

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de
Telecomunicación

Desarrollo de Servicio WEB REST y Aplicación
Android para Eventos de Ocio

Autor: Pavel Blotski

Tutora: M^a Teresa Ariza Gómez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016



Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Desarrollo de Servicio WEB REST y Aplicación Android para Eventos de Ocio

Autor:
Pavel Blotski

Tutora:
M^a Teresa Ariza Gómez
Profesor titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Desarrollo de Servicio WEB REST y Aplicación Android para Eventos de Ocio

Autor: Pavel Blotski

Tutora: M^a Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El secretario del Tribunal

Fecha

A mi familia

A mis amigos

Resumen

Sin duda alguna, en el siglo XXI un dispositivo móvil y un ordenador se han convertido en las herramientas imprescindibles para el ser humano, ya que son utilizadas para obtener cualquier información deseada por medio de las numerosas aplicaciones instaladas en estos dispositivos.

Como consecuencia, cabe destacar el significativo crecimiento de estas tecnologías, lo que conlleva a apreciar su importancia.

En este proyecto se han desarrollado un servicio WEB y una aplicación para dispositivos Android que utiliza información de una base de datos gestionada por dicho servicio.

La aplicación informa al usuario de los eventos de ocio que organizan los diferentes tipos de locales de su ciudad. Además tiene la opción de visualizar la localización de dichos locales en un mapa junto con la localización actual del usuario. Otra herramienta presente es la opción de poder compartir el contenido en las redes sociales que resulta ser muy importante para expandir la información acerca de la aplicación.

Abstract

Undoubtedly, in the XXI century a mobile device and a computer have become in the essential tools for the human, as they are used to get any desired information through numerous applications installed in these devices.

As a result of this, include the significant growth of these technologies, which leads to appreciate its importance.

In this project have developed a WEB service and an application for Android devices that use information from a database managed by the service.

The application informs the user of entertainment events organized by different types of locals in your city. Also you have the option to see these locals in a map together with the actual location of the user. Another present tool is the option to share content through social networks that turns out to be very important to expand the information about the application.

Glosario

BASE DE DATOS - bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

PRIMARY KEY – en el diseño de las bases de datos relacionales, así se le llama a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla.

FOREIGN KEY – en el diseño de las bases de datos relacionales, es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla que se refiere a una columna o grupo de columnas en otra tabla.

ANDROID – es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets y automóviles.

SERVICIO WEB – es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

CLIENTE – SERVIDOR – es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a servidor, quien le da la respuesta.

URI (Uniform Resource Identifier) – es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

URL (Uniform Resource Locator) – es un identificador de recursos uniforme (URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo.

POST, GET, PUT, DELETE – son cuatro métodos de peticiones HTTP.

API (Application Programming Interface) – es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

HTTP (Hypertext Transfer Protocol) – es el protocolo de comunicación que permite las transferencias de información en la World Wide Web (www).

REST (Representational State Transfer) – arquitectura de servicio web que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etc.) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

HTML (Hypertext Markup Language) – hace referencia al lenguaje de marcado para la elaboración de páginas web.

XML (eXtensible Markup Language) – es un lenguaje de marcas desarrollado por el World Wide Web Consortium utilizado para almacenar datos en forma legible.

SERVLET – es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

UML (Unified Modeling Language) – es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, construir y documentar un sistema.

FRAMEWORK - define un conjunto estandarizado de conceptos, prácticas y criterios para enfocar

un tipo de problema particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

JAVA – es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

INTERNAL ERROR 500 - el servidor web encuentra una condición inesperada que le impide completar la solicitud del cliente para acceder a la URL requerida.

ERROR 404 – Intento de acceder a una URL que no existe en el servidor.

CARTRIDGE – en el contexto del proyecto se refiere a las partes que se pueden agregar a una aplicación creada en el Openshift.

WIDGET- es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor.

SLIDE DE IMÁGENES – rotación de una serie de imágenes en un elemento definido.

Índice

Resumen	viii
Abstract	ix
Glosario	x
Índice	xiii
Índice de Figuras	xv
1 Introducción	19
1.1 Motivación	19
1.2 Escenario completo	20
1.3 Escenario del proyecto	21
1.4 Objetivos	22
1.4.1 Base de datos	22
1.4.2 Servicio WEB REST	22
1.4.3 Aplicación Android	23
1.4.4 Funcionalidad conjunta	24
1.5 Fases de realización	25
1.6 Estructura del documento	26
2 Tecnologías y Entornos Utilizados	29
2.1 Mac OS X El Capitan	29
2.2 MySql	29
2.3 XAMPP	30
2.4 phpMyAdmin	31
2.5 MySqlWorkBench	31
2.6 Spring	32
2.7 Spring Tool Suite	33
2.8 REST	33
2.9 JSON	34
2.10 .WAR	34
2.11 GIT	35
2.12 Openshift	36
2.13 Android Studio	36
2.14 Google Maps	37
2.15 Dispositivo móvil Samsung Galaxy S4 mini	38
2.16 Genymotion	38

3 Base de Datos	41
3.1 Elección de la Base de Datos	41
3.2 Estructura	41
3.3 Tablas, variables y relaciones	43
3.4 Utilización en el proyecto	44
4 Servicio WEB REST	46
4.1 Concepto	46
4.2 Cumplimiento de los principios REST	46
4.3 Tipo de proyecto elegido para el desarrollo.	46
4.4 Diagrama de paquetes con sus clases	49
4.5 Librerías importadas en en trabajo	50
4.6 Funcionamiento	50
4.7 Exportación del servicio	56
4.8 Subida a la nube utilizando el servicio de Openshift	57
5 Aplicación Android	62
5.1 Introducción	62
5.2 Diagrama de casos de uso	62
5.3 Diagramas de secuencia	63
5.4 Funcionamiento	69
5.5 APIs utilizadas en el proyecto	78
6 Pruebas Realizadas	80
6.1 Pruebas de la Base de Datos	80
6.2 Pruebas del Servicio WEB	80
6.3 Pruebas de la Aplicación Android	81
7 Problemas Encontrados	85
8 Conclusiones	88
9 De Cara al Futuro	90
10 Bibliografía	92
Manual de Usuario	94
Manual de Instalación	103

ÍNDICE DE FIGURAS

Figura 1. Escenario completo	20
Figura 2. Escenario del proyecto	21
Figura 3. Logo de Mac OS X Capitan	29
Figura 4. Logo de MySQL	29
Figura 5. Logo de XAMPP	30
Figura 6. Logo de phpMyAdmin	31
Figura 7. Logo de Workbench	31
Figura 8. Logo de Spring	32
Figura 9. Logo de Spring Tool Suite	33
Figura 10. Logo de JSON	34
Figura 11. Logo de GIT	35
Figura 12. Logo de Openshift	36
Figura 13. Logo de Android Studio	36
Figura 14. Logo de Google Maps	37
Figura 15. Logo de Samsung Galaxy S4 mini	38
Figura 16. Logo de Genymotion	38
Figura 17. Estructura de la base de datos	42
Figura 18. Consulta a la tabla <i>Locals</i>	44
Figura 19. Consulta a la tabla <i>Events</i>	45
Figura 20. Subdirectorio Java Resources/src	47
Figura 21. Directorio WebContent	47
Figura 22. Dependencia Openshift	48
Figura 23. Diagrama de paquetes y sus clases	49
Figura 24. La ruta de los locales y la función para obtenerlos	51
Figura 25. Función <i>GetAllLocals</i> en la clase <i>ProjectMoanagerLocals</i>	52
Figura 26. Método de la clase <i>Database</i>	52
Figura 27. Método <i>GetAllLocals</i> en la clase <i>ProjectLocals</i>	53
Figura 28. Clase <i>Local</i>	54
Figura 29. Servlet en el archivo web.xml	55
Figura 30. Diagrama de funcionamiento del servidor	55

Figura 31. Exportación del servicio	56
Figura 32. Parámetros de la exportación de un servicio	57
Figura 33. Selección de OPENSIFT WEB CONSOLE	58
Figura 34. Pantalla principal de OPENSIFT WEB CONSOLE	58
Figura 35. Aplicación creada en Openshift	60
Figura 36. Comando para clonar el directorio de Openshift a un directorio local	60
Figura 37. Comandos git para subir el archivo .war a la nube	61
Figura 38. Diagrama de casos de uso	62
Figura 39. Consulta de los locales de un tipo	63
Figura 40. Consulta de la información acerca de un local	64
Figura 41. Consulta de los eventos de un local	65
Figura 42. Consulta de la información de un evento	66
Figura 43. Visualización de los locales en el mapa	67
Figura 44. Consulta de la información de contacto	68
Figura 45. Compartir el contenido en Facebook	69
Figura 46. Pantalla principal de la aplicación	70
Figura 47. Fragmento “Locales”	71
Figura 48. Fragmento “Mapa”	72
Figura 49. Información de un local	73
Figura 50. Los eventos programados de un local	74
Figura 51. Información de un evento	74
Figura 52. Compartición del contenido en el perfil de Facebook	75
Figura 53. Respuesta de Facebook a la aplicación	75
Figura 54. Inicio de la clase <i>HTTPDataHabdler</i> y utilización de uno de sus métodos con el parámetro la URL que se quiere consultar	76
Figura 55. Clase <i>HTTPDataHandler</i> (la parte de try)	76
Figura 56. Partición de la cadena recibida en objetos JSON	77
Figura 57. Inicio de la aplicación	94
Figura 58. Pantalla principal	95
Figura 59. Confirmación de “Me gusta”	96
Figura 60. Información de contacto	96
Figura 61. Discotecas disponible a consultar	97
Figura 62. Localización de las discotecas	98
Figura 63. Información de un local	99
Figura 64. Información adicional de un local	100
Figura 65. Eventos programados de un local	100

Figura 66. Información de un evento	101
Figura 67. Información adicional de un evento	102
Figura 68. Elección del instalador de XAMPP	103
Figura 69. Contenido del archivo .dmg de XAMPP	104
Figura 70. Pantalla 1 del instalador de XAMPP	104
Figura 71. Pantalla 2 del instalador de XAMPP	105
Figura 72. Pantalla 3 del instalador de XAMPP	105
Figura 73. Página web de Spring Tool Suite	106
Figura 74. Carpeta de Spring Tool Suite	106
Figura 75. Selección del espacio de trabajo de Spring Tool Suite	107
Figura 76. Pantalla 1 de la descarga de Android Studio	108
Figura 77. Pantalla 2 de la descarga de Android Studio	108
Figura 78. Instalador de Android Studio	109
Figura 79. Pantalla 1 de la página web de Genymotion	110
Figura 80. Pantalla 2 de la página web de Genymotion	111
Figura 81. Pantalla 3 de la página web de Genymotion	111
Figura 82. Pantalla 4 de la página web de Genymotion	112
Figura 83. Instalador de Genymotion	112

1 INTRODUCCIÓN

1.1 Motivación

Es bastante curioso echar la vista atrás, aunque sea por solo una década de años, para recordar los dispositivos móviles y los ordenadores que se utilizaban. Para algunos eran sus primeros terminales. Sin embargo, hoy en día un teléfono móvil o un portátil lo tiene cualquiera y lo utiliza a menudo para obtener la información necesaria. Además el nivel de los terminales hoy en día es mucho más elevado que hace solamente diez años. Este avance de la tecnología ha ocurrido en un periodo de tiempo muy corto si comparamos con los años que lleva el ser humano viviendo en la Tierra.

En consecuencia, el número de los desarrolladores de aplicaciones para los móviles o los ordenadores ha crecido de forma exponencial ya que cada día se desarrollan numerosos proyectos en distintas organizaciones para competir y superar a sus principales rivales.

Ya que existe tanto interés hoy en día por estas tecnologías llama muchísimo la atención y mediante la amplia variedad de aplicaciones existentes hay una cierta facilidad para aprender a manejar estas tecnologías.

De ahí es la razón del interés en este proyecto. Aprender a desarrollar un servicio WEB, que hace consultas a una base de datos, y utilizarlo con un dispositivo móvil Android. Pero claro, no puede ser una aplicación Android cualquiera ya que en este corto periodo de tiempo se han desarrollado tantas aplicaciones que apenas hay ideas para que una nueva sea un éxito.

La idea de la aplicación de este proyecto es básicamente informar al usuario acerca de los eventos de ocio programados para fechas próximas en su ciudad, pero tiene que ser algo muy simple, para que se vea con la mayor claridad posible y que tenga una interfaz agradable. Es lo que busca el usuario hoy en día en una aplicación que le proporciona la información necesaria.

Puede parecer una idea muy sencilla, pero actualmente existen muchas ciudades que no disponen de dicho servicio y que sería bastante útil.

1.2 Escenario completo

El desarrollo de este servicio no solo incluye la aplicación Android, el servidor y la base de datos, sino que hay otra parte que no abarca este proyecto, que corresponde al gestor del contenido de la aplicación y su correspondiente servicio web.

En el caso de que un desarrollador tenga una aplicación como esta no se puede encargar de gestionar miles de eventos programados para cada local. Por esta razón tiene que haber una aplicación que sea un gestor para introducir, modificar o borrar la información de la base de datos sobre el local determinado. Ese gestor se proporcionaría a la empresa o el local contratante del servicio por el se realizan las agregaciones o modificaciones necesarias por alguna persona contratada de dicha empresa o local.

De esta manera el escenario del servicio completo quedaría de la forma que podemos ver en la figura 1.1 donde hay presentes la base de datos, dos servidores WEB que proporcionan el servicio al gestor y a la aplicación Android.

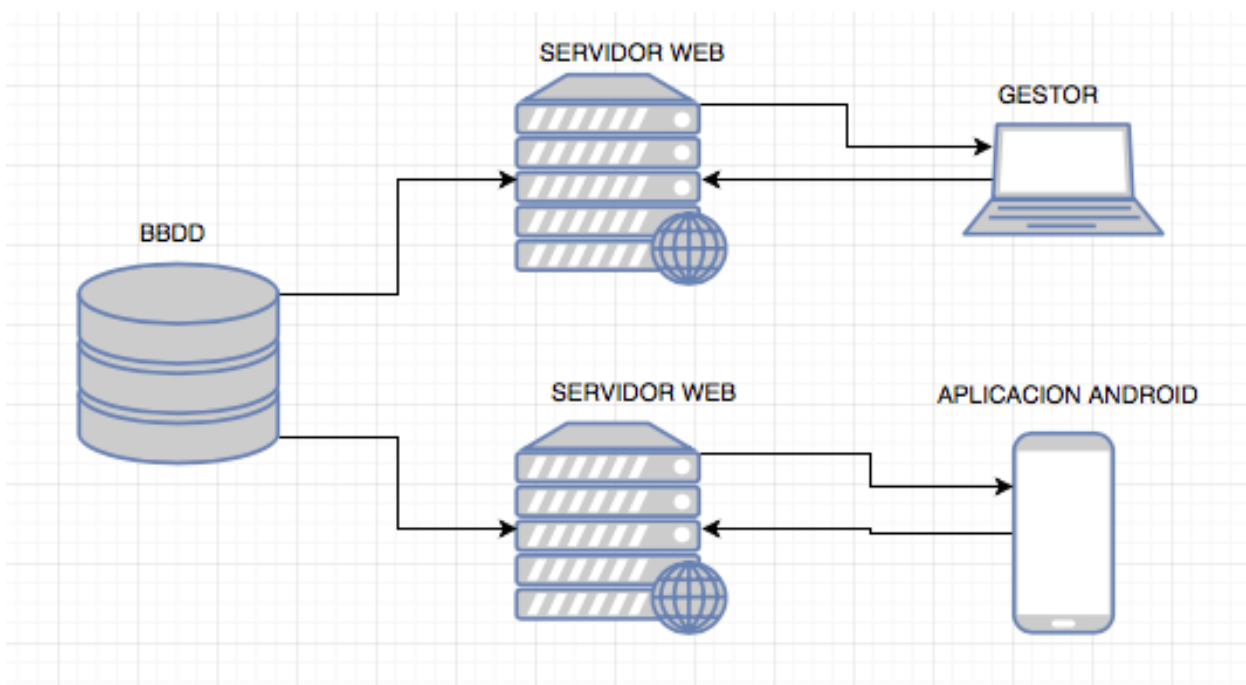


Figura 1. Escenario completo

1.3 Escenario del proyecto

En el apartado anterior lo que se ha visto es el diseño completo del servicio que se ha desarrollado. Ahora el enfoque se va a centrar a la parte que corresponde a este proyecto.

Las partes presentes son las siguientes:

- Está la base de datos a la cual le llegan las consultas desde el servicio WEB. Posteriormente, dicha base de datos realiza una respuesta hacia el servicio.
- El servidor que hace dichas consultas y proporciona la información para la aplicación Android.
- La aplicación Android que utiliza los datos proporcionados por el servidor

En la figura 1.2 se ven claramente dichas partes con las acciones correspondientes realizadas por cada una.

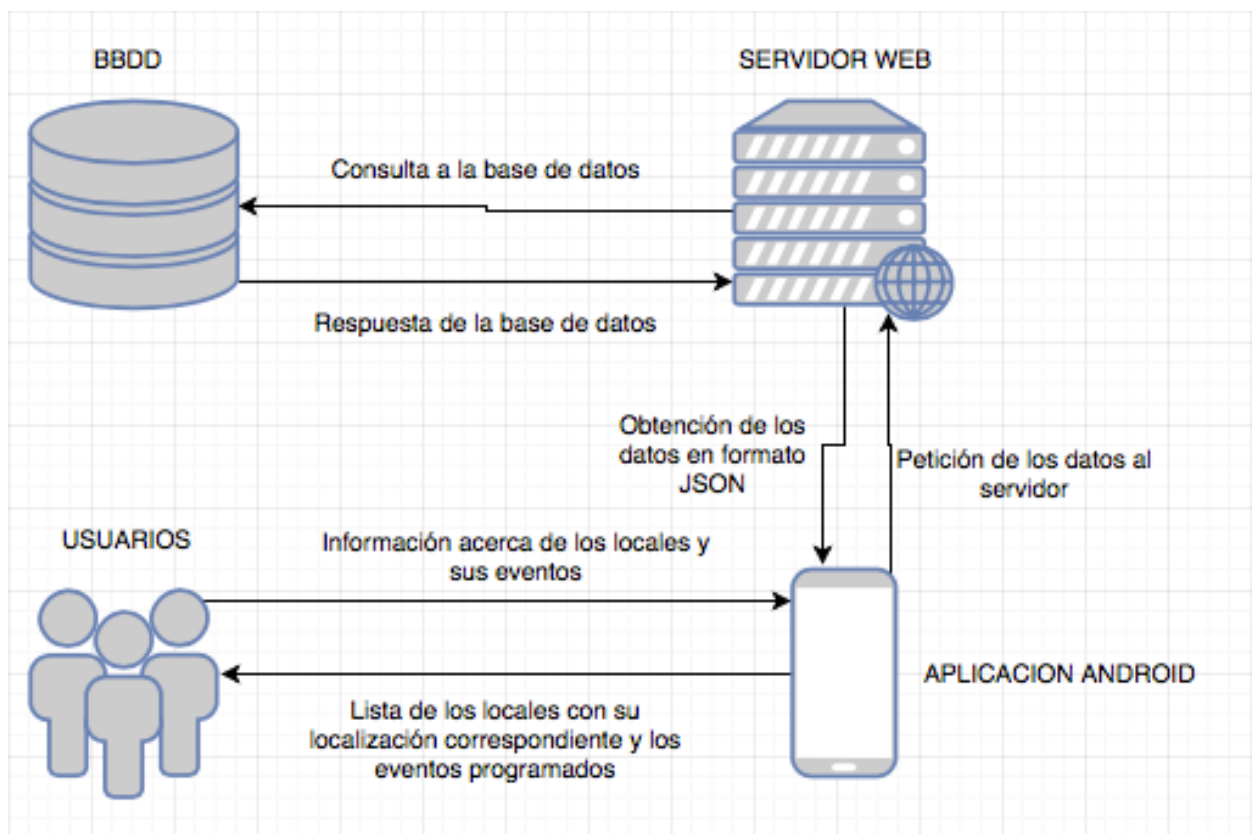


Figura 2. Escenario del proyecto

1.4 Objetivos

El objetivo principal de este trabajo es crear una herramienta por la que el usuario con bastante facilidad pueda saber acerca de los eventos que hay programados en distintos locales en la ciudad que se encuentra. Además se proporcionará un mapa al usuario para ver la localización de dichos locales con el motivo de proporcionar facilidad a la hora de encontrar un local con el evento en el que se ha interesado una persona. También el usuario va a poder consultar la información para contactar con los desarrolladores y va a poder compartir la información de la aplicación en las redes sociales.

Como se ha mencionado anteriormente, en este proyecto podemos destacar tres partes bien definidas, las cuales van a ser descritas detalladamente en sus respectivos capítulos. Hay que dejar claro qué es lo que realmente se quiere conseguir con ellas.

1.4.1 Base de datos

La base de datos que se va a desarrollar tiene que ser compatible para todas las partes del proyecto que hacen usos de los datos almacenados en la misma, por lo que va a ser la misma tanto para el gestor como para la aplicación.

Lo que se quiere conseguir es diseñar una arquitectura lógica para los datos que se van a almacenar.

Los contenidos de las tablas serán los siguientes:

- Información de las personas que se encargan de la gestión de los contenidos.
- Información específica de los locales.
- Descripciones de los eventos programados.
-

1.4.2 Servicio WEB REST

El servicio WEB REST que se ha desarrollado en Java servirá únicamente para este proyecto. Solo la aplicación Android va a poder utilizarlo.

El servidor desarrollado va a ser subido a la nube ya que es una tecnología bastante interesante de cara al futuro y que casi todas las empresas la utilizan, entonces, conviene aprender todo acerca de la misma y , por otro lado, ahorra tener que arrancar múltiples programas en el ordenador donde se va

a desarrollar el proyecto.

El objetivo de dicho servicio es acceder a la base de datos consultando la información pedida por la aplicación Android. La respuesta obtenida de la base de datos tendrá lugar en la URL proporcionada por la nube. Entonces, la aplicación Android podrá acceder a dicha URL y recoger los datos pedidos anteriormente.

1.4.3 Aplicación Android

¿Qué cosas son las que hay que tener en cuenta para que la aplicación sea utilizada por personas? Nadie va a ser forzado a descargarla y utilizarla, por lo que tiene que ser útil, clara y única. Estos son los factores más importante a la hora de desarrollar una aplicación de cara al cliente. A continuación un poco acerca de cada uno:

- **Utilidad de la aplicación**

Pues, el primer factor en el que nos vamos a enfocar es la utilidad. ¿Cuántas veces una persona se pregunta a donde podría ir a pasar su tiempo libre o aprender algo nuevo teniendo en cuenta sus intereses? Pues, muchas y al final lo que uno acaba haciendo es algo que ya haya hecho y le haya gustado, pero porque desconoce las opciones que tiene en la zona cercana a su vivienda.

La aplicación va a ofrecer poder visualizar múltiples eventos disponibles en la ciudad del cliente. Por lo que tiene una gran variedad donde elegir según el interés que muestre por algo en concreto. Entre las opciones que va a ofrecer la aplicación se va a poder encontrar locales de discotecas, cines, conciertos, eventos especiales y de educación. Entonces el cliente se iría a la parte que le interesa y buscaría el evento del tipo que le gusta más. Este es el factor más importante ya que de él depende mucho el número de los usuarios que van a utilizar la aplicación.

- **Claridad de la aplicación**

El segundo factor importante destacado anteriormente es la claridad. El mayor problema que se ve en muchos casos es que las aplicaciones están sobrecargadas de información y aunque uno crea que proporcionar más información sea lo mejor, pues, no lo es, porque conlleva a que el cliente deje de utilizar dichas aplicaciones.

Lo que se pretende conseguir es minimizar los contenidos de la información para que se vea muy claro que es lo que se está ofreciendo. Enfocar al usuario solamente en las cinco opciones existentes descritas en el factor anterior ya que si hay mucha información en la pantalla el cliente tendría que pasar un tiempo leyendo todo lo que se ofrece y entrar en cada apartado lo que al fin y al cabo le acaba aburriendo ya que no encuentra lo que está buscando por lo que busca otra aplicación donde hay mayor facilidad para encontrar lo que busca.

- **Unicidad de la aplicación**

El último factor destacado es que la aplicación sea única. Esto es algo complicado de conseguir hoy en día ya que casi todo está inventado y sacar algo con éxito es muy complicado. Tiene que haber algo único en ella, algo que la haga especial, ya sea el diseño o funcionalidad. Esto es lo que más importa de cara al cliente ya que él va a decidir si tener la aplicación o no.

Lo que se quiere conseguir es una representación agradable de la información y de una forma muy clara, combinando las distintas acciones que se ofrecen para hacer la aplicación única.

- **Expansión de la aplicación**

Aunque no está incluida en los factores principales, pero es algo muy importante, más para la parte de desarrollador ya que le va a generar más beneficios, pero también es muy útil para el lado del cliente, porque si la aplicación es realmente útil, ¿por qué no compartirlo con tus amigos mediante redes sociales? Así más personas podrán disfrutar de las facilidades que se ofrecen.

Lo que se pretende conseguir es diseñar las opciones de poder compartir la información de la aplicación en las redes sociales más utilizadas hoy en día. Eso ayudaría bastante para expandir la información acerca de su existencia lo que conllevaría al aumento de los usuarios de la misma.

1.4.4 Funcionalidad conjunta

Es algo complicado que cada una de esas partes funcione bien con las otras y se obtenga el resultado esperado. Esa es la mayor complejidad de este proyecto.

Como conclusión de todos los apartados de objetivos, lo que se pretende conseguir es el funcionamiento síncrono entre todas las partes implicadas reduciendo el tiempo de la espera para el

usuario.

1.5 Fases de realización

En este apartado se explican las fases seguidas a lo largo del desarrollo de este proyecto:

- **Idea:** Formación de unos objetivos que dan como fruto una idea acerca de una herramienta útil, fácil de manejar y única en forma de aplicación Android.
- **Estudio:** Investigación sobre las herramientas necesarias para llevar a cabo los objetivos anteriores. Pruebas sobre aplicaciones con el funcionamiento similar con el objetivo de obtener ciertas conclusiones de las mecánicas que podrían ser útiles a la hora de encontrar una manera de definir el servicio proporcionado al cliente.
- **Análisis:** Definición de requisitos con el fin de aclarar qué es lo que se pretende conseguir. Hasta qué punto se va a llegar en el proyecto.
- **Diseño:** Definición de la arquitectura de la aplicación mediante diversos diagramas, esquemas y dibujos, proporcionando facilidades para el posterior desarrollo.
- **Implementación:** Esta fase se puede dividir en 3 partes:
 - Creación de la base de datos MySQL diseñada.
 - Programación del servidor WEB REST.
 - Desarrollo de la aplicación móvil para dispositivos Android.
- **Despliegue:** Exportación del servicio WEB REST y su posterior subida a la nube.
- **Pruebas:** Una vez finalizada la implementación de cada una de las partes se comprueba

el funcionamiento conjunto para ver que se cumplen todos los objetivos propuestos proporcionando calidad y efectividad.

- **Documentación:** Descripción por escrito sobre todo lo realizado durante la realización del proyecto.

- **Mantenimiento:** En este proyecto no tiene gran importancia, ya que no se exige el funcionamiento continuo después de la presentación del mismo, pero de cara al futuro sería algo indispensable para realizar cualquier mejora o ampliación de dicho proyecto.

1.6 Estructura del documento

En este apartado se van a describir brevemente las partes que presenta este documento, que la mayoría de ellas coinciden con las fases descritas en el apartado anterior:

En el inicio del documento lo que podemos encontrar es un breve resumen, el glosario y los índices del contenido y de las figuras.

I. Introducción:

Se realiza la introducción dejando claro cual es la motivación de este proyecto, presentando los escenarios completo y el de proyecto, objetivos que se pretende cumplir, las fases de realización que se han seguido y, por último, una breve descripción de todos los capítulos del documento.

II. Tecnologías Utilizadas:

En este capítulo se van a describir las herramientas utilizadas a lo largo del desarrollo del proyecto. No se va a explicar cómo se han utilizado, sino que se va a proporcionar una breve información de lo que es cada una y su funcionalidad general.

III. Base de Datos:

En esta parte de la memoria se hablará de la base de datos desarrollada para este proyecto. Se explicará el tipo del sistema de gestión utilizado, se mostrarán las tablas con sus variables correspondientes, se hablará de cada tabla y sus variables y, para terminar, se mostrará de que forma se realizan las consultas a esta base de datos desde el servidor.

IV. Servicio WEB REST:

En esta parte del documento se hablará de lo que es un servicio web, se demostrará que se cumplen todos los principios REST, se explicará el tipo de proyecto que ha sido escogido para este proyecto y sus componentes. A continuación se mostrará el diagrama de paquetes del servicio, se hablará de las librerías que han sido necesarias para el desarrollo y, para terminar, se explicará detalladamente el funcionamiento del servidor.

V. Aplicación Android:

En este capítulo se proporcionará la información acerca de la aplicación Android desarrollada, que es la parte más significativa de este proyecto. Se explicarán las librerías utilizadas, se mostrarán los diagramas de uso y de secuencia y se verá detalladamente el funcionamiento de la aplicación.

VI. Pruebas Realizadas

En esta parte se van a mencionar todas las pruebas que se han realizado con las tres partes del proyecto a lo largo de todo el desarrollo de las mismas.

VII. Problemas Encontrados

En este capítulo se explicarán los problemas más destacados que se han tenido a lo largo del desarrollo del trabajo y cómo se han solucionado o cuál ha sido el camino alternativo para resolver estos problemas.

VIII. Conclusiones

En esta parte del trabajo se realizará la evaluación de los resultados conseguidos y se hablará de las experiencias obtenidas durante el desarrollo del proyecto.

IX. De Cara al Futuro

En este capítulo de la memoria se hablará de las agregaciones que harían falta para completar el desarrollo de este proyecto y su posible subida a Playstore para que sea disponible a todos los usuarios.

X. Bibliografía

Se pondrán los enlaces de las páginas web consultadas durante el desarrollo de este trabajo.

XI. Manual de usuario

En esta parte de la memoria del proyecto se muestra el manual para el usuario de la aplicación Android explicando todas sus funcionalidades.

XII. Manual de instalación

En este capítulo se explicará cómo instalar las herramientas necesarias para el desarrollo de este trabajo.

2 TECNOLOGÍAS Y ENTORNOS UTILIZADOS

2.1 Mac OS X El Capitan



Figura 3. Logo de Mac OS X Capitan

OS X es un entorno operativo basado en Linux y vendido por Apple Inc.

Mac OS X El Capitan es la versión 10.11 de OS X. Esta versión del sistema operativo es la que ha sido utilizado para el desarrollo de este trabajo. Mantiene la interfaz de su predecesor OS X Yosemite, centrándose en mejorar la experiencia del usuario y el rendimiento del sistema operativo, con la introducción de “Metal for Mac”, característica del sistema que ya existía para iOS y que hace que la unidad de procesamiento gráfico trabaje a un ritmo más veloz. Por ello las aplicaciones se ejecutan más rápido en esta versión.

2.2 MySql



Figura 4. Logo de MySQL

MySQL es un sistema de gestión de bases de datos relacional muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características disponibles en otros sistemas de gestión, es una buena opción tanto para aplicaciones comerciales, como para entretenimiento por su facilidad de uso y tiempo reducido de su puesta en marcha.

MySQL está disponible para múltiples plataformas.

Este sistema de gestión se ha utilizado en el proyecto para gestionar la base de datos donde se consultan los locales y los eventos.

2.3 XAMPP



Figura 5. Logo de XAMPP

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene distintos módulos. XAMPP es un servidor web que se puede instalar de forma local en el sistema operativo utilizado, y con ello implementar un entorno de desarrollo para realizar las pruebas.

En el caso de este trabajo el módulo que ha sido utilizado es phpMyAdmin para gestionar la base de datos desde el navegador donde se muestra este módulo.

2.4 phpMyAdmin



Figura 6. Logo de phpMyAdmin

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de las páginas web. Puede crear y eliminar bases de datos, crear, alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar privilegios y exportar datos en varios formatos.

Con esta herramienta se ha administrado la base de datos de este proyecto.

2.5 MySQLWorkBench



Figura 7. Logo de Workbench

MySQLWorkbench es una herramienta visual de diseño de las bases de datos que integra desarrollo de software, administración, creación y mantenimiento de dichas bases de datos. Es el sucesor de DBDesigner4.

Esta herramienta es la que se ha utilizado para crear toda la estructura de la base de datos del proyecto.

2.6 Spring



Figura 8. Logo de Spring

Spring es un framework para el desarrollo de las aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

La tecnología Spring es la que ha sido utilizada para el desarrollo del servicio en este trabajo. Las razones de uso de este tipo son las siguientes:

- Framework especializado para el desarrollo de aplicaciones para plataforma Java.
- Código abierto.
- Comprende diversos módulos que proporcionan múltiples servicios.
- Proporciona un contenedor que se encarga de gestionar los ciclos de vida de los objetos.

2.7 Spring Tool Suite



Figura 9. Logo de Spring Tool Suite

Spring Tool Suite es un entorno de desarrollo basado en Eclipse que es especializado para el desarrollo de las aplicaciones Spring. Proporciona un entorno preparado para implementar, depurar, ejecutar y desplegar las aplicaciones Spring, incluyendo las integraciones para Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, etc.

Mediante este entorno se ha desarrollado el servidor de este trabajo.

2.8 REST

Ésta es la tecnología utilizada para el desarrollo del servidor del proyecto.

REST afirma que la web ha disfrutado de escalabilidad como resultado de un aserie de diseños fundamentales claves:

- Un protocolo cliente/servidor sin estado: cada mensaje http contiene toda la información necesaria para comprender la petición.
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: http define un conjunto de operaciones, las más importantes son POST, GET, PUT y DELETE.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es

accedido únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML.

2.9 JSON



Figura 10. Logo de JSON

JSON es un formato bastante ligero empleado para el intercambio de datos, siendo un subconjunto de la notación para objetos empleada en JavaScript. Está considerado como un lenguaje independiente de formato de datos cuya especificación está descrita en RFC4627. La sencillez de este formato le ha dado ventaja, permitiendo una gran difusión de la tecnología como alternativa a XML.

2.10 .WAR

Un archivo WAR es un JAR utilizado para distribuir una colección de páginas JavaServer, Servlets, clases Java, archivos XML, librerías de etiquetas y páginas web estáticas que juntos constituyen una aplicación web.

En este trabajo se utiliza este formato para exportar el servidor desde Spring Tool Suite y subirlo a la nube.

2.11 GIT



Figura 11. Logo de GIT

Git es un software de control de versiones diseñado por Linus Torvalds, pensado en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente

El mantenimiento del software Git está actualmente supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

En este proyecto se van a utilizar sus comandos para subir el archivo .war a la nube.

2.12 Openshift



Figura 12. Logo de Openshift

Openshift es un producto de computación en la nube. Es un servicio de código abierto con el nombre “Openishft Origin”, y está disponible en GitHub.

Los desarrolladores pueden usar GIT para desplegar sus aplicaciones Web en los diferentes lenguajes de plataforma.

Openshift se encarga de mantener los servicios subyacentes a la aplicación y la escalabilidad de la aplicación como se necesite.

En este proyecto se ha utilizado su servicio para subir el servidor desarrollado a la nube.

2.13 Android Studio



Figura 13. Logo de Android Studio

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013. Reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia de Apache 2.0. Está disponible para muchas plataformas.

En el trabajo ha sido utilizado como el entorno para el desarrollo de la parte de la aplicación para el dispositivo móvil.

2.14 Google Maps



Figura 14. Logo de Google Maps

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle con Google Street View.

En este proyecto se utiliza para mostrar la localización de los locales.

2.15 Dispositivo móvil Samsung Galaxy S4 mini



Figura 15. Logo de Samsung Galaxy S4 mini

Samsung Galaxy S4 mini es un teléfono inteligente de gama media fabricado y desarrollado por Samsung. Utiliza el sistema operativo Android 4.2.2 Jelly Bean, el cual es actualizable a Android 4.4.2 Kitkat. El dispositivo fue anunciado el 30 de mayo de 2013 mediante la página web de Samsung y se presentó formalmente con todas sus características el 20 de junio del mismo año en un evento que la compañía realizó en Londres. El lanzamiento del teléfono se produjo el 1 de julio de 2013.

Este teléfono móvil es el que ha sido utilizado para comprobar el funcionamiento completo del trabajo realizado.

2.16 Genymotion



Figura 16. Logo de Genymotion

Genymotion es un emulador de Android que aprovecha la arquitectura x86 para ejecutar de forma fluida y rápida distintos dispositivos Android. Olvidando la lentitud del emulador nativo de Android se puede ejecutar todo tipo de aplicaciones y juegos en las plataformas como Windows, Mac o Linux.

Genymotion ha conseguido crear una interfaz simple capaz de soportar distintas funcionalidades

accesibles a cualquier usuario.

Este software es el que ha sido utilizado para realizar las pruebas con la aplicación Android antes de utilizar el dispositivo móvil Samsung Galaxy s4 mini.

3 BASE DE DATOS

3.1 Elección de la Base de Datos

El sistema de gestión de la base de datos utilizado en este trabajo es MySQL. ¿Por qué utilizamos este tipo de tecnología y no otro? A continuación se presentan las características claves que explican dicha elección:

- Es muy utilizada en aplicaciones web.
- Es ideal para rápida lectura de datos, que es lo que necesita una aplicación web .
- Soporta gran cantidad de datos.
- Proporciona un amplio conjunto del lenguaje SQL.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Es un sistema de fuente abierta. Cualquier persona puede utilizar el código fuente de MySQL sin pagar.
- Es la más famosa.

3.2 Estructura

Pensar sobre los elementos que debe contener una base de datos puede parecer algo muy simple a primera vista, pero a la hora de realizar una estructura completa de la misma no es una tarea sencilla, ya que se necesita saber detalladamente todo lo que es necesario a la hora de poner en funcionamiento un proyecto. Por eso, es muy importante identificar todas las tablas y relaciones previamente para no volver después a modificar la base de datos. Esto ahorraría mucho tiempo de trabajo.

A continuación se presenta en la figura 17 la estructura de la base de datos realizada para este trabajo mediante el programa MySQLWorkBrench mencionado en el capítulo de las tecnologías utilizadas:

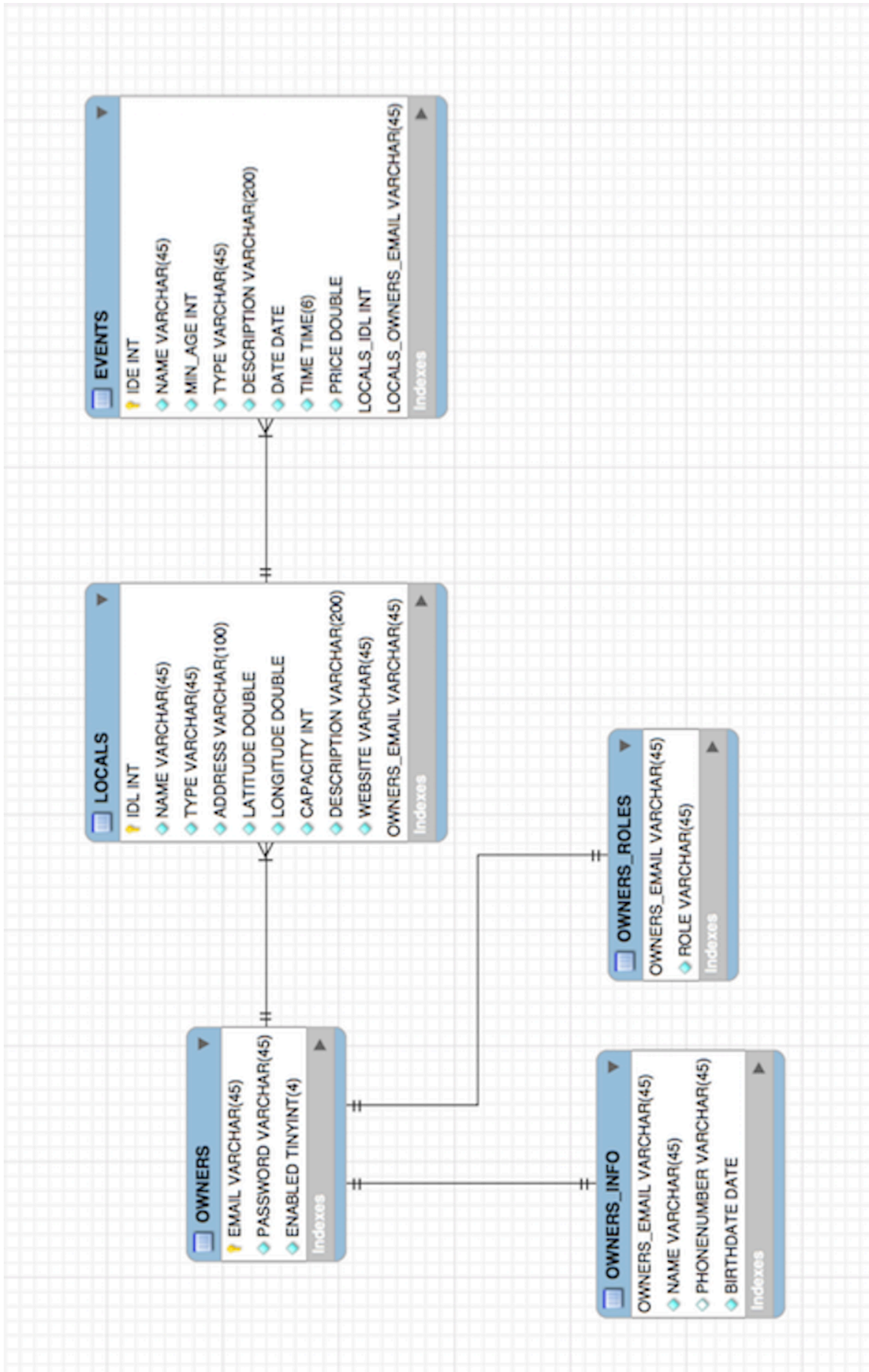


Figura 17. Estructura de la base de datos

3.3 Tablas, variables y relaciones

En este apartado se va a explicar la figura de la estructura de la base de datos mostrada en el apartado anterior. Se hablará de las partes más importantes que son las tablas que contiene, las variables de las mismas y las relaciones existentes.

- Tablas:
 - *Locals*: tabla donde se puede consultar la información principal acerca de los locales ingresados.
 - *Events*: tabla que va a ser utilizada para consultar la información de los eventos existentes de los distintos tipos de locales.
 - *Owners*, *Owners_info*, *Owners_roles*: tablas donde se va a registrar la información principal de los usuarios del gestor. A esas tablas no se va a realizar ningún acceso en este proyecto. Va a ser útil para los agentes que usen el sistema para ingresar la información a la base de datos.
- Campos: los tipos de atributos que se han utilizado en las tablas son INT, DOUBLE, DATE, TIME y VARCHAR. Algunas de ellas son de tipo PRIMARY KEY y FOREIGN KEY, de lo que se va a hablar en el apartado posterior.
- Relaciones: en este proyecto se van a utilizar dos de las tablas existentes, *Locals* y *Events*, y las relaciones que tienen son las siguientes:
 - Tabla *Locals*: campo IDL de tipo INT es la clave primaria de esta tabla y la conecta con la de eventos, ya que un local puede tener varios eventos programados y tienen que ser identificados por un valor del atributo único a la hora de hacer una consulta. El campo OWNERS_EMAIL de tipo VARCHAR de esta tabla es de tipo FOREIGN KEY y la relaciona con la de *Owners* donde ese mismo campo es PRIMARY KEY, ya que un dueño puede tener varios locales.

- Tabla *Events*: campo IDE de tipo INT es PRIMARY KEY para identificar cada evento programado. IDE no relaciona la tabla *Events* con ninguna otra en este proyecto, pero en un futuro cabe una ampliación de la base de datos donde se incluiría el uso de dicho campo. Esta tabla tiene dos parámetros que son FOREIGN KEY y la relacionan con las dos tablas mencionadas anteriormente, *Locals* y *Owners*. LOCALS_IDL de tipo INT se utiliza para identificar a qué local pertenece un evento y LOCALS_OWNERS_EMAIL de tipo VARCHAR se usa para saber quien los ha programado.

3.4 Utilización en el proyecto

Como se ha mencionado en el apartado anterior, en este proyecto se harán consultas a las dos tablas, *Locals* y *Events*, ya que estas dos son las necesarias para proporcionar la información requerida por el cliente mediante la aplicación Android. Las consultas a la base de datos se realizarán mediante el servicio web. A continuación se muestra la parte del código Java del servidor donde se realizan las peticiones SQL.

Tabla *Locals*: la consulta a dicha tabla mostrada en la figura 18 tiene como variable el tipo de los locales por el cual se van a filtrar y, posteriormente, serán mostrados los elementos seleccionados en la aplicación Android.

```
PreparedStatement ps = connection.prepareStatement("SELECT IDL,NAME,TYPE,"  
    + "LONGITUDE,LATITUDE,ADDRESS,DESCRIPTION,CAPACITY,WEBSITE"  
    + " FROM LOCALS WHERE TYPE='"+type+"'");  
ResultSet rs = ps.executeQuery();
```

Figura 18. Consulta a la tabla *Locals*

Tabla *Events*: la consulta mostrada en la figura 19 presenta como variable LOCALS_IDL por la cual se filtrarán todos los eventos del local seleccionado por el cliente de la aplicación Android. Además aparecerán ordenados por la fecha en la que tendrán lugar.

```
PreparedStatement ps = connection.prepareStatement("SELECT IDL,NAME,TYPE,"  
    + "LONGITUDE,LATITUDE,ADDRESS,DESCRIPTION,CAPACITY,WEBSITE"  
    + " FROM LOCALS WHERE LOCALS_IDL='"+id_local+"' ORDER BY DATE ASC");  
ResultSet rs = ps.executeQuery();
```

Figura 19. Consulta a la tabla *Events*

En las figuras mostradas anteriormente están presentes tipos de variables desconocidas. *PreparedStatement* y *ResultSet* son unas de las variables de la librería *java.sql* que son imprescindibles para la interconexión del servidor con la base de datos en el lenguaje de programación Java.

4 SERVICIO WEB REST

4.1 Concepto

Un servicio web es una tecnología que, mediante el uso de un conjunto de protocolos y estándares, hace posible el intercambio de datos entre aplicaciones. Es justo lo que se necesita en este trabajo, por lo que es utilizado aquí. Además la arquitectura del servicio web que ha sido seleccionada es REST de cuyos principios se hablará en el apartado siguiente.

4.2 Cumplimiento de los principios REST

Esta es la tecnología utilizada para el desarrollo del servidor del proyecto.

Diseños fundamentales claves de REST:

- Ni el servidor ni el cliente necesita recordar ningún estado de las comunicaciones entre mensajes.
- Operación HTTP utilizada en este proyecto es GET.
- Uso de los parámetros en la URI para especificar el recurso consultado.
- La información y la transición de estado de la aplicación en el proyecto se representa mediante un documento XML.

4.3 Tipo de proyecto elegido para el desarrollo.

La herramienta que se va a utilizar para desarrollar el proyecto es Spring Tool Suite. Ambas tecnologías están presentadas anteriormente en el capítulo de las tecnologías y entornos utilizados.

El tipo de proyecto seleccionado en Spring Tool Suite ha sido MAVEN, porque en su arquitectura presenta los siguientes elementos necesarios para el trabajo:

- El subdirectorio **Java Resources/src** que contiene todos los recursos utilizados por la aplicación (paquetes con sus clases correspondientes) como se puede comprobar en la figura 20.

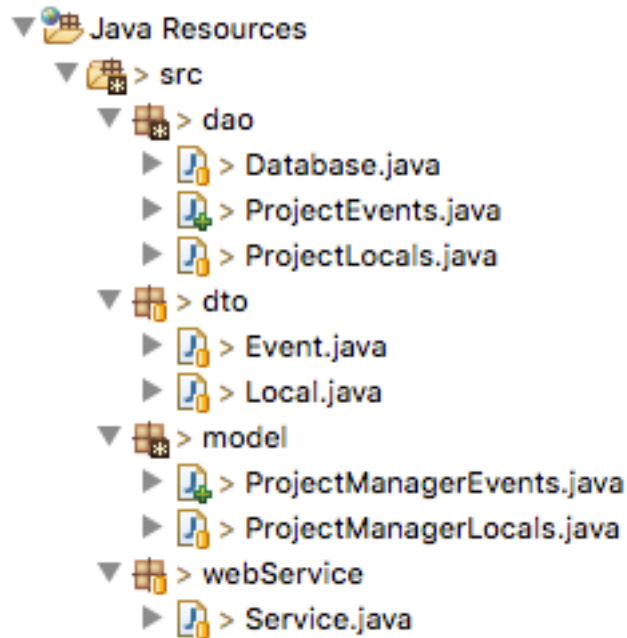


Figura 20. Subdirectorio **Java Resources/src**

- El directorio **WebContent** que alojará todos los archivos que no sean código fuente Java. En el caso de este trabajo es el archivo **web.xml**, dentro del que se define el servlet que se va a utilizar en el servicio, y las librerías incluidas, como se puede ver en la figura 21.

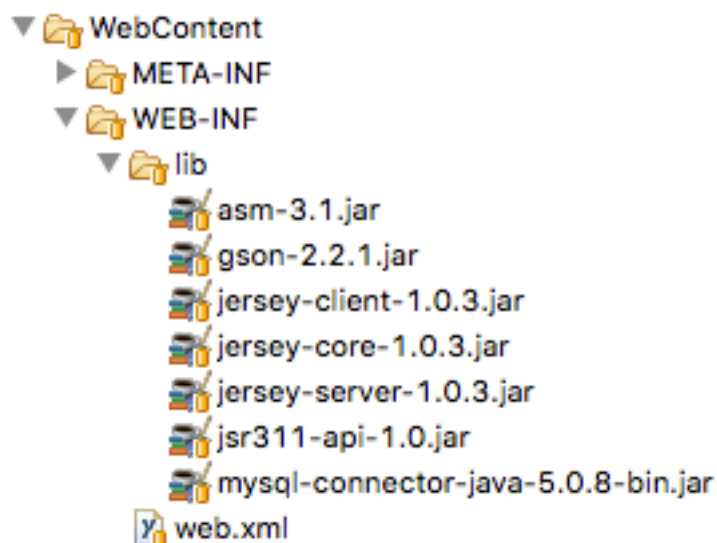


Figura 21. Directorio **WebContent**

- El directorio **target** donde se va a generar el archivo WAR que será utilizado para almacenar y desplegar rápidamente la aplicación.
- El fichero **pom.xml** que contiene las dependencias Maven necesarias. En el caso de este proyecto la dependencia necesaria es la de *Openshift* y su definición se

puede ver en la figura 22.

```
<dependencies>
  <dependency>
    <groupId>com.openshift</groupId>
    <artifactId>openshift-java-client</artifactId>
    <version>2.7.0.Final</version>
  </dependency>
</dependencies>
```

Figura 22. Dependencia Openshift

4.4 Diagrama de paquetes con sus clases

A continuación, mediante el diagrama de paquetes UML mostrado en la figura 23, se muestran los paquetes y las clases desarrollados en la aplicación Spring Tool Suite.

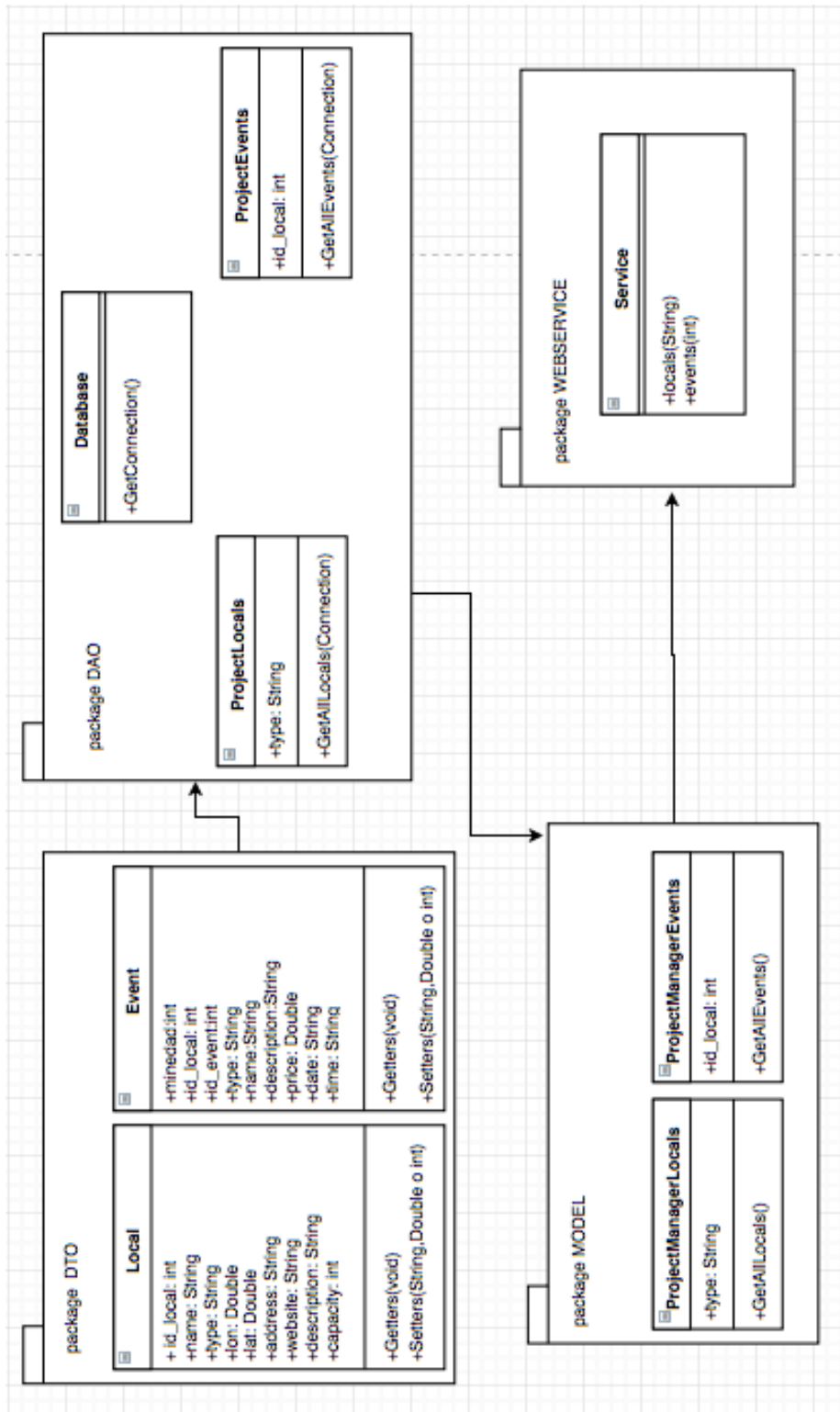


Figura 23. Diagrama de paquetes y sus clases

4.5 Librerías importadas en en trabajo

En este apartado se presentarán las librerías que han sido importadas a este proyecto y se explicará el porqué de su necesidad.

- *Java.util* : es una de las principales librerías que se usa cuando se programa en Java. En este proyecto se utiliza para crear una lista de elementos (ArrayList).
- *Javax.ws.rs* : es la librería que sirve para crear los recursos del servicio REST. En nuestro proyecto lo utilizamos en el fichero *Service* que se encuentra dentro del paquete *webService*. Los métodos que se utilizan de esta librería son GET, PATH, PRODUCES, QUERYPARAM.
- *Com.google.gson* : es una librería que sirve para convertir los objetos Java en objetos JSON. En este proyecto la información que se muestra en las URL es de tipo JSON por lo que es necesario convertirlos de Java a esta tecnología.
- *Java.sql* : esta librería proporciona la API para el acceso y procesamiento de los datos almacenados en una base de datos. En este proyecto se utiliza una base de datos por lo que es necesario la presencia de dicha librería.

4.6 Funcionamiento

En este apartado se explicará cómo funciona el servidor paso a paso cuando le llega una petición desde la aplicación Android.

El servidor web entra en funcionamiento cuando desde la aplicación se solicitan unos datos a mostrar. Por lo que lo primero que llega al servidor es la URL desde donde se quiere leer la información y el parámetro según si se está preguntando por los eventos de un local o listar los locales de un tipo, por lo que como parámetro podría llegar o id de tipo INT de un local para consultar los eventos o el tipo (String) para consultar todos los locales con las mismas características. A continuación en la figura 24 se muestra el ejemplo de una función de código Java donde se puede apreciar una posible ruta y el parámetro que se recibiría.

Como se puede ver el método es GET, lo que hace que se reciba la información del servidor. La URL que se utiliza tiene como último subdirectorio *locales* y convierte los objetos recibidos al formato JSON.

El parámetro que se le pasa a la función es *type*, lo que corresponde al tipo de locales seleccionado. Este parámetro lo pasamos a la instancia creada de la clase que se llama *ProjectManagerLocals* y con el resultado que se obtiene se rellena *ArrayList*, que es lo que se va a mostrar en la URL.

```
@GET
@Path("/locales")
@Produces("application/json")

public String locals(@QueryParam("type") String type)
{
    String alllocals = null;
    try
    {
        ArrayList<Local> AllLocals = null;
        ProjectManagerLocals projectManager= new ProjectManagerLocals(type);

        AllLocals = projectManager.GetAllLocals();

        Gson gson = new Gson();

        alllocals = gson.toJson(AllLocals);

    } catch (Exception e)
    {
        System.out.println("error");
    }
    return alllocals;
}
```

Figura 24. La ruta de los locales y la función para obtenerlos

A continuación se verá que es lo que pasa en el único método de la clase *ProjectManagerLocals*.

Las acciones realizadas en ese método son las siguientes:

- Se crea una instancia de la clase de la base de datos.
- Se establece la conexión con la base de datos.
- Se crea una instancia de la clase *ProjectLocals* y se le pasa el tipo como parámetro.
- Se devuelve la variable con los elementos obtenidos de *ProjectLocals* a la clase *Service*. Esos datos son los que se muestran en la URL.

En la figura 25 se puede ver el código de la función de la clase *ProjectMangerLocals* explicada paso a paso.

```

public ArrayList<Local> GetAllLocals()throws Exception {
    ArrayList<Local> alllocals = null;
    try {
        Database database= new Database();
        Connection connection = database.Get_Connection();
        ProjectLocals project= new ProjectLocals(type);
        alllocals=project.GetAllLocals(connection);

    } catch (Exception e) {
        throw e;
    }
    return alllocals;
}

```

Figura 25. Función *GetAllLocals* en la clase *ProjectManagerLocals*

Ahora hay que ver qué es lo que hay en las dos instancias de la clases creadas en el método anterior.

Primero se mostrará el contenido del método de la clase *Database*, pero la configuración será para el uso del servicio desde el ordenador personal, sin subirlo a la nube, ya que en el caso de subirlo se tendrían que poner el usuario y la contraseña para la base de datos. En la figura 26 se puede visualizar el método.

```

public Connection Get_Connection() throws Exception
{
    try
    {
        String connectionURL = "jdbc:mysql://localhost:3306/4EvenDB";
        Connection connection = null;
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connection = DriverManager.getConnection(connectionURL, "root", "");

        return connection;
    }
    catch (SQLException e)
    {
        throw e;
    }
    catch (Exception e)
    {
        throw e;
    }
}

```

Figura 26. Método de la clase *Database*

Ahora se va a mostrar el contenido del método utilizado de la otra clase que queda por ver, *ProjectLocals*. En la figura 27 se puede ver que contiene lo siguiente:

- Definición de una lista de instancias de la clase *Local*.
- Realización de una consulta a la base de datos con el parámetro que viene desde la petición de la aplicación Android, que en este caso es tipo de locales.
- Un bucle para introducir la información en el *ArrayList* de *Locals* que se llama *AllLocals*.
- Se devuelve el *ArrayList* con los objetos *Local* a la clase *ProjectManagerLocals*.

```
public ArrayList<Local> GetAllLocals(Connection connection) throws Exception
{
    ArrayList<Local> AllLocals = new ArrayList<Local>();
    try
    {
        PreparedStatement ps = connection.prepareStatement("SELECT IDL,NAME,"
            + "TYPE, LONGITUDE, LATITUDE, ADDRESS, DESCRIPTION, CAPACITY, "
            + "WEBSITE FROM LOCALS WHERE TYPE='"+type+"'");
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            Local local = new Local();
            local.setId_local(rs.getInt("IDL"));
            local.setName(rs.getString("NAME"));
            local.setType(rs.getString("TYPE"));
            local.setLon(rs.getDouble("LONGITUDE"));
            local.setLat(rs.getDouble("LATITUDE"));
            local.setAddress(rs.getString("ADDRESS"));
            local.setDescription(rs.getString("DESCRIPTION"));
            local.setCapacity(rs.getInt("CAPACITY"));
            local.setWebsite(rs.getString("WEBSITE"));

            AllLocals.add(local);
        }
        return AllLocals;
    }
    catch(Exception e)
    {
        throw e;
    }
}
```

Figura 27. Método *GetAllLocals* de la clase *ProjectLocals*

Lo que queda por mostrar es la clase *Local*, cuyas instancias se realizan en la clase *ProjectLocals*. En la figura 28 se puede visualizar el contenido de la misma que incluye:

- Definición de las variables características que pertenecen a un local.
- Definición de los métodos GET y SET de las variables.

```
public class Local {  
  
    private int id_local;  
    private String name;  
    private String type;  
    private Double lon;  
    private Double lat;  
    private String address;  
    private String website;  
    private String description;  
    private int capacity;  
  
    public int getId_local() {  
        return id_local;  
    }  
    public void setId_local(int id_local) {  
        this.id_local = id_local;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

Figura 28. Clase *Local*

No se han mostrado todos los métodos, ya que son todos iguales para todas las variables, solo lo que cambia es el tipo de la variable en las funciones.

Instancias de esa clase se realizan en la clase *ProjectLocals*.

Respecto a las consultas de los eventos, el procedimiento es muy similar pero el proceso se realiza por las clases distintas a las mencionadas en este apartado y el ArrayList de las instancias se obtiene de la clase *Event* en vez de *Local* que tiene variables características de un evento. En el diagrama de paquetes mostrado en la figura 9 se pueden visualizar todas

las clases, sus variables y los métodos utilizados en cada una.

Eso es respecto al código Java del servidor, pero para que la información se muestre en una URL necesitamos definir un servlet en el archivo **web.xml** que se encuentra en el subdirectorio **WebContent/WEB-INF**. En la figura 29 se puede ver como en este archivo se define el nombre de la clase y el patrón para dicho servlet, que son necesarios para que el servicio muestre la información solicitada.

```

<servlet>
  <servlet-name>ServletAdaptor</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>ServletAdaptor</servlet-name>
  <url-pattern>/REST/*</url-pattern>
</servlet-mapping>

```

Figura 29. Servlet en el archivo **web.xml**

Básicamente, el diagrama del funcionamiento del servidor sería como se muestra en la figura 30.

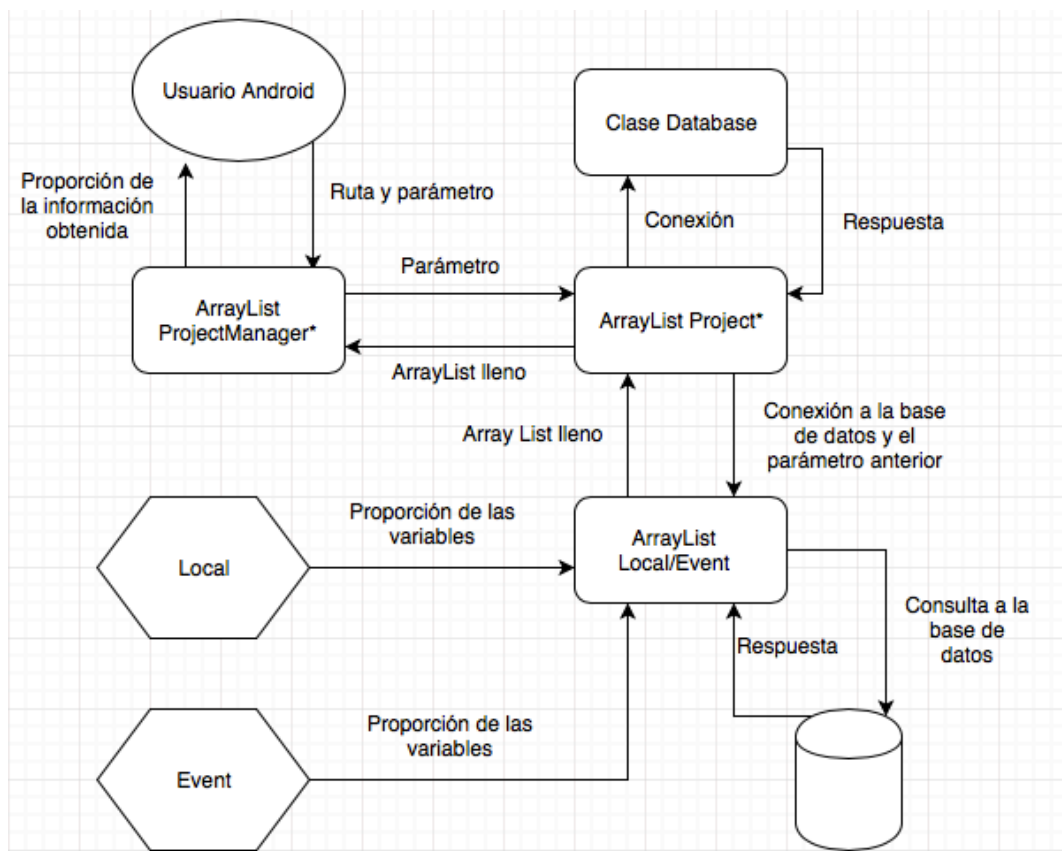


Figura 30. Diagrama del funcionamiento del servidor

4.7 Exportación del servicio

Al terminar el desarrollo del servidor se genera un archivo .WAR , que se hace exportando todo el código implementado a algún directorio del sistema que se utiliza, ya que lo que se quiere en este trabajo es subir el servicio a la nube, que posteriormente se verá el procedimiento de esa tarea.

Para exportar el proyecto de Spring Tool Suite lo que hay que hacer es clicar con el botón derecho sobre el proyecto seleccionado e ir a la opción Export -> WAR file, como se puede ver en la figura 31.

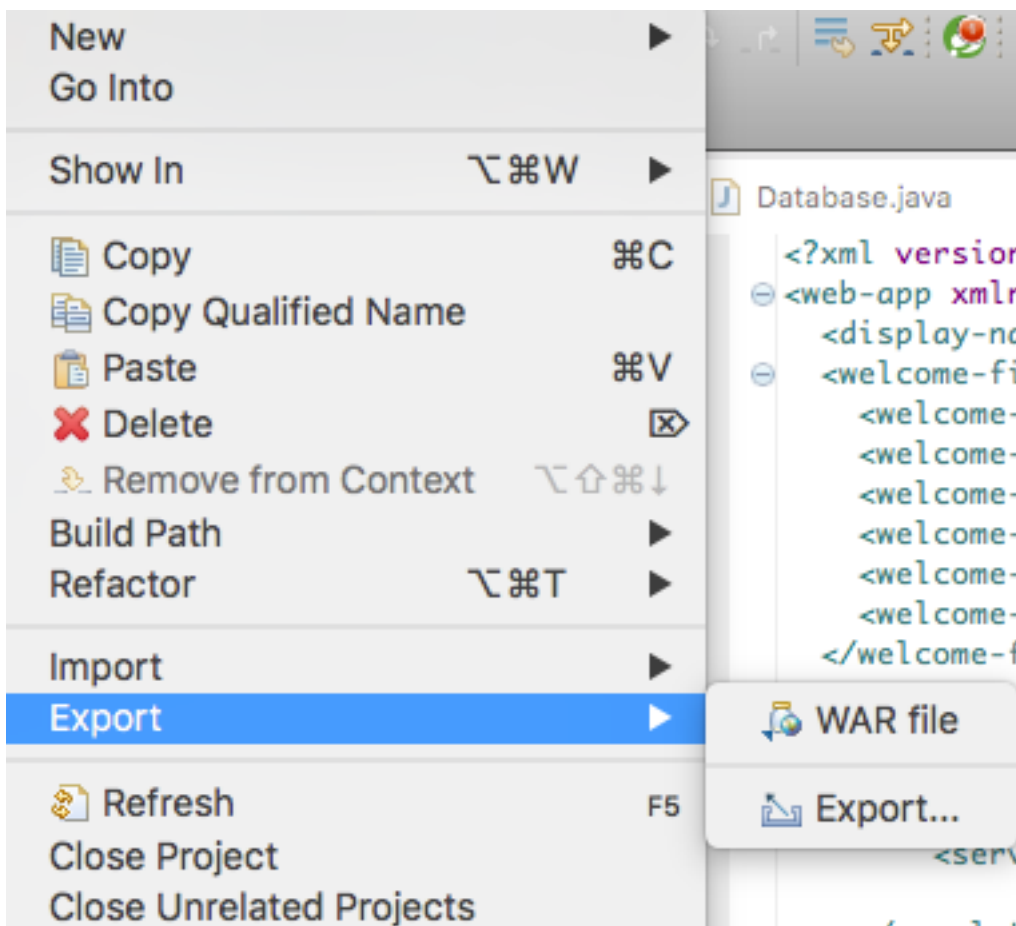


Figura 31. Exportación del servicio

Posteriormente seleccionar el nombre del proyecto y el destino y dar a finalizar, esta acción se puede visualizar en la figura 32. Y con esto ya se tiene el proyecto exportado.

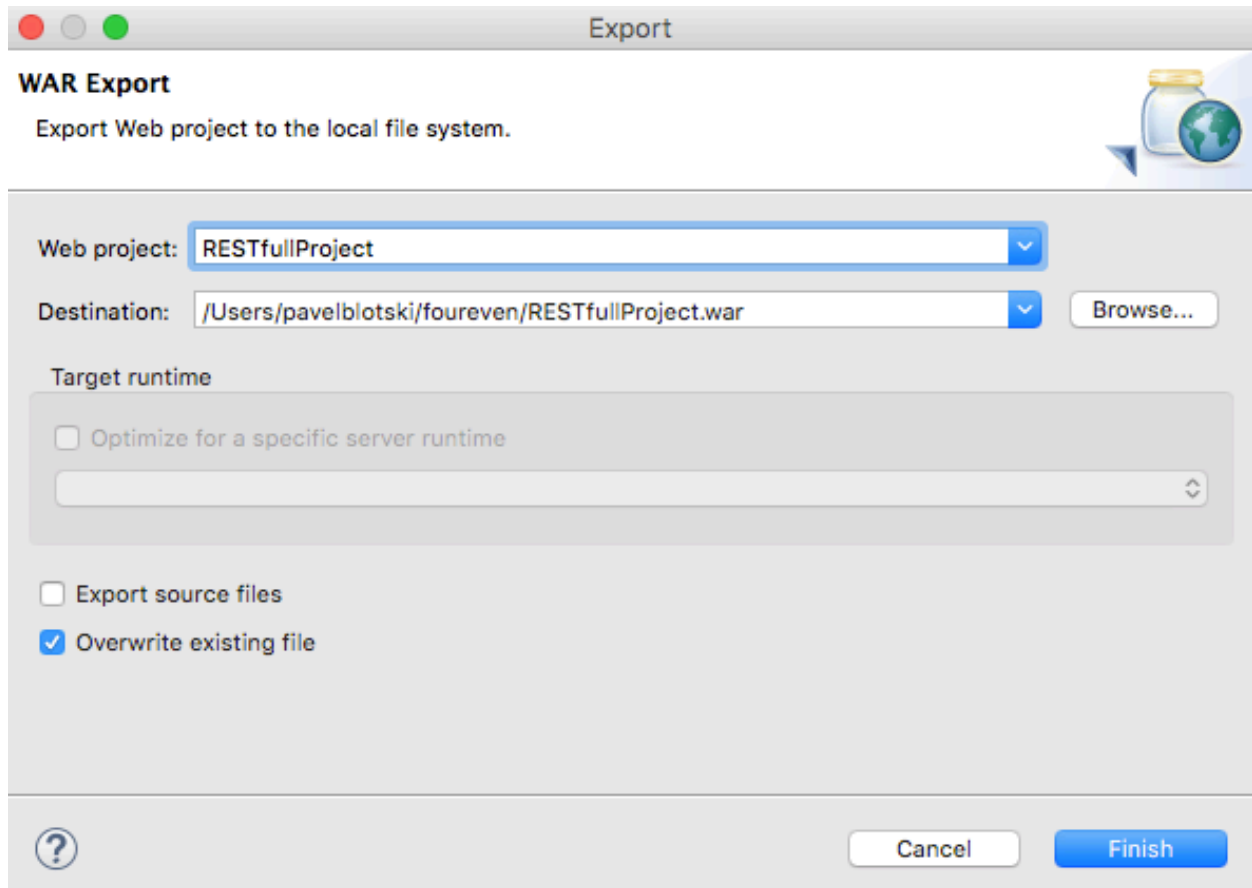


Figura 32. Parámetros de la exportación de un servicio

4.8 Subida a la nube utilizando el servicio de Openshift

En un principio durante el desarrollo del servicio para hacer distintas pruebas se ha utilizado el servidor que estaba instalado en el mismo sistema, entonces el acceso desde un navegador al servicio era http://localhost:8080/URL_aplicación, pero para utilizar una aplicación desde un terminal móvil el servicio tiene que estar subido en la nube.

Para terminar el capítulo del servicio web, se va a explicar el proceso de subida de un servidor a la nube, que en este caso el servicio seleccionado ha sido Openshift, ya que es el espacio especificado para servidores Java y tiene la opción de utilizarlo gratuitamente.

Entonces, los pasos para subir el servidor a Openshift son los siguientes:

- Hay que registrarse en www.openshift.com.
- Verificar por el correo electrónico el registro.
- Seleccionar en el menú desplegable superior derecho la opción “OPENSIFT WEB CONSOLE”, como se muestra en la figura 33.

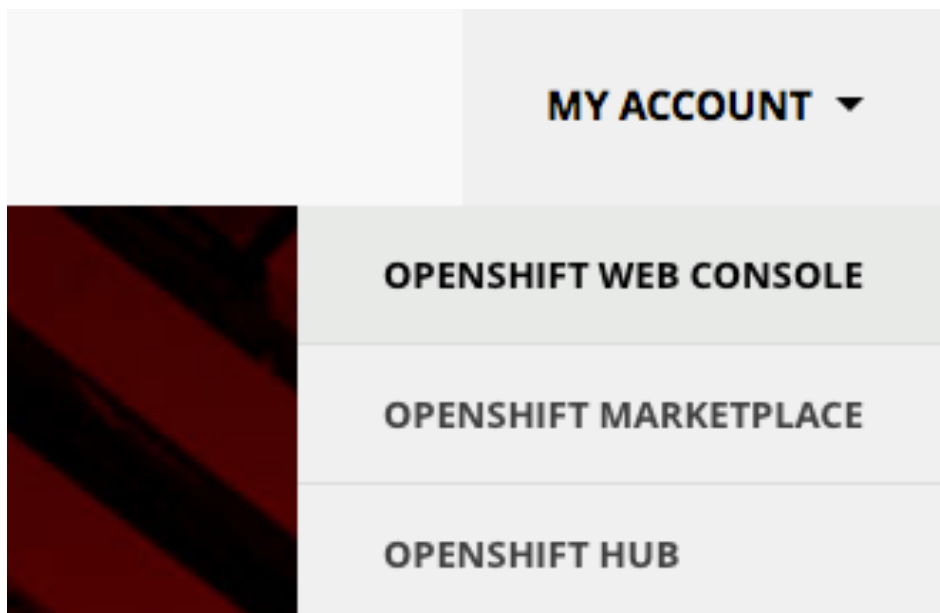


Figura 33. Selección de OPENSIFT WEB CONSOLE

- En la siguiente pantalla lo que se visualiza son las aplicaciones creadas por el usuario registrado y si no hay ninguna creada se empieza a crear una dando al botón de “Add Application”. En la figura 34 se ve esta pantalla.

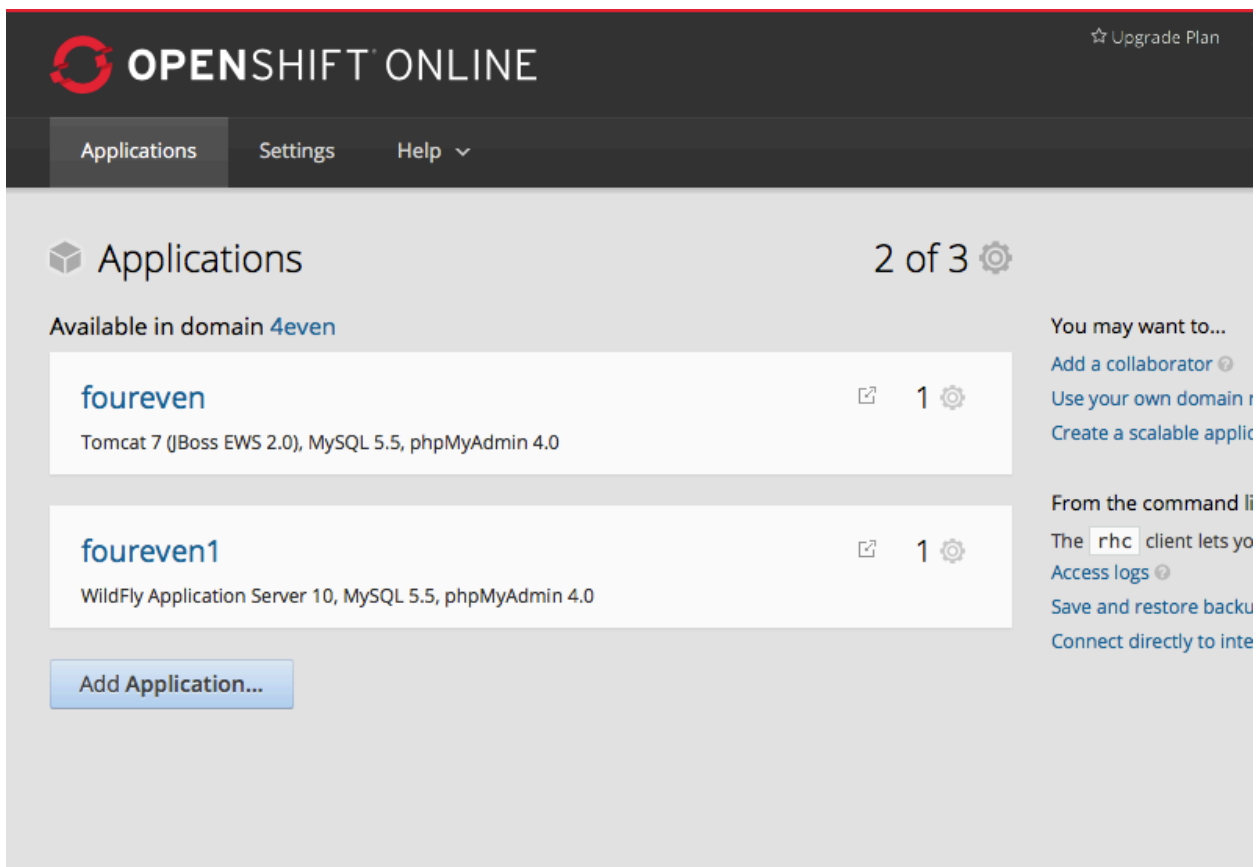


Figura 34. Pantalla principal de OPENSIFT WEB CONSOLE

- Una vez que se clicca en el botón “Add Application” Openshift ofrece una lista de tipos de servidores disponibles. En el caso de este proyecto ha sido seleccionado servidor “WildFly Application Server 10”, que utiliza la versión 1.8 de Java, la misma versión utilizada por el Spring Tool Suite, lo que hace que ambos sean compatibles. Hay que tener cuidado a la hora de seleccionar el tipo de servidor porque si no son compatibles Openshift dará como respuesta el error interno 500. Una vez seleccionado el servidor se pasa a la siguiente pantalla de configuración del mismo.
- Los parámetros que se pueden configurar una vez se ha seleccionado el servidor son los siguientes:
 - La URL pública del servidor.
 - Fuente del código del servidor desarrollado (opcional).
 - Los contenedores que arrancan la aplicación, que en este caso no es posible cambiar ya que se va a utilizar la versión gratuita. Por tanto, la opción es “small”.
 - Las partes que contiene la aplicación o “Cartridges”, que se van a agregar posteriormente, no ahora.
 - Elegir si la aplicación va a ser escalable o no.
 - Seleccionar la región donde se encuentra el servidor de Openshift.
 - Y, al final, se pulsa a “Create Application”. El proceso de la creación no es instantáneo, puede tardar unos dos minutos.
- Ya se tiene la aplicación creada. La pantalla que aparece muestra la información básica de los pasos siguientes. Hará falta clonar el directorio de la aplicación de Openshift al sistema del usuario y posteriormente, mediante comandos de git, trabajar con dicho directorio, pero eso se explicará en los apartados posteriores. Ahora lo que se hará es cliccar en “Applications” lo que llevará a la lista de las aplicaciones existentes creadas por el usuario y se selecciona la aplicación creada en el apartado anterior. En la figura 35 se puede ver dicha pantalla.

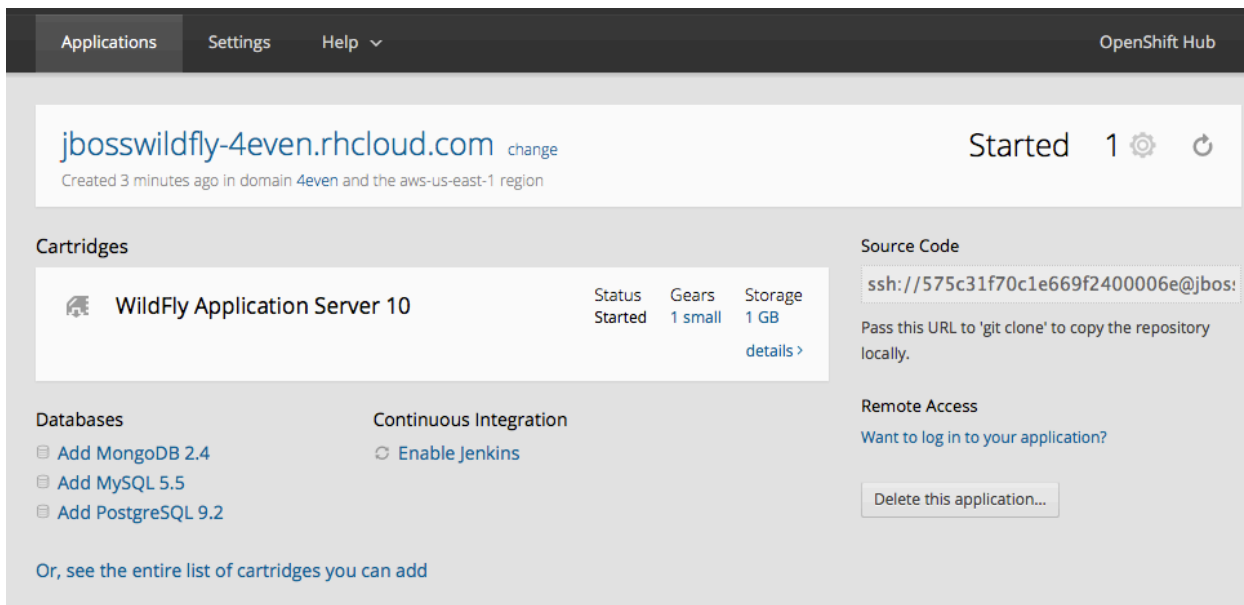


Figura 35. Aplicación creada en Openshift

- Lo que se hará posteriormente es la agregación de los cartridges necesarios para el trabajo. En el caso de este proyecto los que se han agregado son los cartridges MySQL 5.5 y phpmyadmin 4.0.
- Una vez agregadas estas partes al proyecto se puede gestionar el sistema de la base de datos mediante el gestor phpmyadmin que tiene su usuario y contraseña.
- El siguiente paso a realizar será clonar el directorio de la aplicación proporcionado anteriormente. Esto se realizará mediante comando *git clone*. La dirección del directorio de la aplicación se puede ver en la figura 35, que está debajo de la etiqueta “Source Code” y el segundo directorio es el elegido por el usuario en su sistema, es donde se va a clonar el directorio de la aplicación de Openshift. El comando que hay que ejecutar se muestra en la figura 36.

```
git clone <git_url> <directory_to_create>
```

Figura 36. Comando para clonar el directorio de Openshift a un directorio local

- Lo siguiente en realizar será copiar el archivo `.war` del servicio desarrollado y exportado de Spring Tool Suite en el subdirectorio **deployments** de la carpeta clonada en el apartado anterior.

- Y, por último, lo que queda por hacer es ejecutar una serie de comandos git en el terminal necesarios para subir el archivo .war a la nube para que sea utilizado por la aplicación. Los comandos se muestran en la figura 37.

```
cd directorio_clonado
git add .
git commit -m 'Comentario'
git push
```

Figura 37. Comandos git para subir el archivo .war a la nube

Al realizar la secuencia de estos pasos se obtiene el resultado idéntico ejecutando la misma dirección, solo que hay que cambiar “localhost:8080” por la URL proporcionada por la aplicación de Openshift.

5 APLICACIÓN ANDROID

5.1 Introducción

En este capítulo se va a proporcionar toda la información sobre la estructura y el funcionamiento de la aplicación. se mostrarán los diagramas UML para explicar el uso completo, se hablará de cómo funciona la aplicación y, a continuación, se describirán las herramientas incluidas en este proyecto.

5.2 Diagrama de casos de uso

A continuación, en la figura 38, se muestran todos los casos de uso disponibles para el usuario de la aplicación.

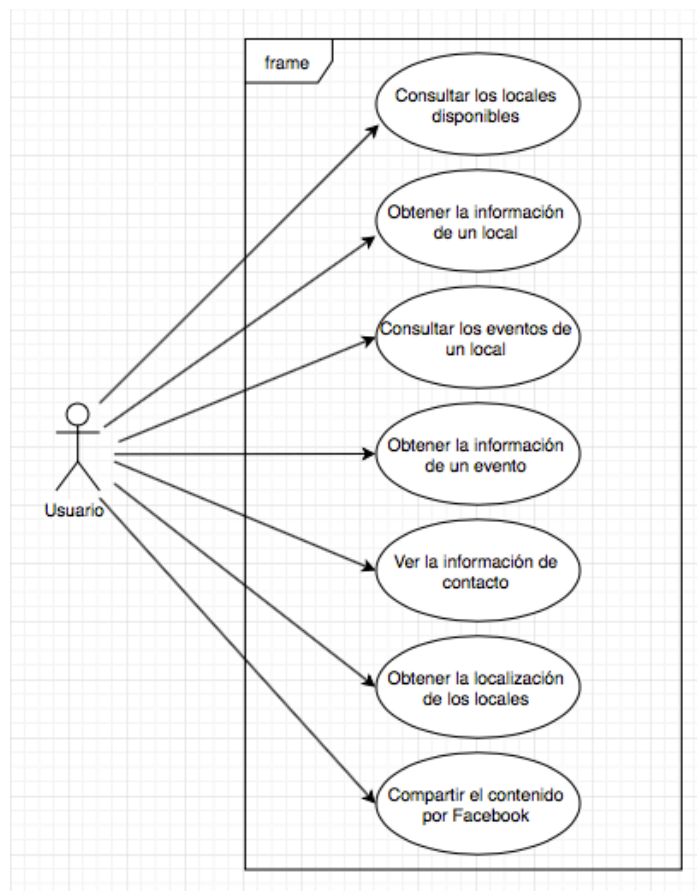


Figura 38. Diagrama de casos de uso

5.3 Diagramas de secuencia

En este apartado se va a ver la secuencia de los pasos que hay que seguir para completar cada caso de uso:

- Consultar los locales disponibles agrupados por el mismo tipo de local. Su visualización está disponible en la figura 39.

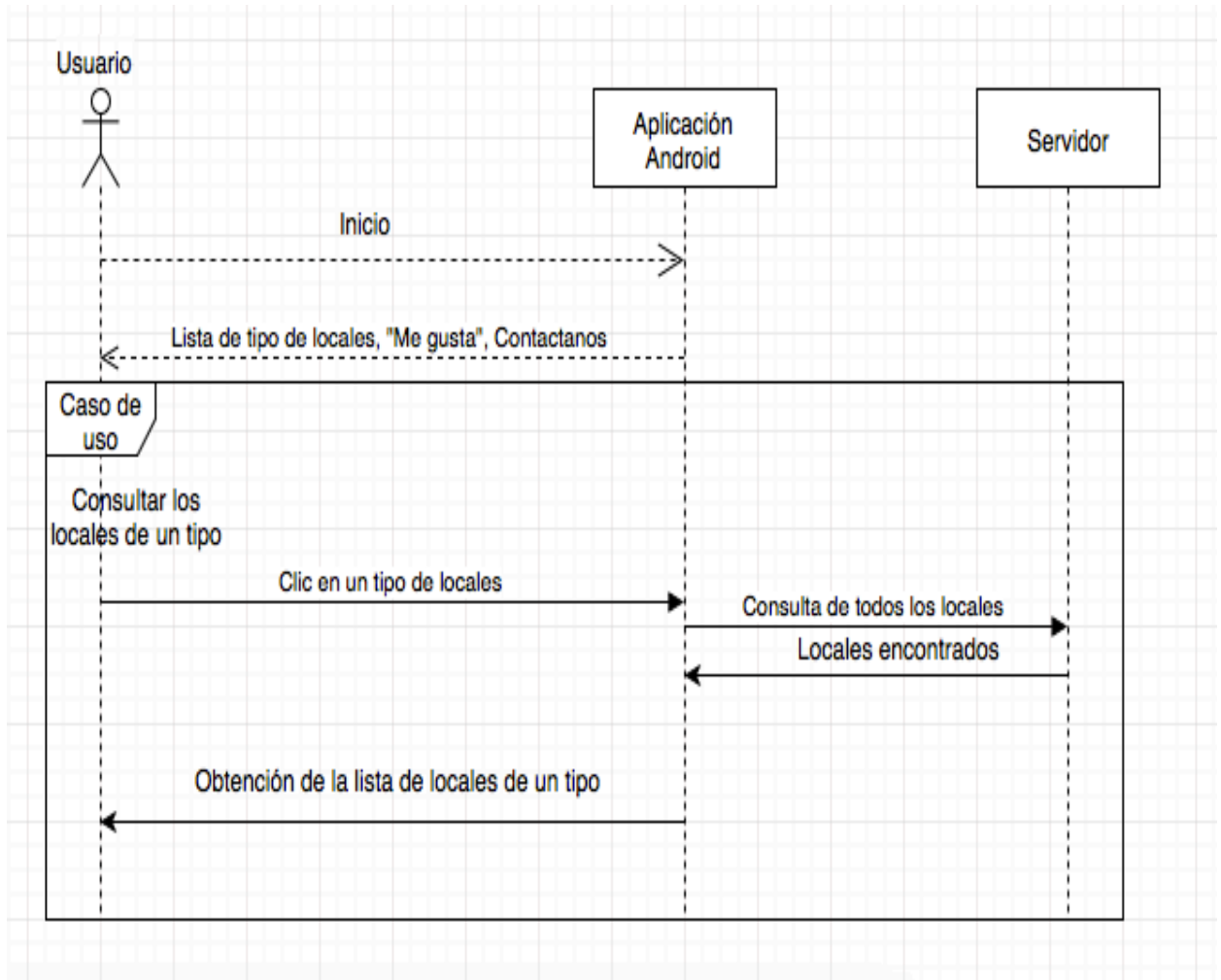


Figura 39. Consulta de los locales de un tipo

- Obtener la información de un local. Su diagrama de secuencia se puede ver en la figura 40.

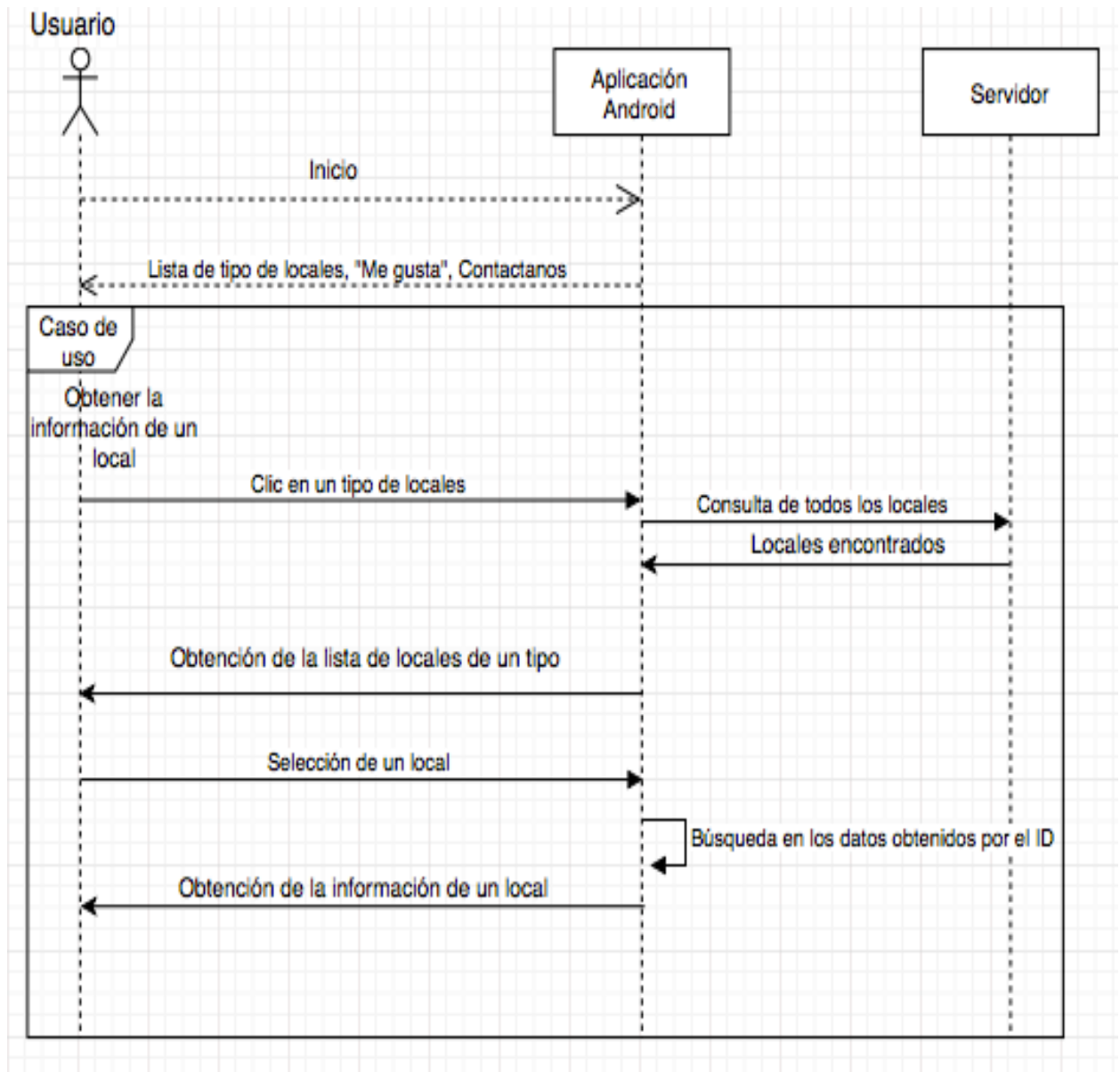


Figura 40. Consulta de la información acerca de un local

- Consultar los eventos de un local seleccionado. Su diagrama de secuencia se puede ver en la figura 41.

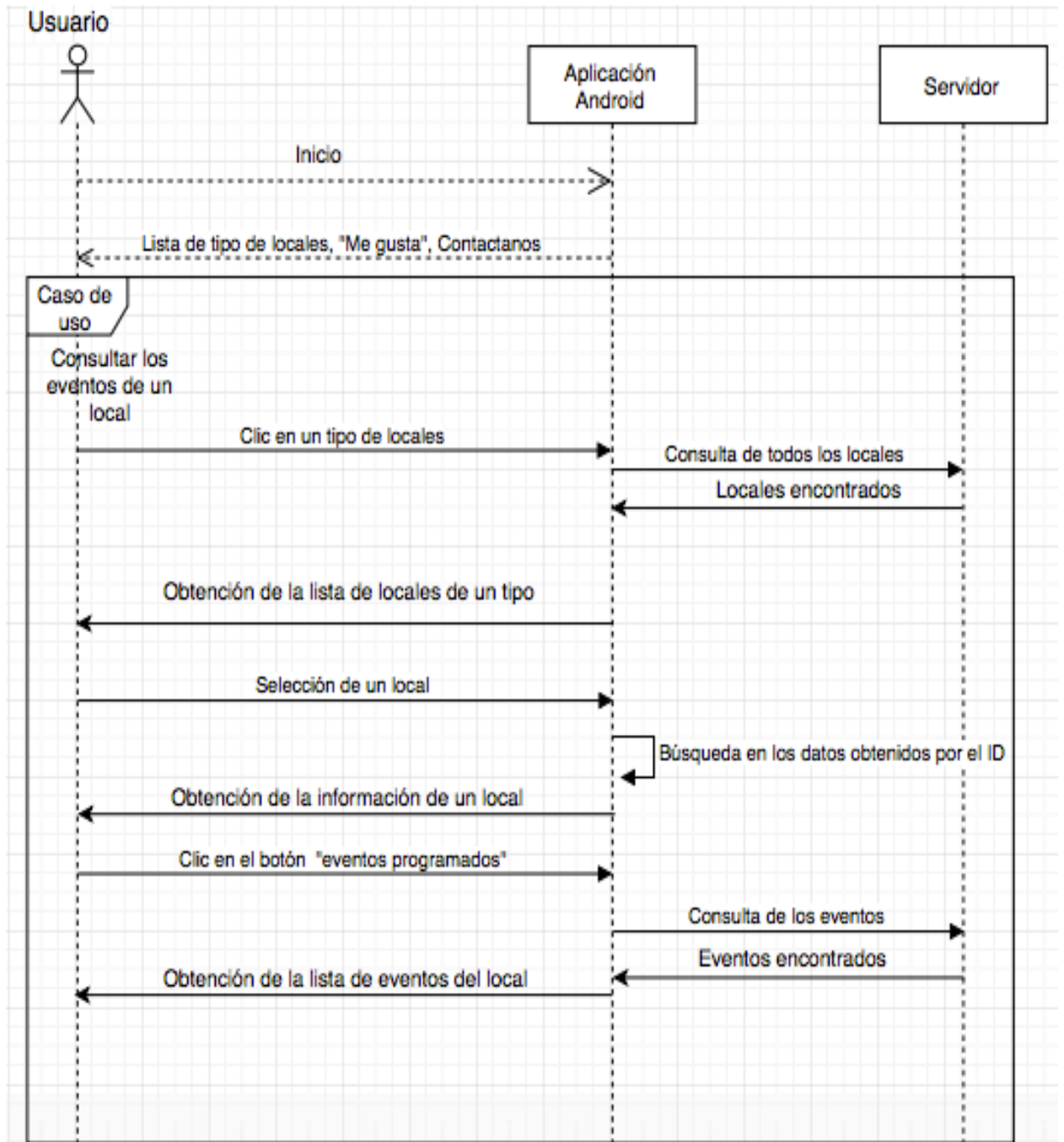


Figura 41. Consulta de los eventos de un local.

- Consulta de la información de un evento. Su diagrama de secuencia se puede ver en la figura 42.

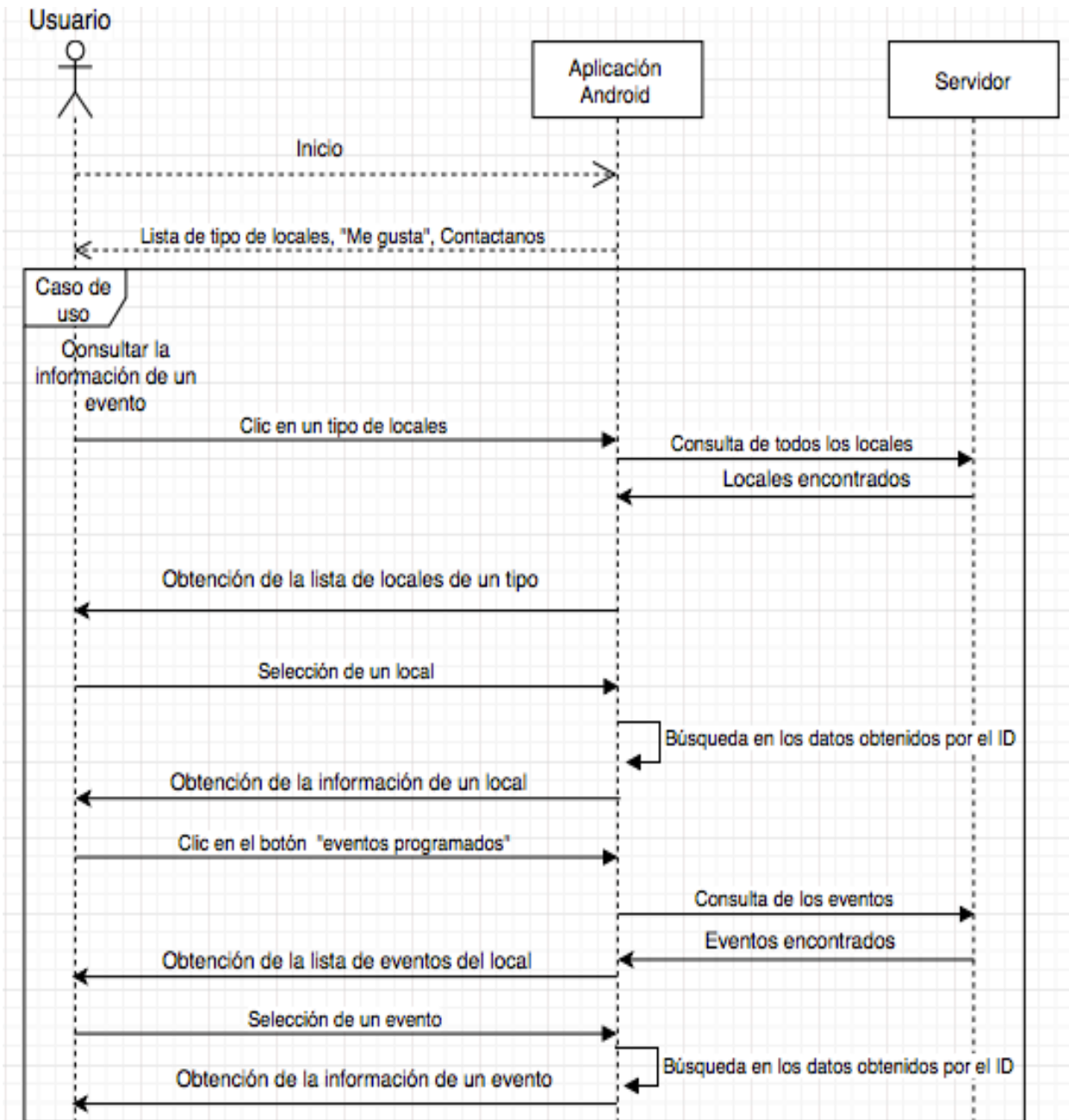


Figura 42. Consulta de la información de un evento

- Visualizar la localización de los locales en el mapa. Su diagrama de secuencia se puede ver en la figura 43.

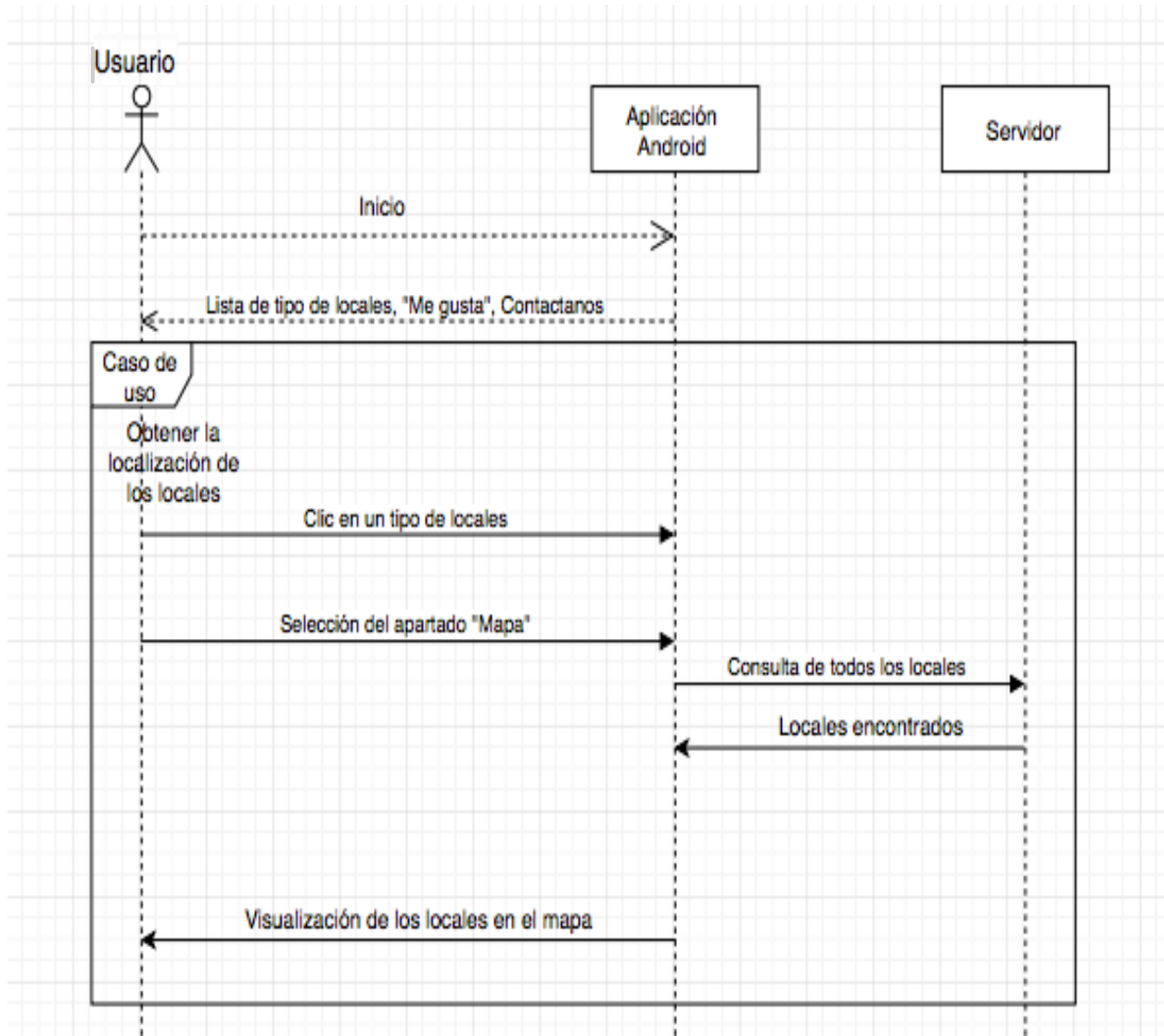


Figura 43. Visualización de los locales en el mapa

- Obtención de los datos de contacto. Su diagrama de secuencia se puede ver en la figura 44.

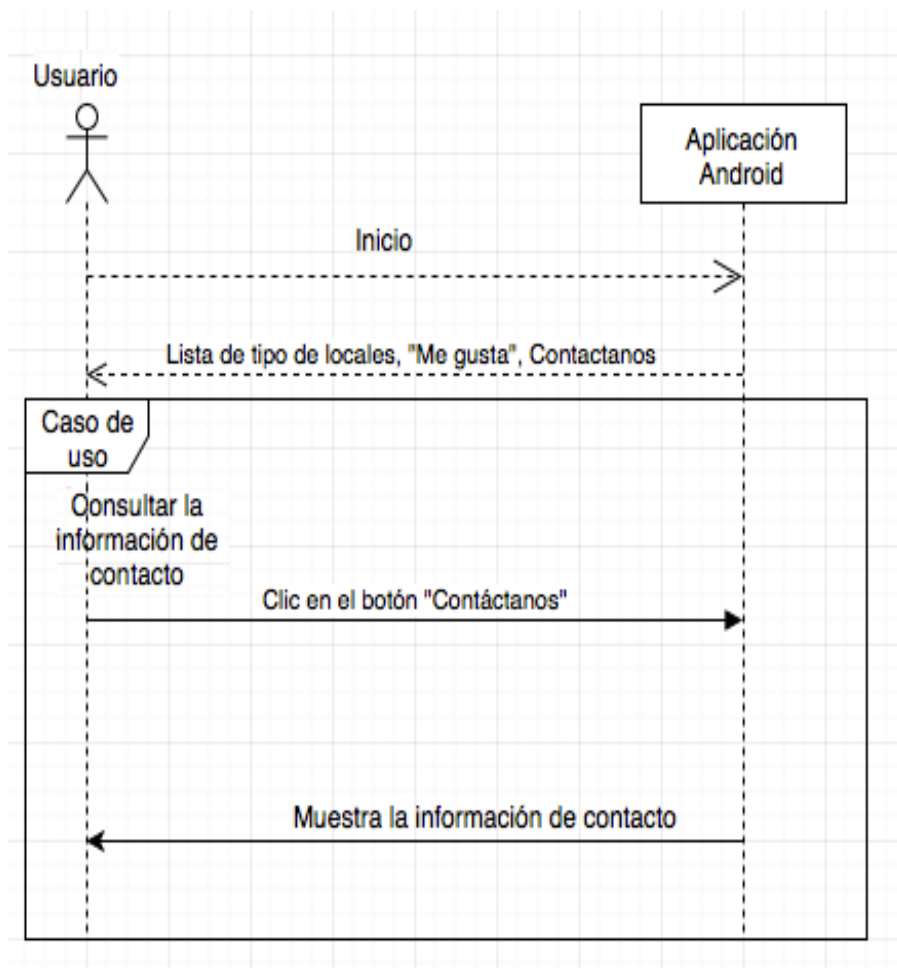


Figura 44. Consulta de la información de contacto

- Compartir el contenido en Facebook. Su diagrama de secuencia se puede ver en la figura 45.

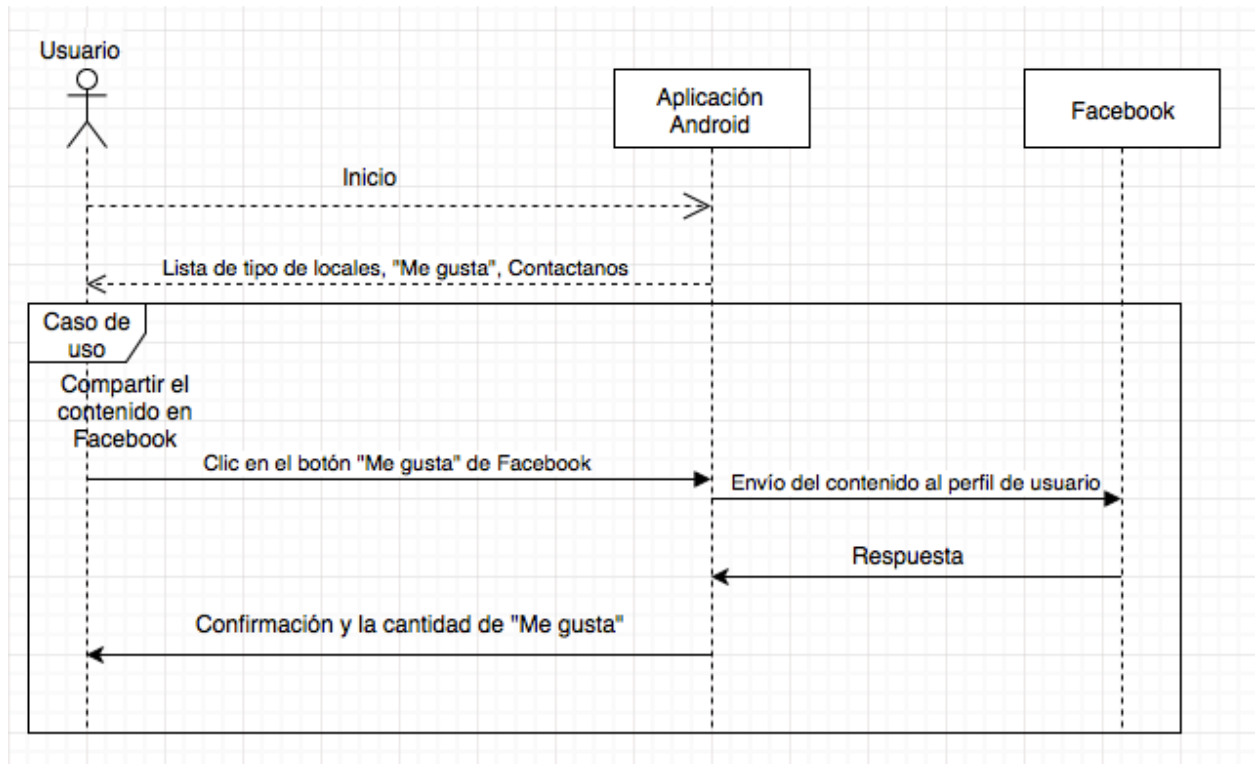


Figura 45. Compartir el contenido en Facebook.

5.4 Funcionamiento

En este apartado se va a explicar como funciona la aplicación. Para hacer esto se hará uso de los diagramas de secuencia anteriores, así quedará claro que es lo que se hace en el código en cada caso. Posteriormente se pondrá una parte de código para explicar el funcionamiento entre la aplicación y el servidor.

- **Explicación de la pantalla principal.**

Lo que se tiene al abrir la aplicación es un slide de imágenes estáticas y una lista estática de los tipos de locales disponibles. Al final de la pantalla lo que se tiene son dos botones, que son “Me gusta” para compartir el contenido en Facebook y el otro botón “Contáctanos” para ver la información de contacto. En la figura 46 se muestra esta pantalla principal.

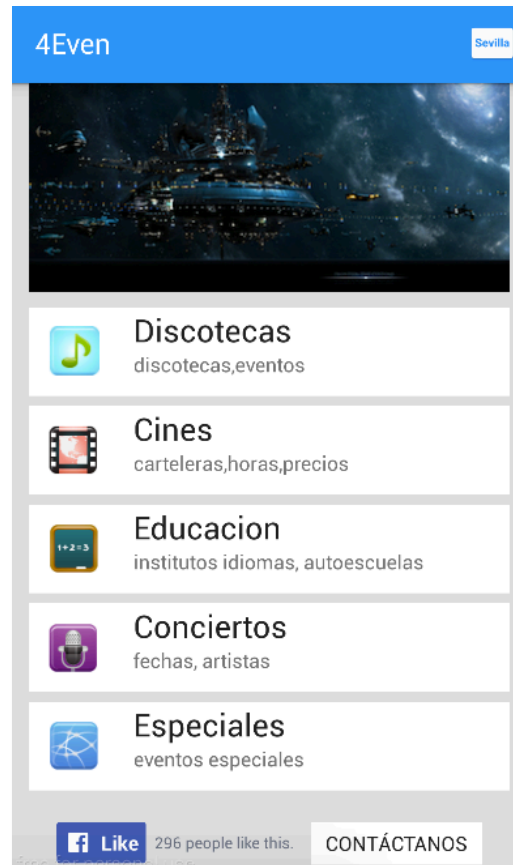


Figura 46. Pantalla principal de la aplicación

- **Explicación de los diagramas de secuencia de consultar los locales de un tipo y de ver la localización de los mismos .**

Ahora se selecciona un tipo de locales, lo que inicia otra actividad de la aplicación y se pasa como parámetro el tipo seleccionado que servirá para hacer la consulta al servidor, que se encarga de obtener los locales de dicho tipo desde la base de datos.

Lo que se tiene en esta actividad son dos fragmentos, pero se muestran por separado según el elemento que está seleccionado en la barra. Los elementos son “Locales” y “Mapa”.

En el fragmento “Locales” se visualiza la lista de los locales que se obtienen del servidor. Los parámetros pasados son la URL a consultar y el tipo de locales. Se obtiene la información en formato JSON y se va rellenando la lista de los locales de este tipo en dicho fragmento. Se puede visualizar este fragmento en la figura 47.



Figura 47. Fragmento “Locales”

En el fragmento “Mapa” se incorpora el mapa de Google Maps. Para hacer el uso de este mapa previamente hay que obtener la clave de desarrollador de esta API e integrarla en el proyecto.

En este fragmento se hace la consulta de la información de los locales seleccionados para sacar los parámetros de localización (longitud y latitud) de los objetos JSON obtenidos, los que posteriormente son utilizados para colocar dichos locales en el mapa definido anteriormente. También se agrega la posición actual del usuario.

Todos los locales están situados en la misma ciudad, así que el foco del mapa es el centro de la ciudad y la ampliación es adecuada para que abarque todo su tamaño. Estos parámetros están configurados en el fragmento anterior que se puede ver en la figura 48.



Figura 48. Fragmento “Mapa”

- **Explicación del diagrama de secuencia de la información acerca de un local.**

Ya se obtenido toda la información acerca de los locales seleccionados y ahora lo que se hace es clicar sobre un elemento de la lista del fragmento “Locales” lo que lleva a la siguiente actividad. Esta actividad recibe como parámetros toda la información obtenida del objeto JSON de la actividad anterior que describe el local seleccionado. No se hace ninguna consulta. Se muestra en la pantalla toda la información obtenida en una lista. Aparte se puede ver un slide de imágenes estáticas y dos botones. El primer botón lleva a una actividad que solamente muestra la descripción completa del evento y el otro botón “Eventos programados” que lleva a la actividad que será descrita en el siguiente apartado. Esta actividad se puede ver en la figura 49.

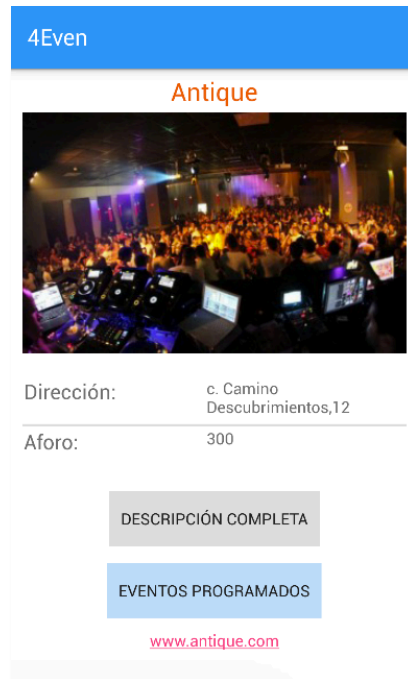


Figura 49. Información de un local

Explicación del diagrama de secuencia de obtener los eventos de un local seleccionado.

A partir de la actividad anterior, donde se obtiene toda la información acerca de un local, se clica en el botón “Eventos programados” y esto inicia otra actividad nueva que recibe como parámetros el nombre del local y el id para realizar la consulta al servidor y seleccionar los eventos que le corresponden. Los elementos obtenidos se colocan en una lista mostrando la fecha, el nombre del evento y su tipo. El nombre del local se pone en la parte superior de la pantalla para informar al usuario sobre el local que se consulta. A continuación en la figura 50 se puede ver dicha pantalla.



Figura 50. Los eventos programados de un local

- **Explicación del diagrama de obtención de la información acerca de un evento.**

Ya se ha obtenido toda la información acerca de los eventos mediante la consulta al servidor. La información recibida se ha guardado en los objetos JSON. Ahora lo que se hace es seleccionar un elemento de la lista mostrada en la figura 50. Esto lleva a una nueva actividad y los parámetros pasados son los que describen el evento por completo. Esta pantalla se puede ver en la figura 51. Lo que se muestra es una lista con los parámetros obtenidos. El elemento “Descripción” de esa lista lleva a una actividad nueva, que muestra solamente el texto que proporciona la información adicional acerca del evento .



Figura 51. Información de un evento

- **Explicación de los diagramas de secuencia de los botones en la pantalla principal.**

Los botones “Me gusta” y “Contáctanos” son visibles en la pantalla principal y al realizar el clic sobre alguno de ellos se realiza lo siguiente:

- El botón “Me gusta” comparte el contenido de una página HTML en el perfil del usuario de Facebook. En este proyecto no se ha diseñado dicha página por lo que se ha cogido una para probar que funciona. En la figura 52 se muestra cómo se comparte dicho contenido.



Figura 52. Compartición del contenido en el perfil de Facebook

Y lo que se puede ver en la aplicación como respuesta al realizar esta acción se puede ver en la figura 53.

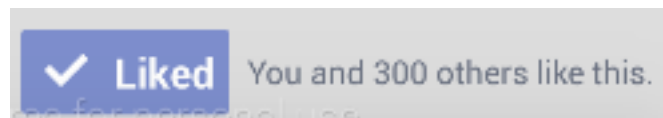


Figura 53. Respuesta de Facebook a la aplicación

- El botón “Contáctanos” inicia una actividad nueva que no recibe ningún parámetro. Lo que hace es mostrar un texto que contiene la información de contacto con los desarrolladores de la aplicación.

- **Conexión entre la aplicación y el servidor.**

Lo que se hace es definir la URL en un cadena de caracteres incorporando en ella los parámetros necesarios para realizar la consulta apropiada. Esa cadena es utilizada como la dirección URL para realizar la conexión y obtener los datos. El contenido que se obtiene se guarda en otra cadena de caracteres. Posteriormente la cadena obtenida es transformada a un JSONArray donde la información se parte en objetos. Así se puede obtener y mostrar la lista de los elementos proporcionados desde la URL consultada. En las figuras 54, 55 y 56 se muestra la parte del código correspondiente.

```
HTTPDataHandler hh = new HTTPDataHandler();
stream = hh.GetHTTPData(urlString);

// Return the data from specified url
return stream;
```

Figura 54. Inicio de instancia de la clase HTTPDataHandler y utilización de uno de sus métodos con el parámetro la URL que se quiere consultar

```
public class HTTPDataHandler {

    static String stream = null;

    public HTTPDataHandler(){
    }

    public String GetHTTPData(String urlString){
        try{
            URL url = new URL(urlString);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();

            // Check the connection status
            if(urlConnection.getResponseCode() == 200)
            {
                // if response code = 200 ok
                InputStream in = new BufferedInputStream(urlConnection.getInputStream());

                // Read the BufferedInputStream
                BufferedReader r = new BufferedReader(new InputStreamReader(in));
                StringBuilder sb = new StringBuilder();
                String line;
                while ((line = r.readLine()) != null) {
                    sb.append(line);
                }
                stream = sb.toString();
                // End reading.....

                // Disconnect the HttpURLConnection
                urlConnection.disconnect();
            }
        }
    }
}
```

Figura 55. Clase HTTPDataHandler (la parte de try)

```

protected void onPostExecute(String stream) {

    if (stream != null) {
        try {
            // Get the full HTTP Data as JSONArray
            reader = new JSONArray(stream);

            ArrayList<Lista_entrada3> datos = new ArrayList<>();

            for(i=0; i<reader.length();i++) {

                coord = reader.getJSONObject(i);

                name_e = coord.getString("name");
                tipo_e = coord.getString("type");
                date = coord.getString("date");
                int ID_local = coord.getInt("id_local");

                // SE AÑADEN A LA LISTA LOS LOCALES
                if(id_local == ID_local) {
                    datos.add(new Lista_entrada3(date, name_e, tipo_e));
                }
            }
        }
    }
}

```

Figura 56. Partición de la cadena recibida en objetos JSON

En este capítulo también es importante destacar todas las partes del proyecto Android que son necesarias para hacer que la aplicación funcione y su diseño sea adecuado. Son las siguientes:

- **AndroidManifest.xml** Es un archivo de configuración donde se puede aplicar las configuraciones básicas de la aplicación. En este proyecto ha sido necesario modificarlo para agregar todas las actividades necesarias y para agregar los parámetros de configuración para el uso de las APIs de Facebook y Google Maps.
- **res:** es un subdirectorío del proyecto donde se encuentran todos los archivos con los recursos que usa la aplicación. Los subdirectoríos que contiene son los siguientes:
 - **anime:** se utiliza en el proyecto para configurar la duración de los elementos como slide de imágenes o la presentación del logo antes de la pantalla principal.
 - **drawable:** es útil en el proyecto para utilización de todas las ilustraciones de la aplicación.
 - **layout:** sirve en el proyecto para configurar las interfaces visuales asociadas a las actividades.

- **menu**: utilizado para configurar la barra del menú de la aplicación.
- **values**: contiene todas las variables utilizadas por la aplicación como colores, dimensiones, cadenas definidas y estilos.
- **Gradle scripts**: es un subdirectorio proporcionado por Android Studio que ayuda a construir un proyecto. Se ha modificado en este proyecto para agregar las dependencias necesarias y para configurar las versiones herramientas de Android.

5.5 APIs utilizadas en el proyecto

En este apartado se van a explicar las APIs que están incluidas en el desarrollo de esta parte y para qué sirven. Son las siguientes:

- **Android.io**: proporciona los servicios básicos del sistema operativo, el paso de mensajes, y la comunicación entre procesos en el dispositivo. En la aplicación se utiliza para:
 - Pasar una serie de parámetros entre las actividades.
 - Enviar mensajes y objetos relacionados con la cola de mensajes de un hilo.
 - El uso apropiado y fácil de subprocessos de la interfaz de usuario.
- **Android.content**: se utiliza para el acceso y la publicación de los datos en el dispositivo. En el proyecto sirve para iniciar las distintas actividades implementadas.
- **Android.view**: Se encarga del diseño de la pantalla y la interacción con el usuario. En el proyecto se utiliza para mostrar los distintos menús y todos los objetos que aparecen en la pantalla.
- **Android.support.v7**: ofrece una serie de herramientas que no están integradas en el framework. En el caso de esta aplicación se utiliza para agregar funciones a la barra de acción.
- **Android.widget**: Permite utilizar los widget. En el proyecto se utiliza para la agregación de los botones, las listas y las imágenes.
- **Android.location**: Define los servicios basados en localización. En la aplicación se utiliza para mostrar gráficamente la localización de los locales en el mapa.

- Com.facebook: Proporciona las herramientas para compartir la información de la aplicación en Facebook. En el proyecto existe un botón “me gusta”, que lo que hace es compartir el contenido en el perfil de Facebook del usuario.
- Com.google: Permite el uso de los mapas de Google. En esta aplicación sirve para incorporar el mapa, que es utilizado para mostrar la localización de los locales consultados.
- Java.util: Contiene el framework de las colecciones, modelo de eventos, facilidades para la fecha y la hora y otras herramientas de diversa utilidad. En el proyecto es necesario para la utilización de las listas y para definir la duración de los eventos.
- Java.io: Proporciona la entrada y la salida del sistema a través de los flujos de datos. En la aplicación se utiliza para recibir los datos de la URL consultada.
- Java.net: Proporciona las clases para la implementación de aplicaciones de red. En el proyecto se utiliza para establecer las conexiones HTTP.

Org.json: Para el uso de las herramientas relacionadas con JSON. En esta aplicación se utilizan JsonObject y JsonArray para trabajar con los datos recibidos en la API anterior.

6 PRUEBAS REALIZADAS

6.1 Pruebas de la Base de Datos

En la fase de planificación de la base de datos no se había definido la estructura final, ya que durante el desarrollo se ha tenido que cambiar algún tipo de los parámetros o incluso alguna de las tablas. Entonces, las pruebas realizadas con esta parte del proyecto corresponden también con las de servicio WEB y las de aplicación Android, ya que esas partes son las que han necesitado realizar los cambios en la base de datos. En los apartados posteriores se verá qué pruebas se habían realizado en dichas partes.

En el principio del desarrollo del proyecto se estaba utilizando el servicio de XAMPP que proporciona el módulo de MySQL, pero conforme se ha ido avanzando en la aplicación se ha necesitado tener la base de datos incorporada en la nube por lo que posteriormente, al crear la aplicación en Openshift, se han agregado los cartridges MySQL y phpmyadmin, que son los necesarios para la definición y la gestión de la base de datos en dicha aplicación.

6.2 Pruebas del Servicio WEB

A continuación se presentan las pruebas realizadas en esta parte del proyecto y son las siguientes:

- Uso de los ejemplos básicos de spring para la creación de un servicio sencillo, que ha servido de apoyo para el desarrollo de esta parte del proyecto.
- El servidor Tomcat que se utilizaba al principio era el que está incorporado en Spring Tool Suite, con lo que todas las consultas se realizaban a la URL http://localhost:8080/* con la que se probaba el funcionamiento.
- En la nube las primeras pruebas fueron realizadas con el servidor Tomcat 7 seleccionado en la creación de la aplicación, pero siempre terminaba dando el error interno 500, ya que las versiones de Java utilizadas eran distintas entre el servidor Tomcat 7 y el servicio WEB desarrollado.
- Se cambió por el servidor Wildfly 10, que funciona perfectamente con el servicio

desarrollado.

- En cuanto al código, primero todas las consultas eran estáticas, pero al ver que las respuestas eran muy largas y que había mucha información innecesaria en una petición se han cambiado dichas consultas integrando los parámetros obtenidos desde la aplicación Android, haciendo que la consulta devuelva solamente los datos solicitados.
- Otra modificación en el código ha sido agrupar todas las clases de los eventos. Estaban separados por tipos, pero como era un código muy similar para todas las clases se ha decidido ponerlas en una clase única, ya que de momento no se especializa ninguna acción que podría necesitar separar dichas clases.
- En principio el correcto funcionamiento del servicio desarrollado se ha ido probando por la URL para ver que las respuestas se obtienen correctamente según la información solicitada.
- Al ver que todo funciona correctamente se ha pasado a utilizar las direcciones URL del servicio en la aplicación Android.

6.3 Pruebas de la Aplicación Android

En este apartado se van a describir las numerosas pruebas realizadas en esta parte del proyecto, ya que a la aplicación Android se ha dedicado la mayor parte del tiempo. Son las siguientes:

- En principio se estaba trabajando con el emulador de Genymotion para ver cualquier cambio realizado en el código y comprobar que todo funciona como es debido.
- Posteriormente se ha utilizado el dispositivo móvil Samsung Galaxy S4 mini para ver como funciona la aplicación en realidad y que , efectivamente no presenta ningún problema respecto al emulador de Genymotion.
- Pruebas de la pantalla de presentación del logo de la aplicación:
 - Pruebas de aparición del icono de la aplicación al iniciarla.
 - Control del tiempo de esta actividad para que no sea muy larga y el cliente no tenga que esperar mucho para pasar a la pantalla principal.
 - Ajuste de los colores y los tamaños de los elementos.
- Pruebas de la pantalla principal:

- Diseño de la lista de los tipos de locales.
- Incorporación del slide de imágenes y gestionar el correcto funcionamiento.
- Control del tiempo de aparición de las imágenes en el slide.
- Comprobación de que todos los elementos, tanto la lista como los botones agregados, al clicar realizan alguna acción.
- Comprobar que el parámetro que se pasa al clicar en un elemento de la lista es el correcto.
- En el caso de que no se vean todos los elementos conseguir que todo el contenido de esta pantalla sea deslizable, para poder desplazar la pantalla.
- Ajustar los tamaños y los colores de los elementos presentes.
- Pruebas de los botones “Me gusta” y “Contáctanos”
 - Comprobación de que se comparte el contenido en el perfil de Facebook del usuario al darle el clic sobre el botón “Me gusta”.
 - No se puede dar al botón “Me gusta” si no está instalada la aplicación de Facebook en el dispositivo utilizado. El botón aparece en color gris.
 - Comprobación de que se muestra la información de contacto al clicar sobre el botón “Contáctanos”.
- Pruebas realizadas con la pantalla que tiene dos fragmentos, “Locales” y “Mapa”:
 - Agregación de los fragmentos en el submenú. Comprobación de que no se produce ningún error al cambiar de un fragmento al otro.
 - Obtención de los datos correctos en el fragmento “Locales” según el tipo de locales que se ha seleccionado.
 - Comprobación de la información que aparece en la lista de los locales consultados.
 - Probar que todos los elementos de la lista son seleccionables y que al clicar en uno se inicia otra actividad.
 - Al clicar sobre la imagen dentro de la lista de los locales se inicia otra actividad

distinta. Solo el elemento seleccionado debe aparecer en el mapa que sale en esta actividad.

- En el fragmento “Mapa” comprobar que aparece el mapa agregado a la aplicación. Si no hay ningún local de los buscados el mapa mostrará una visión muy alejada mostrando casi toda la Tierra. En el caso de que hay locales del tipo que se consulta el mapa se enfocará en el centro de la ciudad, que en este caso es Sevilla y ver como aparecen los locales marcados. También comprobar como aparece la localización del usuario si se encuentra en la ciudad.
- Comprobar que los parámetros al clicar en un elemento de la lista son pasados correctamente.
- Ajustar los tamaños y los colores de los elementos presentes en ambos fragmentos.
- Pruebas realizadas con la actividad que muestra la información de un local.
 - Comprobar el correcto funcionamiento del slide de imágenes incorporado. Las imágenes no pueden ser más grandes que 750x500 , si no se desconfigura el ajuste de la pantalla.
 - Comprobar que los parámetros recibidos contienen la información correcta. Son datos del local que se ha elegido.
 - La página web del local ponerla como un link y probar que funciona al clicar sobre él.
 - Probar el funcionamiento de los dos botones agregados a esta actividad. Cuando se clicca en cada uno se tiene que ir a las actividades correspondientes en cada caso.
 - Ajuste de los colores y los tamaños de esta actividad.
- Pruebas de la pantalla que tiene la lista de los eventos de un local.
 - Comprobar que la consulta al servidor obtiene los datos esperados.
 - Obtener la lista ordenada por fecha.
 - Comprobar que todos los elementos de la lista al realizar un clic llevan a otra actividad.
 - Probar que los parámetros que se pasan al realizar un clic en un elemento de la lista son los correspondientes al evento seleccionado.
 - Ajustar los colores y los tamaños de los elementos de esta actividad.

- Pruebas de la actividad que contiene la información de un evento.
 - Comprobar que la información que aparece corresponde al evento seleccionado.
 - El elemento “Descripción” de la lista lleva a otra actividad, en la que solamente se muestra la descripción detallada.
 - Ajustar los colores y los tamaños de los elementos de esta pantalla.

7 PROBLEMAS ENCONTRADOS

A lo largo de todo el desarrollo del proyecto se han encontrado numerosos problemas debido a la falta de conocimientos de estas tecnologías. Unos problemas han sido menos significantes, otros más y algunos no se han podido solucionar por lo que se ha tenido que buscar otro camino para obtener una solución. En este apartado se van a describir los problemas principales encontrados durante la realización del proyecto. Son los siguientes:

- **Tipo DATETIME en la base de datos.** En un principio en la tabla de eventos de la base de datos la fecha y la hora estaban en un campo, pero al rellenar la información y a la hora de consultarla por la aplicación Android se ha tenido muchos problemas, ya que no es un tipo fácil de manejar. Este campo se ha cambiado por dos campos que separan la fecha de la hora de un evento, con lo que se han solucionado todas las dificultades que se han tenido con el tipo anterior.
- **Selección del servidor en Spring Tool Suite.** Spring Tool Suite tiene en su servicio muchos tipos de servidores que pueden ser creados, pero ¿cuál es el que le viene bien al servidor desarrollado? Se han probado muchos de ellos hasta conseguir el resultado esperado. Con el servidor Apache Tomcat 7 el servidor funciona perfectamente.
- **Servidor de la aplicación de Openshift.** Se ha perdido mucho tiempo con este problema, ya que al elegir el mismo servidor para la aplicación de Openshift que el usado en el propio sistema el servicio siempre devolvía un error. Muchas de las veces se recibía el error 404, ya que no se tenía el conocimiento suficiente de qué URL debería ser usada al tener el servicio subido a la nube. Otras veces se recibía el error interno 500, que significa que el servicio no es compatible con el servidor de Openshift, por lo que no puede mostrar el contenido en la URL. Todo eso es debido a que el servidor de la aplicación y el servicio del proyecto tenían versiones distintas de Java. Después se cambió el servidor Tomcat 7 por el Wildfly 10, que es el que utiliza la misma versión de Java que el servicio desarrollado solucionando el problema que había anteriormente.

- **Servidor de Openshift no disponible.** A veces el servicio de Openshift no estaba disponible, ya que la versión utilizada es totalmente gratuita con lo que los accesos son limitados. Al realizar muchas pruebas con la aplicación Android se accedía muchas veces al servidor, por lo que se llegaba a obtener el error 500. Hay que esperar un tiempo para obtener más accesos al servicio. Otro factor que influye en el funcionamiento del servidor es el tiempo que está funcionando. A veces después de un día entero para de funcionar. Entonces, hay que entrar en la consola de Openshift, donde están todas las aplicaciones creadas del usuario, y rearrancar el servicio.

- **Numerosos programas corriendo en el sistema al mismo tiempo.** Cuando el servicio no estaba subido en la nube para trabajar con el proyecto se estaban utilizando:
 - XAMPP
 - Spring Tool Suite
 - Android Studio
 - Genymotion

Todas estas herramientas tenían que estar funcionando para hacer posible realizar las pruebas del proyecto completo. Todos los procesos involucrados ocupaban gran cantidad de la memoria RAM, por lo que el sistema se ralentizaba mucho a pesar de que dispone de 8 GB de esta memoria. Con la subida a la nube del servicio y utilizando el dispositivo móvil Samsung Galaxy s4 mini, que estaba conectado por el puerto USB con el sistema, se ha conseguido que solamente en el sistema esté corriendo Android Studio, lo que resolvía el problema de ralentización del trabajo por el ordenador empleado en esta tarea.

- **Problemas con el diseño de las actividades con abundantes elementos.** No es que sea un problema grande, pero se ha empleado mucho tiempo en ajustar las pantallas de las actividades con muchos elementos, por ejemplo, la pantalla principal. Se han hecho muchas modificaciones del contenido en el layout que pertenece a esta actividad. Lo que era más complicado en esta tarea es que se consiga rotar todo el contenido de la actividad , teniendo una lista, un slide de imágenes y los dos botones. Después de realizar muchas pruebas se ha conseguido el resultado esperado con un layout final complicado de implementar.

- **Problema del GPS incorporado en Genymotion.** A la hora de realizar las pruebas con las actividades que contienen el mapa no funciona bien el GPS del dispositivo, por lo que hay que configurar los parámetros de donde se encuentra el usuario manualmente. Solo es para ver que realmente funciona. En el dispositivo móvil Samsung Galaxy S4 mini la localización del usuario se encuentra automáticamente.
- **Share content de Facebook.** En un principio la idea era compartir la información acerca de un local o compartir algún evento programado al perfil público de Facebook de un usuario, pero al realizar muchas pruebas no se ha conseguido que funcione correctamente, por lo que se ha añadido el botón de me gusta que comparte la información que aparece en una página HTML, que contiene alguna información de la aplicación.

8 CONCLUSIONES

En este capítulo se va a hablar de los objetivos conseguidos y los conocimientos obtenidos durante el desarrollo de este trabajo.

En este trabajo fin de grado se ha desarrollado un sistema que es capaz de informar a sus usuarios acerca de los eventos existentes en diferentes tipos de locales en la ciudad donde se encuentran.

Al hacer numerosas pruebas a la aplicación se ha llegado a la conclusión de que los objetivos propuestos en la introducción se cumplen, pero el trabajo no está finalizado porque le faltan algunos detalles. Lo importante es que se ha conseguido que funcionen las tres partes del proyecto como un sistema único y un usuario puede consultar los locales disponibles y sus eventos programados. Esto es el objetivo principal de este trabajo.

En cuanto a la experiencia obtenida cabe destacar los siguientes puntos:

- Se ha conseguido que el servidor haga las consultas con los parámetros para escoger solamente la información pedida por el usuario.
- Se ha aprendido subir un servicio web a la nube y ponerlo en marcha, que es una de las partes más interesantes, ya que va a servir bastante para el futuro y no debe ser olvidada.
- Se ha mejorado bastante el uso del lenguaje de programación Java, tanto para Android como para el servidor desarrollados.
- Se han repasado las consultas de MySQL.
- Se ha aprendido a controlar el estilo de todos los elementos que aparecen en la pantalla de una actividad.
- Se han conocido muchas herramientas nuevas para el desarrollo de este proyecto. Entre las cuales podemos destacar:

- XAMPP
- Genymotion
- Spring Tool Suite
- Android Studio
- Servicio de Openshift

Se espera que todo el conocimiento obtenido durante el desarrollo sea útil para el futuro. Si el futuro trabajo está relacionado con estas tecnologías será un gran plus, ya que se tiene un nivel un poco más que básico de experiencia desarrollando aplicaciones Android, creando una base de datos y programando un servidor.

Pero lo mejor que podría ocurrir es que, al terminar de desarrollar la aplicación, se suba a Playstore, y que sea utilizada por los usuarios para ver si está siendo útil. Todo esto implicaría un gran trabajo hasta conseguirlo, ya que no solo está el código de las partes del proyecto, sino que posteriormente hay que ofrecer la aplicación a los clientes que van a contratar el servicio para su local.

La única manera de que la aplicación tenga éxito es que sea realmente imprescindible. Tiene que ser algo único, algo tan útil que sería utilizado por muchísimos usuarios, aunque eso sería a largo plazo, ya que todas las aplicaciones no han tenido el éxito al inicio, sino que han tardado un tiempo en ganar sus usuarios.

9 DE CARA AL FUTURO

En este capítulo se va a hablar acerca de lo que se quiere conseguir en el futuro. Cómo será la aplicación terminada para poder subirla a Playstore y que esté disponible para los usuarios.

Las modificaciones necesarias son:

- **Sustitución de las imágenes estáticas en los slide por las imágenes obtenidas de la consulta de la base de datos.** Este cambio implica modificaciones de las tres partes del proyecto. En la base de datos habrá que crear la tabla correspondiente a las imágenes y en el servicio habrá que agregar nuevas consultas a la base de datos. Se haría uso de las imágenes en las actividades de la aplicación Android que contienen los slide.
- **Agregación del share content para Facebook y otras redes sociales.** Hay que agregar los botones de compartir la información en distintas redes sociales para conseguir la máxima expansión de la aplicación que podría llevar al éxito.
- **Agregación de las notificaciones.** Añadir una opción para que los usuarios puedan inscribirse a los eventos de un local, entonces, recibirían las notificaciones acerca de los eventos que tendrán lugar en dicho local.
- **Consultar a varios usuarios su opinión acerca de la aplicación.** Es importante saber lo que opina la gente ya que son los que van a instalar dicha aplicación en su dispositivo móvil. Entonces, al realizar esta tarea, puede que haya que modificar un poco el contenido e incluso el funcionamiento del proyecto.
- **Subir el servicio a la nube de pago,** por ejemplo, a Amazon que ofrece unas características muy favorables y no es muy caro.
- **Subir la aplicación a PlayStore para que esté disponible para todos los usuarios.** Para conseguir la información en la aplicación habrá que ofrecer el servicio a los dueños de los locales.

- **Involucrar otras ciudad.** De momento la aplicación se ha pensado para los locales que se encuentran en Sevilla. Hay que realizar estudios acerca de si se esta siendo útil la aplicación. Una vez obtenidos los resultados quizás hay que plantear la opción de ir a otra ciudad y conseguir más usuarios y, con ello, más beneficios.
- **Numerosas aplicaciones instaladas.** Uno de los grandes problemas que se podría encontrar una vez que la aplicación esté en Playstore es que los usuarios no quieran descargarla por el motivo de que ya existen numerosas aplicaciones en sus dispositivos móviles y que no quieren instalar más. Entonces, habría que pensar algún tipo de beneficio que obtendría el usuario por utilizar la aplicación. Quizás algún descuento o alguna oferta especial que podría proporcionar un local a sus clientes. Esto podría beneficiar a todas las partes:
 - Usuarios obtendría sus descuentos o aprovecharían algún tipo de oferta.
 - El local generaría más beneficios.
 - La aplicación sería utilizada por más usuarios, con lo que los locales mantendrían el servicio contratado.

10 BIBLIOGRAFÍA

En este capítulo se van a poner todas las páginas que han sido útiles para recibir la información acerca del desarrollo de las tres partes del proyecto.

Android:

<https://developer.android.com/index.html>

https://www.android.com/intl/es_es/

<https://es.wikipedia.org/wiki/Android>

<http://stackoverflow.com/>

<http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=150&inicio=>

<http://smallbusiness.chron.com/start-new-activity-parameters-android-33251.html>

<http://www.androidcurso.com/index.php/tutoriales-android/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/114-layouts>

<http://www.sgoliver.net/blog/interfaz-de-usuario-en-android-layouts/>

https://romannurik.github.io/AndroidAssetStudio/icons-generic.html#source.space.trim=1&source.space.pad=0&size=24&padding=8&color=33b5e5%2C100&name=ic_example

<http://androidopentutorials.com/android-image-slideshow-using-viewpager/>

<http://javapapers.com/android/android-image-slider-tutorial/>

<http://kb4dev.com/article/android-image-slider-with-swipe-gesture>

<https://developers.google.com/maps/documentation/android-api/?hl=es>

<https://elbauldelprogramador.com/programacion-android-intents-conceptos/>

<http://www.arquitecturajava.com/el-concepto-de-android-intent/>

<http://www.vogella.com/tutorials/AndroidFragments/article.html>

http://www.tutorialspoint.com/android/android_fragments.htm

https://github.com/codepath/android_guides/wiki/Creating-and-Using-Fragments

<http://simpledeveloper.com/how-to-share-an-image-on-facebook-in-android/>

<https://inthecheesefactory.com/blog/how-to-add-facebook-like-button-in-android-app/en>

Servicio WEB REST:

<http://www.mkyong.com/webservices/jax-rs/jax-rs-queryparam-example/>

<http://stackoverflow.com/>

<http://www.concretepage.com/spring/spring-mvc/spring-mvc-requestmapping-annotation-example-with-value-method-headers-params-consumes-produces-requestparam-pathvariable>

<http://www.baeldung.com/spring-requestmapping>

<http://docs.spring.io/spring/docs/current/javadoc-api/org.springframework.web.bind.annotation.RequestMapping.html>

<http://www.studytrails.com/frameworks/spring/spring-mvc-controller-input.jsp>

<http://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example>

http://www.tutorialspoint.com/spring/spring_hello_world_example.htm

<https://spring.io/guides>

<https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-spring-config.html>

<https://www.openshift.com/>

Base de datos MySql:

<https://www.mysql.com>

<http://stackoverflow.com/q>

<http://dba.stackexchange.com/questions/6310/how-to-show-the-column-names-of-a-table-with-a-simple-sql-query>

<https://es.wikipedia.org/wiki/MySQL>

MANUAL DE USUARIO

En esta parte de la memoria se va a explicar todo el funcionamiento de la aplicación Android. Todas las acciones que pueden ser realizadas por el usuario. Se describirá la acción que hay que realizar y se mostrará el resultado correspondiente en las distintas figuras.

Las acciones que se pueden realizar son las que incluye la figura 38, que es la que muestra los casos de uso de la aplicación Android. A continuación se describen una por una.

- **Primer contacto.** El usuario dará un clic sobre la aplicación instalada en su dispositivo y lo que va a ver a continuación es, primero, una animación del logo de la aplicación, que se muestra en la figura 57 y, posteriormente, la pantalla principal de la aplicación, que se puede visualizar en la figura 58.



Figura 57. Inicio de la aplicación

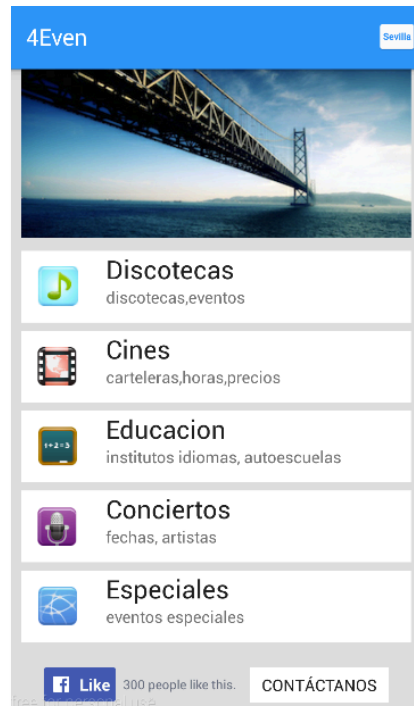


Figura 58. Pantalla principal

- Selección del botón “Like” de Facebook.** Esta acción lo que realizará es compartir una página de la aplicación desarrollada con HTML en el perfil de Facebook del usuario. Una vez se le da al botón se pide la confirmación. Al confirmar se vuelve otra vez a la pantalla de inicio y en el perfil de Facebook se puede ver la información compartida a través de la aplicación. A continuación en la figura 59 se muestra la pantalla de la confirmación de que se ha dado a me gusta. Lo que aparece en el Facebook se puede ver en las figura 52 y, posteriormente, en la figura 53 se muestra la aparición de la cantidad de personas que también han compartido ese contenido.

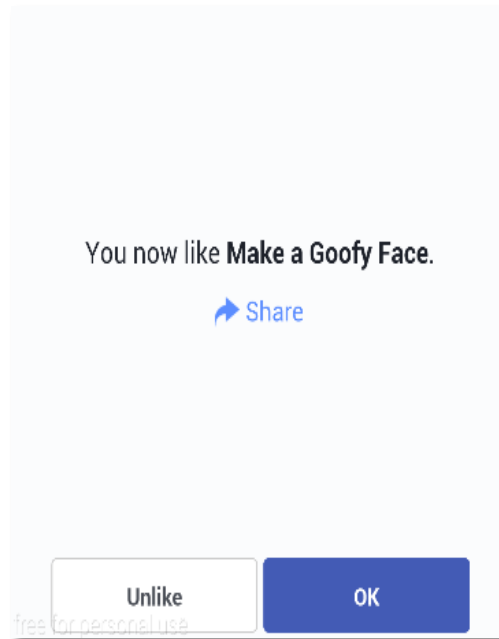


Figura 59. Confirmación de “Me gusta”

- **Selección del botón “Contáctanos”**. Al realizar esta acción lo que el usuario va a obtener es la información para contactar con el desarrollador. Esta actividad se muestra a continuación en la figura 60.



Figura 60. Información de contacto

- **Tipos de locales consultados.** Bueno, en esta acción el usuario se encuentra en la pantalla principal y selecciona un elemento de la lista según a qué eventos quiere asistir. En este ejemplo se mostrará el contenido que se obtiene al elegir la opción “Discotecas”. A continuación en la figura 61 se muestra la lista de los locales disponibles para estos tipos de eventos. Como se puede ver aparte de la lista de los locales también hay otro apartado, que es “Mapa”. En la figura 62 se puede observar esta parte de la aplicación que muestra la localización de los locales anteriores en Sevilla.

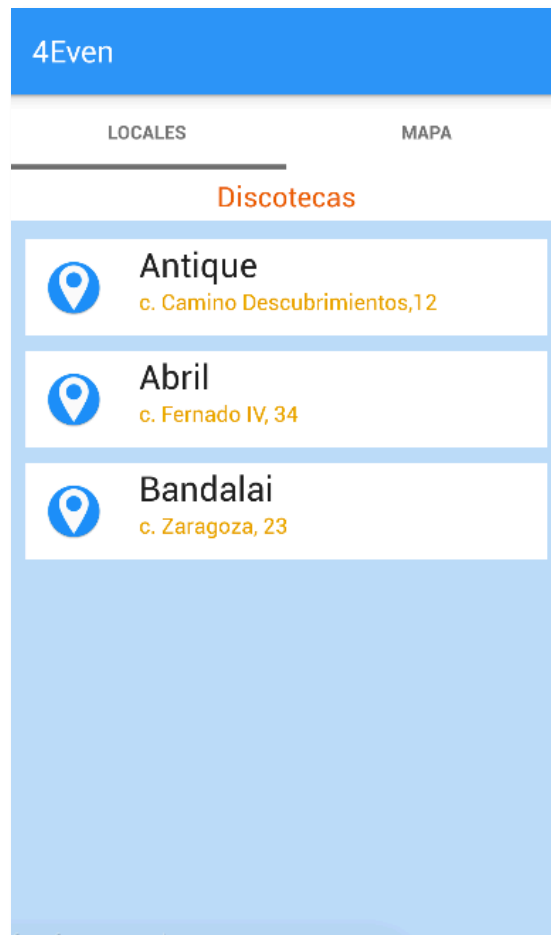


Figura 61. Discotecas disponibles a consultar



Figura 62. Localización de las discotecas

- **Información de un local.** El usuario se encuentra en la pantalla donde está la lista de los locales disponibles y selecciona un local. Lo que ve es la información y las fotos de dicho local y, además tiene para utilizar dos botones, que serán explicados posteriormente. En la figura 63 se muestra esta pantalla.

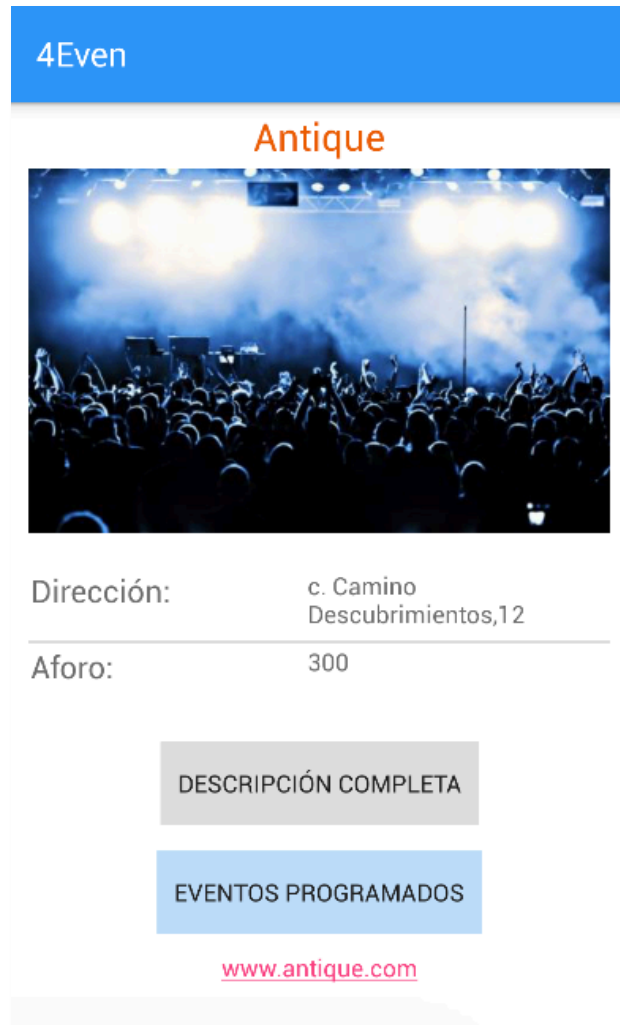


Figura 63. Información de un local

- **Botón “Descripción completa”**. El usuario se encuentra en la actividad donde se muestra la información de un local y pulsa el botón donde pone “Descripción completa”. Se inicia una nueva actividad que muestra todos los datos que el contratante del servicio haya querido poner como adicionales para describir mejor su local. En la figura 50 se muestra esa pantalla.

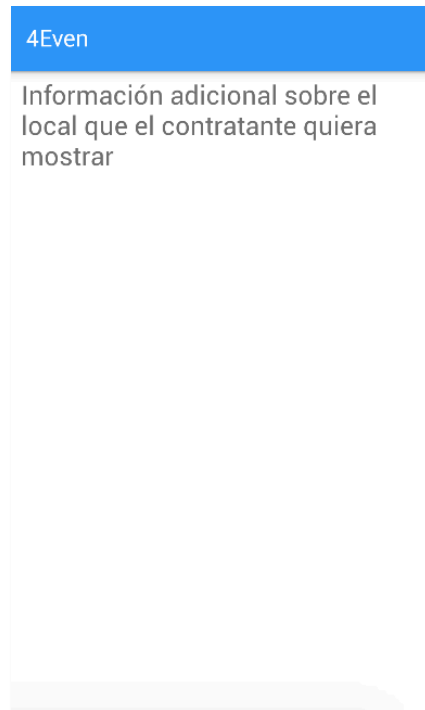


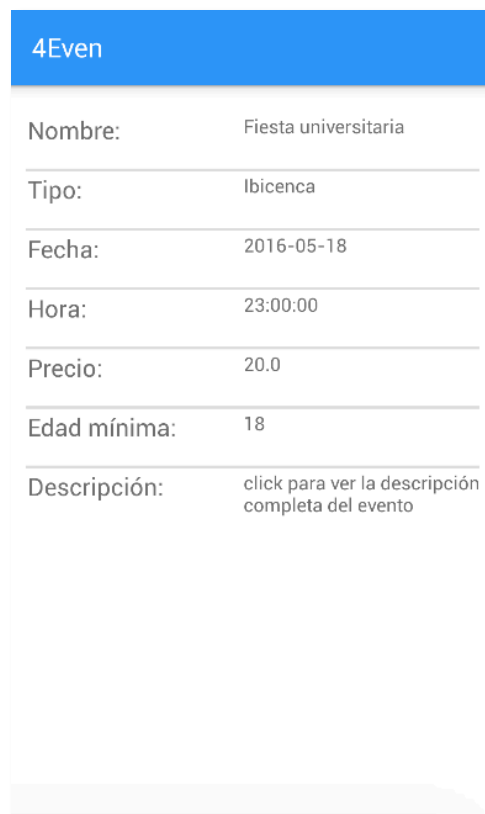
Figura 64. Información adicional de un local

- **Botón “Eventos programados”.** El usuario se encuentra en la actividad que muestra la información y las fotos del local y realiza un clic sobre el botón “Eventos programados”. Esa acción le lleva a otra actividad que muestra los eventos programados de dicho local. La información se muestra en una lista. En la figura 65 se muestra esta pantalla.



Figura 65. Eventos programados de un local

- **Información de un evento.** El usuario está en la pantalla donde se muestran todos los eventos disponibles. Si está interesado en alguno tiene que dar un clic sobre el elemento de la lista que le interesa. Esa acción inicia una nueva actividad que muestra la información del evento seleccionado. A continuación en la figura 66 se muestra la pantalla que contiene la información de un evento.



4Even	
Nombre:	Fiesta universitaria
Tipo:	Ibicenca
Fecha:	2016-05-18
Hora:	23:00:00
Precio:	20.0
Edad mínima:	18
Descripción:	click para ver la descripción completa del evento

Figura 66. Información de un evento

- **Descripción detallada del evento seleccionado.** El usuario está en la pantalla de la información de un evento. En la lista que aparece el usuario puede darle clic sobre el elemento “Descripción” para ver la información adicional acerca del evento que le interesa. En la figura 67 se muestra la pantalla que se inicia.

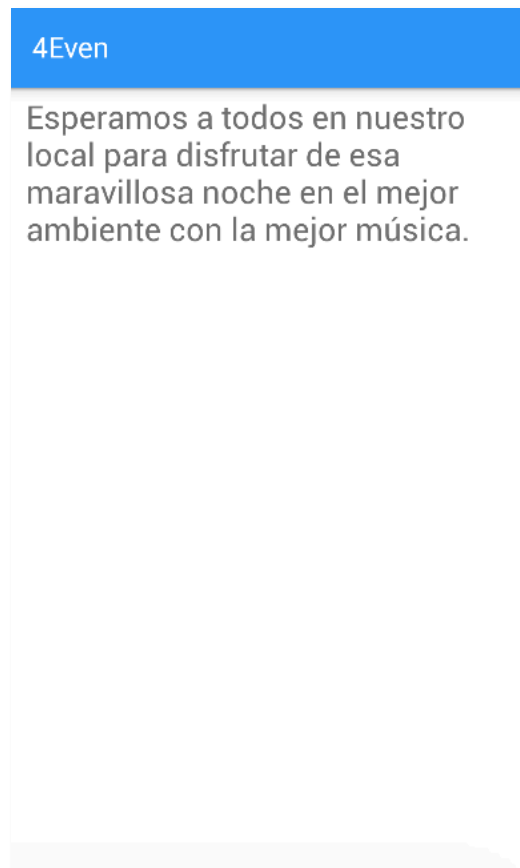


Figura 67. Información adicional de un evento

Estas son todas las acciones que se pueden realizar utilizando la aplicación Android. En el futuro se agregarán más funcionalidades, con lo que se modificará el manual de usuario.

MANUAL DE INSTALACIÓN

En este capítulo se van a mostrar los pasos que hay que seguir para instalar las herramientas necesarias utilizadas en este trabajo. Los procedimientos que se van a exponer son específicamente para el sistema operativo utilizado, Mac OS X Capitan.

Instalación de XAMPP.

La página web de donde se descarga el archivo con formato .dmg es

<https://www.apachefriends.org/es/index.html>. Hay que clicar sobre “XAMPP para OS X v5.6.21”. Este paso se muestra en la figura 68.



Figura 68. Elección del instalador de XAMPP

Abrir el archivo `xampp-osx—5.6.21-0-installer.dmg`. En la figura 69 se muestra el contenido de este archivo.



Figura 69. Contenido del archivo .dmg de XAMPP

Se da doble clic sobre el elemento que aparece en la figura 69 y aparece el instalador. A continuación, en las figuras 70, 71 y 72 se pueden ver las pantallas del instalador mostradas al usuario hasta completar la instalación de XAMPP. Para completar la instalación hay que pulsar el botón “next” en cada pantalla.

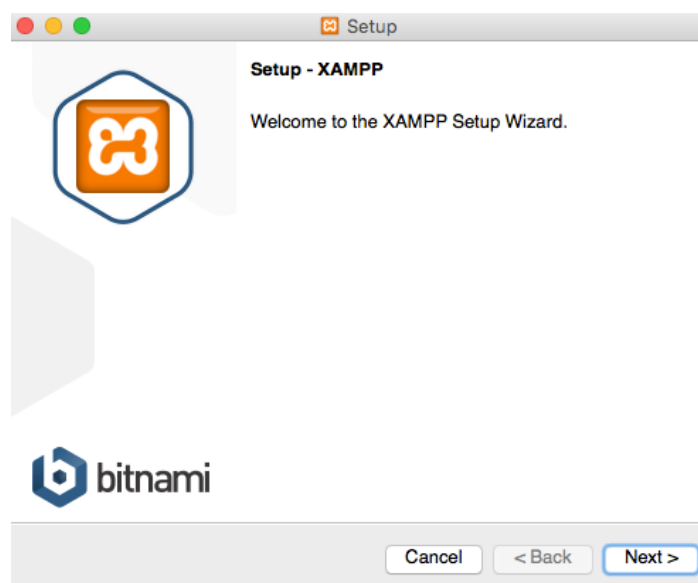


Figura 70. Pantalla 1 del instalador de XAMPP

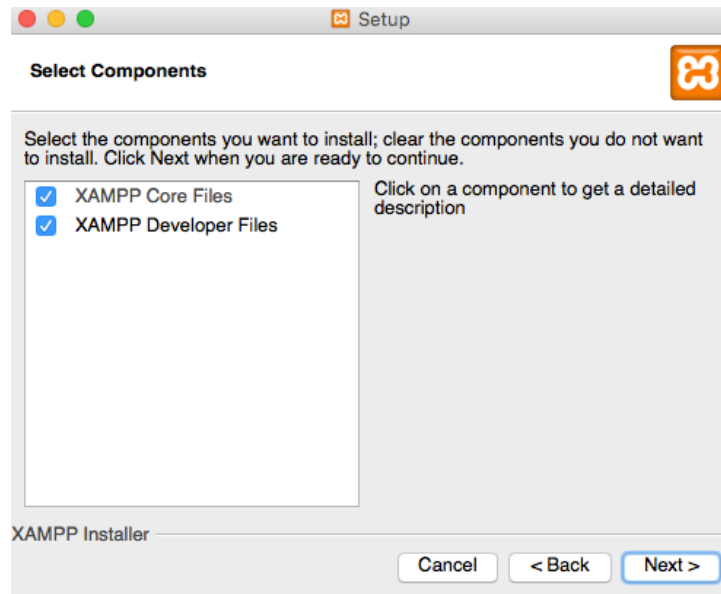


Figura 71. Pantalla 2 del instalador de XAMPP

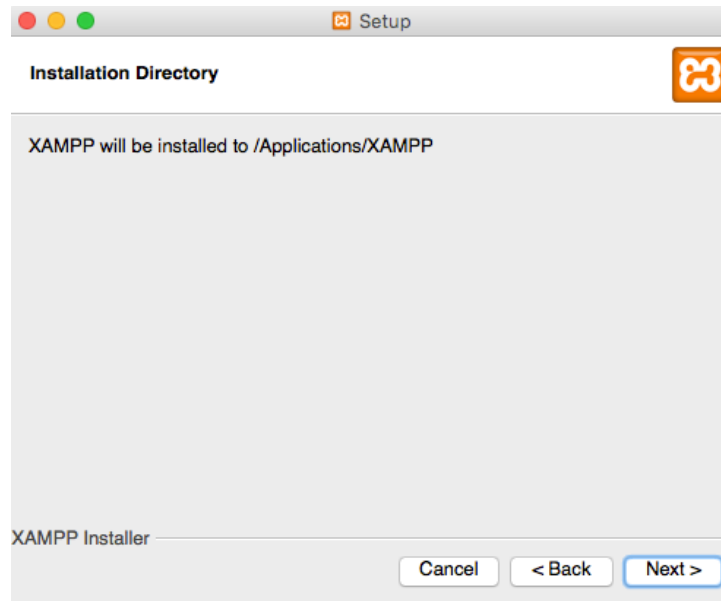


Figura 72. Pantalla 3 del instalador de XAMPP

Instalación de Spring Tool Suite.

La página web de donde se descarga el instalador de STS es <https://spring.io/tools>. Se clica en el botón “DOWNLOAD STS (3.7.3.RELEASE for Mac). En la figura 73 se muestra esta pantalla.

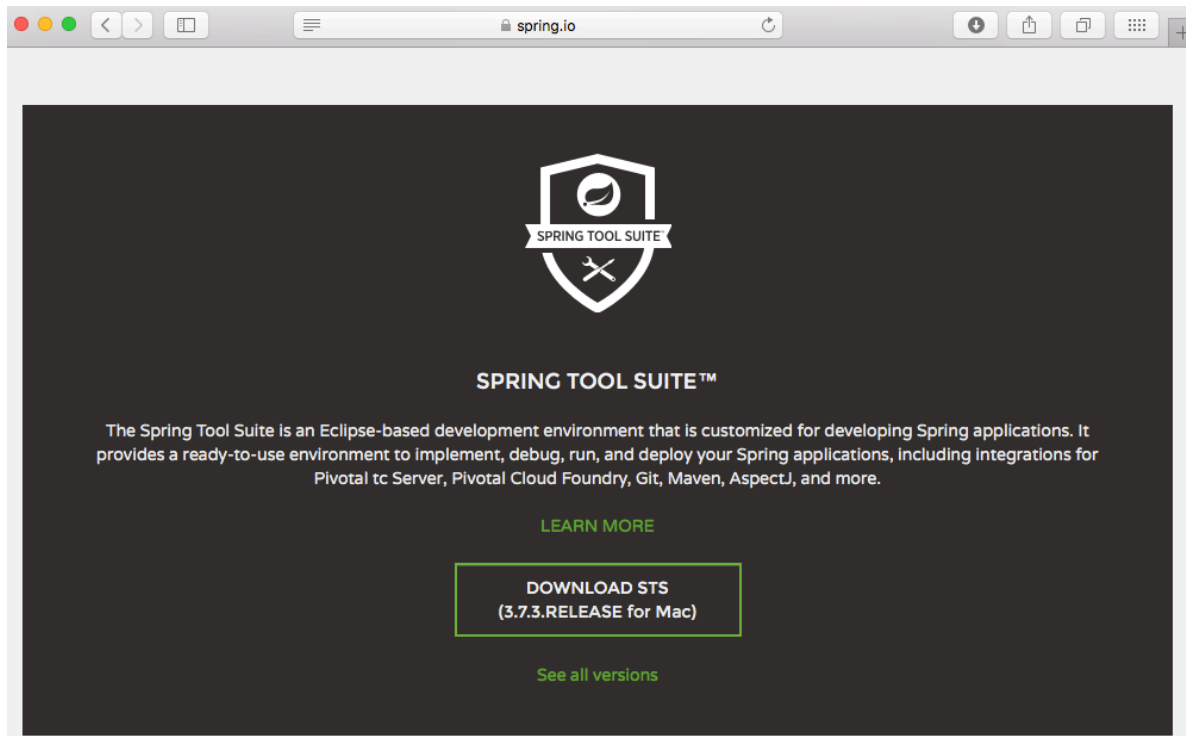


Figura 73. Página web de Spring Tool Suite

Se descarga el archivo .tar.gz y se descomprime en cualquier directorio del sistema. Se abre la carpeta descomprimida “sts-bundle” y se clica dos veces en el logo de Spring. En la figura 74 se muestra dicha carpeta con el contenido correspondiente.

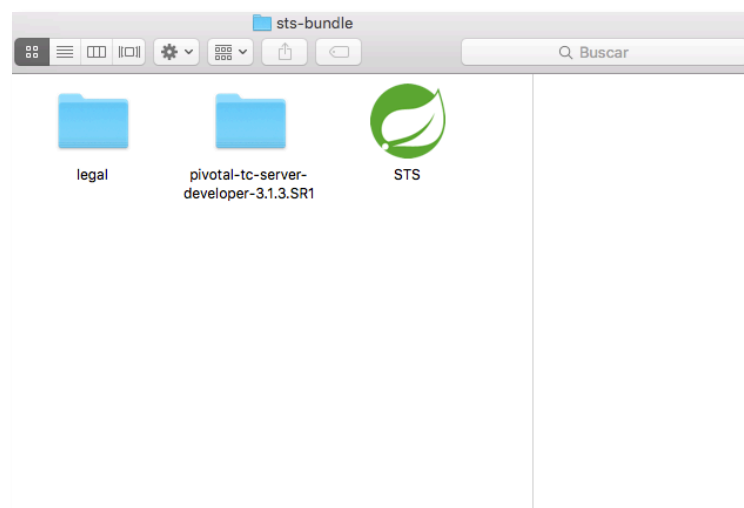


Figura 74. Carpeta de Spring Tool Suite

Al realizar el doble clic en el logo de Spring aparece la pantalla de selección del espacio de trabajo, que indica el directorio donde se quiere guardar los proyectos que se van a realizar y se clica en “OK” . Esto se muestra en la figura 75.

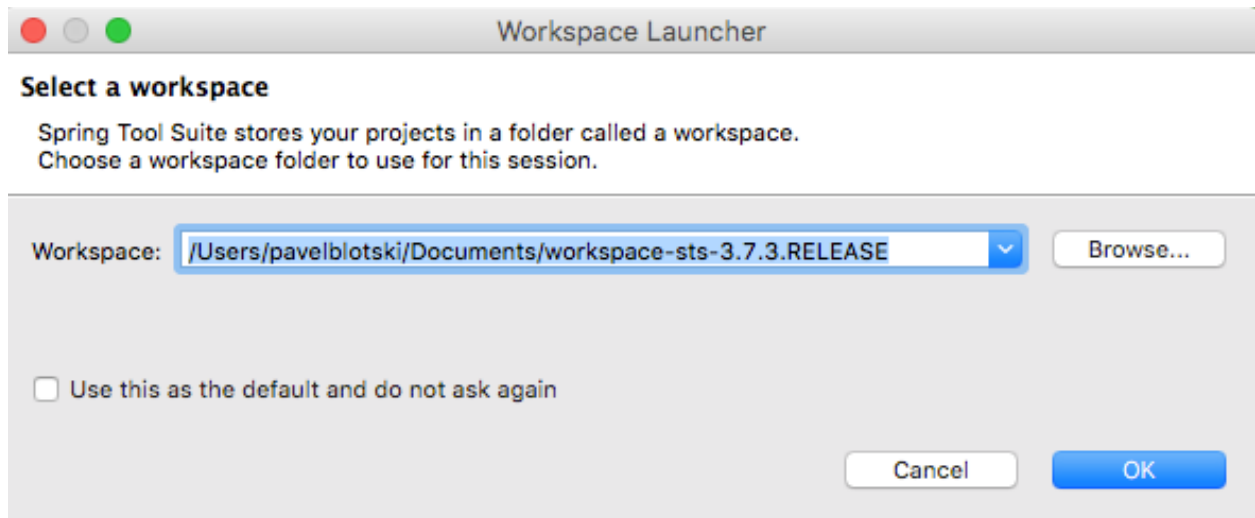


Figura 75. Selección del espacio de trabajo de Spring Tool Suite

Instalación de Android Studio.

Se descarga el archivo .dmg de la página web

<https://developer.android.com/studio/index.html>. Primero se clica en el botón de descargar el software. En la pantalla siguiente se aceptan los términos y las condiciones de uso y se procede a la descarga de Android Studio. Estas dos pantallas se pueden ver en las figuras 76 y 77.

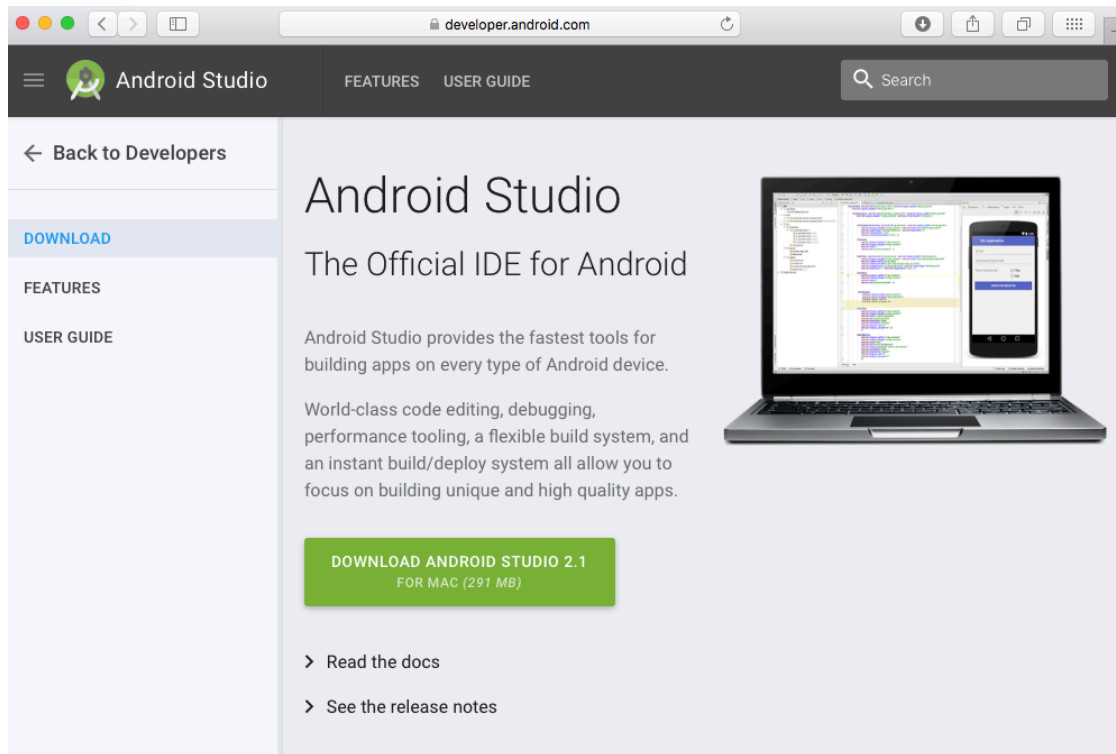


Figura 76. Pantalla 1 de la descarga de Android Studio

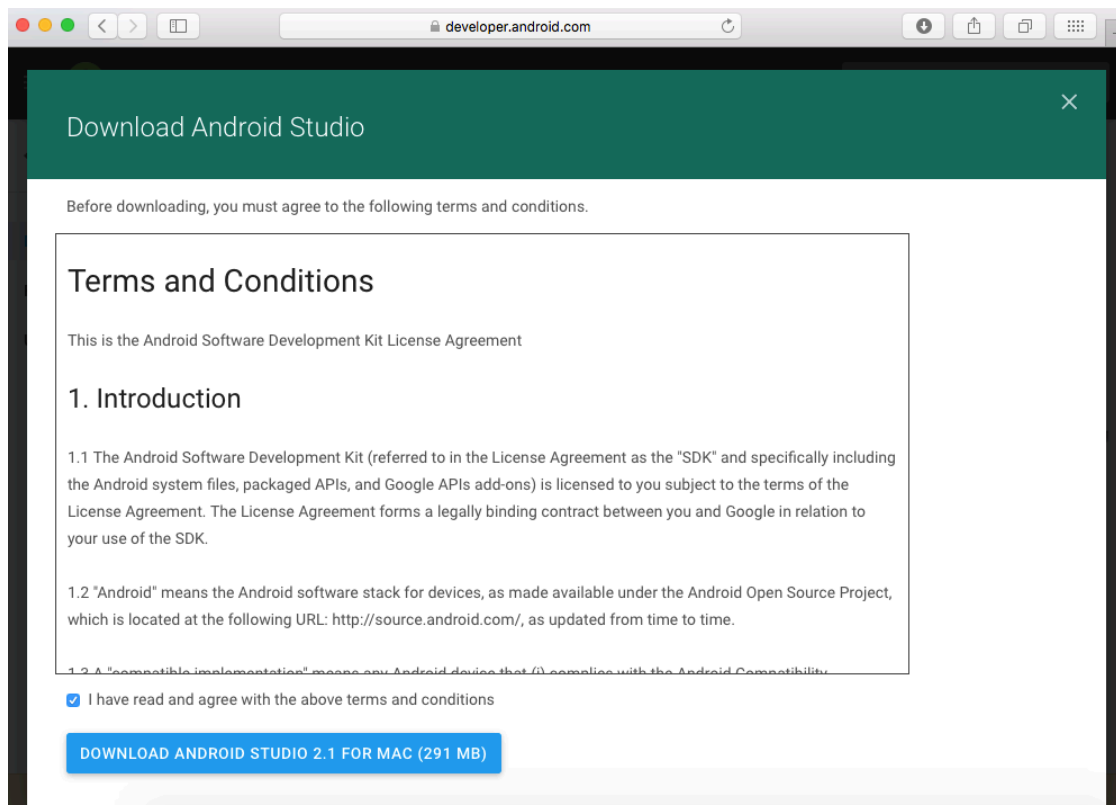


Figura 77. Pantalla 2 de la descarga de Android Studio

Una vez descargado el archivo .dmg se abre. En la pantalla que aparece a continuación hay que arrastrar el logo de Android Studio a la carpeta “Applications”, que es el directorio donde están todos los programas instalados en el sistema Mac OS X. En la figura 78 se muestra esta pantalla.



Figura 78. Instalador de Android Studio

Instalación de Genymotion.

Se va a la página <https://www.genymotion.com/features/> y al final de la misma se clica en el botón “Get started”. Esta pantalla se muestra en la figura 79.

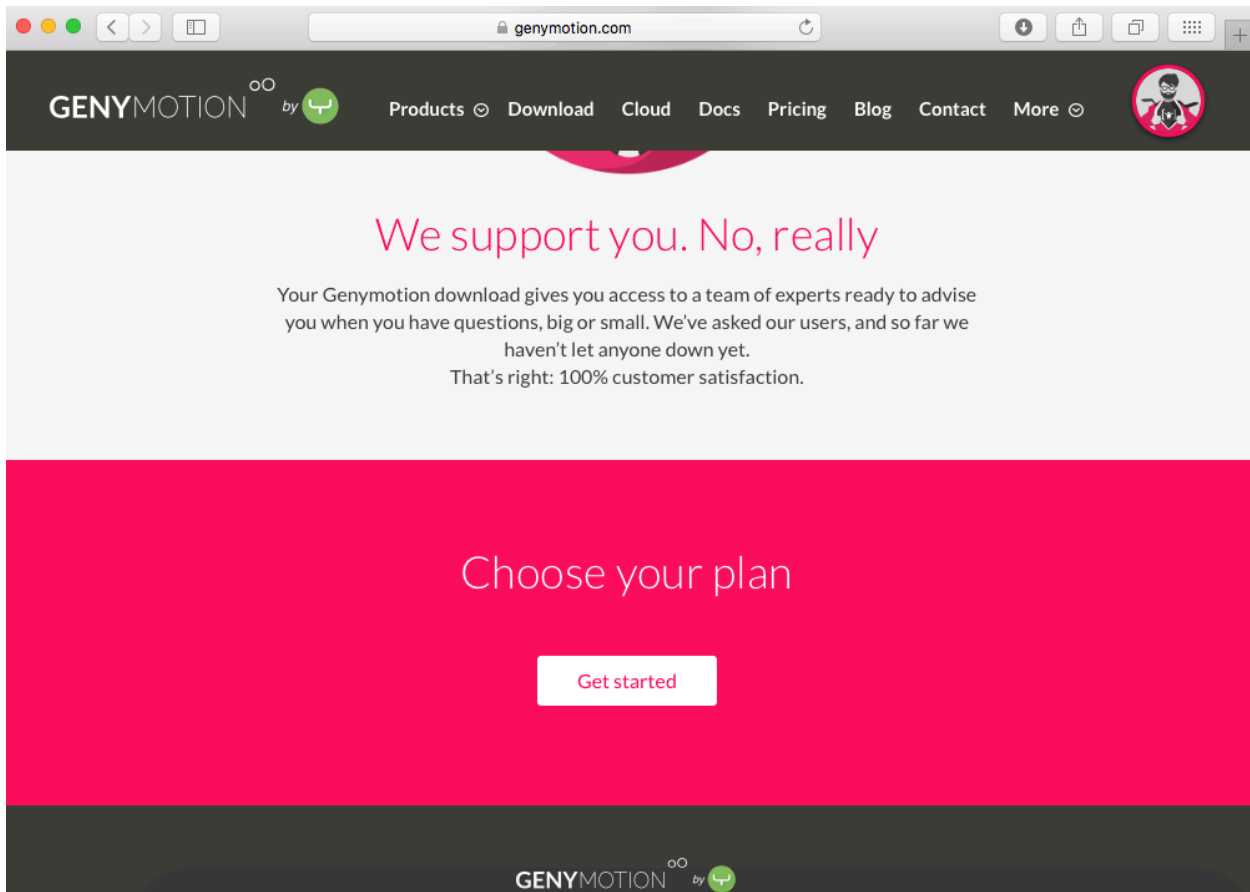


Figura 79. Pantalla 1 de la página web de Genymotion

Después de clicar en el botón de la figura anterior aparece la siguiente pantalla donde se elige el apartado “individual” y, posteriormente, se clica en el botón “Get started” del elemento “BASIC”, ya que es la versión gratuita del software. Esta pantalla se muestra en la figura 80.

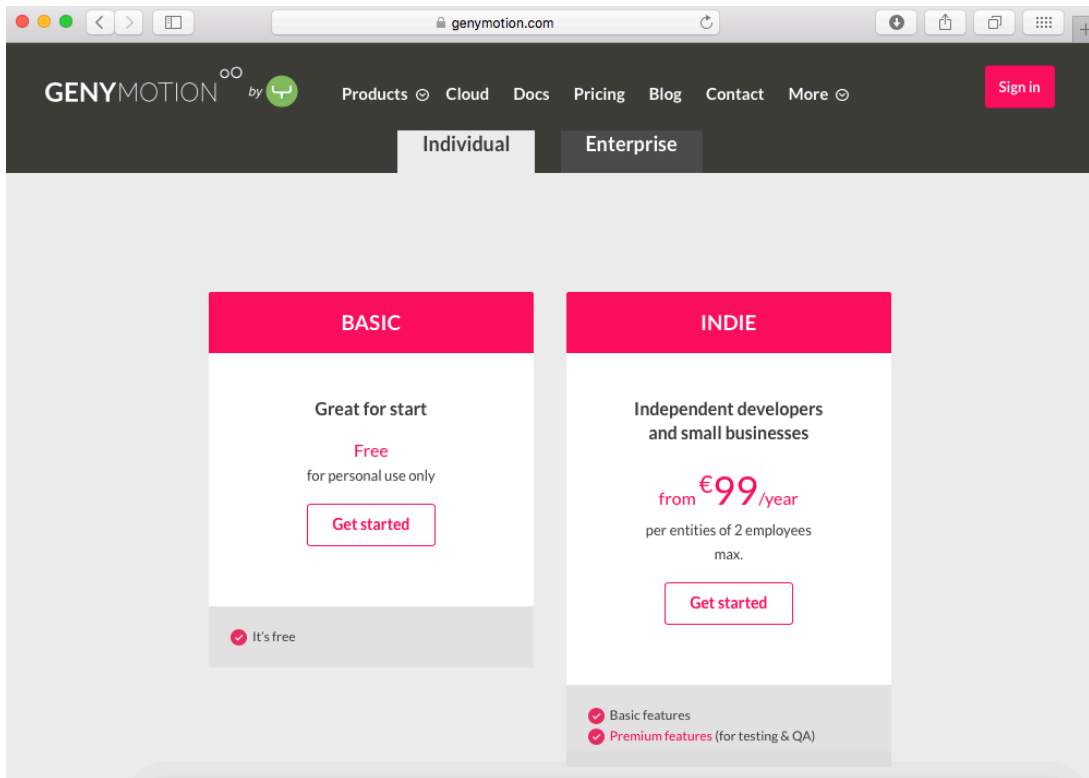


Figura 80. Pantalla 2 de la página web de Genymotion

Al pulsar en el botón “Get started” del elemento “BASIC”, que se muestra en la figura anterior, aparece otra pantalla donde se clic en el botón “Download Genymotion package”. Esta pantalla se muestra en la figura 81.

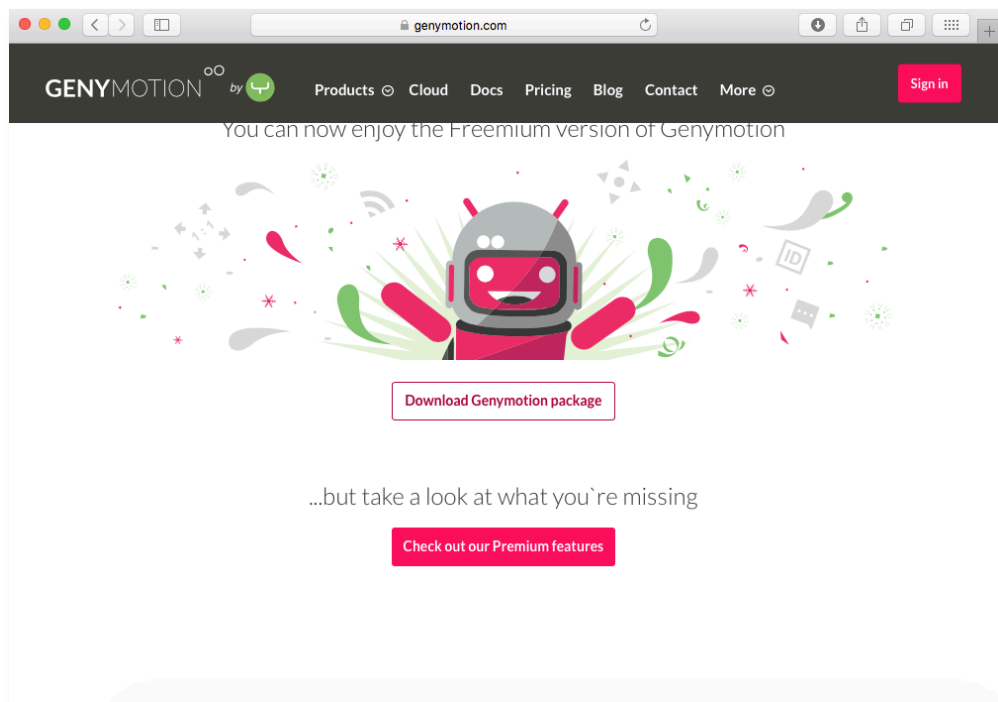


Figura 81. Pantalla 3 de la página web de Genymotion

Una vez se haya clicado en el botón “Download Genymotion package” aparece otra pantalla donde se indica el sistema operativo para el que se va a descargar este software. Se da clic en el botón “Download for Mac OSX” y se procede a la descarga del archivo .dmg. Esta pantalla se muestra en la figura 82.

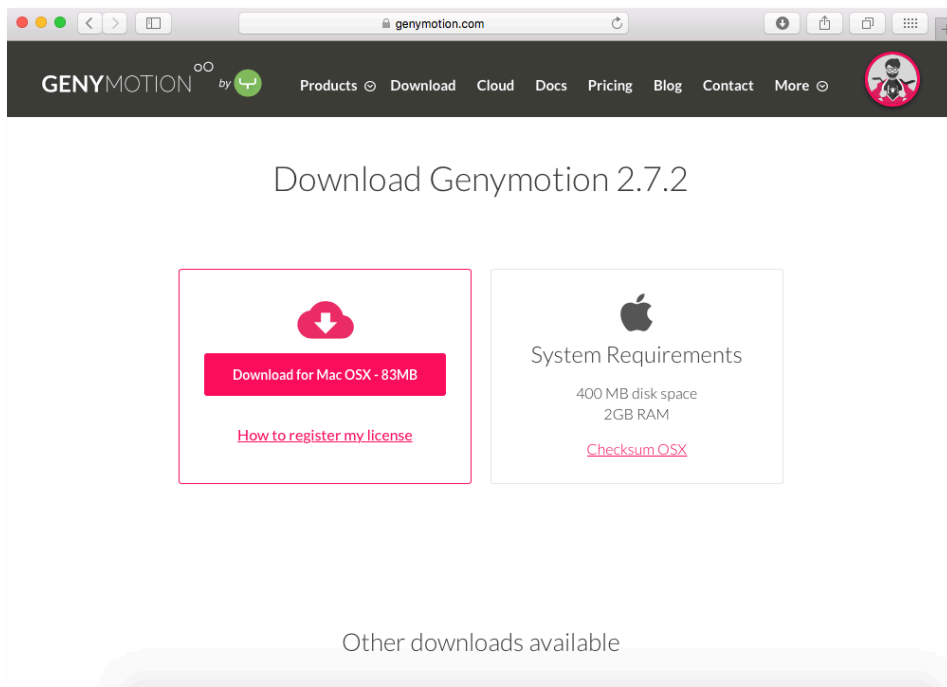


Figura 82. Pantalla 4 de la página web de Genymotion

Se abre el archivo .dmg descargado desde la página web de Genymotion y se arrastra el logo del software a la carpeta “Applications”. Esta pantalla se muestra en la figura 83.



Figura 83. Instalador de Genymotion

