UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Safe Events: control de aforos mediante RFID

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Gallego Ferrer, Marcos Manuel

Tutor: Penades Gramage, María Carmen

2016-2017

Resumen

El objetivo del TFG es proporcionar un sistema de control e identificación de usuarios/asistentes a un evento en concreto. La aplicación móvil está destinada para que los empleados del evento conozcan en tiempo real la información mas importante de un evento, como es el numero de personas que está presente en el recinto. De está manera, las autoridades competentes podrán conocer si se está cumpliendo con el aforo permitido. El sistema, se basa en una aplicación móvil disponible tanto en IOS como Android, desarrollada con la tecnología de Google GWTP (mediante el modelo MVP, vista controlador) y con el motor de aplicaciones donde corre la aplicación y está la base de datos. La aplicación móvil consiste en un navegador web, pero pasado por la herramienta Cordova, la cual con los ajustes necesarios, nos permite que el navegador embebido se comporte como una aplicación nativa, sin tener que desarrollar la aplicación para cada sistema operativo. Para el control del acceso de los usuarios, se cuenta con un lector capaz de identificar pulseras pasivas RFID que portará el usuario; se identificará tanto la entrada como la salida. El sistema será capaz de comunicarse con el *backend* necesario para verificar que el usuario puede acceder al recinto, además de proporcionar información en tiempo real del aforo actual. Por lo tanto, también se ha de desarrollar un programa capaz de identificar las pulseras, y comunicarse con el *backend*. Otras ventajas que supone esta tecnología, es que se evita la reventa de entradas, ya que las pulseras van asociadas al asistente del evento. Los objetivos serán el desarrollo de la aplicación móvil, del *backend* y el desarrollo del sistema de lectura. El plan de trabajo es seguir una metodología ágil basada en *sprints* de tres semanas, acordando al inicio de cada *sprint* qué componentes y trabajos se van a llevar a cabo, y a la finalización del *sprint*, realizar una reunión, para ver qué puntos se han cumplido y cuales no, además de mostrarle las sucesivas progresiones del producto.

Palabras clave: control de aforo, aplicación móvil, RFID, backend, app.

Resum

L'objectiu del TFG és proporcionar un sistema de control e identificació d'usuaris/assistents a un esdeveniment en concret. L'aplicació mòbil està destinada perquè els empleats de l'esdeveniment coneguen en temps real la informació mes important d'un esdeveniment, com és el nombre de persones que està present al recinte. D'aquesta manera, les autoritats competents podran conèixer si s'està complint amb l'aforament permès. El sistema, es basa en una aplicació mòbil disponible tant en IOS com Android, desenvolupada amb la tecnologia de Google GWTP (mitjançant el model MVP, vista controlador) i amb el motor d'aplicacions on corre l'aplicació i on està la base de dades. L'aplicació mòbil consisteix en un navegador web, però passat per l'eina Cordova, la qual amb els ajustos necessaris, ens permet que el navegador encastat es comporte com una aplicació nativa, sense haver de desenvolupar l'aplicació per a cada sistema operatiu. Per al control de l'accés dels usuaris, es compta amb un lector capaç d'identificar polseres passives RFID que portarà l'usuari; s'identificarà tant l'entrada com l'eixida. El sistema serà capaç de comunicar-se amb el *backend* necessari per verificar que l'usuari pot accedir al recinte, a més de proporcionar informació en temps real de l'aforament actual. Per tant, també s'ha de desenvolupar un programa capaç d'identificar les polseres, i comunicar-se amb el *backend*. Altres avantatges que suposa aquesta tecnologia, és que s'evita la revenda d'entrades, ja que les polseres van associades a l'assistent de l'esdeveniment. Els objectius seran el desenvolupament de l'aplicació mòbil, del *backend* i el desenvolupament del sistema de lectura. El pla de treball és seguir una metodologia àgil basada en *sprints* de tres setmanes, acordant l'inici de cada *sprint* quins components i treballs es van a dur a terme, i a la finalització del sprint, fer una reunió, per veure quins punts s'han completat i quins no, a més de mostrar-li les successives progressions del producte.

Paraules clau: control d’aforament, aplicació mòbil, RFID, backend, app.

Summary

The aim of this project is providing a concrete event with a special control and identification system for users and visitors. The mobile app is assigned to the workers to know about the most important information of the event in real time, such as the amount of people which is inside of the enclosure at that moment. This way, the authorities would be able to know if the assistant limit is being respected. The system is based on a mobile app which is available in IOS and Android, and developed with Google GWTP technology (throughout MVP model, controller mode) and there is an app runner in which the app works and where the data base is located. The app basically consists on a web navigator, but it has been passed through Cordova tool, which, with the required setting, allows us to treat the embedded navigator as a native app, without being necessary to develop the app for each system. Regarding the users' access control, we count on a lector with the ability to identify RFID passive bracelets which will be expected to be worn by the assistants; they will be identified in the entrance as much as in the exit. The system will be able to communicate with the necessary backend to verify that the user can get into the enclosure, and also it will supply with real time information about the current capacity. Therefore, it's necessary to develop a program which has to be able to identify the bracelet and to communicate with the backend. This kind of technology involves other advantages, such as avoiding tickets resale, thanks to the connection between the assistants and their bracelets. The objectives will be the development of the mobile app, the backend and the lecture system. The working plan is focused on following an agile methodology based on three-weeks sprints, guided by the agreement about which kind of components and projects are going to be accomplished. After that, in the end of the sprint, a reunion will be celebrated in order to know whether or not the aims have been achieved, as well as showing the actual improvement of the product.

Keywords: capacity control, mobile app, RFID, backend, app.

Tabla de contenido

Índice de figuras

1. Introducción

1.1 Motivación

El principal motivo de desarrollar este Trabajo Fin de Grado (TFG) surge tras la finalización del proyecto WiSafe en la empresa vBote en la cual he desarrollado prácticas de empresa. El objetivo de WiSafe es controlar la entrada y salida a recintos como por ejemplo colegios o parques de atracciones de niños y niñas, para que de esta manera los padres conozcan dicha información en tiempo real.

El proyecto que aquí se presenta se basa en la misma tecnología, la cual consta de una app móvil, un sistema de lectura junto un dispositivo capaz de manejar los eventos y un *backend* que se encarga de recibirlos y tratarlos.

Tras la finalización del proyecto y con la creciente celebración de eventos multitudinarios como festivales de música, junto con la normativa de aforos que será de obligatorio cumplimiento a partir de septiembre de 2016 en la Comunidad Valenciana, que obliga a los promotores del evento a tener un dispositivo técnico para disponer del aforo exacto en eventos que superen los 2000 participantes. (normativa presente en el decreto 143/2015 artículo 185, ver Anexo 1).

En la memoria queda el trágico accidente del Madrid Arena, ocurrido en 2012, donde perdieron la vida 5 chicas que disfrutaban de la celebración de la fiesta de Halloween, pero que debido a una acumulación de negligencias, como la venta por encima del aforo permitido de entradas para el evento, tuvo tan fatídico desenlace. Si se hubiera dispuesto de soluciones tecnológicas como la identificación por radiofrecuencia, las autoridades y cuerpos de seguridad del estado hubieran sido conocedores en todo momento del aforo actual del recinto, pudiendo haber cancelado la entrada de mas asistentes al recinto, o cancelar dicho evento por incumplir las normas establecidas en el permiso para dicho evento.

Además, otro de los alicientes para adoptar tal tecnología a los promotores de eventos es la seguridad que muestra a las autoridades a la hora de conceder licencias para celebrar sus eventos, ya que se proporciona una herramienta a las autoridades de modo que en cualquier momento de su celebración son conocedores del aforo y su estricto cumplimiento.

Otra posibilidad que brinda la tecnología de radiofrecuencia a los promotores de eventos es la capacidad de reforzar su marca como festival o evento particular, ya que cada asistente está registrado previamente de su entrada al recinto, con la entrada que ha adquirido previamente en plataformas digitales, donde si lo desea, puede introducir sus datos de perfiles en redes sociales, de modo que si lo autoriza, cuando el asistente entre al evento o este situado en uno de los escenarios del festival, se puede publicar de manera automática en su perfil de la red social autorizada un post donde muestre que artista esta disfrutando en ese momento.

Por último, son muchos los eventos que ofrecen la posibilidad de pagar sin efectivo ni el canjeo de dinero por tokens (moneda de los festivales) de modo que el asistente

no se debe de preocupar por hacer largas colas en hacer el canjeo por tokens, o de preocuparse de su dinero guardado en una cartera que al fin y al cabo puede extraviar. La identificación por radiofrecuencia ofrece la posibilidad de asociar un monedero virtual donde el asistente previamente a la celebración del evento puede recargar con el saldo que quiera, por lo que no ha de preocuparse por su cartera, todo queda asociado en su pulsera. Esto también ofrece una ventaja a los promotores del evento, pues les brinda una liquidez previa al evento, de modo que pueden prever de una manera más exacta los recursos que deberán adquirir. Además, durante el festival, son conocedores en todo momento del consumo de cada una de las barras del festival, de modo que podrán reabastecer cada barra en caso de que estén cerca de agotar las existencias mas fácilmente.

Por tanto, la identificación por radiofrecuencia de los asistentes a un evento, además de aportar un mejor y más estricto control del aforo, brinda ventajas a los asistentes como a los promotores que adopten esta tecnología de control.

1.2 Objetivos

El objetivo principal de este TFG es dar servicio de manera ágil y accesible a los promotores de eventos ante la necesidad de disponer del aforo de los mismos en tiempo real para el cumplimiento de la ley, de una manera sencilla y económica, de modo que no suponga un gran desembolso ni adaptación costosa para el evento.
Este objetivo principal se divide en tres de la siguiente forma:

- Desarrollar una app que sirva de interfaz a los trabajadores del evento y a las fuerzas de seguridad, de manera que sean conocedores de una manera sencilla del aforo.
- Ofrecer un lector y pulseras económicas para la solución.
- Desarrollar un *backend* donde procesar todas las entradas y salidas del evento, y que se pueda adaptar a cualquier clase de pulsera/lector que se desee distribuir en el evento.

De esta manera, será posible conocer el aforo mediante un Smartphone con conexión a internet, a través de la app, además de facilitar los trámites de acreditación que deben realizarse para efectuar el acceso a un evento.

El bajo coste del hardware y de las pulseras permite cumplir con el objetivo de ofrecer una solución económica para el cliente.

Y por último, crear un sistema que sea adaptable a los distintos modelos de pulseras o identificadores que deberá portar el asistente, como puede ser pulseras con código de barras, que cumple con el segundo y tercer objetivo, ofrecer una solución adaptable y económica.

1.3 Estructura del documento.

El documento se estructura en un total de siete capítulos junto un anexo donde se amplia la información referida.

En el segundo capítulo se presenta una serie de alternativas actuales y los motivos que diferencia el producto frente otras soluciones.

En el tercer capítulo se describe el alcance y ámbito del proyecto, junto la metodología a seguir para su desarrollo.

El cuarto capítulo describe las tareas que implica cada uno de los componentes del proyecto.

El quinto, sexto y séptimo capítulo muestran cada una de las actividades y gestión de los tres *sprints* en que se ha dividido el proyecto. Un *sprint* por cada componente que conforman el sistema.

El capítulo octavo presenta las conclusiones una vez finalizado el proyecto, junto las posibles ampliaciones del sistema.

Tras las referencias bibliográficas , se muestran los anexos.

En el [Anexo I] se presenta la ley de control de aforo referente a la Comunidad Valenciana.

El [Anexo II] muestra una guía para el uso de cada una de las funcionalidades que brinda la aplicación.

2. Estado del arte

Actualmente, no existe ningún tipo de tecnología similar la cual ofrezca tantas posibles aplicaciones como la mencionada de pagar con la misma pulsera que se accede al evento, además de suponer costes superiores para el control del aforo.

2.1 Alternativas

Una de las primeras alternativas existentes serían los tornos de acceso, como los que hay presentes en gimnasios y centros deportivos, los cuales disponen de lectores de huellas o también por identificación por radiofrecuencia, los cuales únicamente niegan o conceden el paso al usuario como funcionamiento normal. Bien cierto es que se podría adaptar para registrar el aforo de un evento, por lo que debería disponerse de 2 tornos, uno para la entrada y otro para la salida. El primer punto en contra de esta solución es su elevado coste (rondan los dos mil euros por torno). Además, se deben fijar a tierra para asegurar su estabilidad, además de ser aparatos voluminosos que no facilitan el montaje a la hora de festivales que no suelen superar la duración de 3 días.

Figura 1: Torno

Una alternativa similar a la anterior descrita, serían los sensores laser, que asemejan su funcionamiento a las puertas de los ascensores. Se emite un haz de luz el cual llega a un sensor situado en la trayectoria del haz de luz. Si este haz se ve interrumpido, se produce un evento el cual en el caso del ascensor consistiría en la apertura de la puerta. En nuestro caso, al producirse el evento sumaría o restaría una unidad del aforo según el lector se encuentre en la entrada o salida. Pero se repite la situación anterior, se debe disponer de unas torres donde colocar el laser y el sensor, lo cual requiere de su correcta instalación anclándolo al suelo. Además, no es un sistema con un cien por cien de acierto, por lo que el aforo reflejado podría distar del real (el sistema alcanza un noventa y nueve por ciento de acierto).

Figura 2: sensor laser

Alfombras de presión: este tipo de herramienta, permite identificar el numero de personas que la atraviesan a través de una serie de sensores que lleva incorporada la alfombra, y mediante un software que incorpora es capaz de mostrar el numero de personas que la atraviesan. Es una tecnología reciente, por lo que su coste es bastante elevado comparándolo con las otras alternativas. También, para evitar lecturas erróneas, los asistentes deben atravesar la alfombra ordenadamente, lo cual en ámbitos como festivales suele resultar un tanto caótico.

Figura 3: alfombra de presión

Cámaras térmicas: las cámaras térmicas son cámaras destinadas a contar el número de personas que pasan por un determinado lugar, mediante el software correspondiente. La principal desventaja que implica son su montaje, ya que requieren de un punto fijo situado a una altura mínima de 2.5m, en la cual la lectura no es certera, por lo que se deben elevar bastante más y no abarca una gran superficie de lectura, por lo que en eventos donde se requieran puntos de acceso amplios, se debe tener atención en los solapes entre cámaras, para evitar la duplicidad de lectura.

Figura 4: cámara térmica

2.2 Propuesta

Se trata de un sistema innovador al brindar un gran abanico de posibilidades, como el mencionado método de pago con la misma pulsera o la publicación de posts en las redes sociales según la posición del asistente en el evento. El coste de cada lector no supera los cincuenta euros y el coste de cada pulsera no supera el euro por unidad. Los lectores no necesitan instalarse ni anclarse, solo necesitarían una conexión Ethernet o Wifi para poder comunicarse con el *backend* y de una fuente de alimentación.

Figura 5: pulseras NFC

El lector que se propone está formado por una Raspberry pi 3 [Raspberry], un mini ordenador bastante popular debido a su pequeño tamaño y su reducido coste (unos cuarenta euros por unidad, menos en casos de grandes pedidos), al que se le conecta el lector RFID y mediante un software de control, se configura para que registre cada lectura con el *backend* destinado a procesar cada petición de entrada o salida al recinto.
La Raspberry ofrece la posibilidad de conectarse a la red por Ethernet o por Wifi, dependiendo de las necesidades del evento. Si se desea tener un lector portátil, únicamente se debería conectar la Raspberry a la red Wifi, un pequeño altavoz o auricular para conocer el resultado de la operación y una batería para alimentar el lector.

Figura 6: Raspberry Pi 3 junto con lector NFC

Incluso cabe la posibilidad de conectar cada lector a un monitor, de modo que se pueda validar la entrada del asistente al evento, impidiendo la entrada en el caso de que ya se haya entrado al evento previamente, lo cual supondría un fraude por parte del asistente.

2.3 Ventajas

Es bastante habitual ver como asistente a eventos encuentran recovecos por donde pasarse las pulseras, de modo que una persona que se encuentre fuera del evento, puede intentar acceder al evento con la misma pulsera de un compañero. Mediante este sistema, al identificar cada persona en su entrada y quedar registrado, no podría volver a entrar al evento si no ha sido registrada su salida del evento, por lo que este tipo de fraude no se puede producir. Al conectarle una pantalla al lector, este al enviar una petición de entrada, puede mostrar si la entrada es válida o en caso contrario, mostrar un error informando al operario que dicha persona no puede entrar al recinto.

Además, esta solución implica un cien por cien de acierto, ya que se lee cada una de las pulseras de los asistentes al recinto, sin dejar espacio al error. Lo que aporta un nivel de fiabilidad que muestra el aforo en tiempo real en todo momento mediante la aplicación, la cual puede ser utilizada por los operarios del evento o los mismos cuerpos de seguridad del estado para asegurar el cumplimiento del aforo establecido. Esta posibilidad y fiabilidad es lo que hace que se diferencie el producto respecto a las otras alternativas para el control de aforo.

Otra ventaja del sistema, es el enfoque de implementación de la aplicación. Al tratarse de una aplicación híbrida (combinación de aplicación nativa y aplicación web), permite desarrollar la aplicación móvil (Android o IOS) una vez, sin mucho esfuerzo gracias al framework Apache Cordova [CORDOVA], que crea un navegador embebido dentro de la aplicación, apuntando a la aplicación web que despleguemos. Lo cual supone una ventaja frente a cambios que solicite el cliente en la aplicación, o futuras mejoras a nivel de interfaz, ya que todos estos cambios se realizan en el servidor y no en la propia aplicación del dispositivo móvil, por lo que no se deberá actualizar la aplicación del dispositivo por cada cambio que se realice, evitando posibles errores.

Resumiendo, la principal ventaja que ofrece el sistema es la siguiente: Control de aforo en tiempo real con una fiabilidad del cien por cien a bajo coste y con acceso a los datos en todo momento mediante la aplicación.

3. Propuesta

Este capitulo tiene como propósito definir el alcance del proyecto y todas las funcionalidades que se van a integrar para la primera versión de los tres componentes que lo conforman así como las posibles aplicaciones que implica el proyecto.
Además, se indica la metodología ágil que va a seguir durante el desarrollo.

3.1 Visión y análisis del proyecto

La posible aplicación del producto son las salas y eventos que se celebren con un aforo previsto superior a 2.000 personas, ya que desde septiembre de 2016 están obligados por ley [RD-143/2015] a disponer de medios técnicos para disponer en todo momento del aforo presente dentro del recinto, de manera que si la autoridad competente lo exige poder mostrarlo con total seguridad y certeza.

Actualmente, son muy numerosos los eventos que se celebran en la Comunidad Valenciana relacionados con festivales que superan esta cifra de aforo, de modo que deben llevar un control riguroso para controlar tales cifras de asistentes. Pero además del ámbito de festivales, también son numerosos los clubs nocturnos en la ciudad de València que superan estas cifras de asistentes. Y por último lugar, las numerosas ferias y congresos que se celebran relacionados con distintos ámbitos.

Por tanto, el producto pretende dar servicio a esta nueva necesidad que deben implementar en sus eventos todos los organizadores y propietarios de los recintos donde se celebre dicha actividad. De esta manera, serán conocedores en todo momento del aforo presente en el evento.

El proyecto está organizado en tres *sprints*, debido a la estructura en la que se compone el proyecto, tres componentes. La primera de ellas, el desarrollo del frontend/backend de la aplicación web. La segunda etapa corresponde al desarrollo de la app para iOS y la etapa final a la integración del software en los lectores NFC sobre Raspberry Pi 3.

- Fase de desarrollo de la aplicación web. Primer *sprint*: En esta fase se desarrollará una primera versión del mismo, a partir de un mapa de navegación de la aplicación y su correspondiente boceto de interfaz.

- Fase de desarrollo aplicación iOS. Segundo *sprint*: Esta fase se centrará en integrar las funcionalidades de la aplicación web en la app móvil.

- Fase de integración del sistema de lectura y cierre de proyecto. Tercer *sprint*: Esta fase se centrará en la integración del sistema en su totalidad, pruebas de funcionalidad y cierre del mismo.

Análisis de la estructura

a) App: Cordova app junto al Front-end proporcionado por GWTP [GWTP].
b) *Backend*: *Backend* desarrollado en GWTP junto al Datastorage de AppEngine donde se despliega la Webapp.
c) Lectores: Tarjeta NFC/RFID acoplada a una Raspberry Pi para la gestión de lectura y envío de los eventos al *backend* el cual se encargará de validar el evento y gestionar el aforo.

3.2 Especificación de requisitos

- Datos en tiempo real: para que el proyecto cumpla con su función debe estar disponible independientemente del nivel de carga que tenga y actualizar los datos continuamente en la parte del cliente.
- Seguridad: la aplicación debe constar de un acceso controlado por usuario y contraseña previamente dado de alta por el administrador del sistema.
- Escalabilidad: Debe escalar fácilmente para poder dar servicio ininterrumpido independientemente del nivel de carga.
- Arquitectura modular: Cada uno de sus componentes debe ser independiente, especialmente la capa de lectura de etiquetas NFC, por si se quiere variar el sistema de lectura a otras tecnologías como códigos QR.
- Facilidad de uso: la aplicación debe mostrar una interfaz clara y sencilla para facilitar la labor de los trabajadores del evento.

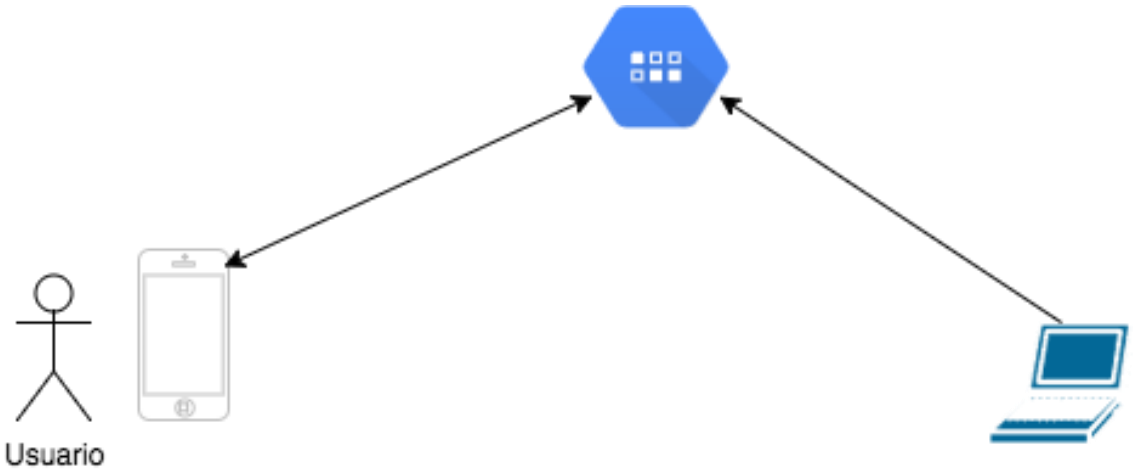

Figura 7: Arquitectura del sistema.

3.3 Especificaciones técnicas

El desarrollo consta de tres capas principales como se muestra en la Error! Reference source not found. que componen el sistema.

La primera de ellas es la app móvil, que se encarga de extender las funcionalidades del *frontend*. En este caso el framework utilizado es Apache Cordova, la versión *open source* de Adobe Phonegap.

La segunda capa es el *frontend/backend* desarrollado en GWTP y desplegado en App Engine. Esta tecnología permite desplegar la Webapp proporcionado un servicio único donde se desarrolla el *frontend/backend* en un único proyecto junto con el modelo de persistencia de objetos.

La tercera capa es el lector encargado de enviar los eventos de entrada y salida del evento. El lector es una tarjeta NFC junto a una Raspberry Pi 3 encargada de manejar el lector y enviar los eventos al *backend* mediante un *Servlet*.

4. Backlog

Para iniciar el desarrollo del *sprint*, primero procedemos al análisis de cada tarea necesaria para la implementación del sistema. Cada *sprint* está dividido por cada unidad funcional del sistema.

Para ello, primero calculamos el total de horas disponibles para el desarrollo de las tareas. Hemos definido los *sprint* de tres semanas cada uno, con un total cuatro horas diarias, lo cual hace un total de sesenta horas por *sprint*, exceptuando el tercer *sprint* que consta de veintiocho horas.

TAREA	**DESCRIPCIÓN**
Análisis de los casos de uso – 5h	En esta tarea se debe identificar cada una de las acciones que se van a implementar para cumplir con los objetivos marcados, y dotar de una interfaz sencilla a la vez de útil para el usuario. Para la identificación de cada caso de uso, utilizaremos una plantilla con los campos Actor, Precondición, Postcondición y Descripción de cada uno de los casos de uso, de modo que quede completamente descrita cada una de las necesidades a cubrir. La tarea completa tiene una estimación de cinco horas.
Bocetos de la interfaz de usuario – 6h	Creación de los bocetos de cada una de las pantallas junto el mapa de navegación del total de la aplicación web, para su posterior validación con el cliente, que en este caso excepcional coincide el *product owner* con el equipo de desarrollo. Una vez validados los bocetos se puede proceder a la implementación de cada una de las pantallas, por lo que esta tarea tiene mas prioridad que la implementación de las pantallas hasta tener la aprobación de los bocetos. Una tarea que sí que se puede adelantar a la aprobación del cliente podría ser la lógica de la aplicación, pues pese algún cambio menor, ofrecerá una interfaz de métodos independientes a la interfaz de usuario con la que interactuar.

	Por ejemplo, independientemente del aspecto que vaya a tener el acceso al área privada, sabemos que será necesario de un método el cual valide los datos de usuario y contraseña introducidos, por lo que se puede adelantar esta tarea. No se va a aplicar en este caso ya que no aplica, pero sería una alternativa factible. Esta tarea tiene una estimación de seis horas en total.
Identificación de las entidades de persistencia – 3h	Para esta tarea se deben identificar cada una de las entidades necesarias para poder llevar el conteo del aforo actual, los tags que están registrados junto su estado (entrada/salida) junto un conjunto de usuarios e información importante relativa al evento, como el aforo máximo permitido. Al elegir Google Cloud Data Store, esto nos permite hacer una abstracción basándonos totalmente en objetos para su definición, lo cual facilita la implementación de la lógica de negocio y su persistencia. Esta tarea tiene una estimación de tres horas en total.
Implementación de las interfaces de usuario – 20’5 h	Esta tarea engloba varias tareas, divididas por cada una de las pantallas que conforman la aplicación web de la siguiente manera: • **Pantalla de inicio** (tres horas): pantalla donde el usuario debe introducir usuario y contraseña para acceder a la parte privada de la aplicación. En caso de no recordar la contraseña, puede iniciar el proceso de recuperar contraseña desde esta pantalla. • **Pantalla principal** (cuatro horas): pantalla donde se muestra el conteo de aforo y se puede acceder

	a otras pantallas del área privada como los ajustes de sesión o los mensajes que pueda haber recibido el usuario. • **Acceso a ajustes** (media hora): pequeño icono que despliega un menú donde se da las opciones de cerrar la sesión del usuario, o bien acceder a los ajustes de usuario. • **Mensajes** (tres horas): pantalla donde se muestran todos los mensajes que pueda haber recibido el usuario por parte del administrador de la aplicación. • **Ajustes** (tres horas): Interfaz donde cambiar la contraseña y desactivar la opción de mantener la sesión abierta. • **Recuperar contraseña** (dos horas): pantalla la cual se accede desde la pantalla de inicio con el propósito de cambiar la contraseña si no se recuerda la contraseña del usuario, por lo que se debe introducir el email del usuario, el cual recibirá un enlace en su correo para cambiar la contraseña en una nueva pantalla. • **Recuperar contraseña** (dos horas): Esta pantalla es la segunda parte del proceso de recuperar contraseña mencionado en el punto anterior. Una vez validada la integridad del enlace enviado, se muestra el formulario para cambiar la contraseña. • **Enlazar las vistas según el mapa de navegación** (tres horas): una vez finalizada la implementación de cada una de las pantallas, hay que enlazar cada pantalla mediante los métodos de acceso definidos, como los botones de navegación o

	el proceso de autenticación, que en caso de ser correcto, se dirige a la pantalla principal del área privada.
Implementación de la lógica de negocio – 25,5 h	Esta tarea comprende un subconjunto de tareas relacionadas con la lógica para procesar las peticiones y mostrarlas correctamente, además del control de las sesiones de usuario que pueden acceder a la plataforma, como el envío de los correos electrónicos para la recuperación de contraseña. Son las siguientes tareas: • **Definición de las clases/persistencia** (seis horas): definición de todas las clases que van a intervenir en la lógica de negocio y su representación en la base de datos. • **Definición e implementación de las llamadas RPC** (cinco horas): diseño e implementación de todos los métodos necesarios para implementar las funciones como el acceso a la parte privada de la aplicación, la cual valida los datos introducidos por el usuario. • **Definición e implementación de los Servlet** (cinco horas): principalmente la aplicación web necesita dos *Servlet*, uno de ellos para aumentar o disminuir el número de aforo presente, y otro para la validación del enlace enviado por el correo electrónico de recuperación de contraseña. • **Implementación de la mensajería** (tres horas y media): Implementación de la construcción y envío de mensajes como el correo electrónico para recuperar la contraseña. Para ello

	definiremos una base genérica la cual extenderemos según la necesidad de cada tipo de mensaje.
Despliegue en App Engine de la primera versión – 3 h	Primer despliegue sobre la plataforma donde se va a ejecutar la aplicación web, para descubrir posibles fallos y asegurarnos que todo funciona como se había diseñado, ya que por ejemplo los correos electrónicos únicamente se envían una vez desplegado en App Engine, en desarrollo únicamente imprime los logs por consola, por lo que no nos podemos asegurar que funciona como debe hasta que se despliega sobre la plataforma.
Testing general de la interfaz y Servlets – 3 h	Testeo manual de las principales funciones que ofrece la aplicación, así como los dos *servlet* de validación de enlace de recuperar contraseña, y la suma o resta de aforo debido a un evento de entrada o salida. Al tratarse de un framework que ya he utilizado anteriormente, las estimaciones pueden parecer un poco justas, pero se debe a la experiencia adquirida junto a la facilidad que ofrece esta herramienta para crear interfaces de calidad y dotarles de funcionalidad con muy poco esfuerzo en su desarrollo. Para el desarrollo de las tareas, se va a seguir el orden presentado tal y como se han descrito.
Instalación Apache Cordova en Mac – 4h	Descarga e instalación del framework para el desarrollo de la aplicación.
Instalación Xcode – 3h	Descarga e instalación de la herramienta desarrollo necesaria para compilar la aplicación para IOS.
Boceto pantalla e iconos – 5h	Creación de los bocetos para la aprobación por parte del cliente. En este caso desarrollador y cliente son la misma figura, por lo tanto la aprobación es acto seguido a su finalización.

Implementación pantalla de carga – 5h	Implementación de la pantalla de carga que muestra Cordova para esconder la pantalla blanca que aparece mientras se carga la aplicación web, ya que Cordova en una definición simple, es un navegador web donde se muestra la aplicación web.
Creación de la aplicación – 3h	Proceso a seguir para crear la estructura del proyecto y primeras configuraciones de los archivos encargados de la carga y mensajería entre aplicaciones.
Creación de los elementos gráficos – 8h	Diseño final de todos los elementos gráficos como los iconos previamente aprobados por el cliente, los cuales deben tener unos tamaños concretos definidos por Apple.
Creación del splash de carga – 5h	Implementación de la pantalla de carga previamente definida, con los métodos Javascript para conocer el estado de la carga de la aplicación web, y de esta manera ocultarse para mostrar la aplicación, o mostrar en caso de error un mensaje que notifique al usuario el problema.
Desarrollo de la comunicación entre aplicaciones – 16h	Para dotarle de completa funcionalidad, se debe crear una serie de métodos para permitir la comunicación entre ambas aplicaciones, como por ejemplo el estado de la carga.
Carga de los elementos gráficos – 5h	Añadir todos los elementos gráficos creados como los iconos y que de esta manera queden enlazados al proyecto.
Despliegue en dispositivo físico – 1h	Proceso de preparación de la aplicación en un dispositivo físico real, ya que el testeo en el emulador puede ocultar errores que se producen en un terminal físico. Por ejemplo, hay gestos o movimientos sobre la pantalla que podrían producir un error en la aplicación.
Testing manual – 5h	Testeo manual de la aplicación basado en los test anteriormente pasados, para asegurar que se mantienen todas las funcionalidades intactas, además de comprobar que el aspecto y comportamiento de la aplicación móvil es

	el deseado.
Instalación del sistema operativo (Raspbian) – 5h	Se debe instalar el sistema operativo en una tarjeta micro-sd desde la cual se ejecuta en la Raspberry. Para ello se ha seleccionado la distribución mas completa de Linux para Raspberry, Raspbian, la cual proporciona un entorno gráfico en el caso de que se necesite, lo cual no aplica al proyecto.
Configuración Raspberry – 3h	En esta tarea se incluye la actualización del sistema operativo y de la última versión de Python, que es el lenguaje en el cual se va a desarrollar el controlador del lector.
Desarrollo del controlador del lector – 5 h	Desarrollo del programa encargado de la lectura de las pulseras mediante la API que proporciona la tarjeta mediante el lenguaje Python.
Pruebas de lectura – 3h	Una vez desarrollado el lector, se deben hacer una serie de pruebas con las pulseras que van a ser leídas por el sistema, para comprobar que su lectura es correcta y sin problemas.
Configuración de arranque – 3h	Una vez desarrollado el programa y comprobado que las lecturas son correctas y precisas, se procede a la configuración para que la Raspberry ejecute el programa de lectura cuando esta se inicia, sin cesar su ejecución, porque debe estar en todo momento disponible.
Pruebas programa individual – 3h	Tras configurar la Raspberry, se deben pasar una serie de pruebas a nivel local, sin dirigir los eventos todavía a la aplicación web, únicamente mostrando los resultados por consola, para comprobar que la Raspberry se inicia ejecutando el programa de lectura, y las lecturas se producen correctamente y sin interrupciones.
Pruebas del sistema completo – 5h	Para asegurar la integridad del sistema y su correcto funcionamiento como conjunto, se realizarán una serie de pruebas con pulseras dadas de alta en el

	sistema, para asegurar que el conteo se produce correctamente, y que devuelva error en el caso que se quiera entrar 2 veces con la misma pulsera. Los cambios en el aforo se deben ver reflejados automáticamente en el sistema.
Entrega del sistema completo – 1h	Una vez finalizado y comprobado el funcionamiento del sistema, se procede a la entrega y formación del producto al cliente.

5. Primer sprint

En este *sprint* nos centramos en el desarrollo de la aplicación web que se encargará de procesar las peticiones mediante un Servlet, y ofrecer la interfaz de usuario la cual consumirá la aplicación móvil.

5.1 Inicio del *sprint* – Selección de tareas

1. Análisis de los casos de uso – 5h
2. Bocetos de la interfaz de usuario – 6h
3. Identificación de las entidades de persistencia – 3h
4. Implementación de las interfaces de usuario – 20’5 h
   4.1. Pantalla de inicio (tres horas)
   4.2. Pantalla principal (cuatro horas)
   4.3. Acceso a ajustes (media hora)
   4.4. Mensajes (tres horas)
   4.5. Ajustes (tres horas)
   4.6. Recuperar contraseña (dos horas)
   4.7. Recuperar contraseña segunda parte (dos horas)
   4.8. Enlazar las vistas según el mapa de navegación (tres horas)
5. Implementación de la lógica de negocio – 25,5 h
   5.1. Definición de las clases/persistencia (seis horas)
   5.2. Definición e implementación de las llamadas RPC (cinco horas)
   5.3. Definición e implementación de los Servlet (cinco horas)
   5.4. Implementación de la mensajería (tres horas y media)
6. Despliegue en App Engine de la primera versión – 3 h
7. Testing general de la interfaz y Servlets – 3 h

5.1.1 Análisis de los casos de uso

Acceso parte privada

Actor	Usuario
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema para poder acceder a la zona privada.
Postcondición	Si las credenciales introducidas son correctas el usuario accede a la parte privada de la app. En caso contrario se notifica que los datos introducidos no son correctos.
Descripción	1. El usuario inicia la aplicación. 2. Tras la pantalla de carga se muestra la pantalla de inicio con los campos de correo electrónico y contraseña para acceder a la parte privada. 3. Una vez introducidos los datos por parte del usuario y pulsar el botón para acceder, si los datos son correctos se muestra la zona privada. En caso contrario se notifica para que el usuario compruebe los datos introducidos. Si el usuario ha pulsado el botón con los dos campos vacíos o uno de ellos, se muestra un mensaje para que complete el campo.

Mantener sesión iniciada

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema para poder acceder a la zona privada, acceder a la zona privada y seleccionar esta opción.
Postcondición	Si el usuario cierra la aplicación y posteriormente la vuelve a abrir, no deberá volver a introducir los datos de nuevo. En la base de datos se verá reflejado el cambio inmediatamente una vez confirmado el ajuste de mantener la sesión iniciada.
Descripción	1. El usuario inicia la aplicación. 2. Tras la pantalla de carga se muestra la pantalla de inicio con los campos de correo electrónico y contraseña para acceder a la parte privada. 3. Una vez introducidos los datos por parte del usuario y pulsar el botón para acceder, si los datos son correctos se muestra la zona privada. 4. Una vez se haya accedido a la zona privada, el usuario debe pulsar sobre el botón situado en la barra superior derecha. 5. Se muestra un menú donde debe seleccionar la opción 'ajustes'. 6. Una vez pulsado se muestra la pantalla de ajustes donde el usuario debe pulsar sobre el botón bimodal para

	mantener la sesión iniciada o dejar de mantener la sesión iniciada.

Cambiar contraseña

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema para poder acceder a la zona privada.
Postcondición	El cambio de contraseña se verá reflejado inmediatamente en la base de datos tras confirmar el proceso en la aplicación si los datos introducidos eran los correctos.
Descripción	1. El usuario inicia la aplicación. 2. Tras la pantalla de carga se muestra la pantalla de inicio con los campos de correo electrónico y contraseña para acceder a la parte privada. 3. Una vez introducidos los datos por parte del usuario y pulsar el botón para acceder, si los datos son correctos se muestra la zona privada. 4. Una vez se haya accedido a la zona privada, el usuario debe pulsar sobre el botón situado en la barra superior derecha. 5. Se muestra un menú donde debe seleccionar la opción 'ajustes'. 6. Una vez pulsado se muestra la pantalla de ajustes donde el usuario debe introducir su actual contraseña seguido de la nueva contraseña, con la opción de mostrar la contraseña introducida para verificar que es la que el usuario tiene pensado cambiar. Una vez introducidos los datos debe pulsar el botón 'cambiar'. Si el usuario introduce como contraseña actual una contraseña errónea, se muestra un mensaje de error para que compruebe los datos introducidos. Si el usuario deja los dos campos vacíos o alguno de ellos vacío se muestra un mensaje de error para que complete los campos vacíos.

Acceso mensajes

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema para poder acceder a la zona privada.
Postcondición	En el caso de que el usuario haya leído mensajes por leer, los mensajes cambiaran su estado a leídos en la base de datos y el icono de mensajes que debía estar rojo antes de acceder a la

	pantalla de mensaje, pasa a estar de color blanco.
Descripción	1. El usuario inicia la aplicación. 2. Tras la pantalla de carga se muestra la pantalla de inicio con los campos de correo electrónico y contraseña para acceder a la parte privada. 3. Una vez introducidos los datos por parte del usuario y pulsar el botón para acceder, si los datos son correctos se muestra la zona privada. 4. Una vez se haya accedido a la zona privada, el usuario debe pulsar sobre el icono situado en la barra superior izquierda. [caso 1] Si el usuario tiene mensaje por leer, el icono de mensajes se muestra de color rojo. [caso 2] Si el usuario no tiene mensajes o los mensajes que tiene ya han sido leídos, el color del icono es blanco. 5. [caso 1] Se muestran los mensajes en color rojo si no han sido leídos por el usuario. [caso 2] Si el usuario no tiene ningún mensaje aparece un texto donde informa que el usuario no tiene mensajes. [caso 3] Si el usuario tiene mensajes pero ya han sido leídos previamente, los mensajes son mostrados pero sin ser destacados.

Recuperar contraseña – A

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema de modo que pueda recibir el correo electrónico con el enlace para poder recuperar la contraseña.
Postcondición	Si el usuario está dado de alta, recibirá un correo electrónico con el enlace para acceder a la siguiente pantalla para recuperar la contraseña. En caso contrario, no se recibirá ningún correo electrónico si el usuario no está dado de alta en el sistema.
Descripción	1. El usuario inicia la aplicación. 2. Para acceder a la pantalla de recuperar contraseña el usuario debe pulsar sobre el enlace “He olvidado mi contraseña”. 3. A continuación se muestra la pantalla de recuperar contraseña, donde el usuario debe introducir el correo electrónico de su cuenta y pulsar el botón de recuperar

	contraseña. 4. Si el campo no está vacío y es una dirección de correo valida, aparece un mensaje notificando que el correo electrónico ha sido enviado, esté dado de alta o no. 5. La aplicación vuelve a la pantalla de inicio de sesión.

Recuperar contraseña – B

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema de modo que pueda recibir el correo electrónico con el enlace para poder recuperar la contraseña. El usuario debe haber realizado el primer paso de recuperar contraseña y de esta manera recibir el correo para cambiar la contraseña.
Postcondición	La contraseña deberá cambiar si el link que se ha recibido en el correo es el correcto y se verá reflejado el cambio reflejado inmediatamente en la base de datos tras finalizar el proceso.
Descripción	1. El usuario debe de pulsar el enlace que ha recibido en su correo electrónico tras hacer el caso de uso 'Recuperar contraseña – A'. 2. Tras pulsar se mostrará la pantalla donde el usuario debe introducir la nueva contraseña. 3. Una vez introducida la nueva contraseña, el usuario debe pulsar el botón para confirmar los cambios. 4. Si el proceso ha finalizado satisfactoriamente se muestra un mensaje de éxito y se navega hacia la pantalla de acceso a la parte privada. En caso de reutilizar un link se muestra un mensaje de error, ya que el enlace de recuperación solo es válido una vez.

Cerrar sesión

Actor	Usuario registrado
Precondición	El usuario debe estar dado de alta por parte del administrador del sistema para poder acceder a la zona privada. El usuario debe acceder a la parte privada.
Postcondición	Si el usuario inicia de nuevo la aplicación, esta mostrará la pantalla inicial donde debe introducir las credenciales, pese a que haya fijado la opción de mantener sesión iniciada, la cual pasa a

	estar desactivada cunado se cierra la sesión.
Descripción	1. El usuario inicia la aplicación. 2. Tras la pantalla de carga se muestra la pantalla de inicio con los campos de correo electrónico y contraseña para acceder a la parte privada. 3. Una vez introducidos los datos por parte del usuario y pulsar el botón para acceder, si los datos son correctos se muestra la zona privada. 4. Una vez se haya accedido a la zona privada, el usuario debe pulsar sobre el botón situado en la barra superior derecha. 5. Se muestra un menú donde debe seleccionar la opción 'cerrar sesión'. 6. Tras haber pulsado se muestra un dialogo con la opción de continuar con el proceso de cerrar sesión o cancelar el proceso de cerrar sesión. 7. El usuario debe pulsar el botón de 'Aceptar' cerrar la sesión. 8. Una vez finalizado el proceso se muestra la pantalla de inicio de sesión.

Tarea completada dentro del tiempo establecido. No presentaba ningún tipo de riesgo ya que se trata únicamente de la identificación de los casos de uso.

5.2.2 Bocetos de la interfaz de usuario

En esta tarea se va a describir cada una de las pantallas que conformarán la interfaz de usuario. Como ya se ha mencionado, al tomar el papel de desarrollador y cliente al mismo tiempo, la aprobación se da una vez finalizada la tarea.

Todos los componente que se utilizarán para la implementación de la interfaz de usuario de la aplicación son de *Materialize*, un framework que permite el desarrollo de interfaces móviles para WebApps basado en *Material Desing* creado por Google. Estos componentes implementan en armonía HTML5, CSS3 y Javascript proporcionando una excelente experiencia de usuario muy similar a la de una aplicación nativa. Todos los componentes se inician en la carga de la aplicación, de manera que no hay tiempos de espera entre la navegación de las diferentes pantallas que componen la aplicación permitiendo una navegación fluida.

Los siguientes bocetos por lo tanto siguen la línea estética del diseño final de la app.

Pantalla de inicio – Login.

La pantalla de inicio es la que se presentará cuando el usuario cargue la app.
Consta de dos campos donde introducirá el email y la contraseña asignados.

Si el usuario no recuerda la contraseña asociada a su podrá pulsar el enlace "He olvidado mi contraseña" donde será dirigido a la pantalla para realizar el proceso de recuperación de contraseña.

Una vez introducidos los datos, el usuario deberá pulsar el botón "Acceder", y si los datos son correctos, entrará en la parte privada de la aplicación con la información del evento en el que esté asociado.

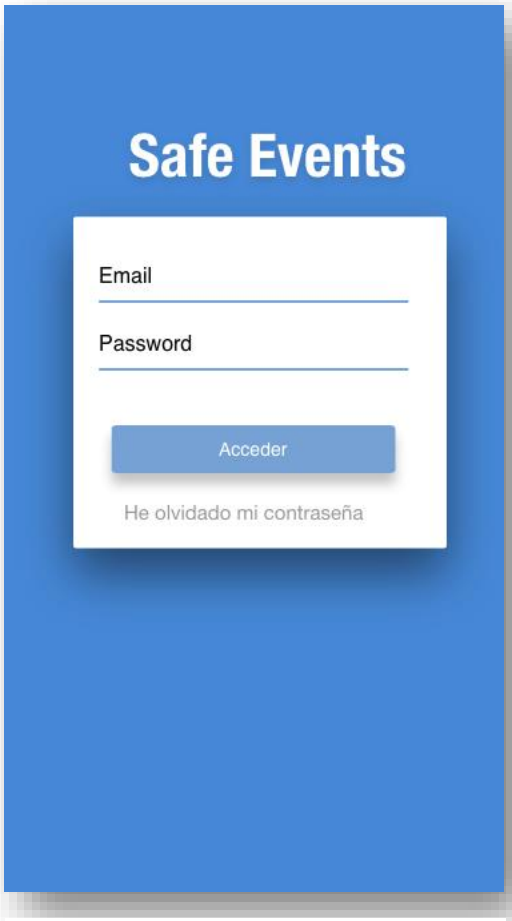

Figura 8: pantalla de inicio

Pantalla principal

La pantalla principal consta de la cifra actual de asistentes del evento asociado al usuario, recargándose conforme los asistentes validen su entrada al evento.

Además en la parte superior izquierda está el botón para acceder a los mensajes que tenga el usuario. En caso de que reciba un mensaje, el icono cambia de color a rojo si los mensajes no han sido leídos.

En la parte superior izquierda se sitúa el botón para acceder a los ajustes de la aplicación o cerrar la sesión.

Figura 9: pantalla principal

Acceso a ajustes

Una vez pulsado el botón de ajustes, se despliega el menú donde se muestran las opciones de acceder a los ajustes de la aplicación o cerrar la sesión del usuario.

Figura 10: pantalla principal - Ajustes

Mensajes

En la pantalla de mensajes se muestra el listado de mensajes que tenga el usuario, con titulo, texto y hora del mensaje.

Si el mensaje no se ha leído estará marcado en rojo para que sea mas visible para el usuario.

En la parte superior izquierda se sitúa el botón para regresar a la pantalla privada desde la que el usuario ha accedido.

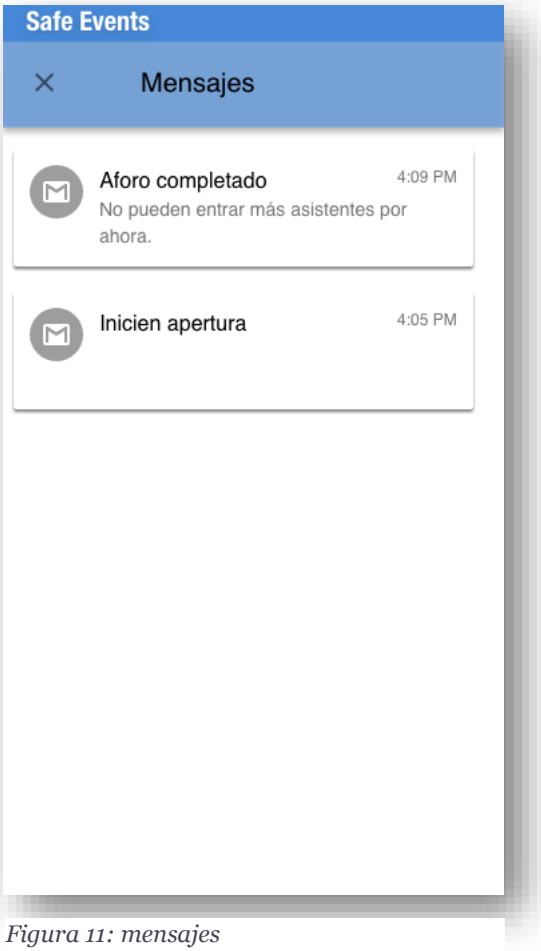

Figura 11: mensajes

Ajustes

En la pantalla de ajustes está el campo de cambiar contraseña, donde el usuario debe introducir la contraseña actual y la nueva contraseña que quiere establecer. Si todos los datos son correctos se muestra un mensaje al usuario notificando de que el cambio se ha producido correctamente. En caso contrario, notifica al usuario para que compruebe los datos introducidos.

Más abajo está el botón bimodal para confirmar que se mantenga la sesión iniciada en el dispositivo o en caso contrario que se cierre la sesión al cerrar la app.

En la parte superior izquierda se sitúa el botón para regresar a la pantalla privada desde la que el usuario ha accedido.

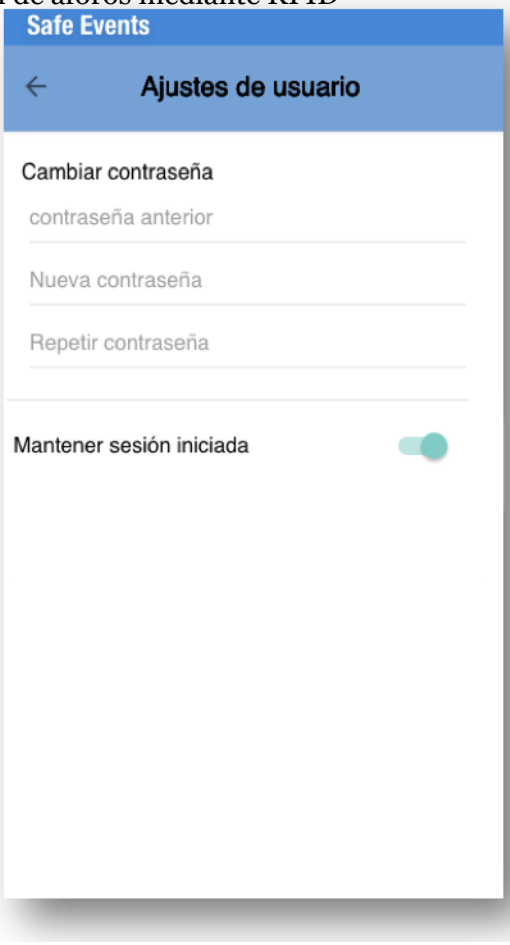

Figura 12: ajustes

Recuperar contraseña – mail

Esta es la pantalla que se muestra tras pulsar en el enlace de “He olvidado mi contraseña” de la pantalla de inicio.
Consta de un campo de texto que el usuario ha de rellenar con el email de su cuenta y un botón que ha de pulsar tras completar el campo.

Si el usuario no ha introducido un correo electrónico válido se le notifica para que compruebe los datos introducidos.

Una vez validados los datos se muestra un mensaje confirmando que se ha enviado un email a la cuenta del usuario para continuar con el proceso de recuperación de contraseña.
En la esquina superior izquierda se sitúa un botón para regresar a la parte publica desde la que ha accedido el usuario.

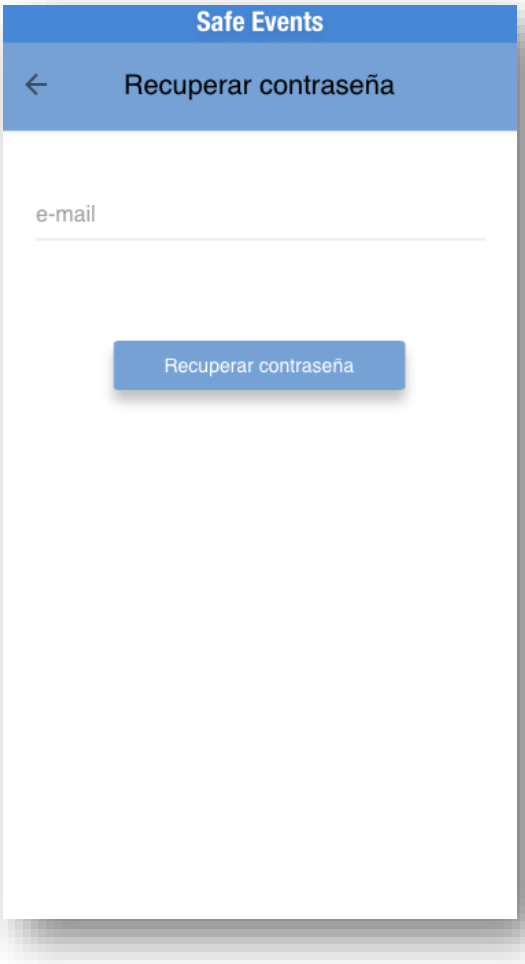

Figura 13: recuperar contraseña - 1

Recuperar contraseña – validada

Si el usuario pulsa sobre el enlace que ha recibido en su cuenta de correo electrónico tras iniciar el proceso de recuperación de contraseña, es redirigido a la siguiente pantalla.

Consta de dos campos donde el usuario ha de introducir la nueva contraseña que quiera fijar.
Debajo de los campos se sitúa el botón para que el usuario una vez haya introducido los datos, confirme el proceso.

En la parte superior izquierda se sitúa el botón para regresar a la pantalla pública, en caso de que el usuario no vaya a cambiar finalmente la contraseña.

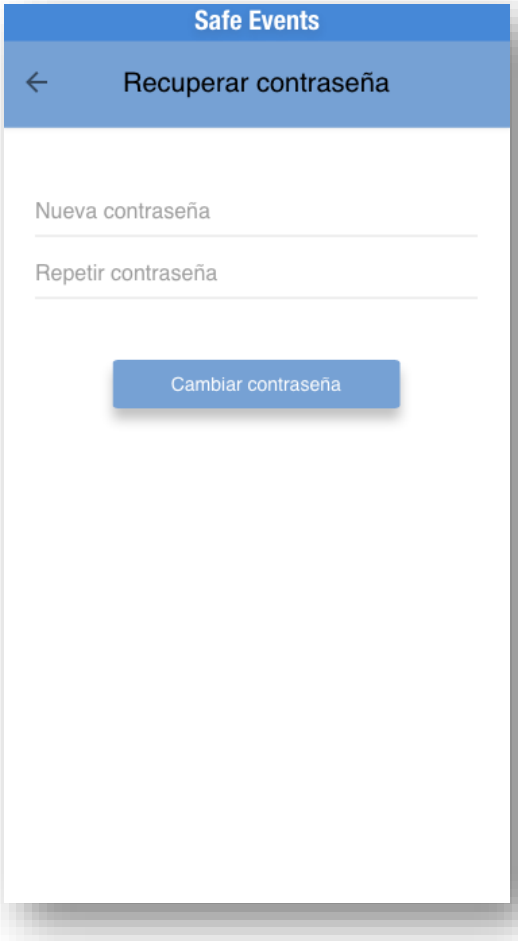

Figura 14: recuperar contraseña - 2

Mapa de navegación

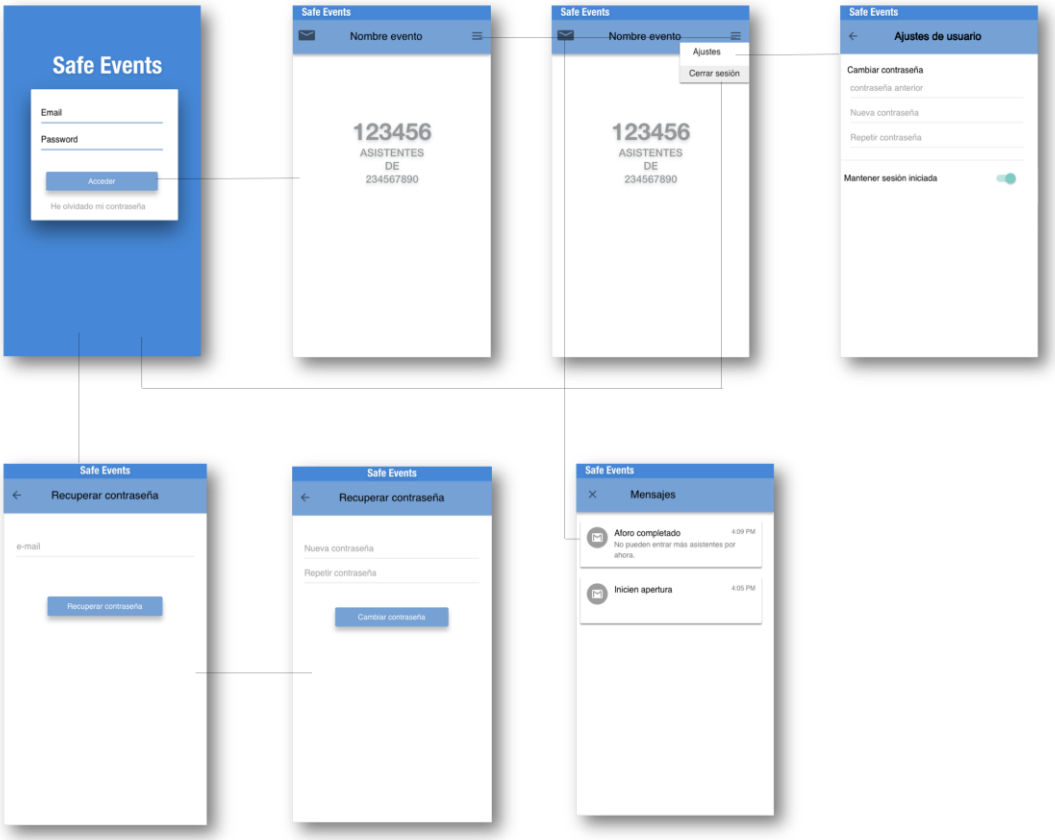

Figura 15: mapa de navegación

Como ya hemos descrito y podemos ver en la figura 15, este es el modo con el que se navegará por cada una de las pantallas de la aplicación web.

5.2.3 Identificación de las entidades de persistencia

Diagrama de persistencia

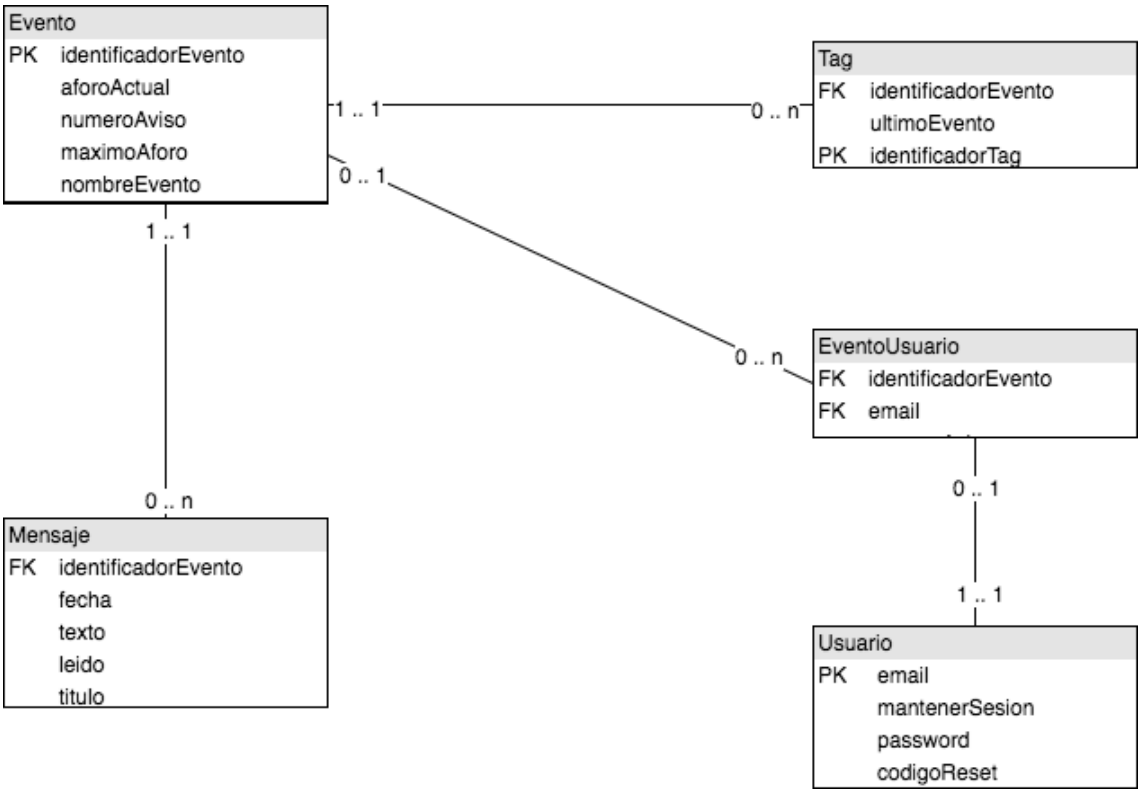

Figura 16: diagrama de persistencia

Para describir el diagrama analizaremos cada una de las tablas que conforman el diagrama.

Evento

Para la representación de un evento en nuestro sistema se requieren los siguientes campos:

1. aforoActual: número de asistentes actuales del evento.
2. numeroAviso: número fijado por el cual si la resta del aforo máximo permitido del evento y la cifra del aforo actual es menor o igual se produce un aviso en la aplicación.
3. identificadorEvento: número que identifica el evento.
4. maximoAforo: número de aforo máximo permitido para el evento.
5. nombreEvento: nombre del evento asociado.

Tag

1. identificadorEvento: identificador del evento.
2. ultimoEvento: campo para asegurar que el identificador NFC no entra repetidas veces consecutivas, o por el contrario sale consecutivamente, situaciones que no son las normales del funcionamiento.
3. identificadorTag: identificador de cada etiqueta NFC.

Mensaje

1. identificadorEvento: número identificativo del evento.
2. fecha: fecha en la que se ha escrito el mensaje.
3. texto: cuerpo del mensaje a mostrar.
4. leído: flag para mostrar si se ha leído o no el mensaje.
5. email: email del usuario que sirve a la vez de identificador.
6. titulo: titulo del mensaje a mostrar.

Usuario

1. email: email del usuario que sirve a la vez de identificador.
2. mantenerSesion: flag para confirmar que el usuario quiere mantener la sesión iniciada en el dispositivo móvil.
3. password: contraseña con la cual entra en la app el usuario.
4. codigoReset: código para verificar que el enlace que se envía al usuario para restablecer la contraseña es el original enviado por la app para evitar accesos fraudulentos.

EventoUsuario

1. identificadorEvento: Identificador del evento.
2. email: Email del usuario que sirve a la vez de identificador.

5.2.4 Implementación de las interfaces de usuario

GWTP sigue un modelo vista controlador [MVC], lo que permite la implementación de la vista separando la lógica de negocio de cada una de las interfaces que conforman la aplicación.

Dentro del proyecto, tenemos dos partes diferenciadas, el cliente y el servidor. El cliente es la parte que se va a desarrollar en este *sprint*.

Dentro del cliente, se debe crear un paquete donde desarrollar las vistas, con sus respectivas interfaces para comunicarse con el *presenter*, que implementa la lógica de la vista.

Esta tarea se centra en el desarrollo de la interfaz únicamente, siguiendo los bocetos aprobados por el cliente. Es decir, localizar y montar cada uno de los componentes necesarios para su funcionamiento.

Por ejemplo, para la pantalla login se debe seguir los siguientes pasos:

- Creación de LoginView.ui.xml, fichero donde se representa cada uno de los componentes de la vista, similar al formato HTML.
- Creación de LoginView.java, fichero encargado de mapear cada uno de los componentes identificados en LoginView.ui.xml. Esta es la clase que interactuará con cada uno de los componentes.
- Creación de LoginUIHandlers.java, interfaz encargada de declarar los métodos con los que la vista se comunica con el presenter.
- Creación de LoginPresenter.java, clase encargada de toda la lógica de negocio de la vista.

Pero para evitar toda esta configuración y posibles errores, Eclipse tiene un *plugin* de GWTP el cual crea todos los ficheros, y solo hay que preocuparse de la implementación.

Lo primero de ello, sería añadir cada uno de los componentes a la vista, es decir el fichero LoginView.ui.xml.

Este es el código necesario para el botón de acceso a la plataforma junto los campos de usuario y contraseña:

```
<m:MaterialTextBox ui:field="tUsername" type="EMAIL" placeholder="Email" marginTop="10"/>

<m:MaterialTextBox ui:field="tPassword" type="PASSWORD" placeholder="Password" marginTop="10" />

<m:MaterialButton ui:field="btnLogin" waves="LIGHT" text="Log In" addStyleNames="{res.style.btnLogin}"/>
```

Cada uno de estos componentes tiene un atributo ui:field el cual es el identificador de cada uno de ellos.

Más tarde, en la clase LoginView.java se deben mapear de la siguiente forma:

```
@UiField
MaterialButton btnLogin;
@UiField
static MaterialTextBox tUsername;
@UiField
static MaterialTextBox tPassword;
```

De esta forma, ya se puede interactuar con cada uno de los componentes mapeados. Por ejemplo, cuando un usuario pulse sobre el botón para acceder la plataforma, espera una respuesta de la plataforma, concediéndole el acceso si todos los datos son correctos, o un error en caso contrario. Para ello, debemos añadirle un handler al btnLogin para que sepa que debe hacer cuando pulsan sobre él. Esto se define de la siguiente forma:

```
@UiHandler("btnLogin")
void handleClick(ClickEvent e) {
        getUiHandlers().onDoLogin(tUsername.getText(), tPassword.getText());

}
```

De esta manera, hemos definido un método onDoLogin, donde se pasan los valores introducidos por el usuario, para que el presenter se encargue de su validación comunicándose con el servidor.

Para ello, hay que definir en la interfaz dicho método, que implementará el *presenter*. Únicamente definimos el método, su implementación pertenece a otra tarea del *sprint*.

```
interface LoginUiHandlers extends UiHandlers {
        void onDoLogin(String username, String password);
```

Con todos estos pasos hemos definido todos los componentes junto con el control de datos para que se comunique con su respectivo presenter.

5.2.5 Implementación de la lógica de negocio

Continuando con el ejemplo anterior de la pantalla de autenticación en la plataforma, se va a mostrar cada uno de los pasos seguidos.

Definición de las clases y la persistencia

Para definir las clases que se van a manejar en la lógica de nuestra aplicación, primeramente definiremos el modelo como hemos presentado en el diseño. Para ello, crearemos las clases en la parte dedicada al servidor. Por ejemplo, para definir la clase de evento se declararía de la siguiente manera:

```
@PersistenceCapable
public class Event implements Serializable{
	@PrimaryKey
	@Persistent(valueStrategy =IdGeneratorStrategy.IDENTITY)
	private Key key;
	@Persistent
	private String name;
	@Persistent
	private int maxAforo;
	@Persistent
	private int aforoActual;
	@Persistent
	private int eventID;
	@Persistent
	private int alertNumber;

	public Event(String name, int maxAforo, int aforoActual, int eventID, int alertNumber)
{
		this.name = name;
		this.maxAforo = maxAforo;
		this.aforoActual = aforoActual;
		this.eventID = eventID;
		this.alertNumber = alertNumber;
	}

	public int getAlertNumber() {
		return alertNumber;
	}

	public void setAlertNumber(int alertNumber) {
		this.alertNumber = alertNumber;
	}

	public Key getKey() {
		return key;
	}

	public String getName() {
		return name;
	}

	public int getMaxAforo() {
		return maxAforo;
	}

	public int getAforoActual() {
		return aforoActual;
	}

	public int getEventID() {
		return eventID;
	}
```

Como podemos observar, están representados todos los atributos de las tablas que hemos diseñado, acompañados de anotaciones donde se define como entidades de persistencia, similar a las anotaciones utilizadas por *Hibernate* para definir el mapping. Además la clase cuenta con el constructor, junto con los *getter* y *setter* que no se muestran por su obviedad.

Una vez definido el modelo, se debe crear una clase tipo *DAO* (*Data Acces Object*) por cada clase definida de nuestro modelo para manejar las operaciones con la base de datos como lectura o escritura de cada entidad.

Y finalmente, para manejar en el cliente cada una de estas entidades, se debe crear una clase de tipo *DTO (Data Transfer Object*). De esta manera podemos manejar los datos extraídos de la base de datos en el cliente.

De esta manera se define la clase DTO:

```
public class EventDTO implements IsSerializable {

	private String eventName;
	private int maxAforo;
	private int aforoActual;
	private int eventID;
	private int alertNumber;

	public EventDTO(){

	}

	public EventDTO(String eventName, int maxAforo, int aforoActual, int eventID, int alertNumber){
		this.eventName = eventName;
		this.maxAforo = maxAforo;
		this.aforoActual = aforoActual;
		this.eventID = eventID;
		this.setAlertNumber(alertNumber);
	}
```

Como se observa, es similar a la clase del modelo, exceptuando las anotaciones, ya que estas clases se manejan en el cliente.

Definición e implementación de las llamadas RPC

Las llamadas *RPC (Remote Procedure Call*, llamadas a procedimiento remoto) son capaces de encapsular las comunicaciones entre cliente y servidor, lo cual facilita el desarrollo de todas las operaciones entre ambos componentes.

Esta es la estrategia seguida para la comunicación entre cliente y servidor. Para ello, definimos una interfaz común para aunar todas las operaciones.

En el ejemplo del acceso a la plataforma, se debe validar los datos introducidos por el cliente, y de esta forma conceder o denegar el acceso.

Estas operaciones se invocan desde el cliente, por lo que se invocará siempre desde el *presenter*, el encargado de manejar la lógica de la parte cliente.

Una vez definida la operación en la interfaz, se invocará desde el presenter, para ser procesada por el servidor, el cual realizará las operaciones necesarias para devolver la respuesta.

Por ejemplo, este es el código necesario a nivel de cliente para invocar la función.

```
rpcManager.doLogin(username, privatePass,getView().keepLoginState() , new AsyncCallback<Boolean>() {

	@Override
	public void onSuccess(Boolean result) {
		if(result){
			successLogin(username,privatePass);
		}else{
			getView().showDataError("Error");
		}

	}

	@Override
	public void onFailure(Throwable caught) {
		if(caught instanceof UserException){
			UserException us = (UserException) caught;
				if(us.getType() == UserException.INVALID_DATA){
			getView().showDataError("Email o password incorrectos");
			}else if(us.getType() == UserException.DISLABED_USER){
				getView().showDataError("Usuario desactivado");
					}
				}

			}
		});
```

Se debe manejar también en caso de error en la comunicación las excepciones que pueden producirse.

Definición e implementación de los Servlet

Para cada una de las operaciones que se van a definir, la validación del enlace enviado por correo electrónico para el cambio de contraseña, y la operación de validación de datos al producirse un evento de entrada y salida en el lector.

Ambos se definen en el lado del servidor, donde se manejaran dichas operaciones. El servlet se encarga de des-serializar cada uno de los parámetros enviados para su posterior manejo.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                String eventType = request.getParameter("eventType");
                String event = request.getParameter("event");
                String tagID = request.getParameter("tagID");

                if(eventType != null && event != null && tagID != null){
                        System.out.println(event + " " + eventType + " "+ tagID);
                }
```

Mediante el código presentado, se extrae cada uno de los parámetros para su posterior tratamiento.

Como podemos observar, extrae los parámetros “eventType”, el cual será de tipo entrada o salida, “event” el cual es el identificador del evento, y “tagID” que corresponde a una pulsera previamente dada de alta en el sistema. Si alguno de estos parámetros no es coherente con el sistema, se devuelve un código de error que recibirá el lector.

Implementación de la mensajería

Para la implementación de la mensajería, en este caso mediante correo electrónico, se define una clase base de la cual heredarán todas las clases, que conformaran un tipo de mensaje diferente en el caso que se desee enviar varios tipos de mensaje, como el dar de alta un usuario, aunque en este caso se da de alta un usuario por parte del administrador del sistema.

Por lo tanto, se define una clase llamada BasseMessage, la cual tiene como atributos “sendTo” dirección destino del correo y “subject” titulo del mensaje.

Una vez definida la clase, creamos otra que herede de ella, en este caso creamos la clase correspondiente a la recuperación de contraseña.

Esta clase se encarga de rellenar el cuerpo del mensaje junto con la dirección y tema del correo, y finalmente se envía al destinatario.

5.2.6 Despliegue en App Engine de la primera versión

Para el despliegue de la aplicación en App Engine, primero se debe crear una cuenta, y registrarse en la plataforma de Google cloud.

Una vez creada la cuenta, se debe crear el proyecto para posteriormente proceder a su subida.

Desde Eclipse, una vez compilado el proyecto sin errores, se procede de la siguiente forma:

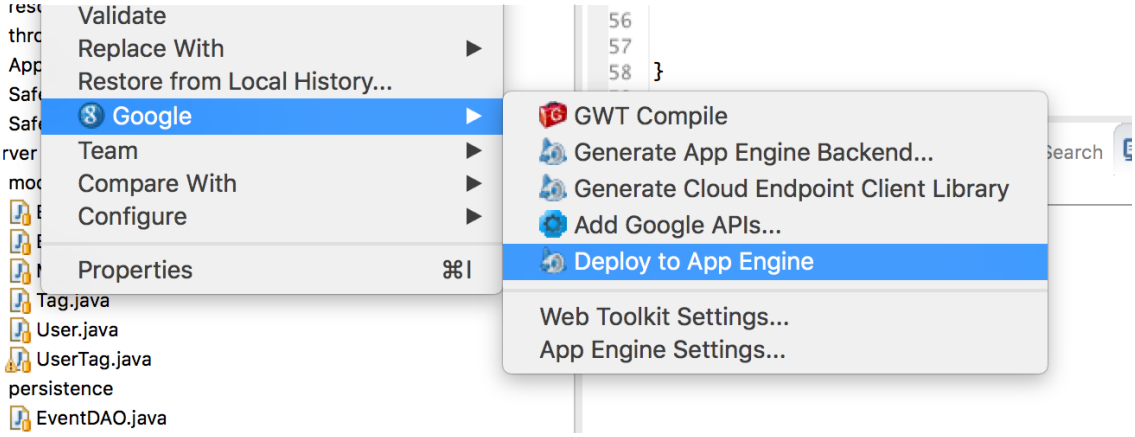

Figura 17: Panel de eclipse

Si eclipse está enlazado correctamente con la cuenta de desarrollador, empezará su compilación y subida.

Si todo ha ido correctamente, sin fallos de compilación o otros, se abrirá una pantalla en el navegador con nuestra aplicación web funcionando por primera vez sobre App Engine.

App Engine proporciona también una interfaz para poder acceder a la base de datos y realizar cambios en ella fácilmente como se muestra a continuación:

Figura 18: Panel Google Cloud

5.2.7 Testing general de la interfaz y los Servlet

Una vez desplegada la aplicación, podemos iniciar la fase de testeo manual. Al haber redactado los casos de uso, podemos utilizarlos para validar su correcto funcionamiento.

Para el testeo del servlet debemos crear una cuenta en la plataforma con un correo electrónico valido para recibir el enlace y asegurarnos que tanto el envío del correo como el procesamiento del enlace se produce de manera satisfactoria.

En cambio, para testear el servlet encargado de la operación de validación de entrada y salida, deberemos crear una petición http la cual lanzaremos desde el navegador con los parámetros previamente introducidos en la base de datos, y de esta manera obtendremos la respuesta en el propio navegador, además de verse en el caso de que se procese correctamente la petición, el cambio en la base de datos.

Cabe destacar que en el proceso de testeo surgieron problemas en el envío de correos electrónicos, debido a una configuración incorrecta de App Engine, por lo que no se recibían los correos.

La tarea se alargó un poco mas de lo esperado tratando de solucionar dicho problema.

5.3 Conclusiones y retrospectiva del *sprint*

El *sprint* se ha desarrollado prácticamente sin contratiempos debido a la experiencia acumulada anteriormente en el desarrollo de la plataforma, salvando el caso del correo electrónico, la cual llevó unas dos horas extra para corregir el error.

Por lo tanto, se han cumplido todos los objetivos marcados para el *sprint* dedicado a la aplicación web.

6. Segundo sprint

6.1 Identificación y estimación de las tareas

Este *sprint* basa sus tareas en el desarrollo de la aplicación móvil que sirve de enlace con la aplicación web, extendiendo las funcionalidades que ofrece, y en los detalles visuales para dotarle de un aspecto atractivo.

Para su desarrollo se ha seleccionado el framework Apache Cordova, el cual permite el desarrollo de aplicaciones móviles multiplataforma con un código común. Si se desea añadir funcionalidades nativas se debería extender el desarrollo en cada plataforma, aunque esto sucede en situaciones excepcionales, ya que Cordova dispone de una gran cantidad de funciones Javascript con las cuales es capaz de comunicarse con el dispositivo para llevar a cabo funcionalidades nativas, como puede ser el acceso a la cámara del dispositivo.

La aplicación que se va a desarrollar es para IOS, aunque con unos pequeños cambios se podría disponer de la de Android. Estos cambios son únicamente los iconos y recursos gráficos de los que se sirve la aplicación para mostrarse en el dispositivo. No implicarían cambios en el código.

El único requisito que tiene el desarrollo de una aplicación móvil para IOS es tener un ordenador Mac, y de las herramientas de desarrollador, conocidas como XCode. No se puede compilar una aplicación fuera de este entorno de trabajo, aunque la versión comercial de Cordova, conocida como Phonegapp, permite la compilación en la nube, por lo que no sería necesario disponer de un Mac para su desarrollo.

Estas son las tareas que forman parte del *sprint*:

6.2 Inicio del *sprint* – Selección de tareas

1. Instalación Apache Cordova en Mac – 4h
2. Instalación Xcode – 3h
3. Boceto pantalla e iconos – 5h
4. Implementación pantalla de carga – 5h
5. Creación de la aplicación – 3h
6. Creación de los elementos gráficos – 8h
7. Creación del splash de carga – 5h
8. Desarrollo de la comunicación entre aplicaciones – 16h
9. Carga de los elementos gráficos – 5h
10. Despliegue en dispositivo físico – 1h
11. Testing manual – 5h

6.2.1 Instalación Apache Cordova en Mac

Como se ha mencionado anteriormente, para la compilación de la aplicación IOS es necesario un entorno de trabajo Mac, junto las herramientas de desarrollador de Mac, conocido como Xcode.

Lo primero de todo, instalar la plataforma Apache Cordova. Para ello, necesitamos tener instalado en el equipo Node.js.

Una vez instalado, desde terminal introducimos los siguientes comandos:

```
$ sudo npm install –g cordova
```

Con ello se procede a la descarga e instalación de la herramienta. Una vez finalizado el proceso, que suele tomar bastante tiempo, procedemos a revisar si la instalación se ha producido correctamente, con el comando $cordova –version.

Este comando, en el caso de que se haya instalado correctamente, devolverá la versión instalada en el equipo de Cordova.

Apache Cordova maneja toda creación de un proyecto por terminal, no dispone de cliente gráfico, por lo que para facilitar el posterior desarrollo y visionado de los resultados, debemos instalar Xcode.

6.2.2 Instalación Xcode

Según la versión de MacOs que esté instalada en el equipo, este proceso puede solicitarse incluso antes de la instalación de Cordova, por lo que esta tarea y la anterior suelen solaparse.

Para proceder a la instalación se debe acceder a la Mac Store, donde el desarrollador previamente se ha dado de alta, y acto seguido se procede a la descarga de Xcode. Es un paquete bastante pesado, por lo que se tomará su tiempo.

Una vez descargado, la instalación requiere permisos de administrador, y también toma bastante tiempo debido a la gran cantidad de herramientas que contiene Xcode, como un emulador de IOS, para corregir posibles errores durante el desarrollo.

Una vez instalado, se puede proceder a crear el proyecto mediante Cordova, que posteriormente importaremos a Xcode para facilitar el desarrollo.

6.2.3 Boceto pantalla e iconos

Para mostrar al cliente que aspecto va a tener el inicio de la aplicación junto los iconos que se van a mostrar se crearán unos bocetos para la posterior validación del cliente.

Los recursos que se deben crear son dos, la pantalla de carga de la aplicación, que es la que se muestra al cargar la aplicación web, mientras esta establece conexión, por lo que

se requiere conexión a internet en todo momento. Y los iconos que se muestran tanto en la pantalla de inicio del móvil como cuando se muestra la pantalla de multitarea.

Por otra parte, también se debe crear un pantalla de carga para la aplicación web, donde se compruebe si el usuario tiene la sesión iniciada en el dispositivo, para mantenerlo conectado, o en caso contrario mostrar la pantalla de autenticación de usuario.

Para ello, se han creado los siguientes bocetos:

Este es el aspecto que va a seguir el icono de la aplicación en la pantalla home del dispositivo.

Figura 19: Logo app

La siguiente imagen muestra el aspecto de la pantalla de carga, la cual muestra inicialmente el logotipo de la aplicación, para posteriormente mostrar un spinner de carga.

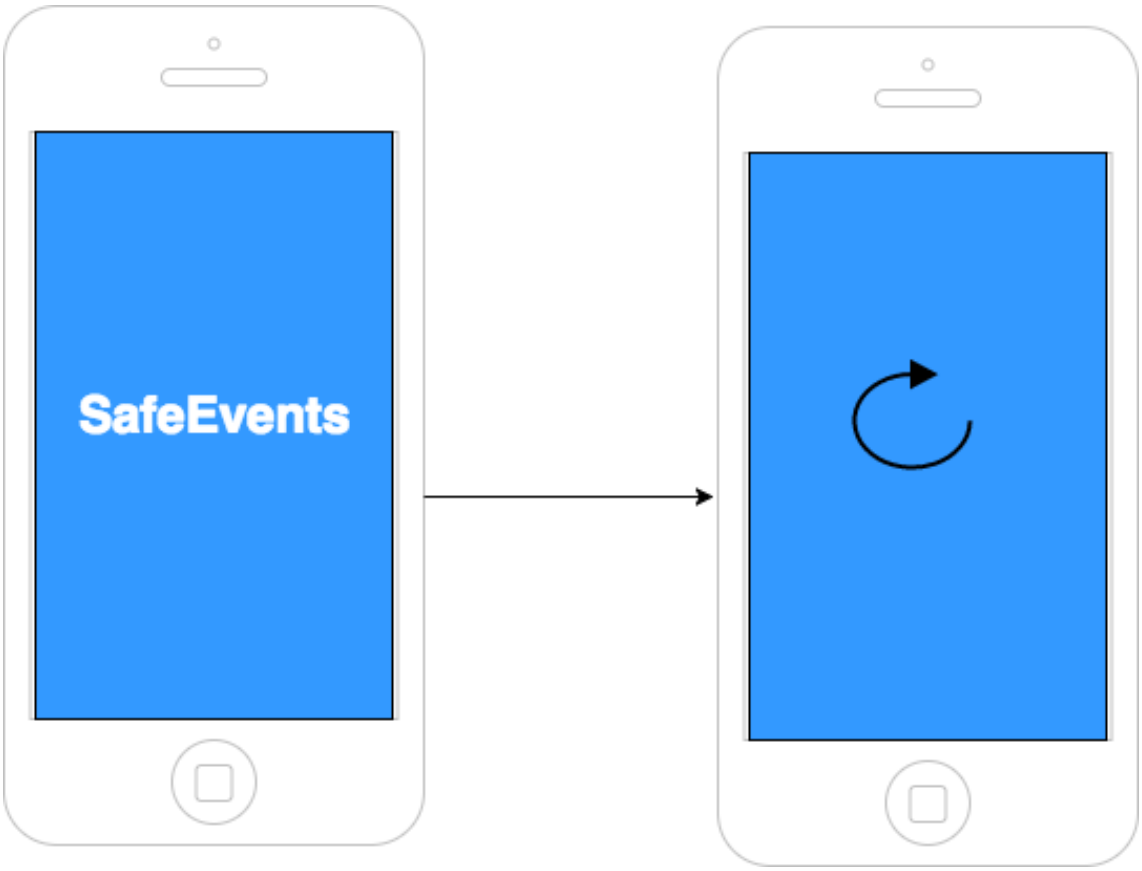

Figura 20: Transición de carga (boceto)

6.2.4 Implementación pantalla de carga

Una vez aprobados los bocetos anteriores, procedemos a su desarrollo.
Cordova se basa básicamente en incrustar un navegador web, al cual se le extienden funcionalidades propias de las aplicaciones nativas como las notificaciones push.
Lo que permite crear recursos como la pantalla carga, a partir de una pantalla HTML que se adapte al tamaño del dispositivo, y dotarlo de funcionalidad mediante Javascript.
Para ello, creamos la pantalla de carga, emplazando el título y ajustando los estilos, junto un temporizador que una vez agotado mostrará la rueda de carga. Si la rueda de carga se mantiene durante mucho tiempo debido a que no es capaz de cargar la aplicación, se debe mostrar un mensaje de error.

Una vez acabada la pantalla, este es el aspecto que tiene en la carga o en caso de error:

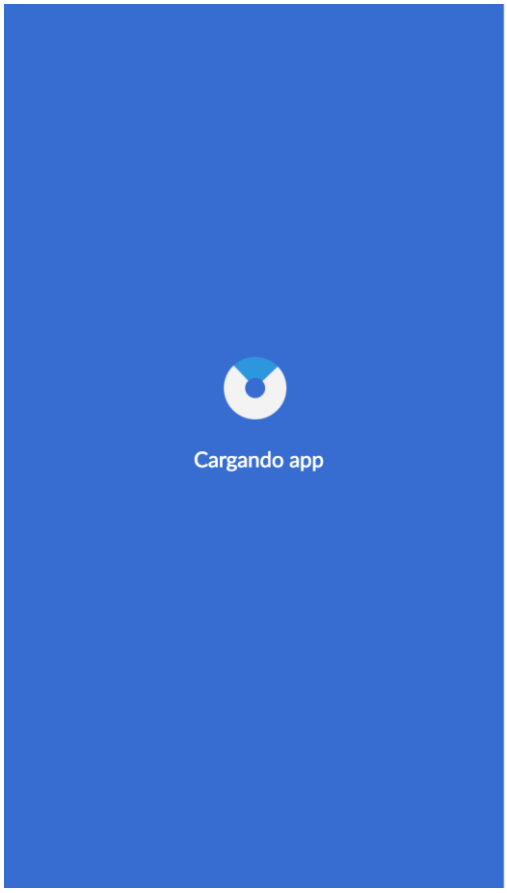

Figura 21: Spinner real

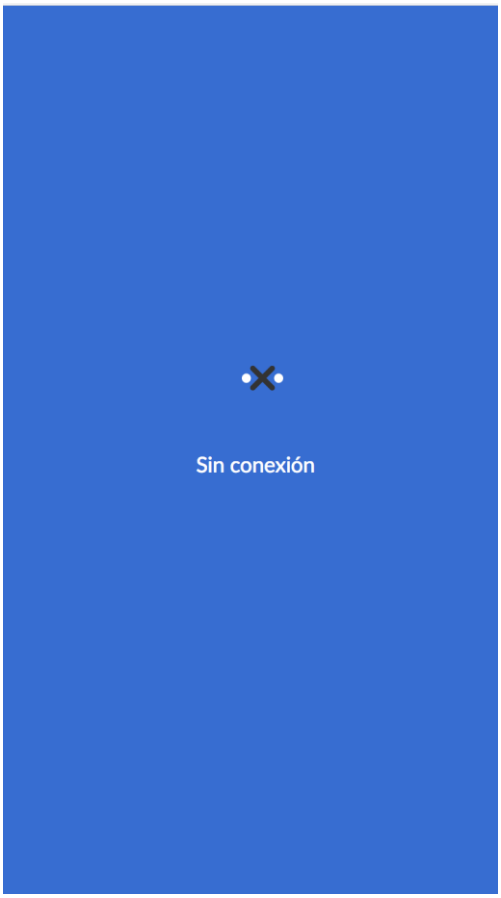

Figura 22: Error de carga

6.2.5 Creación de la aplicación

Para crear un proyecto nuevo en Cordova, abrimos un terminal, y nos situamos con los comandos sobre el directorio donde queremos crear el proyecto.

Una vez situados, introducimos el siguiente comando:

```
$ cordova create safeevents com.tfg.upv SafeEvents
```

Una vez ejecutado, se crea un directorio donde contiene todo el código necesario para su desarrollo. Pero todavía queda añadir la o las plataformas que queremos desarrollar.

Para ello, se introduce el siguiente comando.

```
$ cordova platform add ios
```

Este comando crea la plataforma la cual vamos a modificar en Xcode. Por tanto, una vez creada, procedemos a la importación del proyecto en Xcode.

Hay que tener en cuenta, que el código no se modifica en el proyecto generado de IOS, error común, ya que después de realizar una compilación, los cambios se reemplazan por los ficheros padre del proyecto, que es donde se deben realizar los cambios.

Por tanto, el proceso de desarrollo, sigue estas pautas:

- Modificar los archivos generados en la raíz del proyecto.
- Compilar por terminal el proyecto.
- Revisar los cambios realizados.
- Lanzar la aplicación sobre el emulador o dispositivo.

Estos ficheros son los que se encuentran bajo la carpeta www:

Figura 23: Directorio Cordova

El fichero index.html es el anteriormente creado, al que debemos añadirle los métodos Javascript correspondientes para comunicarse con la aplicación web, como por ejemplo, para asegurarse que la aplicación web se ha cargado correctamente.

La aplicación móvil hace una petición a la aplicación web, y si todo va según lo previsto, esta responde con un mensaje que recoge la aplicación notificando que la carga ha sido un éxito.

Dicho desarrollo corresponde a una tarea posterior.

6.2.6 Creación de los elementos gráficos

Para la creación de los iconos que hemos realizado el boceto, tenemos que tener en cuenta las medidas especificadas por Apple para enlazarlas con la aplicación. Estas imágenes deben ser de una resolución y tamaño concretos según el dispositivo.

Para ello, revisamos la documentación que ofrece Apple [Apple] al desarrollador y las medidas son las siguientes:

Estas son las medidas junto el convenio para nombrar a los ficheros, de lo contrario Xcode produce un error.

Image Size (px)	File Name
57x57	Icon.png
114x114	Icon@2x.png
72x72	Icon-72.png
144x144	Icon-72@2x.png
29x29	Icon-Small.png
58x58	Icon-Small@2x.png
50x50	Icon-Small-50.png
100x100	Icon-Small-50@2x.png

Figura 24: Referencia de Apple

Para crear el icono, y a partir de el los diferentes tamaños, se ha utilizado Adobe Photoshop. Además, IOS requiere de una imagen de carga, por lo que seguimos el diseño creado inicialmente, para que se acomode a la pantalla HTML de carga. Al desconocer la herramienta, ha supuesto que la tarea llevara bastantes horas para poder extraer un resultado óptimo como se muestra a continuación:

Figura 26: icono real app

Figura 25: Splash real

6.2.7 Creación del splash de carga

Para posibilitar la comunicación entre aplicación móvil y aplicación web, hay que implementar en la aplicación web una serie de funciones para pasar los mensajes a la aplicación móvil.

De esta manera se establece la comunicación entre ambas aplicaciones. La llamada que necesita la aplicación es la notificación de que está completamente cargada para poder mostrarse, de esta manera no se producen pantallas blancas, ocultando su naturaleza de aplicación web. De esta manera el usuario apenas percibe que se trata de una aplicación web.

Por tanto, se debe definir una pantalla de carga, la cual carece de elementos gráficos, pero se encarga de mandar el mensaje una vez se ha cargado.

Una vez finalizado, se debe desplegar nueva versión en App Engine.

6.2.8 Desarrollo de la comunicación entre aplicaciones

Una vez implementada y desplegada la tarea anterior, comprobamos los trazas impresas por consola de la aplicación web para asegurarnos que se ejecutan todos los mensajes añadidos.

A continuación, se deben crear una serie de funciones que identifiquen cada uno de los mensajes, para el correcto funcionamiento. De esta manera, cuando no exista conexión con la aplicación web o cualquier tipo de error, la aplicación móvil es capaz de manejar la excepción.

Para ello, se implementan en el fichero *index.html*, junto el fichero *native-support.js* donde definiremos todas las funciones necesarias.

Por ejemplo, esta es la función para mostrar el mensaje de error en caso de que no se pueda establecer la conexión y salte el tiempo de espera:

```
function onLoadWebAppTimeout()
        {
           if (gwt_htmlLoaded===false)
           {
                    document.getElementById('connect-panel').className='wisafe-connect-animation
wisafe-no-connect';
                     document.getElementById('connect-message').innerHTML='Sin conexi&oacute;n';
              setTimeout(loadWebApp, 1000);
              setTimeout(onLoadWebAppTimeout, 23000);
              return;
           }

        }
```

6.2.9 Carga de los elementos gráficos

Para mostrar los iconos y la pantalla de carga que hemos creado previamente, debemos crear dentro del proyecto una carpeta destinada a almacenar los recursos de manera ordenada.

Una vez se ha creado, añadimos todos los recursos creados, y anotamos la ruta dentro del proyecto para posteriormente añadirlo al fichero donde se mapean cada uno de estos archivos.

A continuación, en Xcode, abrimos el fichero config.xml, donde se añaden todos los recursos, como la siguiente línea:

```
<icon src="res/ios/icon-60@3x.png" width="180" height="180" />
```

Una vez finalizado el archivo, se debe compilar desde terminal, y comprobamos que los ficheros propios de IOS se han actualizado con los cambios realizados.

6.2.10 Despliegue en dispositivo físico

Para testear el conjunto de aplicación móvil junto aplicación web, debemos compilar previamente la aplicación por terminal de Cordova.

Una vez compilada, conectamos un IPhone o IPod touch (que es el tamaño de pantalla para el cual se ha diseñado la app).

En Xcode, se debe pulsar en la esquina superior izquierda sobre el listado de dispositivos y emuladores disponibles, y aparecerá el dispositivo conectado al ordenador. Se selecciona y pulsa sobre el botón "play" situado a la izquierda. Este proceso despliega la aplicación en el terminal físico.

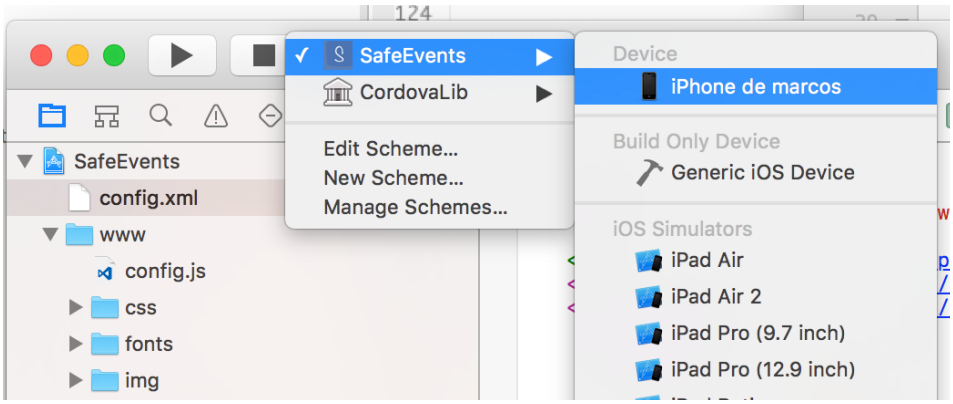

Figura 27: Panel de Xcode

El aspecto de la aplicación en el dispositivo es el mostrado en la figura. Ya es completamente funcional. Si se pulsa sobre ella, la aplicación es lanzada y si se dispone de conexión a internet se muestra la pantalla de acceso a usuarios.

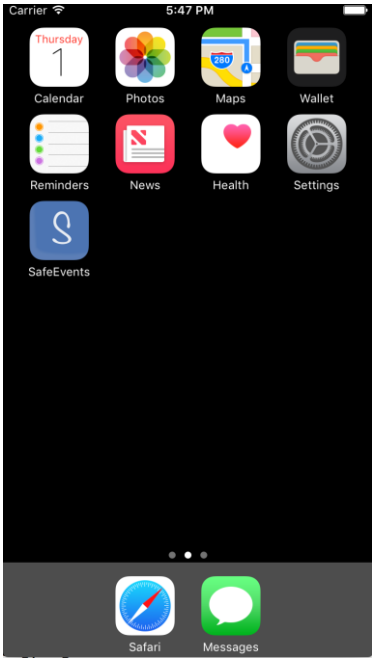

Figura 28: App en el dispositivo

6.2.11 Testing manual

Para el testeo de la aplicación, se vuelven a pasar los test pasados al acabar el desarrollo de la aplicación web, pues es el servicio que debe ofrecer.

Afortunadamente, el único error encontrado ha sido en la pantalla de carga, no se mostraba correctamente el mensaje que notifica que no hay conexión, por lo que el único cambio que se ha realizado posterior al testeo ha sido de estilos en el fichero de carga *index.html*.

6.3 Conclusiones y retrospectiva del sprint

El *sprint* ha transcurrido sin contratiempos. Solo se ha debido rectificar aspectos visuales, y la tarea que mas dificultad ha llevado quizás haya sido el diseño de las pantallas de carga, tanto la versión HTML como las imágenes, para asemejarse lo máximo posible a lo esbozado inicialmente, y que siguiera la misma imagen que la aplicación web.

7. Tercer sprint

7.1 Identificación y estimación de las tareas

Este *sprint* se centra en el desarrollo del lector para dar soporte a la entrada y salida de los asistentes al evento o espectáculo particular. Consta de menos horas, un total de veintiocho horas debido a que implica un menor desarrollo.

Para su desarrollo, se dispone de una Raspberry Pi 3, que es un ordenador del tamaño de una tarjeta de crédito, el cual ejecuta como sistema operativo distribuciones de Linux fácilmente configurable dependiendo del uso que vaya a tener.

Además, se le debe conectar la tarjeta lectora mediante el puerto GPIO que incorpora la Raspberry, la cual se encarga de la lectura de las pulseras, y enviar la información a la aplicación web que se encarga del manejo de los datos.

Se debe configurar de manera que el operario únicamente conecte la Raspberry, y acto seguido tras el arranque del sistema operativo, comience la lectura sin tener que ejecutar ningún comando, de modo que la única preocupación del operario sea disponer de corriente eléctrica para alimentar el dispositivo.

Por tanto, las tareas que conforman el *sprint* son las siguientes:

7.2 Inicio del sprint – Selección de tareas

1. Instalación del sistema operativo (Raspbian) – 5h
2. Configuración Raspberry – 3h
3. Desarrollo del controlador del lector – 5 h
4. Pruebas de lectura – 3h
5. Configuración de arranque – 3h
6. Pruebas programa individual – 3h
7. Pruebas del sistema completo – 5h
8. Entrega del sistema completo – 1h

7.2.1 Instalación del sistema operativo – Raspbian

Para la instalación de Raspbian, primero debemos descargar la distribución desde la pagina web de Raspberry Pi. Se ha escogido la distribución que incorpora el entorno gráfico por si se requiere hacer comprobaciones desde el mismo terminal, aunque no es estrictamente necesario, ya que se puede mostrar mediante el servidor VNC desde otro equipo, únicamente se requiere estar dentro de la misma red.

Una vez descargado, se debe descargar el programa para Mac "Etcher", que es nuestro entorno de desarrollo. Dicho programa graba la imagen del sistema operativo y la prepara para únicamente insertar la tarjeta en la Raspberry y arrancarla.

Una vez finalizado el proceso, se debe conectar la Raspberry a un monitor, teclado y ratón o bien conectarla mediante un cable Ethernet a la misma red de nuestro equipo, de modo que se pueda acceder mediante SSH conociendo su dirección IP. Esta última es la alternativa seleccionada, ya que es muy sencilla de realizar.

7.2.2 Configuración Raspberry

Tras acceder a la Raspberry mediante SSH, configuramos las credenciales de la red Wifi para poder conectar la Raspberry y asegurarnos que los puertos GPIO están habilitados de modo que podamos conectar la tarjeta lectora sin problemas.

Tras las comprobaciones, podemos proceder a la instalación de la tarjeta sin problemas. Necesita que se reinicie el dispositivo para aplicar los cambios realizados, en el caso de que el puerto GPIO estuviera deshabilitado. Además, se debe modificar la dirección IP de la Raspberry y configurarla como fija, de modo que esta no varíe, para poder acceder a ella en todo momento sin que cambie su IP.

Finalizados estos pasos, la Raspberry queda totalmente configurada para nuestro propósito.

7.2.3 Desarrollo del controlador del lector

El lector escogido es una tarjeta PNEV512R, la cual conectamos previamente con el puerto GPIO habilitado. La tarjeta ofrece una API mediante la cual controlar las operaciones de lectura en Python. Es un lenguaje sencillo mediante el cual podemos ejecutar con una pocas líneas de código el propósito del lector.

Por tanto, los requisitos a tener en cuenta, son que el lector debe estar en todo momento leyendo, y si se produce una lectura, enviar los datos a la aplicación móvil, con los datos de la pulsera, los datos del evento y si es un evento de entrada o salida. Este último parámetro lo marca cada Raspberry. Por ejemplo, para poder gestionar debidamente un evento, son necesarios como mínimo 2 lectores, uno de ellos encargado de los eventos de entrada situado en la entrada del recinto, y otro de ellos situado en la salida para procesar dichos eventos.

El siguiente código muestra un sencillo ejemplo de cómo se maneja la lectura:

```
mifare = nxppy.Mifare()
url = "http://192.168.1.38:8888/safeevents-rfid?"
event = "event="
eventType = "&eventType="
tagID= "&tagID="

while True:
    try:
        uid = mifare.select()

except nxppy.SelectError:
        pass
```

Después de identificar un pulsera, se construye una petición dirigida a la aplicación web. De este modo quedan registrados los eventos de entrada y salida.

7.2.4 Pruebas de lectura

Mediante el código anterior, y seleccionado el identificador de cada pulsera, se debe mostrar mediante la función print() para mostrarse por consola.

Las pruebas se pasan sin ningún problema, las pulseras son leídas correctamente.

7.2.5 Configuración de arranque

Una vez creado el programa que controla la tarjeta, debemos otorgarle los permisos para que se pueda ejecutar. Esto se realiza de la siguiente manera desde terminal:

```
sudo chmod 755 archivo
```

El siguiente paso es modificar el archivo situado en /etc/rc.local y añadiendo los comandos para ejecutar el programa, de forma que cada vez que se inicie el sistema operativo ejecutará el programa.

7.2.6 Pruebas del programa individual

Una vez finalizado el desarrollo y configuración para que arranque el programa al iniciar el sistema operativo, ejecutamos el primer fichero creado que solo imprime los identificadores de la pulsera por pantalla.

Para pasar las pruebas, primero se debe reiniciar el sistema y posteriormente comprobar que sin ejecutar ningún comando en el dispositivo, empieza la lectura y se imprimen los identificadores por pantalla.

Pasadas las pruebas, podemos continuar con las pruebas del sistema completo.

7.2.7 Pruebas del sistema completo

Las pruebas del sistema completo, consisten en agregar el fichero que sí que realiza las peticiones a la aplicación móvil y comprobar que tras el inicio del sistema operativo estas peticiones se realizan por cada lectura.

Las pruebas son pasadas correctamente, y se muestra en caso de error por consola un mensaje para evitar la entrada del asistente, además de reflejarse en la aplicación la suma o resta de asistentes presentes en el evento.

Con estas pruebas concluye el desarrollo del sistema

7.2.8 Entrega al cliente

Al tratarse de la misma figura cliente y desarrollador, no requiere de periodo de formación ni adaptación. Solo hay que tener en cuenta la configuración de cada Raspberry, y la administración de los eventos y usuarios que pueden acceder a la aplicación para ver el aforo en tiempo real.

El funcionamiento del sistema es el diseñado inicialmente, cumpliendo los plazos establecidos, y los objetivos principales, como es el bajo coste de los componentes que conforman el sistema.

7.3 Conclusiones y retrospectiva

El desarrollo del sistema concluye satisfactoriamente sin ningún contratiempo ya que el desarrollo era menor comparado con los dos anteriores *sprint*, además de encontrar las facilidades que ofrece el lector para desarrollar un controlador con las funciones que deseemos.

Por tanto, se concluye el proyecto en este *sprint* y se le hace entrega al cliente del proyecto, tal y como se había pactado al inicio del proyecto.

8. Conclusión y trabajos futuros

La conclusión respecto a la tecnología empleada es muy buena. GWTP permite un desarrollo rápido y sencillo de las interfaces de usuario lo que ayuda a centrar el desarrollo en la lógica del proyecto.

En conjunto con Apache Cordova, permite realizar cambios en el cliente web sin necesidad de realizar actualizaciones nativas de la app, lo que ahorra trámites al usuario de la app, y posibles fallos de ejecución por no tener la app actualizada como puede ocurrir en desarrollos puramente nativos.

Se ha cumplido los principales objetivos marcados en el TFG en los plazos concretados.

Respecto a la finalidad que desea satisfacer, pienso que es una herramienta necesaria y de obligatorio uso no solo en la Comunidad Valenciana, si no en todo el territorio español, para tratar de evitar situaciones vividas como la trágica noche de Halloween, donde tuvo un fatal desenlace, debido al sobre aforo del recinto.

Casualmente he tenido la oportunidad de trabajar en Casfid, empresa tecnológica dedicada a dar soluciones al control de aforo en eventos como congresos (IDCongress: rama dedicada al sector de congresos) y festivales (IDASFEST: rama dedicada al sector festivales y conciertos). Esta última rama de la empresa, ofrece soluciones con la tecnología RFID-NFC la cual permite el pago ‘Cashless’, es decir, el pago mediante pulsera previamente recargada.

Esta solución permite al usuario no llevar dinero en efectivo al evento, y controlar mejor sus gastos, además de tener el promotor un registro en tiempo real de que productos se están vendiendo en su evento. El último evento realizado por la empresa superó la cifra de ciento sesenta y cinco mil asistentes, utilizando el sistema ‘Cashless’ para efectuar pagos y tener un registro del aforo en tiempo real, del cual eran conocedores en todo momento las autoridades como policía local y guardia civil.

Además, se tiene en todo momento registro de los datos del asistente, como puede ser teléfono de contacto, email, DNI… etc.

La solución que ofrece IDASFEST es un panel web completamente adaptable a la tecnología empleada para identificar al asistente, como códigos de barras EAN-13 o chips NFC. Tiene este aspecto:

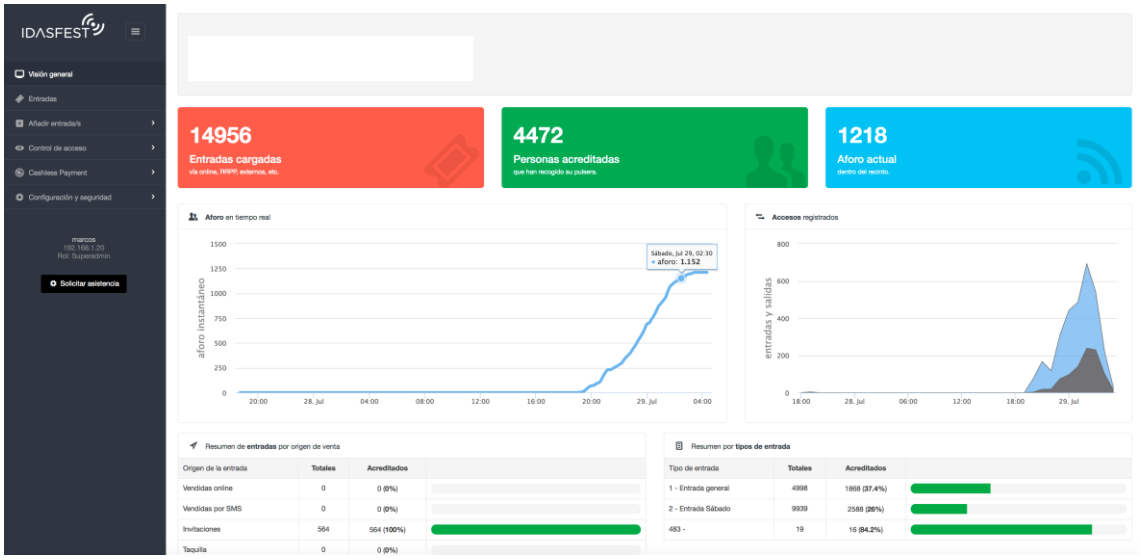

Figura 29: Panel web idasfest

En definitiva, es una tecnología muy útil y flexible para dar solución al control de aforo y una infinidad mas de funcionalidades que se le pueden ofrecer al asistente para su comodidad y ante todo seguridad en un evento.

Como posibles funcionalidades adicionales que pueden admitir el sistema desarrollado en el TFG y mejorarlo, es integrar el mencionado pago por pulsera o control de acceso a zonas especificas dentro de un recinto, otorgando permisos a cada pulsera, como por ejemplo, accesos a zonas VIP.

Este TFG me ha permitido conocer más sobre las tecnologías de radiofrecuencia y alternativas de desarrollo ágil de código, como es la herramienta Cordova, además de utilizar medios de persistencia tan útiles como Google Data Storage.

Bibliografía

[Apple] Guía de diseño de Apple para interfaces gráficas:

https://developer.apple.com/ios/human-interface-guidelines/graphics/app-icon/

[CORDOVA] Guía de Apache Cordova:

https://cordova.apache.org/docs/es/latest/guide/overview/index.html

[GWTP] Guía de GWTP:

https://dev.arcbees.com/gwtp/

[MVC] Modelo Vista Controlador

https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[NFC] Soporte del lector NFC:

http://www.nxp.com/products/identification-and-security/nfc-and-reader-ics/nfc-frontend-solutions/explore-nfc-exclusive-from-element14:PNEV512R?tab=Design_Support_Tab

[SCRUM] SCRUM:

https://proyectosagiles.org/

[Raspberry] Guías Raspberry:

https://www.raspberrypi.org/help/

[RD-143/2015] Real decreto de la Comunitat Valenciana, de Espectáculos Públicos, Actividades Recreativas y Establecimientos Públicos:

http://addient.com/ocavalencia/pdf/normativas/6_Decreto_143_2015.pdf

[UPV] Material docente de la UPV, asignatura PIN:

http://www.upv.es/

Anexo I

Anexo I

Artículo 185. Control de los aforos

1. En los espectáculos públicos extraordinarios, singulares o excepcionales y en los celebrados en establecimientos con licencia distinta a la regulada por la normativa de espectáculos, cuando el aforo exceda de 2.000 personas, los organizadores deberán efectuar un control de acceso del público a través de sistemas técnicos de cuenteo automático de manera que se compute con exactitud el número de personas que acceden y que se hallan, asimismo, en el interior del local, sin perjuicio del número de puertas o accesos al establecimiento o lugar del evento.

2. La medición del control de acceso por cuenteo automático será obligatoria, asimismo, en salas de fiestas, discotecas, salas de baile y pubs cuando su aforo autorizado sea superior a 2.000 personas.

3. Cuando los establecimientos referidos en los apartados anteriores se estructuren en espacios o recintos separados o compartimentados independientes unos de otros, y su aforo particular e individualizado exceda de 2.000 personas, cada espacio o recinto deberá contar con su respectivo sistema de cuenteo automático de aforo.

4. Los sistemas de cuenteo automático para la determinación del aforo que se establezcan, deberán cumplir con lo establecido en la normativa vigente sobre sistemas de cuenteo y control de afluencia de personas en locales de pública concurrencia.

Anexo II

Anexo II

Manual de uso.

El usuario únicamente tiene que ponerse en contacto con el administrador del sistema para contratar el servicio de control de aforo, por el cual se le otorga un usuario y contraseña de la aplicación móvil para tener acceso de modo que pueda controlar en todo momento el aforo actual del evento.

Para ello, tiene que disponer de un móvil con sistema operativo IOS o Android y descargarse la aplicación desde el mercado de aplicaciones de su dispositivo.

Acceso a la aplicación

Para poder acceder a la vista donde se muestra el aforo, el usuario debe introducir el usuario y contraseña facilitados por el administrador.

Cambio de contraseña

Para cambiar la contraseña, el usuario tiene que acceder a su cuenta y desde ajustes de la aplicación pulsar sobre cambiar contraseña, donde introducirá la nueva contraseña que desea.

Recuperar contraseña

Para recuperar la contraseña que por algún motivo no se recuerda, el usuario debe acceder desde la pantalla de inicio a la sección de recuperar contraseña, donde debe introducir el usuario que le ha facilitado el administrador y acto seguido si el usuario es correcto, se envía a su correo electrónico un mail con un enlace donde puede cambiar la contraseña.

Mantener sesión iniciada

Si el usuario no quiere tener que introducir los datos de acceso cada vez que inicia la aplicación, debe acceder a su sesión y en el apartado de ajustes marcar la opción de mantener la sesión iniciada.

Mensajes

Para leer los mensajes, que son enviados por el administrador del sistema, el usuario debe acceder a su sesión. En la pantalla donde se muestra el aforo, en la parte superior izquierda está situado el botón que tras presionarlo muestra todos los mensajes enviados.