chapter 6

Testing the Application

After each functionality was added to the application, a series of unit tests had to be ran, to guarantee that the architecture requirements were met and nothing escaped validation. In addition to unit tests, later during the development of MySportsFace, after a first complete version of the application, usability tests were conducted with real users to ensure that all features were working properly from a client point of view and to extract possible increments and suggestions during the testing, to make the application better.

6.1 Unit Tests

The unit tests were conducted using a computer with an Intel CORE i7 2.3GHz quad-core processor and a Windows 8.1 operative system with 8GB RAM available. These tests were mainly used to make sure that no part of the MySportsFace application was running with errors or design flaws. As such, the unit tests were divided into three categories: model, controller and integration tests.

Also, in order for the tests to be conducted, a special database had to be added, by using fixtures, which are elements specified with tag names, that are parsed by the testing framework to create real objects in the testing database.

To run the unit tests, a command line program called "rake" had to be ran with the argument "test". The programmer can specify any subset of the tests to be ran at a given time by adding additional arguments to the command. For instance, to run just the integration tests, the programmer should write "rake test:integration". After the tests are ran, the program will show how many assertions were successful and how many were not.

6.1.1 Database Fixtures

The database fixtures were split in multiple files, one for each model/class defined in the application, and they may include any number of tagged objects, with the respective variables defined so as to directly input them into a real object once parsed.
6.1.2 Model Tests

The model tests were necessary to guarantee that each application model/class respected its constraints and the necessary architecture requirements. For example, a "User" class object may only have passwords with 8 or more characters and the email has to respect a given template, to make sure that no random strings are inserted into that field.

The same reasoning was applied to confirm other validations, such as variable presences, lowercase or length requirements and even database ordering (i.e. to check if elements are being stored in an ascending or descending order of creation dates, when required by the application).

Therefore, each model present in the application has a test file of its own, which uses the respective database fixture and additional resources created in a "setup" method before running the tests. An example of this can be seen in Figure 6.1.

```ruby
require 'test_helper'

class OrganizationTest < ActiveSupport::TestCase
  def setup
    @organization = organizations(:exampleorg)
  end

test "should be valid" do
  assert @organization.valid?
end
end
```

Figure 6.1: Excerpt of a model test code with a setup method creating an organization and a single test confirming its validity.

6.1.3 Controller Tests

On the other hand, the controller tests are used to validated the routing paths of the application. For example, to make sure that a certain page is successfully loaded when a HTTP GET request is performed or to prohibit certain pages from being shown to non-registered users.

Similarly to the model tests, these ones also make use of the database fixtures and other helper methods to create various different application states, to be tested.

Figure 6.2 shows a set of tests for the controller of the MySportsFace user posts, which validate the redirection mechanism when a user does not have enough rights to perform certain actions in the website, such as the creation or deletion of posts.

6.1.4 Integration Tests

Finally, the integration tests are used to test each page interface as a whole, mixing in various actions that a user might perform, to ensure that no matter what specific options a user takes while doing something in the website, the outcome should always be the same. As an example, it is easy to validate
require 'test_helper'

class MicropostsControllerTest < Action.Controller::Test
  def setup
    @user = users(:marcos)
    @micropost = microposts(:firsty)
  end

  test "should redirect create when not logged in" do
    assert_no_difference 'Micropost.count' do
      post :create, micropost: { content: "Test" }
    end
    assert_redirected_to login_url
  end

  test "should redirect delete when not logged in" do
    assert_no_difference 'Micropost.count' do
      delete :destroy, id: @micropost
    end
    assert_redirected_to login_url
  end

  test "should redirect delete when wrong user" do
    log_in_as(@user)
    micropost = microposts(:other_post_0)
    assert_no_difference 'Micropost.count' do
      delete :destroy, id: micropost
    end
    assert_redirected_to root_url
  end
end

Figure 6.2: Excerpt of a controller test code showing three different states of the application where different outcomes should happen.

a single object within the database, or to validate that a certain link brings a user to the correct page, but the process of posting a comment or "starring" an article is complex and, therefore, requires a different set of tests, which are used to confirm that specific data in the database either remains unchanged after a set of operations or that it is changed to the correct and expected values.

This type of tests usually includes a mix of both the model and controller tests, by using page redirects and model object creations, as well as various asserts to variables and HTML page element counts. In Figure 6.3, is shown an example of an integration test code for asserting the validity of the login system at MySportsFace.
test "login with valid info" do
  get login_path
  post login_path, session: { email: @user.email, password: 'password' }
  assert_redirected_to root_url
  follow_redirect!
  assert_template 'static_pages/home'
  assert_select "a[href=?]", login_path, count: 0
  assert_select "a[href=?]", logout_path
  assert_select "a[href=?]", user_path(@user)
end

test "login with invalid info" do
  get login_path
  assert_template 'sessions/new'
  post login_path, session: { email: '', password: '' }
  assert_template 'sessions/new'
  assert_not flash.empty?
  get root_path
  assert flash.empty?
end

test "valid login and logout" do
  get login_path
  post login_path, session: { email: @user.email, password: 'password' }
  assert_redirected_to root_url
  follow_redirect!
  assert_template 'static_pages/home'
  assert_select "a[href=?]", login_path, count: 0
  assert_select "a[href=?]", logout_path
  assert_select "a[href=?]", user_path(@user)
  delete logout_path
  assert_not is_logged_in?
  assert_redirected_to root_url
  # Simulating a second window logout after already logging off
  delete logout_path
  follow_redirect!
  assert_select "a[href=?]", login_path, count: 2
  assert_select "a[href=?]", logout_path, count: 0
  assert_select "a[href=?]", user_path(@user), count: 0
end

test "login with remembering" do
  log_in_as(@user, remember_me: '1')
  assert_not nil cookies['remember_token']
end

test "login without remembering" do
  log_in_as(@user, remember_me: '0')
  assert nil cookies['remember_token']
end

Figure 6.3: Excerpt of a integration test code which validates the logging in and out of the application, through multiple scenarios.
6.2 Usability and Satisfaction Tests

An application is not fully tested until its users are allowed to test it for themselves. Thus, using the application requirements sheet (shown in Appendix B), a sequence of tasks was picked to be part of the tests specification. Afterwards, the tests were initially conducted with a group of 10 users, which were told to complete each task the way they preferred. The results from this first set of usability tests (along with the suggestions and comments from the users) were taken into consideration before conducting the tests a second time, with a larger group of users.

Therefore, in this section, the user profiles are described, along with the various application requirements that led to the creation of the tests specification sheet and the criteria to evaluate the tasks performed by the users during the tests. Additionally, the results of each testing phase are then shown and commented in the end of this section.

6.2.1 Test-user Profile

As the application contains mostly sport related content, the targeted group of users for the tests consisted mainly in those which practice or enjoy sports in general and those that belong to any sports organization.

Furthermore, considering that the application is entirely written in English, but was tested in Portugal, the target users had to be of age 16 or older, in order to have a complete understanding of the information shown by the application, as well as to try and target users which also had interest in sport communities and organizations.

6.2.2 Use-Case Scenarios and Tests Specification

In order to create the tasks necessary for the users to perform the usability tests, two different use-case scenarios were written, showing the various important tasks a fictional user could perform at a given time.

After the selection of the tasks for the usability tests, each one had to be measured. In this case, the types of measurement were the time taken to complete the task and the amount of errors made during its completion. So as to arbitrate the lower and upper limits for these two testing measurements, an initial testing run was performed by an expert user, which defined the optimal times for the completion of every task and afterwards, those times were either doubled or tripled to accommodate for slower and more inexperienced users. For the number of errors, the optimal amount for every task was zero and then, depending on its difficulty, the upper limit ranged from one to three errors.

The scenarios previously mentioned were shown in Section 1.3.1 of Chapter 1 and the various words in bold represent an early concept of the various tasks used in the tests specification sheet (see Appendix C).
6.2.3 Test Results

The users were prompted to take 10 to 15 minutes of their time in order to complete the full set of tasks, while in a home environment, without any external disturbances. As each task was performed, a timer started counting and it stopped once the user completed the task, monitored by the test conductor, who also counted the number of errors made.

After the tasks were all completed, the users were asked to answer a small set of questions to help understand their feelings for MySportsFace and to collect information on what they thought were the positive and negative aspects of the application.

During the first testing phase, the MySportsFace application still lacked some features, such as the About and Help pages of the website, as well as the searching functionality for each type of content in the application (articles, users, events and organizations). This influenced the task completion times, as will be seen later, when comparing the results from both testing phases.

For a better comprehension of the test results, the list below enumerates the eleven tasks included in the usability tests:

1. Create a new account at MySportsFace;
2. Create a new post saying “Hello world!”;
3. Follow a user called “Francisco Manuel”;
4. View the posts of “Francisco Manuel” and count how many he has made;
5. View the organization named “Federação de Desportos de Inverno de Portugal" and add it to your favorites;
6. Visit the news page for “Federação de Desportos de Inverno de Portugal”, view the article named “1º Curso de Roller Ski” and leave a comment on it saying “Exceptional”;
7. Create a new event with the following traits — name: “My First Personal Event”, summary: “This is my first event!”, description: “Sign up for my event below”, date: “30-October-15”, time: “10:00”, location: “Alameda, Lisboa, Portugal”;
8. Fill in a schedule for your newly created event with the following entries — “10:00 - Event Check-in”, “11:00 - Event Opening Ceremony”, “12:00 - First Round”, “14:00 - Second Round”, “16:00 - Event Results”, “17:00 - Event Ending”;
9. Sign up for an event called “Surf Summer Split Championship” and upload an image with the name “surf-2.jpg” from the desktop to the gallery;
10. Sign up for another event called “Skyline Ski Tourney” and link any video from YouTube or Vimeo to the event page;
11. Show your appreciation by giving the event a “star”.

67
1st Testing Phase

As mentioned earlier in this section, the first set of tests was conducted with 10 users and the average times and errors for the completion of each task can be seen below, in Figure 6.4. Additionally, the full set of results for each user and task can be seen in Appendix A.1.

![Average Time and Errors for all Tasks](image)

**Figure 6.4**: Average time and errors for each task in the first testing phase.

By analyzing Figure 6.4, one can notice that some tasks required marginally more time to complete, due to their increased difficulty and length determined by the additional number of steps required. This is the case for some of the latter tasks (7 to 10), which involved creating and signing up for events and sharing multimedia content.

In addition to this, the users had more trouble when trying to sign up for an event, saying that the sign up button was not intuitively placed, and therefore, hard to find. Some of the younger users also consistently completed the tasks faster than the older ones, most likely because the younger users have more experience working with gadgets and the Internet in general. Also, the errors made during the completion of the tasks were usually caused by clicking on the wrong links or not respecting certain form constraints, such as the minimum number of characters required to fill in a new user-name or password.

Last but not least, still regarding the first testing phase, there were no outliers, as the results from each user tended to be around the average. There were however, two users who consistently achieved either slower or faster times than the remaining users on most tasks. The user who got the greatest number of slowest times was less experienced with social networks and was quite older than the one who got the greatest number of fastest times was constantly online on social networks, such as Twitter and Facebook and is what is socially considered a digital native (a person who has been attuned to information technologies since an early age).
2nd Testing Phase

For the second set of tests, a series of improvements were made to the MySportsFace application, following the suggestions of some users from the first tests. These improvements included the addition of a search mechanism for website content and re-positioning of certain website layout elements, such as the sign up button for events, to increase the application usability. To check for performance improvements and in order to have a bigger pool of results to analyze, the group of users for this set was increased to 24, where the original 10 users were included. The full list of results for this testing phase can be found in Appendix A.2.

Figure 6.5: Average time and errors for each task in the second testing phase.

Figure 6.5 shows the average times and errors for the same eleven tasks during the second testing phase. Analyzing the results and as seen in Figure 6.6, the number of users making mistakes was lower when compared to the first testing phase, likely due to the new mechanisms added and the improvements made to the application.

Likewise, the time taken to complete the tasks was also reduced in average, as seen in Figure 6.7. Tasks 9 and 10, which required users to sign up for events before uploading multimedia content (videos and images), showed an improvement due to the positional change of the event sign up button.

Moreover, tasks 3, 5 and 6, which required users to look up specific content, showed that the newly implemented search mechanism helped reduce completion times by a small margin.

Besides, the difference in times for the completion of task 8 is significantly lower in comparison to the first testing phase, mostly due to the fact that the former ten users had already been in contact with the interface and knew now what to do, but also because the event layout was revamped to provide a better element disposition and sizes. Finally, also seen in Figure 6.7, the remaining tasks did not show...
a significant difference in the time taken to complete them, as some of them were already simple tasks with few steps required to accomplish, such as creating accounts or “starring” events.

Figure 6.6: Comparison of the average task errors between both testing phases.

Figure 6.7: Comparison of the average task completion times between both testing phases.
Additionally, the test results can be further analyzed in terms of average task completion times by splitting up the users into two groups: **the ones who had had previous experience with the interface** (from the first testing phase) and **those who were using it for the first time** (during the second testing phase). This way, three graphs were created, comparing two groups of users during the various testing phases. Each graph also includes the variance of the average task completion times, in order to spot possible outliers and to give a broad view of the statistical difference between results.

![Comparison of Average Times between Testing Phases for the first 10 users](image)

Figure 6.8: Comparison of the average task completion times between both testing phases for the 10 initial users only.

More specifically, Figure 6.8 compares the average task completion times of the two testing phases, for the ten initial users which were present in both phases only. By analyzing this graph, it can be concluded that Tasks 5 and 10 seem to show an improvement, with also Tasks 3 and 9 close behind. These were the tasks directly influenced by the changes made to the application for the second testing phase. As such, and to further analyze the results, a two-tailed paired Student's t-test (T-TEST) was performed to determine if the task completion times were statistically better during the second phase in comparison to the first, for each task. This was done using Google Spreadsheets and its functions.

As seen in Tables 6.1, 6.2, 6.3 and 6.4, the p-values returned by the T-TEST calculation are very close to 0. Given the **null hypothesis that the mean for each sample set is statistically the same**, then the probability of that happening according to the calculations is equal to $p$-value. In other words, the probability of the results showing an improvement is equal to $1 - p$-value, which means that it is very likely that the results for those tasks improved due to the interface changes, at least for the initial users. The results of the remaining tasks were not considered because there were no interface changes made that could affect their completion times. Therefore, the Student's t-test calculations are omitted.
Table 6.1: Student's t-test data and results for the initial users when completing Task 3.

Table 6.2: Student's t-test data and results for the initial users when completing Task 5.

Table 6.3: Student's t-test data and results for the initial users when completing Task 9.

Table 6.4: Student's t-test data and results for the initial users when completing Task 10.

Another comparison was between the ten initial users and the new ones during the second phase. Figure 6.9, shows that the ten initial users, who had had previous experience, attained better results than the other new users, which is to be expected.

On the other hand, the graph also shows that statistically the results do not differ too much, due to the overlapping variances between initial and new user task completion times. Performing Student's t-test on the results of this graph would not provide any significant conclusions, as it would ultimately just prove the fact that having previous experience influences task completion times.

Therefore, a better way of proving that the interface changes were positive and influential is by comparing the ten initial users during the first testing phase with the new users from the second phase. Figure 6.10 shows the results of that comparison.

In order to further analyze these results, a two-tailed unequal variance Student's t-test was performed, with samples from the same previous tasks: 3, 5, 9 and 10. The results of these tests are presented in Tables 6.5, 6.6, 6.7 and 6.8.

As seen on the p-values of each table, the probability of the null hypothesis is low, which means that the mean of each sample set is statistically different. Therefore, the improvements made to the application were directly influential to the decrease in task completion times for the tasks 3, 5, 9 and 10 and thus, the usability of MySportsFace was also improved.
Figure 6.9: Comparison of the average task completion times between the 10 initial users and the new ones during the second testing phase.

Figure 6.10: Comparison of the average task completion times between the 10 initial users during the first testing phase with the new ones during the second phase.
### TABLE 6.5: Student’s t-test data and results comparing the initial and new users when completing Task 3.

<table>
<thead>
<tr>
<th>Task</th>
<th>Old Users</th>
<th>New Users</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>28.5</td>
<td>17.5</td>
<td>24.0</td>
</tr>
<tr>
<td></td>
<td>27.6</td>
<td>23.2</td>
<td>22.8</td>
</tr>
<tr>
<td></td>
<td>20.4</td>
<td>21.1</td>
<td>26.9</td>
</tr>
<tr>
<td></td>
<td>26.3</td>
<td>23.2</td>
<td>22.8</td>
</tr>
<tr>
<td></td>
<td>20.4</td>
<td>21.1</td>
<td>26.9</td>
</tr>
<tr>
<td></td>
<td>23.2</td>
<td>22.8</td>
<td>20.4</td>
</tr>
<tr>
<td></td>
<td>17.2</td>
<td>20.3</td>
<td>26.8</td>
</tr>
<tr>
<td></td>
<td>18.9</td>
<td>19.2</td>
<td>19.5</td>
</tr>
<tr>
<td></td>
<td>18.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

p-value: 0.05000329159

### TABLE 6.6: Student’s t-test data and results comparing the initial and new users when completing Task 5.

<table>
<thead>
<tr>
<th>Task</th>
<th>Old Users</th>
<th>New Users</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25.1</td>
<td>13.7</td>
<td>18.2</td>
</tr>
<tr>
<td></td>
<td>24.5</td>
<td>20.9</td>
<td>21.3</td>
</tr>
<tr>
<td></td>
<td>20.4</td>
<td>17.4</td>
<td>26.1</td>
</tr>
<tr>
<td></td>
<td>22.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.0</td>
<td>13.3</td>
<td>15.2</td>
</tr>
<tr>
<td></td>
<td>11.1</td>
<td>15.5</td>
<td>12.4</td>
</tr>
<tr>
<td></td>
<td>13.6</td>
<td>14.1</td>
<td>13.4</td>
</tr>
<tr>
<td></td>
<td>16.2</td>
<td>13.5</td>
<td>12.8</td>
</tr>
<tr>
<td></td>
<td>14.0</td>
<td>11.6</td>
<td></td>
</tr>
</tbody>
</table>

p-value: 0.0001036134945

### TABLE 6.7: Student’s t-test data and results comparing the initial and new users when completing Task 9.

<table>
<thead>
<tr>
<th>Task</th>
<th>Old Users</th>
<th>New Users</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>45.1</td>
<td>27.9</td>
<td>35.8</td>
</tr>
<tr>
<td></td>
<td>41.9</td>
<td>36.9</td>
<td>38.3</td>
</tr>
<tr>
<td></td>
<td>40.1</td>
<td>39.0</td>
<td>42.6</td>
</tr>
<tr>
<td></td>
<td>41.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>43.2</td>
<td>38.1</td>
<td>36.6</td>
</tr>
<tr>
<td></td>
<td>39.2</td>
<td>39.4</td>
<td>32.5</td>
</tr>
<tr>
<td></td>
<td>36.8</td>
<td>37.1</td>
<td>29.6</td>
</tr>
<tr>
<td></td>
<td>35.6</td>
<td>36.3</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>31.5</td>
<td>32.4</td>
<td></td>
</tr>
</tbody>
</table>

p-value: 0.08673987172

### TABLE 6.8: Student’s t-test data and results comparing the initial and new users when completing Task 10.

<table>
<thead>
<tr>
<th>Task</th>
<th>Old Users</th>
<th>New Users</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>72.4</td>
<td>78.2</td>
<td>63.7</td>
</tr>
<tr>
<td></td>
<td>51.6</td>
<td>64.4</td>
<td>73.4</td>
</tr>
<tr>
<td></td>
<td>66.8</td>
<td>60.5</td>
<td>70.1</td>
</tr>
<tr>
<td></td>
<td>70.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>47.8</td>
<td>51.6</td>
<td>54.6</td>
</tr>
<tr>
<td></td>
<td>49.1</td>
<td>52.0</td>
<td>60.2</td>
</tr>
<tr>
<td></td>
<td>61.7</td>
<td>57.8</td>
<td>55.3</td>
</tr>
<tr>
<td></td>
<td>68.1</td>
<td>62.4</td>
<td>56.6</td>
</tr>
<tr>
<td></td>
<td>58.0</td>
<td>53.9</td>
<td></td>
</tr>
</tbody>
</table>

p-value: 0.001509153142
Chapter 7

Conclusions

The most important achievements of this work and some remarks regarding future work features are presented in this final chapter.

7.1 Achievements

In the developed work, a web application for supporting sports organizations was successfully created, with the help of the Ruby-on-Rails project framework and various third-party API's and programming libraries.

The final and current version of the MySportsFace application allows its users to register and store their accounts at the website so as to perform a series of other tasks. These tasks can range from socializing with other users through writing comments and posts/"mood updates" to browsing up-to-date content from any of the available sports organizations on the website.

Additionally, with the creation of customized events, users are able to create their own content for the MySportsFace community to view, as well as upload images and videos to the custom event pages.

Furthermore, with the help of the Twitter Bootstrap API, the interface of the whole MySportsFace application was able to be revamped in order to provide users with a more aesthetic and minimalistic environment, with less but important information in every website page.

Moreover, after the completion of the a first iteration of the application, the usability and satisfaction tests conducted were able to find a few flaws in the design, such as the wrong positioning of the sign up button for customized events and the lack of a searching mechanism, but overall, they showed promising results in terms of the usability of the remaining features. On top of this, users also showed enthusiasm while using the application for the first time.

Last but not least, once some changes were made according to the user suggestions and complaints, by creating a standard search mechanism and re-positioning the event sign up button to the top of the event pages, the second run of usability tests started and, this time around, the results indicated that, on average, users spend now less time completing tasks related to browsing website content and also sharing multimedia content of their own. Another set of features of the application which influenced the
user satisfaction results during the second testing phase were the "About" and "Help" pages that offered various simple tips and general information on the usage of the website, making sure that users which got lost during the completion of certain tasks could get back on track by finding helpful information in either of the pages.

7.2 Future Work

There is still an vast number of extras features that could be implemented in the MySportsFace application, some of which would have required more effort and time to include.

First, one of the main issues encountered when developing the current version of the MySportsFace application was finding organizations which had RSS feeds from which the news articles information could be extracted easily, using the Superfeedr Engine to subscribe and retrieve feed entries. This constraint reduced the number of available worldwide sports organizations that could be included within the application.

However, despite the fact that various organizations do not provide RSS feeds, a web crawling or big-data parsing mechanism [36] could be implemented, which would extract any kind of news articles, blogs and other relevant information regarding any organization worldwide, by parameterizing the search results sought.

Among the remaining features, one which was already thought of during the first architecture of MySportsFace was the creation of a payment mechanism and possibly an online shop, which would allow users to subscribe and renew their annual subscriptions at various sports organizations, as well as be able to buy diverse items at the shops. However, this would require the application to have a contract with each of the organizations that allowed for online registration and those that possessed online shopping mechanisms. Additionally, MySportsFace would have to include other external API to deal with payments, such as SIBS or PayPal, which are two of the most frequent methods of payment internationally and both offer a safe way to purchase things.

Once the website contained the said mechanisms, it would need a bigger amount of web servers and databases which could contain more user and transaction information as well as to provide users with a faster and more responsive interface, otherwise the only server as it exists now, could become clogged with an excessive amount of requests and become unresponsive.

Therefore, in order to accommodate for the cost of the additional hardware requirements, the MySportsFace application could include commercial advertisements, with the similarity of various other websites in the Internet, so as to generate the necessary income to keep the website up and running. The application already contains a very adequate spot to showcase these ads, which is located on the left side of most website pages, below the small user profile, thus only requiring to make contracts with other companies that would want their advertisements to be shown.

Moreover, regarding the post-deployment status of the website, there are two steps that could be taken to moderate the content added by its users. As such, whoever ran the website could employ a sufficient number of "moderators" to help view and analyze content posted or uploaded by users.
Alternatively, there could be a vetting system to dispose of unwanted content automatically (which is already being done in what regards videos, since Youtube and Vimeo already perform their own content vetting).

Additionally, in terms of the current "Starring" system of MySportsFace, there is a feature that could be added to bolster the user experience of the application. The creation of a notification system which would not only notify content owners of reaching certain popularity milestones on their posts and other content, but also enable the addition of several other notifications regarding useful information and alerts.

Lastly, during the first testing phase of the application, one of the inquired users suggested that in the future, some users might want to create multi-day or periodical events. In order to implement multi-day events, an extra column needs to be added to the custom events table in the database (to indicate the last day of the event) and the current date field needs to be changed to the event start date. Regarding the periodical events, there would have to be a database field indicating the period between event dates.
Bibliography


Appendix A

Usability Test Results

The following pages include the raw test results and graphs for both usability testing phases of the MySportsFace application.

A.1 First Testing Phase Results
<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
<th>Errors</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
<th>User 6</th>
<th>User 7</th>
<th>User 8</th>
<th>User 9</th>
<th>User 10</th>
<th>AVERAGE</th>
<th>STD DEVI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>23.6</td>
<td>0.1</td>
<td>21.1</td>
<td>0.2</td>
<td>25.8</td>
<td>0.1</td>
<td>31.8</td>
<td>0.1</td>
<td>24.2</td>
<td>0.2</td>
<td>22.1</td>
<td>0.3</td>
<td>23.0</td>
<td>26.5</td>
</tr>
<tr>
<td>Task 2</td>
<td>8.5</td>
<td>0.1</td>
<td>5.5</td>
<td>0.2</td>
<td>8.1</td>
<td>0.1</td>
<td>8.6</td>
<td>0.2</td>
<td>6.4</td>
<td>0.1</td>
<td>7.9</td>
<td>0.2</td>
<td>7.3</td>
<td>7.19</td>
</tr>
<tr>
<td>Task 3</td>
<td>28.5</td>
<td>0.2</td>
<td>17.5</td>
<td>0.0</td>
<td>24</td>
<td>0.0</td>
<td>27.6</td>
<td>0.0</td>
<td>23.2</td>
<td>0.0</td>
<td>22.8</td>
<td>0.0</td>
<td>20.4</td>
<td>23.83</td>
</tr>
<tr>
<td>Task 4</td>
<td>2.3</td>
<td>0.0</td>
<td>1.4</td>
<td>0.0</td>
<td>3.1</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>1.9</td>
<td>0.0</td>
<td>2.2</td>
<td>2.55</td>
</tr>
<tr>
<td>Task 5</td>
<td>25.1</td>
<td>0.0</td>
<td>13.7</td>
<td>0.0</td>
<td>18.2</td>
<td>0.0</td>
<td>24.5</td>
<td>0.0</td>
<td>20.9</td>
<td>0.0</td>
<td>21.3</td>
<td>0.0</td>
<td>20.4</td>
<td>21.02</td>
</tr>
<tr>
<td>Task 6</td>
<td>42</td>
<td>0.0</td>
<td>21.4</td>
<td>0.1</td>
<td>25.6</td>
<td>0.1</td>
<td>40.8</td>
<td>0.1</td>
<td>34.5</td>
<td>0.1</td>
<td>36.6</td>
<td>0.1</td>
<td>30.2</td>
<td>32.19</td>
</tr>
<tr>
<td>Task 7</td>
<td>76.6</td>
<td>0.0</td>
<td>45.8</td>
<td>0.0</td>
<td>59.2</td>
<td>0.0</td>
<td>71.4</td>
<td>0.0</td>
<td>63.9</td>
<td>0.0</td>
<td>69.2</td>
<td>0.0</td>
<td>61.7</td>
<td>64.46</td>
</tr>
<tr>
<td>Task 8</td>
<td>112.5</td>
<td>0.0</td>
<td>75</td>
<td>0.0</td>
<td>77.9</td>
<td>0.0</td>
<td>120.2</td>
<td>0.0</td>
<td>89.3</td>
<td>0.0</td>
<td>103.4</td>
<td>0.0</td>
<td>106.5</td>
<td>97.19</td>
</tr>
<tr>
<td>Task 9</td>
<td>45.1</td>
<td>0.0</td>
<td>27.9</td>
<td>0.0</td>
<td>35.8</td>
<td>0.0</td>
<td>41.9</td>
<td>0.0</td>
<td>36.9</td>
<td>0.0</td>
<td>38.3</td>
<td>0.0</td>
<td>40.1</td>
<td>38.94</td>
</tr>
<tr>
<td>Task 10</td>
<td>72.4</td>
<td>0.0</td>
<td>51.6</td>
<td>0.0</td>
<td>63.7</td>
<td>0.0</td>
<td>78.2</td>
<td>0.0</td>
<td>64.4</td>
<td>0.0</td>
<td>73.4</td>
<td>0.0</td>
<td>66.8</td>
<td>67.15</td>
</tr>
<tr>
<td>Task 11</td>
<td>6.5</td>
<td>0.0</td>
<td>4.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>8.2</td>
<td>0.0</td>
<td>7.1</td>
<td>0.0</td>
<td>6.6</td>
<td>0.0</td>
<td>7.8</td>
<td>6.78</td>
</tr>
</tbody>
</table>

**Averages**

<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
<th>Errors</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
<th>User 6</th>
<th>User 7</th>
<th>User 8</th>
<th>User 9</th>
<th>User 10</th>
<th>AVERAGE</th>
<th>STD DEVI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>23.6</td>
<td>0.1</td>
<td>21.1</td>
<td>0.2</td>
<td>25.8</td>
<td>0.1</td>
<td>31.8</td>
<td>0.1</td>
<td>24.2</td>
<td>0.2</td>
<td>22.1</td>
<td>0.3</td>
<td>23.0</td>
<td>26.5</td>
</tr>
<tr>
<td>Task 2</td>
<td>8.5</td>
<td>0.1</td>
<td>5.5</td>
<td>0.2</td>
<td>8.1</td>
<td>0.1</td>
<td>8.6</td>
<td>0.2</td>
<td>6.4</td>
<td>0.1</td>
<td>7.9</td>
<td>0.2</td>
<td>7.3</td>
<td>7.19</td>
</tr>
<tr>
<td>Task 3</td>
<td>28.5</td>
<td>0.2</td>
<td>17.5</td>
<td>0.0</td>
<td>24</td>
<td>0.0</td>
<td>27.6</td>
<td>0.0</td>
<td>23.2</td>
<td>0.0</td>
<td>22.8</td>
<td>0.0</td>
<td>20.4</td>
<td>23.83</td>
</tr>
<tr>
<td>Task 4</td>
<td>2.3</td>
<td>0.0</td>
<td>1.4</td>
<td>0.0</td>
<td>3.1</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>1.9</td>
<td>0.0</td>
<td>2.2</td>
<td>2.55</td>
</tr>
<tr>
<td>Task 5</td>
<td>25.1</td>
<td>0.0</td>
<td>13.7</td>
<td>0.0</td>
<td>18.2</td>
<td>0.0</td>
<td>24.5</td>
<td>0.0</td>
<td>20.9</td>
<td>0.0</td>
<td>21.3</td>
<td>0.0</td>
<td>20.4</td>
<td>21.02</td>
</tr>
<tr>
<td>Task 6</td>
<td>42</td>
<td>0.0</td>
<td>21.4</td>
<td>0.1</td>
<td>25.6</td>
<td>0.1</td>
<td>40.8</td>
<td>0.1</td>
<td>34.5</td>
<td>0.1</td>
<td>36.6</td>
<td>0.1</td>
<td>30.2</td>
<td>32.19</td>
</tr>
<tr>
<td>Task 7</td>
<td>76.6</td>
<td>0.0</td>
<td>45.8</td>
<td>0.0</td>
<td>59.2</td>
<td>0.0</td>
<td>71.4</td>
<td>0.0</td>
<td>63.9</td>
<td>0.0</td>
<td>69.2</td>
<td>0.0</td>
<td>61.7</td>
<td>64.46</td>
</tr>
<tr>
<td>Task 8</td>
<td>112.5</td>
<td>0.0</td>
<td>75</td>
<td>0.0</td>
<td>77.9</td>
<td>0.0</td>
<td>120.2</td>
<td>0.0</td>
<td>89.3</td>
<td>0.0</td>
<td>103.4</td>
<td>0.0</td>
<td>106.5</td>
<td>97.19</td>
</tr>
<tr>
<td>Task 9</td>
<td>45.1</td>
<td>0.0</td>
<td>27.9</td>
<td>0.0</td>
<td>35.8</td>
<td>0.0</td>
<td>41.9</td>
<td>0.0</td>
<td>36.9</td>
<td>0.0</td>
<td>38.3</td>
<td>0.0</td>
<td>40.1</td>
<td>38.94</td>
</tr>
<tr>
<td>Task 10</td>
<td>72.4</td>
<td>0.0</td>
<td>51.6</td>
<td>0.0</td>
<td>63.7</td>
<td>0.0</td>
<td>78.2</td>
<td>0.0</td>
<td>64.4</td>
<td>0.0</td>
<td>73.4</td>
<td>0.0</td>
<td>66.8</td>
<td>67.15</td>
</tr>
<tr>
<td>Task 11</td>
<td>6.5</td>
<td>0.0</td>
<td>4.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>8.2</td>
<td>0.0</td>
<td>7.1</td>
<td>0.0</td>
<td>6.6</td>
<td>0.0</td>
<td>7.8</td>
<td>6.78</td>
</tr>
</tbody>
</table>
A.2 Second Testing Phase Results
<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK 1</td>
<td>24.3</td>
<td>0.04347826087</td>
</tr>
<tr>
<td>TASK 2</td>
<td>23.2</td>
<td>0.04166666667</td>
</tr>
<tr>
<td>TASK 3</td>
<td>15.6</td>
<td>0.0</td>
</tr>
<tr>
<td>TASK 4</td>
<td>28.8</td>
<td>0.0</td>
</tr>
<tr>
<td>TASK 5</td>
<td>25.3</td>
<td>0.0</td>
</tr>
<tr>
<td>TASK 6</td>
<td>19.0</td>
<td>0.0</td>
</tr>
<tr>
<td>TASK 7</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>TASK 8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>TASK 9</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>TASK 10</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>TASK 11</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

**Average Time and Errors**

- **Time**: AVERAGE 6.833333333
- **Errors**: AVERAGE 0.04166666667
Appendix B

List of MySportsFace Requirements

This and the following pages list the requirements of the MySportsFace application.

Requirement #1

Description: Users must be able to create new accounts on MySportsFace
Rationale: Accounts are needed for performing various tasks through the website
Dependencies: none

Requirement #2

Description: Users must be able to sign in via Facebook or Google+ accounts
Rationale: Additional form of registration to the website so as to add flexibility
Dependencies: 1

Requirement #3

Description: The application must support sports organizations which contain extractable information through RSS feeds
Rationale: The information to be extracted comes from RSS feeds, thus the application needs to support the mechanism for extracting that information
Dependencies: none

Requirement #4

Description: Each sports organization supported must have their own page
Rationale: In order to organize the information, it should be divided on a per-organization basis
Dependencies: none
Requirement #5

**Description:** Users must be able to view a list of the sports organizations available

**Rationale:** In order to view news articles, users have to check the organization's pages

**Dependencies:** 4

---

Requirement #6

**Description:** All of the organizations’ news articles must be visible to logged in users

**Rationale:** Every article has to be visible to everyone so that it can be viewed and commented on

**Dependencies:** 4,5

---

Requirement #7

**Description:** Users must be able to add organizations to their list of favorites

**Rationale:** In order to facilitate the search for organizations users have more interest on

**Dependencies:** 4,5

---

Requirement #8

**Description:** Users must be able to write their comments on news articles and delete them if and when necessary

**Rationale:** So as to enable user interaction and socialization, users need a form of communication

**Dependencies:** 6

---

Requirement #9

**Description:** Users must be able to create new posts, view them on their recent posts feed and delete them if and when necessary

**Rationale:** Another form of communication between users

**Dependencies:** none

---

Requirement #10

**Description:** Users must be able to edit their profiles

**Rationale:** Allow users to customize their public information as they prefer

**Dependencies:** none

---

Requirement #11

**Description:** Users must be able to create custom events

**Rationale:** Custom events are a way to increase the participation of users in real life situations

**Dependencies:** none
Requirement #12

**Description:** Users must be able to add schedules to each event for which they are the creator of

**Rationale:** Event creators need to be able to show participants and other users the lineup for their events

**Dependencies:** 11

Requirement #13

**Description:** Users must be able to write comments on custom events

**Rationale:** Similar to the reasoning behind writing comments on news articles

**Dependencies:** 11

Requirement #14

**Description:** Users must be able to sign up for custom events

**Rationale:** Add a way for users to participate in community events

**Dependencies:** 11

Requirement #15

**Description:** Users who participate on an event must be able to upload images (photos) to that event’s page

**Rationale:** Sharing resources possibly taken during the events with other users

**Dependencies:** 11, 14

Requirement #16

**Description:** Users who participate on an event must be able to link their Youtube and Vimeo videos on the event’s page

**Rationale:** Sharing resources possibly taken during the events with other users

**Dependencies:** 11, 14

Requirement #17

**Description:** Event creators must be able to give the admin role to other participants of their event

**Rationale:** So as to allow event creators to ask other known and trusted users for help managing the events

**Dependencies:** 11, 14

Requirement #18

**Description:** Users must be able to follow and unfollow other users through their profiles

**Rationale:** Create a traversable user network of "followers" and "followees"
Dependencies: none

Requirement #19

Description: Users must be able to view microposts from users that they are following through their recent posts feed
Rationale: Users should be up to date with all the posts created by their friends and people they’re following in general
Dependencies: 9

Requirement #20

Description: Users must be able to view a list of all the custom events
Rationale: Allow the browsing of events
Dependencies: 11

Requirement #21

Description: Users must be able to view a list of all the participants for a specific event
Rationale: Allow the browsing of participants through events
Dependencies: 11,14

Requirement #22

Description: Users must be able to navigate the website intuitively through a navigation bar and additional menu buttons
Rationale: Facilitate the navigation of the website
Dependencies: none

Requirement #23

Description: The application must possess a content voting/rating system
Rationale: To allow users to show their opinion regarding content displayed in the application, through votes or ratings
Dependencies: 8,9,11,13

Requirement #24

Description: Users must be able to log out on demand
Rationale: Give freedom to the user regarding their account
Dependencies: 1,2
Appendix C

Usability Tests Specification

This and the following pages contain the list of tasks used during the usability testing stages.

C.1 Task 1

Description: Create a new account at MySportsFace
Attribute: Ease of use without previous experience
Measure(s): Time that the user takes to complete the task; Number of mistakes made while completing the task
Difficulty: Easy

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Acceptable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-20s</td>
<td>25-30s</td>
<td>40s</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table C.1: Time and error measurements for task 1.

C.2 Task 2

Description: Create a new post saying “Hello world!”
Attribute: Ease of use without previous experience
Measure(s): Time that the user takes to complete the task; Number of mistakes made while completing the task
Difficulty: Easy
### C.3 Task 3

**Description:** Follow a user called “Francisco Manuel”

**Attribute:** Ease of use without previous experience

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Easy

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5-10s</td>
<td>15-20s</td>
<td>40s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table C.2: Time and error measurements for task 2.

### C.4 Task 4

**Description:** View the posts of “Francisco Manuel” and count how many he has made

**Attribute:** Ease of use without previous experience

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Easy

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-3s</td>
<td>4-6s</td>
<td>8s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table C.4: Time and error measurements for task 4.
C.5 Task 5

**Description:** View the organization named “Federação de Desportos de Inverno de Portugal” and add it to your favorites

**Attribute:** Ease of use without previous experience

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Easy

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-15s</td>
<td>10-15s</td>
<td>20-25s</td>
<td>40s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table C.5: Time and error measurements for task 5.

C.6 Task 6

**Description:** Visit the news page for “Federação de Desportos de Inverno de Portugal”, view the article named “1º Curso de Roller Ski” and leave a comment on it saying “Exceptional”

**Attribute:** Ease of use without previous experience

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Medium

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-25s</td>
<td>20-25s</td>
<td>30-35s</td>
<td>45s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table C.6: Time and error measurements for task 6.
C.7 Task 7

**Description:** Create a new event with the following traits → name: “My First Personal Event”, summary: “This is my first event!”, description: “Sign up for my event below”, date: “30-October-15”, time: “10:00”, location: “Alameda, Lisboa, Portugal”

**Attribute:** Ease of use without previous experience

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Hard

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>40-45s</td>
<td>50-60s</td>
<td>90s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table C.7: Time and error measurements for task 7.

C.8 Task 8

**Description:** Fill in a schedule for your newly created event with the following entries → “10:00 - Event Check-in”, “11:00 - Event Opening Ceremony”, “12:00 - First Round”, “14:00 - Second Round”, “16:00 - Event Results”, “17:00 - Event Ending”

**Attribute:** Ease of use without previous experience; Migration of entries

**Measure(s):** Time that the user takes to complete the task; Number of mistakes made while completing the task

**Difficulty:** Hard

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-60s</td>
<td>70-80s</td>
<td>110s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table C.8: Time and error measurements for task 8.
C.9  Task 9

Description: Sign up for an event called “Surf Summer Split Championship” and upload an image “surf-2.jpg” from the desktop to the gallery

Attribute: Ease of use without previous experience

Measure(s): Time that the user takes to complete the task; Number of mistakes made while completing the task

Difficulty: Medium

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>25-30s</td>
<td>35-40s</td>
<td>50s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table C.9: Time and error measurements for task 9.

C.10  Task 10

Description: Sign up for another event called “Skyline Ski Tourney” and link any video from YouTube or Vimeo to the event page

Attribute: Ease of use without previous experience

Measure(s): Time that the user takes to complete the task; Number of mistakes made while completing the task

Difficulty: Medium

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>35-40s</td>
<td>45-55s</td>
<td>70s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table C.10: Time and error measurements for task 10.
C.11 Task 11

Description: Show your appreciation by giving the event a "star"

Attribute: Ease of use without previous experience

Measure(s): Time that the user takes to complete the task; Number of mistakes made while completing the task

Difficulty: Easy

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-5s</td>
<td>7-9s</td>
<td>15s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Optimal</th>
<th>Target</th>
<th>Maximum Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table C.11: Time and error measurements for task 11.