Trabajo de Fin de Grado

Grado en Ingeniería Informática

MPIC – Compositor de Imágenes con Teléfonos Móviles

Mobile Phone Image Composer

Rubén Labrador Páez

La Laguna, 1 de julio de 2019

D. Alejandro Pérez Nava, con N.I.F. 43.821.179-S profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. Fernando Pérez Nava, con N.I.F. 42.091.420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

CERTIFICAN

Que la presente memoria titulada:

"MPIC – Compositor de Imágenes con teléfonos móviles"

ha sido realizada bajo su dirección por **D. Rubén Labrador Páez**, con N.I.F.: 78614807W.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de julio de 2019.

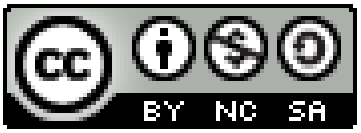
Agradecimientos

A mis tutores, por la idea, por la colaboración, por la libertad para realizar este trabajo.

A mi familia: Carmen, Pedro y Marta, por su apoyo, por el tiempo robado, por no dejarme abandonar, por hacer posible mi éxito en esta aventura.

A mis padres, por su ánimo, por ser soporte moral y material cuando esto se hacía inviable.

Licencia



Resumen

El objetivo de este trabajo ha sido el desarrollo de un sistema que permita la composición de imágenes o figuras, empleando para ello las pantallas de los dispositivos móviles de los asistentes a eventos públicos.

Para cumplir dicho objetivo se ha tenido que realizar el desarrollo de todos sus elementos por completo. Los elementos a desarrollar ha sido los siguiente: aplicación móvil, backend y servidor.

La aplicación móvil ha sido desarrollada para la plataforma Android con el IDE Android Estudio. Es una aplicación móvil sencilla que permite la lectura de un código QR. Este código tiene la información de la zona en la que se encuentra el dispositivo, una vez leído el código, esta información es enviada a un servidor donde el dispositivo móvil se queda conectado a la espera de recibir eventos que le indican que mostrar por pantalla.

La plataforma de backend es una página web que permite la gestión de todo el sistema, en ella, los usuarios administradores disponen de diferentes apartados. El primero de los apartados permite el diseño de lugares o recintos donde se van a celebrar eventos, en el se pueden definir las características del mismo: dirección, zonas, aforo y asientos. El segundo apartado permite la modificación de los recintos creados, permitiendo el cambio de cualquiera de los campos definidos en la creación así como la modificación de zonas. asientos y aforo. El tercero de los apartados permite la creación de eventos, en este apartado, una vez elegido el recito se definen los datos del evento, hora de inicio, duración así como la composición a reproducir, definiendo mediante intervalos que mostrar en las pantallas de cada zona o incluso de cada dispositivo conectado.

Por último el servidor es el encargado de presentar la plataforma de backend, haciendo de interfaz con la base de datos, así como de gestionar la conexión de los dispositivos móviles, siendo el encargado de enviar los eventos a cada dispositivo.

Palabras clave: Android, Java, Javascript, Node, MongoDB, Express, AngularJS.

Abstract

The objective of this work has been the development of a system that allows the composition of images or figures using the screens of mobile devices of people on public events.

In order to complete this objective, it has been necessary to carry out the development of all its elements completely. The elements to be developed have been the following: mobile application, backend and server.

The mobile application has been developed for the Android platform with the IDE Android Studio. It is a simple mobile application that allows reading a QR code. This code has the information of the zone in which the device is located, once the code is read, this information is sent to a server where the mobile device stays connected waiting to receive events that indicates to the mobile whath to show on screen.

The backend platform is a web page that allows the management of the system, in this platform the administrators have different sections. The first section allows the design of places where events will be celebrated, in which the characteristics of the event can be defined: address, zones, capacity and seats. The second section allows the modification of the created sites, allowing the change of any of the fields defined in the creation as well as the modification of zones. seats and capacity. The third of the sections allows the creation of events, in this section, once the site is chosen, the event data, start time, duration as well as the composition to be reproduced, defining by intervals that will be shown on the screens of every device connected by zone.

Finally, the server is responsible of serve the backend platform, interfacing with the database, also managing the connection of mobile devices, being responsible for sending events to each device.

Keywords: Android, Java, Javascript, Node, MongoDB, Express, AngularJS.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Introducción

Mobile Phone Image Composer, MPIC por sus siglas en inglés, es una aplicación que se ha desarrollado como Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática por Rubén Labrador Páez.

La idea detrás de este trabajo es hacer uso de los dispositivos móviles del público asistente a eventos para realizar composición de imágenes, usando las pantallas de los dispositivos móviles como píxeles de un gran mosaico.

Esta idea surge de la unión de otras dos ideas que se emplean habitualmente o que se han venido implementando desde hace mucho tiempo:

- Uso de las linternas o pantallas de los móviles en los conciertos por parte del público, en sustitución de los clásicos mecheros encendidos, para acompañar la música.

- La composición de mosaicos que desde hace mucho tiempo se realiza en eventos deportivos mediante el uso de láminas de diferentes colores.

A modo de breve descripción el uso de la aplicación sería el siguiente:

1. El usuario instala la aplicación.

2. Una vez instalada, el asistente a un concierto o evento deportivo escanea con la aplicación móvil un código QR, mediante la lectura de este código se identifica la posición por zonas o incluso por asiento, si es un evento con entradas numeradas, del usuario.

3. Tras leer el código QR, una vez identificada la posición del usuario con la

información contenida en el mismo, el dispositivo móvil se conecta a un servidor y permanece a la espera de recibir órdenes para mostrar un color determinado en la pantalla, o cambiar a cualquiera de los modos de funcionamiento predeterminados en la aplicación.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de un sistema que permita poner en marcha la idea inicial del trabajo, es decir: desarrollo de una aplicación móvil así como de un sistema que permita diseñar y reproducir composiciones.

Dentro del proyecto se puede definir una serie de objetivos individuales que pasamos a describir a continuación:

- Aplicación móvil: Desarrollo de una aplicación móvil sencilla e intuitiva que permita a los usuarios escanear un código QR con el que establecer su posición, esta aplicación debe conectarse a un servidor y permanecer en escucha para recibir las órdenes de cambio de estado.

- Servidor: Desarrollo de un servidor desde el cual se pueda gestionar la conexión de los dispositivos móviles, así como enviar los cambios de estado.

- Back-End: Desarrollo de una aplicación Back-End para poder gestionar los datos que permiten el funcionamiento de la aplicación: crear recintos, eventos y composiciones, así como la reproducción de las composiciones creadas.

1.3 Estado del arte

Tras realizar diferentes búsquedas por diferentes conceptos relacionados con este Trabajo de Fin de Grado, no se ha encontrado otros sistemas similares que realicen lo que se ha propuesto, por lo que podemos deducir, que no existe un sistema que haga exactamente lo mismo que el que se ha diseñado.

Debido a que no existe un sistema similar al que se ha desarrollado ha habido que realizar un diseño total del mismo, buscando soluciones concretas para cada situación.

Como referencia se ha buscado información sobre el funcionamiento de otras aplicaciones que muestran información en tiempo real, esto ha servido para buscar la manera más eficiente de mantener conectada la aplicación al servidor.

1.4 Planificación

En este apartado se definen las actividades realizadas para llevar a cabo el desarrollo del proyecto propuesto en este Trabajo de Fin de Grado, se detalla además los plazos en los que se ha realizado cada actividad.

1.4.1 Actividades realizadas

Se enumeran a continuación las actividades realizadas para el desarrollo de la aplicación:

1. **Análisis del proyecto y definición de requisitos.** Estudio preliminar de la aplicación, definición de requisitos, diseño del esquema de comunicaciones.

2. **Búsqueda de documentación y soluciones a los problemas más relevantes.** Se busca información y recursos para solucionar los problemas más relevantes encontrada: comunicación entre la aplicación y el servidor de control de los dispositivos, eficiencia en las comunicaciones y lectura de códigos QR.

3. **Diseño y desarrollo de la aplicación móvil.** Desarrollo de la aplicación para dispositivos móviles que instalarán los usuarios.

4. **Diseño y desarrollo de servidor de control de dispositivos móviles.**

5. **Diseño de la base de datos.**

6. **Diseños de la aplicación y servidor de Back-End.**

7. **Integración Back-End, aplicación móvil y servidor.** Fase final en la que se han integrado los tres sistemas desarrollados.

8. **Documentación.** Redacción de la memoria, proyecto de Trabajo de Fin de Grado, reuniones, vídeo presentación del trabajo, preparación de la defensa.

1.4.2 Distribución temporal

En la Tabla 1.1 se puede ver la distribución del tiempo dedicado a cada una de las tareas indicadas en el apartado anterior.

Esta distribución se hace tomando como base las 300 horas de carga de trabajo que se le asignan a la asignatura, cada columna de la tabla representa 5 horas de trabajo.

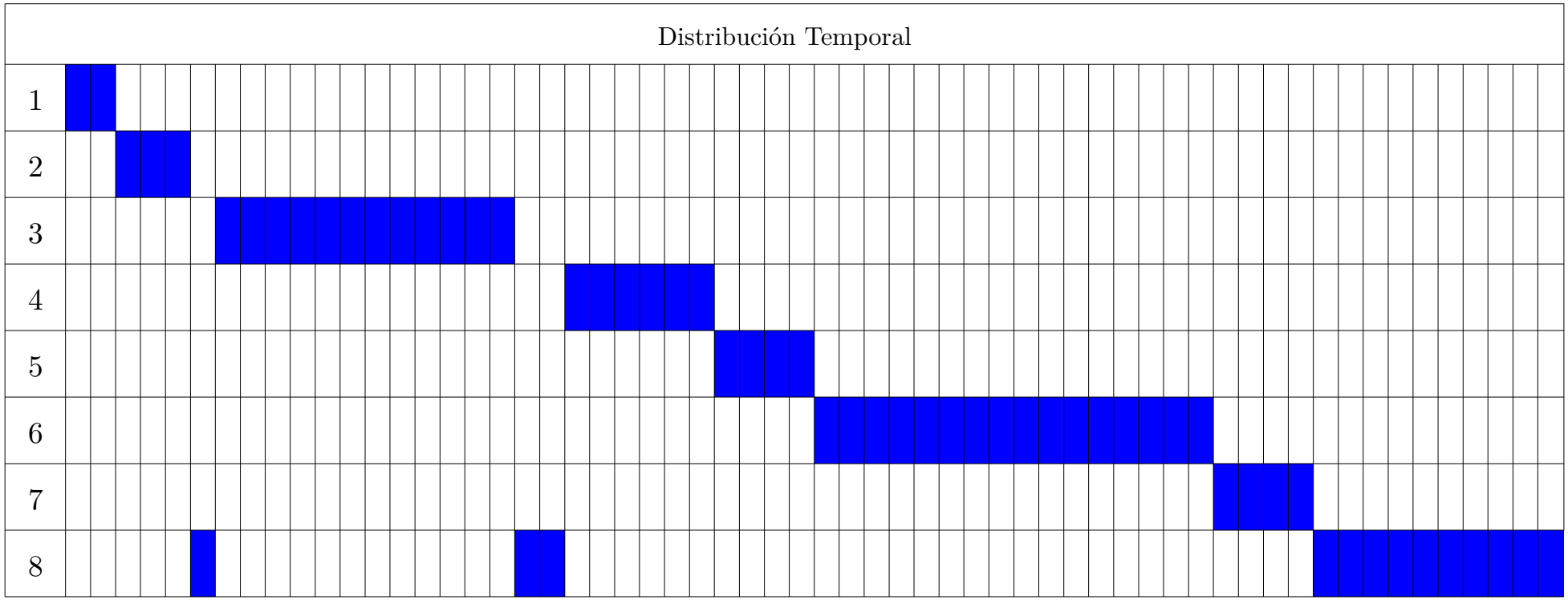


Tabla 1.1: Distribución Temporal

Capítulo 2
Herramientas

2.1 Introducción

En el presente capítulo se describen las herramientas que han sido empleadas para el desarrollo de la aplicación objeto de este Trabajo Fin de Grado.

2.2 Android Studio

Android estudio es el entorno de desarrollo integrado (IDE, Integrated Development Environment) oficial para desarrollo de aplicaciones Android. Este IDE fue anunciado en 2013, fecha en la que sustituyó a Eclipse como IDE oficial para el desarrollo de aplicaciones para Android. Actualmente está disponible la versión 3.1.

Este entorno tiene una serie de ventajas para el desarrollo de aplicaciones en Android:

- Plantillas para diseños de aplicaciones en Android.
- Consola de desarrollo.
- Debugger integrado.

- Refactorización de código específica para Android.
- Dispone de dispositivos Android Virtuales.
- Herramientas para el control de compatibilidad con versiones de Android.
- Editor de interfaz de usuario que permite diseño arrastrando y soltando elemento.

Este IDE se ha empleado para todo el diseño y desarrollo de la aplicación móvil. Al ser una herramienta oficial para Android, nos ha permitido desarrollar y depurar todo desde el mismo entorno. El desarrollo de aplicaciones para dispositivos móviles de manera nativa tienen ventajas e inconvenientes, algunos de ellos se han encontrado en este proyecto:

- Ventajas:
  - Uso de funciones nativas, código más ligero en su ejecución, aplicaciones de menor tamaño.
  - Herramienta con todas las funcionalidades de desarrollo integradas: editor de código, debugger y simulador.
- Desventajas:
  - El desarrollo solo es válido para una plataforma, implica desarrollar para cada sistema que se quiera desplegar la aplicación: Apple, Web, etc.
  - Aunque Android está basado en Java existe mucho código que es específico de la plataforma, lo que hace que la curva de aprendizaje sea mayor.

2.3 Atom

Atom es un editor de código multilenguaje, de código abierto que soporta plug-ins escritos en Node.js, además de soportar el control de versiones con GitHub.

Como se ha indicado anteriormente este editor de código abierto cuenta con gran aceptación por parte de la comunidad de desarrolladores, lo que ha hecho que haya disponible para el mismo infinidad de plug-ins, que hace que se pueda adaptar a casi cualquier lenguaje que se quiera emplear. En el caso de este proyecto se han empleado

plug-ins para el resaltado de lenguaje JavaScript y Html.

En este proyecto, Atom ha sido el editor que se ha empleado para todo el desarrollo, a excepción de la aplicación Android. Se ha empleado diversos lenguajes: JavaScript, Html, Css, MongoDB.

Se ha elegido este editor ya que a pesar de ser de código abierto y de uso gratuito es muy potente, si bien, en la mayoría de los casos solo se le encuentra ventajas al uso de un editor de estas características, también existe algún inconveniente:

- Ventajas:
  - Multilenguaje, un solo editor para toda la parte web y de base de datos del proyecto.
  - Ligero, es un software que a pesar de ser muy potente, su funcionamiento es soportable por cualquier equipo, no requiriendo especiales hardware muy potente.
  - Muy configurable, se pueden encontrar diversos plugins en la web para configurarlo, no solo en lo que respecta al reconocimiento del lenguaje de programación, sino también para añadirle herramientas de navegación por el código, diferentes tipos de resaltado, control de versiones, etc.
- Desventajas:
  - Debido a la gran diversidad de plugins disponibles y al nivel de actualización de la aplicación, se puede dar el caso de que al actualizar la aplicación se pierda compatibilidad con plugins instalados previamente, lo que genera fallos en la aplicación.
  - Dificultad, ante tantas opciones, para elegir la configuración más cómoda o más acertada.

2.4 GitHub

GitHub es una plataforma de desarrollo para alojar proyectos que permite subir y gestionar el código con la herramienta de control de versiones Git.

Esta plataforma normalmente permite alojar el código de forma pública, aunque mediante cuentas de pago, o de estudiante permite la creación de repositorios privados.

Entre otras ventajas, se encuentran las siguientes:

- Repositorio de código online, disponible desde cualquier PC con acceso a internet.
- Control de versiones, en un solo repositorio puedes alojar todas las versiones del código que se puedan realizar, permitiendo además en cualquier momento integrar entre las diferentes ramas de desarrollo.
- Copia de seguridad, al ser un repositorio online se garantiza la disponibilidad del trabajo guardado en el mismo, por lo que no es necesario el mantenimiento de diferentes copias locales.
- Integración completa con Git, git es una herramienta de control de versiones disponible en la consola de los sistemas operativos basados en Linux.

2.5 Cloud 9

Cloud 9 es un IDE online, pone a disposición de los usuarios un entorno de desarrollo online, además de máquinas virtuales para la ejecución de aplicaciones. Los servicios que se ejecuten en las máquinas virtuales son accesibles desde una URL pública.

En este proyecto se ha hecho uso de Cloud 9 para publicar el servidor web de la aplicación desarrollada.

Capítulo 3
Tecnologías

3.1 Introducción

Este capítulo describe los lenguajes, librerías y frameworks empleados en el desarrollo del proyecto. Se ofrece una descripción de los mismos además de indicar el uso que se le ha dado en el desarrollo.

3.2 Java

Java es el lenguaje de programación de alto nivel, orientado a objetos empleado para el desarrollo de aplicaciones, fue desarrollado en 1990 por Sun Microsystems. Desde sus comienzos fue empleado para el desarrollo de aplicaciones en sistemas embebidos, aunque posteriormente adquirió gran popularidad en el desarrollo web. Actualmente es uno de los lenguajes más empleados.

Java es un lenguaje compilado, aunque para su ejecución necesita de una máquina virtual (JVM). Esta dependencia de JVM hace que cualquier programa escrito en Java pueda ser ejecutado en cualquier dispositivo, lo que hace que los programas escritos en Java sean considerados multiplataforma.

Desde el nacimiento del sistema operativo para dispositivos móviles Android, Java ha sido el lenguaje de programación empleado para el desarrollo de aplicaciones nativas en esta plataforma.

Para el desarrollo de aplicaciones en Android, el lenguaje de programación Java se ha visto complementado con una serie de librerías y funciones desarrolladas específicamente para esta plataforma. Estas librerías son actualizadas periódicamente mejorando e incrementando su funcionalidad.

En este proyecto, el lenguaje de programación Java ha sido empleado exclusivamente en el desarrollo de la móvil.

3.3 JavaScript

JavaScript es el lenguaje de programación interpretado que permite implementar acciones complejas en entornos web.

Fue lanzado en junio de 1997, como un lenguaje para emplear en el lado cliente en combinación con html, para permitir realizar funciones más complejas. El lenguaje está desarrollado por: Netscape Communications Corp y Mozilla Fundation.

La última versión estable es ECMAScript 2016. Todos los navegadores modernos soportan al menos los estándares de ECMAScript 5.1.

Actualmente JavaScript es uno de los lenguajes de programación web más empleado, pudiéndose emplear tanto en el desarrollo del back-end como en el front-end. Muchos frameworks y librerías para web se basan en este lenguaje de programación, algunos ejemplos son: NodeJS, Express o AngularJS.

En este proyecto se ha empleado JavaScritpt tanto del lado del servidor como del lado cliente.

3.4 NodeJS

Es un entorno de ejecución multiplataforma para JavaScript, basado en el motor V8 de Google. Es el entorno que posibilita el uso de JavaScript del lado servidor. Fue creado en 2009 por Ryan Dahl y su mantenimiento de desarrollo está apoyada por la empresa Joyent.

Existe diversidad de frameworks que corren sobre node, algunos de ellos son: Express.js, Sails.js, Koa2, NestJS, LoopBack, Derby.js o Meteor.js.

En este proyecto se ha empleado NodeJS como entorno de ejecución para el servidor, que ha sido desarrollado con Express.js.

Algunas de las ventajas del uso de Node.js como entorno de ejecución son:

- Uso del mismo lenguaje de programación tanto en el lado servidor como en el lado cliente.
- Gran cantidad de librerías desarrolladas para este entorno, lo que permite ahorrar tiempo de desarrollo recurriendo a librerías que implementan las funciones necesarias.
- Diseñado para manejar gran cantidad de tráfico, en una aplicación real como la diseñada es un aspecto importante a tener en cuenta, ya que se manejará gran cantidad de conexiones simultáneas.

3.5 ExpressJS

Express.js es un framework para Node.js diseñado para crear aplicaciones web, dispone de diversas funcionalidades para el enrutamiento, gestión de sesiones, cookies, etc.

Es el estándar para la implementación del backend en el stack MEAN (acrónico de: MongoDB, Express.js, Angularjs y NodeJS) para el desarrollo de páginas web en JavaScript.

Su primera versión fue publicada por TJ Holowaychuk en Junio de 2014. El software se distribuye como software libre y de código abierto bajo licencia MIT.

En este trabajo se ha empleado ExpressJS para el desarrollo del servidor, haciendo uso principalmente de sus características para el manejo de rutas, así como la facilidad de acceso a los datos de las consultas.

3.6 AngularJS

AngularJS es un framework de JavaScirpt diseñado para el desarrollo de aplicaciones web de una sola pagina (SPA, single-page application).

Es un framework de código abierto desarrollado por Google, fue lanzado el 20 de octubre de 2010.

Al igual que ExpressJS, forma parte del stack MEAN, en este caso es está ideado para el desarrollo del lado del cliente. Angular sigue el patrón Modelo-Vista Vista-Modelo.

Angular dispone de una serie de directivas que se insertan en el código HTML como etiquetas, lo que permite al motor de Angular controlar el contenido que se muestra.

Este framework traslada parte de la función de control de la vista del lado del servidor a lado cliente, lo que descarga de trabajo el lado servidor, produciendo aplicaciones web mucho más ligeras.

AngularJS ha sido empleado en este trabajo para el desarrollo del frontend, donde se ha empleado para todo el diseño de las vistas y la interacción con el usuario. El proyecto se ha diseñado como una sola página web que cambia su contenido dinámicamente.

3.7 MongoDB

MongoDB es un sistema de base de datos NoSQL. A diferencia de los sistemas de bases de datos relacionales, los datos no son guardados en tablas, sin en estructuras de datos BSON, similares a los objetos JSON.

Este sistema de base de datos dispone una de un esquema dinámico, lo que hace que se pueda adaptar muy fácilmente a cualquier cambio en la estructura de datos, facilitando la integración de datos en ciertas aplicaciones de manera fácil y rápida.

Es el modelo de base de datos empleado en el stack MEAN, su lanzamiento oficial fue realizado en 2009 por la empresa 10gen Inc. actual MongoDB Inc., bajo licencia de código abierto.

Dispone de diversos paquetes Node que ayudan y simplifican su uso en servidores escritos en JavaScript, el más popular de ellos y el empleado en este proyecto es Mongoose. Mongoose se emplea tanto para la definición de datos como para la interacción con la base de datos.

3.8 Bootstrap

Bootstrap es una biblioteca multiplataforma de código abierto ideado para el diseño de interfaces web.

Fue lanzado en agosto de 2011 por Twitter como herramienta para la mejora de la consistencia de las herramientas internas de la compañía.

Entre otras características, dispone de plantillas, tipografías, menús, botones, etc. Pero quizá una de sus características más destacadas es su empleo de rejilla para el desarrollo de páginas web responsivas.

Mediante la definición de tamaños de los elementos y el número de columnas que ocupa, se puede definir la respuesta del contenido ante diferentes tamaños de pantalla. Mediante el uso de etiquetas se puede definir el comportamiento de la web para ser visualizada en diferentes dispositivos, esto permite simplificar el diseño ya que no es necesario realizarlo para varios dispositivos.

En este proyecto se ha empleado Bootstrap, principalmente, para dar un comportamiento responsivo al front-end de la aplicación web.

3.9 HTML

HyperText Markup Language (Lenguaje de marcas de hipertexto), es el lenguaje de marcas empleado en el diseño de páginas web.

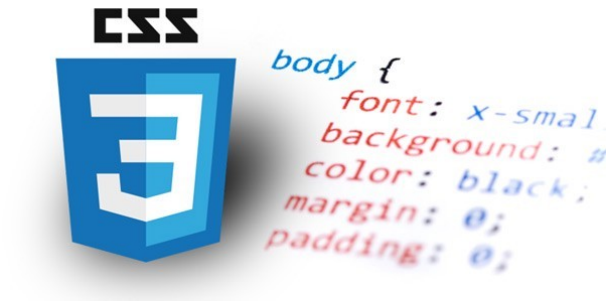
Es un lenguaje de marcado que cuenta con una serie de etiquetas básicas, estas etiquetas son empleadas para definir la posición y el tipo de contenido de una página web.

La primera versión de HTML fue lanzada en 1993, actualmente está en uso la versión 5. World Wide Web Consortium es la encargada del mantenimiento del estándar.

En este trabajo se ha empleado HTML exclusivamente en el diseño de la interfaz de usuario, en combinación con AngularJS y Bootstrap.

3.10 CSS

Hojas de estilo en cascada (CSS por sus siglas en inglés, Cascading Style Sheets) son las encargadas de definir el aspecto gráfico de un documento escrito con un lenguaje de marcado.



La primera versión de CSS fue lanzada en 1996, por lo que prácticamente desde los comienzos de la WWW ha estado unido al HTML, actualmente se encuentra en su versión 3. Al igual que el HTML, World Wide Web Consortium es la entidad encargada del mantenimiento del estándar.

CSS permite la separación del contenido de una página web de la definición de su diseño, de esta manera, el contenido de la página web se puede tratar de manera totalmente independiente de su diseño.

En este trabajo se han empleado hojas de estilo de Bootstrap, así como algunas definidas específicamente para el mismo.

3.11 Server Sent Events (SSE)

Server Sent Events es una tecnología push que permite a un navegador, o en nuestro caso, a una aplicación recibir actualizaciones automáticas, a través de una conexión HTTP, sin que estas sean consultadas por el equipo cliente.

SSE ha sido incluido por el World Wide Web Consortium en el estándar HTML a partir de su versión 5, aunque ya fue incluida en septiembre de 2006, como una característica experimental, en el navegador Opera.

En las comunicaciones HTTP, la máquina cliente es siempre quien inicia la comunicación, por este motivo, si no se emplea SSE, si se quiere que el contenido de una página web se actualice automáticamente, es necesario que el navegador o aplicación cliente esté consultando constantemente si hay actualizaciones.

Con SSE se consigue que, una vez establecida la conexión, sea el servidor quien

comunique al cliente la actualización, de esta manera, se minimiza el número de conexiones necesarias, con la consecuente disminución en el número de recursos necesarios, y se mejora el tiempo de comunicación a la máquina cliente.

Esta tecnología, a diferencia de otras como WebSockets o Long-Polling, establece un canal de comunicación unidireccional, es decir, los eventos solo son enviados desde el servidor hacia el cliente, el canal no se mantiene abierto para comunicaciones del cliente al servidor.

En este proyecto se ha empleado SSE para mantener un canal de comunicación permanente entre el servidor y la aplicación móvil.

3.12 EventSource-Android

Librería escrita en Java para la implementación de SSE en dispositivos Android como cliente (https://github.com/tylerjroach/eventsource-android).

Es una librería diseñada por Tyler Roach (https://github.com/tylerjroach) y publicada en GitHub desde enero de 2015. Actualmente está publicada la versión 1.2.11 publicada en julio de 2016.

El software es distribuido bajo Licencia BSD Simplificada o Licencia FreeBSD.

Esta librería provee de una serie de métodos para realizar la conexión al servidor emisor, además provee interfaces a implementar para la gestión de los eventos, lo que hace que sea una librería totalmente adaptable a cualquier proyecto Android que requiera emplear SSE.

En este proyecto se ha empleado EventSource-Android para la gestión de escucha de eventos del lado cliente en la aplicación Android.

3.13 Tarsos DSP

Tarsos DSP es una librería Java para el procesamiento de audio, provee interfaces para simplificar el uso de distintos algoritmos empleados en procesamiento de audio (https://github.com/JorenSix/TarsosDSP)
(http://0110.be/files/attachments/411/aes53_tarsos_dsp.pdf).

Esta librería empezó a ser publicada en marzo de 2011 en GitHub, y su última actualización publicada es de marzo de 2019. Su versión actual es la 2.4 de diciembre de 2016, aunque constantemente está actualizándose, mejorando los algoritmos existentes e

incluso añadiendo nuevos algoritmos.

Joren Six, Olmo Cornelis, Marc Leman son los desarrolladores de esta librería de procesamiento de audio.

Tarsos DSP se publica bajo Licencia Pública General de GNU (GNU Genera Public License o GNU GPL).

A pesar de ser una librería diseñada inicialmente para Java exclusivamente, gracias a ciertas adaptaciones, puede ser empleada para procesar audio en dispositivos Android, audio procedente de archivos o del captado por el micrófono del dispositivo.

En este trabajo se ha empleado Tarsos DSP para implementar uno de los modos de funcionamiento de la aplicación móvil, este modo de funcionamiento activa el micrófono del dispositivo, y en función del sonido ambiente y la ganancia de dicho sonido, muestra un color en pantalla.

3.14 Google Mobile Vision API

Mobile Vision API de Google es un framework que proporciona a los desarrolladores herramienta para la programación relacionadas con la visión artificial.

Aparte de un bloque común, existe tres grandes bloques en los que Mobile Vision API presta sus funcionalidades: detección de rostros, detección de textos y detección y lectura de códigos 2D y 1D.

Mobile Visiton API inicialmente fue lanzado por Google como un framework independiente, actualmente forma parte del Machine Learning Kit de Google. La versión empleada en este proyecto es Mobile Vision API.

En este proyecto, el framework de visión de Google ha sido empleado para la detección y lectura de códigos de barras integrado en la aplicación móvil, mediante la lectura de estos códigos se identifica la posición del usuario.

3.15 Mongoose

Mongoose es un paquete NodeJS empleado como interfaz con la base de datos MongoDB.

Mongoose proporciona infinidad de funcionalidades para el trabajo con bases de datos MongoDB, simplificando la creación y el trabajo con esquemas de bases de datos.

Este software se publica como Software Libre bajo Licencia MIT. Desde enero de 2018 está disponible la versión 5.0.0

En este proyecto se ha empleado Mongoose para el diseño y la conexión con la base de datos de la aplicación en el servidor.

3.16 Angular QR Code

Angular QR Code es una librería JavaScript que añade a angular la capacidad de generar códigos QR mediante el uso de directivas.

Esta librería ha sido publicada en GitHub bajo licencia MIT (https://github.com/monospaced/angular-qrcode).

En este proyecto Angular QR Code ha sido empleado para generar los códigos QR de cada zona en la interfaz web de la aplicación.

Capítulo 4
MPIC – Mobile Phone Image Composer

4.1 Introducción

En este capítulo se realizará la descripción de todos los componentes desarrollados para el funcionamiento del sistema objeto de este Trabajo de fin de grado.

A modo de resumen podemos dividir el sistema en tres grandes bloques:

- Aplicación móvil.
- Interfaz de administración web.
- Servidor.
- Base de datos.

De estos tres grandes bloques se describirá en cada caso: el funcionamiento básico, elementos de desarrollo implicados, interacción con otros elementos, interacción con el usuario y/o cualquier otro elemento que se considere necesario para una completa descripción de todos los componentes desarrollados.

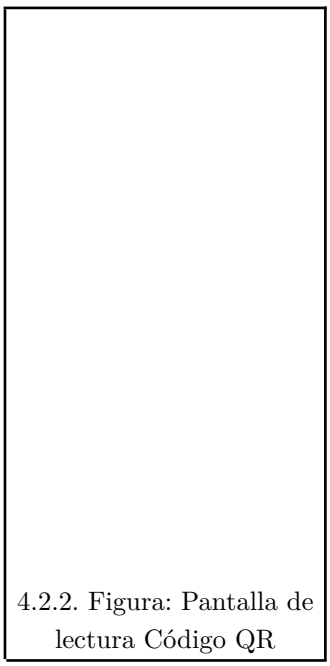
4.2 Aplicación móvil

Dentro del sistema, la aplicación móvil tiene dos funciones, por un lado es empleada para la lectura del código QR, que identifica la posición del dispositivo y por otro es el elemento que recibe los eventos desde el servidor y muestra en pantalla el color en función del evento recibido.

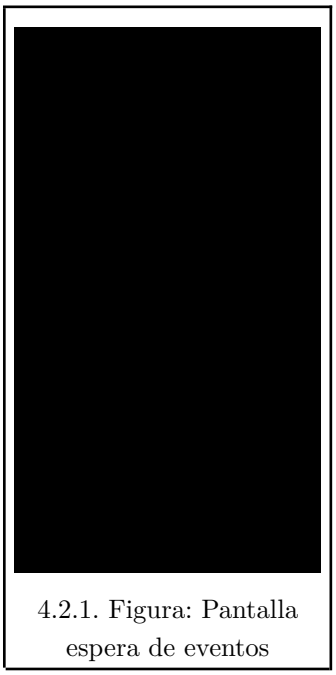
Esta aplicación ha sido desarrollada en Android Studio, empleando el lenguaje de programación Java. Esta aplicación es compatible con dispositivos móviles Android, ha sido desarrollada con el SDK en versión API 26 con compatibilidad desde la version API 22, esto proporciona compatibilidad con el 80,2% de los dispositivos actualmente en uso.

4.2.1 Descripción

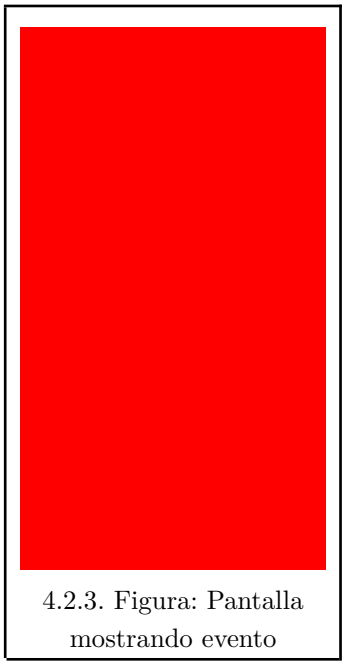
La aplicación móvil está compuesta por dos pantallas:



4.2.2. Figura: Pantalla de lectura Código QR



4.2.1. Figura: Pantalla espera de eventos



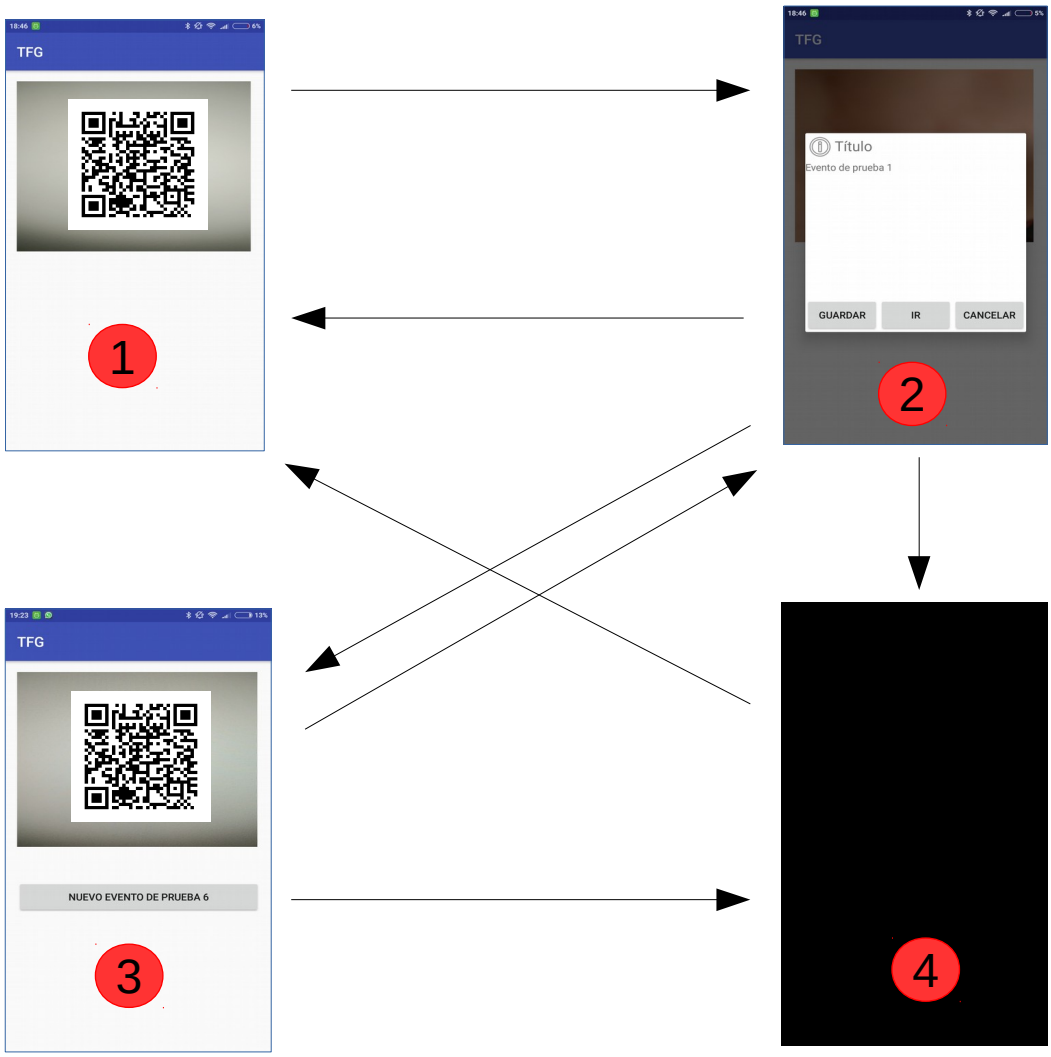
4.2.3. Figura: Pantalla mostrando evento

- Pantalla de lectura de códigos QR, es en la que se realiza la lectura de los códigos QR, así como los eventos en los que el dispositivo se encuentra registrado.
- Pantalla de espera de eventos y de muestra de colores a pantalla completa, esta pantalla es la que se muestra cuando el dispositivo está a la espera de recibir eventos del servidor, a su vez tiene tres estados, espera del primer evento, mostrar un color e interprete de sonido.

4.2.2 Funcionamiento de la aplicación

En este apartado se describe el funcionamiento de la aplicación, la interacción con el usuario, las transiciones entre pantallas y el paso de información entre pantallas.

En el siguiente esquema se muestras las posibles transiciones entre las pantallas de la aplicación:



4.2.4. Figura: Interacción de la aplicación

A continuación se describe que es cada ventana y que funcionalidad cumple dentro de la aplicación:

- 1: Ventana inicial, se muestra cuando se abre la aplicación, solo permite la acción de escanear código QR.
- 2: Ventana emergente, se muestra cuando se ha escaneado un código QR y el servidor devuelve información de un evento próximo en ese recinto.
- 3: Ventana que se muestra cuando el usuario a guardado un evento en 2, permite escanear otro código QR, así como acceder al evento guardado.
- 4: Ventana de espera a recibir eventos desde el servidor, es una ventana a pantalla completa que muestra un color sólido en función del modo de funcionamiento y los

eventos que reciba del servidor.

A continuación se describe las transiciones entre pantallas de la aplicación, que acciones las desencadenas y que información se pasa de la pantalla origen a la pantalla destino.

- 1 → 2 : Cuando se realiza la lectura de un código QR se realiza una consulta al servidor, este devuelve un objeto JSON con los valores del evento más próximo que se va a realizar en ese recinto, con esos datos se expone una ventana emergente que muestra al usuario tres opciones: guardar, ir o cancelar. Datos que se envían a la ventana emergente: {Description:String, EventID:String, ZoneID: String, SiteID: String, Fila: Int, Columna: Int }

- 2 → 1: Estando en la ventana 2, si el usuario pulsa el botón cancelar, la aplicación vuelve a la ventana inicial sin guardar ningún evento.

- 2 → 3 : Este cambio se produce cuando el usuario estando en 2 pulsa la opción de 'GUARDAR', se vuelve la la ventana de escaneo de código QR, pero se genera en la parte inferior un botón para acceder directamente al evento escaneado anteriormente. Datos que se envían al guardar un evento: {Type:String, Description:String, EventID:String, ZoneID: String, SiteID: String, Fila: Int, Columna: Int }

- 2 → 4 : El desplazamiento de la ventana 2 a la ventana 4 se produce cuando el usuario pulsa la opción 'IR' en la ventana emergente de 2. Al generar una nueva ventana se pasan los datos a través un array en el Intent: {Type:String, Description:String, EventID:String, ZoneID: String, SiteID: String, Fila: Int, Columna: Int }

- 3 → 2 : Si estando en 3 se vuelve escanear un nuevo código QR se vuelve a mostrar la ventana emergente con las opciones de 2. Datos que se envían a la ventana emergente: {Description:String, EventID:String, ZoneID: String, SiteID: String, Fila: Int, Columna: Int }

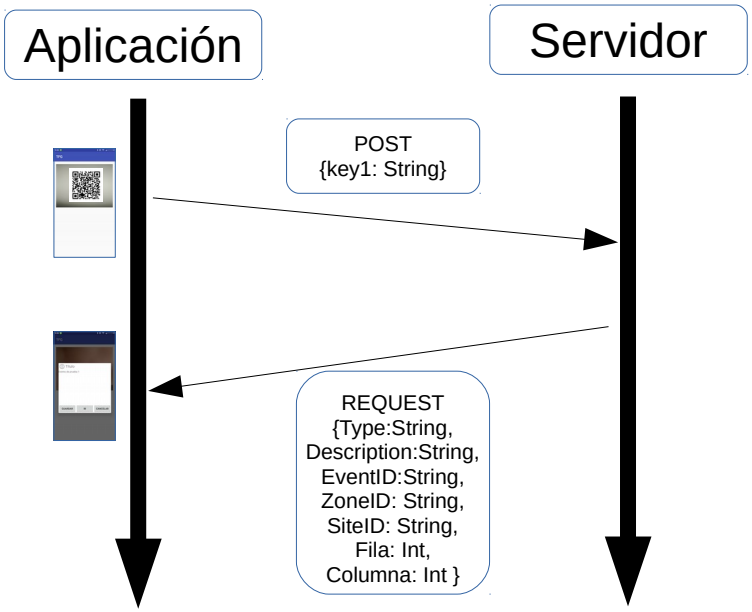
- 3 → 4 : Al pulsar el botón generado tras guardar un evento se cambia a la pantalla de espera de recepción de eventos desde el servidor. Al generar una nueva ventana se pasan los datos a través un array en el Intent: {Type:String, Description:String, EventID:String, ZoneID: String, SiteID: String, Fila: Int, Columna: Int }

- 4 → 1 : Es posible volver a la pantalla inicial si al estar en 4 se pulsa el botón de retorno del dispositivo móvil.

4.2.3 Comunicaciones entre la aplicación y el servidor

Se muestran a continuación las diferentes comunicaciones que se realizan entre la aplicación y el servidor, se hará referencia a la numeración de la figura 4.2.4 para indicar que que pantalla se realiza cada petición.

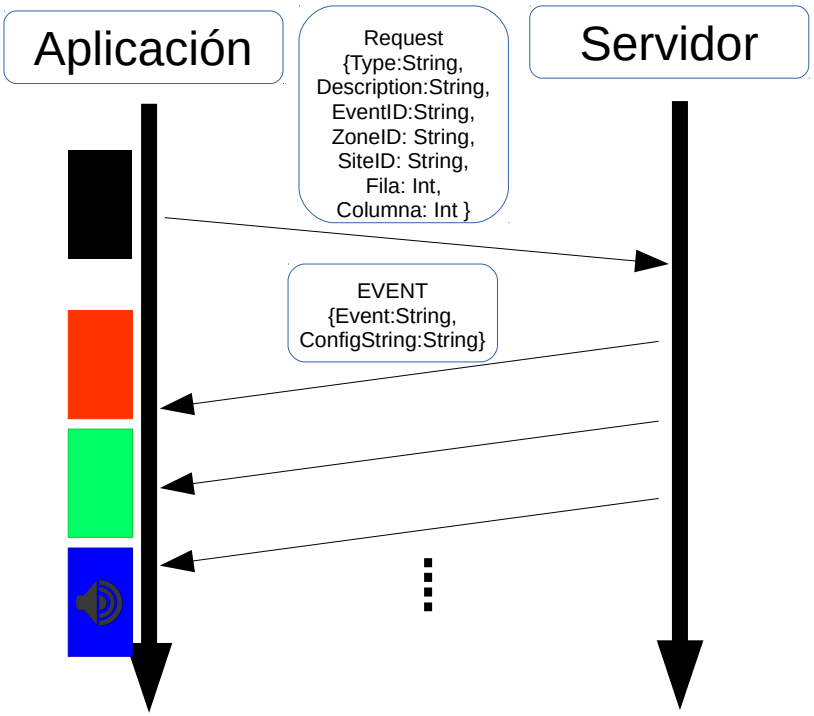
- 1, Solicitud de actos disponibles al leer un código QR. Esta solicitud se realiza mediante una petición post al servidor, se envía mediante un objeto JSON la información contenida en el código QR, en caso de haber eventos programados en el recinto, el servidor devuelve un objeto JSON con la información del evento futuro más próximo.



4.2.5. Figura: Comunicaciones aplicación - servidor 1

- 3, en esta ventana se produce la misma llamada al servidor que en la ventana 1 al escanear un nuevo código QR.

- 4, en esta ventana se realiza la conexión con el servidor para recibir eventos. La comunicación la inicia el dispositivo móvil, envía datos sobre su posición y sobre el evento al que está suscrito, una vez establecida la conexión, se queda esperando eventos del servidor, los eventos son múltiples y separados en el tiempo. En caso de cortarse la conexión por cualquier motivo, la aplicación al detectar esta situación vuelve enviar la petición de conexión.



4.2.6. Figura: Comunicaciones aplicación - servidor 2

4.2.4 Código de la aplicación

En este apartado se describirá brevemente el código desarrollado para el funcionamiento de la aplicación, explicando la estructura de clases y las funciones de cada una.

MainActivity:

Es la clase se define el comportamiento de la ventana principal de la aplicación, se implementa extendiendo la clase AppCompatActivity, que es una clase que pertenece a las librerías de Android. En ella se define las acciones a realizar al detectar un código de barras, se realizar la consulta al servidor con el resultado de la lectura de un código de barras y se muestra un cuadro de diálogo para acceder o almacenar un evento.

En esta clase se hace uso de dos objetos de otras clases: CameraSourcePreview y CustomAlertDialog. Se implementan métodos para: la conexión al servidor, mostrar cuadro de diálogo, añadir botones con eventos y cambiar de activity, aparte de implementar los métodos heredados de AppCompactActivity.

CustomAlertDialog:

Esta clase extiende Dialog e implementa View.OnClickListener. Dialog es una clase que implementa un cuadro de diálogo perteneciente a las librerías de Android, OnClickListener es un interface para el desarrollo de eventos de click en interfaces de aplicaciones Android.

Esta clase tiene dos sobrecargas del método constructor de la clase, de esta manera se pueden generar objetos en función de los datos recibidos en la consulta al servidor.

Se implementan los métodos del Interface OnClickListener para personalizar el comportamiento de los botones del cuadro de diálogo.

CameraSourcePreview:

En esta clase extiende ViewGroup y es la encargada de definir la superficie donde se muestra la imagen de la cámara cuando está escaneando un código de barras.

ColorActivity:

ColorActivity extiende AppCompactActivity. En esta clase se implementa una superficie que se muestra a pantalla completa, esta superficie es la que se emplea para mostrar el color en función de las ordenes que se reciban del servidor.

Para maximizar la superficie que se emplea para mostrar color se oculta la barra de notificaciones. También se maximiza el brillo de la pantalla desde la aplicación.

Esta clase es en la que se realiza la conexión SSE con el servidor para la escucha de eventos, en ella se definen las acciones en función del evento recibido.

Aparte de lo anterior, también se emplea un objeto de la clase SountToColor, para iterpretar los sonidos que se escuchan desde el micrófono del dispositivo.

SoundToColor:

Esta clase, mediante una implementación de la librería Tarsos DSP, se emplea para transformar en color lo que se escucha por el micrófono del dispositivo.

Para este cometido se emplea un PitchDetector y un SilenceDetector, que son objetos de la libería Tarsos DSP.

PitchDetector, es un detector de frecuencia, lo que hace es devolver la frecuencia media de un intervalo.

SilenceDetector, es un medidor de ganancia del sonido escuchado, devuelve la ganancia medida del sonido detectado por el dispositivo en un periodo.

Al constructor de esta clase se le pasan 6 parámetros: Color inicial, Color Final, Frecuencia mínima, Frecuencia máxima, Ganancia mínima y ganancia máxima.

Con los elementos anteriormente indicados y los parámetros pasados lo que se hace es, transformar la frecuencia del sonido escuchado por el micrófono en un color, y el nivel de ganancia en un nivel de transparencia de dicho color, de esta manera se obtiene un interprete que transforma el sonido en color.

4.3 Interfaz de administración web

La interfaz de administración web tienen la función de crear sitios y eventos, además permite generar los códigos QR para identificar las zonas donde se encuentran los usuarios.

Esta interfaz ha sido desarrollada empleando HTML5 y JavaScript como lenguajes de programación. Se han empleado AngulaJS para el diseño funcional de la aplicación. Para el diseño gráfico y distribución de los elementos se ha empleado Bootstrap.

4.3.1 Descripción

La aplicación se ha diseñado como una aplicación en una sola página (SPA, Single Page Application), el contenido cambia de manera dinámica dentro de la web, en ningún momento se cambia página web.

4.3.1. Figura: Vista general de la aplicación web

La parte interactiva de la web está compuesta por cinco apartados:

- Inicio: Lugar de entrada a la web, es la portada y da acceso a los cuatro apartados siguientes:
- Crear Sitio: Es el apartado donde se definen los recintos donde se desarrollarán los actos en los que se va a emplear el sistema.
- Editar Sitio: Permite editar los sitios creados en el apartado anterior.
- Crear Evento: A partir de un sitio permite crear eventos, en este apartado se definen todos los aspectos del evento: Fecha y hora de comienzo, duración y lugar, para luego definir las transiciones.
- Generar QR: En este apartado se puede imprimir los códigos QR a situar en las diferentes zonas de un recinto.

Si bien existe datos comunes entre los diferentes apartados de la web, no existe interacción entre ellos, todo el paso de información se hace a través del servidor y de la

base de datos.

4.3.2 Inicio

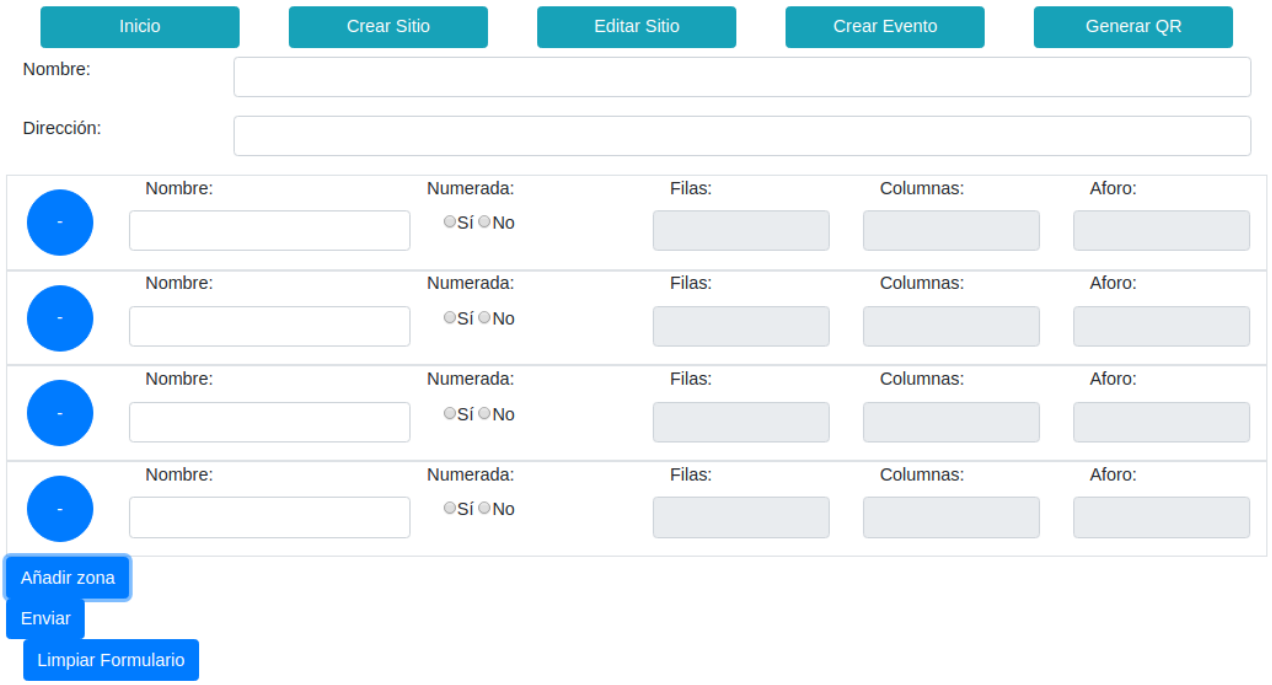
4.3.2. Figura: Vista inicio

Es el lugar de entrada a la web. Al igual que los otros apartados, dispone en la parte superior de una barra de navegación que serve de acceso directo al resto de apartados.

En el cuerpo principal se dispone de cuatro apartados, dispuestos en dos filas y dos columnas, clicando sobre cada uno de estos apartados se puede acceder a las diferentes zonas de la web.

4.3.3 Crear Sitio

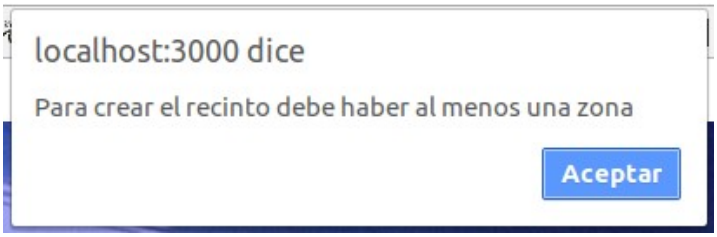
Es el apartado de la web que se emplea para la definición de los recintos donde se desarrollan los eventos.



4.3.3. Figura: Vista Crear Sitio

Para la definición del recinto se indica el nombre y la dirección del mismo, para posteriormente definir las zonas. Se pueden añadir zonas pulsando sobre el botón añadir zona, para eliminarlas se emplea el botón circular con el signo "-" que ha en cada fila, si no se deja al menos una zona aparecerá un mensaje de error, ya que debe haber al menos una zona en un recinto.

Dentro de cada zona habrá que definir el nombre de la misma, para luego definir si la



4.3.4. Figura: Respuesta del servidor

zona es con asientos numerados o no. Los campos a rellenar posteriormente dependen de si la zona es numerada o no.

4.3.5. Figura: Detalle zonas

Como se observa en la imagen anterior, los campos: filas, columnas y aforo se desactivan en función de si la zona es numerada o no.

Finalmente en la parte inferior se dispone de dos botones, el primero de ellos "Enviar" sirve para enviar los datos al servidor y guardarlos en la base de datos.

El último botón "limpiar formulario" se emplea para vaciar el formulario si no se desea guardar el recinto creado.

Todos los datos generado en la web se almacenan paralelamente en un objeto JSON. Este objeto tiene una doble finalidad, guardar los datos de la interfaz web además y ser el portador de los datos que se envían al servidor para ser almacenados en la base de datos. La estructura del objeto es la siguiente:

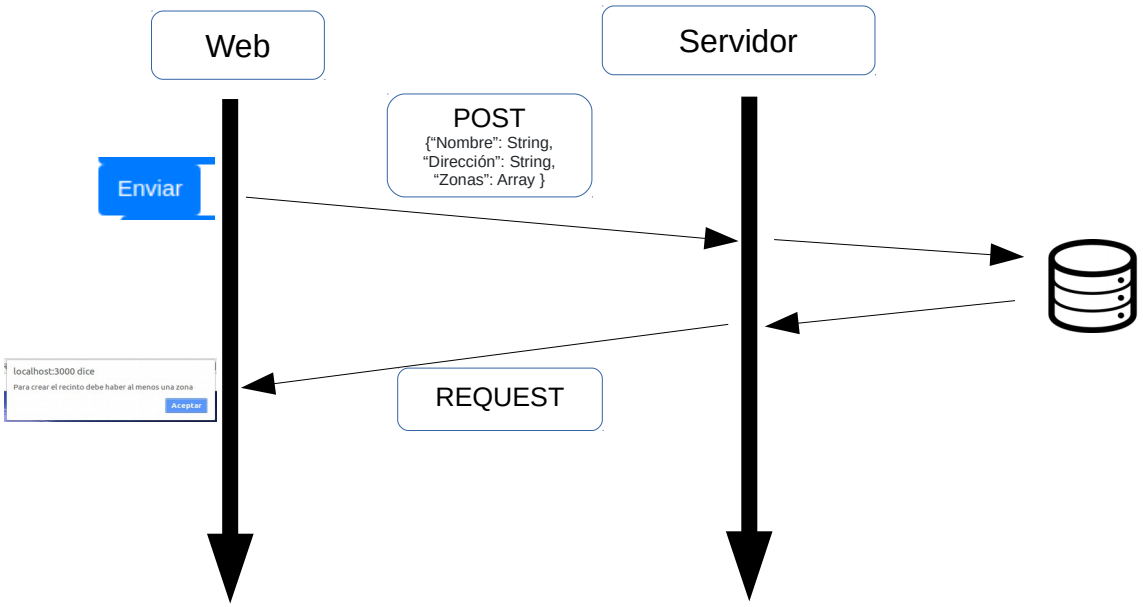
{"Nombre": String,
"Dirección": String,
"Zonas": Array }

Cada zona es a su vez definida por otro objeto JSON que se inserta en el array Zonas.

{"Nombre": String,
"Numerada": Bool,
"Columnas": Int,
"Filas": Int,
"Aforo": Int}

Los valores del objeto JSON se asocian a los campos en la interfaz mediante la directiva Angular ng-model.

Crear sitio establece una conexión con el servidor cuando se pulsa el botón "Enviar", realizar un petición POST adjuntando el objeto JSON que define el sitio creado. En el servidor una vez se recibe la petición POST se realiza el proceso de escritura en la base de datos de la información generada, una vez escrita dicha información se devuelve un mensaje, si todo ha sido correcto se devuelve un código 200, en caso de error se devuelve un código 500 con el mensaje de error.



4.3.6. Figura: Comunicaciones con el servidor

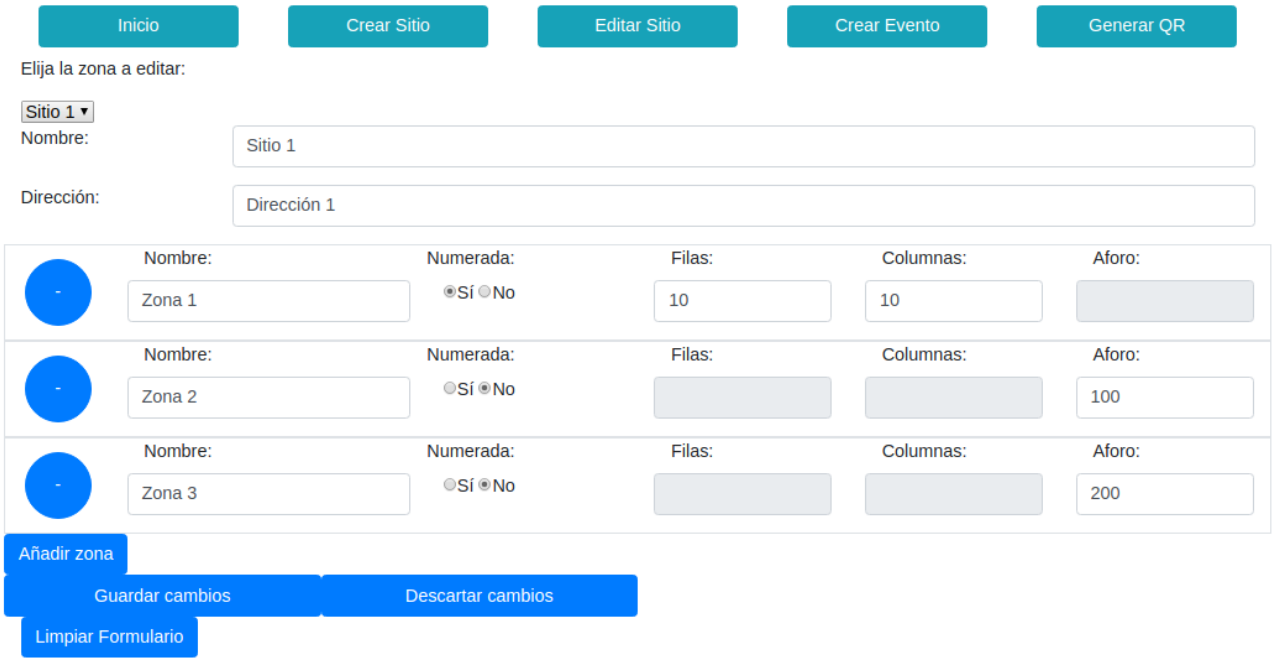
4.3.4 Editar Sitio

Este apartado de la web es similar a Crear sitio, en este caso lo que se permite es modificar un sitio ya creado. Para cumplir esta función inicialmente se da la opción de elegir el sitio que se quiere modificar.

4.3.7. Figura: Detalle selección de sitio

Una vez seleccionado el sitio a modificar se carga en pantalla la información completa del sitio a modificar.

En esta pantalla se podrá modificar cualquier dato del sitio seleccionado: Nombre, dirección, número de zonas, características de la zona y añadir nuevas zonas, adicionalmente existen opciones para salvar los cambios o descartarlos, así como para vaciar el formulario.



4.3.8. Figura: Vista Editar Sitio

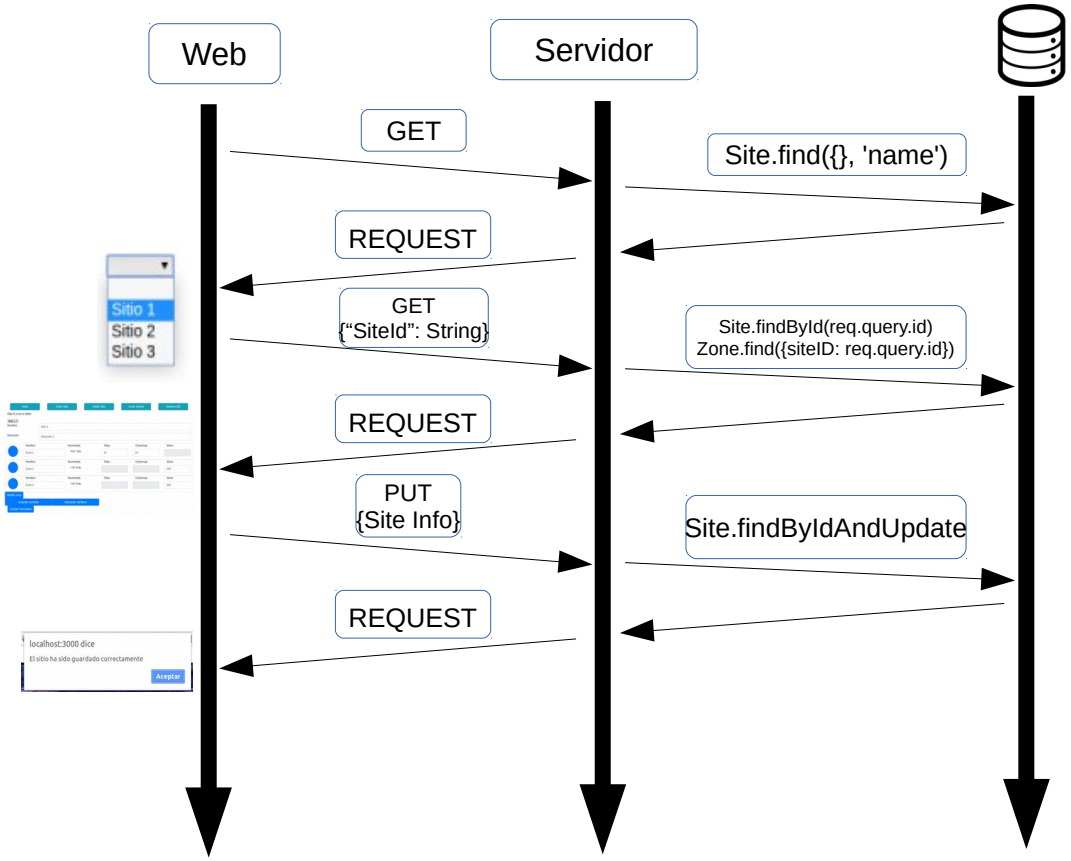
Al igual que en el caso de Crear Sitio, toda la estructura de datos se apoya en un objeto JSON, en este caso no se parte de un objeto JSON vacío sino que al seleccionar el sitio a modificar, se descarga del servidor un objeto JSON con su definición, este objeto es el que sirve de punto de partida para las modificaciones a realizar.

Los diferentes elementos del formulario son relacionados con el objeto JSON mediante la directiva Angular ng-model.

El esquema de comunicaciones habitual con el servidor al trabajar con este apartado sería el siguiente:

- Al cargar “Editar Sitio” se realiza una petición GET para obtener un array con todos los posibles sitios a modificar.
- Al Seleccionar el sitio se realiza otra petición GET con el ID del sitio seleccionado, para descargar toda la información del mismo, en base al objeto obtenido se construye el formulario.
- Una vez realizados los cambios, si se pulsa “Guardar Cambios”, se realiza una petición PUT, con esta petición, se modifican los datos existentes en el servidor, se modifica la entrada en la tabla Sitios y se Añaden o eliminan las zonas correspondientes en la tabla Zonas.

- Al descartar cambios, lo que se realiza es recuperar el objeto JSON original, descartando el modificado.



4.3.9. Figura: Comunicaciones con el servidor y base de datos

4.3.5 Crear evento

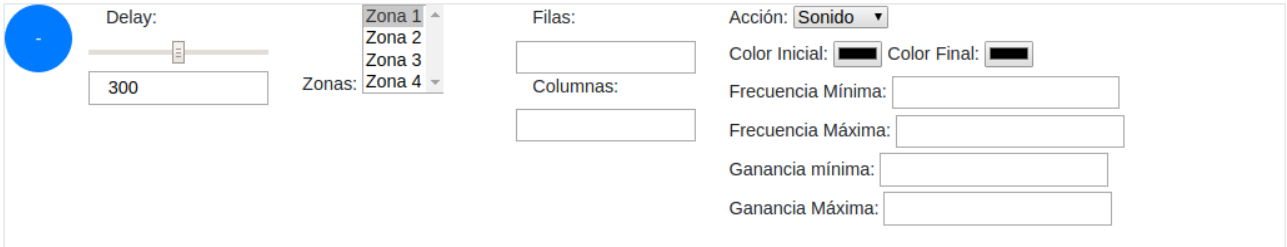
En este apartado de la web se realiza la composición o mosaico que se ejecutará durante el trascurso del evento.

4.3.10. Figura: Vista Crear Evento

Como se muestra en la imagen anterior, al cargar la página se muestra una serie de campos a rellenar, los campos mostrados y habilitados son el nombre del evento y la selección del lugar, hasta que el lugar del evento no es seleccionado no se habilita el resto de campos: fecha y hora de inicio y la duración, además se habilita la posibilidad de añadir transiciones.

4.3.11. Figura: Vista Crear Evento 2

Dentro de cada transición se debe configurar lo siguiente: Delay o retraso respecto al inicio del evento, las zonas que se ven afectadas y la acción, sólo en caso de que se seleccione una sola zona y esta sea numerada, se podrá seleccionar el número de filas y columnas de asientos a los que se ven afectados. Se pueden seleccionar múltiples zonas para una sola transición.



4.3.12. Figura: Detalle de transición

Los campos filas y columnas de cada transición están diseñados para que en en una sola transición se puedan seleccionar múltiples valores de filas o columnas. Por ejemplo, si introducimos la cadena 1,3,5-7, estaríamos seleccionando las filas o columnas 1, 3 y el rango de la 5 a la 7.

El slider Delay funciona del tal manera que el Delay de una transición nunca será inferior al de la anterior. De esta manera, si se aumenta el Delay de una transición esta

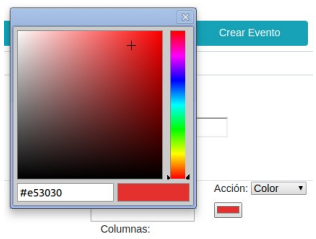
arrastrará a todas las posteriores, en caso de que se vuelva a disminuir, las posteriores volverán a su valor original. En la siguiente secuencia de imágenes se muestra este funcionamiento.

4.3.13. Figura: Funcionamiento sliders de tiempo

Como se observa en el caso 1 los tres valores de los sliders van en aumento, en la figura 2 se aprecia que al arrastrar el primero de ellos los demás se ponen al mismo valor, cuando disminuimos el valor del primer slider, los siguientes vuelven a recuperar su posición original.

En función de la acción seleccionada se habilita una serie de campos a configurar.

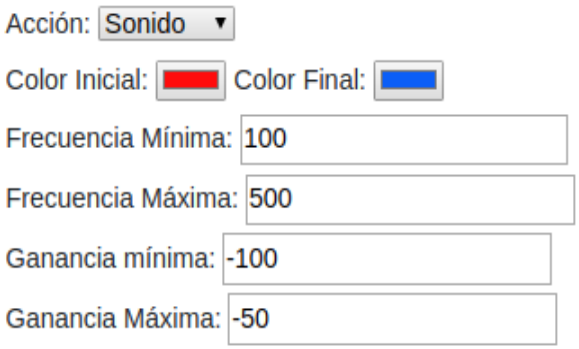
- Al seleccionar la acción color se habilita un selector de color para poder elegir el color a mostrar.



4.3.14. Figura: Detalle selección de color

- Si la acción seleccionada es sonido se habilita dos campos para selección de color así como cuatro campos numéricos para seleccionar la frecuencia mínima, frecuencia

máxima, ganancia mínima y ganancia máxima, siendo estos los seis parámetros necesarios para configurar esta acción.



4.3.15. Figura: Detalle configuración modo sonido

Las transiciones pueden ser añadidas o eliminadas en cualquier momento del diseño del evento, se pueden eliminar aleatoriamente haciendo uso del botón "-" que acompaña a cada transición y añadir una nueva al final de la secuencia haciendo uso del botón "Añadir transición".

Una vez finalizado el diseño, para que sea guardado en base de datos, es necesario pulsar el botón "Enviar".

Al igual que en los casos anteriores, toda la estructura de datos que estructura la web se almacena en un objeto JSON, los campos de los formularios se relacionas con los valores del objeto JSON mediante la directiva Angula ng-model. Este objeto se aprovecha posteriormente para el envío de datos a la base de datos, adjuntándolo a la petición POST.

En este apartado de la web se realiza tres interacciones con el servidor:

- GET para obtener los sitios disponibles para la lista lugar, se obtiene solo el listado de sitios y los ID en la tabla sitios.
- GET para obtener el detalle del recinto una vez seleccionado el mismo, en esta petición de descarga el detalle de zonas.
- POST con toda la información del evento diseñado para ser almacenado en base de datos.

4.3.6 Generar QR

En este apartado de la web se ofrece la posibilidad de generar los códigos QR que identifican a todas las zonas o asientos de los recintos.

Desde el punto de vista del usuario el funcionamiento de este apartado es muy sencillo, simplemente debe seleccionar el recinto y pulsar el botón “Generar QR”.

4.3.16. Figura: Detalle Generar QR

Una vez seleccionado el sitio y pulsado el botón se generará de manera secuencial, en la misma web, todos los códigos QR disponibles para el recinto seleccionado.

4.3.17. Figura: Detalle Generar QR

En la etiqueta QR generada se identifica el sitio y la zona por su ID y la fila y columna si la etiqueta identifica a una localidad numerada, estos datos se codifican de la siguiente manera: en caso de ser una zona numerada <ID Sitio>#<ID Zona>#<fila>#<columna>; en caso de ser una zona no numerada, los campos fila y columna se suprimen.

Este apartado de la web realiza con interacciones con ser servidor para obtener datos:

- GET para obtener los sitios disponibles para la lista lugar, se obtiene solo el listado de sitios y los ID en la tabla sitios.
- GET para obtener el detalle del recinto una vez seleccionado el mismo.

4.4 Servidor

La función dentro de la aplicación es doble, por un lado es el servidor para la web del proyecto, y por otro es el punto de conexión de los dispositivos móviles. El servidor ha sido desarrollado en ExpressJS, empleando como lenguaje de programación Javascript.

Aprovechando las ventajas de ExpressJS, las rutas del servidor son totalmente independientes de la estructura de carpetas del mismo, esto ha permitido plantear una estructura de carpetas para distribuir los archivos de una manera que resulte lógica desde el punto de vista de la programación.

En el servidor también se han implementado una serie de funciones auxiliares para el cumplimiento de las funciones del mismo.

4.4.1 Rutas

app.use("/",express.static('public')):

Ruta por defecto para servir todos los ficheros solicitados, se monta el nivel raíz del servidor en la carpeta public. De esta manera se puede aislar en esta carpeta todos los ficheros para la interfaz web, separándolo de la lógica del servidor que sí se monta en la raíz del proyecto. Por ejemplo, el fichero index.html en el navegador se solicita en /index.html*, pero en el servidor se encuentra en* /public/index.html.

app.use("/img", express.static('public/imagenes')):

Ruta por defecto para localización de las imágenes. Debido a la estructura de carpeta del proyecto, existen ficheros html que se encuentran en subcarpetas de la carpeta public, al igual que la carpeta imágenes, esto generaba una serie de problemas para encontrar las imágenes. Poniendo esta ruta se simplifica el problema, ya que no tienes que tener en cuenta la ruta relativa, sino que todas las imágenes se referencian como /img/nombre de la imagen.

app.use("/node_modules",express.static('node_modules')):

Ruta por defecto para localizar los módulos node necesarios en la interfaz web, se

monta esta ruta para que AngularJS pueda encontrar módulos que necesita para su ejecución.

app.post('/conection', function (req, res){}):

Punto de conexión de la aplicación móvil cuando un código QR es leído, esta conexión se realiza mediante una petición post. Esta petición contiene un objeto JSON con los datos del código QR leído. Dentro de la función de identifica el evento más próximo en función de la localización del dispositivo móvil y se devuelve un objeto JSON con la información de dicho evento.

app.get("/sse", function (req, res , next){}):

Esta ruta es donde conectan los dispositivos móviles para al escucha de eventos. Dentro de la función es donde se desarrolla el sistema de envío de eventos a los dispositivos móviles. La conexión se realiza mediante una petición GET.

SSE se basa en el siguiente modo de funcionamiento. El cliente establece la conexión con el servidor y éste la mantiene abierta. Cuando se establece la conexión se envía un string que establece el tiempo de reintento en caso de perder la conexión:

'retry: 1000\n'

Cada vez que se envía un evento, se envía una trama que define el evento y sus parámetros. Para finalizar el evento se envían dos saltos de línea seguidos:

'event:Color \n'

'data:' + param + '\n\n'

app.post('/set/evento', function (req, res) {}):

Esta ruta es empleada por el servidor para enviar un evento para ser guardado, esta conexión se realiza mediante una petición POST y se adjunta un objeto JSON con la definición del evento para ser guardado en base de datos. Para guardar un evento en la base de datos se guarda una entrada de un objeto Evento asociado mediante su ID a múltiples objetos transición. El guardado de la transiciones se realiza de manera recursiva. Una vez guardado todas las transiciones, si todo ha sido correcto, se devuelve un código 200, en caso de error se devuelve un código 500 con el código de error.

app.post('/set/site', function (req, res) {}):

Esta ruta es la empleada para guardar en base de datos un sitio creado en la interfaz web de la aplicación. La conexión se realiza mediante una petición post que adjunta un objeto JSON con la definición del sitio a crear. Para guardar en un sitio en base de datos

se crea un objeto del tipo sitio en base de datos, que se asocia a múltiples objetos del tipo zona mediante su ID. El guardado de las zonas se realiza de manera recursiva. Una vez guardadas todas las transiciones, si todo ha sido correcto, se devuelve un código 200, en caso de error se devuelve un código 500 con el código de error.

app.get('/get/sites', function (req,res){}):

Ruta empleada para obtener la lista de sitios almacenados en base de datos. En caso de que la consulta sea a base de datos sea correcta, se devuelve un objeto JSON con todos los elementos encontrados, esta respuesta se envía con código 200. En caso de error, se devuelve código 500 acompañado del mensaje de error.

app.get('/get/site', function (req,res) {}):

Ruta para obtener los detalles de un sitio, a diferencia de la ruta anterior, en esta no se obtiene un listado de sitios, se obtiene los detalles de un sitio, el sitio a buscar en la base de datos es especificado en la petición GET mediante el ID del mismo. Si la consulta es correcta se devuelve un objeto JSON con los destalles del sitio buscado con un código de respuesta 200, en caso de error, se devuelve un código 500 acompañado de los detalles del error.

app.get('/getZonesBySiteName', function (req,res) {}):

Esta ruta se emplea para la obtención de las zonas de un sitio especificando el mismo. En la petición GET se recibe el ID del sitio del cual se buscan sus zonas y se devuelve un objeto JSON que contiene un array de zonas con los detalles de cada una de ellas acompañada de un código 200, en caso de error, se devuelve un código 500 acompañado del detalle del error. Esta consulta se emplea para complementar la consulta "/get/site" y obtener todos los detalles del sitio.

app.put('/edit/site', function (req, res) {}):

Esta ruta es empleada para la actualización en base de datos de sitios que han sido modificados en el apartado "modificar sitio" de la web. Es una petición PUT que recibe como parámetro un objeto JSON con los datos del sitio modificado. Para modificar el sitio en base de datos, lo que se hace es eliminar las zonas existentes y crear nuevas zonas con los datos del objeto JSON y posteriormente se modifica objeto correspondiente al sitio de tipo site. Si el sitio es almacenado correctamente se devuelve un código 200, en caso de error se devuelve un código 500 junto con los detalles del error.

4.4.2 Funciones

Se describe en este apartado las funciones implementadas en el servidor, describiendo en cada caso los parámetros que recibe y la función de desarrolla.

function inicializarOnlineClients () {}

Esta función no recibe parámetros, cada vez que es llamada lo que se hace es que se consulta en la base de datos la lista de sitios y se actualiza el objeto OnLineClients, este objeto es donde se almacenan las conexión a los diferentes dispositivos móviles que se conectan. Esta función es llamada solamente en el arranque del servidor. Cada vez que un sitio nuevo es creado, se actualiza OnLineClients solamente añadiendo ese nuevo sitio.

function recorrerEventos () {}

Esta función sin parámetros es llamada en el arranque del servidor y cada vez que se crea un evento, en ella lo que se haces es consultar la lista de eventos en la base de datos, y por cada evento se crea un timer para que sea ejecutado a la hora indicada. Todos estos timers son almacenados en el objeto EventsContainer, que almacena todos los eventos pendiente de ejecución.

function recorrerArray(array, accion, parametros) {}

Esta función es una función recursiva que recibe tres parámetros, el array a recorrer, la acción y los parámetros a enviar a cada uno de los elementos finales del array. En ella lo que se hace es recorrer recursivamente un objeto hasta encontrar una función, función que es ejecutada pasándole la acción y los parámetros indicados en los argumentos. Esta función es empleada para enviar eventos de manera masiva, a todos los clientes conectados a una misma zona.

4.5 Base de datos

MongoDB es una base de datos NoSQL orientada a documentos, que guarda la estructura de datos en BSON (similar a JSON). Este tipo de base de datos cuenta con gran auge en la actualidad debido a las ventajas que presenta respecto a otros sistemas de bases de datos: velocidad, optimización de espacio y versatilidad.

4.5.1 Estructura de la base de datos

Aunque en las bases de datos no relacionales no hay una estructura de datos predefinida en el diseño sí que se ha tenido en cuenta una estructura mínima para el diseño de las colecciones.

El objetivo de la base de datos es almacenar los datos necesarios para definir los sitios y los eventos, con este objetivo se han creado 4 tipos de documentos: sitios, zonas, eventos y transiciones. En la siguiente figura se muestra la estructura de los diferentes objetos y la relación entre los mismos:

4.5.1. Figura: Estructura de la base de datos

Sitio:

En este objeto se almacena la información básica de un sitio, nombre y dirección, la clave primaria es un ID auto asignado, único para cada objeto. La información completa de un sitio se complementa con un array del siguiente objeto.

Zona:

En este objeto se almacena la información necesaria para definir las diferentes zonas que definen un sitio, la clave primaria es un ID auto asignado, único para cada objeto, se vincula al sitio añadiendo la clave primaria del sitio como clave foránea. El resto de parámetros son empleados para definir la zona: nombre, tipo, número de filas, número de columnas y aforo.

Evento:

En este objeto se almacena la información básica de un evento, al igual que los otros objetos, tienen una clave primaria que es un ID auto asignado, se almacena también el ID

del sitio donde se desarrolla el evento, que es almacenado como clave foránea. El resto de valores son de definición del evento: nombre, fecha y hora de inicio y duración.

Transición:

En este tipo de objetos se almacena la información básica de las transiciones que componen los eventos, el objeto cuenta con los campos necesarios para definir una transición, la clave primaria es un ID auto asignado, luego se añade el ID del evento al que pertenece, así como los ID de las zonas que se ven afectadas por esta transición, almacenando sus ID en un array. El resto de campos sirven para definir la propia transición: Tipo, las filas y columnas a las que afecta, la acción a realizar y el string de configuración.

4.5.2 Desarrollo de la base de datos

Para el desarrollo de la base de datos se ha empleado Mongoose. Mongoose es un paquete node que facilita tanto la definición de objetos para las colección con la conexión e interacción con la base de datos.

Para la definición de los objetos de la base de datos se han creado cuatro definiciones en JavaScript almacenadas en sendos ficheros: event.js, site.js,, transition.js y zone.js.

Para la consulta, escritura y actualización de la base de datos se ha hecho uso de las funciones que Mogoose dispone para ello, dentro de las diferentes rutas en el servidor.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

La herramienta desarrollada en este Trabajo de Fin de Grado surge de una idea práctica, algo que podría ser un proyecto comercial, a pesar de esto ha sido de gran aprendizaje y ha requerido dedicarle tiempo de investigación. Este trabajo ha permitido experimentar con lo que podría ser un trabajo real, partir de una idea conceptual y llevarla a cabo empleando diferentes habilidades, lenguajes y tecnologías.

Este trabajo ha permitido poner en práctica múltiples habilidades adquiridas durante el estudio de la carrera, algunos ejemplos son: Bases de datos, diseño de interfaces de usuario, programación de aplicaciones interactivas. En otros casos ha habido buscar información para, combinado con la base adquirida, llegar en encontrar solución a los problemas que se planteaban, un ejemplo claro de este último caso es el empleo de Server Sent Events, una metodología para la cual ha habido que investigar a lo largo de este Trabajo de fin de Grado.

Si bien, este ha sido un trabajo totalmente académico y, por las horas de dedicación, los recursos disponibles y los conocimientos que requiere, no se ha podido llegar al desarrollo de una aplicación comercial. Sin embargo, se han sentado las bases de lo que podría ser una aplicación de uso comercial futura.

5.2 Líneas futuras

Si bien se ha desarrollado una aplicación mínimamente funcional, existe diversos aspectos que pueden ser planteados como líneas de trabajo futuro:

- **Desarrollo a otras plataformas:** Desarrollo de la aplicación móvil para otras plataformas: IOS y Windows.

- **Implementar otros modos de funcionamiento:** Aumentar los diferentes tipos de eventos para no limitar los modos a "color" o "sonido", estos modos podrían ser: aleatorio o sorteo.

- **Securización de la aplicación:** En este desarrollo no se ha tenido en cuenta la aplicación de medidas de seguridad que serían de aplicación en un proyecto real, algunos ejemplos donde hay que emplear medidas de seguridad pueden ser los siguientes: gestión de usuarios, control de sesiones, securización de la base de datos e implantación de medidas de seguridad en la base de datos.

- **Optimización del tráfico:** En una aplicación real de estas características habría que tener en cuenta la gestión de miles o decenas de miles de conexiones simultaneas. Para poder gestionar este tráfico habría que hacer estudios de cuanta carga supone para un servidor y como se podría optimizar.

- **Balanceo de carga:** No solo habrá que optimizar el tráfico, en aplicaciones reales con mucha concurrencia, es necesario el desarrollo de aplicaciones que puedan correr en varios servidores simultáneamente y de manera coordinada.

- **Tolerancia a fallos:** Al igual que en el caso anterior, para una aplicación real hay que tener en cuenta la posibilidad de fallos en el servidor, ya sea por mala ejecución del código o por fallo de la máquina, en este caso hará que plantear desde el desarrollo de la aplicación como de la infraestructura que la soporte, medias que permitan la tolerancia a fallos.

- **Estabilidad de la conexión:** Si bien en un entorno controlado la conexión del dispositivo móvil al servidor es estable, en una aplicación real en la cual el acceso no será mediante wifi, sino a través de la red de telefonía, habría que plantear medidas para controlar la conexión de los equipos, tanto del lado del servidor como del lado cliente.

- **Mejora de los tiempos de respuesta y sincronización:** En una aplicación real, si se quiere una respuesta coordinada de todos los clientes conectados, habría que aplicar medidas de sincronización que permitan la compensación de la diferencia en los tiempos de respuesta que puede haber entre diferentes redes de telefonía, esto puede ser debido a variaciones en los niveles de cobertura, o incluso por el modo de conexión de diferentes dispositivos (4G, 3G).

Capítulo 6
Conclusions and future work

6.1 Conclusions

The tool developed in this End of Degree Project arises from a practical idea, something that could be a commercial project, despite this has been a great learning and has required research time. This work has allowed experimenting with what could be a real work, starting from a conceptual idea and carrying it out using different skills, languages and technologies.

This work has allowed to put in practice multiple skills acquired on degree studies, some examples are: Data bases, design of user interfaces, programming of interactive applications. In other cases, it has been necessary to search for information, combined with the acquired knowledgement, to get a solution, a example of this case is the use of Server Sent Events, a methodology which was necessary to investigate throughout this Final Degree Project.

While this has been an academic work and, by the hours of dedication, the available resources and the knowledge it requires, it is not possible to get the results of a commercial application. However, the foundations of a future commercial application have been laid.

6.2 Future work

Although a minimally functional application has been developed, there are several aspects that can be considered as future work lines:

- **Development to other platforms:** Development of the mobile application for other platforms: IOS and Windows.

- **Implement other modes of operation:** Increase the different types of events to not limit to “sound” or “colour”, these modes could be: random or raffle.

- **Securization of the application:** In this development has not been taken into account the application of security measures that would be applicable in a real project, some examples where it is necessary can be the following: user management, control of sessions, securization of the database and implementation of security measures in the database.

- **Optimization of traffic:** In a real application of these characteristics should be taken into account the management of thousands or tens of thousands simultaneous connections. So that, to manage this traffic is necessary study how much load is it for a server and how it could be optimized .

- **Load balancing:** Not only is necessary to optimize traffic, in real applications with a lot of concurrency, it is necessary to develop applications that can run on several servers simultaneously and in a coordinated manner.

- **Fault Tolerance:** As in the previous case, for a real application it is necessary to take into account the possibility of failures in the server, either due to bad execution of the code or machine failure. In this case it is necessary to take measures to increase fault tolerance, both in the code and in the infrastructure.

- **Stability of the connection:** Although in a controlled environment the connection of the mobile device to the server is stable, in a real application, in which the access will not be via wifi, but through the telephony network, it would be necessary to propose measures for control the connection of the equipment, both on the server side and client side.

- **Improved response times and synchronization:** In a real application, if you want a coordinated response from all connected clients, you should apply synchronization measures that allow the compensation of the different response times that may exist between different telephony networks, this may be due to variations in coverage levels, or even by the connection mode of different devices (4G, 3G).

Capítulo 7
Presupuesto

7.1 Presupuesto

Concepto	**Unidades**	**Precio**	**Coste**
Horas de desarrollo	300	35 €/h	**10.500 €**
Equipo de desarrollo	1	600 €	**600 €**
Total			**11.100 €**

Tabla 7.1: Presupuesto

Capítulo 8
Bibliografía

8.1 Bibliografía.

[1] Wikipedia contributors. “Serven-sent events”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 8 March 2019. Web. 10 May 2019.

[2] W3schools.com. “HTML5 Server-Sent Events”. W3schools.com. Web. 10 May 2019.

[3] Roach, Tyler. “Eventsource-Android”. Github.com. Github.com, 20 Nov 2017. Web. 11 May 2019.

[4] Wikipedia contributors. “Express.js”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 10 May 2019. Web. 13 May 2019.

[5] Expressjs.com. “ExpressJS”. Expressjs.com. Web. 13 May 2019.

[6] Wikipedia contributors. “AngularJS”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 April 2019. Web. 13 May 2019.

[7] Angularjs.org. “AngularJS”. Angularjs.com. Web. 13 May 2019.

[8] Wikipedia contributors. “MongoDB”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 17 April 2019. Web. 14 May 2019.

[9] Joren Six, Olmo Cornelis, Mark Leman. “TarsosDSP, A Real-Time Audio Processiong Framework in Java”. AES 53RD INTERNATIONAL CONFERENCE, London, UK. January 27–29 2014.

[10] Wikipedia contributors. “HTML5”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 8 May 2018. Web. 15 May 2019.

[11] W3schools.com. “HTML5”. W3schools.com. Web. 15 May 2019.