UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A MOBILE APPLICATION BASED ON CORDOVA FRAMEWORK AND WEB TECHNOLOGIES FOR A UNIVERSITY INTRANET

GUILLERMO GARCÍA MENÉNDEZ

2017

TRABAJO FIN DE GRADO

Título:	Diseño e implementación de una aplicación móvil basada en tecnologías web y el framework Cordova para la intranet de una Universidad.
Título (inglés):	Design and Implementation of a mobile application based on Cordova framework and web technologies for a University intranet.
Autor:	Guillermo García Menéndez
Tutor:	Carlos A. Iglesias Fernández
Departamento:	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 FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A MOBILE APPLICATION BASED ON CORDOVA FRAMEWORK AND WEB TECHNOLOGIES FOR A UNIVERSITY INTRANET

Guillermo García Menéndez

Junio de 2017

Resumen

La programación híbrida es una forma de programación móvil basada en tecnologías web en vez de en las interfaces nativas propuestas.

El desarrollo de una aplicación móvil basada en estas tecnologías híbridas evita desarrollos en paralelo para cada sistema operativo móvil, como Android o iOS. Además, la utilización del desarrollo móvil híbrido se completa usando el patrón MVC (Modelo - Vista - Controlador), guiándonos hacia una implementación más escalable y modular del desarrollo de una aplicación móvil.

Este Trabajo Fin de Grado busca desarrollar una aplicación móvil para la intranet de la Universidad Politécnica de Madrid, centrándose en los empleados de la misma. Su uso reducirá algunos aspectos de un desarrollo ordinario para plataformas nativas como:

- Tiempo empleado en desarrollar diferentes códigos fuente para diferentes plataformas.
- Tiempo de mantenimiento del futuro proyecto finalizado, así como de corrección de bugs y futuras ampliaciones de plataformas.
- Esfuerzo necesario para desarrollar una aplicación enfocada a múltiples entornos.

Las principales tareas del proyecto serán: (i) Analizar los requerimientos, (ii) Diseñar la aplicación, (iii) integrar la aplicación con los servicios otorgados por la intranet, (iv) evaluar la aplicación y (v) presentar la propuesta.

Palabras clave: Tecnologías Web, Universidad, Cordova, Híbrido, Móvil, Aplicación, Android, iOS

Abstract

Hybrid mobile development is a way of programming mobile applications based on web technologies instead of native interfaces.

The development of mobile applications based on hybrid technologies also avoids parallel development of mobile applications in the different mobile operating systems, such as Android or iOS. Furthermore, the use of hybrid mobile development has been completed with a MVC (Model - View - Controller) pattern that will lead us to implement a more extensible and modulable application.

This final project aims at developing a mobile application for the UPM intranet, targeting employees of the university. This will be used to reduce some aspects of an ordinary native mobile project like:

- Time cost of developing different sources for different platforms.
- Maintenance time for bugs and new platform upgrades.
- Efforts of deploying apps in multiple environments.

The main tasks of the project will be: (i) analysing the requirements, (ii) design the application, (iii) integrate the application with the services provided by the intranet, (iv) evaluate the application, and (v) write the proposal.

Keywords: Web Technologies, University, Cordova, Hybrid, Mobile, Application, Android, iOS

Agradecimientos

Esta viaje es un largo recorrido que, como poco, es mejor emprender acompañado. Muchísimas gracias a todos aquellos que nunca les costó dar una palmadita en la espalda, aunque su situación tampoco fuese una verbena.

En especial me gustaría agradecer a mi madre y a mi padre, por apoyarme en todo lo que hago y por devolverme al mundo real en cuando la ocasión lo precisaba, aunque siempre me dejen volar a junto a las nubes. A Nuria, por acompañarme durante todo el camino, y más, y por no dejarme caer cuando necesitaba un amarre. A Adrián y a Borja, los grandes culpables de que empezase el viaje; y en especial a Borja, de cuya curiosidad y méritos siempre me hizo partícipe.

A todos mis amigos por dejarme su hombro, ya sea para celebrar o para consolar, a pesar incluso de aquellos casos en los que la distancia parecía inabarcable.

Para acabar, dar las gracias a todos los que formáis clubes, habéis conseguido que crezca como pesona y abra la mente.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

Mobile application development has suffered a great evolution since its beginning. The mobile world started as a simple communication and productivity system but now it has become into a huge revolution that wants to simplify each life task, from TO-DO lists to high level photography editors. Now, "Mobile World" does not only mean "communication", it also means "entertainment", "hobby", "socializing", and so on. Consequently, during that expansion, relevancy was obtained by mobile platforms, which was followed closely by big tech companies like Apple, Blackberry, Google or Microsoft.

The mobile industry is booming like never before, it is an exciting environment, but not without its issues. The industry wants to be able to build applications more efficient than what now is the standard. But at the same time you want the application to have the same feel as any other application on the market. The issue arises when you start to develop and realize that you have to set up several different environments and learn at least two different programming languages, Java and Objective-C/Swift[2].

In spite of the number of mobile operative systems that coexist in the market, only two of them excel: Android and iOS. Taking into account the global market share for smartphone

OS, we can see Android being the main leader with an 86.8% of the market. In second place, iOS is keeping a great quota of 12.6%. Finally, the remaining percentage is shared among others like Windows Phone, UbuntuOS, Blackberry10, etc[9].

Developers are leaded by the previous scenario and they want to build its applications where exists the biggest bunch of users. Furthermore, not only developers but also companies have that target. As a result of this tendency, targeting only two platforms will give a 99.4%, which is more than an acceptable result.

Nevertheless, targeting two platforms means developing an application in two, or more, distinct programming languages. Thus, hybrid programming is in charge of fulfilling this issue. Hybrid programming is a concept born from developers' necessity of developing applications targeting multiple platforms by optimizing developing time costs. Identified that necessity, many solutions has been appeared to solve it, for instance, Xamarin, React Native, NativeScript or Apache Cordova. From all those solutions, Apache Cordova is the only one which is implemented using pure Web Technologies.

To conclude, in order to achieve the objectives of this project, we will use Apache Cordova Framework due to its continuous evolution since its beginning and the community support received, which accredit Cordova as a stable solution to use in this project's development.

1.2 Project goals

In the long term, this project aims at becoming a substitute for the daily actions to University staff. With this aim, the project will study which are the most usual activity on current web platforms. In addition, this application must be able to connect with the remote server, where the information is provided.

Among the main goals inside this project, we can find:

- Design and implement a cross-platform mobile application for consulting personal information from a University intranet.
- Develop local logic that handles dynamic content depending on what kind of user is accessing to the application.
- Implement a *Push Notification System* for alerting users of specific events.
- Publish the mobile application in each platform market: (*Google Play Store and Apple App Store*).

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is the following:
Chapter 1 explains the context in which this project is developed. Moreover, it describes the main goals to achieve in this project.
Chapter 2 provides a description of the main technologies on which this project relies.
Chapter 3 describes the architecture of this project, describing a global vision and dividing it in modules.
Chapter 4 presents the whole final application and its use cases.
Chapter 5 discusses the conclusions drawn from this project, problems faced and suggestions for a future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter, we are going to give an insight into the technologies used in this project. First of all, we are going to explain Cordova Framework, which is the core of this project. Second, we are going to present the rest of technologies ranking them for importance. Finally, the technology that enables us to obtain data from server is going to be presented.

2.2 Cordova Framework

Apache Cordova [11][1] is an open-source mobile development framework. It allows you to use standard web technologies - HTML5, CSS3, and JavaScript for cross-platform development. Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's capabilities such as sensors, data, network status, etc [4].

Cordova simplifies the development of mobile applications by getting us away from the

[1]https://cordova.apache.org/

complexity of learning each programming language of each platform where we want to publish.

A Cordova project is arranged by following a folder structure showed below:

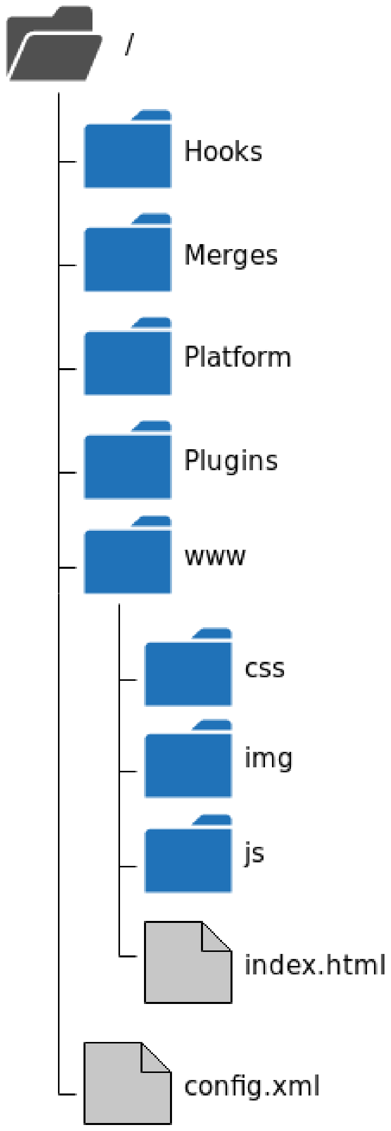


Figure 2.1: Apache Cordova folder structure

Where each folder and file has its own purpose in the project, namely:

- **Hooks:** a *hook* is a script that will be run at a specified time. Despite its name could be overwritten, inside this folder should go every piece of code that performs an action related to the project. Specific times could be *before_prepare, after_compile, after_build*, etc.

- **Merges:** as Cordova is oriented to develop cross-platform projects, maybe there are code or views that should only exist in a specific platform. This folder's goal is to create or overwrite any file inside it at *prepare* time depending on the platform in which the action is performed. Folder structure should be: `/merges/platform-name/fileToBeOverwritten.js`.

- **Platform:** folder to store native files generated, for instance, in a build. It also contains the results of cordova commands like `$cordova build platform`

- **Plugins:** this folder has a similar intention that the previous one but it stores plugin code and dependencies.

- **www:** a Cordova applications is based on a Web App embedded in a native WebView. This folder contains the Web App that will be presented as mobile application.

- **config.xml:** The configuration file of Cordova. Dependencies, global and platform preferences are set by this file. In addition, icons are also set by it.

2.2.1 Plugins

Apache Cordova comes with a wide range of plugins which allow us to control native features from any platform. These plugins enable the use of device capabilities and native APIs such as vibration, status-bar control, or geolocation services. In addition, they also are a bridge from native code to JavaScript, creating an interface for specific functionalities like push notification native services or Google Maps native APIs.

Now, we are going to show the plugins we have used for this project, including a brief explanation for each one.

- **cordova-plugin-console:** as the application is been developed for many platforms at once, it will need basic functionalities like `console.log()` available in iOS. This plugin enables that capabilities.

- **cordova-plugin-device:** this plugin allows us to get unique information about the device where our application is running like the platform or the OS version.

- **cordova-plugin-file:** This plugin implements a File API allowing read/write access to files residing on the device.

- **cordova-plugin-compat:** a plugin which maintains compatibility with older versions of Cordova Framework.

- **cordova-plugin-whitelist:** implements a whitelist policy to control which URLs the WebView itself can be navigated to.

- **cordova-plugin-splashscreen:** due to iOS policies, it is required to show a splash-screen during application's boot.

- **cordova-plugin-statusbar:** this application is based on *Material Design Guidelines*, and to do so, we need to darken the statusbar only on Android platform.

- **cordova-plugin-googlemaps:** enables a native view of googlemaps for better performance.

- **phonegap-plugin-push:** enables the application to listen push services from Google Firebase Service and Apple Push Notification Services.

2.3 jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Furthermore, it is an extensible library. This means that the application can base itself over jQuery to continue adding code which will improve jQuery according to its necessities.

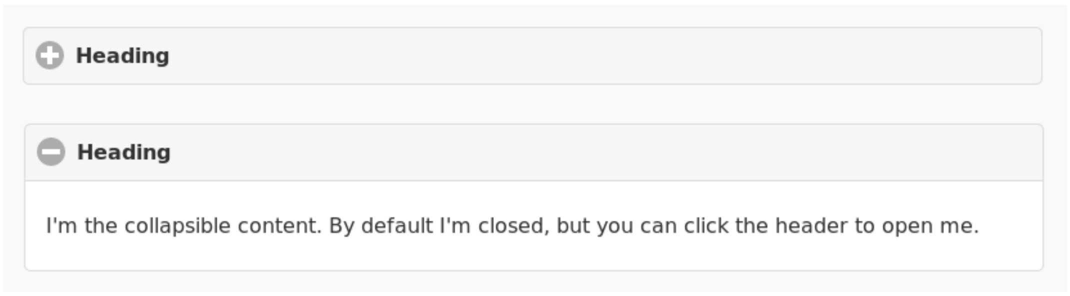
The following sections are going to be an introduction of the frameworks that we use to extend jQuery in our application.

2.3.1 jQuery Mobile

The main purpose of using jQuery Mobile is to have pagination implemented in the project. Our goal using this extension is to automatize the DOM actions for inserting/deleting views of the application. We call this views ”pages” as jQuery Mobile expect.

Using pages, jQuery Mobile triggers custom events for page objects which allow us to handle the page flow as if it are almost native page handling.

In addition, we also use jQuery Mobile to layout the widgets and views of the application. jQuery Mobile offers a wide range of widgets and handlers which simplifies laying views.



I'm the collapsible content. By default I'm closed, but you can click the header to open me.

Figure 2.2: jQuery Mobile collapsible widget

2.3.2 jquery.localize.js

Despite of a first look at the application, where all strings are showed in Spanish, it is prepared at code level to be presented in many languages, for example English or French. This feature is available thanks to a jQuery extension called jquery.localize.js.

The jQuery extension is a open-source project hosted in GitHub [5] that enables to change the inner-text of a DOM element from its html default code to a value previously stored in a json file. We initialize the extension with the following lines in each JavaScript file we use as view controller:

Listing 2.1: jquery.localize.js use

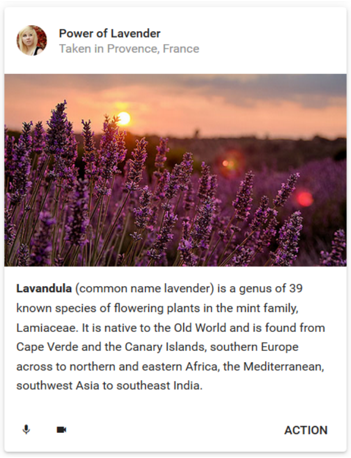
```
$("[data-localize]").localize("header")
      .localize("sidebar")
      .localize("footer");
```

On the basis of previous lines, we view how does this plugin works. It searches all DOM elements which have an attribute called `data-localize` and apply them the main function, `.localize()`, of the plugin. This function is passed a parameter that will be the name of the value we want to load into the DOM element from the json file. The json structure has been selected in relation with page ids used in jQuery Mobile page structure for readability.

2.3.3 NativeDroid2

NativeDroid2 is a theme for jQuery Mobile inspired by Material Design guidelines from Google. The project is distributed under MIT-License and can be installed from `bower` or from sources[2].

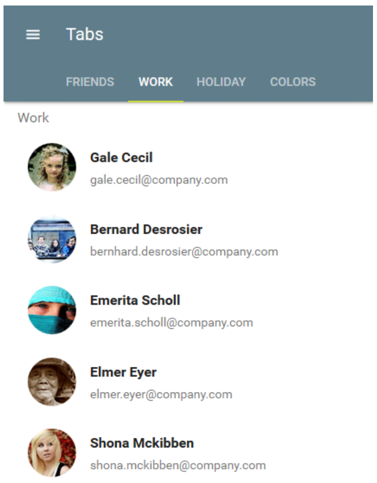
This project also extends jQuery Mobile with some visual widgets like *tabs, cards, bottom sheets, search,* and *toasts*, enabling them with the aspect they are supposed to have according to Material Design guidelines.



Power of Lavender
Taken in Provence, France

Lavandula (common name lavender) is a genus of 39 known species of flowering plants in the mint family, Lamiaceae. It is native to the Old World and is found from Cape Verde and the Canary Islands, southern Europe across to northern and eastern Africa, the Mediterranean, southwest Asia to southeast India.

Figure 2.3: Card widget



Work

Gale Cecil
gale.cecil@company.com

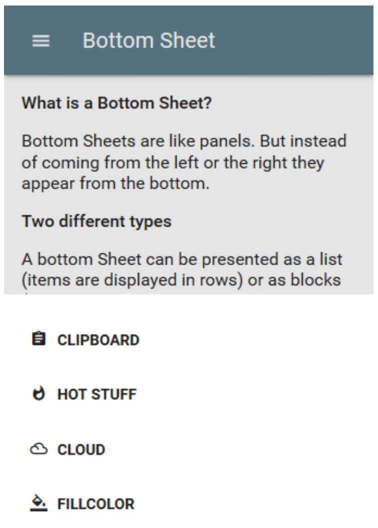
Bernard Desrosier
bernhard.desrosier@company.com

Emerita Scholl
emerita.scholl@company.com

Elmer Eyer
elmer.eyer@company.com

Shona Mckibben
shona.mckibben@company.com

Figure 2.4: Tabs widget



What is a Bottom Sheet?

Bottom Sheets are like panels. But instead of coming from the left or the right they appear from the bottom.

Two different types

A bottom Sheet can be presented as a list (items are displayed in rows) or as blocks

Figure 2.5: Bottom Sheet widget

2.4 Material Design Iconic Font

Material Design Iconic Font is a full suite of official material design icons (created and maintained by Google), with additional community-designed icons and brands icons for easy scalable vector graphics on websites or desktop.

This font will allow the project's user interface to use any Material Design icon it needs. Appending an icon to a HTML page is as easy as adding the following tag into the desired element:

Listing 2.2: example of icon html code

```
<i class=                           ></i>
```

This icon suite is used as dependency of *NativeDroid2* for illustrating warnings or actions in buttons, a bunch of examples are showed in Fig. 2.6.

[2]https://github.com/wildhaber/nativeDroid2

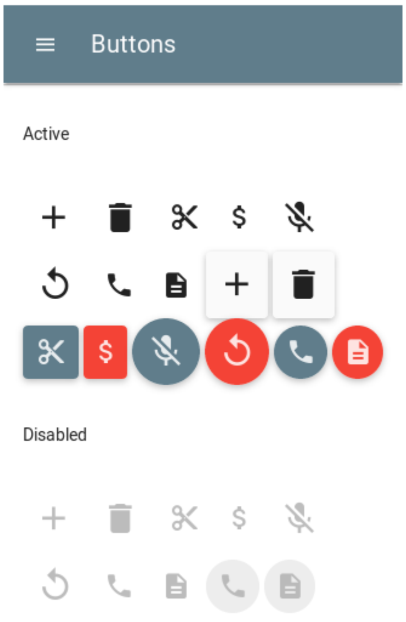


Figure 2.6: Icons in buttons

2.5 Sha256.js

Cryptographic operations are a fundamental pillar of every application which want to store or transport any kind of sensible data. This library is a implementation in JavaScript of the SHA-256 algorithm by Chris Veness published under MIT License. Also, this implementation is available on GitHub [10] for anyone who want to take a look.

In this case, the library is used to provide SHA-256 hash operation, as we need to ensure that any kind of hash function is operative on all platforms. An example of using this library would be:

Listing 2.3: example of Sha256.js use

```
var hashedText = Sha256.hash(text);
```

2.6 Sass

Sass is a CSS preprocessor for writing scripts that will be compiled into css files. It allows developers to use syntaxes from programming languages like variables, functions, nesting, control directives, etc.

CSS preprocessor languages were introduced by the industry as a response to the missing features of CSS. The code written in a CSS preprocessor can include variable and function declarations, which can be used inside CSS selectors. The preprocessor compiler essentially transforms (i.e., transpiles) the function calls and variable uses to pure CSS [7].

To transform `.sass` or `.scss` files into `.css` ones a console command is needed.

2.7 gulp.js

Gulp makes builds and workflows easier to get done. While this project is based on design and implementation of a mobile application, it also faces the last step of every developed application: the distribution. As this project is being built according to university policies, it will need to follow some steps before its distribution, namely: (i) Set application environment to production, (ii) disable log traces and modeDummy, (iii) minify code, (iv) obfuscate code and (v) change and remove readable sources.

Thanks to Gulp, tasks can be created to make the hard work by just double-clicking. Gulp syntaxes looks like:

Listing 2.4: Custom gulp task

```
gulp.task('delete_js_files', ['ofuscar'], function () {
  return gulp.src(all_js_files.slice(0, all_js_files.length-1)).pipe(
      clean());
});
```

CHAPTER 3

Architecture

3.1 Introduction

In this chapter, we are going to explain the architecture of the entire project, from the design phase and implementation details to server requirements. First of all, in the overview we will present a global vision of the project architecture, identifying client and server side. Secondly, we are going to glance at Apache Cordova architecture, focusing on each module and its relation with the project. Finally, the relationship between them will be outlined.

3.2 Overview

In this section, the global architecture of the project is going to be presented, defining the different supermodules, which will contain the main modules that participates in it.

- **DB connection interface:** all the data needed by the mobile application is stored in databases. This interface minimizes failure in case of database changes.
- **Wapi_upm:** this supermodule is a *Web API* for serving the information to the mobile application.

- **Cordova application:** the main supermodule of the project. This will contain the logic and the views of the application.

- **Mobile OS:** the platform where the application will be running.

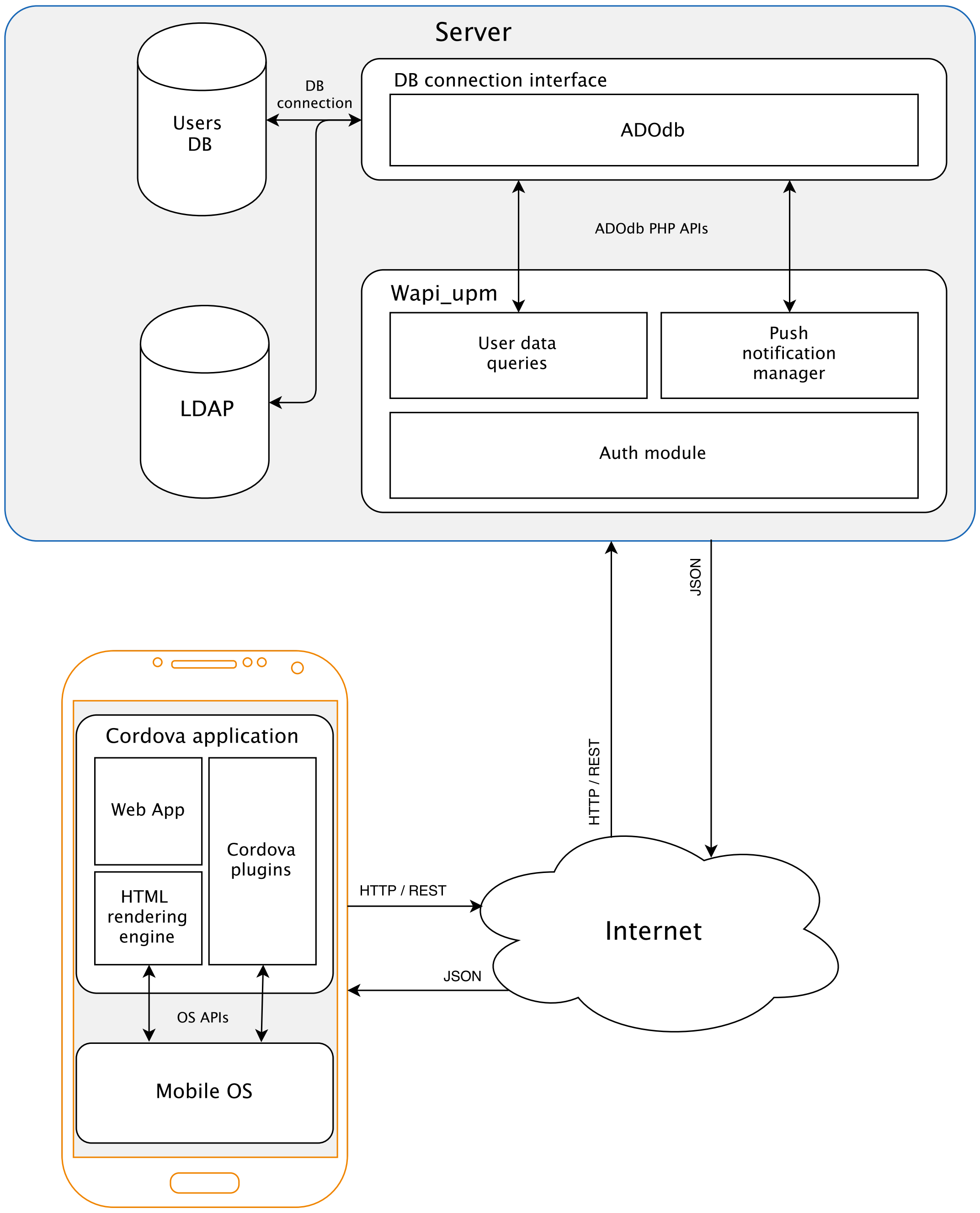


Figure 3.1: Architecture

In the following sections we are going to deeply describe the subsystems involved in the project.

3.3 Server modules

The server is responsible for organizing and sending all the information needed by the mobile application. Its principal purpose is to act as information gatherer and access controller. This means that the server is going to listen to the application's requests addressed to it, and to retrieve the information which it will send in return from different sources, e.g., databases or files.

Furthermore, the server is going to take the responsibility of granting or denying access to the data that is being required by the application. This role will be explained in detail in section 3.3.2.1.

3.3.1 DB connection interface

A database connection interface is a tool for avoiding the complexity of manage connections and links to a database. It acts as a *data access object* design pattern which will work transparently for us so that we only have to worry about queries and the use of gathered data.

ADOdb is a database abstraction layer for PHP. It is not only a database connection interface but an interface of interfaces as well. This means that the server logic will be even abstracted from DBMSs (*DataBase Management System*). The purpose of using an interface to work with databases is mainly to avoid extra maintenance in case of database migrations or database driver changes.

This layer allows developers to use the same code in order to access a wide range of databases. ADOdb contains components for querying and updating databases, as well as an Object Orientated Active Record library, schema management and performance monitoring [6].

Despite it is based on the ADOdb abstraction layer, the purpose of the DB connection interface is not only to obtain database records but also to do log tasks of CRUD (*Create, Read, Update & Delete*) actions and error handling.

3.3.2 Wapi_UPM

This module is a web API whose main purpose is to offer data from the university. The election of these data, despite being requested by the application, is subject to the university's approbation. Furthermore, using this approach defines the project's architecture as

Service-Oriented.

The Web Services used by the application have their origin in this module, which will publish them employing RESTler. RESTler is a crawler that uses *Resource Linking Language* (ReLL) descriptions instead of *Web Application Description Language* (WADL) as instructions for traversing a RESTful service and produces a typed graph of the crawled resources and the links connecting them [1].

The module Wapi_UPM has three submodules as shown in Fig. 3.1 which are detailed below:

3.3.2.1 Authentication module

The authentication module will be responsible for manage access to groups of Web Services depending on several parameters. This authentication is implemented under a RESTler interface called `iAuthenticate` which will require filling two methods:

- `__isAllowed()`: contains all the logic needed for authenticating a request. Should return a boolean. `true` if the request is allowed, `false` else.
- `__getWWWAuthenticateString()`: for offering some information about what the server expects to try an authentication. Should return a string.

In relation to project's https://drive.google.com/drive/my-drivearchitecture, this module is going to allow access to a requested Web Service depending on its headers. In this project, when opening a *"session"*, the application needs to do a POST request to the University servers with the following payload:

Listing 3.1: Payload of login request

```
{
  "cuenta": "sonia.fraguasm@upm.es",
  "password": "p4ssW0rD",
  "app_id": "..."
}
```

When the request is tested by the authentication module, it considers whether the data received is valid or not. If everything is alright, the server will send a response `200 OK` in json format with the following content:

Listing 3.2: Response of login request

```
{
  "token": "...",
  "nombre": "Sonia",
  "apellidos": "Fraguas Montalvo",
  "colectivo": "F"
}
```

Supposing a successful login request, the application will obtain a *“session”* from the server’s response to perform further requests. This *“session”* is defined by a token, which is actually what is obtained from the server’s response, specifically in the `token` field.

The reason of quoting *session* is due to server’s vision of the connection between the mobile application and itself. This connection has been implemented to be session-less so from this moment onwards we will call it *session* despite it is not. Only for simplifying concepts.

The session mentioned above will not be maintained from the very right time it is granted, that we will call $\texttt{t}_{start}$, but it will from a time chosen each ξ minutes, called ξ_1, to the next, ξ_2. This means that users could have sessions whose duration is really short:

$$t_{session} = t_{start} - t_{\xi_1}$$

Where $t_{session}$ is the whole time the server is keeping an eye on a user. Beyond that time, session will be closed and any request from the previous user will be rejected unless it perform a new login request.

The earlier behavior could become annoying to a user because it actually can have a one minute session so a login request will have to be done twice in less than two or three minutes. In order to avoid that kind of problem, the authentication logic was improved as follows: (i) Mobile application sends a request to the server with a header named `X-UPMTOK` which contains the session token obtained in the login, but this time it is expired. (ii) Server checks the token’s validity by comparing it with the current genuine token and the previous one. Just one truth will be enough to granting it access to the Web Services and answering it the corresponding data. (iii) A `X-UPMTOK` header that contains the current genuine token is going to be in the response for communicating it to the mobile application. (iv) Finally, the current genuine token will be stored by the mobile application for further requests., as show in Fig. 3.2.

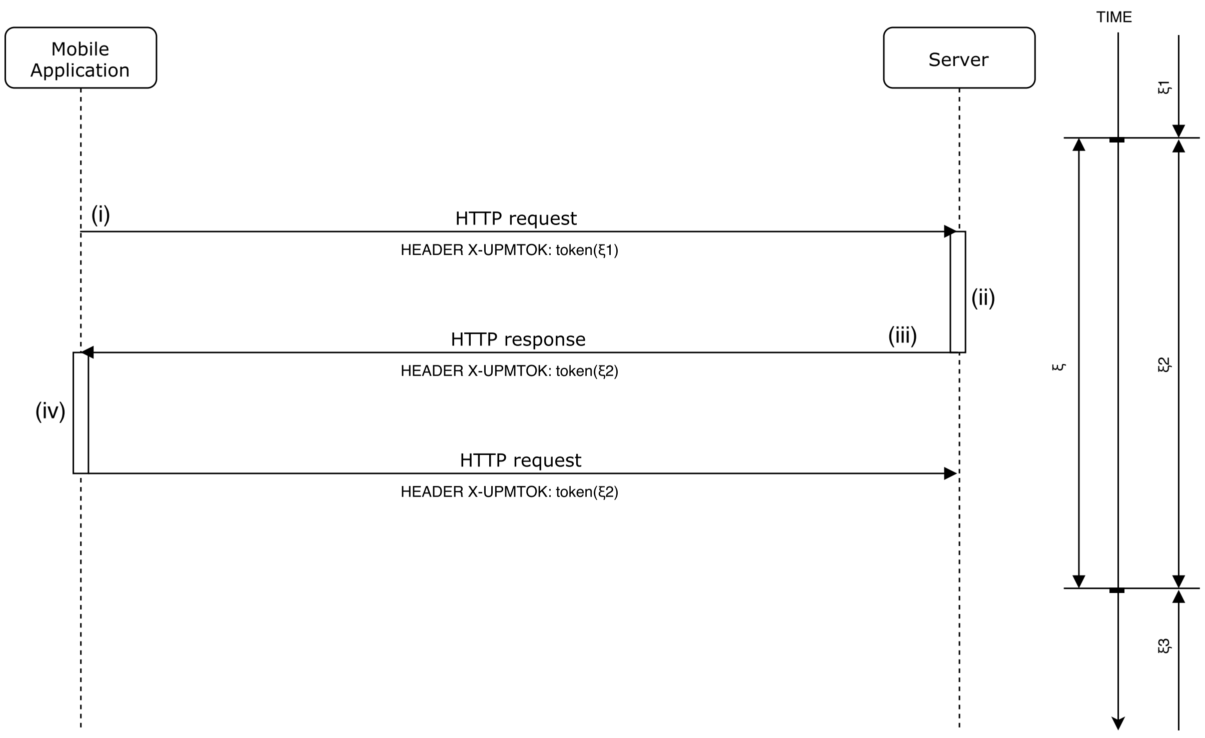


Figure 3.2: Token timeline

Accordingly, sessions will auto-renew themselves instead of being renewed by users each time they expire. This improved behavior will maintain a session up to $2 \times \xi$ minutes if users do not make more requests. However, from user's perspective, its session will last forever while it continues using the mobile application. Finally, the new session's duration would be like:

$$t_{session} = t_{start} - t_{\xi_1}[+(n \times \xi)]$$

3.3.2.2 Push Management module

A push notification is a message sent to an application whose main difference with usual notifications is the fact that it is not requested by the application. The message's delivery is orchestrated by the server itself without any client interaction, so the message is *pushed.*

Since this project goal is to develop an hybrid mobile application, various options would have to be considered for implementing push notifications because project's goal is to run the mobile application on various platforms (Android & iOS). Both platforms count with its own solution to achieve push notifications delivery, namely:

- **Apple Push Notification Service.** Offers the possibility of sending push notfications to the whole family of Apple's devices through `HTTP/2` interface. It has a payload limit size of 4KB with that interface. Previous limit was 2KB for the legacy APNS binary interface[1]. To obtain client-server communication it is required an iOS Certificate for *Apple Push Notification service SSL (Sandbox & Production).* This certificate will identify the mobile application which is going to receive notifications through its bundle ID (see section 3.4.2.1).

- **Firebase Cloud Messaging.** It is a Google service which replaces *Google Cloud Messaging*[2] (GCM) since Firebase platform was released. This solution offers push notification services to not only android devices (smartphones, TVs, wearables, etc) but also to Apple devices. This is possible thanks to an abstraction layer that FCM (and GCM as well) implements by using the certificate explained in the previous point. It has a payload limit size of 4KB through `HTTP` interface.

In the end, and considering server restrictions for implementing both platforms, FCM was the chosen platform to achieve push notification delivery. This decision was also taken by having in mind what data is sent back to the university server by both platforms when it does a push request. While APNS (actually legacy APNS binary interface) gives feedback of a push delivery via *The Feedback Service*, which is another binary communication with Apple's servers, FCM gives the feedback in the answer of a push HTTP request.

The figure 3.3 shows how push management is done. First of all, the mobile application has to specifically request that it is interested in receiving push notifications by registering on FCM. Steps 1 and 1.1 will be explained in section 3.4.2.3. Secondly, after the register is done, the mobile application will receive a token that identifies a mobile plus mobile application bundle. Thirdly, the token will be sent to the University servers to storage it in a proper way. Finally, steps 4, 4.1, and 5 are the result of sending a push notification: FCM is connected by the University servers in order to send a push notification to the mobile application, to do so FCM sends the message to android devices and it delegates the delivery of iOS devices to APNS. All the feedback received by University servers are also properly stored.

[1]During the development of this project, all tests were done with legacy APNS binary interface, due to server's .NET version restrictions.

[2]At the beginning of project's development, GCM was the technology needed to implement push notification over android and iOS devices.

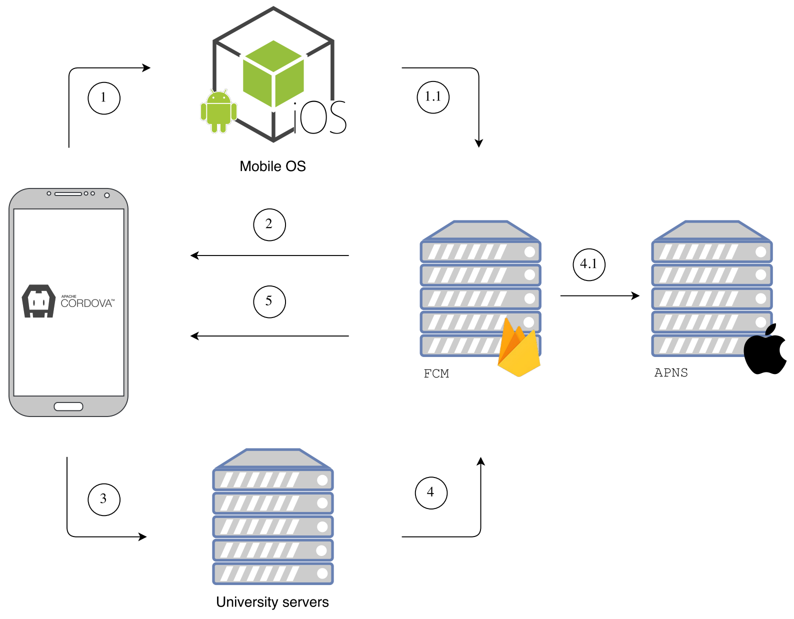


Figure 3.3: Push architecture

3.3.2.3 User Queries module

This module is the Web Services themselves. Behind the authentication module is where they are placed. Nevertheless, all the Web Services are not protected behind the authentication module, the information gathered by those Web Services is public because University's nature.

For more information, see ApiUPM[3], a web portal which joins most of the Web Services offered by the University.

3.4 Client modules

This project is designed to provide the most usual features offered by a University intranet moving them to a mobile perspective. As it has been said before, the mobile platforms where the project is focused are Android and iOS, both smartphones and tablets, because the former and the latter platforms joined make a high market share.

In this section, a deeply sight will be given to explain each facet of the mobile application, from OS core functionalities to high level Cordova code. In addition, how a Cordova mobile application are developed will be explained for a better project comprehension. Finally, during the explication of all development's process, relations between server features and

[3] `https://www.upm.es/apiupm/`

client necessities are going to be bound.

3.4.1 Mobile OS

The differences between the operative systems where the mobile application can run creates different development's paths. Depending on the election of these paths, the application will be partially or fully operative. For this reason, the project has two secondary goals: minimizing gaps between all platforms the project is interested in and maximizing shared code by every platform.

Some of the differences found during the project's development are:

- Permission access.
- Main properties of principal graphic layout.
- Amount of different devices per platform.
- Operative System version.
- WebViews.

Most of the previous differences only need a patch to be resolved definitely. However, there are differences that will require attention on every new version that is planned to be published. Those differences are principally related to graphics and user experience.

3.4.2 Cordova application

Since the installation of Cordova Framework is done, main steps to have a mobile application based on Cordova are [4]:

- **Create the App.** Following lines are needed to create an empty new Cordova project:

Listing 3.3: jquery.localize.js use

```
C:\>cordova create PersonalUPM es.upm.personal PersonalUPM
```

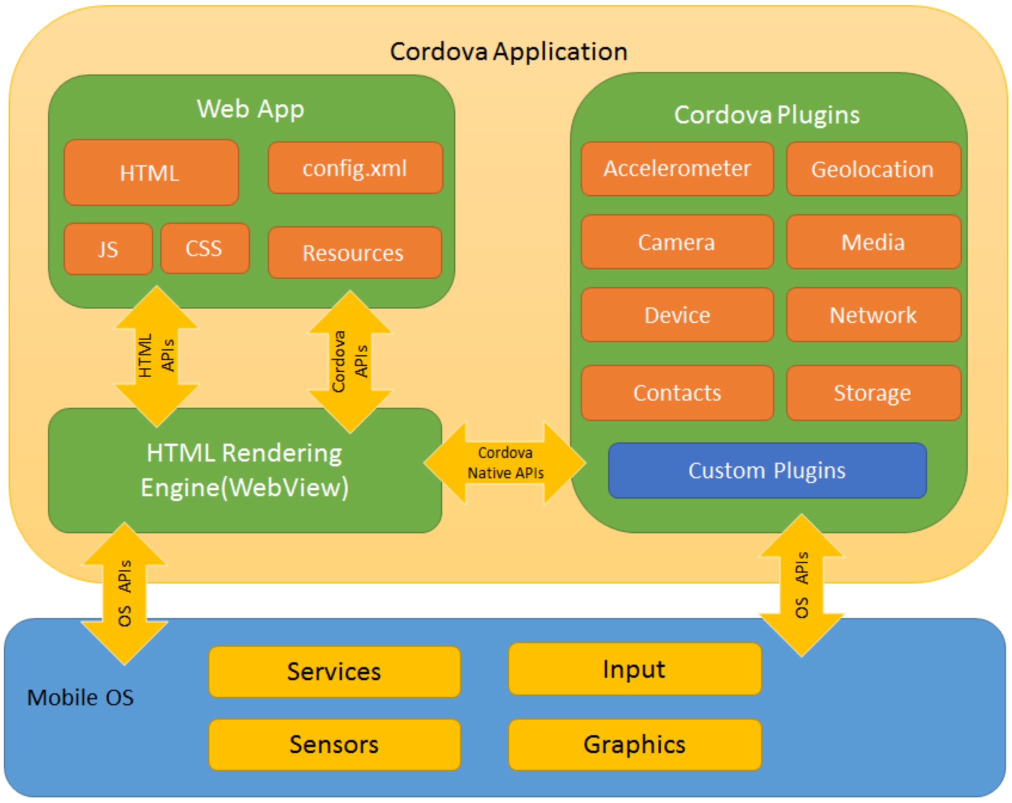


Figure 3.4: Cordova architecture

Where:

i *cordova* is the CLI command to invoke Cordova features.

ii *create* is the order to generate a new project.

iii *PersonalUPM* is the path where the project is going to be stored. It can be either relative or absolute.

iv *es.upm.personal* is the *Reverse domain-style identifier*. Should not change during application lifetime. This identifier will be the one which identifies the application in the app stores.

v *PersonalUPM* will be the name of the mobile application. This value can change during application lifetime.

- **Add platforms.** The addition of platforms will be required for building the application. The build uses native SDKs (*System Development Kit*) to compile the application code into native binaries. This project uses Android and iOS[4] platorms,

[4]The possibility of adding the iOS platform is exclusively available for OSX devices because only they have access to Xcode toolkit.

so:

Listing 3.4: Console command for adding a platform

```
C:\PersonalUPM> cordova platform add android ios
```

The previous line will allow the system where the project is developed access to build the application on those platforms.

- [Optional] **Add plugins.** If the application needs some utilities which only are obtainable through Cordova plugins, the way to install them is:

Listing 3.5: Console command for installing plugins

```
C:\PersonalUPM> cordova plugin install {name-of-the-plugin}
```

A huge quantity of plugins is available through the Cordova Plugin Library, where many of them are maintained by the community. One of the greatest characteristics of this Library, due to its wideness, is that most of the development needs are covered. As a consequence, the probability of having to develop a new plugin is very low. Although, a research work is needed for assuring the validity of a selected plugin, because for instance, maintenance could be out of date or the requirements of a plugin version exceed the ones of the project.

- **Build the App.** This is the last Cordova step before testing/publishing the mobile application. This order generates the Web App and embeds it in the native WebView, but its result will be different depending on the targeted platform.

  – Android: there are two types of building a Cordova application in android, according to the desired environment, *Development* or *Production*. The most usual build is the former and it is done with the following command:

  Listing 3.6: Command for generating an android development build

  ```
  C:\PersonalUPM> cordova build android
  ```

  To generate a build for the latter, an option needs to be appended to the previous command:

  Listing 3.7: Command for generating an android release build

```
C:\PersonalUPM> cordova build android --release
```

The difference between both builds is that a *Development* build will generate a *debug* `.apk` file for testing purposes while a *Production* build will generate an `.apk` ready to be signed and uploaded to a production environment.

– iOS: the command needed to compile the application shares the order of Cordova CLI but identifying "ios" as the platform to be builded:

Listing 3.8: Command for generating an iOS development build

```
C:\PersonalUPM> cordova build ios
```

Nevertheless, Apple's restriction of building applications only on OSX devices also prevents the generation of executable iOS files directly from Cordova CLI. Because of that, a `build` command generates a Xcode project file (`.xproject`) or a Xcode workspace file (`.xworkspace`).

Now, each project's client side module is going to be explained and related with its purpose inside a Cordova project.

3.4.2.1 config.xml

As its own name indicates, the `config.xml` file is the responsible of any configuration needed by the project. Its structure is related with the W3C's widget specification. The World Wide Web Consortium (W3C) provides a set of specifications collectively known as the Widget family of specifications. A Widget is defined by W3C (`http://dev.w3.org/2006/waf/widgets-land/`) as "an end-user's conceptualization of an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device." [12]

Following those specifications, file's structure is:

Listing 3.9: Project config.xml

```
<?xml version='1.0' encoding='utf-8'?>
<widget xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.
    apache.org/ns/1.0" id="es.upm.personal" version="1.1.1" android-
    versionCode="14" ios-CFBundleVersion="1.1.1.20170210">
```

```
<name>Personal UPM</name>
<description>
  Aplicacion para Personal de la Universidad Politecnica de Madrid
</description>
<author email="servicios.tic@upm.es" href="http://www.upm.es">
  Vicerrectorado de Servicios Informaticos y de Comunicacion
</author>

<!--project's plugins-->
<plugin name="cordova-plugin-console" version="1.0.3"/>
  ...
<plugin name="phonegap-plugin-push" spec="https://github.com/phonegap/
    phonegap-plugin-push#v2.0.x">
  <variable name="SENDER_ID" value="xxxxxxxxxxxx" />
</plugin>
<!--both api key needed for googlemaps plugin-->
<plugin name="https://github.com/phonegap-googlemaps-plugin/cordova-
    plugin-googlemaps">
  <variable name="API_KEY_FOR_ANDROID" value="xxxxxxxxxxxx"/>
  <variable name="API_KEY_FOR_IOS" value="xxxxxxxxxxxx"/>
</plugin>

<access origin="*" />
<allow-intent href="http://*/*" />
<allow-intent href="https://*/*" />
<allow-intent href="tel:*" />
<allow-intent href="sms:*" />
<allow-intent href="mailto:*" />
<allow-intent href="geo:*" />

<!--cordova preferences-->
  ...

<!--plugin spreferences-->
  ...

<platform name="android">
  <hook type="after_prepare" src="scripts/android_copy_to_drawable.js" />

  <!-- custom platform preferences and allow-intents -->

  <icon density="ldpi" src="res/iconos/android/icon-36-ldpi.png" />
    ...
  <splash src="res/splash/android/land-hdpi.png" density="land-hdpi"/>
    ...
  </platform>
  <platform name="ios">
    ...
  </platform>
</widget>
```

The previous is, in short, a widget from the W3C specification mentioned above with its information tags (*name, description, and author*). Widget's custom attributes like *android-versionCode* and *ios-CFBundleVersion* are required to implement each platform versioning. Remaining tags are:

- `<plugin>`: specifies a cordova plugin which will be included into the project. In addition, it can have nested tags like `<variable>` whose value will be passed as a plugin option during building time.
- `<access>`: allows the application to communicate with external domains defined in this tag.
- `<allow-intent>`: defines which URLs the application is allowed to send to the system for opening.
- `<preference>`: this tag sets application/plugin properties. It can be nested in a `<platform>` tag so its scope will not spread to all platforms.
- `<platform>`: generates a scope for each platform that is going to be used in the project. Tags contained inside a `<platform>` tag will only affect to the platform defined by it.
- `<icon>`: sets icon for the mobile application. A tag will be necessary for each icon size. In android, icons are sorted by pixel density (*ldpi, mdpi, hdpi, xhdpi, xxhdpi, and xxxhdpi*) while in iOS they are sorted by determinate pixel×pixel size.
- `<splash>`: this tag has the same behavior as `<icon>` tag, even by platform. However, the plugin *cordova-plugin-splashscreen* is required by Cordova to understand how apply it.

3.4.2.2 HTML

The HTML module contains the views of the application. Its function is to represent all the content needed by the UX (*User eXperience*) so user interactions are obtained correctly.

The navigation diagram of the application represents how HTML views are connected. Views are “pages” inside the jQuery Mobile argot, so each view will be a page to load into the DOM of the Web App. The HTML objects that are loaded inside pages are predefined by jQuery Mobile framework.

Mainly, the application views are structured using the same template for all pages except for the `index.html` which takes care of setting `meta` tags and loading JavaScript and CSS files.

Listing 3.10: Project’s index.html

```
<!DOCTYPE HTML>
<html><head>
  <meta charset="utf-8"/>
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
  <meta http-equiv="Content-Security-Policy" content="default-src gap://
      ready *; img-src * data:; script-src 'self' 'unsafe-inline' *;style-
      src 'unsafe-inline' *;font-src *">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
      maximum-scale=1.0, user-scalable=no" />
  <meta name="mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-capable" content="yes" />
  <meta name="apple-mobile-web-app-status-bar-style" content="black" />

  <!--CSS links-->
  <link rel="stylesheet" href="css/global.css" />

  <script type="text/javascript" src="cordova.js"></script>
  <!--More <script> tags-->

  <title>Personal UPM</title>
</head><body>
  <!--PAGE TEMPLATE-->
</body></html>
```

`cordova.js` is an auto-generated file which will be inserted during building time by Cordova Framework. It does not belong to the project itself but referencing it in `index.html` is necessary for obtaining Cordova events and plugin initializations.

As seen in section 2.2, all `.html` files, except `menu.html`, are located in `/www` folder, which is the Web App’s root directory. The file `menu.html` is located in `/www/fragments` folder because it contains only a piece of HTML code that will be inserted after `deviceready` event is triggered.

Listing 3.11: Project's page template

```
<!DOCTYPE HTML>
<html><head>
  <title>{pageTitle}</title>
</head><body>
<div data-role="page" id="{pageID}">
  <div data-role="header" id="{pageID}-header">
    <div class="toolbar">
      <span class="toolbar-nav-icon">
        <a id="{pageID}-button-menu" class="ui-btn" ><i class="zmdi zmdi-
            menu"></i></a>
      </span>
      <span class="toolbar-title">
        <h1><div data-localize="{pageID}.startHeader">{pageTitle}</div><
            span class="subtitle" style="display:none;"></span></h1>
      </span>
      <span class="toolbar-action-icons">
        <a id="{pageID}-button-logout" data-rel="popup" data-position-to="
            window" data-inline="true" data-transition="pop" class="ui-btn"
            ><i class="zmdi zmdi-power-off"></i></a>
      </span>
    </div>
  </div>
  <div role="main" id="{pageID}-content" class="ui-content page-content"></
      div>
</div>
</body></html>
```

A new HTML page is created by using the template showed in the listing above. It will be identified by the `pageID` variable, and it will be written in a new file. In this scope, a rule is followed by the project: *"new view, new page"*, in order to get a better code comprehension.

The HTML *toolbar* object has been defined exclusively to follow *Material Design* guidelines by dividing the whole width space into three blocks.

3.4.2.3 JavaScript

Business logic is implemented through JavaScript programming language. The project's main logic structure is arranged by the functionality of each JavaScript file. As a consequence, there are two main folders for organizing JavaScript inside `/www` folder: `vendor`, and `js`. The former folder groups together all JavaScript files related with third-party direct dependencies. The latter gathers project's own JavaScript files. This section is going to take a look of the latter's folder structure and also explain each file main functionality.

Firstly, in order to make the project's business logic easier to understand, the methodology used to develop JavaScript objects follows the pattern showed below:

Listing 3.12: Self-executing anonymouns function

```javascript
var object = (function () {
  var privateVar = 'foo';

  var privateFn = function () {
    ...
  };

  var thisWillBePublic = 'bar';

  return {
    variable: thisWillBePublic
}
})();

object.privateVar;     // returns undefined
object.privateFn();    // TypeError: object.privateFn is not a function
object.variable;       // returns 'bar'
```

The listing 3.12 shows the pattern called *self-executing anonymous function* [13] and it allows the JavaScript object to have not only public properties but also private ones. This approach offers some benefits like enabling the use of auxiliary functions that will be invisible to the rest of the project, or avoiding the rewrite of object properties.

Thus, the methodology used has been explained. Now, recovering principal purpose, project's JavaScript files are going to be explained.

The JavaScript files stored under *controllers* folder are responsible of administrating how a view is initialized or hided, and also taking care of all events triggered while its subordinated view is active for the user. Two essential functions are needed to control a view, and they will be executed after specific jQuery Mobile events are fired. With this information, a template has been created for minimizing further page additions:

Listing 3.13: Controller template

```
var {pageID} = (function() {

  /* Logic goes here */

  var initialize = function(){
    ...
  };

  var hide = function(){
    ...
  };

  var bindEvents = function(){
    $(document).on("pagebeforeshow", "#{pageID}", initialize)
      .on("pagehide", "#{pageID}", hide);
  };

  return {
    bindEvents: bindEvents
  };
})();
{pageID}.bindEvents();
```

The control of when a page is loaded into the DOM is leaded by `pagebeforeshow` event and the same occurs when the page is hided with `pagehide` event. They are jQuery Mobile events that will allow the application to know in which moments should do what actions.

The next stage of JavaScript files is *models* folder, where all the properly named objects of the application will be stored. Only one model follows the `prototype` pattern of JavaScript: `Alert.js`, because it is supposed to be instantiated many times. The remaining objects have been designed to act as *singleton* [3], they are:

- `Cache.js`: is the responsible of saving in disk the desired responses from Web Services requests. The cached files are only available while a Session is opened.
- `Preferences.js`: it is a custom preferences manager. The `localStorage` API

from HTML was used to fulfill this purpose, however, iOS WebView memory management do not keep some stored data, so `localStorage` was declined in favor of this implementation. It uses `FileEntry`, `FileReader`, and `FileWriter` APIs to manage preferences and to save them in disk.

- `Pushmanager.js`: is a wrapper for using the *Phonegap-plugin-push*. Its initialization contains the following tasks: (i) registering the mobile to *Firebase Cloud Messaging* platform, (ii) obtaining the *token* from the previous request, (iii) sending the *push token* through a Web Services to the University servers, and (iv) waiting for push notifications.

- `Session.js`: once login request has been successfully done, a token (see section 3.3.2.1) is received. This object will be in charge of checking if the connection between University servers and the mobile application is still valid.

- `User.js`: this file stores all the information related with the logged user and centralizes it.

Finally, other JavaScript files are contained just at the root of the `js` folder. This emplacement was selected due to the functionalities contained in those files have a global purpose in the project.

- `webservices.js`: this file gathers all the requests made by the mobile application. It also implements a wrapper of how ajax queries are supposed to be done.

- `utils.js`: a function library. It contains every shared function of the project. In addition, it contains logging functions for debugging purposes.

- `config.js`: it is a project's configuration file. This is where options like which environment to use (production or development) or timeout of a request are set.

- `App.js`: this file is the responsible of controlling application's global variables, to managing callbacks for global events. Furthermore, it initializes the mobile application itself when `ondeviceready` event is triggered.

3.4.2.4 CSS

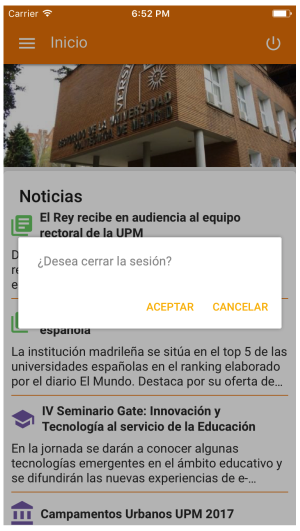
A mobile application, by default, is organized in different views that will arrange all the content. These views are designed to follow some patterns depending on which platform is been targeted by the application. Nevertheless, those patterns are not a restriction but an advice from the big companies behind the development of a platform.

As this application is not a one-platform development, many patterns are available for selecting the design, more precisely they are the *Material Design* by Google and the *iOS Human Interface* by Apple.

In this particular project, *Material Design* guidelines has been selected for leading the mobile application design. This selection was done because Google guidelines are more detailed and because android has a greater percent of market share than iOS as seen in section 1.1. However, the application style is not strictly guided by Google guidelines, due to it also has custom styles to provide the project its own personality.

On one hand, all the styles are written using Sass because its modularity avoids to have `.css` files with a huge number of lines. In this scope, the project follows the same rule as the HTML scope: *"new view, new page"*, but this time, instead of a *page* there will be a *.scss file*. In addition, there are more sass files to control general purpose goals like centralizing shared css values in variables, defining "functions" (sass' *mixins*) or overwriting values from frameworks. All those modular files are recognizable by their name. If the name of a sass file starts with the character "_", it means that it is prepared for being imported in a "main" sass file.

On the other hand, predefined Material Design styles can be used by the application thanks to NativeDroid2 jQuery Mobile theme. A summary of the use of NativeDroid2 elements is showed in figures 3.5, 3.6, and 3.7.



El Rey recibe en audiencia al equipo rectoral de la UPM

¿Desea cerrar la sesión?

española

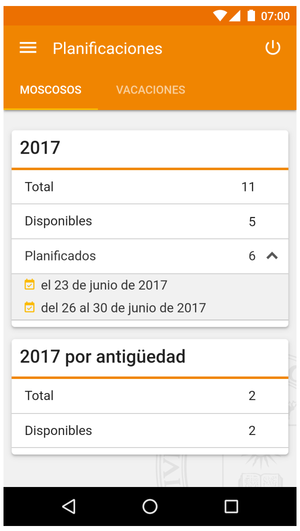
La institución madrileña se sitúa en el top 5 de las universidades españolas en el ranking elaborado por el diario El Mundo. Destaca por su oferta de...

IV Seminario Gate: Innovación y Tecnología al servicio de la Educación

En la jornada se darán a conocer algunas tecnologías emergentes en el ámbito educativo y se difundirán las nuevas experiencias de e-...

Campamentos Urbanos UPM 2017

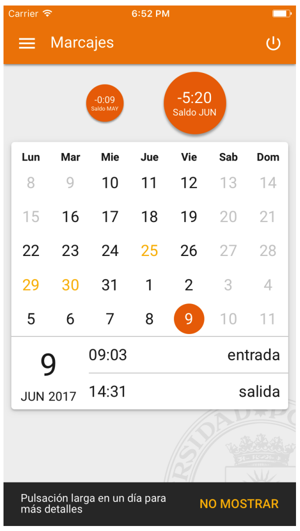
Figure 3.5: Logging out modal - iOS



2017	
Total	11
Disponibles	5
Planificados	6
el 23 de junio de 2017	
del 26 al 30 de junio de 2017	

2017 por antigüedad	
Total	2
Disponibles	2

Figure 3.6: Cards in Planification view - Android



Lun	Mar	Mie	Jue	Vie	Sab	Dom
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

9 JUN 2017		
	09:03	entrada
	14:31	salida

Pulsación larga en un día para más detalles

NO MOSTRAR

Figure 3.7: Snackbar with actions - iOS

As is shown in the figures above, the layout is shared across platforms, except for those elements that are part of the mobile OS system, like the status-bar that has different styles per platform, or the navigation-bar, which is only an element of Android, and iOS does not have it.

CHAPTER 4

Use case

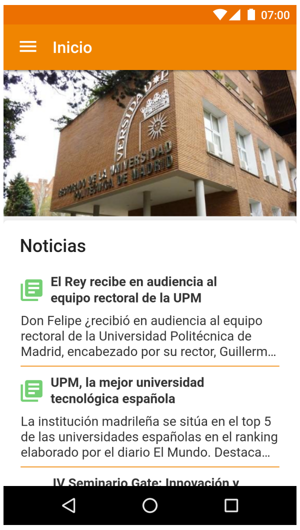
4.1 Introduction

This chapter describes the process needed in order to allow the information to be provided to a user by using the mobile application. This process will be presented as a walk-through among all the options offered by the application.

4.2 Mobile application

Once a user has installed the application and it is started, the first view showed, Fig. 4.1, will present an static image of University administration building and a list of University related news. The news presented in this page, are a link for viewing the full news information on another view, as shown in Fig. 4.2. That secondary view will show a Google Maps widget only if the news is located at a specific place. In addition, while the maps widget is showed, the user could click to the black icon in the upper-right corner to open Google Maps application for using, for example, GPS directions.

Regarding navigation, a side menu is revealed when the user clicks on the upper-left *menu* icon. That icon allows the user to display the menu which provides navigation to



El Rey recibe en audiencia al equipo rectoral de la UPM

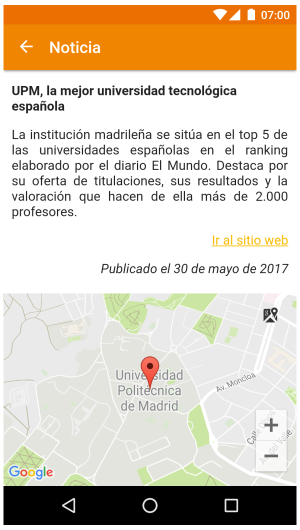
Don Felipe ¿recibió en audiencia al equipo rectoral de la Universidad Politécnica de Madrid, encabezado por su rector, Guillerm…

UPM, la mejor universidad tecnológica española

La institución madrileña se sitúa en el top 5 de las universidades españolas en el ranking elaborado por el diario El Mundo. Destaca…

IV Seminario Gate: Innovación y

Figure 4.1: Main page



UPM, la mejor universidad tecnológica española

La institución madrileña se sitúa en el top 5 de las universidades españolas en el ranking elaborado por el diario El Mundo. Destaca por su oferta de titulaciones, sus resultados y la valoración que hacen de ella más de 2.000 profesores.

Ir al sitio web

Publicado el 30 de mayo de 2017

Figure 4.2: News detailed view

each page available from the whole application pages but those whose purpose is to display a *detailed* information of the previous page, as shown in figures 4.2, 4.5, 4.9, 4.7, and 4.12.

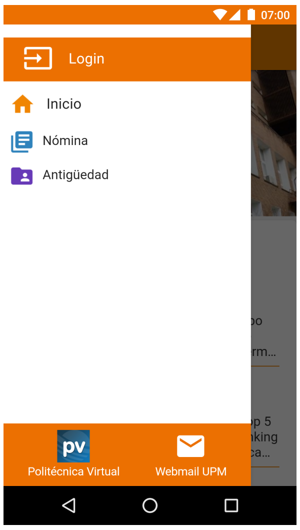


Figure 4.3: Initial navigation menu

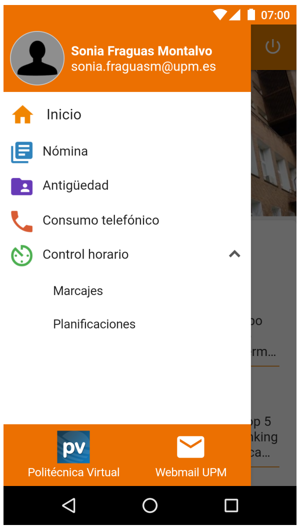


Figure 4.4: Navigation menu with user logged

The left side-menu changes its appearance depending on if a user is authenticated or not. The former state makes menu showing all the possibilities that the user can do, as shown in Fig. 4.4. Not all logged users see the same possibilities because, for instance, not every people have a mobile phone line associated or have to clock in, thus, the options displayed in menu rely upon which kind of user has logged in the application. The latter state is the state by default, Fig. 4.3, and makes menu display a link to the login page, and other links (one per option) that will also forward the user to the login page, as shown in Fig. 4.5, although when it does a correct login, it will be redirected to the page of the previous option selected. The menu will save the information of the last logged user for showing, although authentication is not provided, all the available options it had. To logout, the user needs to click on the upper-right *power-off* icon in order to show a dialog which will ask it to accept or not the logout action.

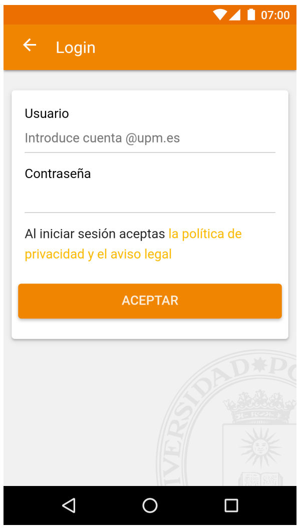

07:00
Login
Usuario
Introduce cuenta @upm.es
Contraseña
Al iniciar sesión aceptas la política de privacidad y el aviso legal
ACEPTAR

Figure 4.5: Login

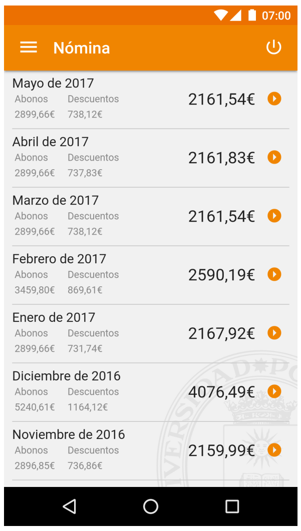
Now, each possible option is going to be explained for revealing all actions that the user can do. They are going to be explained in order of appearance in the menu layout.

4.2.1 Salary

This option will show a list of clickable elements that acts as a summary for all the month incomings computer-registered. Each element shows the final incoming, and the global value of payments and deducts for a month, as shown in Fig 4.6. When the user clicks on an element, a new view will appear without leaving the current page with the previous information itemized shorted by contract and a fixed ribbon that reminds the global values (Fig. 4.7). finally, in this second view, menu icon will not be displayed and back-button behavior is to go back to the summary view.

4.2.2 Seniority

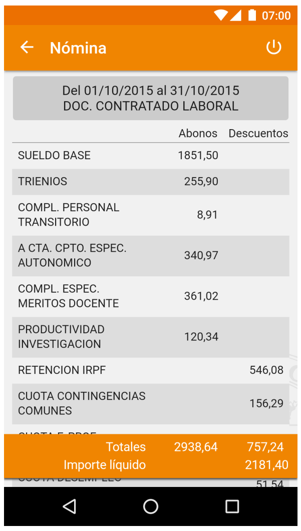
This page shows the seniority of the logged user. It can be specified in two types: *academic* and *research* seniority. If the user does not have any of those, its activity will be grouped


07:00

Nómina

Mayo de 2017 Abonos 2899,66€ Descuentos 738,12€	2161,54€	
Abril de 2017 Abonos 2899,66€ Descuentos 737,83€	2161,83€	
Marzo de 2017 Abonos 2899,66€ Descuentos 738,12€	2161,54€	
Febrero de 2017 Abonos 3459,80€ Descuentos 869,61€	2590,19€	
Enero de 2017 Abonos 2899,66€ Descuentos 731,74€	2167,92€	
Diciembre de 2016 Abonos 5240,61€ Descuentos 1164,12€	4076,49€	
Noviembre de 2016 Abonos 2896,85€ Descuentos 736,86€	2159,99€	

Figure 4.6: Salary view

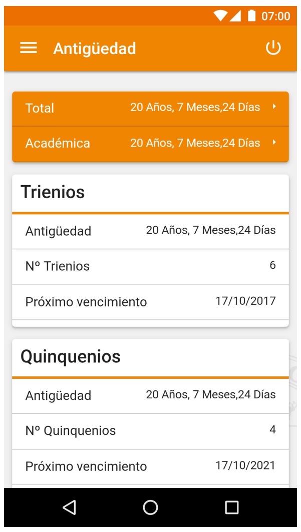

07:00

Nómina

Del 01/10/2015 al 31/10/2015
DOC. CONTRATADO LABORAL

	Abonos	Descuentos
SUELDO BASE	1851,50	
TRIENIOS	255,90	
COMPL. PERSONAL TRANSITORIO	8,91	
A CTA. CPTO. ESPEC. AUTONOMICO	340,97	
COMPL. ESPEC. MERITOS DOCENTE	361,02	
PRODUCTIVIDAD INVESTIGACION	120,34	
RETENCION IRPF		546,08
CUOTA CONTINGENCIAS COMUNES		156,29
Totales	2938,64	757,24
Importe líquido		2181,40

Figure 4.7: Month incoming detailed


07:00

Antigüedad

Total	20 Años, 7 Meses,24 Días
Académica	20 Años, 7 Meses,24 Días

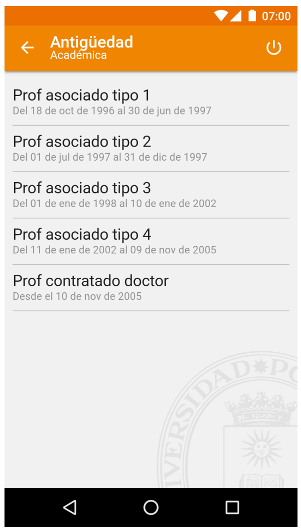
Trienios

Antigüedad	20 Años, 7 Meses,24 Días
Nº Trienios	6
Próximo vencimiento	17/10/2017

Quinquenios

Antigüedad	20 Años, 7 Meses,24 Días
Nº Quinquenios	4
Próximo vencimiento	17/10/2021

Figure 4.8: Seniority outline view


07:00

Antigüedad
Académica

Prof asociado tipo 1
Del 18 de oct de 1996 al 30 de jun de 1997

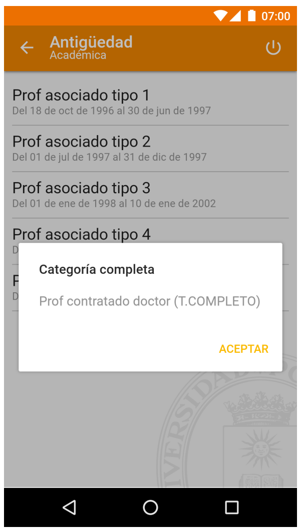
Prof asociado tipo 2
Del 01 de jul de 1997 al 31 de dic de 1997

Prof asociado tipo 3
Del 01 de ene de 1998 al 10 de ene de 2002

Prof asociado tipo 4
Del 11 de ene de 2002 al 09 de nov de 2005

Prof contratado doctor
Desde el 10 de nov de 2005

Figure 4.9: Academic seniority history


07:00

Antigüedad
Académica

Prof asociado tipo 1
Del 18 de oct de 1996 al 30 de jun de 1997

Prof asociado tipo 2
Del 01 de jul de 1997 al 31 de dic de 1997

Prof asociado tipo 3
Del 01 de ene de 1998 al 10 de ene de 2002

Prof asociado tipo 4

Categoría completa

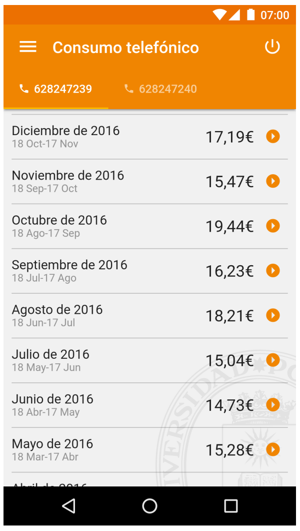
Prof contratado doctor (T.COMPLETO)

ACEPTAR

Figure 4.10: Academic seniority history detailed

into *total* seniority. In the main view of this option the user can see an overview of its trienniums, quinquenniums, and six-year terms, as shown in Fig. 4.8. Quinquenniums are only obtainable with academic activity, and six-year terms are only achievable with research activity. By clicking on an element of the orange card, a new view, as shown in Fig. 4.9, will appear showing a list that contains every contract done inside the seniority scope clicked. Each list element are also clickable and a click will show a dialog with the full name of the contract (Fig. 4.10).

4.2.3 Mobile phone bills

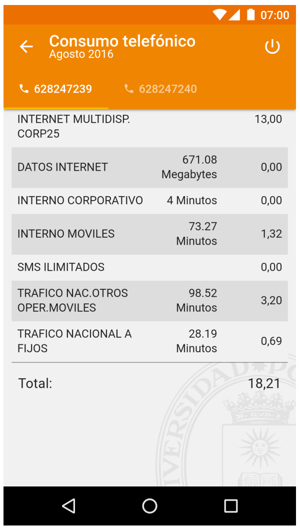

07:00

Consumo telefónico

628247239 628247240

Diciembre de 2016 18 Oct-17 Nov	17,19€
Noviembre de 2016 18 Sep-17 Oct	15,47€
Octubre de 2016 18 Ago-17 Sep	19,44€
Septiembre de 2016 18 Jul-17 Ago	16,23€
Agosto de 2016 18 Jun-17 Jul	18,21€
Julio de 2016 18 May-17 Jun	15,04€
Junio de 2016 18 Abr-17 May	14,73€
Mayo de 2016 18 Mar-17 Abr	15,28€

Figure 4.11: Mobile phone bills


07:00

Consumo telefónico
Agosto 2016

628247239 628247240

INTERNET MULTIDISP. CORP25		13,00
DATOS INTERNET	671.08 Megabytes	0,00
INTERNO CORPORATIVO	4 Minutos	0,00
INTERNO MOVILES	73.27 Minutos	1,32
SMS ILIMITADOS		0,00
TRAFICO NAC.OTROS OPER.MOVILES	98.52 Minutos	3,20
TRAFICO NACIONAL A FIJOS	28.19 Minutos	0,69
Total:		18,21

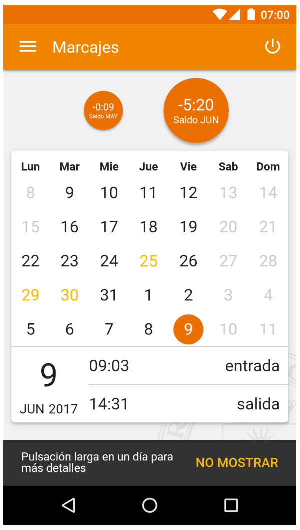
Figure 4.12: Month bill detailed

This option will be available for only those users which have a mobile phone line associated beside their post. If more than a line is assigned to a user, changing between them is as easy as clicking in a tab. Each line is identified by the phone number. As regards layout, page's main view is similar to salary's main view, showing a list of bills ordered by month whose elements display the total cost of a bill and the billing cycle (Fig. 4.11). When an element is clicked, a detail view will be shown with an itemization of charges, as shown in Fig. 4.12.

4.2.4 Clock in

The page showed in Fig. 4.13 is accessible for staff that clock in when they arrive to/go from work. This view displays a calendar showing the last thirty days approximately, they can be colored in three ways: (i) **black** represents a normal day, where the user has its clockings in perfectly done; (ii) light-gray days are disable for user interaction because they represent non-working days, for instance, public holidays; and (iii) **light-orange** ones are days that have some anomalies, which can be, e.g. vacation or off work days. Currently selected day is represented with a **dark-orange circle**. Finally, if the user does a long click on an day, as the bottom bar suggests, a dialog will appear, as shown in Fig. 4.14, which details worked time of day selected and total computed month time in that moment.

The two bubbles floating at the top of the view displays total computed time for the current and the previous month. The number inside can be positive, when the user has

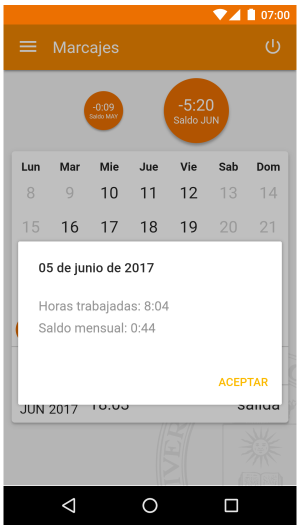


Lun	Mar	Mie	Jue	Vie	Sab	Dom
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

9 JUN 2017
09:03 entrada
14:31 salida

Pulsación larga en un día para más detalles NO MOSTRAR

Figure 4.13: Clocking in view



05 de junio de 2017

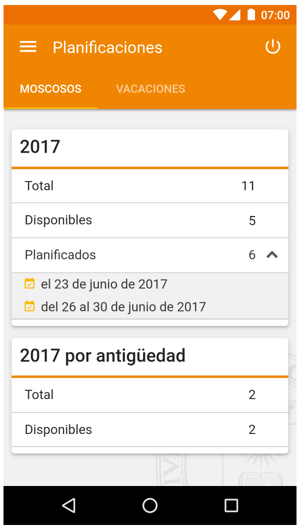
Horas trabajadas: 8:04
Saldo mensual: 0:44

ACEPTAR

Figure 4.14: Day detail modal

work more time than it should, or negative, when it have done less time.

4.2.5 Holidays



2017

Total	11
Disponibles	5
Planificados	6

el 23 de junio de 2017
del 26 al 30 de junio de 2017

2017 por antigüedad

Total	2
Disponibles	2

Figure 4.15: Planifications view

This last page shows a summary of the vacation days available for the current logged user, shorting them by type. For choosing what type of vacation days has to show the application, the user only has to click on the tab it desires.

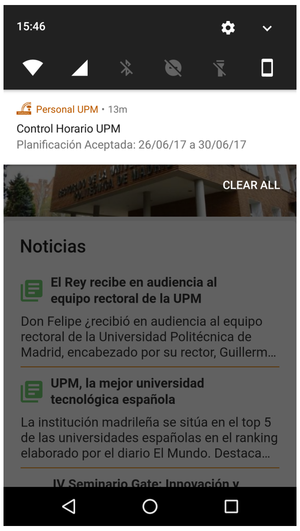
Each type view is arranged in cards that contains all the information shorted by year. This information indicates total amount of vacation days, and which of them are either available or planned. Planned vacation days only appear if the days asked for holidays are accepted, as shown in Fig. 4.16. Moreover, the icon next to the holidays date changes according of the state of the planned period: a light-orange calendar icon for those which have not occur yet, and a green check icon for those that have happened.

4.3 Push notifications

One last feature of the mobile application are the push notifications. They allows the user to be alerted when a specifically action has happened. Figure 4.16 shows a notification which notifies the user that the days it asked for being holidays has been accepted.

Current notifiable events are:

- Approval of vacation days,
- Salary payment, and
- New phone bill receipt.


15:46

Personal UPM • 13m
Control Horario UPM
Planificación Aceptada: 26/06/17 a 30/06/17

CLEAR ALL

Noticias

El Rey recibe en audiencia al equipo rectoral de la UPM
Don Felipe ¿recibió en audiencia al equipo rectoral de la Universidad Politécnica de Madrid, encabezado por su rector, Guillerm...

UPM, la mejor universidad tecnológica española
La institución madrileña se sitúa en el top 5 de las universidades españolas en el ranking elaborado por el diario El Mundo. Destaca...

IV Seminario Gate: Innovación y

Figure 4.16: Push notification example

CHAPTER 5

Conclusions and future work

5.1 Introduction

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

5.2 Conclusions

In this project we have developed a mobile application for a University intranet, where the most usual user actions can be done. For this purpose, hybrid programming has been the methodology followed to reduce developing time, in particular, Apache Cordova has been the selected framework to embed a Web Application into multiple mobile platforms.

This project has developed the client side of the architecture, which is formed by two modules, the mobile OS and the Apache Cordova application. The former offers some native features like sensors, graphics, and services. The latter is where the Web App will support to provide all its features. Finally, the server side of this project is the responsible of storing all user data and providing them to the client side.

In the following sections I will describe in depth the achieved goals, the problems faced and some suggestions for a future work.

5.3 Achieved goals

In the following section I will explain the achieved features and goals that are available in this project.

Collect main necessities from users. We have been study which actions are performed the most among those qualified as deployable. The actions that has been considered are the ones from the web platform *Politécnica Virtual*. Thanks to this platform, the main functionalities of the mobile application were founded.

Session establishment. In our project, one of the most significant points was the connection between University servers and the mobile application. The importance of this point is derived from a secondary goal: reducing server load by not keeping traditional sessions. This goal was also achieved thanks to jQuery Ajax implementation, which allows us to personalize it the way we need.

Enable Push Notification Services. Every mobile application with login interface has a big chance of needing the implementation of Push Notification Services. Informing users of new milestones related to the application's functionalities is almost a requirement for every mobile application. Thanks to the Cordova plugin *phonegap-plugin-push* this goal has been able to be achieved.

Use Sass to simplify the application styling. Initially, all application styles were defined in a `.css` file. This implementation was far enough to be maintainable so a mid-project new goal was traducing the previous `.css` file into a modular structure of `.scss` files.

Develop the application using Apache Cordova. This was the main goal of the project. It was achieved thanks to the use of multiple frameworks as jQuery, NativeDroid2 or Sha256.js which improved and simplified programming tasks like Ajax requests, DOM transformations or ciphering sensible data.

5.4 Problems faced

During the development of this project we had to face some problems. These problems are listed below:

- Maps implementation: When an event provides the location where it will be presented, that location is showed in a Google Maps native view. The plugin used to achieve this is named *cordova-plugin-googlemaps*. It adds a native view of Google Maps centered in some location. Problems became when a html view was placed above, at least, a part of the space occupied by the native view. The plugin places the native view behind the WebView generated by Cordova and sets transparency on to that WebView. However, if some html view is placed after the Maps view is initialized, user interactions will be picked up by the Maps view. This problem happened for instance with the logout dialog. To solve it, we need to toggle the Maps instance clickability each time an html view will be placed above it.
- Migration from GCM to FCM: Initially, the first push implementation in the project was done using GCM services. However, Google evolved this services into FCM. Thus, the mobile application had to evolve as well. The problem faced was managing the specifically version of the Cordova plugin which will enable us to implement again Push Services. Finally, the problem was solved by changing which plugin's branch is used by the application.

5.5 Future work

In the following section I will explain the possible new features or improvements that could be done to the project.

Notification channels. Currently, the user can only receive notifications about when its salary has been emitted, when a vacation day is approved and when its phone bill has been collected. The use of Push Services can improve this by letting the user choose which notifications likes to receive, and offering notification channels like "University events".

Migrate to Materializecss framework. This is the near future of this project's development. Although the application runs smoothly, with the addition of further capabilities will make it increase in size and computational work. By using only one framework, we can reduce application's size and also implement some new features like an event carousel.

Implement Web Workers. The Web Workers element runs scripts in the background that can't be interrupted by other scripts or user interactions [8]. This speeds up background tasks and could be implemented for attending graphic tasks in separated threads, which will improve user experience.

Add POST requests. Current version of the mobile application can only show user information. The idea is to develop a way to asking for, i.e, vacation days or opening a clock issue. This can also be extended by letting the responsible people accept or deny those requests from the mobile application.

Bibliography

[1] Rosa Alarcón and Erik Wilde. Restler: crawling restful services. In *Proceedings of the 19th international conference on World wide web*, pages 1051–1052. ACM, 2010.

[2] Oscar Axelsson and Fredrik Carlström. Evaluation targeting react native in comparison to native mobile development, 2016. Student Paper.

[3] Ethan Brown. *Learning JavaScript: JavaScript Essentials for Modern Application Development.* " O'Reilly Media, Inc.", 2016.

[4] The Apache Software Foundation. Cordova documentation - overview, 2016. Available on: https://cordova.apache.org/docs/en/latest/guide/overview/.

[5] Jim Garvin. Github - coderifous/jquery-localize.

[6] John Lim. Adodb library for php manual. *EEUU: phpLens. Agosto*, 2006.

[7] Davood Mazinanian and Nikolaos Tsantalis. An empirical study on the use of css preprocessors. In *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, volume 1, pages 168–178. IEEE, 2016.

[8] Mark Pilgrim. *HTML5: Up and Running: Dive into the Future of Web Development.* " O'Reilly Media, Inc.", 2010.

[9] OS Smartphone. Market share, 2016 q3. *IDC [on-line]. http://www.idc.com/promo/smartphone-market-share/os*, 2017.

[10] Chris Veness. Github - chrisveness/crypto.

[11] John M Wargo. *Apache Cordova 4 Programming.* Pearson Education, 2015.

[12] Scott Wilson, Florian Daniel, Uwe Jugel, and Stefano Soi. Orchestrated user interface mashups using w3c widgets. In *International Conference on Web Engineering*, pages 49–61. Springer, 2011.

[13] Nicholas C Zakas. *Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers.* No Starch Press, 2016.