Universidad
de La Laguna

# Sistema centralizado de gestión de usuarios para Innova7.

*Centralized user management system for Innova7.*

Raúl Marrero Rodríguez

Dpto. de Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática

Trabajo de Fin de Grado

La Laguna, 10 de junio de 2014

D. **Jesús Miguel Torres Jorge**, con N.I.F. 43.826.207-Y Profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática de la Universidad de La Laguna

**C E R T I F I C A**

Que la presente memoria titulada:

"*Sistema centralizado de gestión de usuarios para Innova7.*"

ha sido realizada bajo su dirección por D. **Raúl Marrero Rodríguez**, con N.I.F. 79.060.363-W.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2014

# Agradecimientos

Quería agradecer al equipo informático de la asociación Innova7 por haberme brindado la oportunidad de participar en un proyecto tan ambicioso.
También dar gracias tanto a Gara Miranda Valladares como a Alberto Francisco Hamilton Castro por su magnífica labor en la coordinación de la asignatura "Trabajo Fin de Grado".
Por último, agradecer a mi familia y a mi novia, que me han apoyado durante el desarrollo del proyecto y de mi progreso académico en general.

**Resumen**

*El objetivo de este trabajo ha sido la creación de un sistema robusto de gestión de usuarios para la asociación sin ánimo de lucro "Innova 7". Además, se ha tenido que crear una aplicación que haga uso de este sistema para comprobar su robustez y escalabilidad.*

*Ya que este sistema va a ser utilizado de forma real por miles de usuarios, las fases de preparación y documentación son críticas en el proyecto. Serán necesarias reuniones previas al desarrollo así como un tiempo de prueba prudencial antes de que el proyecto vea la luz, para evitar posibles fallos o inconvenientes cuando estén en producción.*

**Palabras clave:** Sistema centralizado, usuarios, estructura, organización, innova7, aplicación, web, sso, autenticación, autorización.

**Abstract**

The main goal of this project has been the development of a strong user system manager for "Innova7" association. It also covers the creation of an application to test the reliability of the main structure. This data base must be scalable and robust.

The core of the system is going to be used in production as a real user community. It requires a critical documentation and preparation phase before the development. It is also essential to have time for testing at the end of the process in order to avoid future problems in real time.

***Keywords:*** *Central system, users, structure, organization, innova7, application, web, sso, authentication, authorization.*

# Índice general


**1. Introducción** **1**
- 1.1. Necesidades de la asociación . . . 1
- 1.2. Planteamiento del problema . . . 1
- 1.3. Estado actual . . . 1
- 1.4. Requisitos . . . 2

**2. Toma de Contacto** **3**
- 2.1. Estado del arte . . . 3
- 2.2. Reuniones preliminares . . . 5
- 2.3. Propuestas . . . 5
  - 2.3.1. LDAP y CAS . . . 5
  - 2.3.2. OpenID y OAuth2 . . . 6
  - 2.3.3. Desarrollo Node.js . . . 7
  - 2.3.4. API interna . . . 7

**3. Decisiones** **9**
- 3.1. Herramientas a utilizar . . . 9
- 3.2. Metodología . . . 9
- 3.3. Desarrollo . . . 10

**4. Planteamiento de la aplicación** **11**
- 4.1. Documentación . . . 11
- 4.2. Requisitos de la aplicación . . . 12
- 4.3. Especificación de la aplicación . . . 12

**5. Desarrollo de la aplicación** **14**
- 5.1. Herramientas utilizadas . . . 14
- 5.2. Desarrollo . . . 15
- 5.3. Funcionamiento de la app . . . 16
- 5.4. Tareas y funcionalidades . . . 16
- 5.5. Administración de los datos . . . 18
- 5.6. Integración con redes sociales . . . 18
- 5.7. Mejoras . . . 19

**6. Testeo en pre-producción** **21**
6.1. Migración al servidor . . . . . 21
6.2. Problemas . . . . . 22
6.3. Tiempos de respuesta . . . . . 22

**7. Conclusiones y Trabajos Futuros** **24**
7.1. Problemas generales . . . . . 24
7.2. Trabajos futuros . . . . . 25
7.2.1. Mejoras en el sistema . . . . . 25
7.2.2. Mejoras en la aplicación . . . . . 25
7.2.3. Mantenimiento . . . . . 26

**8. Summary and Conclusions** **27**
8.1. General problems . . . . . 27
8.2. Future work . . . . . 28
8.2.1. System improvements . . . . . 28
8.2.2. Application improvements . . . . . 28
8.2.3. Maintenance . . . . . 28

**9. Presupuesto** **30**
9.1. Tabla . . . . . 30

**Bibliografía** **31**

# Índice de figuras

1.1. Sistema actual y objetivo propuesto . . . . . 2

2.1. Propuesta API . . . . . 8

5.1. Funcionamiento de la app . . . . . 16
5.2. Captura de la app (menú) . . . . . 17
5.3. WebSocket Pusher . . . . . 20
5.4. Atmosphere Framework . . . . . 20

6.1. Esquema de la base de datos . . . . . 21

# Índice de tablas

2.1. Tabla de sistemas SSO . . . . . 5

6.1. Tabla de tiempos . . . . . 22

9.1. Tabla de presupuesto . . . . . 30

# Capítulo 1

# Introducción

## 1.1. Necesidades de la asociación

Innova7 es una asociación sin ánimo de lucro destinada al fomento del uso y la aplicación de nuevas tecnologías. Creada en 2007 por jóvenes canarios con intereses profesionales y personales centrados en el mundo de las Tecnologías de la Información y la Comunicación.

En los últimos años, dicha organización ha participado en la creación y fomento de eventos multidisciplinares con miles de asistentes. Durante este tiempo, se ha creado una base de datos de usuarios con intereses y aficiones comunes que ha ido creciendo de manera exponencial. Se hace por tanto indispensable el control de estos usuarios y la correcta gestión de los mismos para continuar con la expansión de dicha base de datos.

## 1.2. Planteamiento del problema

Después de lo introducido en el apartado anterior, se plantea la necesidad de definir muy bien las herramientas que se van a utilizar para el desarrollo de esta estructura de base de datos. Existen muchos proyectos similares y una infinidad de diseños que podrían ser utilizados para lograr el objetivo de la asociación.

Además, al tratarse de una asociación sin ánimo de lucro, sus miembros carecen de la disponibilidad necesaria para trabajar diariamente en el proyecto. Por lo tanto, se ha de buscar una solución que sea rápida, no requiera demasiadas horas de trabajo y cumpla con los requisitos que se especificarán en el punto siguiente.

## 1.3. Estado actual

Hoy en día, la organización cuenta con diferentes bases de datos dependiendo de su utilidad. Cada año, con la realización del evento más grande al que dan soporte, "TLP Tenerife", se genera una nueva base de datos de usuarios que deben registrarse para realizar la inscripción a dicho evento. Esto hace que los propios usuarios deban cumplimentar formularios una y otra vez rellenando la misma información.

Ante la previsible expansión generalizada de la asociación, se plantea un cambio de

paradigma en el que se utilice un único sistema de autentificación/registro para dar cabida a las múltiples aplicaciones de las que se dispone, así como futuros servicios que se encuentran en desarrollo.

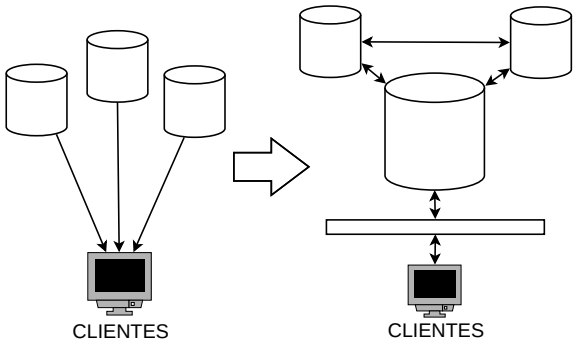


Figura 1.1: Sistema actual y objetivo propuesto

## 1.4. Requisitos

El principal objetivo a cumplir es que el sistema ha de ser lo más modular posible. Debe poder soportar un gran número de usuarios accediendo a su información, dando de alta nuevos perfiles o modificando los propios datos sin que el resto de aplicaciones se vean bloqueadas.

El hecho de que el sistema sirva para la futura implementación de nuevas funcionalidades hace que su planificación inicial sea crítica, ya que la escalabilidad es clave para evitar futuras rectificaciones sobre el código principal.

Además, será necesario dejar todo bien documentado desde la primera fase del proyecto, ya que se trata de un equipo de personas que trabajarán conjuntamente en un futuro ampliando y haciendo uso de este sistema.

Una vez el sistema esté funcionando, se desarrollará una aplicación innovadora que da servicios a los usuarios y clientes de la entidad. Podrá ser desarrollada en cualquier lenguaje y plataforma mientras cumpla los objetivos propuestos. El aplicativo será la prueba de fuego de todo el proyecto. Debe ser lo suficientemente complejo para ser tomado en cuenta como testeo real. Será un ejemplo claro de un programa externo completamente nuevo que hará uso de datos alojados en el sistema central.

Por tanto, como conclusión inicial, se plantea la creación de un sistema “Single Sign On” muy similar al que utiliza Google para acceder a sus distintas aplicaciones (YouTube, Drive, Gmail, etc), donde el usuario solamente tenga que hacer LogIn una vez, para tener a su disposición diferentes aplicativos que, en principio, no tienen por qué ser similares en funcionamiento y pueden estar alojados en diferentes servidores.

# Capítulo 2

# Toma de Contacto

En el capítulo anterior se ha introducido el planteamiento del problema y las necesidades y requisitos del proyecto. En este capítulo, abordaremos las herramientas que se utilizaron para que el proyecto comience su desarrollo.

## 2.1. Estado del arte

Comencemos recalcando la diferencia entre autorización y autenticación.

- Autenticación: Se trata de la comprobación de los datos o credenciales en los intentos de conexión. Se envía una petición cliente-servidor de acceso remoto en forma de texto plano o cifrado.
- Autorización: Es la comprobación de aceptación de la conexión. La autorización tiene lugar después de que el cliente se haya autenticado.

Centraremos la investigación pues, en la autenticación. Ya que no existe autorización sin un previo mecanismo de comprobación, parece razonable pensar que el primer paso de todo sistema de *login* es la autenticación.

Todo sistema de identificación, ha de poseer unas determinadas características para que sea viable. Para empezar, tiene que ser fiable, factible económicamente para la organización y debe soportar cierto tipo de ataques. Sobre todo, ha de ser aceptable e intuitivo para el usuario. De nada sirve que el usuario tenga que recordar contraseñas de 30 caracteres de forma obligatoria, o requiriera de un esfuerzo excesivo por su parte.

Ahora bien, lo difícil de este sistema no es identificar al usuario en sí, sino comprobar que esa persona es quien realmente dice ser. Los métodos de autenticación se suelen dividir en tres grandes categorías:

- Algo que el usuario sabe
- Algo que el usuario posee
- Características físicas del usuario (autenticación biométrica)

Descartaremos el último método, ya que no procede para el desarrollo de la actividad. Sin embargo, tanto el primero como el segundo, serán utilizados activamente por la asociación.

En cuanto a "algo que el usuario sabe", será la autenticación básica y estándar de las aplicaciones. Tanto las web, como las de escritorio o móvil, necesitan validar los datos del usuario antes de atender a sus peticiones y facilitar los datos.

"Algo que el usuario posee". En la realización del evento anual que organiza la asociación, la autenticación de los usuarios en la entrada de dicho evento se realiza por lectura de tarjeta RFID. De esta forma, se validan los datos y se da acceso al recinto.

Actualmente, la aplicación de autenticación para las tarjetas está desarrollada en Python, y utiliza una base de datos local. Una vez el proyecto esté terminado, se podrá acoplar este aplicativo al sistema centralizado de usuarios para darle mayor dinamismo al proceso, y así no tener que mantener otra base de datos separada.

Ahora que sabemos lo básico acerca de la autenticación, procedemos a la definición de lo que se busca en este proyecto. El "Single Sign-On" (SSO) o "Sistema centralizado de autenticación y autorización" es un procedimiento que permite a los usuarios acceder a varios sistemas con una sola instancia de la autenticación.

Existen varios tipos de sistemas SSO, algunos son:

- Enterprise single sign-on (E-SSO): Autenticación primaria. Intercepta los requerimientos de login que se le presentan por parte de otras aplicaciones para completarlos con las credenciales del usuario.

- Web single sign-on (Web-SSO): Solamente trabaja con aplicaciones y recursos en la web. A diferencia del E-SSO, los accesos no se interceptan directamente, sino a través de un proxy o un componente en el servidor destino. Utiliza cookies para su fincionamiento.

- Kerberos: Uno de los métodos más populares para externalizar la autenticación. Utiliza un sistema de tickets donde el usuario ha de presentarlo a las aplicaciones clientes una vez que el servidor se los asigna.

Las principales ventajas de utilizar un sistema SSO son más que evidentes. Por un lado, reducimos el cansancio que puede provocarle al usuario tener diferentes nombres de usuarios y/o contraseñas. Por otro lado, también reducimos el tiempo que dichos usuarios se pasan introduciendo credenciales en los formularios.

Por supuesto, también tiene desventajas. La principal, es que si el sistema central falla, o es atacado, todas las aplicaciones y datos de usuario de los distintos programas se verán comprometidos.

En la siguiente tabla, podemos encontrar algunos de las compañías o aplicaciones que utilizan esta metodología de SSO. Actualmente son muchos más los sistemas que lo usan. Sin embargo, se especifican las siguientes a modo de ejemplo.

| **Sistema** | **Uso** |
|---|---|
| Active Directory | Sistema de Microsoft |
| CAS | Protocolo SSO servidor/cliente |
| Facebook Connect | SSO para aplicaciones externas a Facebook |
| SAML Suite | SSO para ASP.NET |

Tabla 2.1: Tabla de sistemas SSO

## 2.2. Reuniones preliminares

Una vez el proyecto se encontró completamente definido, comenzaron las reuniones del equipo de desarrolladores. En ellas, se planteó la forma de trabajo, los miembros que finalmente participaron de forma activa en el desarrollo así como las herramientas utilizadas para que salga adelante.

En estos primeros encuentros, se pusieron de manifiesto dos técnicas muy utilizadas en el la planificación del desarrollo de software. Son el "Brainstorming" y el "Juicio de Expertos". Se comienzan a elaborar propuestas y a descartar ciertas herramientas que no van a satisfacer las necesidades principales.

## 2.3. Propuestas

A continuación se destacan cuatro de las propuestas que finalmente se tomaron en consideración antes de comenzar con el desarrollo real.

### 2.3.1. LDAP y CAS

La primera de las propuestas es la de integrar dos herramientas bien conocidas y utilizadas en sistemas de usuarios de muchas organizaciones e instituciones alrededor del mundo.

- LDAP: Corresponde a las siglas en inglés *"Lightweight Directory Access Protocol"* (en español, Protocolo Ligero de Acceso a Directorios). Se trata de un protocolo a nivel de aplicación que es capaz de permitir o negar el acceso a los usuarios que intentan encontrar información en un sistema de red. También puede considerarse una base de datos en la que se realizan consultas. Sin embargo, el sistema de almacenamiento que usa no tiene por qué asemejarse a una base de datos propiamente dicha.
- CAS: Siglas de *"Central Authentication Service"* (Servicio de Autenticación Central, en español). Se trata de un sistema de autenticación creado en la Universidad de Yale para proveer a las aplicaciones de un sistema seguro y de confianza para que los usuarios se identificaran.

La utilización de ambas herramientas en conjunto pueden dar una solución al problema. Para su integración, simplemente es necesario utilizar un Handler para LDAP que soporta oficialmente CAS (según su documentación).

- Ventajas
  - Ambas herramientas tienen una fuerte documentación y una comunidad establecida.
  - Son herramientas muy utilizadas en producción.
  - Se complementan de una forma excelente y cumplen las expectativas del proyecto.
- Desventajas:
  - Requieren un estudio previo y una investigación extensa.
  - Al utilizar LDAP, no está claro que aplicaciones web, de escritorio o móviles puedan utilizar el sistema de autenticación sin problemas.

### 2.3.2. OpenID y OAuth2

La siguiente propuesta vuelven a ser dos herramientas muy utilizadas en la identificación de usuarios, sobre todo vía web. Esta propuesta se inicia pensando en un sistema que utilice ambas.

- OpenID: Protocolo de *autenticación* basado en la familia OAuth2. Utiliza mensajes REST/JSON para realizar la comunicación de los datos.
- OAuth2: Se trata de un protocolo abierto, esta vez de *autorización*, como método simple para aplicaciones web, de escritorio y de dispositivos.

Por lo tanto, podemos asegurar que OpenID es la capa de identidad sobre el procolo base de OAuth2. La principal diferencia entre ambos protocolos es que OpenID proporciona una misma *autenticación* para muchos sitios mientras que OAuth2 permite *autorizar* que un sitio consuma información de otro.

Ya que necesitamos ambos métodos, autenticación y authorización, OpenID puede sernos más útil, ya que, al estar basado en OAuth2, puede implementar fácilmente los dos protocolos. De esta forma, la propuesta final se elabora con esta herramienta como base.

- Ventajas:
  - Protocolo ligero y seguro.
  - Muy extendido y utilizado (Google, Microsoft, etc).
- Desventajas:
  - La autorización de OAuth parece diseñada para aplicaciones de terceros.
  - Requiere implementación extra para tener un sistema propio funcionando.

### 2.3.3. Desarrollo Node.js

La tercera propuesta consiste en un trabajo de investigación y desarrollo bastante avanzado. Trata de poner en común todas las necesidades actuales para que el sistema salga adelante y codificarlas utilizando para ello Node.js.

- Node.js: Plataforma construida bajo el motor JavaScript de Chrome. Utilizada para desarrollar aplicaciones web rápidas y escalables.

Utilizar javascript del lado del servidor puede parecer extraño a primera vista. Sin embargo, el uso de Node.js se ha extendido de manera exponencial en los últimos años.

Node.js tiene un amplio repositorio de paquetes y librerías que pueden realizar diferentes trabajos o tareas. Desde autorización o autenticación, hasta conectores para las principales bases de datos, o monitorización de servidores.

Se estudian y se ponen en práctica varias aplicaciones desarrolladas por terceros que implementan un desarrollo de SSO en Node.js. Existen infinidad de proyectos en *GitHub* que, a priori, pueden servir para el proyecto.

Se prepara una demo con los seleccionados, esta opción parece muy prometedora.

- Ventajas:
  - Customización adaptable.
  - Basado en herramientas como OAuth2 y OpenID, que disponen de amplia documentación.
- Desventajas:
  - Requiere un trabajo de investigación importante.
  - También requiere un desarrollo continuado y un equipo de trabajo dedicado casi exclusivamente a su codificación.

### 2.3.4. API interna

Por último, se pone sobre la mesa una propuesta mucho más simple. Dado que todas las aplicaciones, actuales y futuras, van a ser controladas desde los propios servidores de la asociación, no es necesario un enfoque de utilización de aplicaciones de terceros.

Se puede controlar la comunicación entre las distintas APIs de las aplicaciones directamente, permitiendo un origen u otro de los datos, o utilizando algún método como el de los tickets, tokens, etc.

Se decide utilizar esta opción, por su sencillez y potencia (Figura 2.1).

Una capa de autenticación enviará los datos al servidor central. Éste devolverá un token al usuario, y la comunicación entre las diferentes aplicaciones utilizando la API interna tendrá en todo momento información de los tokens generados.

Para comprender mejor todo el proceso, se especifica a continuación una autenticación hipotética. Supongamos que un usuario desea acceder a la aplicación 2 de la figura 2.1. Además, esta aplicación necesita datos de otra de las aplicaciones del sevidor, en este caso, la aplicación 1.

- 1) El usuario (cliente) introduce sus credenciales.
- 2) La capa de autenticación pregunta a la API del Sistema Central si dichas credenciales son correctas.
- 3) Si el usuario ha introducido datos correctos, se genera un token que se envía tanto al cliente como a la aplicación a la que el usuario quiere acceder.
- 4) La comunicación entre las aplicaciones es transparente para el usuario, se comprueban que todas las conexiones tienen un origen válido y la información se comparte.
- 5) La aplicación devuelve datos al usuario que realiza la petición, comprobando que es el mismo token que se generó en el Sistema Central.

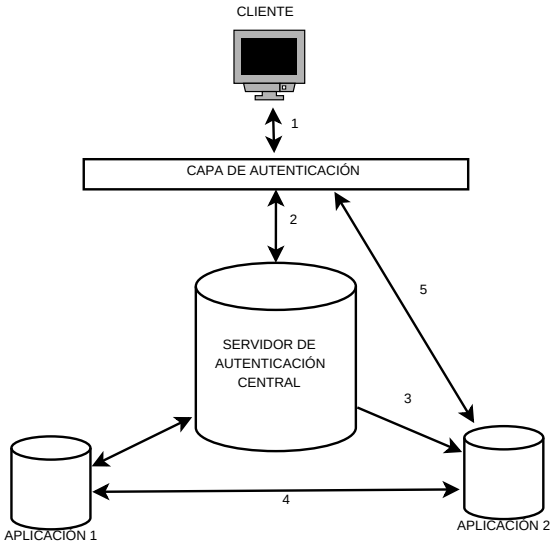


Figura 2.1: Propuesta API

# Capítulo 3

# Decisiones

Una vez la propuesta ha sido aprobada por todos los integrantes del equipo de trabajo, se dispuso a comenzar el desarrollo del proyecto.

## 3.1. Herramientas a utilizar

Debido a que este sistema iba a sustituir al actual, y aprovechando la migración de las bases de datos de la organización a un sistema PostgreSQL, se decidió que la base de datos estará alojada utilizando esta tecnología.

Además, para los datos de usuario, también se aprovechó un nuevo registro inminente de miles de personas que se realizó en los meses de mayo/junio, semanas previas a la inauguración del evento "TLP Tenerife". De esta forma, se contaba ya con una base de datos de usuarios en PostgreSQL, que sirve para autenticar usuarios de las futuras aplicaciones. La primera aplicación que hará uso de este sistema será la realizada por el propio alumno.

El sistema de inscripción de dicho evento es una aplicación en Python/Django. No requiere ninguna modificación, ya que solamente se encarga de inscribir nuevos usuarios en la base de datos.

Aprovechando esta aplicación, se desarrolla la capa de autenticación sobre ella. De esta forma, la interacción con la base de datos principal se hará a través de Django, y las credenciales se comprobarán por esta vía.

## 3.2. Metodología

Se proponen algunas metodologías de uso para crear el sistema.

- Desarrollo en Cascada: En este tipo de metodología, se ordenan las tareas de forma descendente y secuencial. Una tarea no puede comenzar si la anterior no ha terminado. Al final de cada etapa, se realiza una revisión final que especifica si se puede avanzar a la siguiente fase. Es una metodología estándar muy extendida y que se lleva usando mucho tiempo.

- TDD: El desarrollo guiado por pruebas (Test Driven Development, en inglés), es una práctica que consiste en desarrollar las pruebas antes que la aplicación. Generalmente se escriben pruebas unitarias (test units). En un primer lugar, se escribe una prueba y se verifica que falla. A continuación, se implementa el código principal para que la prueba se pase, y se refactoriza el código.

A pesar de que en un primer momento se optó por la segunda opción (TDD), surgen una serie de contratiempos que obligan al equipo a elegir una metodología más tradicional (cascada).

En primer lugar, el proyecto debe estar terminado y funcionando en apenas dos meses. Ya que la asociación va a utilizar este sistema, es necesario que no se retrase lo más mínimo.

Por otra parte, el equipo dedica sus horas libres al desarrollo, lo que hace muy difícil ponerse de acuerdo para sesiones de trabajo conjunto, o poder sacar más tiempo del necesario.

## 3.3. Desarrollo

Una vez se ha decidido la metodología de trabajo y se tienen claras las herramientas a utilizar, se comenzó con el desarrollo.

La mayor parte del trabajo consiste en preparar la base de datos de PostgreSQL para que la aplicación de inscripción (Django) la utilice.

Se cambiaron los conectores desde la antigua base de datos hacia la nueva. Se preparó también la capa de autenticación y quedó lista para que las aplicaciones y los usuarios puedan obtener credenciales sobre ella.

Esta capa actúa como conector entre los usuarios y el sistema centralizado de usuarios principal. Se trata pues de una API que gestiona las peticiones desde el exterior hacia el sistema. Simplemente se encarga de supervisar que las peticiones y las respuestas se efectúan de manera correcta, genera excepciones y controla errores.

Además, se encarga de devolver la información de las distintas aplicaciones al usuario, una vez se ha comprobado la autenticación del mismo. En definitiva, esta capa es simplemente un nivel más entre los clientes y los servicios internos de la asociación.

A partir de ese momento, comienzó el desarrollo de la aplicación cliente para comprobar el sistema.

# Capítulo 4

# Planteamiento de la aplicación

El sistema se encuentra funcionando correctamente. Después de participar en reuniones y en el desarrollo de la API interna que hará de sistema centralizado, se planteó la propuesta de aplicación para realizar las pruebas.

## 4.1. Documentación

En primer lugar, se planteó la elección del tipo de aplicativo que se iba a desarrollar. Podía ser de escritorio, web, dispositivos móviles, etc.

Ya que mayoritariamente la asociación utiliza aplicaciones web, se optó por proponer una aplicación de este tipo, teniendo además soporte para dispositivos como móviles o tablets. De esta forma, se puede probar el funcionamiento y comportamiento tanto de la aplicación como del sistema en dos grandes plataformas reales y usadas por la organización.

Existieron diferentes opciones a la hora de realizar la aplicación. Las siguientes se tuvieron en consideración.

- AngularJS: Framework de código abierto en JavaScript para la realización de aplicaciones web One Site (de una página). Actualmente está siendo mantenido por Google. Utiliza el Modelo Vista Controlador (MVC), orientado a que el desarrollo y las pruebas sean más fáciles. Al contrario que otras utilidades JavaScript para el desarrollo de aplicaciones web, como por ejemplo jQuery, Angular no modifica directamente el DOM del HTML. En su lugar, utiliza atributos en las etiquetas del código HTML y abstrae al programador de la manipulación pura de las mismas. Además, aboga por un desarrollo en paralelo de la parte FrontEnd y BackEnd, de modo que aplicaciones cliente-servidor puedan ser codificadas al mismo tiempo.

- Python/Django: Django es un framework de desarrollo web de código abierto escrito en Python. Al igual que AngularJS, utiliza un paradigma Modelo Vista Controlador para facilitar la tarea a los desarrolladores de aplicaciones web. Cabe destacar el énfasis que da Django en el re-uso y la conectividad de los componentes. La modularidad y la intención de no repetir bloques de código. Parece interesante tener a

consideración esta propuesta, ya que el sistema central está desarrollado en Django. Sin embargo, debido a que la diversidad de herramientas y lenguajes sería más productiva a la hora de realizar pruebas reales, puede que no sea la opción más acertada.

- Android SDK: Herramientas de desarrollo para Android. La opción de realizar la aplicación para dispositivos móviles también sería muy interesante. Son muchos los usuarios Android que utilizarían la aplicación. Sin embargo, existe el problema de otras plataformas como iOS o Windows Phone. El desarrollo en esta propuesta sería bajo el lenguaje Java, sobre el que trabajan las aplicaciones nativas para Android.
- PhoneGap: Phonegap es un framework para el desarrollo de aplicaciones móviles utilizando herramientas genéricas de desarrollo web como JavaScript, HTML5 y CSS3. Las aplicaciones que se realizan con este framework son híbridas, es decir, no se desarrollan en nativo para ningún dispositivo. De esta forma, una misma aplicación serviría tanto para web como para móviles. Al no ser un desarrollo específico para una plataforma, la aplicación podría renderizarse para los distintos sistemas operativos móviles que hay. De esta forma, la misma aplicación podría servir para múltiples plataformas utilizando el mismo código.

Finalmente se optó por la última opción. El desarrollo de la aplicación en PhoneGap permitiría ahorrar tiempo y se podrían realizar pruebas en distintas plataformas y dispositivos.

## 4.2. Requisitos de la aplicación

Existen una serie de requisitos que debe cumplir la aplicación para que se considere exitosa:

- Debe suplir una necesidad real de la asociación.
- Será utilizada por miles de usuarios, asistentes a diferentes eventos que la organización realiza.
- Deberá ser desarrollada de forma robusta para que se compruebe el sistema centralizado de usuarios.
- Además, la integración con las redes sociales debe ser una prioridad para que se cumplan los objetivos de la aplicación.

## 4.3. Especificación de la aplicación

Ya que el evento de "TLP Tenerife" es el mayor que realiza la asociación a lo largo del año, la aplicación va a ofrecer servicios reales y actualizados a los usuarios de dicho evento. Se trata de una app principalmente móvil, que ofrecerá información y noticias en tiempo real, así como permitirá que los usuarios gestionen sus actividades dentro del recinto.

La aplicación deberá contar con una base de datos específica para almacenar datos propios. A pesar de que el usuario utilice la autenticación central para algunas de las funcionalidades, la mayoría de características se podrán utilizar sin necesidad de realizar una identificación del usuario.

# Capítulo 5

# Desarrollo de la aplicación

En este capítulo, se entrará en detalle acerca del proceso que se siguió para realizar la aplicación.

## 5.1. Herramientas utilizadas

Al utilizar PhoneGap, la totalidad de herramientas y tecnologías que se usaron para el desarrollo de la aplicación fueron las mismas que para la creación de un proyecto web.

Del lado del cliente:

- HTML5: Quinta revisión del popular lenguaje de marcado para la creación de sitios web. Implementa nuevas funcionalidades en comparación con sus predecesores que hacen que el desarrollo de aplicaciones web sea muchísimo más potente que hace algunos años.

- CSS3: Lenguaje de hojas de estilo que se utiliza para describir el formato y la apariencia de un documento escrito en lenguaje de marcas. Su integración con HTML5 es muy buena. Con una sintaxis muy simple, esta nueva versión de CSS permite funcionalidades muy vistosas como animaciones, modificaciones en 3D, etc.

- JavaScript: Lenguaje de programación interpretado presente en la mayoría de navegadores web. Se implementa en el propio navegador desde el lado del cliente para el desarrollo de páginas dinámicas.

- jQuery: Una de las principales librerías JavaScript para el desarrollo de páginas web. Rápido y muy ligero, permite acciones que modifican directamente el DOM abstrayendo al usuario con una API de uso simple. También permite tareas como peticiones AJAX, de las que hablaremos más adelante.

Por otra parte, la aplicación en el servidor se creó utilizando:

- PHP: Lenguaje de programación de uso general que lleva siendo utilizado desde hace muchos años del lado del servidor a la hora de desarrollar contenido dinámico para

la web. El código es interpretado en el servidor, y devuelve al cliente información procesada para su uso.

- PostgreSQL: Sistema de gestión de bases de datos relacional. Se utilizará una nueva base de datos para guardar información inherente a la aplicación. Aprovechando la infraestructura original de la asociación, los scripts en PHP accederán a los datos de las nuevas tablas después de que los usuarios realicen su identificación contra el sistema central.

- API: La esencia del sistema centralizado. La API nos dará las credenciales necesarias para autenticar a los usuarios y que hagan uso de la información del servidor.

La integración de todas estas herramientas ha propiciado que los principales requisitos formulados en el apartado anterior se puedan cumplir.

## 5.2. Desarrollo

La aplicación PhoneGap cuenta con el siguiente esquema o árbol de directorios:

```
TLPConnect
├── hooks
├── merges
├── platforms
├── plugins
└── www
    ├── css
    ├── img
    ├── js
    └── index(html)
```

Es generado automáticamente por el framework. La aplicación web está almacenada en el directorio *www*. Existe una amplia variedad de plugins para PhoneGap que abstraen al usuario a la hora de utilizar funcionalidades del dispositivo móvil tales como acelerómetro, cámara, geolocalización, etc.

El propio framework integra un comando para instalar los *plugins* de manera sencilla. En principio no se utilizará ninguno de ellos. Sin embargo, es posible que en futuras versiones de la aplicación se añadan para agregar o mejorar funcionalidades básicas de cara al usuario.

Bajo el directorio *platforms* se especifican las plataformas para las cuales la aplicación va a ser desarrollada. En un principio, se efectuaron todas las pruebas en android por simplicidad. Más tarde se incorporará iOS como plataforma. Se estudia la posibilidad de añadir también Windows Phone, aunque es poco probable debido a la poca utilización del sistema.

## 5.3. Funcionamiento de la app

La aplicación se puede dividir en dos partes. Cada una de ellas son pasos secuenciales que se ejecutarán, de forma general, para realizar todas y cada una de las tareas que el usuario necesite.

- Conexión: El usuario introducirá los datos de conexión (usuario y contraseña). Se genera un token único de usuario que se utilizará para la sesión.
- Petición: Cuando el usuario realiza una petición a la aplicación, ésta comprobará las credenciales del token enviado por el cliente contra el sistema central. Si todo funciona correctamente, se ejecutará la petición y se devolverán los datos en la respuesta. La comunicación entre el la aplicación (en este caso, en PHP) y la API del sistema central para obtener el token se realiza tal y como se especificó en un primer momento. Se realizan las conexiones de manera interna y segura especificando el origen de la petición.

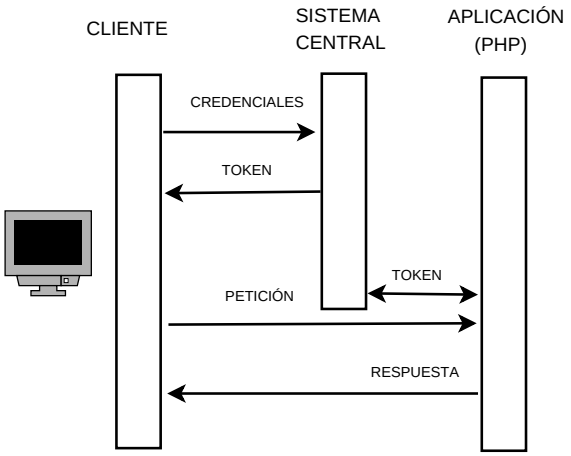


Figura 5.1: Funcionamiento de la app

## 5.4. Tareas y funcionalidades

La aplicación cuenta con una serie de tareas que se detallan a continuación. Estas son las tareas básicas que se han acordado en reuniones con la asociación. Cada una de ellas se puede seleccionar desde el menú principal.

En la siguiente figura, podemos ver que el diseño del menú se ha realizado utilizando las herramientas descritas anteriormente. Se intenta crear una interfaz sencilla e intuitiva, que cumpla con las principales características de la usabilidad. Además, se han utiliazdo colores corporativos, similares a los que la asociación usa en sus otras aplicaciones o en su página web principal.


Figura 5.2: Captura de la app (menú)

Tareas:

- Noticias: Son muchas las noticias que se generan tanto antes como durante la realización del evento. Es muy importante que esta información se actualice en tiempo real para que los usuarios puedan estar al tanto de las novedades. Una petición AJAX al servidor será suficiente para saber si existen nuevas noticias y traer sus datos. Para esta funcionalidad no se requiere autenticación. Por tanto, los datos del *request* se enviarán desde la base de datos en el servidor hasta los clientes en forma de texto plano.
- Horarios y torneos: Similar a la funcionalidad de las Noticias. Se realiza una petición de información básica, y se devuelve actualizada.
- Notificaciones: La parte más importante del aplicativo son las notificaciones en tiempo real. Existe un gran número de aplicaciones para esta funcionalidad. A través del aplicativo podemos chequear nuestras actividades (torneos a los que nos hemos apuntado, charlas que nos interesan, ponencias o talleres, etc). De esta forma, podemos activar un *flag* que guardará la fecha de dicho evento para que nuestro móvil nos avise, utilizando la alarma, antes de que dicho evento se produzca.
- Mapa: Además, se cuenta con un mapa interactivo del evento. Dicho mapa, desarrollado íntegramente con JavaScript, nos permite una navegación similar a la de *Google Maps*, mostrando información de las distintas áreas o zonas, así como permitiendo

al usuario obtener datos acerca de las actividades que se desarrollan en cada una de ellas.

## 5.5. Administración de los datos

Una vez la parte del cliente fue desarrollada, se hacía necesaria una herramienta de gestión de la base de datos inherente a la aplicación. Ya que las notificaciones y los horarios podían ser alterados durante la realización del evento, modificar las bases de datos directamente en producción podía no ser una buena opción.

Se desarrolló, por tanto, un pequeño panel de administración accesible vía web por los administradores. En dicho panel se puede consultar la información actual, así como crear/editar/eliminar nuevas notificaciones a los usuarios, horarios de torneos y ponencias, etc.

## 5.6. Integración con redes sociales

Una requisito indispensable a la hora de diseñar la aplicación fue la integración con las principales redes sociales que utiliza la asociación: *Twitter y Facebook*. De esta forma, los usuarios pueden ofrecer información acerca de sus actividades dentro del evento a través de dichas redes sociales sin necesidad de acceder a sus respectivas aplicaciones.

Un buen ejemplo sería la publicación de un tweet anunciando la asistencia a una charla o taller importante con un solo click.

Para implementar esta funcionalidad, fue necesario un estudio e investigación de las APIS de las diferentes redes sociales. Es importante y necesario que el usuario dé acceso y permiso para utilizar sus credenciales a la hora de autenticarse.

- Twitter: Esta API fue la que más problemas ha dado. La actualización de los términos y del uso de su API REST. La versión anterior (v1) no requería ninguna autorización para realizar las peticiones. Sin embargo, esta nueva versión (v1.1) necesita que nos autentiquemos utilizando OAuth. Es necesaria esta autenticación si queremos realizar peticiones a la API REST general, y a la API de búsqueda de Twitter.

- Facebook: La API de Facebook permite a los desarrolladores utilizar funcionalidades propias de la red social, tales como utilizar el botón ´´Me Gusta", Obtener información de estados, realizar LogIn con sus credenciales, etc. Para poder utilizarla, es necesario obtener una api-key. Para ello, debemos registrarnos como "desarrolladores"en Facebook. Sin embargo, existen muchos plugins ya implementados que nos facilitan el trabajo a la hora de codificar ciertas tareas, que no requieren comenzar el desarrollo desde cero. Por ejemplo, es posible construir el código del plugin "Me Gusta" especificando únicamente la dirección del sitio en Facebook que queremos mostrar. Utilizando el siguiente cuadro de texto en la propia página de la red social, se nos genera el código para cualquier plataforma (web, Android, iOS, etc).

## 5.7. Mejoras

Existen una serie de mejoras que se han propuesto. La mayoría de ellas son funcionalidades nuevas de la aplicación o un incremento de la eficiencia general del aplicativo. En cuanto a las posibles funciones nuevas:

- Reserva de plazas: Debido a que los talleres y conferencias tienen un número limitado de asistentes, se podrían reservar un porcentaje de esas plazas para los usuarios que se hayan descargado la aplicación y hayan hecho uso de ella. De esta forma, se puede obtener información de antemano acerca de cuántos asistentes irán a los eventos, para tomar medidas como aumentar plazas, cambiar el lugar de celebración, etc.

- Gestor de incidencias: Aprovechando la comunicación directa del servidor de la organización con los usuarios que hayan descargado la app, se podría implementar un servicio de envío de información desde el cliente a la aplicación. De esta forma, no sería muy complicado añadir esta funcionalidad, ya que se podrían gestionar las incidencias desde el propio panel de administración de notificaciones.

- Promociones o concursos: Para incentivar el uso de la aplicación, se ha pensado incluir algunas promociones o concursos que requieran tener descargada la app y utilizarla de alguna manera. Una vez el usuario envía alguna de las peticiones, se podría guardar información acerca de ese usuario y tener una lista de concursantes que optarían a un premio.

También se estudia la posibilidad de mejorar el sistema de peticiones utilizando un servicio de *pushing* de los datos. Actualmente esta propuesta está en desarrollo. Se han intentado varios ejemplos de prueba, pero existen algunas restricciones a la hora de utilizar productos que implementen la funcionalidad.

Se trata de un sistema de comunicación entre cliente-servidor que utiliza un WebSocket como canal. Un WebSocket utiliza un socket TCP para proporcionar dicha comunicación. Servicios como "Pusher"hacen uso de esta tecnología para evitar continuas peticiones del cliente hacia el servidor para evitar sobrecargarlo.

Lo que ocurre es que, es el servidor quien envía un mensaje a los clientes cuando se ha realizado algún cambio, o se necesita una actualización de la información. De esta manera se mejora notablemente la eficiencia del sistema en general, y se reduce el tráfico de peticiones hacia el servidor.

La primera herramienta de tipo *Push* que se intentó implementar fue Pusher (Figura 5.3).

Pusher ofrece una API alojada en internet. De modo que solo se requieren unas pocas líneas de código en la aplicación y en el servidor para que todo funcione. Se puede indicar a los scripts de PHP que, al realizar algún cambio, lo notifiquen mediante la API al Pusher, y éste envíe el mensaje a todos los clientes. Sin embargo, existen una serie de limitaciones al usar este servicio, por lo que, en principio, queda descartado para su utilización:

- Conexiones: El plan básico y gratuito solo permite 20 conexiones simultáneas. Teniendo en cuenta que el aforo del evento supera los 2000 usuarios, es completamente inviable.


Figura 5.3: WebSocket Pusher

- Mensajes: Además, el número de mensajes por día es de 100.000
- Seguridad: No se ofrece ningún tipo de seguridad en los planes básicos. A partir de un plan mediano (que tendría un coste de 49 dólares/mes), se puede acceder a una protección SSL opcional.

Otra herramienta similar a Pusher fue "PubNub". Ésta tiene muchas más funcionalidades además del sistema de *push* de datos de las que no hablaremos. Cuenta con un sistema de *queue* o cola, para que no se desborde la conexión. Sin embargo, también es de pago.

Tras un poco de investigación, se encontraron dos soluciones open source que podrían solventar el problema.

- Slanger: Implementación en Ruby del protocolo Pusher. Está especialmente diseñado para escalar horizontalmente. Está empaquetado como una gema de Ruby, *RubyGem*. Utiliza Redis, entre otros servicios, para ofrecer sus funcionalidades básicas.
- Atmosphere: Se trata de un sistema más complejo. Un framework para desarrollar aplicaciones asíncronas utilizando WebSockets, entre otros.

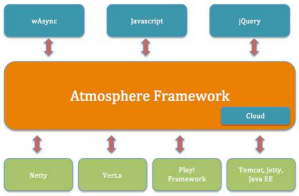


Figura 5.4: Atmosphere Framework

# Capítulo 6

# Testeo en pre-producción

Una vez el desarrollo de una versión inicial de la app estuvo terminado, se comenzó la migración de los scripts y la creación de la base de datos en el servidor principal para comenzar las tareas y pruebas de pre-producción.

## 6.1. Migración al servidor

La migración del servidor se ha realizado en dos fases principales.

- Base de datos: Primero se ha creado la base de datos con sus usuarios respectivos de lectura y de administración. El usuario de lectura será el encargado de realizar las consultas sobre la base de datos para devolvérselas a los usuarios a través de los scripts de PHP. El usuario de administración es el que utilizará la organización a través del panel descrito en el capítulo anterior. Se ha utilizado el siguiente esquema de base de datos para la aplicación:

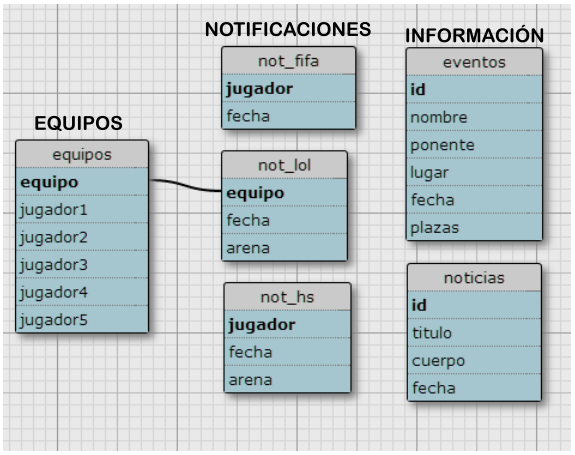


Figura 6.1: Esquema de la base de datos

- Por un lado, tenemos las tablas "noticias" y "eventos" que son las de información general. Las tablas de notificaciones vienen precedidas por el sufijo "not". De esta forma, podemos tener controladas las diferentes notificaciones para los distintos usuarios que hagan uso de los eventos o funcionalidades específicos. La tabla "equipos" simplemente se utiliza a modo de ayuda, ya que algunas notificaciones se enviarán a miembros de un equipo completo y no solo a jugadores o usuarios determinados.

- Scripts PHP: Cuando la infraestructura básica se generó en el servidor, fue la hora de publicar los scripts de la aplicación. Se generó una ruta a través de los subdominios de la asociación. De este modo, los dispositivos podrán enviar sus peticiones AJAX directamente utilizando URLs personalizadas para cada tipo de funcionalidad.

## 6.2. Problemas

Durante la migración al servidor, surgieron una serie de problemas de fácil solución. En primer lugar, las peticiones del cliente a la aplicación no devolvían los datos correctamente. Después de un período de investigación, se llegó a la conclusión de que los permisos del usuario de lectura no se habían configurado correctamente. La restauración del usuario solucionó este problema de conexión a la base de datos. También hubo algún problema a la hora de integrar el panel de administración en el servidor. Algunas peticiones se estaban realizando de forma síncrona, lo cual bloqueaba el navegador y el resto de procesos del panel. Al cambiarlas a un tipo asíncrono, todo funcionó correctamente. El resto de fallos fueron pequeñas correcciones en vivo del código. Alguna mejora de última hora en una tarea determinada, etc.

## 6.3. Tiempos de respuesta

Después de realizar la migración completa, se comienzan una serie de pruebas para comprobar el rendimiento en cuanto a tiempo de la aplicación. Como nota, recalcamos que las pruebas se realizaron cuando se contaba con una red de datos con velocidad buena (3G o superior).

Nos fijamos en que los tiempos de respuesta desde el servidor son muy rápidos en operaciones con poca información, y llegan a ser casi el doble a la hora de traer las noticias o realizar las operaciones de LogIn. Aun así, está dentro de los límites permitidos, una gran noticia para el desarrollo de la aplicación.

| **Función** | **Tiempo medio (s)** |
|---|---|
| Identificación | 0.96 |
| Noticias | 1.92 |
| Notificaciones | 0.69 |
| Horarios | 0.88 |

Tabla 6.1: Tabla de tiempos

Para la realización de las medidas se hicieron diferentes pruebas. Se optó por elegir la media aritmética de cada una de las pruebas. Se realizaron varias series de 100 peticiones por segundo hacia el servidor. De esta forma, se pudo comprobar, tanto el tiempo que tarda el servidor en realizar los trámites de identificación y autenticación del usuario, como el acceso a diferentes fuentes de información por parte de la aplicaicón cliente.

# Capítulo 7

# Conclusiones y Trabajos Futuros

A grandes rasgos, ha sido un proyecto ambicioso que ha contado con muchísimas propuestas, tanto de herramientas como de métodos de trabajo, para aportar la solución a un problema real de una entidad que trabaja con miles de usuarios cada año.

Lo más interesante del proyecto es su aplicación real. No se trata de un trabajo que se ha realizado y no se le va a dar un uso o una continuidad más allá de la de ser evaluado, sino que se podrán sacar conclusiones después de su puesta en funcionamiento.

Además, me ha dado la oportunidad de trabajar con un equipo de profesionales que comparten la pasión por el desarrollo y la aplicación de nuevos métodos y tecnologías en Canarias. Por tanto, creo que ha sido una experiencia muy enriquecedora para crecer, tanto en lo profesional como en lo personal. El ser miembro de ese grupo de entusiastas ha hecho que aprenda muchísimo del funcionamiento de proyectos en el mundo real, y no solo en el ámbito académico.

Es cierto que, en un principio, se encontraron algunas dificultades que hicieron que el proyecto retrasara su inicio, una vez hubo empezado el trabajo todo comenzó a rodar. Las pruebas finales dieron buenos resultados, se tuvo que trabajar en la solución de pequeños errores que, finalmente, fueron corregidos.

## 7.1. Problemas generales

A continuación, se especifican en detalle algunos problemas encontrados.

- Equipo de trabajo: Todo el proyecto comenzó con las reuniones iniciales para poner sobre la mesa el trabajo a realizar, así como especificar el equipo de desarrolladores que estaría implicado directamente en él, tal y como se explicó en el Capítulo 1. Al tratarse de una organización sin ánimo de lucro, el tiempo que le dedican sus socios a la puesta a punto de proyectos como este es limitado. Todos los miembros se dedican a esto como un "hobby". Por tanto, encontrar el tiempo necesario para realizar las reuniones, comenzar la documentación, realizar las pruebas y tener a punto el sistema fue algo complicado. Sin embargo, se encontró que un pequeño grupo pudo continuar con las tareas para que el sistema saliera adelante, lo cual fue un gran avance, e hizo que se pudiera continuar casi sin contratiempos hasta su

finalización.

- Margen de tiempo: Al tratarse de un sistema que sería implantado a lo largo del año 2014, la línea de tiempo era muy estricta. El mayor evento que realiza la asociación tendría lugar en Julio. Por tanto, todo debía estar preparado para los primeros días de mayo. La aplicación sí podría retrasarse un poco, ya que sería utilizada durante el propio evento.

- Pruebas reales: Realizar las pruebas en el servidor de producción fue un poco caótico a priori. Se comenzaron a realizar en el mes de mayo. El mismo mes en el que la inscripción del evento se abría, y el mismo mes en el que los pagos de dicho evento se iban a realizar de forma online. Debido a esto, se tuvo que trabajar conjuntamente para no interrumpir los servicios en ningún momento. Teniendo en cuenta la gran cantidad de peticiones que se realizan al servidor principal durante el período de pago (del orden de 3000 peticiones por segundo). Las pruebas tuvieron que suspenderse durante esos días críticos para la máquina.

## 7.2. Trabajos futuros

A pesar de que, tanto el sistema principal y la aplicación se han completado al 100 % para la fecha prevista, existen muchas mejoras que se pueden aplicar, a modo de tareas opcionales en un futuro.

### 7.2.1. Mejoras en el sistema

El sistema principal está actuando como una API que gestiona los tokens para las diferentes aplicaciones. Es posible aplicar algunas mejoras de seguridad sobre las comunicaciones entre dicha API y las aplicaciones internas.

### 7.2.2. Mejoras en la aplicación

La aplicación cuenta con las funcionalidades básicas para que se pueda hacer un buen uso de ella en el evento. Sin embargo, existen tareas avanzadas que podrían ser implementadas para que los usuarios tuvieran una mejor experiencia dentro del evento. Algunas pueden ser:

- Wifi: El acceso wifi dentro del evento está controlado. Una buena opción para que los participantes pudieran usarlo, sería utilizar las credenciales del sistema de usuarios centralizado. Una vez un usuario se autentica, la dirección MAC de su dispositivo queda almacenada para permitir usar la red de conexión inalámbrica del recinto.

- Tienda online: Dentro del evento son muchos los servicios que se ofrecen. Existen *stands* de promociones, venta de objetos y *merchandising* así como acceso a la cafetería y restaurante del recinto. Una buena opción de reservar o pre-comprar algunos de estos artículos o servicios sería un ahorro de tiempo y dinero para los usuarios.

- Servicio Push: El servicio de push descrito en el *Capítulo 5* sigue en desarrollo. Sería una buena manera de mejorar los tiempos de acceso así como la carga del servidor desde la aplicación cliente. Además, se podría implementar de forma que el código se reutilice para futuras aplicaciones de la asociación.

### 7.2.3. Mantenimiento

El sistema centralizado no requiere de mayor mantenimiento que la monitorización durante períodos de sobreexplotación, tales como inscripción masiva, desarrollo del evento, etc.

Sin embargo, para que la aplicación pueda cumplir con sus objetivos, será necesario realizar un trabajo de actualización de la información en el servidor. Aprovechando la herramienta del panel de administración, se requerirá una actualización de los horarios diaria para que las notificaciones lleguen a los dispositivos lo más actualizadas posibles.

# Capítulo 8

# Summary and Conclusions

Broadly speaking, it has been an ambitious project that has had many proposals of both tools and working methods to bring the solution to a real problem in an organization that works with thousands of users each year.

The most interesting part of the project is the actual implementation. This is not a job that has been done and forgotten, but it will be used in real life.

It has also given me the opportunity to work with a team of professionals who share a passion for the development and usage of new technologies in Canary Islands. So I think that has been a very useful experience that have made me to grow, both professionally and personally. Being a member of that group of enthusiastic has allowed me to learn a lot about project running in the real world, not just in the university.

It is true that, at first, it was a bit difficult to start the project in the right dates. Final tests were successful, we had to work on the solution of small errors that eventually were corrected in order to complete the whole project.

## 8.1. General problems

Some problems were:

- Teamwork: The whole project started with the first meetings to discuss about the future work and to specify the exact developer team behind the system, as we covered along the first Chapter. As we read in the "Introduction", this is a nonprofit association. Because of this, members can't spend so much time doing activities, so their implication in projects is limited to their free time. All members are involved in the organization as an external hobby, never as their main job. For that reason, finding the necessary time for the meetings, documentation or investigation and tests was difficult at the beginning of the development. However, a little group of members found the time to start and continue the main tasks of the project. This was so good, in general, for the project to be finished almost without any big problem.

- Time: The system would be implanted along this year (2014). For that reason, the deadline was so strict. The main event of the association would take place in July. For

that reason, everything should work fine on early May. The client app development could delay a bit more.

- Real testing: It was quite difficult to test the system in production server at the beginning. Tests started on May. The main event registration and the online payment were in that date too. For that reason, the administration team and myself had to work together to avoid server issues and guarantee these services. In fact, some tests had to be suspended during these critical days due to the big increment of server requests (3000 requests per second).

## 8.2. Future work

The main system and the application are fully developed and working at 100 % by the specified schedule. However, there are plenty of new upgrades that can be included in the future, as new tasks and/or functionalities.

### 8.2.1. System improvements

Main system is acting as an API which manages authentication tokens for different applications. It would be possible to apply some security improvements to the inner connections between these apps and the API.

### 8.2.2. Application improvements

The app is fully working with basic needed tasks to be a strong tool in the main event. However, there are of course advanced funcionalities that could be developed in order to allow users to have a better experience. Some of them could be:

- Wifi: Wireless connection is controlled by the organization inside the event. A good choice could be take advantage on the main authentication system to allow users using the connection. When an user Logs In, device MAC is registered so he can use wifi over and over again during the event.

- Online Shop: There are a lot of different services inside the event building. There are promotion *stands*, item sold and all kind of *merchandising*, access to restaurant, etc. A good option would be booking or pre-purchase some of these services or articles in order to save money and time for the users.

### 8.2.3. Maintenance

Centralized users system does not require extra mantenance. It only has to be monitorized during overload periods, as massive registration, event performing, payments, etc.

However, the application needs some maintenance to get the main objectives. For example, it would be necessary to update database information. Using the recently developed

administration panel, schedules will need to be updated almost everyday in order to guarantee notifications to be displayed correctly in client devices.

# Capítulo 9

# Presupuesto

El proyecto no ha tenido ningún gasto en cuanto a materiales físicos (hardware) se refiere. De igual manera, no han surgido gastos de licencias, software o soporte de ningún tipo.

Sin embargo, mediremos el costo total de proyecto en horas de trabajo invertidas, así como en desplazamientos al local de la organización, donde se desarrolló la mayor parte del trabajo de investigación y las reuniones iniciales del sistema central.

El desarrollo de la aplicación fue desarrollado por el alumno de forma autónoma, con lo cual no se extraen gastos de desplazamiento en este etapa.

Para finalizar, al realizar las pruebas sí que fue indispensable reunirse con el equipo de administradores del sistema servidor de la asociación. Estos encuentros se realizaron de igual manera en el local de *Innova7*.

## 9.1. Tabla

| **Concepto** | **Precio** |
|---|---|
| Horas de trabajo | 250 x 15e = 3750e |
| Desplazamientos | 300e |
| TOTAL | 4050e |

Tabla 9.1: Tabla de presupuesto

# Bibliografía

[1] OAuth. http://oauth.net/documentation/.

[2] Open ID. http://openid.net/connect/faq/.

[3] PhoneGap Doc. http://docs.phonegap.com/en/3.0.0.

[4] PostgreSQL. http://www.postgresql.org/docs/.

[5] Python/Django. https://www.djangoproject.com/.