

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**GRADO UNIVERSITARIO EN
INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE
TELECOMUNICACIÓN**

TRABAJO FIN DE GRADO

**Design and Development of a Multiplatform Mobile
Application for a non-profit foundation based on the
framework Cordova**

**Alejandro Garrido Chasco
2019**

TRABAJO DE FIN DE GRADO

Título: Diseño y desarrollo de una aplicación móvil multiplataforma basada en el framework Cordova para una Fundación sin ánimo de lucro

Título (inglés): Design and Development of a Multiplatform Mobile Application for a non-profit foundation based on the framework Cordova

Autor: Alejandro Garrido Chasco

Tutor: Carlos Ángel Iglesias Fernández

Departamento: Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente: —

Vocal: —

Secretario: —

Suplente: —

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes



TRABAJO DE FIN DE GRADO

Design and Development of a Multiplatform Mobile
Application for a non-profit foundation based on the
framework Cordova

ENERO 2019

Resumen

El Trabajo de Fin de Grado consiste en la realización de una aplicación móvil multi-plataforma utilizando Cordova, el framework Onsen UI, NodeJS y una base de datos de tipo MySQL.

Esta aplicación está particularizada para una Fundación sin ánimo de lucro llamada Masnatur, la cuál realiza cada fin de semana actividades con personas con discapacidad. La aplicación permite controlar el acceso de todos los miembros de la Fundación, ofreciéndoles una interfaz intuitiva y sencilla. Cada usuario dispone de un login distinto según su rol dentro de la Fundación (y por tanto, de la aplicación). Los usuarios pueden apuntarse a futuras actividades si son voluntarios o incluso editarlas si trabajan en la oficina de la Fundación. La aplicación permite una comunicación sencilla a través del correo electrónico entre los voluntarios y los trabajadores. Además, los usuarios pueden editar sus perfiles e inscribirse a futuras actividades con sólo un click.

La aplicación se conecta a la base de datos de tipo MySQL para extraer y manejar la información almacenada. Los datos generados por la aplicación se guardan en tablas diferentes y no están conectados con la base de datos de la Fundación por seguridad.

Esta es la primera version funcional de un ambicioso proyecto que pretende ser capaz de soportar todos los distintos roles de los usuarios, unificar todas las bases de datos en una sola, ser lo suficientemente segura, etc.

El fin de este trabajo es crear una aplicación que aumente la fama de una determinada Fundación y mejore la comunicación entre todos sus miembros. La aplicación intenta ser lo más fluída, intuitiva, amigable y segura posible. Hay que tener en cuenta que el rango de edades de las personas participantes en la Fundación es muy amplio y por tanto, lo más importante es la sencillez.

Palabras clave: Aplicación móvil, Híbrida, Apache Cordova, Onsen UI, NodeJS, Mysql, PopSQL, Android, iOS.

Abstract

This Final year Project (TFG) consists in the creation of a multiplatform mobile application using Cordova, the Onsen UI framework, NodeJS and MySQL database.

This application is specific for a non-profit Foundation called Masnatur, which carries out activities every weekend with people with disabilities. It controls the access of every member to the Foundation, offering a simple and intuitive interface. Each user has a different login according to their role within the Foundation (and therefore, the application). The users are able to sign up for future activities if they are volunteers or even edit that activities if they work in the office. The application allows an easy communication between volunteers and workers via email. Users can also edit their profile and sign up for the activities they want just with one click.

The application also connects with a MySQL database to request and modify the stored information. The data is stored in different tables and it is not connected with the Foundations database for safety.

This is just the first functional prototype of an ambitious project that pretends to be able to manage other roles, unify every database in just one, become more and more secure, etc.

The purpose of this work is to create an application that increases the fame of a particular Foundation and improves communication among all its members. The application tries to be as fluid, intuitive, friendly and safe as possible. We must bear in mind that the age range of the users is very wide and therefore, the most important thing is simplicity.

Keywords: Mobile Application, Hybrid, Apache Cordova, Onsen UI, NodeJS, Mysql, PopSQL, Android, iOS.

Agradecimientos

Este es el fin de un camino duro e intenso,
donde he aprendido a combatir con esmero
infinitos miedos... Casi tantos como ineptos.
Pero hoy no toca hablar de ellos.

¿Debería contar hasta mi nota y fingir?
Quizás se me queda corta, para qué mentir.
La vida siempre te permite elegir
y estos versos, querido amigo, van para ti.

He reído más que llorado, gracias a ti.
He cantado, he bailado, he jugado,
he disfrutado, he compuesto y me he descompuesto.
Todo ello, también, gracias a ti

Si has aguantado este viaje a mi lado,
seguramente sin pelo te has quedado,
no por lo duro sino por lo largo...
Piensa que al menos, el cielo te has ganado.

Mis versos pueden parecer inconexos
o quizá soy yo quien no tiene remedio,
pero jamás podrás negarme que el mensaje
que llevan dentro es un bonito y sincero

TE QUIERO.

...para ti, que estás, y para los que se fueron...

Contents

Resumen	VII
Abstract	IX
Agradecimientos	XI
Contents	XIII
List of Figures	XVII
1 Introduction	1
1.1 Context	2
1.2 Project goals	3
1.3 Structure of this document	4
2 Enabling Technologies	5
2.1 Cordova	6
2.1.1 Architecture	7
2.1.2 Directory Structure	8
2.1.3 Plugins	9
2.2 NodeJS	11
2.2.1 Modules	11
2.3 Onsen UI	13
2.3.1 Components used	13
2.3.2 Onsen UI Playground	14
2.4 MySQL	15
3 Architecture	17
3.1 Overview	18
3.1.1 General blocks diagram	18
3.1.2 Levels	19
3.2 Front-End	20

3.2.1	Competitors	20
3.2.2	Application Directory	21
3.2.3	Onsen UI	23
3.2.4	Pug	24
3.2.5	Mock-Up	25
3.3	Back-End	26
3.3.1	Competitors	26
3.3.2	NodeJS	26
3.4	Database	28
3.4.1	Competitors	28
3.4.2	PopSQL and MySQL	28
3.5	Connection between Front-End, Back-End and the Database	30
4	Case study	31
4.1	Create a Personal Profile for the app	32
4.2	Access and edit your Personal Profile	33
4.3	Log in to the app using your email and your password	34
4.4	Watch a list of activities and search for specific ones	35
4.5	Access to a description of the activities	36
4.6	Sign up for an activity	37
4.7	Choose between the future or the past activities	38
4.8	Watch your activities	39
4.9	Send an email through the app	40
4.10	Access to all of the social networks of the Foundation	41
5	Conclusions	43
5.1	Conclusions	44
5.2	Problems faced	44
5.3	Future Objectives	45
A	Appendix	i
A.1	Global impacts and responsibilities	ii
A.2	Social Impact	ii
A.3	Economical Impact	ii
A.4	Responsibilities	ii
B	Appendix	iii
B.1	Context	iv

B.2	Physical resources	iv
B.3	Human resources	iv
B.4	Licenses	iv
Bibliography		v

List of Figures

2.1	Apache Cordova	6
2.2	Architecture of a Cordova Application [1]	7
2.3	Cordova Directory Structure	9
2.4	NodeJS	11
2.5	Onsen UI	13
2.6	Onsen UI Playground	14
2.7	MySQL	15
3.1	General blocks diagram	18
3.2	Application Directory	21
3.3	index.html code	22
3.4	app.js code	22
3.5	Pug file example	24
3.6	Mock-Up Views	25
3.7	package.json code	27
3.8	NodeJS modules explained with config.js file	27
3.9	PopSQL interface	28
3.10	“actividades” table in PopSQL	29
3.11	“perfiles” table in PopSQL	29
3.12	Code inside of the routes.js file for the login	30
4.1	Creation of a new profile	32
4.2	Profile created stored in the database	32
4.3	verPerfil.pug code and view	33
4.4	editarPerfil.pug code and view	33
4.5	Login views of the application	34
4.6	List of activities	35
4.7	Description of the activities	36
4.8	Signing up for an activity	37
4.9	Future activities	38
4.10	My personal activities	39

4.11 Sending an email	40
4.12 Goals and links of the Foundation	41

CHAPTER 1

Introduction

This chapter introduces the context of the project, including a brief overview of all the different parts discussed in the project. It also breaks down a series of objectives carried out during the realization of the project. Moreover, it introduces the structure of the document with an overview of each chapter.

1.1 Context

Nowadays, the easiest, fastest and most effective way to access information is with our mobile phones. For this reason, there are all kinds of applications, from calculators to maps, through notebooks, scanners, instruments, remote controls ... If your product is not accessible through an application, you have a very important handicap that will prevent you reaching your maximum potential.

In a constantly changing and increasingly technological world there are sectors that have not joined the change yet or have joined it making many mistakes. That is because creating a useful and attractive application is not easy at all. It requires programming skills, maintenance and motivation that is hard to achieve in most cases since today anyone can make a small application at home.

We are talking about the Third Sector entities. More specifically, Foundations and Non-Governmental Organizations that desperately need volunteers, but seek them in an antiquated way. This project aims to create a generic application for any Non-Governmental Organization and particularize it for the Masnatur Foundation, which lacks modern technological tools and urgently needs a renewal. In this way, if this project is successful it could be extended to other similar foundations to try to make this sector more visible and more accessible.

Moreover, eight years ago the operating systems of mobile devices were much more distributed among Android, IOS, Windows phone, Blackberry and other OS, but currently the market is controlled by IOS (16%) and Android (85.9%)¹. Ironically, before this duopoly, PhoneGap arose first and later Apache Cordova (its open source version) that allow to build hybrid applications, that is to say, applications that are not native but work in any operating system.

Although it may seem that this solution comes too late, the reality is that Apache Cordova it is a very useful tool that reduces development time by half. This framework also allows the application to be accessible to all kind of users.

¹<https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-27144>

1.2 Project goals

The main objective of this project is the creation of a generic, simple, intuitive and useful application that facilitates communication between volunteers, users and foundations. This project aims to be a base on which future functionalities can be developed.

The **main objectives** we want to achieve are:

- Design and implement a cross-platform mobile application to facilitate communication between the Foundation and its members.
- Local server that safely stores the personal information of users and activities.
- Smart login that handles dynamic content depending on who enters in the application.
- Access all the social networks of the Foundation through the application.
- Simplify the registration process in addition to facilitate and promote the participation of volunteers in the activities of the foundation.

1.3 Structure of this document

The remaining of this document is structured as follows:

Chapter 1 Explains the context in which this project is developed and also describes the main goals to achieve in this project.

Chapter 2 Describes the technologies and programs used during the creation of the application.

Chapter 3 Provides a general view of the architecture of the project. Describes each part and the way they are connected.

Chapter 4 Explains all the desired and achieved use cases.

Chapter 5 Comments the conclusions extracted from this project, the problems faced and the future work planned.

Enabling Technologies

In this chapter we analyze the technologies used during the development of this project. We describe in detail Cordova Framework, Onsen UI and Nodejs since they are the main base of the application. We also explain other tools that have been used, such as the database management system (MySQL) and the Mock-Up editor.

2.1 Cordova

Cordova[1][2] is a software previously known as “PhoneGap”, then “Apache Callback”, finally “Apache Cordova”. PhoneGap is Adobe’s commercial version of Cordova along with its associated ecosystem. Other famous tools and frameworks (Ionic, Onsen UI, Monaca, etc) have been built on top of Cordova instead of PhoneGap.

As previously mentioned, Apache Cordova is an open-source mobile development framework that allows you to develop cross-platform apps using HTML5, CSS3 and JavaScript instead of relying on platform-specific APIs like those in Android, iOS, etc. The resulting applications are neither native mobile application nor purely Web-based, they are hybrid. Cordova provides the Cross-platform (CLI) workflow, used to run on as many different mobile operating systems as possible, with little need for platform-specific development. In other words, Cordova allows us to create new projects, build them on different platforms, and run them within an emulator.

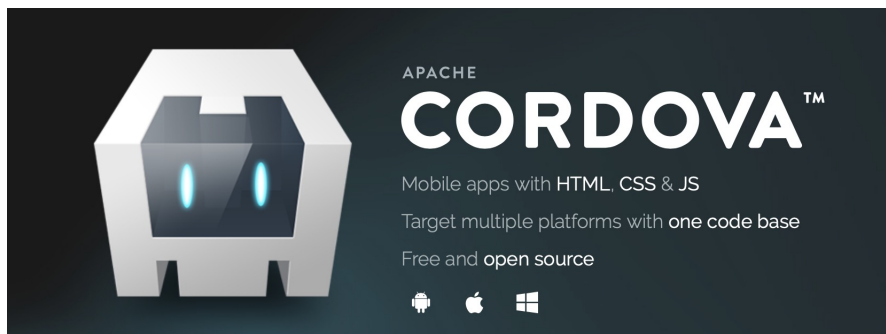


Figure 2.1: Apache Cordova

2.1.1 Architecture

There are several components to a cordova application. The following diagram shows a high-level view of the cordova application architecture extracted directly from the Cordova website.¹ We are also going to describe each part of the diagram:

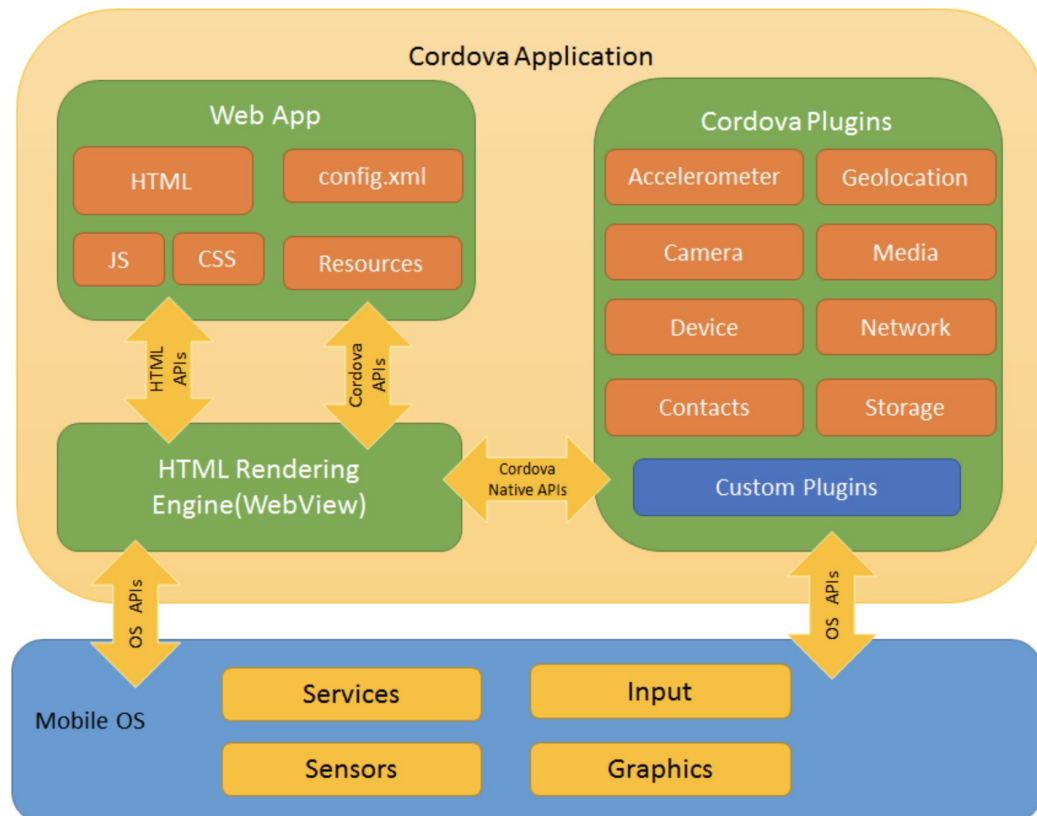


Figure 2.2: Architecture of a Cordova Application [1]

Here we will describe the different parts shown in the figure above. Inside the Cordova Application we can see the WebView, Web App and the Plugins. The Development Paths correspond to the workflows provided to create the application. Here we explain everything much more detailed:

- **WebView:** The Cordova-enabled WebView may provide the application with its entire user interface. It is a HTML rendering engine that can be mixed with native application components on some platforms. It is used together with the Web App files.

¹<https://cordova.apache.org/docs/en/latest/guide/overview/index.html>

- **Web App:** This is the part where the application code resides as it includes all the HTML5, CSS3, and JavaScript files. The application itself is implemented as a web page, by default a local file named `index.html`, that references everything else. It also contains the essential file `config.xml` described before.
- **Plugins:** The plugins provide an interface for Cordova and native components to communicate with each other and bindings to standard device APIs. Because of them, we can invoke native code from JavaScript. Cordova offers a set of plugins called Core Plugins which are easy to implement but there are also some useful several third-party plugins. The plugins we can see in the figure are not the ones installed in our project.
- **Development Paths:** Cordova provides two basic workflows to create a mobile application. The one we have used in this project is the Cross-platform (CLI) workflow which allows the application to run on as many different mobile operating systems as possible.

2.1.2 Directory Structure

The directory structure is always the same for all Cordova projects. It contains the same folders, although in some of them, the content changes. For example, the plugins installed by the developer are the only ones included inside the `plugins/` folder, so in some cases, that folder could be empty.

The Cordova projects have the following **Directory structure:** ²

Every Cordova project has this structure shown in the figure above. We are going to name each part and resume how it works in the following list:

- **Config.xml:** file used to configure and custom your application. It provides information about the app and specifies parameters affecting how it works
- **www/** Folder which contains essential files for a proper functioning of the application. This files are `.html` `.css` and `.js`, and in our case, the `www/` folder also contains the dynamic `.pug` views.

²<https://cordova.apache.org/docs/en/latest/reference/cordova-cli/index.html>

```
myapp/  
|-- config.xml  
|-- hooks/  
|-- merges/  
| | |-- android/  
| | |-- windows/  
| | |-- ios/  
|-- www/  
|-- platforms/  
| |-- android/  
| |-- windows/  
| |-- ios/  
|-- plugins/  
    |--cordova-plugin-camera/
```

Figure 2.3: Cordova Directory Structure

- **platforms/** Folder that contains all the source code and build scripts for the platforms added to our project.
- **plugins/** Any added plugins are extracted or copied into this directory. Some plugins helps the application look more native.
- **hooks/** It is a folder that contains scripts used to customize cordova-cli commands. A typical use may be creating a script that rebuilds the project for testing updates quickly.
- **merges/** It is a directory that can contain .html .css or JavaScript files. These files placed under merges/ will override matching files in the www/ folder.

2.1.3 Plugins

As we have introduced before, a Cordova plugin creates an interface for the browser based code to access native device functionality. It can be used regardless of the framework.³

Though there are lot of plugins available, we have not used many of them yet. It is important to explain that this plugins can be installed or removed whenever the developer wants, so there will surely be future updates. For the moment, this are some of the plugins used in this project:

³<https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/index.html>

- **cordova-plugin-console:** This plugin supports many methods of the console object defining a global console object. This is really useful as the application works in many Operating Systems. This way, we ensure that `console.log()` is as helpful as it can be.
- **cordova-plugin-device:** Because of this plugin we will be able to obtain lot of properties such as the model, platform, version, uuid, etc from the device running the application. It supports almost every Cordova platform and this plugin is only available after the `deviceready` event.
- **cordova-plugin-whitelist:** Because of this plugin, the user will be able to navigate to the URLs included in a whitelist policy. It must be used carefully because we do not want to download any malware.

2.2 NodeJS

Node.js[3] is a server-side Javascript environment, with great performance, based on events and, therefore, asynchronous. This language works on the vast majority of servers. One of the biggest advantages of Node.js is that it offers numerous modules that contain easy-to-add functionality to our project.



Figure 2.4: NodeJS

2.2.1 Modules

There are many modules available but we have used just a few of them. Here we can see a selection of the most important ones:

- **body-parser:** It is a middleware that extracts the entire body portion of an incoming request stream and exposes it on `req.body`. It parses the JSON, buffer, string and URL encoded data submitted. This module was included in express 3 but since the update to express 4, it must be installed separately.
- **express:** It is a web application framework for Node.js. It also provides numerous tools to ensure proper management of http servers. The application is able to respond to requests with route support due to the use of this module. It is essential for the application to work properly.
- **express-session:** It is a simple session middleware for Express which allows managing sessions but not storing them. That's why we need to use it with the `express-mysql-session` module.

- **express-mysql-session:** It is a MySQL session store that creates a connection pool and a database table to save session data. It is an extension of express-session used to stored the data.
- **file system:** This module allows us to read, upload, update, rename or delete files from the file system of our computer.
- **mysql:** It is used to manipulate MySQL databases creating connections and querying of database.
- **nodemailer:** This module allows the application to send emails after creating a transport object, which in our case is the SMTP transport.
- **xoauth2:** This module is needed to use XOauth2 token generators. XOAuth2 generator generates required accessToken itself if it is missing or expired. In this application it is used together with Nodemailer to allow gmail to send an email through the application to a personal mail.
- **pug:** It is a template engine, formerly known as Jade, for dynamic views that allows us to inject data and then produce HTML. In our project, we work with this dynamic views but also with static content such as images, JS and CSS files.

2.3 Onsen UI

Onsen UI[4] is an open source software that presents lots of UI components packed with ready-to-implement features specially designed for mobile apps. Every component follows the native iOS and Android design standards so its a great option for making our Hybrid app fell native. Throughout this project we have used some Onsen components although we have had to edit some of them.



Figure 2.5: Onsen UI

2.3.1 Components used

Every single view we have in our application has an “onsen-page” tag instead the “body” one. So this framework is a heavy component. Here we can see some of the components used in the application even though we have included others:

- **Speed Dial:** A button that displays a menu with elements when its clicked. In our case we use this component for showing the social network links: Facebook, Twitter, Youtube, Website, etc.
- **The Splitter:** It creates a swipeable menu which is attached to a specific side. The content of the view changes when the user clicks on one of the items in the menu. In our case we use this component in a view that explains the main objectives of the Foundation. Thanks to this component, the user can change from one view to another very easily and without exiting the original view.
- **Input:** It provides a normal input text but with the Onsen UI style. The button incorporated had been changed to include the property of “submit”. This input is used in the index.html view to login.

- **Select Input:** This component displays a select box with an arbitrary number of options. In our case it is used to choose the role of the users from our application.

2.3.2 Onsen UI Playground

This framework offers a “playground” in the website to test the different components⁴. It is very intuitive and offers the user to test all the possible components, along with other templates that bring the most important ones together. This playground has been really useful throughout the project development as we have been able to compare and choose the best option.

Here we can see how the playground looks like:

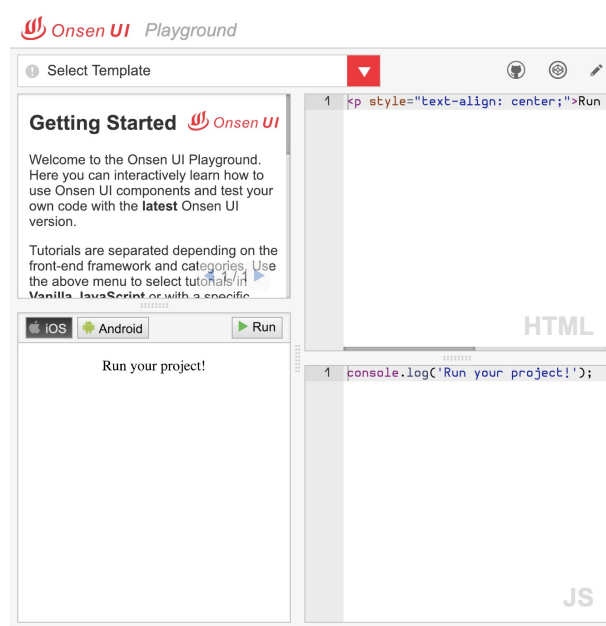


Figure 2.6: Onsen UI Playground

⁴<https://onsen.io/playground/>

2.4 MySQL

MySQL[5] is a database management system that uses Structured Query Language (SQL) where databases are relational. Its an open Source software backed by Oracle Corporation. It runs as a server and allows multiple users to manage and create numerous databases. MySQL can also run on various platforms such as UNIX, Linux, Windows, etc. It is an important component in the LAMP (Linux, Apache, MySQL and PHP) stack of open source web application software that is used to create websites.

Our application requires MySQL to store and manipulate all of its data. This information is stored in four main tables: “perfiles” which contains the basic data from the users, “actividades” which stores the different activities of the Foundation, “volunACT” which contains the activities with users that have signed up, and “sessions” that stores essential information from the users that have logged in the application.



Figure 2.7: MySQL

Architecture

This chapter explains the architecture of the entire project, including the design phase, the implementation details and the server requirements. We show a global vision of the project to identify the different parts involved, focusing on each module and its relation with the project.

3.1 Overview

3.1.1 General blocks diagram

Here we can find a brief description of the architecture of the whole project. The user interacts with the Front-End throughout the mobile phone application which has been downloaded from the proper market. The views displayed in the mobile phone are loaded throughout routes as it will be explained afterwards. The Back-End translates the Front-End interactions made by the user to a language and guidelines that the computer can understand. All the useful information is stored inside the database in tables. This database is kept in a server and can only be accessed with queries throughout the Back-End code. The user will not have the access to the Back-End code or the database, which means that the only way he can manage his personal information is by throughout the Front-End code. The developer however, can access directly to the Back-End code and control the whole application. The server is local and contains the Application code and the database. In the next figure we can see the diagram of the application:

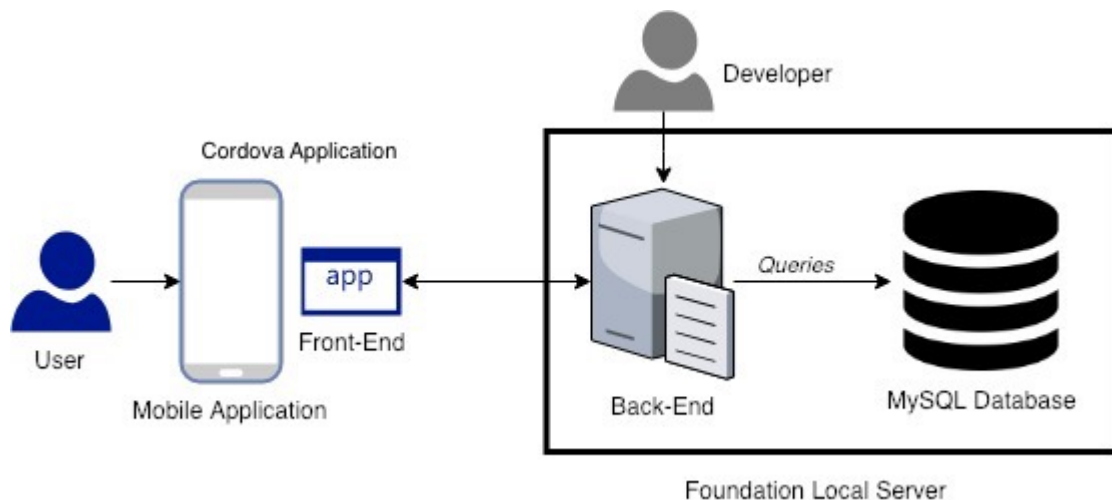


Figure 3.1: General blocks diagram

3.1.2 Levels

The architecture of the developed application, as shown in 3.1, consist of three modules:

- **Front-End:** Known as the user interface that allows them to interact with our code. As we are working with Cordova, we will use the standard web technologies for developing this phase instead of using Vue.js¹, Angular.js² or React.js³. We have chosen to work with pure javascript because it generates a faster and cleaner code. However, we have used some components of the framework “Onsen UI”.
- **Back-End:** Known as the code that runs on the server-side. In this case, there are also lots of different frameworks (Django⁴, Ruby-On-Rails⁵, Flask..) but we selected Express as we are working on the run-time environment called NodeJS. This duo work with JavaScript and have lots of advantages such as a huge community, the possibility of creating cross-platforms apps, etc.
- **Database:** Here we store all the information generated by our app. We have three similar options: MongoDB⁶, MySQL or PostgreSQL⁷. The main difference between them is that the first one is non-relational and the second and third one work as a relational database management system. MySQL and PostgreSQL are very similar, we picked MySQL but there would not have been much difference if we had selected PostgreSQL. For local tests, we used the MySQL server.

¹<https://vuejs.org>

²<https://angularjs.org>

³<https://reactjs.org>

⁴<https://www.djangoproject.com>

⁵<https://rubyonrails.org>

⁶<https://www.mongodb.com/es>

⁷<https://www.postgresql.org>

3.2 Front-End

3.2.1 Competitors

There are many different options for the Front-End design. The Front-End of an application is one of the most important factors to succeed because it is the first thing that a user sees. This factor not only makes the code look beautiful, it also make the application easy to use.

Even though we are going to talk about the Onsen UI framework, there are other possible frameworks which are really useful and interesting. In this case, using one component does not limit your application to just one framework. In fact, the majority of our components are from Onsen UI but we have used Bootstrap ones too.⁸ Bootstrap allows us to build easily a particular component in our application by writting a few lines in our HTML and CSS files. It is important to remember that the CSS file is normally an url loaded by our code so it means that we need to be connected to the internet to use the component. This is solved by copying and saving the CSS code to a file inside our application directory, to the same folder in which Onsen UI components have been saved.

Another competitor to our Onsen UI framework is Ionic. One important difference is that Onsen UI has an available playground to test the different components and try new features. Also, the possibility to work only with pure JavaScript was another important factor that made us choose Onsen UI.

We also found many competitors when we had to decide which view engine to use. The three main competitors are Handlebars, EJS and Mustache. Every of these view engines works properly and they are quite similar, so it was not an essential choice.

Even so, the reasons that made us pick .pug were:

- **Handlebars**⁹: The aspect of a Handlebars file is very similar to an HTML one, but with special components that loads the variables you want. The worst thing about this view engine is the lack of quality tutorials. If you want to use this files perfectly, you need to make your own tests and practice a lot.

⁸<https://getbootstrap.com/docs/4.2/components/alerts/>

⁹<https://handlebarsjs.com>

- **EJS**¹⁰: This type of files are not so friendly and similar to HTML as the rest of the competitors. That is why EJS was rejected.
- **Mustache**¹¹: Handlebars is an extension of this type of files, so if we rejected Handlebars, it would have made no sense to choose this one. The appearance is also not very friendly so it is not very easy-reading. Other important factor is that the basic tasks are a little bit difficult in this type of files.

3.2.2 Application Directory

As we have said before, we are using HTML5, CSS3, and JavaScript to develop the application. All these created files are inside the `www/` folder of the Cordova directory. The files are distributed as follows:

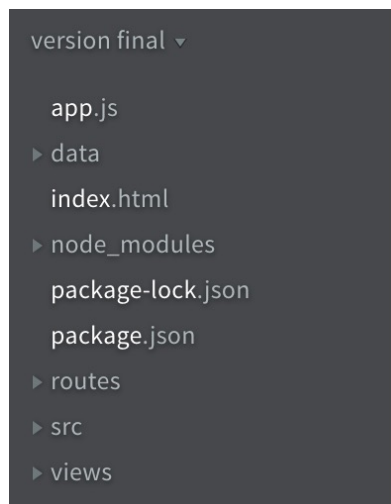


Figure 3.2: Application Directory

The folders and files shown above compose the whole application. Each folder has one or more files but only the most important ones will be explained. On top of the image you can find the name of the project “version final”. We will describe each part of the directory in the following list:

- **index.html** contains the main view in HTML format. As we can see in the figure, the file is very simple: it is composed by a logo and a button. This is the cover of the

¹⁰<https://ejs.co>

¹¹<https://mustache.github.io>

application as Cordova needs a HTML file to load everything else. After clicking the button, the user is redirected to the index.pug file, which is the login view.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <!--Links that loads the CSS and JS files-->
5      <link rel="stylesheet" href="src/todo.css">
6      <script type="text/javascript" src="/routes/routes.js"></script>
7      <title>Masnatur</title>
8    </head>
9    <body class='bodyprincipal'>
10     <div style="text-align: center; margin-top: 30px;">
11       <!--Image of the Foundation logo and button that starts the application-->
12       <img src='src/logo%20viejo.jpg' width='40%'>
13     </div>
14     <form action="http://localhost:3002/" method="get">
15       <button class='buttonprincipal' type="input">Access to the app</button>
16     </form>
17   </body>
18 </html>

```

Figure 3.3: index.html code

- **app.js** starts the server. It contains information of the MySQL server inside of a code that loads the NodeJS modules, sets pug as the view engine and starts the server in the correct port. In the next figure we can see an extract of the app.js file.

```

1  // Require packages and set the port
2  const express = require('express');
3  const port = process.env.PORT || 3002;
4  const bodyParser = require('body-parser');
5  const app = express();
6  const routes = require('./routes/routes');
7  var path = require('path');
8
9  app.use('/', express.static(path.join(__dirname, 'src')));
10
11 var session = require('express-session');
12
13 var MySQLStore = require('express-mysql-session')(session);
14
15 var options = {
16   host: 'localhost',
17   user: 'root',
18   password: 'password',
19   database: 'masnaturCF',
20   clearExpired: true,
21   checkExpirationInterval: 900000,
22   expiration: 86400000,
23   createDatabaseTable: true,
24   endConnectionOnClose: true,
25   charset: 'utf8mb4_bin',
26   schema: {
27     tableName: 'sessions',
28     columnNames: {
29       session_id: 'session_id',
30       expires: 'expires',
31       data: 'data'
32     }
33   }
34 };
35
36 var sessionStore = new MySQLStore(options);
37
50 // View engine: PUG
51 app.set('view engine', 'pug');
52
53 // Use Node.js body parsing middleware
54 app.use(bodyParser.json());
55 app.use(bodyParser.urlencoded({
56   extended: true,
57 }));
58
59 routes(app);
60
61
62 // Here we start the server
63 const server = app.listen(port, (error) => {
64   if (error) return console.log(`Error: ${error}`);
65   console.log(`Server listening on port
66     ${server.address().port}`);
67 });
68
69
70 module.exports = sessionStore;

```

Figure 3.4: app.js code

- **routes/** contains files with the different navigation routes that makes our application fluid. The main file is called `routes.js` and it will be described at the end of this chapter.
- **src/** contains the CSS and JS files that make the application beautiful and friendly. Inside this folder we can also find the pictures used in our application. As we have written in the `app.js` file, the static files will be stored inside this directory.
- **views/** contains the dynamic `.pug` views that compose the application. These files are loaded through the routes that will be explained at the end of this chapter.

3.2.3 Onsen UI

Inside the folders `css/` and `css-components-src/` we find all the necessary files for the Onsen UI components to work. By placing all the files within these folders, we decrease the loading speed of the application although it uses more storage on our phone. In our directory we can find the CSS and JavaScript files needed for a correct functionality of every single Onsen UI component. We are using only the components described in the chapter 2 we have been working only with four files: “`onsen-css-components.css`”, “`onsenui.css`”, “`theme.css`” and “`onsenui.js`”. Apart from them, we have sometimes included our personal CSS and JS files. Onsen also allows us to access to the CSS and JS files and edit them but it is a tough task because of the length of the files.

3.2.4 Pug

We are using dynamic views that are rendered through routes. These .pug files are hosted in the views/ folder. There are many view engines available: ejs, pug, handlebars, etc, but pug has a free online converter and is very easy to understand thanks to its similarity with html. The .pug code shows a html structure but without the typical characters. Below we can see an example of a .pug file with comments explaining each part involved.

```
1 doctype html
2 head
3   // Links to CSS and JS files and character encoding for Unicode
4   link(rel='stylesheet' href='css/onsenui.css')
5   link(rel='stylesheet' href='css/onsen-css-components.css')
6   link(rel='stylesheet' href='css-components-src/src/theme.css')
7   link(rel='stylesheet' href='todo.css')
8   script(src='js/onsenui.js')
9   script(src='oleoleJS/JSindex.js')
10  meta(http-equiv='Content-Type' content='text/html; charset=utf-8')
11  // We need to work with ons-pages instead of body in order to use Onsen UI components
12  ons-page(class='onspagefondo')
13    div(style='text-align: center; margin-top: 30px;')
14      // Title of the application
15      font(color='green' size='20') MASNATUR
16      br
17      // Form to access to the information of the Foundation
18      form(action='http://localhost:3002/quienSomos' method='get' id='myform')
19        button(class='botoncitos' type='submit') ¿Quiénes s&oacute;mos?
20      br
21      br
22      // Form to login into the application
23      // This is an Onsen UI component
24      form(action='http://localhost:3002/loginFIN' method='post')
25        p
26          ons-input#correo(type='email' name='correo' modifier='underbar' placeholder='Correo' float='')
27        p
28          ons-input#password(name='password' modifier='underbar' type='password' placeholder='Password' float='')
29        br
30        p(style='margin-top: 30px;')
31          button(class='botoncitos' type='submit') Entrar
32      // Alert that pops up when you have forgotten your password
33      form
34        button(class='botoncitos' modifier='quiet' onclick='alert("ayyy pillin.. Llama a la oficina!");') ¿Olvidaste la
35        contrase&ntilde;a?
36      br
37      // Link to create a new profile
38      div(style='text-align: center; margin-bottom: 30px;')
39        form(action='http://localhost:3002/crearPerfil' method='get')
40          button(class='botoncitos' type='submit') Si a&uacute;n no tienes un perfil, pincha aqu&iacute;
```

Figure 3.5: Pug file example

3.2.5 Mock-Up

Before this project started, we created a mock-up in which we built a purely visual prototype that included all the desired use cases, as well as navigation between views and a possible graphic design. This was an initial prototype and therefore, during the subsequent development of the project, we have changed certain functionalities in order to gain a complete application.

Below we can see a screenshot with all the views available in the mock-up, but also you can watch the entire Mock-Up through this references[6][7]

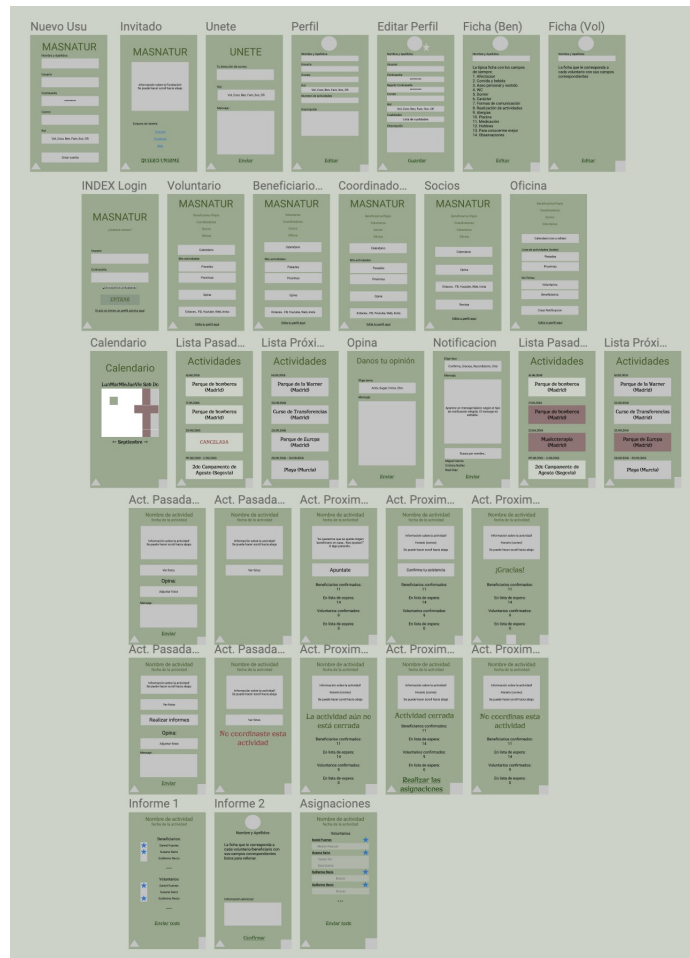


Figure 3.6: Mock-Up Views

3.3 Back-End

3.3.1 Competitors

In this part of the project we find the most varied competitors as we can program in many different languages. If you want to program in Ruby, then Ruby-on-Rails is the best framework option, but if you prefer programming in Python, then you may choose Django. In our case the decision was very easy because the main programming language in our university is Java. That is why NodeJS was the first and only choice as it works with JavaScript. Even though it is relatively modern, it is very solid, specially if it is used together with ExpressJS.

3.3.2 NodeJS

As described before, Node.js is an application runtime environment that allows us to write server-side applications in JavaScript. It comes with many APIs suitable for backend development. We have chosen Node because using it as our server technology gives us a great boost that comes from using the same language on both the front end and the back end.

Thanks to NodeJS and its modules, we are able to connect the user and the database. Once we install the modules, the “dependencies” property of our package.json file changes. In the figure 3.7 we can see how this package.json file looks like:

As we can see in the application directory, there is a file called package-lock.json very similar to the one above. The package-lock.json file records the exact version of each installed package which allows us to re-install them. Because of this, we will be able to build an identical dependency tree in future installs. The file shown in the figure records only the minimum version needed. The version updates won't be reflected here.

Below we can see some commented code explaining the behavior of the modules. It is an extract of the config.js file, which contains information about the connections, using the NodeJS modules, between our code and the MySQL database.

```

1  {
2    "name": "rest",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "author": "alex",
9    "license": "ISC",
10   "dependencies": {
11     "body-parser": "^1.18.3",
12     "express": "^4.16.4",
13     "express-mysql-session": "^2.0.1",
14     "express-session": "^1.15.6",
15     "fs": "0.0.1-security",
16     "localStorage": "^1.0.4",
17     "mysql": "^2.16.0",
18     "nodemailer": "^4.7.0",
19     "pug": "^2.0.3",
20     "request": "^2.88.0",
21     "xoauth2": "^1.2.0"
22   },
23   "devDependencies": {},
24   "description": ""
25 }
26 |

```

Figure 3.7: package.json code

```

1  // Here we LOAD the MySQL module that allows the connection with the database.
2  const mysql = require('mysql');
3
4  // Here we set the database connection credentials for the pool
5  const config1 = {
6    host: 'localhost',
7    user: 'root',
8    password: 'password',
9    database: 'masnaturCF',
10 };
11 // Here we create a MySQL pool with the "config1" properties
12 const pool = mysql.createPool(config1);
13
14 // We set the database connection credentials
15 // We can see that multiple statements are allowed in this connection, that option is false by default to increase security.
16 // That is why we have created a new connection to unlock this property.
17 // This connection will only be used to delete database content.
18 const config2 = {
19   host: 'localhost',
20   user: 'root',
21   password: 'password',
22   database: 'masnaturCF',
23   multipleStatements: true,
24 };
25 // Here we create a MySQL pool with the "config2" properties
26 const connection = mysql.createConnection(config2);
27
28 // We need to export this connections to be used in other files.
29 module.exports = pool;
30 module.exports = connection;
31

```

Figure 3.8: NodeJS modules explained with config.js file

3.4 Database

3.4.1 Competitors

As we have commented at the beginning of this chapter, we were hesitating between MongoDB, Postgresql and MySQL. The first one was rejected because it is a non-relational database management system, so even if it is one of the best choices to manage databases, in our case it was useless because we need to connect the different tables to obtain useful results. An example of this is the voluACT table which joins information of users and activities in the same place. So the choice was between the other two. They are very similar but MySQL works better with tables. As this Foundation (Masnatur) stores their data in tables, MySQL was the best option. We interact with this database through queries written in SQL language. For security, by default only one connection is allowed in the same query as we have described before.

3.4.2 PopSQL and MySQL

We have chosen the program PopSQL[8] (the free version) to work directly with the MySQL database and because of this we have been able to quickly verify that all the changes had been made correctly. This program allows us to collaborate in realtime with other developers, we can also share queries by URL, and organize them in folders, visualize our data automatically and not only works with MySQL. The bad news are that PopSQL does not support MongoDB, but in our case that does not really matter. Here is a screenshot of the PopSQL interface:

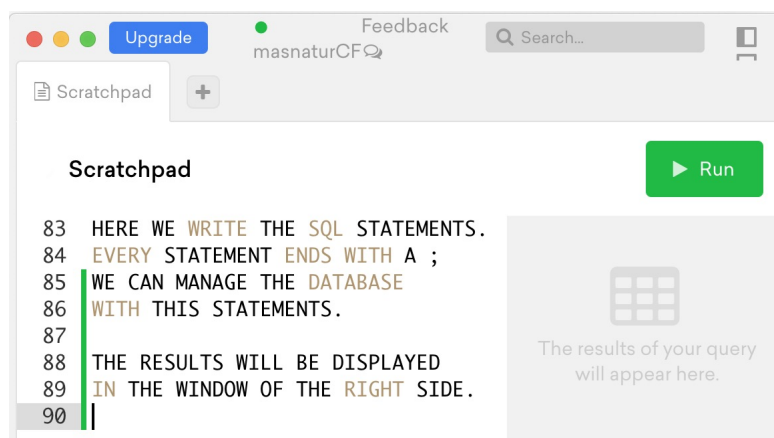


Figure 3.9: PopSQL interface

The information stored by our application is distributed between the following tables: perfiles, actividades, volunACT and sessions. The fields of these tables will be explained in the use cases from the chapter 5, so here we will only show the content and the statements used to create the “actividades” and “perfiles” tables (the most important ones) in the next screenshots:

```
CREATE TABLE actividades (
  act_id INT unsigned NOT NULL AUTO_INCREMENT,
  dia DATE,
  actividad VARCHAR(250) DEFAULT '',
  lugar VARCHAR(50) NOT NULL,
  descripcion VARCHAR(600) DEFAULT '',
  PRIMARY KEY (act_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SELECT * FROM actividades;
```

act_id	dia	actividad	lugar	descri
1	1999-04-04		Madrid	
2	2011-05-04	Día musical en el matadero	Matadero	Vamos
3	2018-11-24	Bolera	Parque Sur	Bolera
4	2018-11-25	Bolera	Parque Sur	Lo mis
5	2018-12-01	Teatro inclusivo	Sede	Pues t
6	2018-12-02	Teatro inclusivo	Sede	Otra rr
7	2018-12-09	Taller de musicoterapia	Sede	Musiq
8	2019-01-01	Año nuevo	Casa	FELIZ
9	2018-12-16	Fiesta de Navidad	Hotel	Fiestoi

Figure 3.10: “actividades” table in PopSQL

```
CREATE TABLE perfiles (
  id INT unsigned NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(40) DEFAULT '',
  correo VARCHAR(50) NOT NULL,
  password VARCHAR(20) NOT NULL,
  rol VARCHAR(15) DEFAULT '',
  descripcion VARCHAR(400) DEFAULT '',
  session_id VARCHAR(150) DEFAULT '',
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

SELECT * FROM perfiles;
```

id	nombre	correo	password	rol	descripc
1	asraf	asd@ale.com	aaa	voluntario	Es un cur
2	lucas	lucas@hotmail.com	pp	voluntario	
3	pepe	pepe@hotmail.com	pepe	Socio	
4	Tito	tito@hotmail.com	con	Familiar	
5	aass	aass@hotmail.com	aass	voluntario	
5	titi	titi@hotmail.com	titi	beneficiario	
7	Diego	dieguito@gmail.com	die	beneficiario	
3	sandra	saandraaa@hotmail	guapa	beneficiario	

Figure 3.11: “perfiles” table in PopSQL

3.5 Connection between Front-End, Back-End and the Database

Queries to the database are made through the code written in the routes.js file which is connected to the forms inside the .pug views. The user interacts with the form included in the .pug file and the stored data is managed by the routes.js file with queries to the database, thanks to NodeJS and its modules. Below we can see a detailed and commented example of these connections.

In the next figure we will find the main route of the application explained. We will only see the code inside of the routes.js file. The login is controlled with two queries that compares the data inserted by the user with the data stored in the “perfiles” table from the database. We can also notice that the session store object used is very similar to the request.session storage. In fact, there is no need to create the session store object, but we have decided to do it because it is easier to control the information that stores.

```

44 // LOGIN ROUTE
45 app.post('/loginFIN', (request, response) => {
46 // Global variable that stores the session.
47 sessionStore = request.session;
48 // Query that compares the email introduced with the emails stored in the database.
49 pool.query('SELECT * FROM perfiles WHERE correo = ?', request.body.correo, function
(error, results, fields) {
50 if (error) {
51 response.send('error');
52 }
53 else{
54 sessionStore.idperfil = results[0].id;
55 if(results.length >0){
56 // If the password exists:
57 if(results[0].password == request.body.password){
58 // Stores the session id in the row of the user.
59 pool.query('UPDATE perfiles SET session_id =? WHERE id = ?',
[request.session.id, results[0].id], (error, results) => {
60 if (error) throw error;
61 });
62 }
63 request.session.correo = results[0].correo;
64 sessionStore.correo=results[0].correo;
65 // Renders the main view if the login is correct.
66 response.render('voluntarios');
67 }
68 else{
69 // Reloads the login view if the login is incorrect.
70 response.render('index1');
71 }
72 };
73 });
74 });
75 });

```

Figure 3.12: Code inside of the routes.js file for the login

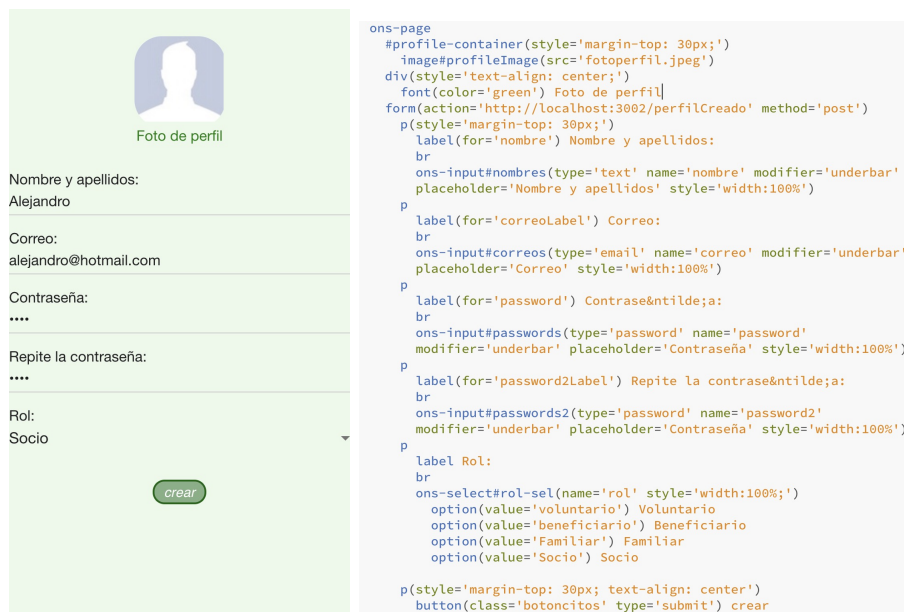
The rest of the login code was explained above in this chapter when we described the differences between the .pug and the .html files. There we can find the login form that complete this route. This particular form is an Onsen UI component.

Case study

This chapter describes the needs of the different users, presented as a walk-through among all the options offered by the application. We will also show the code that allows this use cases work and the corresponding view. Here is a list of the user cases:

4.1 Create a Personal Profile for the app

Every user that enters the app must have a profile and provide us some personal information which is stored in the table called “perfiles” from our database. The information we ask for is: name, email, password, a brief description of the personality of the user and the role that the user has inside the Foundation (volunteer, “beneficiary”, worker...).



The figure shows a web form for creating a new profile on the left and its corresponding HTML code on the right. The form has a light green background and includes a profile picture placeholder labeled "Foto de perfil". Below it are input fields for "Nombre y apellidos:" (containing "Alejandro"), "Correo:" (containing "alejandro@hotmail.com"), "Contraseña:" (masked with dots), "Repite la contraseña:" (also masked), and a "Rol:" dropdown menu (set to "Socio"). A green "crear" button is at the bottom. The code on the right is a snippet of HTML using the `ons-` prefix for components, showing the structure of the form with labels, inputs, and a select element for the role.

Figure 4.1: Creation of a new profile

The JavaScript used in the creation of the profile controls that the email is realistic and the password is the same in both fields. As we can see in the figure above, the password is hidden and the type of the email input forces the user to introduce a valid email.

After clicking the “crear” button, the route stores the information inserted by the user inside of the database table “perfiles” as we can confirm in the next figure:

id	nombre	correo	password	rol	descripcion	session_id
9	Alejandro	alejandro@hotmail.com	jfff	Socio		

Figure 4.2: Profile created stored in the database

4.2 Access and edit your Personal Profile

Entering to your profile allows you to control your information stored in the database and also change it. There are two views that allow the user to do all this: `verPerfil.pug` and `editarPerfil.pug`. The first one requests information from the database and show it in the view, the second one offers the user the possibility to modify the fields in the “perfiles” table by clicking the button “Confirmar cambios”.

Here we can see how the `verPerfil.pug` file works first and then the `editarPerfil.pug`:

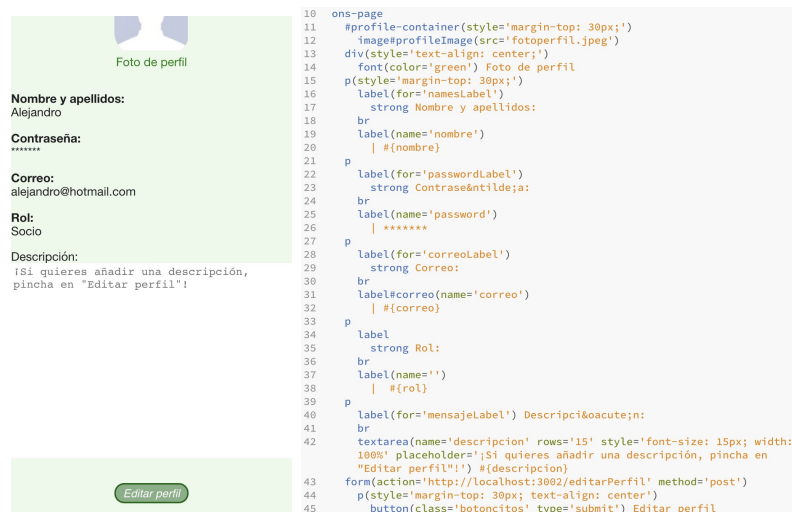


Figure 4.3: `verPerfil.pug` code and view

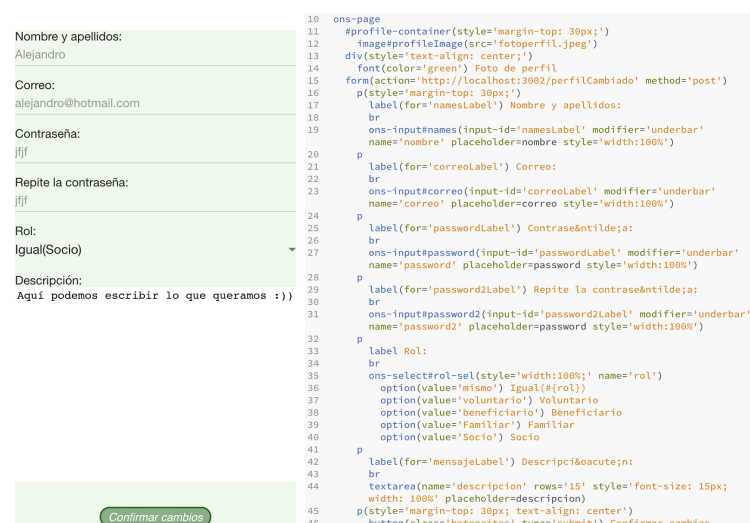


Figure 4.4: `editarPerfil.pug` code and view

4.3 Log in to the app using your email and your password

The user inserts his email and his password in a form inside the .pug file and we check if this data matches and exists in the database. This way, the user can access to our application and see the customized information. If the information entered inside the form does not match with the stored data, the user will be redirected to the main screen.

The files that make this use case possible, have been shown in the other chapters so in this case, the following figures will only show the views of a login attempt. In the first one, we can see the data inserted in the form, and the second picture shows the main view of the application after the user clicks the “Entrar” button.



Figure 4.5: Login views of the application

4.4 Watch a list of activities and search for specific ones

At first, all the activities are shown in the same list but the user can choose between different groups of them. The activities are stored in the database in a table called “actividades”, which saves the date, the name, the place and a description of each activity. Only the name, place and date are shown in the list, however the description is reserved for the next use case. Here we can see how the list of activities is shown without any modification.

TODAS	PROX	Ene	Feb	Mar	Abr	TODAS	PROX	Ene	Feb	Mar	Abr
May	Jun	Jul	Ago	Sep	Oct	May	Jun	Jul	Ago	Sep	Oct
1-12-2018						2-2-2019					
Teatro inclusivo						Cuanto arte tienes!					
Sede						Sede					
Ver actividad						Ver actividad					
2-12-2018						23-2-2019					
Teatro inclusivo						Fin de Semana					
Sede						Soto de Henares					
Ver actividad						Ver actividad					
9-12-2018											
Taller de musicoterapia											

Figure 4.6: List of activities

The activities shown in the figure have been grouped by month. In the first picture, we can see the activities done in December and in the second one, the activities that will be done in February. If we click the button “Ver actividad” we can enter to the complete description of the activity, which is the next use case.

4.5 Access to a description of the activities

The user can access to the page of a particular activity to obtain more information. An entire view is reserved to show the details of the activity and depending on the date of the activity, the information shown changes. This information is loaded from the database through a query inside of a route and it is shown to the user with a dynamic view.

Here we can see the two different possible views. The first image shows the description of an activity when it has already passed: instead of a button to sign up for the activity, there is a phrase that informs the user that the activity is no longer available. The second image gives the user the opportunity of sign up for the activity described by clicking the “Apuntate” button.

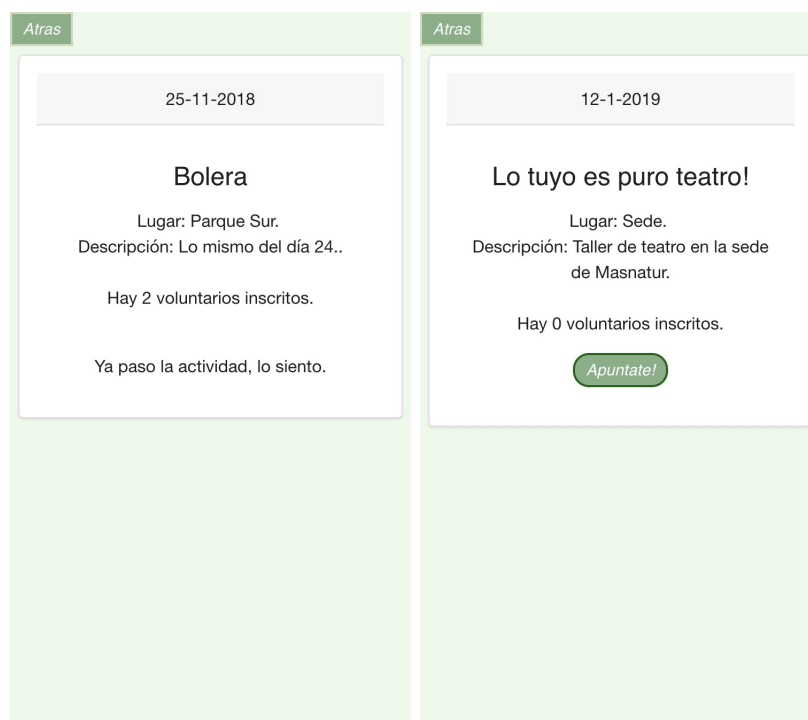


Figure 4.7: Description of the activities

4.6 Sign up for an activity

A user interested in participating in an activity can sign up through the page of that activity (previous use case). After signing up, this user is stored in the database in a table called “volunACT” that includes several fields: id of the activity, id of the volunteer, a boolean that informs if the volunteer has confirmed his presence, an evaluation... Hence we can, for example, obtain a list of the volunteers participating in a specific activity or calculate the number of people available for that activity.

In the next figure we will demonstrate how this works by showing the state of the activity before and after signing up. The image of the left shows an activity without volunteers. The description informs the user about this and offers the possibility of join the activity. If the user clicks the button, the description of that activity changes: there is no button available for this particular user and the count of volunteers has increased for every user as we can see in the image of the right.

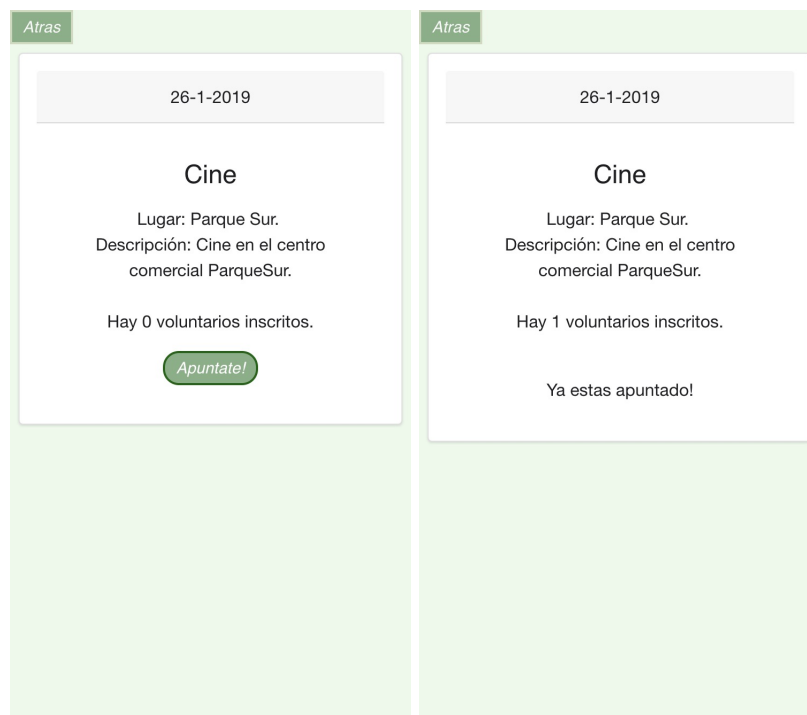


Figure 4.8: Signing up for an activity

4.7 Choose between the future or the past activities

One of the options offered by the app is to group the activities depending on their date. The user can choose to see the activities that have already happened, although he will not be able to sign up for them. But the user can also see the upcoming activities and sign up as we have explained in detail in the previous use case. The .pug view that shows these activities is the same in both cases, the only difference is that the code extracts the date of the activity, compares it with the current date, and shows different data according to a variable that indicates whether the activity have happened or not.

This code is also used for grouping activities as we can select just the future activities or search for every single activity of the database. At the top of the view, we can see the different buttons that group the activities and also a specific button to group only the future activities. This button works as we have described before and shows this:

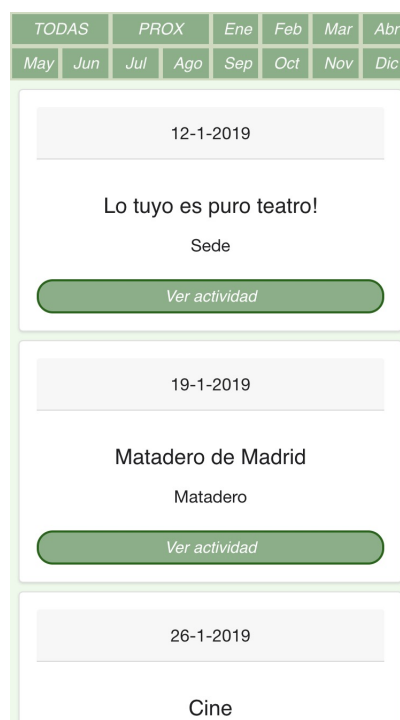


Figure 4.9: Future activities

4.8 Watch your activities

On the main page of the application, the user can choose to access only to his own activities instead of the list of all activities, which means that he can access the activities in which he had participated or in which he has registered and will participate. In this case, the code examines the information stored in the “volunACT” table of the database and searches only those activities that include the user id.

We can see the case of the user created for these examples: “Alejandro” with email “alejandro@hotmail.com”. For this use case, we will also sign this user up for the last activity in February: the weekend in Soto de Henares. The next figure shows in the first image that the user has not participated in any activity yet but he has signed up to two future activities as we can see in the second image. We can see in the figure the result of clicking the “past” and “future” personal activities:

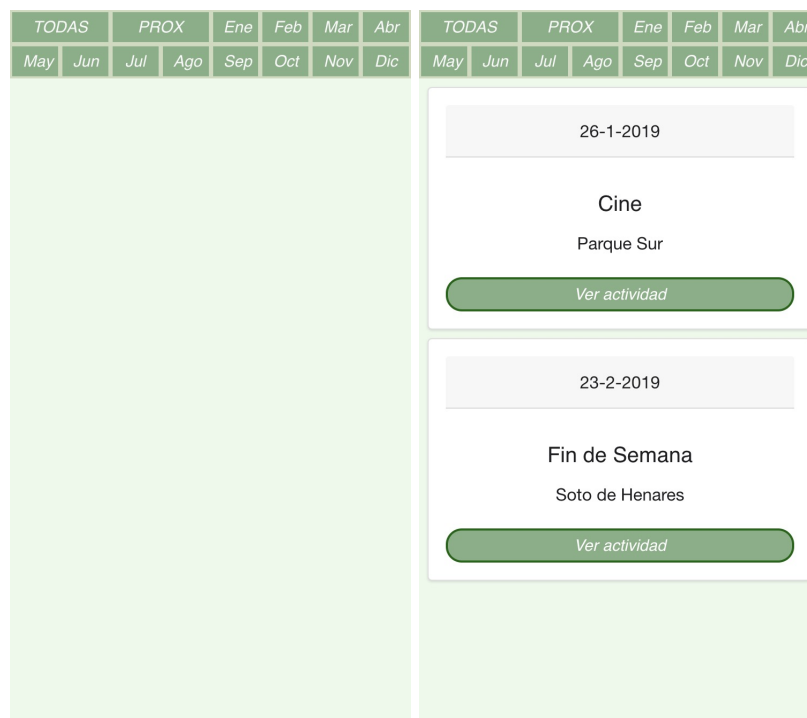


Figure 4.10: My personal activities

4.9 Send an email through the app

The user can send an email and contact with the workers of the Foundation using the application. Currently the only way of communication between volunteers and Foundation is by email or phone, and for this reason it is important that the application offers this possibility. This email can be send thanks to the smtp transport included in the NodeMailer module. As we have explained before, this module is used to send messages.

The email service used is Gmail because the Foundation uses this same service. For this tests, we have also used the XOauth2 protocol which requires an authenticated google member. This data has been placed inside the route that sends the email, as a property of the smtp transport. The received email has always the same structure in order to find it quickly and easily. In the following figure, we can see the email send through the application and recieved in my personal email.

The figure shows a web form for sending an email and a screenshot of the received email in a client.

Form: Envíanos tu correo

Name *
Prueba

Email *
correoprueba@hotmail.com

Message *
Esto es solo una prueba para comprobar que funciona!

Submit

Correo de la Aplicación ➤ Recibidos ✕

Prueba
para [www.pepeunpurito](#)

ATENCION ANTENCION: Prueba (correoprueba@hotmail.com) says: Esto es solo una prueba para comprobar que funciona!

Figure 4.11: Sending an email

4.10 Access to all of the social networks of the Foundation

The application offers different links and information about the Foundation. Through the login page we can access to information focused on attracting new users to the Foundation. There are four Masnatur goals described as we can see in the next figure. The user can change between the goals very easily. The first two images show an example of this:



Figure 4.12: Goals and links of the Foundation

Throughout the application we can find some links that redirect users to the Facebook, Twitter, YouTube or Instagram of Masnatur. This are also very accessible because they all appear after clicking a button at the bottom of the mobile phone. In the third picture we can see the aspect of the view after clicking the button.

The tools used above are edited components from Onsen UI. In the second picture we can see the “Vision” view selected with the “Side Menu” component and also how the “Speed Dial” works. The last picture is a normal Onsen UI “Action Sheet” that redirects the user to the different social networks.

Conclusions

In this chapter we describe the conclusions extracted from this project, problems, achievements and suggestions about future work.

5.1 Conclusions

In this project we have developed a multiplatform mobile application for a non-profit Foundation called Masnatur. We have selected the framework Apache Cordova in order to reduce developing time and offer as many different mobile operating systems as possible.

This project contains some basic features that allows us to release the application as it is, such as a full login for any volunteer which was a main priority. In summary, we have used the top tools to develop the application in the most effective way. There may be better solutions, but this one is very optimized. Onsen UI components are a great complement for Cordova applications, and NodeJS works perfect with MySQL databases.

In the following sections we explain the problems faced and some future objectives.

5.2 Problems faced

While installing the environment we faced several problems mainly with the versions of the different elements (Android Studio¹, JDK², Gradle³, the environment variables, etc). Every problem was solved by googling the solution in forums or by searching the FAQs of each element and installing the correct version.

Connecting the database with the code also generated problems until the routes started to work properly. The problem was introducing correctly SQL sentences and it was solved after installing the program called PopSQL. With this program, we were able to try different sentences and find the correct one.

At the beginning of the project we were working with static views until we faced lot of problems when trying to request the information stored in the database and show it in the view. This was a really difficult problem to solve. After changing the .html views too .pug, everything started to work correctly. This solution was suggested by the tutor Carlos Ángel Iglesias.

¹<https://developer.android.com/studio/?hl=es-419>

²<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

³<https://gradle.org>

5.3 Future Objectives

As we have commented before, this is just a prototype and we will update and improve the application as much as we can. We have lots of future objectives if we want to create a complete application. Here we can see some of them:

- Add new features to the actual application such as uploading a photo to the user profile, sending photographs attached to the emails, adding a customized style to the whole application, etc. In summary, improve the current application.
- Publish the mobile application in each platform market (Google Play Store and Apple App Store at least). At the beginning the apk will be distributed by email to a few members to test the application. After proving it works, we will extend it to the different markets.
- A complete login for every possible role in the Foundation. This way, the application will be useful to every single member of the Foundation. To gain this objective, we will need to extend the tables of the database, create different views, and increase security. This objective must be done carefully as we will work with delicate data.
- Combine all the databases in the same one. We will need to learn in depth the database used at the Foundation and combine it with ours. Depending on the complexity of this database, this will be a reachable objective or not.

Appendix

In this appendix we will discuss the global impact and responsibilities related to the project. We have summarized this impacts throughout the project but in the following page we will go into detail. The objective of this appendix is to explain the possibles environments and try to guess the viability of the application nowadays.

A.1 Global impacts and responsibilities

The application we have described during the project tries to offer something different in one particular stage: the third sector. This sector has improved a lot in the last decades but they are still using some ancient methods. And there is where this application appears. Mobile applications for third sector members is an empty market that will be developed sooner or later. Now we are going to explain the different impacts and responsibilities:

A.2 Social Impact

As we have explained before, this project is focused on helping the third-sector members and promote a more fluid communication between the parts involved. This application has been designed for disabled people with little technological knowledge and in some cases, also little mobility. That is why our created interface tries to be as friendly and easy as possible. Of course, the data that the application handles is a very delicate one, and needs to be properly protected. This security will be in constant development.

A.3 Economical Impact

This application will be donated to the Foundation Masnatur to prove it works properly. After testing and improving the application, the base model will be sold to other members of the third sector in order to update and modernize the sector. The maintenance of the project will be responsibility of the project author during the first versions. That way, this project pretends to be a 100% viable.

A.4 Responsibilities

As we have mentioned before, this is a delicate point as we are working with people that are very protected by law. In fact, the first versions will only be available for the volunteers of the Foundation. Once this application gets bigger and grows, we will have to manage carefully the future changes and updates together with the lawyers and heads of the Foundation. At least, for now, the application works properly and offers a responsible commitment.

Appendix

In this second appendix we will discuss the budget needed to bring about the project. There are some costs derived from the typical salaries of developers, and other costs have been chosen from the minimum hardware needs.

B.1 Context

In this particular case, the first application will be a donation for the Foundation Masnatur, so the costs for them will not be the same as the ones explained and detailed here. Even though they get the first version free, they will need to undertake some other costs in order to obtain the required licenses and ensure a maintenance of the application.

B.2 Physical resources

The application created in this project has some basic needs. It is developed to run in many devices but not all. According to the amount of data managed, the average of connections required and the future updates planned, we have selected this minimum requirements:

- **RAM:** over 2GB
- **ROM:** over 8GB
- **Internet connection:** Wifi and 3G.

B.3 Human resources

As we have commented in the context, this pricing is different for the first Foundation as they will not need to afford an initial price. Even though, every Foundation will require an individual maintenance for the future years. We estimate this costs in the following way:

- **Initial price:** Considering 4 hours of work per day for the past 4 months including the last 2 weekends, with a salary of 10 €/per hour, makes a total of **3.360 €** which means 840 €/per month.
- **Maintenance price:** We keep the same terms to calculate the maintenance price. In this case, we estimate working 2 weekends every month on updates, which gives us a **320 €** salary per month.

B.4 Licenses

We have been careful to select only open sourced softwares during the project. So in this case, there are no costs in Licenses although in the future might be some depending on how much the application grows. One of the posible costs are purchasing the premium version of the program PopSQL which increments the salary only **7€** per month.

Bibliography

- [1] Apache Cordova website. *Cordova Information*. Available at <https://cordova.apache.org/docs/en/latest/>.
- [2] Wikipedia. *Cordova Information*. Available at https://en.wikipedia.org/wiki/Apache_Cordova.
- [3] TutorialesProgramacionYa website. *Tutorial*. Available at <https://www.tutorialesprogramacionya.com/javascriptya/nodejsya/detalleconcepto.php?punto=1&codigo=1&inicio=0>.
- [4] OnsenUI website. *Onsen UI Information*. Available at <https://onsen.io/v2/guide/>.
- [5] MySQL official website. *MySQL guide*. Available at <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.
- [6] Alejandro Garrido Chasco. *Mock-Up Prototype*. Available at <https://www.figma.com/proto/DcjAPZdtyvjxZZSPrxo7yM9H/Untitled?node-id=0%3A1&scaling=scale-down>.
- [7] Alejandro Garrido Chasco. *Mock-Up General View*. Available at <https://www.figma.com/file/DcjAPZdtyvjxZZSPrxo7yM9H/Untitled?node-id=0%3A1>.
- [8] PopSQL. *PopSQL page*. Available at <https://popsql.io/>.
- [9] Alejandro Garrido Chasco. Design and development of a multiplatform mobile application for a non-profit foundation based on the framework cordova. Master thesis, Escuela Técnica Superior de Ingenieros de Telecomunicación, ETSI Telecomunicación, Madrid, January 2019.