



**Universidad  
Rey Juan Carlos**

**GRADO EN INGENIERIA EN TELEMATICA**

Curso Académico 2021/2022

Trabajo Fin de Grado

**TÍTULO DEL TRABAJO EN MAYÚSCULAS**

Autor : David González Endrinal

Tutor : Dr. Gregorio Robles Martínez



# **Trabajo Fin de Grado**

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

**Autor :** David González Endrinal

**Tutor :** Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día        de  
de 202X, siendo calificada por el siguiente tribunal:

**Presidente:**

**Secretario:**

**Vocal:**

y habiendo obtenido la siguiente calificación:

**Calificación:**

Fuenlabrada, a        de        de 202X



*Dedicado a  
mi familia / mi abuelo / mi abuela*



# **Agradecimientos**

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.



# **Resumen**

Este proyecto consiste en la realización de una Aplicación de Web Progresiva (PWA) que ayudará y facilitará a los docentes el proceso de ofrecer a los alumnos los cursos/seminarios y jornadas tecnológicas que se impartirán en la Universidad durante el curso. Esta aplicación y todo su desarrollo se enmarca dentro del modelo cliente-servidor, donde por un lado tendremos por parte del cliente la PWA y por parte del servidor toda la administración y gestión de los contenidos que contendrá dicha aplicación. Para el desarrollo de la parte servidor se han utilizado diferentes entornos y lenguajes, destacando Python, Node y MongoDB para la gestión y almacenamiento en bases de datos. Con la creación de esta aplicación se intenta, por el lado docente, buscar una eficiencia a la hora de repartir y adjudicar las distintas aulas de las que se dispone para la realización de los distintos cursos y seminarios, por otro lado, por el lado de los alumnos, se busca tener un entorno de fácil manejo y simpleza a la hora de informarse, apuntarse y obtener los créditos correspondientes de dichos cursos y seminarios.



# **Summary**

This project consists of the realization of a Progressive Web Application (PWA) that will help and facilitate the teachers the process of offering the students the courses / seminars and technological conferences that will be taught at the University during the course. This application and all its development is framed within the client-server model, where, on the one hand, we will have the PWA on the part of the client and on the part of the server all the administration and management of the contents that said application will contain. For the development of the server part, different environments and languages have been used, highlighting Python, Node and MongoDB for management and storage in databases. With the creation of this application, an attempt is made, on the teaching side, to seek efficiency when distributing and allocating the different classrooms available to carry out the different courses and seminars, on the other hand, on the side of The students seek to have an environment that is easy to use and simple when it comes to obtaining information, signing up and obtaining the corresponding credits for said courses and seminars.



# Contents

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Contexto . . . . .	1
1.1.1	Estilo . . . . .	1
1.2	Estructura de la memoria . . . . .	3
<b>2</b>	<b>Objetivos</b>	<b>5</b>
2.1	Objetivo general . . . . .	5
2.2	Objetivos específicos . . . . .	5
2.3	Planificación temporal . . . . .	5
<b>3</b>	<b>Estado del arte</b>	<b>7</b>
3.1	Angular . . . . .	7
3.1.1	@zxing/ngx-scanner v.3.0.0 . . . . .	8
3.1.2	@angular/pwa v.0.1101.1 . . . . .	8
3.2	TypeScript . . . . .	9
3.3	HTML . . . . .	9
3.4	CSS . . . . .	9
3.5	Bootstrap . . . . .	10
3.6	Node . . . . .	10
3.7	MongoDB . . . . .	11
3.8	Python . . . . .	11
<b>4</b>	<b>Diseño e implementación</b>	<b>13</b>
4.1	Arquitectura general . . . . .	13
4.1.1	Cliente . . . . .	14

4.1.2	Servidor . . . . .	15
4.2	Modelo de datos . . . . .	16
4.2.1	MongoDB . . . . .	16
4.2.2	Registros de Alumnos y Eventos . . . . .	18
4.2.3	Plantilla documento de obtención créditos . . . . .	19
4.3	Diseño e implementación por funcionalidad . . . . .	21
4.3.1	Cliente . . . . .	21
4.3.1.1	Login . . . . .	21
4.3.1.2	Usuario . . . . .	22
4.3.1.3	Administrador . . . . .	27
4.3.2	Servidor . . . . .	31
<b>5</b>	<b>Experimentos y validación</b>	<b>35</b>
5.1	Server Workers . . . . .	35
5.2	Instalación en un dispositivo móvil . . . . .	36
5.3	Optimización . . . . .	37
5.4	Seguridad . . . . .	38
<b>6</b>	<b>Resultados</b>	<b>39</b>
<b>7</b>	<b>Conclusiones</b>	<b>41</b>
7.1	Consecución de objetivos . . . . .	41
7.2	Aplicación de lo aprendido . . . . .	41
7.3	Lecciones aprendidas . . . . .	42
7.4	Trabajos futuros . . . . .	42
<b>A</b>	<b>Manual de usuario</b>	<b>43</b>

# List of Figures

4.1	Arquitectura general.	14
4.2	Modelo Eventos.	16
4.3	Modelo Usuarios.	17
4.4	Modelo Salas.	17
4.5	Modelo Salas.	18
4.6	Modelo del archivo CSV.	18
4.7	Plantilla PDF creditos.	20
4.8	Página Login	21
4.9	Página Login	22
4.10	Menú principal de la aplicación	22
4.11	Página de inicio.	23
4.12	Pestaña de validacion de QR	24
4.13	Pestaña de validacion de QR con mensaje de error	24
4.14	Página de calendario	25
4.15	Página de descripción del evento y botón de apuntarse	26
4.16	Página para obtener el certificado de créditos	26
4.17	Pop-up con la vista del certificado	27
4.18	Vista de administrador de la pestaña Lista de Eventos	28
4.19	Contenido de Eventos.zip descargado al presionar "Descargar PDF Todos"	29
4.20	Vista de administrador de la ventana informativa del evento de Telefónica	30
4.21	Pestaña Calendario Eventos	31
5.1	Service Workers funcionando en la PWA	36
5.2	PWA en la pantalla de inicio del dispositivo móvil	37

5.3 Salida tras ejecutar python optimizarSalas.py . . . . .	38
---	----

# **Chapter 1**

## **Introducción**

El siguiente trabajo consta de la realización de una aplicación web progresiva (PWA) que ayudará y facilitara a los docentes el proceso de ofrecer a los alumnos los cursos/seminarios y jornadas tecnológicas que se impartirán en la Universidad durante el curso.

### **1.1 Contexto**

En cuanto a la parte web, para los alumnos, supondrá una herramienta de fácil manejo donde ver estos horarios, apuntarse a ellos y obtener los creditos correspondientes por acudir a estas charlas, agilizando y quitando carga de trabajo al docente encargado de atender a las peticiones y facilitar el documento que certifica que ese alumno acudió a esas charlas dándole los créditos que le corresponden por ello. La aplicación, en su parte backend, destaca en la optimización de gestionar las salas disponibles en la universidad en función de los alumnos apunados a cada uno de estos seminarios, quitando así una gran carga de trabajo al docente encargado de gestionar y modificar las salas y sus horarios de forma manual para que todos los alumnos apuntados a un seminario puedan acudir a el con total comodidad y garantizando un sitio a cada uno de ellos.

#### **1.1.1 Estilo**

Recomiendo leer los consejos prácticos sobre escribir documentos científicos en L<sup>A</sup>T<sub>E</sub>X de Diomidis Spinellis<sup>1</sup>.

---

<sup>1</sup><https://github.com/dspinellis/latex-advice>

Lee sobre el uso de las comas<sup>2</sup>. Las comas en español no se ponen al tuntún. Y nunca, nunca entre el sujeto y el predicado (p.ej. en “Yo, hago el TFG” sobre la coma). La coma no debe separar el sujeto del predicado en una oración, pues se cortaría la secuencia natural del discurso. No se considera apropiado el uso de la llamada coma respiratoria o *coma criminal*. Solamente se suele escribir una coma para marcar el lugar que queda cuando omitimos el verbo de una oración, pero es un caso que se da de manera muy infrecuente al escribir un texto científico (p.ej. “El Real Madrid, campeón de Europa”).

A continuación, viene una figura, la Figura ???. Observarás que el texto dentro de la referencia es el identificador de la figura (que se corresponden con el “label” dentro de la misma). También habrás tomado nota de cómo se ponen las “comillas dobles” para que se muestren correctamente. Nota que hay unas comillas de inicio (“) y otras de cierre (”), y que son diferentes. Volviendo a las referencias, nota que al compilar, la primera vez se crea un diccionario con las referencias, y en la segunda compilación se “rellenan” estas referencias. Por eso hay que compilar dos veces tu memoria. Si no, no se crearán las referencias.

A continuación un bloque “verbatim”, que se utiliza para mostrar texto tal cual. Se puede utilizar para ofrecer el contenido de correos electrónicos, código, entre otras cosas.

```
From gaurav at gold-solutions.co.uk Fri Jan 14 14:51:11 2005
From: gaurav at gold-solutions.co.uk (gaurav_gold)
Date: Fri Jan 14 19:25:51 2005
Subject: [Mailman-Users] mailman issues
Message-ID: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>
```

Dear Sir/Madam,

How can people reply to the mailing list? How do i turn off  
this feature? How can i also enable a feature where if someone  
replies the newsletter the email gets deleted?

Thanks

```
From msapiro at value.net Fri Jan 14 19:48:51 2005
From: msapiro at value.net (Mark Sapiro)
Date: Fri Jan 14 19:49:04 2005
Subject: [Mailman-Users] mailman issues
```

---

<sup>2</sup><http://narrativabreve.com/2015/02/opiniones-de-un-corregidor-de-estilo-11-recetas-par.html>

In-Reply-To: <003c01c4fa40\$1d99b4c0\$94592252@gaurav7klgnyif>

Message-ID: <PC173020050114104851057801b04d55@msapiro>

gaurav\_gold wrote:

>How can people reply to the mailing list? How do i turn off  
this feature? How can i also enable a feature where if someone  
replies the newsletter the email gets deleted?

See the FAQ

>Mailman FAQ: <http://www.python.org/cgi-bin/faqw-mm.py>  
article 3.11

## 1.2 Estructura de la memoria

En esta sección se debería introducir la estructura de la memoria.

Así:

- En el primer capítulo se hace una intro al proyecto.
- En el capítulo 2 (ojo, otra referencia automática) se muestran los objetivos del proyecto.
- A continuación se presenta el estado del arte en el capítulo 3.
- ...



# **Chapter 2**

## **Objetivos**

### **2.1 Objetivo general**

La realización de este trabajo de fin de frado consiste en crear una Aplicación Web Progresiva o PWA que proporcionara a alumnos y a docentes una herramienta de fácil manejo con la que estar informado y gestionar los distintos cursos/seminarios y jornadas tecnológicas que se impartirán en la Universidad durante el curso.

### **2.2 Objetivos específicos**

- Automatizar el reparto de las aulas durante los días y las horas disponibles
- Informar al docente o administrador de la aplicación del número de alumnos apuntados a los distintos cursos/seminarios y jornadas
- Informar a los alumnos del horario de los distintos cursos/seminarios y jornadas tecnológicas
- Gestionar (apuntando y borrando) a los alumnos interesados en los seminarios disponibles

### **2.3 Planificación temporal**

A mí me gusta que aquí pongáis una descripción de lo que os ha llevado realizar el trabajo. Hay gente que añade un diagrama de GANTT. Lo importante es que quede claro cuánto tiempo

llevas (tiempo natural, p.ej., 6 meses) y a qué nivel de esfuerzo (p.ej., principalmente los fines de semana).

# Chapter 3

## Estado del arte

A continuacion se detallan las tecnologías utilizadas en el desarrollo del proyecto.

### 3.1 Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript (lenguaje del que hablaremos más adelante), de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Entre sus virtudes se destacan:

- La posibilidad de utilizar templates declarativos, aplicar inyecciones de dependencias y crear componentes reutilizables.
- Es modular, por lo cual se basa en un core y en módulos que permiten acceder a más características cuando es necesario. Permite crear componentes, razón por la cual se destaca en la reutilización de los elementos.
- Ejecuta la primera vista de tu aplicación en node.js, .NET, PHP, y otros servidores para renderizado de forma casi instantánea obteniendo solo HTML y CSS. También abre posibilidades para la optimización del SEO del sitio, incluyendo configuración.
- Son aplicaciones que cargan rápidamente gracias al nuevo enrutador de componentes.

Este ofrece una división automática de códigos para que los usuarios solo carguen el código necesario para procesar la vista que solicitan.

- Por último, dentro de las cosas que debemos tener en cuenta de Angular, es que nuestros proyectos deben ser compilados para traducirlos en una aplicación Web que podamos publicar. Este compilador ha sido mejorado a partir de Angular 5, permitiendo ahora compilación incremental y logrando como resultado que los re-build sean más rápidos.

A continuación destacamos dos paquetes npm que son importantes y sobre los que gira gran parte de este proyecto de fin de grado:

### 3.1.1 @zxing/ngx-scanner v.3.0.0

Paquete con el que por medio de una simple linea de código permite a nuestra aplicación angular (PWA) activar la cámara de nuestro dispositivo y pudiendo leer códigos QR o Códigos de barras simples obteniendo información de los mismos. En este proyecto daremos uso a este paquete a la hora de validar nuestra presencia en cualquiera de las charlas o seminarios en los que estemos inscritos y así poder certificar que hemos asistido a ellos para más tarde poder pedir el certificado con los créditos correspondientes.

### 3.1.2 @angular/pwa v.0.1101.1

Es el paquete principal que usaremos para convertir nuestra aplicación Angular en una Aplicación de Web Progresiva (PWA), mediante el comando *ng add @angular/pwa* se añadirán una serie de archivos y se modificarán otros existentes para poder realizar este cambio, dichas modificaciones son:

1. Se agregará el paquete *@angular/service-worker* al proyecto
2. Habilita la compatibilidad con la compilación del trabajador de servicios en la CLI.
3. Importa y registra al trabajador del servicio en el módulo de la aplicación.
4. Actualiza el archivo *index.html* incluyendo un enlace para agregar el archivo *manifest.webmanifest*.
5. Instala archivos de iconos para admitir la aplicación web progresiva (PWA) instalada.

6. Crea el archivo de configuración del service worker *llamado ngsw-config.json*, que especifica los comportamientos de almacenamiento en caché y otras configuraciones.

## 3.2 Typescript

Typescript es un superset de JavaScript, decimos que una tecnología es un superset de un lenguaje de programación, cuando puede ejecutar programas de la tecnología, Typescript en este caso, y del lenguaje del que es el superset, JavaScript en este mismo ejemplo.

Esto permite que se pueda integrar Typescript en proyectos existentes de JavaScript sin tener que reimplementar todo el código del proyecto en Typescript, de hecho, es común que existan proyectos que introduzcan tanto Typescript como JavaScript.

La principal característica de Typescript es el tipado estático, decimos que un lenguaje es de tipado estático cuando no es necesario definir las variables antes de su uso. Esto implica que la tipificación estática tiene que ver con la declaración explícita (o inicialización) de las variables antes de que se empleen.

## 3.3 HTML

HTML son las siglas de HyperText Markup Language, se trata del lenguaje estándar utilizado para la creación de páginas web. Fue creado por Tim Berners-Lee en los años ochenta. Es un lenguaje de marcas, esto quiere decir que para indicar la estructura del documento se utilizan etiquetas. Al ser un estándar, todos los navegadores leen e interpretan el lenguaje HTML y lo presentan de un modo similar.

## 3.4 CSS

CSS son las siglas de Cascading Style Sheets, es un lenguaje que permite establecer la apariencia de un documento escrito mediante un lenguaje de marcas. CSS permite asociar reglas a los elementos que aparecen en una página web, las reglas indican como debe representarse el contenido de esos elementos.

## 3.5 Bootstrap

Bootstrap es un framework front-end que permite la construcción de páginas web adaptativas. Se trata de un software de código abierto creado en el año 2011 por Mark Otto y Jacob Thornton, empleados de Twitter.

Bootstrap ha evolucionado desde ser un proyecto enteramente basado en CSS hasta incluir múltiples plugins JavaScript e iconos junto con formularios y botones. Presenta una cuadrícula de doce columnas y 940px de ancho.

La creación de páginas web mediante Bootstrap se simplifica al disponer de elementos pre-definidos fácilmente incluibles en cualquier página web, tales como menús desplegables, botones, iconos, ventanas emergentes, etc.

## 3.6 Node

Node.js es un entorno JavaScript que nos permite ejecutar en el servidor, de manera asíncrona, con una arquitectura orientada a eventos y basado en el motor V8 de Google.

El motor V8 de Google compila Javascript en código máquina nativo en vez de interpretarlo en el navegador, consiguiendo así una velocidad mucho más alta, además de la alta velocidad de ejecución, Node.js dispone del Bucle de Eventos (Event Loop), que permitirá gestionar enormes cantidades de clientes de forma asíncrona. Tradicionalmente para trabajar de forma asíncrona las aplicaciones se valían de la programación basada en hilos (programming threaded applications), pero esto supone la utilización (normalmente ineficaz) de un espacio de memoria que va escalando a medida que la cantidad de clientes conectados a nuestra aplicación aumenta.

Node.js resuelve este problema cambiando la manera de realizar las conexiones con el servidor. En vez de generar un nuevo hilo E/S para cada cliente, cada conexión dispara la ejecución de un evento dentro del proceso del motor de Node. De este modo, Node permite que un solo servidor que lo ejecute pueda soportar decenas de miles de conexiones. Por lo tanto, si necesitamos gestionar grandes cantidades de conexiones no tendremos que ampliar el número de servidores.

En conclusión, es una plataforma que agiliza y facilita las conexiones cliente servidor por lo que es una tecnología que avanza muy rápidamente y cada vez está más presente en el mercado.

## 3.7 MongoDB

Es una de las bases de datos noSQL y orientada a documentos que existen, esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON. Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

Aunque se suele decir que las bases de datos NoSQL tienen un ámbito de aplicación reducido, MongoDB se puede utilizar en muchos de los proyectos que desarrollamos en la actualidad, dos puntos a tener en cuenta sobre esta base de datos son:

No existen las transacciones: Aunque nuestra aplicación puede utilizar alguna técnica para simular las transacciones, MongoDB no tiene esta capacidad. Solo garantiza operaciones atómicas a nivel de documento.

No existen los JOINS: MongoDB está destinado a proyectos en los que los datos no tengan que ser estructurados en tablas y que tampoco tengamos que hacer relaciones entre ellas.

## 3.8 Python

Python es un lenguaje de scripting desarrollado a principios de los noventa por Guido van Rossum, independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Entre sus muchas características destacamos:

- Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- Interpretado: Quiere decir que no se debe compilar el código antes de su ejecución. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

- Interactivo: Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- Orientado a Objetos: La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.
- Gran cantidad de funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.
- Sintaxis clara: Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

# **Chapter 4**

## **Diseño e implementación**

A continuación se detalla el proyecto de Aplicación de Web Progresiva (PWA) del que trata este trabajo de fin de grado

### **4.1 Arquitectura general**

Este proyecto gira en torno a la arquitectura software del modelo cliente-servidor, como se detalla en la figura 4.1. Destacamos la parte del cliente principalmente manejada por Angular y más concretamente la PWA y por otro lado, la parte del servidor controlada en su punto central por NodeJs, con una base de datos MongoDB y un par de programas desarrollados en python que ayudarán a la automatización, optimización y organización de las salas.

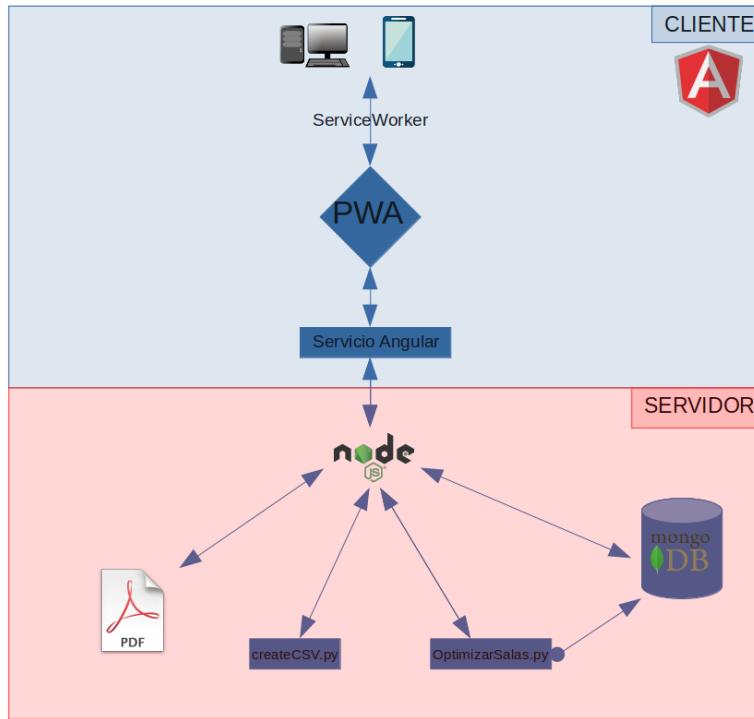


Figure 4.1: Arquitectura general.

### 4.1.1 Cliente

En el lado del cliente tenemos toda la parte desarrollada con Angular, mas concretamente con el paquete `@angular/pwa v.0.1101.1` al tratarse de una PWA y tratar con el framework Bootstrap permite que todas las pantallas de la aplicación sean *responsive*, lo que quiere decir que éstas pantallas se ajustarán automáticamente a la resolución del dispositivo en el que se muestran, centrado principalmente en dispositivos móviles llegándose a asemejar a una app nativa.

Debido al uso de la cámara del dispositivo, los server workers proporcionados por el paquete `@angular/pwa v.0.1101.1` y un login inicial de usuario (usuario y claves) es de obligado cumplimiento que la página se muestre con el protocolo seguro de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol Secure o HTTPS) para garantizar que la transferencia de datos no se verá comprometida en ningún momento y será seguro, esto se consigue con un certificado de clave pública para el servidor web.

En las vistas del cliente diferenciamos dos variantes principales una vez que se hace login, para los usuarios que serán generalmente alumnos y para el administrador de la aplicación, éste último tendrá unas vistas propias y unos privilegios por parte del servidor que no tendrá ningún

otro usuario. Estos privilegios le permitirá modificar elementos de la base de datos, obtener información de todas las charlas, seminarios y de los alumnos apuntados a cada una de ellas.

### 4.1.2 Servidor

En el lado del servidor se realiza el desarrollo back-end al completo, desde la recepción de peticiones HTTPS hasta el acceso a datos o la descarga de ellos en formato CSV o PDF. La parte central de este servidor gira en torno a NodeJs y mas concretamente en su framework Express que nos proporciona los mecanismos para poder manejar las peticiones HTTPS (get, post, put y delete) de los distintos usuarios. Una vez recibida una de estas peticiones, por medio del manejador proporcionado por express realizaremos una u otra acción, entre las principales de login, registro, getEvento entre otros, destacaremos dos:

- **/getCreditoEventos:** Esta petición en concreto se encarga de recibir como parámetros strings el nombre, apellidos y el DNI del usuario, que por medio de la pestaña de *certificado* y más concretamente apretando el botón “*Recibir Créditos*” rellena una plantilla PDF suministrada por la URJC donde se adjudican al alumno los créditos correspondientes por asistir a una de estas charlas o seminarios que previamente se apuntó
- **/zip:** Esta petición solo es posible recibirla cuando es el administrador el que la requiere, se recibe como parámetros dos elementos, el primero el identificador del evento dentro de la base de datos y el nombre del evento. Esta petición lanzará un python llamado createCSV.py que se encarga de crear el CSV con los datos obtenidos de la base de datos y dejarlo en una ruta específica dentro del servidor, donde más tarde se recojerá y se mandara via HTTPS al usuario administrador.

Esta petición en concreto tiene dos caminos:

- Si el administrador quiere un evento en concreto, el id del evento corresponderá a su identificador dentro de la base de datos devolverá un CSV que llenará con los datos obtenidos de la base de datos en función de ese identificador recibido.
- Si el administrador quiere obtener todos los eventos, el id del evento será 0, cuando obtenemos este identificador el servidor creará un .zip que contendrá varios CSV

con los eventos en los que haya más de un alumno inscrito. Estos CSV tendrán el mismo formato y la misma información que el CSV devuelto en el caso anterior.

## 4.2 Modelo de datos

### 4.2.1 MongoDB

Para el almacenamiento de las distintas aulas disponibles, los alumnos inscritos y los horarios de las distintas charlas y seminarios tenemos los siguientes modelos de datos almacenados en la base da datos de MongoDB:

- Eventos: En este Modelo de datos tendriamos almacenados JSON del con el formato siguiente:

Eventos	
aula	string
hora	<i>Horario</i>
nombre_evento	string
descripcion	string
id_evento	integer
id_qr_evento	integer
id_usuarios	[Arr. Usuarios]

Figure 4.2: Modelo Eventos.

- Aula: Nombre correspondiente al aula donde se impartirá la charla o seminario almacenada en el Modelos de datos de *Salas*
- Hora: Hora correspondiente a la que se impartirá la charla o seminario, corresponderá a una hora almacenada en el Modelo de datos de *Horario*
- Nombre\_evento: Nombre correspondiente al Evento que se expondrá a la hora y en el aula especificado en los dos puntos anteriores
- Descripcion: Descripción del evento presentado
- Id\_Evento: Identificador interno del evento presentado

- Id\_qr\_evento: Identificador del QR correspondiente al evento presentado
- Id\_usuarios: Array del mode de datos de *Usuarios* donde se especifica los usuarios apuntados al evento presentado
- Usuarios: En este Modelo de datos tendríamos almacenados JSON del con el formato siguiente:

Usuarios	
nick	string
password	string
id_usuario	integer
logueado	boolean
nombre_apellidos	string
DNI	string

Figure 4.3: Modelo Usuarios.

- Nick: Nombre correspondiente al alumno reconocido internamente en la URJC
- Password: Contraseña encriptada en AES192
- Id\_usuario: Identificador interno del usuario
- Logueado: Indicador true (conectado) o false(desconectado) del usuario
- Nombre\_apellidos: Nombre y apellidos del Usuario
- DNI: DNI del Usuario
- Salas En este Modelo de datos tendríamos almacenados JSON del con el formato siguiente:

Salas	
aula	string
tam	string

Figure 4.4: Modelo Salas.

- Aula: Nombre correspondiente al aula y como se la reconoce internamente en la URJC.
- Tam: Tamaño máximo de la sala.
- Horario En este Modelo de datos tendriamos almacenados JSON del con el formato siguiente:

Horario	
hora	string

Figure 4.5: Modelo Salas.

- Hora: Horas disponibles para todas la aulas en los dias especificados.

#### 4.2.2 Registros de Alumnos y Eventos

Este apartado solo será visible por parte el administrador de la aplicación o del docente encargado en gestionar la semana cultural donde se imparten estas charlas y seminarios. Como vimos anteriormente, el administrador tiene la posibilidad de descargar un .CSV con toda la información de un Evento o de todos ellos, obteniendo un listado de los alumnos que se han apuntado a un determinado evento y que han validado su asistencia por medio de nuestra aplicación. El formato que tienen estos CSV es el siguiente:

	A	B	C
1	Usuario	Id Usuario	Validado
2	dendrina		1 True
3	mperez		53 False
4	jserrano		44 True
5	agarcia		78 True
6	jlopez		56 False

Figure 4.6: Modelo del archivo CSV.

- Usuario: Corresponde a un string almacenado en la base de datos de MongoDB con el formato de modelo de datos de *Usuarios*
- Id\_usuario: String correspondiente campo id\_usuario del modelo de datos de *Usuarios*
- True o False en el campo "validado" hace referencia a si el alumno ha asistido al evento y ha escaneado el código QR disponible al finalizar el mismo dando constancia así que asistió a él, campo a True, o por el contrario está apuntado pero no asistió a él siendo en este caso False el valor de este campo.

### 4.2.3 Plantilla documento de obtención créditos

El pdf que obtiene el alumno es un documento oficial de la Universidad Rey Juan Carlos en el que se le otorga un determinado numero de créditos (fijados por el responsable de los seminarios o charlas) por acudir a las los seminarios o charlas que previamente se apuntó mediante la aplicación de la que estamos tratando en este TFG. La obtención de este certificado se hará de forma automática una vez que el alumno haya acudido a un número mínimo de seminarios fijados por el responsable de las mismas. El formato que tendra el documento para la obtención de los créditos será el siguiente:

 Universidad  
Rey Juan Carlos

**1** **D.Gregorio Robles Martínez**, Subdirector de Investigación, Relaciones Internacionales y Relaciones con la Industria de la Universidad Rey Juan Carlos

CERTIFICA: Que el **2** alumno/a de la Universidad **Rey Juan Carlos**, Don/ña **3** con DNI nº **4** ha participado en el Foro de Empleo de la URJC durante los días **6 y 7 de Marzo de 2020** y ha cumplimentado satisfactoriamente los requisitos exigidos para la obtención de: **5 0,5 CREDITOS ECTS.**

Y para que conste, y a petición del interesado, expido el presente certificado en Madrid, a **de** de **2022**

**6** 

Figure 4.7: Plantilla PDF creditos.

- 1: Nombre del profesor encargado de los seminarios y charlas
- 2: Nombre del alumno al que se le otorgarán los créditos.
- 3: DNI del alumno
- 4: Fechas correspondientes a los días en los que tendrán lugar las charlas o seminarios
- 5: Número de créditos que obtendrá el alumno por asistir a los seminarios
- 6: Firma del profesor o responsable encargado de otorgar los créditos al alumno.

## 4.3 Diseño e implementación por funcionalidad

### 4.3.1 Cliente

#### 4.3.1.1 Login

La aplicación se integra dentro de las aplicaciones de la Universidad Rey Juan Carlos para el uso tanto de profesores como de alumnos, es por ello por que guarda una estética distintiva de la universidad, con el uso de sus logos y colores.

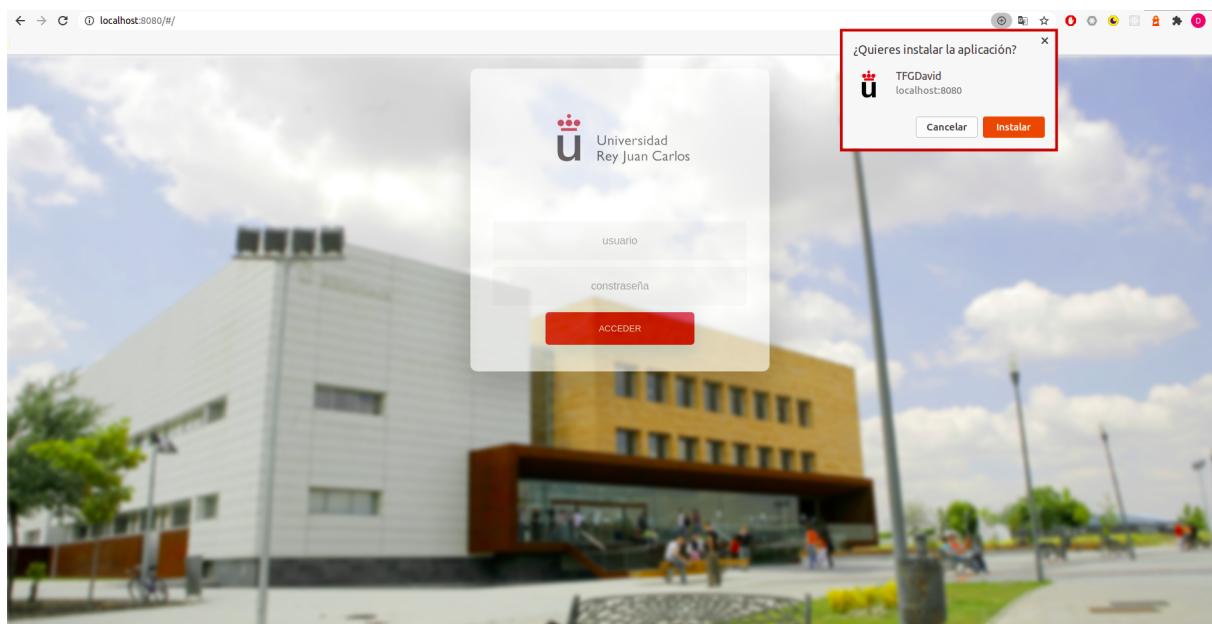


Figure 4.8: Página Login

La página de *login* consta de un menú de login donde el usuario ingresará sus credenciales propias de la Universidad Rey Juan Carlos. En ésta pagina destacamos dos puntos clave de por qué esta aplicación es una PWA. En primer lugar, tal y como se observa en la figura 4.8 el navegador tanto en ordenador como en dispositivos móviles nos da la opción de instalar la aplicación en nuestro dispositivo, creando un acceso directo a ella como si de una aplicación nativa android (en el caso móvil) o como una extensión chrome (en el caso de utilizar un navegador chrome en el ordenador) se tratase. En segundo lugar, como observamos en la figura 5.1 la aplicación dispone también de un *service worker*.

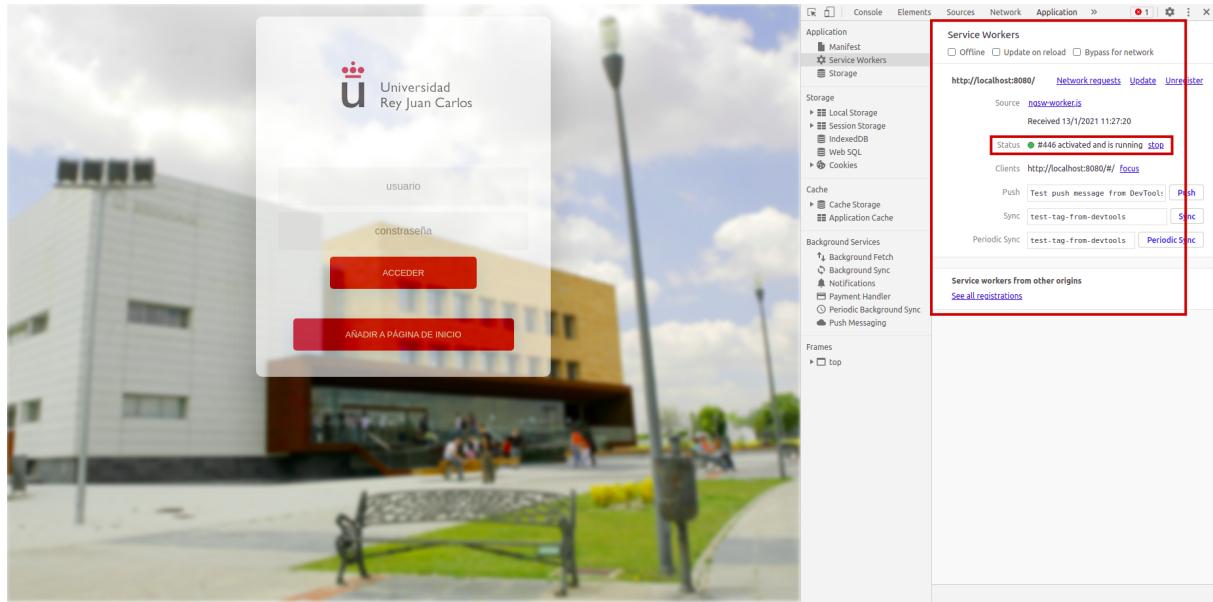


Figure 4.9: Página Login

Un *service worker* es un script que permite interceptar y controlar los requests de la red y el almacenamiento del cache del navegador. Gracias a los *service workers*, los programadores web pueden crear sitios web con una experiencia offline y utilizar la aplicación sin conexión únicamente con las páginas que previamente se hayan guardado en la caché del dispositivo.

Como los *service workers* tienen acceso al manejo de los requests, pueden llegar a ser muy poderosos (y peligrosos), especialmente si son usados con mala intención. Esto es uno de los motivos por los cuales usar *HTTPS* es muy importante, para prevenir ataques de terceros.

#### 4.3.1.2 Usuario

Nada más ingresar en la aplicación con el nombre de usuario y contraseña, cargará la página principal, destacar que durante todas las ventanas de la aplicación dispondremos de un menú, en el que estarán accesibles las 3 páginas principales de nuestra aplicación (1,2,3) y un botón para hacer *logout* (4) de la misma.



Figure 4.10: Menú principal de la aplicación

1. Página principal: En la siguiente vista dispondremos de la información sobre las charlas

y seminarios a los que el alumno se inscribió, en el campo **Título** se describe el título en si de la charla o seminario, en el campo **Aula** se informa del aula dentro de la Universidad Rey Juan Carlos donde tendrá lugar la charla, y por último, el campo **Hora** que especifica la hora a la que tendrá lugar el evento. Dentro de cada fila del evento en el que estamos apuntados y tenemos previsto ir aparecerá dos botones, "Validar" (1) y "Desapuntarse" (2). El botón validar solo estará accesible una vez el evento haya finalizado para que los alumnos que han podido asistir puedan validar mediante la aplicación que han ido a dicha charla. Una vez validado el evento y habiendo recibido la validacion por parte del servidor, debajo del campo aula aparecerá un icono (3) mostrando que la validación se ha desarrollado con normalidad.



The screenshot shows a user interface for managing events. At the top, there is a dark header bar with the university logo, navigation links for 'Página Principal', 'Calendario', and 'Certificado', and a 'Salir (david)' link. Below the header is a table listing three events:

Titulo	Aula	Hora
Telefonica	aula 323 3	11:00-11:30
BBVA	aula 324 3	10:00-10:30
Kabel	aula 321 1 2	10:00-10:30

For the Kabel event, two buttons are visible at the bottom of the row: a green 'Validar' button (labeled 1) and a red 'Desapuntarse' button (labeled 2). To the right of the 'aula 321' entry, there is a small green checkmark icon with the number '3' next to it, indicating successful validation.

Figure 4.11: Página de inicio.

Una vez presionemos el botón de validar ya que hemos asistido al evento en cuestión y queremos dar fe de ello, la aplicacion nos llevará a la pestaña de validación. En esta pestaña, la aplicacion utilizará la cámara del dispositivo, ya sea un móvil, tabley u ordenador portatil para leer el codigo QR correspondiente al evento que queremos validar.

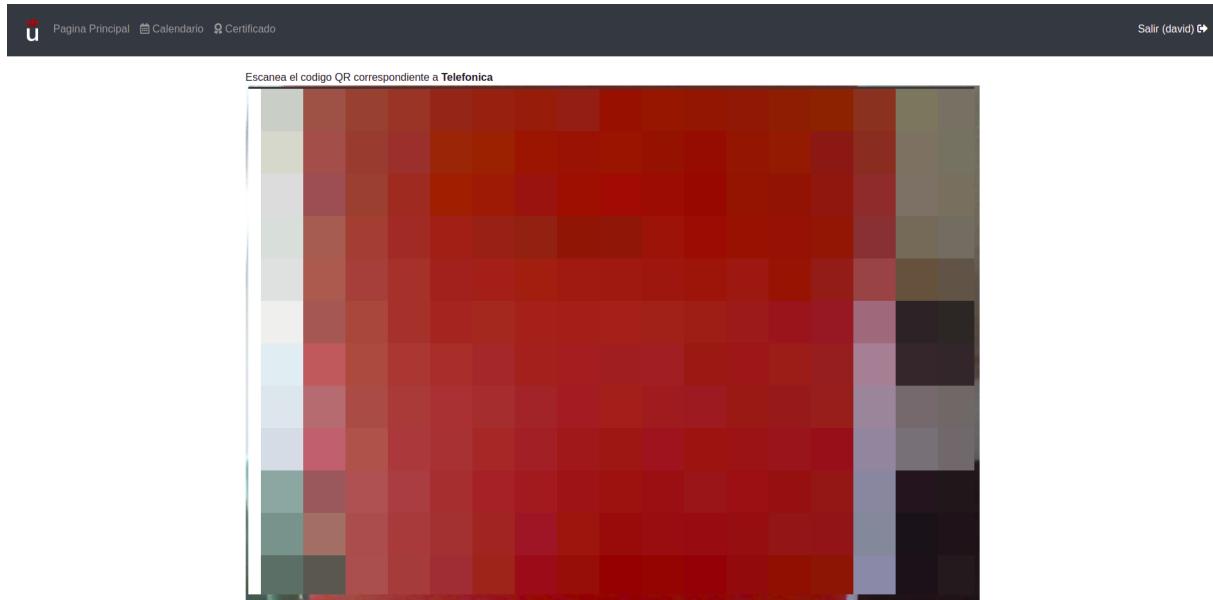


Figure 4.12: Pestaña de validacion de QR

Para validar el evento, basta con mostrar un QR valido y la aplicacion obtendra un mensaje de aceptación por parte del servidor, si por el contrario mostramos un QR que no correspondería al QR del evento en cuestión, se nos avisará con un mensaje en la pantalla que el QR proporcionado no es válido. En este caso escaneamos un QR correspondiente al evento del BBVA cuando debería ser el QR del evento Telefónica.

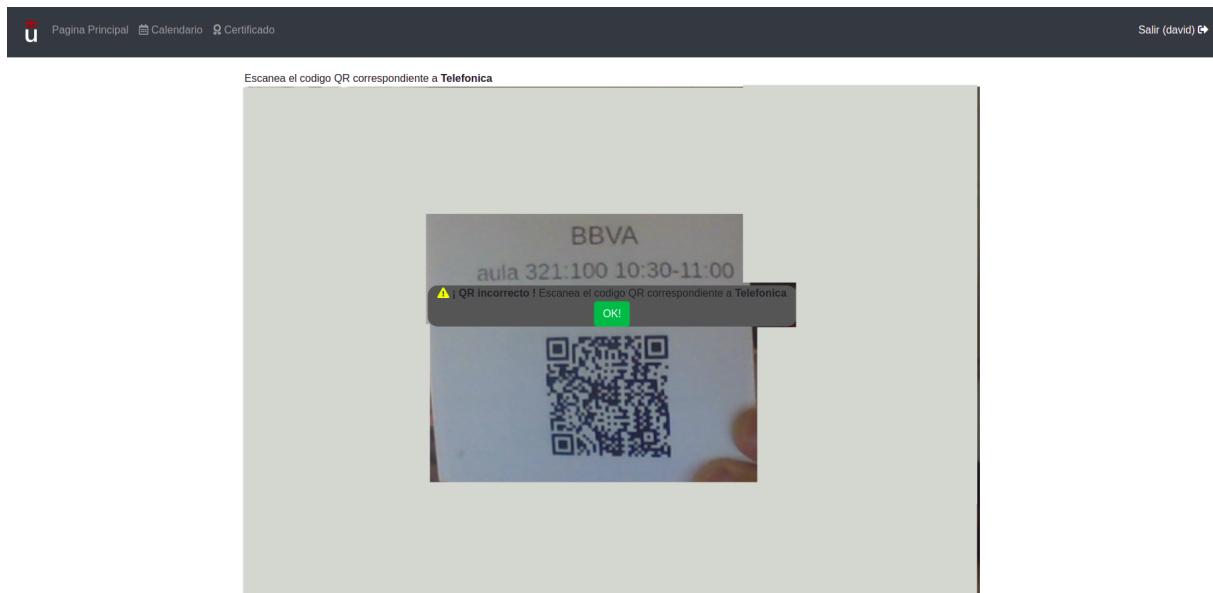


Figure 4.13: Pestaña de validacion de QR con mensaje de error

2. Calendario: En la vista siguiente 4.14 obtenemos un calendario con las principales charlas y seminarios disponibles durante las jornadas habilitadas por la Universidad Rey Juan Carlos.

The screenshot shows a university calendar interface. At the top, there are navigation links: 'Página Principal' (Home), 'Calendario' (Calendar), and 'Certificado'. On the right, it says 'Salir (david)' with a user icon. The main content is a grid-based calendar for four classrooms:

HORAS / AULAS	AULA 323	AULA 321	AULA 322	AULA 324
10:00-10:30	Huawei	Kabel	Thales Alenia Space	BBVA
10:30-11:00	Westcon	IBM	Accenture	
11:00-11:30	Telefonica	Askey-ASUS	SAP	
11:30-12:00	CT Ingenieros	Deloitte		

Figure 4.14: Página de calendario

Cada evento ocupa una posición en el calendario, así mismo cada uno de ellos actúa como un botón, en el que presionándolo nos redirige a la página de información del mismo. En ésta página de información es donde el alumno se podrá apuntarse, dando click en "Apuntarse" y expresar su intención de asistir a él, también hay una pequeña descripción del evento que estamos viendo.

Aula: aula 323 - Hora: 11:00-11:30

La Fundación Telefonica, en colaboración con la Universidad Rey Juan Carlos estará presente en el seminario donde contribuir a la mejora de la actividad profesional de los profesores y el desarrollo de sus alumnos y, por ello, del sistema educativo. Con este evento se busca contribuir al desarrollo y adaptación del sistema educativo a un entorno fuertemente digitalizado y cambiante: formación de los profesores del siglo XXI, para que puedan promover un aprendizaje del siglo XXI con estudiantes del siglo XXI.

El seminario está dirigido a todas aquellas personas interesadas en conocer las posibilidades que ofrecen las herramientas digitales en el aula. En la sesiones se intercalarán ponencias y talleres participativos llevados a cabo por referentes nacionales en el ámbito de la educación y la tecnología.

[Apuntarse](#)

Figure 4.15: Página de descripción del evento y botón de apuntarse

3. Certificado: Esta vista se compone de un botón que el alumno tendrá disponible únicamente cuando pueda tener la posibilidad de obtener los créditos que le corresponden por asistir a estos seminarios, el requisito de mínimo de asistencia para poder obtener el certificado de los créditos lo impondrá el profesor encargado de ello o en su defecto el administrador que se encargue del mantenimiento de la página y de dichas jornadas propuestas aquí.

En ésta página podrás obtener el certificado que tendrás que subir a la plataforma de reconocimiento académico de créditos para así poder recibir los 0.5 ETCS correspondientes tras haber asistido a un mínimo de 3 seminarios impartidos durante la semana

[Recibir Créditos](#)

Figure 4.16: Página para obtener el certificado de créditos

Una vez el alumno tiene disponible el botón de "Recibir Créditos" un pop-up saldrá en pantalla sirviendo una vista previa del pdf que será el certificado que le otorgue los créditos y que el alumno obtendrá para poder justificar los mismos. Estará a su vez disponible un botón de descarga del pdf para poder almacenarlo en su dispositivo.

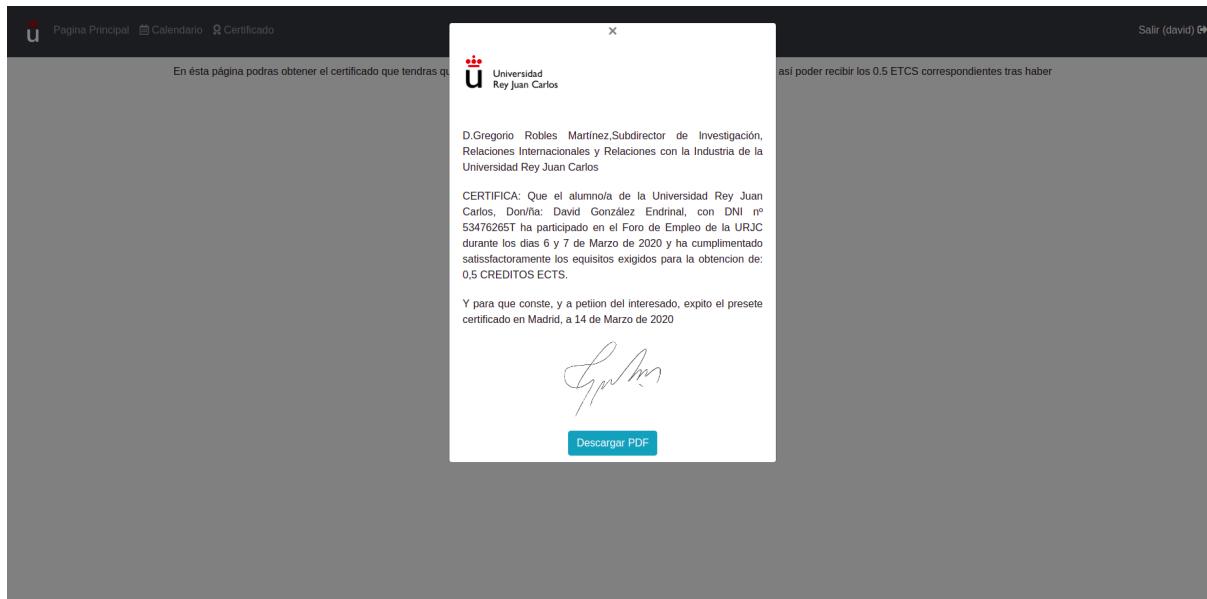
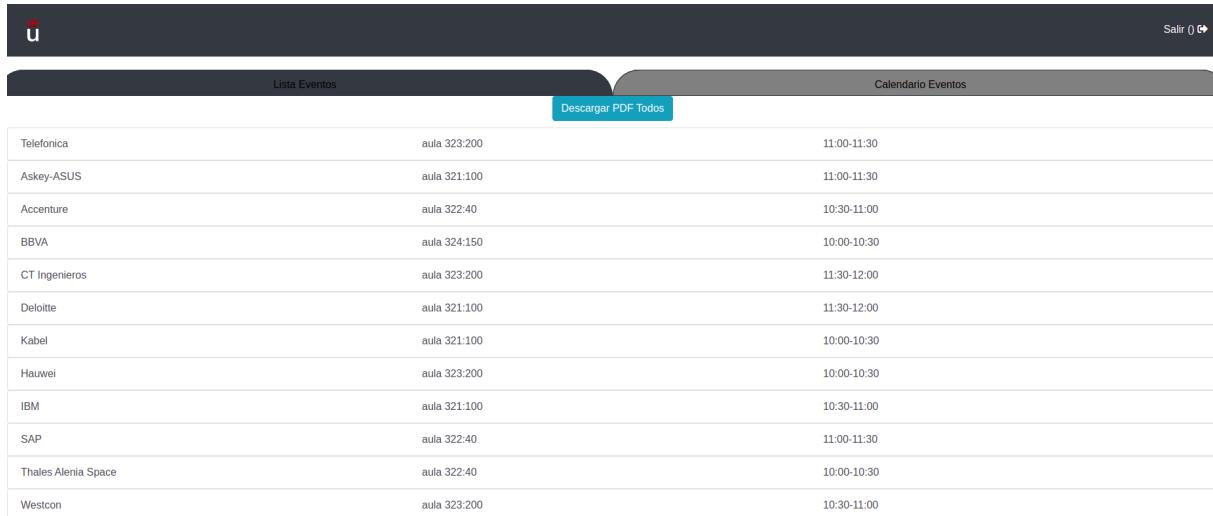


Figure 4.17: Pop-up con la vista del certificado

#### 4.3.1.3 Administrador

Una vez iniciado sesión con la cuenta del administrador o de la persona encargada de mantener y administrar la aplicación entraremos en la misma donde veremos dos pestañas claramente diferenciadas:

1. Lista de Eventos: En esta pestaña el administrador de la aplicación tendrá un listado completo de todos los eventos o seminarios que habrá durante la semana o los días que duren cada uno de ellos. En ésta primera vista se dará la información de: Nombre del evento, aula donde se impartirá y la hora del día a la que tendrá lugar dicho evento.



The screenshot shows a web application interface for managing events. At the top, there's a dark header bar with a logo on the left and a 'Salir' button on the right. Below it, a navigation bar has two tabs: 'Lista Eventos' on the left and 'Calendario Eventos' on the right. A blue button labeled 'Descargar PDF Todos' is positioned between the tabs. The main content area contains a table with the following data:

Telefonica	aula 323:200	11:00-11:30
Askey-ASUS	aula 321:100	11:00-11:30
Accenture	aula 322:40	10:30-11:00
BBVA	aula 324:150	10:00-10:30
CT Ingenieros	aula 323:200	11:30-12:00
Deloitte	aula 321:100	11:30-12:00
Kabel	aula 321:100	10:00-10:30
Hauwei	aula 323:200	10:00-10:30
IBM	aula 321:100	10:30-11:00
SAP	aula 322:40	11:00-11:30
Thales Alenia Space	aula 322:40	10:00-10:30
Westcon	aula 323:200	10:30-11:00

Figure 4.18: Vista de administrador de la pestaña Lista de Eventos

Destacamos en esta vista el botón de "*Descargar PDF Todos*", presionandolo eviaremos una petición al servidor que nos contestará con un .ZIP llamado "Eventos.zip" que contendrá todos los CSV de los Eventos en los que haya apuntada mas de una persona. Este CSV es del mismo formato que el CSV que se descargaría dando uno por uno a cada evento con el botón "Generar CSV" que explicaremos más adelante en el siguiente punto. El .ZIP descargado quedaría de la siguiente manera:

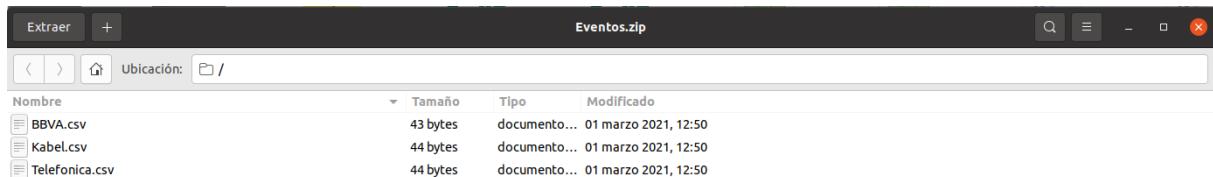


Figure 4.19: Contenido de Eventos.zip descargado al presionar "Descargar PDF Todos"

Haciendo click en cada una de las filas correspondientes a cada uno de los eventos entraremos en una ventana donde nos informará más detalladamente sobre el evento en cuestión, con una breve descripción del evento (1), su QR de validación correspondiente (2) y el listado de los usuarios inscritos a él, así como que usuario ha validado o no la asistencia al evento con el campo "*Validado*" true o false (3). Destacamos de ésta vista el botón "*Generar CSV*" (4) que descargará en nuestro equipo en formato CSV el mismo listado de usuarios que estamos viendo en esta misma página.



Figure 4.20: Vista de administrador de la ventana informativa del evento de Telefónica

2. Calendario Eventos: En esta pestaña el administrador tendrá de forma visual la información de cuando y donde va a tener lugar un evento cualquiera, seguido del nombre del evento, entre parentesis, hay un numero correspondiente al numero de alumnos apuntados a dicho evento.

Destacamos varias cosas en esta pestaña, en primer lugar el botón "*Optimizar*", éste botón envia un mensaje al servidor que lanzará un python de optimización para reubicar los eventos en horarios y aulas de la forma más óptima posible, más adelante, en el siguiente punto, cuando se explique la parte correspondiente al servidor desarrollaré más su funcionamiento. Este botón modificará la posición de todos o la gran mayoría de los eventos dentro del marco horario mostrado en esta pestaña.

Por otro lado, es posible modificar de forma individual un evento, para ello bastará con hacer click sobre él, toda su información se volcará en los desplegables que se muestran en la parte inferior de la imagen (1), se podrá modificar tanto el aula como las horas que estén disponibles para todos los eventos, en este caso entre las aulas 321,321, 322 y 324 y en los horarios de 10:00 a 12:00.

Presionando el botón "*Modificar*" las modificaciones que hayamos hecho en los desplegables serán enviadas al servidor y podremos verlas en esta misma vista.

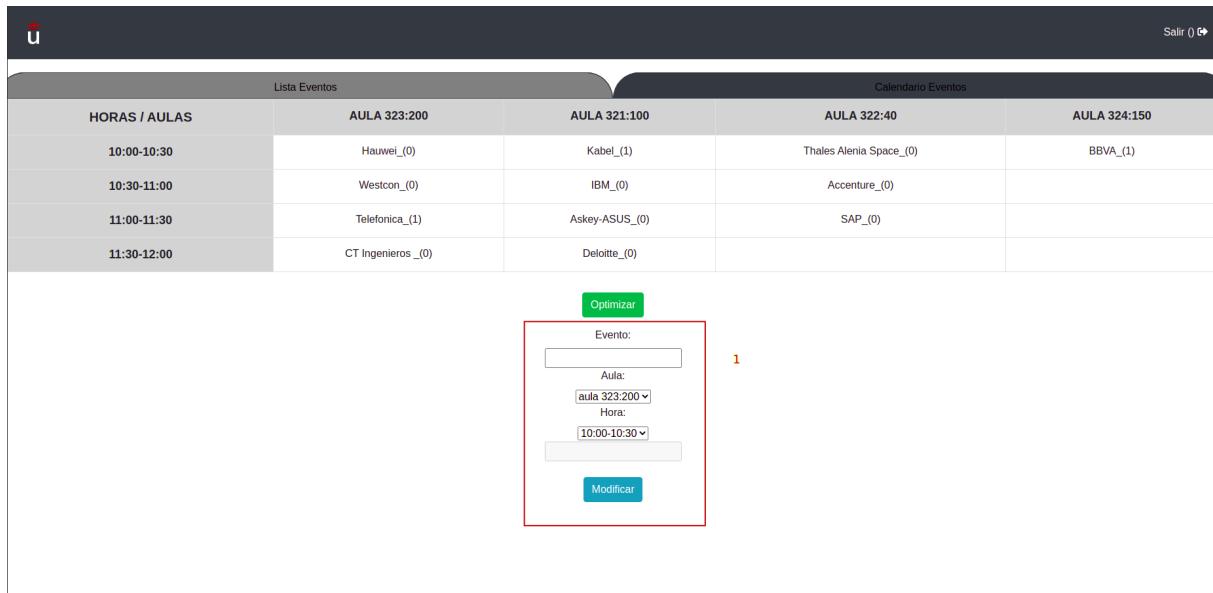


Figure 4.21: Pestaña Calendario Eventos

### 4.3.2 Servidor

El servidor está desarrollado en un entorno de trabajo que funciona en tiempo de ejecucion, de código abierto y multiplataforma, este entorno es *Node*. Añade un soporte para APIs donde se incluye una comunicacion HTTP entre el cliente y él mismo. Por todo esto obtenemos un gran rendimiento de la aplicación y tambien una gran escalabilidad. Dentro de Node el framework mas usado y que uso en este caso es **Express** cuya libreria se compone de un gran numero de framework populares dando mecanismos para el manejo de peticiones HTTP y HTTPS (en mi caso) y mantener así una comunicacion segura cliente-servidor. Los métodos de comunicacion usados en el servidor mediante HTTPs son los típicos usados en este tipo de aplicaciones:

- GET: El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
- POST: El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
- PUT: El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
- DELETE: El método DELETE borra un recurso en específico.

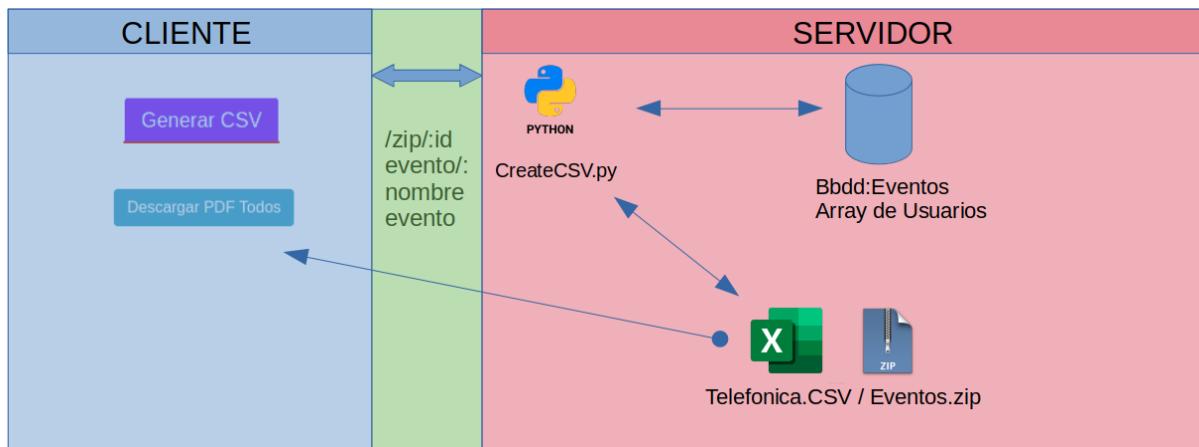
Al margen ya de la parte Node del servidor, destacamos dos programas desarrollados en python de los que el servidor hace uso cuando el cliente así lo requiere, destacamos que el único cliente o usuario que puede realizar una petición al servidor y poder lanzar así estos programas es el usuario bajo el nick de "admin", en los siguientes puntos pasaremos a explicar el funcionamiento de dichos programas:

1. CreateCSV.py: Cuando el administrador presiona dentro de la pestaña "Lista Eventos" cualquier fila correspondiente a un evento, y dentro de ella el botón "GenerarCSV" o bien en la misma pestaña de "Lista Eventos" presiona el botón "Generar PDF Todos", el servidor dentro de la url '/zip/:id\_evento/:nombre\_evento' recibe la petición del cliente que quiere obtener o bien un CSV o bien un .zip con todos los CSV de los eventos dentro de él. Desde node se lanza el python mediante el comando ./createCSV.py -idevento (el parámetro idevento solo se pondrá en el caso en el que se quiera el CSV de un evento en concreto, para obtener el .zip de todos el parámetro se omitirá).

Una vez el programa en python de createCSV.py ha sido lanzado, pasa a buscar en la base de datos mediante un select de idevento o recorriendose todos los eventos que haya obteniendo así una lista de usuarios asociados al evento con el campo validado true o false según corresponda para cada uno de los usuarios.

Cuando se obtiene esta lista y se finaliza la ejecución del select, pasamos a volcar los datos obtenidos en un fichero .CSV, cuando terminamos de llenar el fichero, en el caso de querer uno en concreto la ejecución terminaría aquí y sería devuelto el fichero al cliente que solicitó tal acción al servidor, por otro lado, en el caso de querer todos los eventos se irá completando la acción pasada con todos los eventos en los que haya más de una persona inscrita en él, y cuando termine pasaría a crear un .zip con todos estos .CSV creados. Una vez terminado, como en el caso anterior, le será devuelto al cliente un .zip como requirió con su petición inicial.

La forma en la que el servidor y el cliente están conectados para poder enviar la información en formato .CSV o en formato .ZIP es mediante un *pipe stream*.



captionEsquema funcionamiento createCSV.py

2. optimizarSalas.py: El siguiente programa se ejecuta cuando el administrador dentro de la pestaña "Calendario Eventos" presiona el botón "Optimizar", este botón lanza un mensaje a la ruta '/optimizar' dentro del endpoint donde está escuchando el servidor, que ejecuta el comando './optimizarSalas.py' lanzando el programa en python.

Este programa tiene como objetivo la optimización y reubicación de los eventos en función de los huecos libres que quedan con respecto a la gente que se ha apuntado al evento o quiere asistir y con los huecos libres que quedan en la sala. Funciona de la siguiente manera:

El programa, una vez se ejecuta, lee de la base de datos cada uno de los eventos que están programados que se van a llevar a cabo en los días que transcurran todos ellos, de cada evento obtiene el número de alumnos o personas que están apuntados a él, con este número calcula los huecos libres que tienen en todas las aulas disponibles, desechar los números negativos que se obtienen cuando van a asistir más personas que huecos libres tiene la sala.

Una vez tenemos relacionados todos los eventos con las aulas en función de los huecos libres que hay, el diccionario se ordena de menor a mayor, teniendo prioridad de posición los eventos que tengan menor número de huecos libres en una sala determinada.

La adjudicación del evento a una sala se va realizando de forma ascendente y ordenada hasta que la aula en cuestión complete todas las horas disponibles que tiene dentro de los días que están programados que tengan lugar estos eventos. En nuestro caso, se

completara cuando en una sala se haya completado el horario de 10:00 a 12:00, cuando esto ocurre el evento no tendrá la opción de ser colocado en esa aula y pasará a la siguiente aula donde en función del tamaño, las personas que vayan y el número de huecos disponibles sea el más óptimo.

Si se llega a el caso en el que un evento no encuentra lugar dentro de ningún aula, este evento quedará excluido y no se le adjudicará ningun aula. El evento se mostrara fuera de la matriz de "Calendario Eventos" y tendrá que ser el administrador quien lo posicione a mano, dentro de un aula en concreto teniendo en cuenta la limitación de aforo mediante el botón "Modificar" que se encuentra en la parte inferior de la pestaña de "Calendario Eventos" y que ya explicamos su funcionamiento en el apartado anterior.

# **Chapter 5**

## **Experimentos y validación**

El principal objetivo sobre el que gira este proyecto de fin de grado es el de crear una aplicación web híbrida que funcione tanto en dispositivos móviles como en navegadores web, esto lo hemos conseguido creando una Aplicación Web Progresiva o PWA.

Para ver que realmente hemos conseguido que nuestra aplicación sea una PWA basta con realizar las siguientes pruebas:

### **5.1 Server Workers**

Nuestra aplicación contiene *server workers* que son propios del paquete `@angular/pwa v.0.1101.1` para ello basta con mirar en la consola de desarrolladores de Google Chrome o de cualquier navegador, ir al apartado service workers y ver que está activado y corriendo figura 5.1

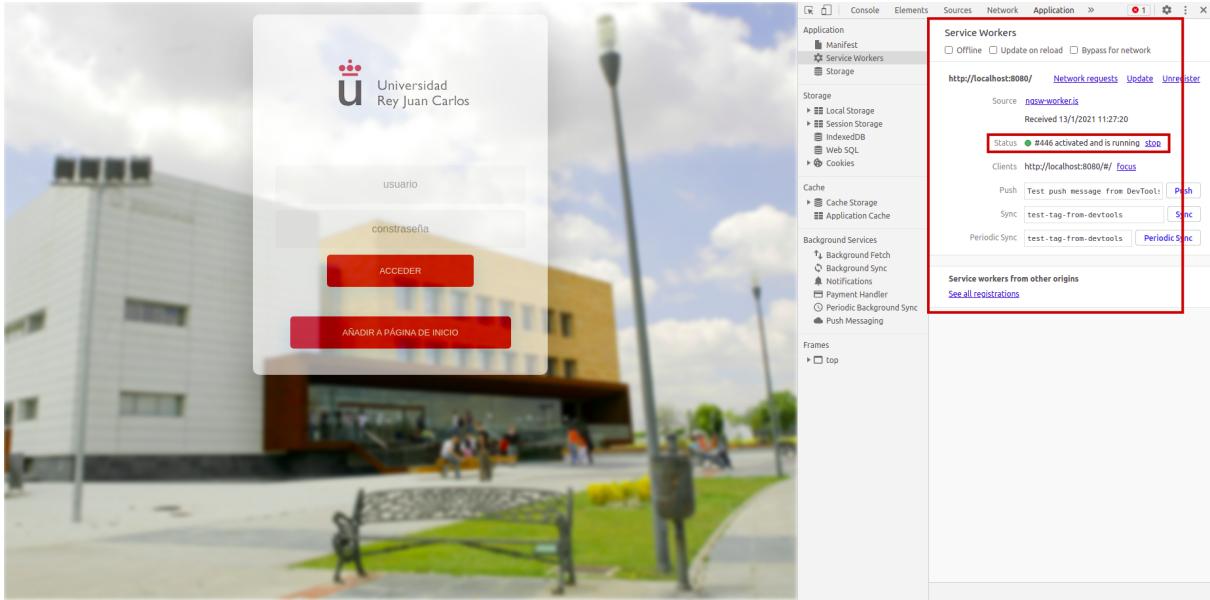


Figure 5.1: Service Workers funcionando en la PWA

## 5.2 Instalación en un dispositivo móvil

Podemos instalar nuestra aplicación en un dispositivo móvil como si de una aplicación nativa, android o IOS, se tratase. Para ello presionamos el botón "Añadir a página de inicio" una vez que estemos navegando con el dispositivo móvil en cuestión por nuestra aplicación. Una vez hayamos aceptado y se haya instalado, nuestra aplicación PWA se verá como una aplicación nativa dentro de nuestro dispositivo móvil figura 5.2

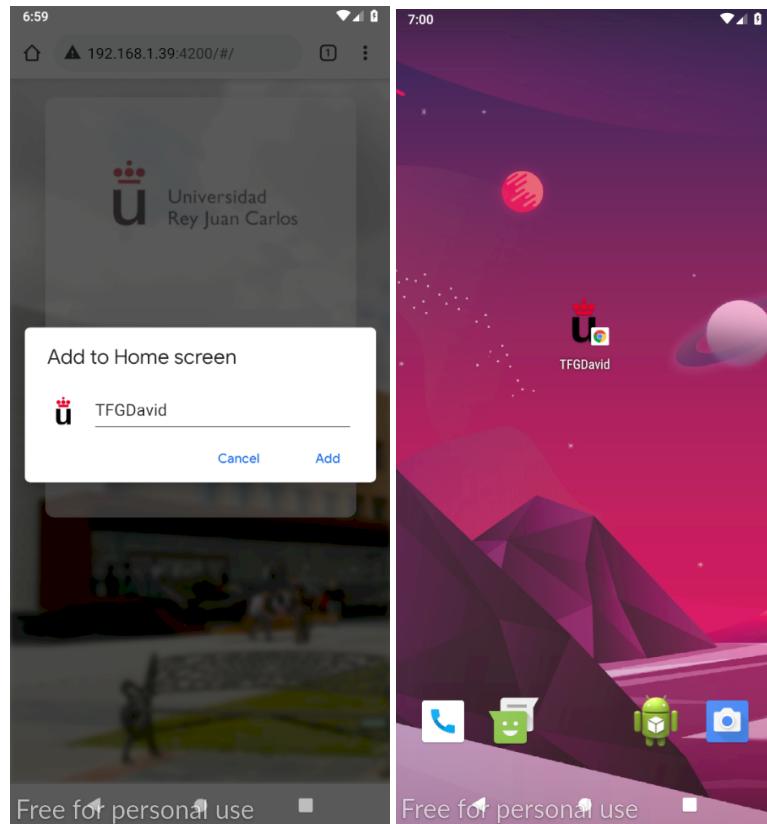


Figure 5.2: PWA en la pantalla de inicio del dispositivo móvil

## 5.3 Optimización

En cuanto a la parte del servidor, un punto importante a tener en cuenta sobre la validación del proyecto y el buen funcionamiento de éste es que el python encargado de optimizar las salas en función al número de alumnos apuntados y al número de huecos disponibles funcione correctamente. Para ello podemos lanzar manualmente el propio python encargado de ello con el comando `python optimizarSalas.py` obtendríamos un resultado como vemos a continuación:

```
david@davidUbuntu:/Escritorio/TFG/backEnd$ python optimizarSalas.py
...
{'Kabel': 86, 'Westcon': 94, 'IBM': 134, 'Huawei': 74, 'BBVA': 0, 'Askey-ASUS': 4, 'Thales Alenia Space': 109, 'Deloitte': 188, 'Telefonica': 139, 'Accenture': 7, 'CT Ingenieros ': 68, 'SAP': 191}
10:00-10:30
Al evento Kabel van (86) personas que va a al aula aula 321 a la hora 10:00-10:30 con 14 huecos libres
-----
10:30-11:00
Al evento Westcon van (94) personas que va a al aula aula 321 a la hora 10:30-11:00 con 6 huecos libres
-----
10:00-10:30
Al evento IBM van (134) personas que va a al aula aula 324 a la hora 10:00-10:30 con 16 huecos libres
-----
11:00-11:30
Al evento Huawei van (74) personas que va a al aula aula 321 a la hora 11:00-11:30 con 26 huecos libres
-----
10:00-10:30
Al evento BBVA van (0) personas que va a al aula aula 322 a la hora 10:00-10:30 con 40 huecos libres
-----
10:30-11:00
Al evento Askey-ASUS van (4) personas que va a al aula aula 322 a la hora 10:30-11:00 con 36 huecos libres
-----
10:30-11:00
Al evento Thales Alenia Space van (109) personas que va a al aula aula 324 a la hora 10:30-11:00 con 41 huecos libres
-----
11:00-11:30
Al evento Deloitte van (188) personas que va a al aula aula 323 a la hora 10:00-10:30 con 12 huecos libres
-----
11:00-11:30
Al evento Telefonica van (139) personas que va a al aula aula 324 a la hora 11:00-11:30 con 11 huecos libres
-----
11:00-11:30
Al evento Accenture van (7) personas que va a al aula aula 322 a la hora 11:00-11:30 con 33 huecos libres
-----
11:30-12:00
Al evento CT Ingenieros van (68) personas que va a al aula aula 321 a la hora 11:30-12:00 con 32 huecos libres
-----
10:30-11:00
Al evento SAP van (191) personas que va a al aula aula 323 a la hora 10:30-11:00 con 9 huecos libres
-----
Actualizo en la tabla: Kabel a la hora 10:00-10:30 con aula 321:100
Actualizo en la tabla: Westcon a la hora 10:30-11:00 con aula 321:100
Actualizo en la tabla: IBM a la hora 10:00-10:30 con aula 324:150
Actualizo en la tabla: Huawei a la hora 11:00-11:30 con aula 321:100
Actualizo en la tabla: BBVA a la hora 10:00-10:30 con aula 322:40
Actualizo en la tabla: Askey-ASUS a la hora 10:30-11:00 con aula 322:40
Actualizo en la tabla: Thales Alenia Space a la hora 10:30-11:00 con aula 324:150
Actualizo en la tabla: CT Ingenieros a la hora 11:30-12:00 con aula 321:100
Actualizo en la tabla: Telefonica a la hora 11:00-11:30 con aula 324:150
Actualizo en la tabla: Accenture a la hora 11:00-11:30 con aula 322:40
Actualizo en la tabla: Deloitte a la hora 10:00-10:30 con aula 323:200
Actualizo en la tabla: SAP a la hora 10:30-11:00 con aula 323:200
```

Figure 5.3: Salida tras ejecutar python optimizarSalas.py

Revisando dichos resultados nos damos cuenta que el programa tiene en cuenta los huecos libres de las aulas, el tamaño de la misma y los usuarios inscritos optimizando su distribución.

## 5.4 Seguridad

Cabe destacar el uso de HTTPS, (HyperText Transfer Protocol Secure, Protocolo de transferencia de hipertexto) es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web. En este caso al usar partes vulnerables del móvil, como puede ser la cámara al leer el código QR o en el momento de autenticar al usuario, es necesario el uso de una capa más de seguridad.

En el caso de la autenticación la contraseña del usuario, viaja cifrada dentro de este protocolo y dentro de la base de datos permanece cifrada también por si ésta base de datos sufre un ataque no se vea comprometida la seguridad de los usuarios de la aplicación. El cifrado de la contraseña es del tipo AES192 (Advanced Encryption Standard) y para validar este cifrado basta con ver la base de datos y el valor del campo "password" dentro de la tabla "Usuarios", en este caso para el usuario *david* el valor del campo es: *4232a5e307a998a3ef2601c7d0a4c704*, un valor esperado para este tipo de cifrados.

# **Chapter 6**

## **Resultados**

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.



# Chapter 7

## Conclusiones

### 7.1 Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

### 7.2 Aplicación de lo aprendido

En cuanto a conocimientos y habilidades obtenidas durante todo el Grado de Ingeniería Telemática son importantes todos los relacionados con la programación, tanto front como back, destacando los siguientes lenguajes:

1. Todo lo relacionado con Web (HTML, CSS y Javascript) en la Asignatura de *Aplicaciones Telemáticas* con los profesores Jesús M. González Barahona y Gregorio Robles Martínez.

2. Seguridad Web con la asignatura *Seguridad en redes de Ordenadores* con el profesor Enrique Soriano.
3. Parte back con el lenguaje de programación Python en la asignatura de *Laboratorio de Administración y Gestión de Redes y Sistemas* con el profesor Miguel Ortúñoz.
4. Por último destacaría todo mi paso general por la carrera ya que de una u otra manera he ido adquiriendo conceptos, conocimientos, habilidades y el manejo de distintas herramientas en todas las asignaturas en mayor o menor medida.

### 7.3 Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

### 7.4 Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

# Apéndice A

## Manual de usuario

- Front: Dentro de `/TFGDavid/` ejecutamos el proyecto de Angular: **ng serve -ssl** (si queremos que corra bajo HTTPS) o `ng serve` en HTTP normal, destacamos que ambos comandos permiten ejecutar el proyecto en angular en modo "desarrollo" en `localhost:4200`, si queremos ejecutarlo en producción, como se ejecutaría en un servidor final para verlo en internet, tenemos que realizar dos procedimientos, uno ejecutar **ng build -prod**, compilará la aplicación Angular en un directorio de salida llamado `dist/` y por otro lado, para ejecutar el proyecto compilado en producción usaremos el comando:

```
http-server -p 8080 -c-1 ~/dist/TFGDavid
```

- Back: Dentro de `/backEnd/` para lanzar el servidor:

```
node app.js
```

En el caso de querer lanzar el python de `optimizarSalas` o `createCSV`

```
python optimizarSalas.py o python createCSV.py
```

- Base de datos MongoDB: Se deberá ejecutar y mantenerse escuchando en `localhost:27017` dirección que por defecto utiliza esta base de datos.

```
sudo systemctl start mongod
```

Para comprobar que está corriendo el proceso podemos usar el comando:

```
sudo systemctl status mongod
```

