Grado en Ingeniería Informática

Ingeniería del Software

Proyecto Fin de Grado

SocialMusFest

Aplicación para la gestión de eventos musicales mediante dispositivos Android

---

Autor

Rachid Boudhar Tannaoui

Director

José Ángel Vadillo Zorita

Junio 2016

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi director de proyecto José Ángel Vadillo Zorita por aceptar y confiar en nosotros (Rachid Boudhar Tannaoui y Jon Junguitu Iturrospe) a la hora de llevar a cabo este proyecto tan complejo. También me gustaría dar las gracias a Jon Junguitu Iturrospe por compartir esta experiencia de llevar a cabo un proyecto en conjunto.

En segundo lugar me gustaría agradecer a mi familia que año a año durante la carrera les ha supuesto un gasto y esfuerzos para que yo, hoy en día, pudiera estar aquí. Muchas Gracias de todo corazón.

Muchísimas gracias también a todos los profesores y compañeros de universidad ya que sin ellos estos últimos cuatro años, no se hubieran desarrollado de este modo, agradezco todas esas clases llevadas a cabo, las tutorías, esas explicaciones repetidas con el único objetivo de aprender o aprender, a todas esas consultas entre alumnos y ayudas prestadas... etc, todas ellas, llevadas de un modo altruista y por el simple hecho del compañerismo.

Me gustaría también agradecer a la Junior Empresa - Magna SIS de la cual formo parte y a todos sus miembros la colaboración prestada y el desarrollo de estos últimos dos años. Muchas gracias chicos.

Por último y no menos importante, me gustaría agradecer a los usuarios *testers* por haber confiado en mí y haberme ayudado en el transcurso de mejora de la aplicación en la fase de pruebas. Aupa ahí!!

Muchas Gracias a Todos!!

Resumen

Este documento contiene la memoria del Proyecto Fin de Grado sobre la aplicación Android SocialMusFest desarrollada por Rachid Boudhar Tannaoui para obtener la titulación del Grado de Ingeniería Informática en la facultad de informática de Donostia / San Sebastián en la UPV/EHU.

En esta memoria se pueden encontrar todos los puntos referentes hacia el correcto desarrollo de este proyecto tales como la planificación inicial del proyecto, la captura de requisitos, el análisis, la arquitectura del sistema, el diseño, la implementación , las pruebas, la gestión, una conclusión final y las posibles líneas de extensión del proyecto.

La finalidad de la aplicación Android SocialMusFest es la creación de un sistema de comunicación rápida y eficaz sobre la gestión de eventos (festivales o conciertos de música), para poder conocer y administrar qué tipo de eventos existen y en cuales deseamos participar, todo esto añadiéndolo a un sistema de red social para aumentar la participación entre los usuarios.

La aplicación está basada en el sistema operativo Android realizada con el software propio de Google Android Studio para el desarrollo de App bajo esta plataforma, algunas librerías de código abierto y otros recursos tales como imágenes, iconos, etc también sin derechos de autor.

La aplicación interactúa con el servidor donde está almacenada la base de datos y backend de este mismo proyecto en su versión Web. El acceso al servidor se realiza mediante la API de Meteor (ver capítulo 7.3.3) y el acceso a los datos mediante DDP (*Distributed Data Protocol*). Además en la implementación se han incluido los servicios de inicio de sesión de Google y Facebook.

Índice general

Índice de figuras

CAPÍTULO 1.

1. Introducción

El presente documento es la memoria del Proyecto Fin de Grado de la titulación del Grado de Ingeniería Informática con especialidad en el desarrollo del software, realizada por Rachid Boudhar Tannaoui bajo la dirección de José Ángel Vadillo Zorita, llevado el proyecto a cabo durante el segundo cuatrimestre del curso académico 2015-2016.

Se ha desarrollando en paralelo otro TFG por el alumno Jon Junguitu Iturrospe el cual implementa en su página web el modelo de datos (lado servidor) que se explica en el punto 7.2 de esta memoria. Su aplicación, bajo el mismo dominio, utiliza Meteor como herramienta tecnológica para el desarrollo, esta basada en Node.js y con frontend Javascript tanto en el lado cliente como en el servidor. Su página web además incorpora otras funcionalidades como por ejemplo la interacción entre usuarios, el poder ver un usuario a otro y decidir si “seguirle” o no.

En este punto del documento nos dedicaremos a describir la estructura de la memoria y daremos un breve resumen capitulo a capitulo sobre sus contenidos. Antes de ello, se desea mencionar un poco la historia del dispositivo Smartphone, su implicación en la sociedad y la idea o figura que representa. Como conclusión final, se describe lo que se intenta conseguir en esta aplicación para aprovechar en todo lo posible todos estos aspectos.

1.1. Antecedentes

En el año en el que nos encontramos, la tecnología relacionada con el entorno socio-personal no ha dejado de crecer a unos márgenes en los cuales a día de hoy, es casi una mera obligación estar incluido en ella, ya sea para poder evolucionar con el estilo de vida actual, o vivir en un entorno de mayor comodidad o facilidad, ya son muy pocas personas las cuales todavía no han escuchado de esa tan extraña palabra llamada “Smartphone” y por enlace a esta misma, el sistema tan amplio de aplicaciones que la cataloga.

Ya podremos recordar esa vieja época donde la tecnología no era aún una de nuestras más grandes amigas en lo que vida diaria se refiere, donde para poder asistir a algún evento o simplemente para poder conocer de algún acto o evento, se movía casi todo por el boca a boca y lo que pudieras escuchar. Con la inmersión del dispositivo móvil cuyas funcionalidades básicas que en aquel entonces, eran simplemente las llamadas, los mensajes de texto y ya dependientemente de la época, algunas funcionalidades añadidas que intentaron caracterizar lo aún más. Nos podemos dar cuenta, de cómo ha cambiado y todavía sigue cambiando todo el

sistema de calidad de vida desde aquel entonces, a niveles en los que prácticamente con un simple dispositivo de estas características, casi podemos realizar cualquier acción de comunicación o información.

Esta es la idea a la cual se quería enfocar SocialMusFest App, bien podríamos decir que la idea así como lo que representa, igual ya esta bastante vendida, no obstante en este caso se ha querido diferenciar del resto, dirigiéndose a un único tipo de eventos, los festivales y conciertos de música, además de ello, se buscaba conseguir una rápida comunicación hacia el público y la necesidad dé, dar al usuario, el conocimiento del entorno de eventos musicales en el que este se encuentre.

SocialMusFest App nace entonces, de la necesidad de entregar al usuario el conocimiento o información de los conciertos y festivales de música de una forma rápida y concentrada, pudiendo el usuario gestionar, buscar y visualizar toda la información relevante a estos, los conciertos y festivales de música que pudiera llegar a tener cerca y cuando serian sus fechas de apertura, así mismo, también se consigue que la parte organizadora pueda difundir sus eventos de una forma más rápida y objetiva, ya que los usuarios que tengan instalada esta aplicación será porque les interesa el tema.

1.2. Organización del documento

La presente memoria está estructurada de la siguiente manera:

- **Capítulo 1:** Se inicia con la introducción y presentación del proyecto, la motivación y la evolución del proyecto a desarrollar desde sus inicios y por último se explica la estructuración de este mismo documento.
- **Capítulo 2:** En este capítulo se encuentra la documentación que hace referencia a los objetivos del proyecto, la planificación realizada al inicio del este y su evolución conforme se avanza en el proyecto, también podemos encontrar en este capítulo, el alcance, el método de trabajo seguido y realizado, la planificación temporal llevada a cabo, la calidad del producto y los recursos externos utilizados.
- **Capítulo 3:** Aquí se han definido las tecnologías utilizadas para el desarrollo del producto, las herramientas utilizadas para la gestión de versiones, que sistema de almacenamiento ha parecido más óptimo y adecuado, y que herramientas o software se ha utilizado para el desarrollo de esta documentación.
- **Capítulo 4:** Debemos de conocer primero cual es la arquitectura del sistema operativo en el que tenemos pensado desarrollar una aplicación, por ello en este capítulo se ha descrito de qué y cómo está compuesto Android para explicar más tarde el MDV llevado a cabo, los requisitos mínimos de instalación de la App y los permisos necesarios.

- **Capítulo 5:** En este capítulo se ha llevado a cabo la captura y el análisis de requisitos, se han ideado y desglosado una serie de casos de uso básicos para el desarrollo de la aplicación.
- **Capítulo 6:** Los diseños de los casos de uso de este capítulo se han realizado manteniendo el orden por el que se trataron en el anterior capítulo, además de visualizar el diseño de cada caso de uso, se han explicado las funciones que las conforman. En este capítulo también podemos encontrar el diseño de las actividades de la App y el diseño del modelo de daños.
- **Capítulo 7:** Seguido del anterior capítulo, se desglosa la implementación de la aplicación compuesta por una extensa explicación de todos los componentes que lo conforman, desde las APIs, los documentos desarrollados, los diseños de ventanas y las librerías externas utilizadas.
- **Capítulo 8:** Pruebas y testeo, en este capítulo hablamos de cómo se realizó y qué metodología de depuración se decidió utilizar para la fase de *debugging*, así mismo también se recogen la serie de pruebas que se realizaron.
- **Capítulo 9:** En este capítulo se recoge la gestión del proyecto llevado a cabo, el control y seguimiento, el coste y el tiempo dedicado.
- **Capítulo 10:** Terminando la memoria, se ha llevado a cabo una conclusión y se ha recogido la experiencia personal del autor frente a este nivel de desafíos, también se han recogido las líneas futuras y propuestas de extensión de la aplicación, ya que se tiene pensado dentro de unos días volver a la fase de desarrollo para llevar la aplicación a un entorno de explotación.
- **Bibliografía:** Para finalizar en este último punto de la memoria se recoge la bibliografía utilizada describiendo los enlaces que se encontraron de utilidad.

CAPÍTULO 2.

2. Documento de objetivos del proyecto (DOP)

En este capítulo se detallan los objetivos del proyecto, el alcance del mismo, el método de trabajo explicando cada una de sus iteraciones, la planificación temporal a llevar a cabo, la calidad tanto del producto como de la documentación y el uso de los recursos externos utilizados. Dentro del alcance se desglosan las características en un mínimo deseado de la aplicación, la líneas de ampliación en caso de completar el alcance mínimo, y finalmente se documentan las exclusiones del proyecto. La planificación temporal a su vez, cuenta con la recogida de entregables, los hitos y el diagrama de Gantt del proyecto.

2.1. Objetivos

El objetivo de este proyecto es el desarrollo de una aplicación Android encargada de visualizar eventos sobre conciertos y festivales de música, añadiendo la posibilidad de que los usuarios puedan interactuar con ellos para decidir si desean asistir o no a un evento. Para ello se necesitará un sistema de cuentas que caracterice a la App con el objetivo de que los usuarios puedan registrarse en ella. En los eventos, un usuario podrá visualizar qué festivales y conciertos se encuentran en sus alrededores, esto ocurrirá una vez dada por parte del usuario una localización próxima o bien si lo desea este, puede ser localizado vía GPS. Una vez encontrado un evento en el que el usuario desee participar podrá registrarse en el, no obstante todos los eventos tendrán una capacidad máxima que no se podrá superar, una vez llegado a tal límite, se deshabilitan los registros de otros usuarios a este.

No menos importante el otro objetivo del proyecto también ha sido el poder conseguir trabajar en equipo con otro miembro respecto a la idea, y de ahí, a las partes que se distinguen en ambos proyectos como lo son sus respectivas plataformas y desarrollo.

2.2. Alcance

En este apartado se detalla el alcance mínimo exigible, las posibles líneas de ampliación, las exclusiones, la estructura de trabajo, la planificación, los entregables y la calidad del producto final.

2.2.1. Mínimo exigible

La aplicación contará por lo menos las siguientes características:

- Usuarios:
    - No registrados
        - Podrán iniciar sesión
        - Introducir su localización
        - Buscar eventos
        - Ver un evento concreto
        - Ver la información de la App.
    - Conocida su localización
        - Podrán iniciar sesión
        - Actualizar su localización
        - Visualizan una gama de eventos según su localización
        - Ver un evento concreto
        - Buscar eventos
        - Ver la información de la App.
    - Registrados
        - Podrán actualizar su perfil
        - Cambiar la contraseña
        - Cambiar el idioma
        - Cerrar sesión
        - Actualizar su localización
        - Visualizan una gama de eventos según su localización
        - Ver un evento concreto
        - Registrarse a un evento
        - Publicar un evento
        - Ver la información de la App.

- Secciones de la aplicación:
    - Página principal : SocialMusFest
        - Menú principal
            - Página principal
            - Mi Cuenta
            - Publicar
            - Buscar
            - Información
        - Icono y texto de bienvenida
        - Botón para introducir una localización
    - Página principal : Eventos (sin tener iniciada sesión)
        - Menú principal

                - Texto de la localización
                - Botón para actualizar la localización
                - Página principal
                - Mi Cuenta
                - Publicar
                - Buscar
                - Información
            - Tiene tres apartados
                - Eventos publicados
                - Eventos
                - Eventos registrados

    - Página principal : Eventos (con sesión iniciada)
        - Menú principal
            - Imagen del usuario
            - Nombre de usuario
            - Email del usuario
            - Texto de la localización
            - Botón para actualizar la localización
            - Página principal
            - Mi Cuenta
            - Publicar
            - Buscar
            - Información
        - Tiene tres apartados
            - Eventos publicados
            - Eventos
            - Eventos registrados

    - Mi Cuenta
        - Botón para iniciar sesión con Google
        - Botón para iniciar sesión con Facebook
        - Botón para iniciar sesión con Formulario
        - Botón para registrarse con Formulario

    - Iniciar sesión
        - Botón atrás
        - Formulario para iniciar sesión
        - Enlace “No tienes cuenta?”
        - Enlace “Recuperar contraseña”

            - Abrir una ventana para introducir un email de recuperación.
    - Registrarse
        - Botón atrás
        - Formulario para registrarse
        - Enlace “Ya tienes una cuenta? Inicia Sesión”
    - Perfil
        - Botón atrás
        - Se visualizará el perfil de usuario pudiendo editar sus campos
            - Imagen
            - Información de cuenta
                - Usuario
                - Email
            - Información personal
                - Nombre
                - Género
                - Cumpleaños
                - Lugar
                - Estilo de música
        - Botón de opciones
            - Opciones
            - Cerrar sesión
    - Opciones
        - Botón atrás
        - Botón “cambiar contraseña”
        - Botón “cambiar idioma”
    - Publicar
        - Botón atrás
        - Texto informativo para el usuario
        - Botón para abrir explorador web y dirigir a publicar en web.
    - Buscar
        - Botón atrás
        - Formulario para buscar eventos
            - Lugar
            - Fecha
        - Botón para búsqueda de eventos
            - Mínimo introducir lugar.

    - Ver eventos
        - Botón atrás
        - Lista de eventos en la DB que cumplan los filtros de búsqueda
    - Visualizar un evento
        - Botón atrás
        - Características e información del evento
        - Botón para registrarse al evento

- La aplicación estará traducida a tres idiomas: Castellano, Inglés y Euskera.

2.2.2. Líneas de ampliación

En caso de haber terminado el alcance mínimo de la aplicación, se proponen las siguientes ampliaciones:

- Secciones de la aplicación:
    - Visualizar un evento
        - Características e información del calendario
            - Usuarios totales registrados en este calendario, se podrá ver que usuarios, (si así lo ha permitido un usuario) está registrado en este evento.
        - Botón para registrarse al calendario.
            - Opción de que otros usuarios vean o no, si me he registrado en este evento
    - Ver eventos
        - Botón atrás
        - Lista de eventos a los que el usuario se ha registrado
        - Compartir evento mediante otras aplicaciones

2.2.3. Exclusiones

Quedan fuera del alcance del proyecto los siguientes requisitos:

- Correcto funcionamiento sin acceso a Internet
- Correcto funcionamiento en versiones anteriores a Android 2.3
- El análisis de carácter legal sobre los datos relativos a la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) y la Ley de Servicios en la Sociedad de la Información (LSSI)
- La implementación de un sistema de seguridad más allá de lo que es la App base.

2.3. Método de trabajo

Se han definido cinco fases para el desarrollo del proyecto, en las cuales se determinará los objetivos y plazos a cumplir para tener un mayor control del desarrollo de este. Las fases por las que se desglosa el proyecto son las siguientes:

- **Primera fase:** Se trata de la fase de la planificación del proyecto.

  En esta fase se desarrolla la planificación temporal del proyecto, tales como los antecedentes, el alcance, los modelos de dominio, hitos, dedicación prevista … etc , se llevará a cabo en la primera semana del ciclo de vida del proyecto.

- **Segunda fase:** En esta fase se mantendrá un primer contacto con las tecnologías de desarrollo del proyecto.

  Se realizará un análisis de la mayoría de tecnologías mayormente conocidas para el desarrollo de aplicaciones en el sistema operativo Android, posteriormente se tomará la elección de qué herramienta se utilizaran, con que fin fue tomada esa decisión y que beneficios enfrenta hacia el resto de tecnologías recogidas en el estudio realizado, mismamente se realizarán diferentes pruebas e introducción en diferentes herramientas tecnológicas también para buscar qué método, herramienta, y plataforma es la más fácil de maniobrar y trabajar, ya que el tiempo es un bien escaso en este proyecto, por lo que no podemos esperar una dedicación de 2 meses al aprendizaje de una herramienta o tecnología.

- **Tercera fase:** Desarrollo de la aplicación Android.

  Por tratarse de una fase compleja y de gran tamaño, esta fase quedará desglosada en cuatro partes.

1. **Captura y Análisis de requisitos.** Como inicio de la fase se tomarán las premisas necesarias, las necesidades y funcionalidades que queremos tener en la aplicación y qué objetivos se desean conseguir, por ello para una mayor facilidad en la toma de requisitos se tomarán los casos de uso y el modelo de dominio que comprenda al proyecto.
2. **Diseño.** En esta parte de detalla que estructura, sistema de información, interfaz… etc, se desea conseguir respecto a la captura de requisitos ya realizada en la fase anterior, esto lo realizamos con el objetivo de facilitar el trabajo a la siguiente fase.
3. **Desarrollo e Implementación.** Una vez tomados los requisitos de un diseño, se pasara al desarrollo de la interfaz cumpliendo en todo lo posible al diseño que se llevó a cabo en la fase anterior, así mismo se programaran las funcionalidades necesarias en la aplicación

4. **Testeo.** En el transcursos del desarrollo de la aplicación, se realizarán todas las pruebas posibles con tal de "romper" la App de alguna forma para poder subsanar los errores, se realizarán en diferentes dispositivos móviles y versiones de Android, así mismo se buscará también un sistema de pruebas público con tal de hallar un análisis y nota del objetivo al que está dirigida la aplicación, facilidad de interacción, interfaz simple, funcionalidades claras … etc.

- **Cuarta fase:** Documentación de la memoria a un nivel más detallado con informes finales.

  En esta fase se llevará a cabo la mayor parte de la redacción de la memoria del proyecto, al ser una documentación extensa normalmente se realizan pequeños aportes y añadidos a la memoria respecto al ciclo de vida del proyecto, sin embargo, no será hasta que finalice la fase de desarrollo e implementación de la aplicación cuando finalmente se pueda profundizar en esta fase, tendrá una dedicación respecto al tiempo de vida del proyecto bastante alta, contando además que una vez finalizada esta fase se tendrá que llevar a cabo una preparación para la defensa del proyecto.

- **Quinta fase:** Gestión del proyecto y Finalización.

  En esta fase abarcan todas las gestiones desde la planificación del proyecto, control y seguimiento, y las lecciones aprendidas, viendo la planificación del proyecto se podrán distinguir con el control y seguimiento que se lleva diariamente, el alcance previsto y logrado hasta el momento, los trabajos y tareas realizadas junto a las horas reales invertidas, para ello se cuenta con un documento Excel en el que se anota el trabajo realizado, inicio y fin de este, junto al conteo de horas al final del día, teniendo así una suma de lo que haya llevado de tiempo el realizar una tarea o fase en concreto. Quedará constancia de ello, las desviaciones sufridas, problemas encontrados y soluciones tomadas.

2.4. Planificación temporal

Debido al gran tamaño del proyecto se detalla una planificación temporal. Esto se realiza para tener una primera visión del proyecto, una estructura fija de cumplimientos y ver el trabajo esperado a realizar, decimos esperado ya que posteriormente se llevará a cabo un control y seguimiento a lo largo de todo el ciclo de vida del proyecto, con el objetivo de encontrar problemas en el desarrollo de alguna tarea para evitar desviaciones en fechas previstas de finalización. Para llevar a cabo lo dicho se identifican las tareas, entregables, hitos y la estimación en tiempo de cada tarea.

2.4.1. Entregables

A continuación se detallan los entregables identificados del proyecto.

- **Memoria**. Documento formal y detallado que contendrá toda la información relativa al desarrollo del proyecto.
- **Aplicación**. Todo lo referente en cuanto al desarrollo de la aplicación, código fuente, APK, etc.

2.4.2. Hitos

Planificación

- **Inicio:** Lunes 29/2/2016
- **Fin:** Domingo 6/03/2016

Estudio de las tecnologías

- **Inicio:** Lunes 07/03/2016
- **Fin:** Domingo 13/03/2016

Captura de requisitos y Análisis

- **Inicio:** Lunes 14/03/2016
- **Fin:** Domingo 20/03/2016

Diseño y Estructura de la Aplicación

- **Inicio:** Lunes 21/03/2016
- **Fin:** Domingo 22/05/2016

Aprendizaje de la tecnología de desarrollo

- **Inicio:** Lunes 28/03/2016
- **Fin:** Domingo 3/04/2016

Desarrollo de la Aplicación

- **Inicio:** Lunes 4/04/2016
- **Fin:** Domingo 29/05/2016

Memoria del Proyecto

- **Inicio:** Lunes 23/05/2016
- **Fin:** Domingo 24/06/2016

Lecciones Aprendidas

- **Inicio:** Lunes 29/02/2016
- **Fin:** Jueves 24/06/2016

Seguimiento y Control

- **Inicio:** Lunes 29/02/2016
- **Fin:** Jueves 24/06/2016

Preparación de la Defensa

- **Inicio:** Lunes 24/06/2016
- **Fin:** Domingo 30/06/2016

2.4.3. Diagrama de Gantt

En la figura 2.4.3 se puede visualizar el diagrama de Gantt del proyecto, este diagrama representa la gestión de tiempo planificado por cada tarea, para el desarrollo de todo el ciclo de vida del proyecto.

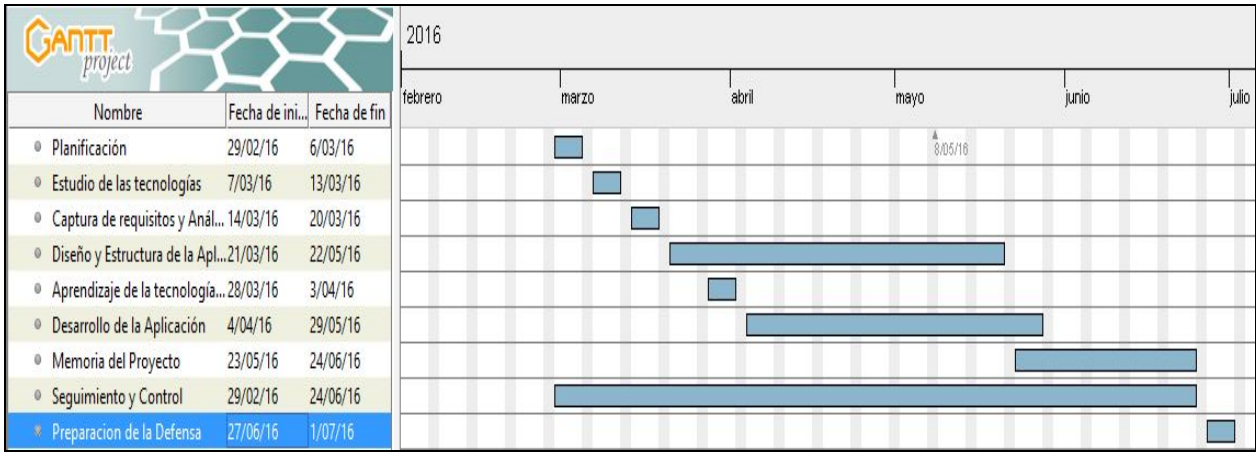


Figura 2.4.3: Diagrama de Gantt

2.5. Calidad

En este punto de redactan los métodos a utilizar para asegurar la calidad del proyecto.

- **Producto.** Para asegurarnos de la calidad del producto se realizará una vez completadas las fases *Alpha* de la aplicación determinadas pruebas por usuarios reales para determinar si el diseño, la implementación, la funcionalidad… etc, es correcta, y está bien diseñada, sino, se tendrían que realizar modificaciones según los criterios de los *beta-testers*, además de ello cómo serán los *beta-testers* encargados de instalar, probar y testear la aplicación en otros dispositivos Smartphone, también se buscarán problemas de incompatibilidades entre dispositivos y versiones Android
- **Memoria.** Para asegurarnos de la calidad de la memoria, se planifica una lista de hitos a cumplir a lo largo de la vida de esta y revisiones constantes para la realización de modificaciones y ajustes.

2.6. Recursos externos

En este proyecto será llevado a cabo un sistema de almacenamiento externo para la supervivencia del mismo, se contarán con copias de seguridad realizadas en Google Drive, Mega y el ordenador de uso personal del Autor.

Así mismo, el producto será almacenado en GitHub para una mejor administración de las versiones que se irán desarrollando.

CAPÍTULO 3.

3. Elección tecnológica

En este punto se detallarán las tecnologías, librerías y herramientas utilizadas para el desarrollo del producto y la memoria justificando la elección de los mismos

3.1. Desarrollo del producto

3.1.1. Android Studio

Android Studio es un entorno de desarrollo integrado para Android, anunciado y desarrollado por Google remplazando como IDE oficial para el desarrollo de aplicaciones Android a Eclipse, convirtiéndose así en la herramienta más utilizada para el desarrollo de aplicaciones Android. Este, está basado en el software IntelliJ IDEA de JetBrains siendo gratuito bajo licencia Apache 2.0, está disponible para Windows, Mac OS X, y Linux. Lo caracteriza su renderización en tiempo real, su consola de desarrollo, las plantillas para crear diseños en Android y tener pre-instalado el SDK de Android y diferentes servicios como lo son Git, Navigation editor, AVD Manager… etc.

Al verse que esta era la herramienta más completa, oficial y más intuitiva entre sus competidoras, no se dudo en elegir esta opción como herramienta de desarrollo.

3.1.2. Meteor API

Se da uso a la *API* de Meteor ya que la versión web de este proyecto está desarrollado bajo Meteor. Meteor es una plataforma para crear aplicaciones web en tiempo real construida sobre Node.js, esta se localiza entre la base de datos de la aplicación y su interfaz de usuario y se encarga que las dos partes estén sincronizadas. Al mismo tiempo Meteor API permite a servicios de uso exterior el poder interactuar con esta plataforma para el uso de sus recursos, datos y funcionalidades.

En este punto la decisión de utilizar esta tecnología no residió en mi poder, al formar parte de los requisitos no funcionales del proyecto de Jon Junguitu Iturrospe paso también a formar parte de mis requisitos.

3.1.3. Scalingo

Scalingo es un hosting gratuito que ofrece por un periodo de tiempo el almacenamiento de una página web con compatibilidad a diferentes tecnologías, entre ellas nos interesa Meteor, cuenta con la posibilidad de añadir plugins y personalizar la configuración del sitio web que se desea mantener. Gracias a estas características se convirtió como única opción posible para la publicación de la página web de este proyecto en su versión web.

La elección de este hosting como se ha comentado, se redujo dé una amplia variedad a una limitada lista de opciones, quedando esta como única. El proyecto en su versión web en un inicio tenía previsto ser almacenado y subido al propio servicio que Meteor ofrece, siendo gratuito y realizable con un solo comando desde la consola, sin embargo, en el transcurso del desarrollo del proyecto pasó a ser de uso privado y de pago, encontrándonos en esta situación en diferentes ocasiones según el hosting que se estudiará, quedando al final este hosting como único que cumplía las necesidades mínimas requeridas.

3.2. Control de versiones

3.2.1. GitHub

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git, el código se almacena de forma pública aunque también se puede hacer de forma privada con opciones de pago, también proporciona diversas herramientas como una wiki para el mantenimiento de las distintas versiones, un sistema de seguimiento de problemas, una herramienta de revisión de código actualizado, y un visor de ramas donde poder comparar las diferentes versiones de código.

Se decidió utilizar este sistema de control de versiones porque se deseaba adquirir el conocimiento para su utilización, ya que se ha visto que grandes proyectos de código abierto están alojados en esta página web.

3.2.2. Git

Git es un software de control de versiones diseñado por Linus Torvalds pensando en la eficiencia y la confiabilidad del

mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Además de GitHub, se tuvo que utilizar esta herramienta para poder subir y almacenar el proyecto en su versión web en Scalingo, para subir un proyecto o hacer *push* sin necesidad de añadidos, siempre podemos utilizar la consola que GitHub nos ofrece *Git Shell*, en cambio, en esta ocasión para poder subir un proyecto al hosting de Scalingo, este nos solicitaba una *SSH Key* para autentificar la subida, después de la formación y documentación para realizar tal propósito se dio cuenta de que no se podía realizar la creación de la *SSH Key* con la consola de GitHub sino que era necesaria la consola *Git Bash* y las dependencias instaladas junto a Git. Por ello se usa este software.

3.3. Sistema de almacenamiento

3.3.1. Google Drive

Google Drive es un servicio de alojamiento de archivos desarrollado, publicado y mantenido por Google desde el 24 de abril del 2012, cada usuario cuenta a su disposición de 15GB de espacio gratuito para el almacenamiento de archivos siendo este ampliable mediante diferentes formas de pago. Es bastante completo ya que al ser multiplataforma permite el acceso desde diferentes plataformas, además de ello cuenta con Google Sync para poder sincronizar todo tu gestor en tu ordenador personal.

Se decidió utilizar este servicio de almacenamiento al tener conocimiento y facilidad de uso con el, se utiliza para el almacenamiento del proyecto, y se eligió por tener una muy baja probabilidad de pérdidas de datos ya que los documentos o archivos almacenados en él no se borran del sistema aún eliminándolos directamente.

3.3.2. Mega

MEGA es un servicio de almacenamiento de archivos que al igual que Google Drive posee un fuerte sistema de cifrado de datos, haciéndose visible la diferencia de almacenamiento en comparación a su competidor Google Drive siendo este la oferta de 50GB de espacio gratuito por usuario, al mismo tiempo también cuenta con un sistema de sincronización, es multiplataforma y tiene múltiples herramientas de uso popular.

El motivo de su elección como doble capa de seguridad para almacenar el proyecto también se rige a sus fuertes medidas de protección, al amplio uso del autor del proyecto hacia esta herramienta y a su amplia capacidad en caso de que se requiera un volumen de datos mayor.

3.4. Documentación

3.4.1. GanttProject

GanttProject es un software de código libre bajo licencia GPL basado en Java, el software se encarga de realizar Diagramas de Gantt caracterizando las gestión de los proyectos. Este software funciona en Windows, Linux y Mac OS X.

Se utilizó este software para la creación de los Diagramas de Gantt que se encuentran en esta memoria, la decisión de utilizar este software fue por recomendación de Jon Junguitu Iturrospe ya que es fácil de utilizar y obtiene buenos resultados.

3.4.2. Yuml Diagram

Yuml es una página web dedicada a la creación de todo tipo de diagramas, está caracterizado por tener un sistema lógico de programación a la hora de realizar cualquier diagrama.

Esta herramienta ha sido utilizada para la creación de todos los diagramas de diseño que se puedan encontrar en esta memoria, al igual que el software GanttProject, esta página web también fue recomendada por Jon Junguitu Iturrospe por su rapidez y facilidad de uso a la hora de realizar los diagramas.

3.4.3. Creately

Creately es un programa web creado por Cinergix, Pty Ltd, para el desarrollo de diagramas de sistema, infografías, diagramas de flujo, diagramas de Gantt, organigramas, diseños UML, mapas mentales, diseños de placa de circuito, el arte del *doodle* y muchos otros tipos de diagramas. Al ser una herramienta multiplataforma puede ser instalada en diversos entornos además de contar con una versión web de desarrollo la cual ha sido la plataforma utilizada.

Esta herramienta ha sido utilizada para la creación de todos los diagramas de sistema que se puedan encontrar en esta memoria, su uso y recomendación les precede de las dos anteriores.

CAPÍTULO 4.

4. Arquitectura del sistema

En este apartado de la memoria se explica la arquitectura del sistema operativo Android, como se visualiza el Modelo Vista Controlador en este sistema operativo, las capas de programación, los niveles de clases, la arquitectura del sistema de funciones características de Android, además de realizarse un repaso a los requisitos mínimos solicitados por la aplicación y se describen los permisos necesarios para su instalación.

4.1. Arquitectura

Como ya se había comentado anteriormente Android es una plataforma para dispositivos móviles, lo que no se había visto hasta el momento es que contiene una pila de software donde se incluye un sistema operativo, un middleware y diferentes aplicaciones básicas para el usuario. En las siguientes líneas se dará una visión global por capas para ver la arquitectura empleada en Android, desde la capa más externa a la más interna. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la siguiente figura:

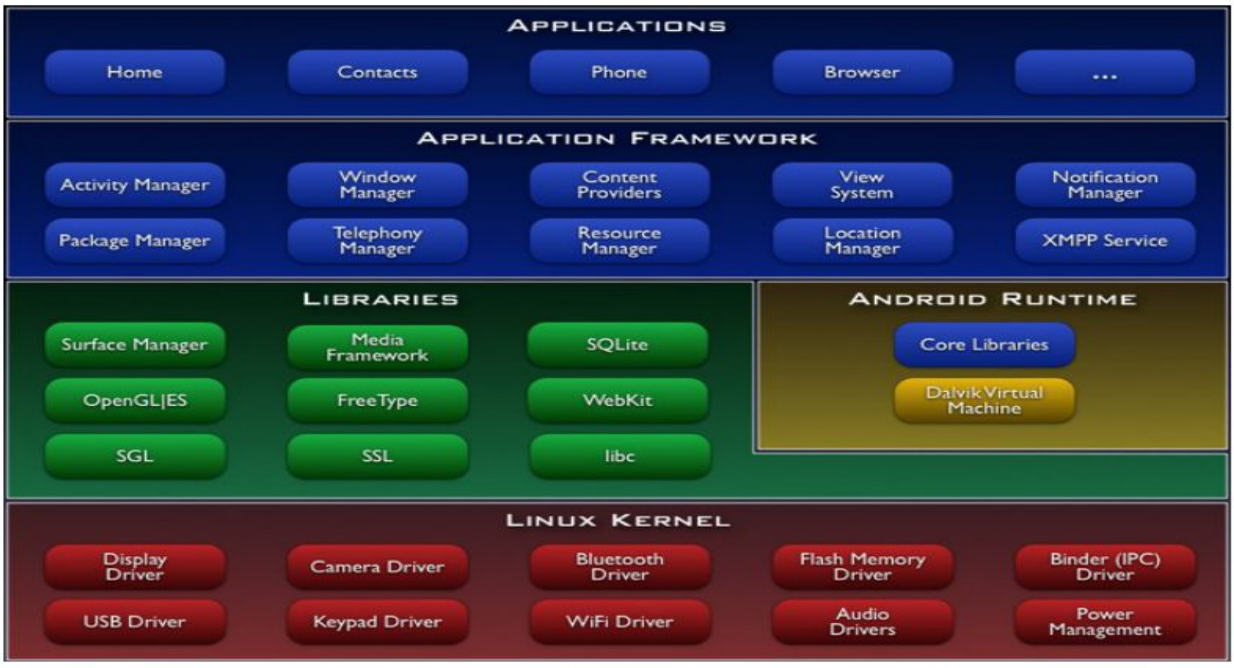


Figura 4.1.: Diseño de la arquitectura de Android

A continuación, se pasa a explicar y detallar los diferentes niveles y componentes de la arquitectura de Android:

- **Aplicaciones:** Este nivel contiene muchas aplicaciones base, entre ellas incluyen un proveedor de contenido, un cliente de correo electrónico, un gestor de recursos, programa de SMS, gestor de notificaciones, calendario, mapas, navegador, contactos y otros, además de las que el usuario vaya añadiendo posteriormente, este conjunto de interfaces son ejecutadas bajo la máquina virtual Dalvik aunque a partir de la versión 5.0 de Android paso a ser ART (Android Runtime) el nuevo entorno de ejecución. Normalmente las aplicaciones están escritas en Java, aunque también pueden ser realizadas en C/C++, dependiendo de esto se utiliza el SDK (Software Development Kit) o el NDK (Native Development Kit) de Android siendo el último para el desarrollo bajo C/C++, aquí también podemos encontrar el lanzador, el cual es el encargado de ejecutar otras aplicaciones, y donde se muestran los escritorios donde se pueden colocar accesos directos a estas aplicaciones.

- **Framework de Aplicaciones:** Esta es la capa que nos interesa a los desarrolladores, ya que en ella encontramos todas las librerías Java que necesitamos para programar nuestras aplicaciones, en ella podemos encontrar todas las clases, servicios y herramientas que utilizan directamente las aplicaciones para realizar sus funciones, todas las aplicaciones que se desarrollen para Android utilizan el mismo conjunto de API y el mismo framework, esto está ideado de esta forma para la reutilización de los componentes, tales como funciones, métodos, recursos, etc … código en general, para evitar la necesidad de crear todo desde cero, este framework de aplicaciones es tan completo que podemos encontrar:

  - El **Administrador de actividades** se encarga de administrar la pila de actividades de nuestra aplicación así como su ciclo de vida, esto se ha utilizado para la realización e iteración de las actividades como hemos visto anteriormente en el capítulo 6 de diseño.

  - **Administrador de ventanas** se encarga de organizar lo que se mostrará en pantalla, básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades, en nuestra aplicación se ha creado quince ventanas de pantalla, seis ventanas de diálogo, dos cabeceras de menú, tres navegadores de ventana y otros múltiples archivos en formato .XML donde se ha definido estilos y formas de otras ventanas, todo esto gracias a esta librería.

  - **Administrador de paquetes**, esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la

instalación de nuevos paquetes, al referirse a paquetes hacemos un llamamiento al término APK que queda explicado en los siguientes puntos. En nuestra aplicación se podría decir que esta librería se utiliza de forma abstracta cuando se instala en algún dispositivo móvil.

- **Proveedor de contenido** Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener después control de cómo se accede a la información.

- Las **vistas** en Android, sos elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas o elementos avanzados como navegadores web, Google map, Google profile … etc, en nuestra aplicación la mayor parte está conformada por estos elementos tan vitales que caracterizan a una aplicación Android.

- El **Administrador de notificaciones** engloba los servicios para notificar al usuario cuando algo requiere su atención mostrando alertas en la barra de estado, también nos permite la modificación de parámetros del dispositivo móvil tales como el sonido, el vibrador, las luces LED y otras funcionalidades. En nuestra aplicación al no realizar ningún tipo notificación no ha a hecho falta la utilización de esta librería.

- **Administrador de telefonía**, con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

- Con el **Administrador de recursos** podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos y layouts, en SocialMusFest se ha utilizado múltiples veces para poder guardar o acceder a la imagen de perfil o a los documentos color.xml, values.xml, string.xml y otros tantos donde tenemos los parámetros e idioma a modificar por el usuario en la aplicación.

- El **Administrador de localizaciones** permite determinar la posición geográfica del dispositivo movil mediante GPS o redes disponibles y trabajar con mapas, en nuestra aplicación hemos utilizado esta librería para obtener la localización del dispositivo móvil vía GPS.

- **Administrador de sensores**, permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de

campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.

- ➢ **Cámara**, con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.
- ➢ La librería **Multimedia** permite reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

- ❖ **Librerías:** La componen las bibliotecas nativas de Android, también llamadas librerías, están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma "más eficiente". Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:
  - ➢ **Libc:** Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
  - ➢ **Surface Manager:** Se encarga de componer las imágenes que se muestran en la pantalla a partir de capas gráficas 2D y 3D, además de gestionar también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
  - ➢ **OpenGL/SL y SGL:** Representan las librerías gráficas, OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones.
  - ➢ **Biblioteca multimedia:** Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android, vídeo, audio, imágenes estáticas o animadas… etc, lo que permite visualizar, reproducir o grabar numerosos formatos de imagen, vídeo y audio como JPG, GIF, PNG, MP4, MP3… etc .
  - ➢ **FreeType:** Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes tipográficas tanto basadas en mapas de bits como de renderizado vectorial.
  - ➢ **SSL:** Proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).
  - ➢ **SQLite:** Creación y gestión un potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.

- **WebKit:** Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android. En la versión 4.4, WebKit ha sido reemplazada por Chromium/Blink, que es la base del navegador Chrome de Google o Safari de Apple.

- **Entorno de ejecución:** Como podemos apreciar en el diagrama, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí encontramos las librerías con la funcionalidades habituales de Java así como otras específicas de Android aunque el componente principal del entorno de ejecución de Android es la máquina virtual Dalvik pero como hemos comentado en anteriores puntos a partir de la versión 5.0 de Android le releva su sucesor ART que consigue una eficiencia del 33%. Aquí en las máquinas virtuales las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute, la ventaja de esto es que las aplicaciones se compilan solamente una vez y de esta forma estarán listas para distribuirse y ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

- **Núcleo Linux:** El núcleo de Android está formado por el sistema operativo Linux versión 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos, elementos de comunicación (networking), etc. El soporte de drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Por ejemplo si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea, para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android. Por eso un desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores.

4.1.1. Formato .APK

El formato .APK (***A****ndroid Application* ***P****ackage*) es un paquete de compresión ZIP utilizado para la distribución e instalación de aplicaciones en la plataforma Android, es una variante del formato JAR de Java. En este archivo se definen una serie de parámetros que durante el desarrollo de una aplicación se deberían definir cautelosamente ya que esto permitirá que

versiones posteriores de Android también puedan ser capaces de ejecutar la aplicación sin ningún problema.

Un archivo APK contiene los siguientes ficheros:

- ➢ **Android Manifest:** Este archivo contiene las características principales que tendrá nuestra aplicación como por ejemplo los permisos, su versión, las versiones previas soportadas, las dimensiones de la pantalla, etc.
- ➢ **Classes.dex:** Este será el fichero compilado preparado para ejecutarse en la Máquina Virtual Dalvik o en su defecto ART.
- ➢ **Carpeta Resources:** Aquí encontramos todos los archivos externos que usamos para construir nuestro proyecto, como por ejemplo nuestros iconos, audio, archivos planos de texto, los archivos .xml de diseño, etc.
- ➢ **Librerías nativas:** Contiene aquellas librerías de las cuales depende la aplicación.
- ➢ **Carpeta META-INF:** Aquí guardamos las firmas digitales de la aplicación, además debes indicar tu ID de desarrollador si es que deseas ser reconocido y autenticado en procesos de comercialización.

4.1.2. Ventajas de Gradle

En este capítulo se quiere defender el uso y ventajas de Gradle, ya que su utilización en el desarrollo de SocialMusFest App fue una gran ventaja al permitirnos bajar el tiempo de compilación de más de cuatro minutos de espera (por cada vez que se requería compilar) a unos ridículos quince o veinte segundos cuando la aplicación tomaba unas dimensiones considerables en tamaño. Demos por tanto una breve explicación de lo que es:

Gradle es una herramienta de automatización que permite la construcción de nuestro código, se apoya en Groovy y en DSL (*Domain Specific Language*) para construir el *build* y dispone de un sistema de gestión de dependencias.

A continuación, se defienden los puntos fuertes de Gradle frente a sus competidores:

- ❑ Una de las mayores ventajas que tiene Gradle es que le **entrega** el poder del flujo de construcción al programador, quiere decir, que es el desarrollador quien decide el orden de ejecución de las tareas, así como elegir que archivos compilar primero, cuando detener la compilación, establecer condiciones para que se recompile o no el código y muchas situaciones más.

- El poder de las **ejecuciones incrementales**, esta característica le ahorra al programador gran cantidad de tiempo de espera. Al haber construcciones incrementales podemos decidir hasta qué punto queremos que se compile nuestra aplicación, es decir, si no es necesario compilar una parte del código debido a la ausencia de errores, entonces se procede a compilar la sección que aún no ha sido probada.
- **Múltiples versiones:** Gradle permite que construyamos varias versiones de nuestra aplicación. Por ejemplo, si deseas construir tu aplicación para Jelly Beans y para Kitkat entonces solo debes especificar que el proyecto tendrá dos variantes de empaquetado, configurando la versión del SDK usada para cada versión.
- La ejecución y las **pruebas** se pueden realizar en un mismo proyecto y puedes ejecutar tareas en hilos diferentes para optimizar el proceso de construcción.

4.2. El Modelo Vista Controlador

En Android utilizamos el patrón de arquitectura llamado Modelo Vista Controlador (MVC), cuya principal función es proponer la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador**, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. A continuación se describen las tres capas:

- **Modelo:** Nos referimos con modelo a las representaciones que construiremos basadas en la información con la que operará nuestra aplicación. En esta parte del modelo juega la decisión de qué modelo se puede utilizar para almacenar información, ¿Base de datos? ¿Web services?. El modelo que se elige al final depende obviamente de las necesidades de información de la aplicación.
- **Vista:** La vista no es más que la interfaz con la que va a interactuar el usuario. En Android, las interfaces las construimos en XML. Construimos nuestro esqueleto en XML, posteriormente, con ayuda de estilos, que también los escribimos en XML, podemos empezar a darle formato de colores, posiciones, formato, etc. a nuestro esqueleto.
- **Controlador:** Finalmente nos topamos con el controlador que responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo'. Por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

Figura 4.2: Modelo Vista Controlador en Android

4.3. Términos y requisitos

En este punto se han redactado los términos que definen y aclarar ciertas informaciones de utilidad de cara a los usuarios.

Así mismo, se redacta la siguiente información:

1. **Permisos, licencia y condiciones de uso:** Los permisos necesarios para la instalación de SocialMusFest App se recogen en el punto 4.4 de la memoria, además se le asegura al usuario de que siempre se le solicitarán los permisos de instalación antes de que esta se instale. Las condiciones de uso están redactadas en la página web pudiendo el usuario desde la aplicación acceder a esta ventana a través de la opción “Información” del menú, sección “Términos del servicio”.

2. **Derechos propios y de terceros:** Se recoge al inicio de esta memoria el tipo de licencia por el que queda registrado la totalidad de este proyecto, además se da uso de recursos y otros contenidos bajo licencias de código libre o sin derechos de autor.

3. **Entorno de edad:** Esta App está dirigida a usuarios mayores de edad ya que el entorno y contexto en el que se mueve suele tener una edad media de 22 años.

4. **Privacidad y geolocalización:** SocialMusFest respeta la privacidad de sus usuarios, por ello tiene implementado un sistema de búsqueda de eventos sin necesidad de dar una localidad o ubicación al sistema. En cambio para poder apuntarse a un evento es necesario contar con una cuenta de usuario, el nombre de usuario y el email son dos campos obligatorios pudiendo el usuario eliminar o borrar la información adicional que personaliza su cuenta personal. Para más información el usuario puede acceder a las Políticas de Privacidad redactadas en la página web y que además pueden ser consultadas desde la App

5. **Autor:** En la aplicación Android se deja constancia en la venta de “Versión de la App” el autor del proyecto, y en la memoria consta en la primera página de presentación.
6. **Publicidad:** SocialMusFest no recibe ni recibirá ningún tipo de publicidad, y no se monetiza ni monetizara de ninguna forma ni de ningún método.
7. Según los **requisitos mínimos del sistema** de SocialMusFest App, está es compatible con la mayoría de dispositivos móviles que utilizan una versión de Android superior a la 2.3, pero esto no quiere decir que se asegure su correcto funcionamiento en todos los Smartphone del mercado.

4.4. Permisos

Cuando se instala una aplicación en el sistema operativo Android, siempre se nos muestra una lista de permisos, si estamos de acuerdo con ellos podemos aceptarlos para que la aplicación se instale y en caso contrario, podemos negarnos y cancelar la instalación en el momento que decidamos que no queremos dar un permiso concreto. La mayoría de las aplicaciones para su correcto funcionamiento les es necesario el permiso o los permisos que soliciten en el momento de su instalación, aunque muchas veces y cada vez más, existen aplicaciones que abusan de esta práctica.

La aplicación SocialMusFest App necesita cinco permisos para su correcto funcionamiento:

- **Permiso para acceder a Internet:** Este permiso se solicita con “android.permission.INTERNET”, permite a la aplicación el acceso a internet. Por lo que se puede ver es un permiso básico que normalmente está incluido en casi todas las aplicaciones Android de hoy en día.
- **Permiso para comprobar si el dispositivo móvil está conectado a la red:** Este permiso nos permite detectar si el dispositivo móvil está conectado a alguna red a través de la tecnología wifi, esto se realiza con la finalidad de si se diera el caso en el que el dispositivo móvil no se encontrara conectado a una red en el momento en que un usuario diera uso de la App, esta muestra una serie de avisos de tipo *Toast* indicando que no se puede iniciar sesión, registrarse, modificar, ni obtener ningún evento..etc. La solicitud del permiso solicitado es el siguiente "android.permission.ACCESS_WIFI_STATE"
- **Permiso para comprobar si el dispositivo móvil está conectado a una red de datos:** Esta comprobación se realiza gracias al permiso "android.permission.ACCESS_NETWORK_STATE", SocialMusFest App necesita este permiso para la misma finalidad que el visto en el permiso del punto anterior. En este caso, este permiso nos permite comprobar si el dispositivo móvil está conectado a una red de telefonía o si tiene los datos de navegación habilitados.

- **Permiso para el uso de credenciales:** Este permiso nos es necesario por petición de Google y Facebook para poder acceder y utilizar la funcionalidad de inicio de sesión con sus servicios, necesitamos solicitar al usuario previamente a la instalación este permiso ya que después Google, realiza la obtención de las credenciales del usuario directamente dando a entender de que si la aplicación está instalada en el dispositivo móvil del usuario, este ya ha dado su consentimiento para la obtención de sus datos. Facebook en cambio, la primera vez que realizamos el intento de inicio de sesión, nos vuelve a recordar los permisos necesarios como se había comentado anteriormente en la definición de la actividad AccountActivity.java, por que este en cada intento de conexión solicita los permisos de uso de credenciales. Para la implementación de este permiso se ha hecho uso de "android.permission.USE_CREDENTIALS".
- **Permiso para la activación del GPS y obtención de la localización:** Este permiso junto a la API de Google *LocationServices.API* nos permite solicitar al usuario la activación del GPS del dispositivo móvil en caso de encontrarse apagado y también la posibilidad de obtener la ubicación del dispositivo móvil si así lo ha solicitado el usuario en la ventana de la actividad LocationActivity.java. El permiso que se ha utilizado es el siguiente "android.permission.ACCESS_FINE_LOCATION"

CAPÍTULO 5.

5. Captura de requisitos

En este punto se estudian los casos de uso y el modelo de dominio planteado para SocialMusFest App. Se han identificado tres tipos de actores diferentes, estos actores interpretan el papel de un usuario según el estado en el que se encuentre:

- Usuario anónimo
- Usuario conocida su localización
- Usuario registrado

5.1. Casos de uso

En este apartado se definen los casos de uso de SocialMusFest App, estos varían según el rol (anónimo, conocida su localización o registrado) en el que se encuentre el usuario, por ello se ha querido separar los diagramas de casos de uso según el estado de este, respetando los actores vistos en el punto anterior. En cada diagrama se explica cada caso de uso de forma individual, se resaltan sus características y el flujo de eventos que realizan cada uno de ellos.

5.1.1. Casos de uso del usuario anónimo

El diagrama de casos de uso en la figura 5.1.1 representa la iteración de un **usuario anónimo** con la aplicación.

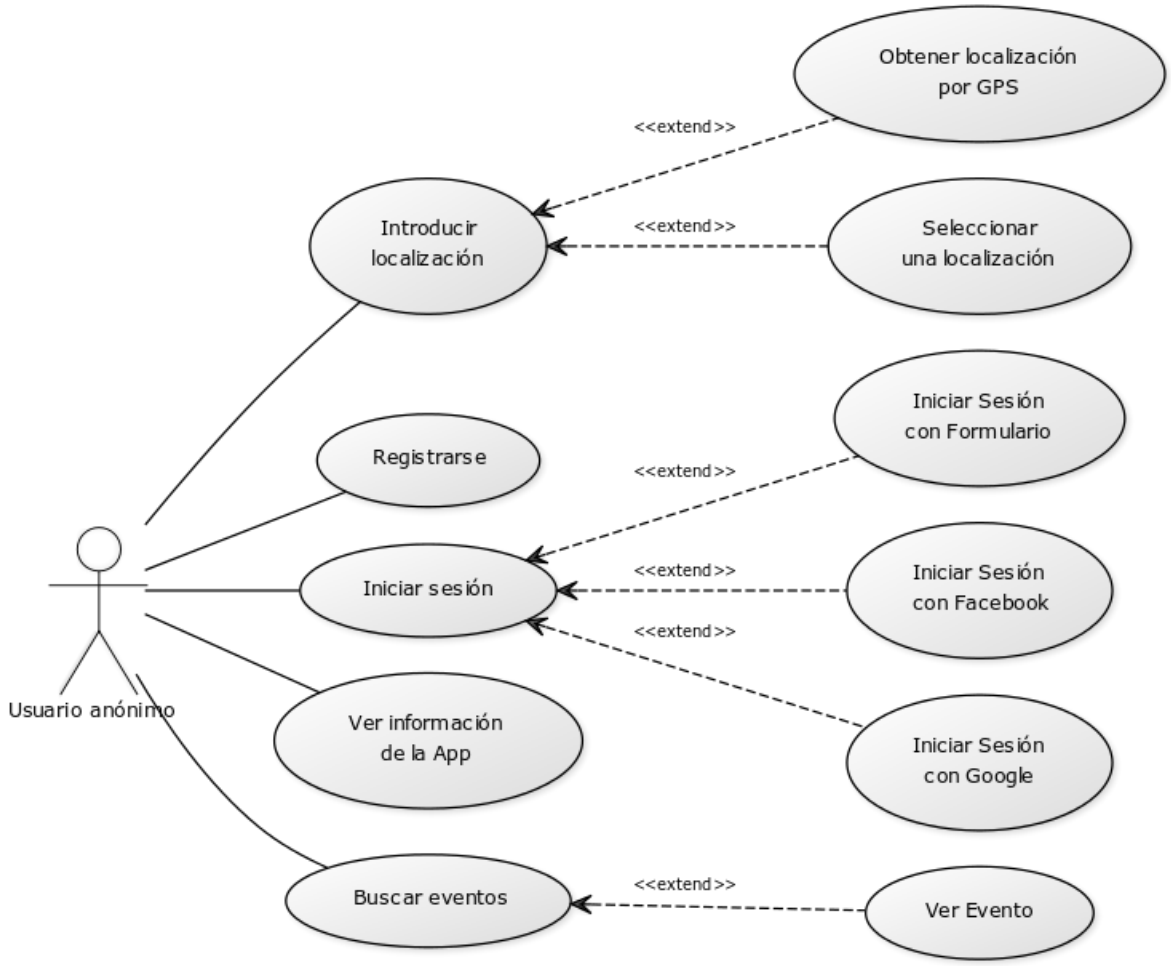


Figura 5.1.1: Diagrama de casos de uso del usuario anónimo

5.1.2. Casos de uso del usuario conocida su localización

El diagrama de casos de uso en la figura 5.1.2 representa la iteración de un **usuario conocida su localización** con la aplicación.

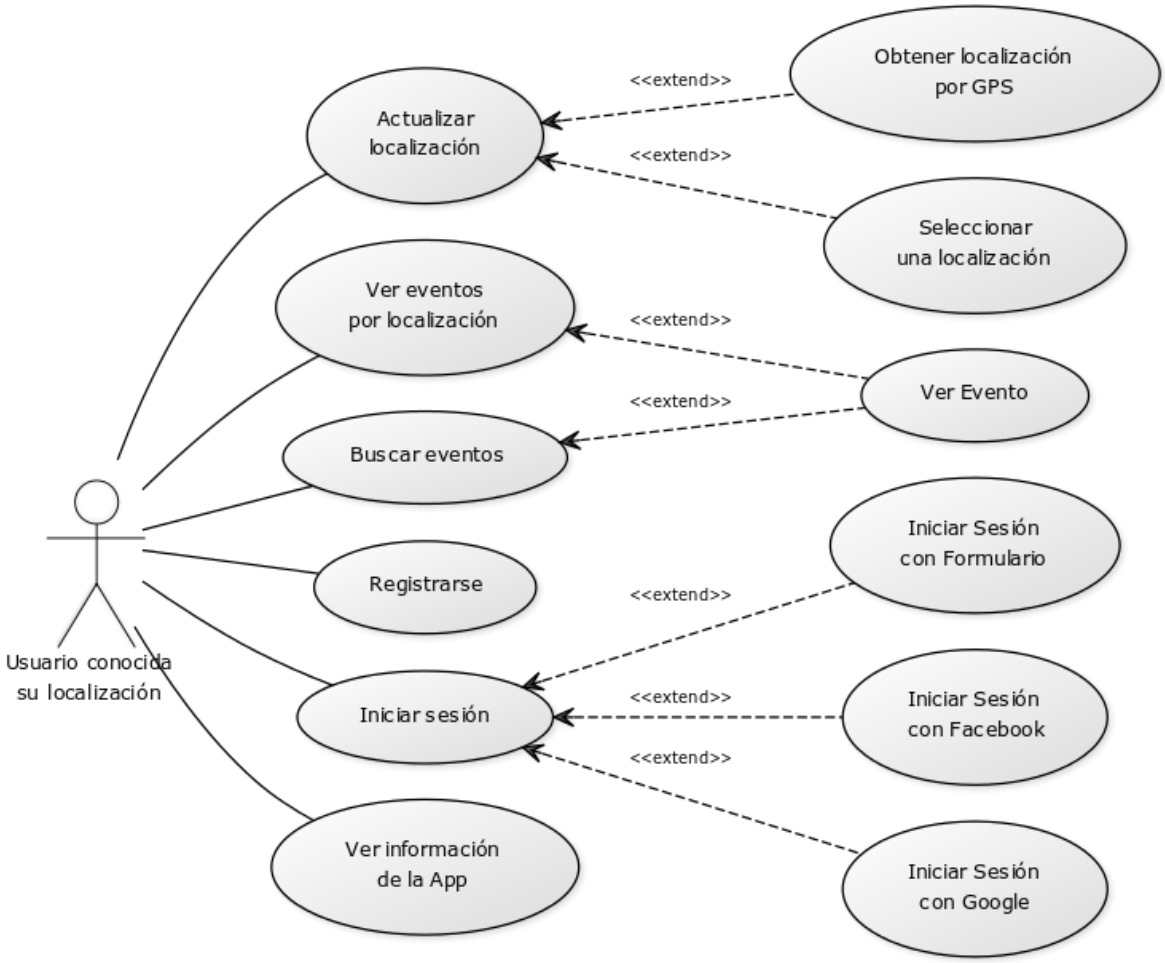


Figura 5.1.2: Diagrama de casos de uso del usuario conocida su localización

5.1.3. Casos de uso del usuario registrado

El diagrama de casos de uso en la figura 5.1..3 representa la iteración de un **usuario registrado** en la aplicación.

Figura 5.1.3: Diagrama de casos de uso del usuario registrado

5.2. Explicación de los casos de uso

5.2.1. Introducir localización

En este caso de uso se le permite al usuario la introducción de una localización para su posterior uso a la hora de mostrar la lista de eventos que irán ordenadas según este criterio. El usuario al pulsar en el botón de "Localización" que se encontrara en la ventana principal, se le redirigirá a otra ventana, la cual llamaremos ventana de localización, el usuario podrá optar por pulsar el botón "Obtener localización por GPS" para obtener su ubicación con la funcionalidad GPS o bien seleccionar de entre la lista de localizaciones más populares la localización que este desee.

- **Actor:** Usuario anónimo
- **Precondición:** El usuario tiene instalada e iniciada la App.
- **Escenario principal:**
    1. El usuario anónimo pulsa el botón "Introducir localización" de la página principal
    2. La aplicación redirecciona a la ventana de localización
    3. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

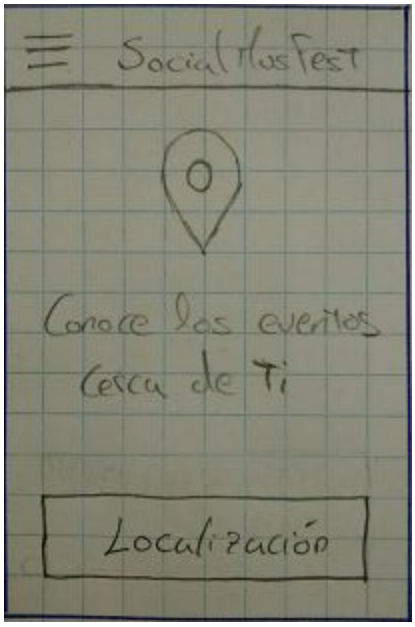


Figura 5.2.1: Prototipo de la ventana principal "Introducir localización"

5.2.1.1. Obtener con GPS (apagado)

Este caso de uso, SocialMusFest App obtiene la localización del dispositivo móvil con la funcionalidad GPS del sistema, estando esta utilidad apagada. Después, actualiza el estado de la localización del usuario anónimo con la ubicación adquirida.

- **Actor:** Usuario anónimo
- **Precondición:** El usuario anónimo se encuentra en la ventana de localización, el GPS del dispositivo se encuentra apagado.
- **Escenario principal:**
    1. El usuario anónimo pulsa el botón “Obtener localización por GPS”
    2. El sistema intenta obtener la ubicación del dispositivo pero detecta que el GPS está apagado y solicita al usuario el permiso para poder activarlo.
    3. Si al usuario acepta la solicitud el sistema obtiene la ubicación del dispositivo, actualiza la localización del usuario anónimo y redirecciona a la página principal, por lo contrario si el usuario no acepta la solicitud muestra un mensaje de error correspondiente.
- **Postcondición:** (Ninguna)

5.2.1.2. Obtener con GPS (encendido)

Este caso de uso obtiene la localización del dispositivo móvil con la funcionalidad GPS del sistema estando el GPS encendido, después actualiza el estado de la localización del usuario anónimo con la ubicación adquirida.

- **Actor:** Usuario anónimo
- **Precondición:** El usuario anónimo se encuentra en la ventana de localización, el GPS del dispositivo se encuentra encendido
- **Escenario principal:**
    1. El usuario anónimo pulsa el botón “Obtener localización con GPS”

2. El sistema intenta obtener la ubicación del dispositivo, si la obtiene, actualiza la localización del usuario anónimo y redirecciona a la página principal, si por lo contrario no obtiene la localización muestra un mensaje de error.

- **Postcondición:** (Ninguna)

5.2.1.3. Seleccionar una localización

En este caso de uso se obtiene la localización por parte del usuario, quien es el encargado de seleccionar de entre la lista de las localizaciones la que desee, una vez seleccionada una localización de la lista, la aplicación actualiza su estado.

- **Actor:** Usuario anónimo
- **Precondición:** El usuario anónimo se encuentra en la ventana de localización.
- **Escenario principal:**
    1. El usuario anónimo selecciona una localización de entre la lista de localizaciones.
    2. La aplicación obtiene la opción seleccionada por el usuario, actualiza la localización del usuario anónimo con esta nueva opción y redirecciona a la página principal.
- **Postcondición:** (Ninguna)

Figura 5.2.1.3: Prototipo de la ventana “Localización”

5.2.2. Registrarse

Este caso de uso permite al usuario el poder registrarse en la aplicación para poder utilizar posteriormente una serie de funcionalidades que sólo están operativas para usuarios registrados, para ello deberá acceder a la ventana de registrarse con formulario desde la ventana cuenta y pulsar en el botón “Registrarse”.

- **Actor:** Usuario anónimo o Usuario conocida su localización
- **Precondición:** El usuario se encuentra en la página de registrarse con formulario
- **Escenario principal:**
    1. El usuario rellena el campo del nombre de usuario, el email, la contraseña y pulsa en el botón “Registrarse”.
    2. La aplicación validará primero los datos, si el nombre de usuario no está vacío y no existe en la base de datos, el email no está vacío, es un email válido y no está en la base de datos y la contraseña es mayor a cuatro caracteres, la aplicación creará una nueva cuenta e iniciara sesión con los datos proporcionados, finalmente redirigirá al nuevo usuario a la página principal pudiendo ver este, su nuevo estado en el menú. Si los datos no fueran válidos se mostraría un mensaje de error correspondiente.
- **Postcondición:** (Ninguna)

Figura 5.2.2: Prototipo de la ventana “Registrarse”

5.2.3. Iniciar sesión

En este caso de uso se le permite al usuario iniciar sesión con una cuenta ya existente o con sus cuentas de Google o Facebook, para ello el usuario deberá acceder a la ventana de cuenta a través de la opción “Mi Cuenta” del menú desplegable lateral.

- **Actor:** Usuario anónimo o Usuario conocida su localización.
- **Precondición:** El usuario se encuentra en la página principal
- **Escenario principal:**
    1. El usuario despliega el menú lateral y pulsa en la opción del menú “Mi Cuenta”
    2. La aplicación le redirigirá a la página de cuenta
    3. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

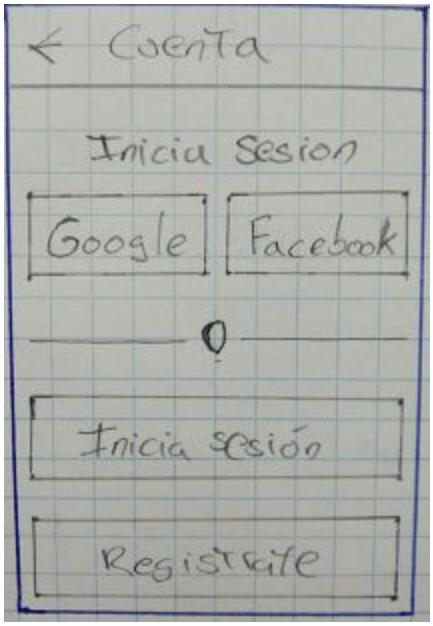


Figura 5.2.3: Prototipo de la ventana “Cuenta”

5.2.3.1. Iniciar sesión con formulario

En este caso de uso el usuario puede iniciar sesión mediante formulario, introduciendo los campos de email y contraseña para acceder, para ello debe de entrar a la página de iniciar sesión que se encuentra en la página cuenta una vez clickeado el botón “Iniciar sesión”.

- **Actor:** Usuario anónimo o Usuario conocida su localización.
- **Precondición:** El usuario se encuentra en la página de cuenta
- **Escenario principal:**
    1. El usuario clickea en la opción “Iniciar sesión”
    2. La aplicación redirecciona al usuario a la página de iniciar sesión con formulario
    3. El usuario rellena el campo del nombre de usuario, el email, la contraseña y pulsa en el botón “Registrarse”.
    4. La aplicación intentará iniciar sesión al usuario si el email y contraseña no están vacíos y son válidos y correctos, si se da el caso, iniciara sesión y redirecciona al usuario a la pagina principal, si por lo contrario no lo consigue, mostrará el correspondiente mensaje de error.
    5. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

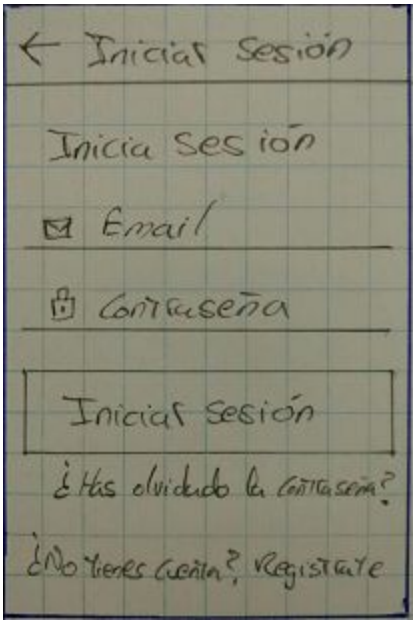


Figura 5.2.3.1: Prototipo de la ventana "Iniciar sesión con formulario"

5.2.3.2. Iniciar sesión con Google

En este caso de uso el usuario puede iniciar sesión mediante Google, para ello debe de clickear en el botón "Google" que se encuentra en la página de cuenta.

- **Actor:** Usuario anónimo o Usuario conocida su localización.
- **Precondición:** El usuario se encuentra en la página de cuenta
- **Escenario principal:**
    1. El usuario clickea en la opción "Google"
    2. (SocialMusFest solicitará al usuario qué cuenta de Google desea utilizar si es que este tuviera multi-cuentas en el dispositivo móvil)
    3. (El usuario seleccionara la cuenta con la que desee iniciar sesión en la aplicación).
    4. La aplicación intentará iniciar sesión al usuario, si lo consigue, iniciara sesión y redirecciona al usuario a la pagina principal, si por lo contrario no lo consigue, mostrará el correspondiente mensaje de error.
- **Postcondición:** (Ninguna)

5.2.3.3. Iniciar sesión con Facebook

En este caso de uso el usuario puede iniciar sesión mediante Facebook, para ello debe de clickear en el botón "Facebook" que se encuentra en la página de cuenta.

- **Actor:** Usuario anónimo o Usuario conocida su localización.
- **Precondición:** El usuario se encuentra en la página de cuenta
- **Escenario principal:**
    1. El usuario clickea en la opción “Facebook”
    2. (La aplicación cargará un formulario de inicio de sesión propio de Facebook si es que el usuario no estuviera ya iniciado con alguna otra aplicación de Facebook).
    3. (El usuario rellenará el formulario y pulsa en iniciar sesión).
    4. La aplicación intentará iniciar al usuario, si lo consigue, iniciara sesión y redirecciona al usuario a la pagina principal, si por lo contrario no lo consigue, mostrará el correspondiente mensaje de error.
- **Postcondición:** (Ninguna)

5.2.4. Ver información de la App

El usuario podrá ver las diversas informaciones que conforman la aplicación, tales como la ayuda, los términos de servicio, las políticas de privacidad y la versión de la App. Para poder acceder a la ventana donde están estas diversas opciones necesitamos clickear en la opción “Información” del menú desplegable lateral.

- **Actor:** Usuario anónimo / Usuario conocida su localización / Usuario registrado
- **Precondición:** El usuario se encuentra en la ventana de información de la App.
- **Escenario principal:**
    1. El usuario puede pulsar en cualquiera de las siguientes opciones:
        a. Ayuda
        b. Términos de Servicio
        c. Políticas de Privacidad
        d. Versión
    2. La aplicación realizará las siguientes funcionalidades según la opción clickeada por el usuario:
        a. *Opción a), b), o c):* SocialMusFest App abrirá el navegador web por defecto del dispositivo móvil, y re-dirigirá a la página web de este mismo proyecto en su versión web, concretamente a la sección que el usuario haya pulsado.

b. *Opción d):* La aplicación redirigirá al usuario a la ventana de ver versión de la App, donde el usuario podrá visualizar diversas informaciones de la aplicación.

- **Postcondición:** (Ninguna)

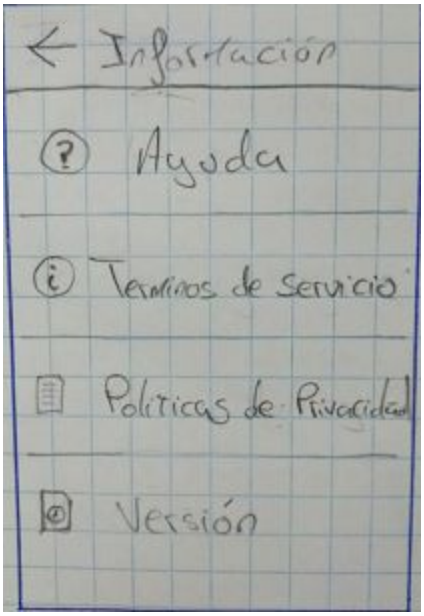


Figura 5.2.4: Prototipo de la ventana "Ver información de la App"

5.2.5. Buscar eventos

Este caso de uso permite al usuario buscar eventos según la localización o ubicación que este desee, además puede añadir al filtro una fecha o día concreto del evento como versión optativa, para ello deberá acceder a la ventana de búsqueda de eventos a través de la opción "Buscar" que se encuentra en el menú lateral.

- **Actor:** Usuario anónimo / Usuario conocida su localización / Usuario registrado
- **Precondición:** El usuario se encuentra en la ventana de búsqueda de eventos.
- **Escenario principal:**

1. El usuario rellena el campo de lugar y fecha(optativo), y después clickea en el botón "Buscar".
2. La aplicación buscará en la base de datos del servidor los eventos que cumplan las mismas condiciones, lugar y fecha del día del evento, una vez hallados cargará en una siguiente página una lista de todos los eventos encontrados para que el usuario pueda verlos.

- **Postcondición:** (Ninguna)

Figura 5.2.5: Prototipo de la ventana“Buscar eventos”

5.2.6. Ver evento

Este caso de uso, un usuario puede visualizar la totalidad de un evento concreto, para poder ver un evento se puede acceder a esta desde la ventana de resultados de “buscar evento” o bien haberlo visto directamente desde la sub-ventana “ver lista de eventos por localización”, la cual se encuentra en la página principal si el usuario tiene dada una localización.

- **Actor:** Usuario anónimo / Usuario conocida su localización / Usuario registrado
- **Precondición:** El usuario se encuentra en la página principal concretamente en la sub-ventana del caso de uso “Ver eventos por localización”, o bien el usuario se encuentra en la página de resultados de buscar eventos.
- **Escenario principal:**
    1. El usuario clickea un evento de la lista de eventos
    2. La aplicación recogerá la acción y obtendrá el objeto Evento seleccionado de la lista, posteriormente cargará ese mismo objeto Evento en una siguiente página, pudiendo ver en esta, la imagen a mejor resolución, las características que conforman al evento y debajo del todo, el botón “Apuntarse” o “Desapuntarse” por si el usuario desea registrarse a este evento o des-registrarse, pero esto sera unicamente si el usuario está registrado, si no, se mostrará como desactivado y se verá un mensaje de aviso.

3. El usuario puede volver atrás

- **Postcondición:** (Ninguna)

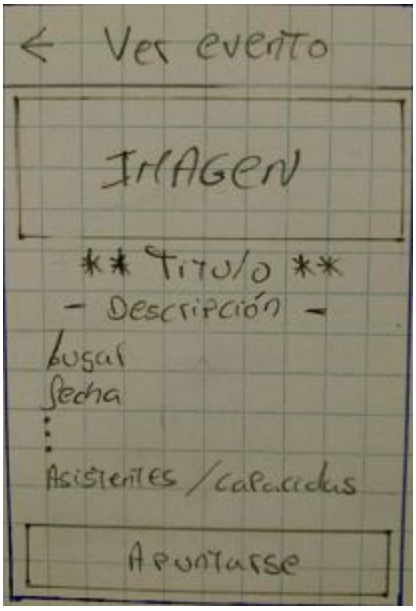


Figura 5.2.6: Prototipo de la ventana“Ver evento”

5.2.7. Actualizar localización

En este caso de uso el usuario puede actualizar su localización, esto se llegaría a realizar si usuario hubiera cambiado de ubicación y quisiera adaptar la en la aplicación, para poder actualizar la localización el usuario deberá nuevamente acceder a la venta de la “localización” como hemos visto en la caso de uso introducir localización del punto 4.1.1, por ello existirá un botón con el icono de actualizar en el menú desplegable lateral junto al texto de su localización actual.

- **Actor:** Usuario conocida su localización / Usuario registrado (conocida su localización)
- **Precondición:** El usuario ya tiene dada una localización previamente
- **Escenario principal:**

    1. El usuario abre el menú desplegable lateral y pulsa en el botón con el icono de actualizar.
    2. La aplicación redirecciona a la venta de localización.
    3. El usuario puede volver atrás

- **Postcondición:** (Ninguna)

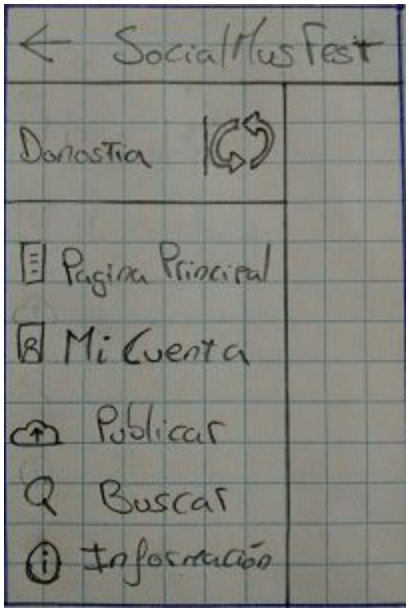


Figura 5.2.7: Prototipo de la ventana "Actualizar localización"

5.2.8. Ver eventos por localización

Este caso de uso permite que se visualice una lista de eventos que concuerden en localización con la ubicación dada por el usuario, esta lista se encuentra en la página principal, sub-ventana "Eventos".

- **Actor:** Usuario con localización conocida / Usuario registrado (con localización conocida)
- **Precondición:** El usuario tiene dada una localización.
- **Escenario principal:**
    1. El usuario puede acceder a la página principal desde el menú, y si ya está en ella, puede deslizarse en el navegador de ventanas hasta la sub-ventana "Eventos".
    2. SocialMusFest App obtiene de la base de datos del servidor los eventos que concuerden con la ubicación dada por el usuario en el caso de uso "Introducir localización".
- **Postcondición:** (Ninguna)

Figura 5.2.8: Prototipo de la ventana "Ver eventos por localización"

5.2.9. Ver eventos publicados

En este caso de uso se le permite al usuario una vez iniciado sesión, ver los eventos que este haya publicado, esta opción se encontrará en el navegador de ventanas en la página principal, sub-ventana "Publicados".

- **Actor:** Usuario registrado
- **Precondición:** El usuario tiene instalada y abierta la aplicación.
- **Escenario principal:**
    1. El usuario puede acceder a la página principal desde el menú desplegable lateral, y si ya está en la página principal puede deslizarse en el navegador de ventanas hasta la sub-ventana "Publicados".
    2. La aplicación obtiene de la base de datos los eventos que el usuario haya publicado y los muestra en la lista de eventos publicados.
- **Postcondición:** (Ninguna)

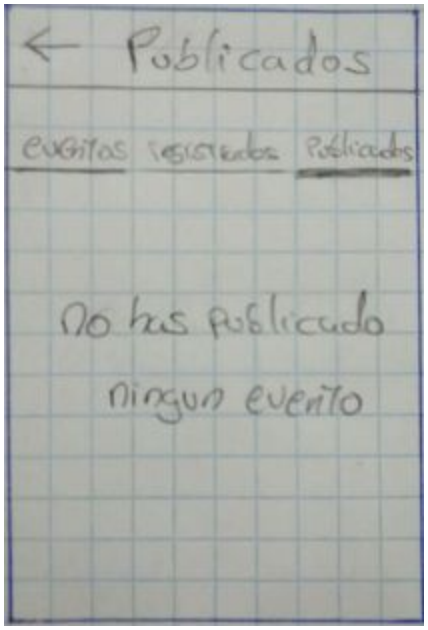


Figura 5.2.9: Prototipo de la ventana “Ver eventos publicados”

5.2.10. Ver eventos registrados

En este caso de uso se le permite al usuario una vez iniciado sesión, ver los eventos en los que se haya registrado o apuntado, esta opción se encuentra en la sub-ventana “Registrados” de la página principal.

- **Actor:** Usuario registrado
- **Precondición:** El usuario tiene instalada y abierta la aplicación.
- **Escenario principal:**
    1. El usuario puede acceder a la página principal desde el menú desplegable lateral, y si ya se encuentra en ella, puede deslizarse en el navegador de ventanas hasta la sub-ventana “Registrados”.
    2. La aplicación obtiene de la base de datos los eventos en los cuales se haya registrado el usuario, después los muestra en la lista de eventos registrados.
- **Postcondición:** (Ninguna)

Figura 5.2.10: Prototipo de la ventana “Ver eventos registrados”

5.2.11. Apuntarse o Desapuntarse

El usuario puede apuntarse/registrarse a un evento, o en su contrario desapuntarse/desregistrarse, para ello debe de acceder a la ventana “ver evento”, que es donde se encuentra el botón para llevarlo a cabo.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la ventana “ver evento” visualizando un evento cualquiera.
- **Escenario principal:**
    1. El usuario se desliza por la pantalla “ver evento” hasta abajo del todo y clickea en el botón “Apuntarse” o “Desapuntarse” según el estado de este.
    2. La aplicación intentará apuntar o desapuntar al usuario de este evento, para apuntarse únicamente se llevará a cabo si el número de asistentes es menor a la capacidad, y para desapuntarse no se requiere ningún requisito. Si el registro o desregistro fuera satisfactorio, el usuario vería actualizado el número de asistentes en las características del evento y el botón de “Apuntarse” o “Desapuntarse” cambiarían de texto y color, siendo apuntarse verde y desapuntarse rojo.
    3. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

5.2.12. Publicar evento

En este caso de uso el usuario puede publicar un evento. En cambio, al verse que es un método largo y complicado, viendo que los formularios requeridos para tal objetivo cuentan con bastante complejidad para que el usuario pudiera interactuar con ellos en pantallas móviles y además de observar que en aplicaciones similares queda ausente este tipo de funcionalidad, se decidió no implementar la publicación de eventos a través de SocialMusFest App, aun así, en lugar de no implementar este caso de uso, se decidió dar un aviso al usuario por si deseaba publicar un evento, en el cual se le informará de que podría realizarlo desde la versión web del proyecto, abriéndole así su navegador web por defecto y redireccionado a este mismo apartado.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la página principal.
- **Escenario principal:**
    1. El usuario despliega el menú desplegable lateral y pulsa en la opción “Publicar”
    2. La aplicación recoge la acción y redirecciona a la ventana de publicar evento
    3. El usuario se informa de cómo puede publicar un evento gracias al texto informativo y pulsa el botón “Publicar”.
    4. La aplicación abre el navegador web por defecto del dispositivo en el que se ejecuta la aplicación y redirecciona a la versión web de este proyecto para que el usuario pueda publicar un evento a través de la página web.
    5. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

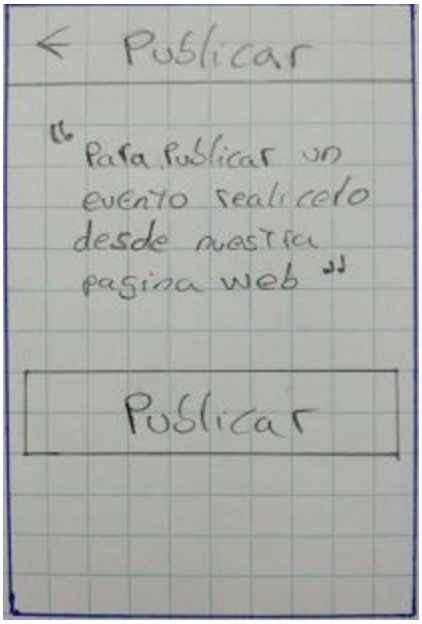


Figura 5.2.12: Prototipo de la ventana “Publicar evento”

5.2.13. Ver perfil

Un usuario con sesión iniciada puede ver su propio perfil, para ello debe de pulsar la opción “Mi Cuenta” del menú desplegable lateral.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la página principal
- **Escenario principal:**
    1. El usuario despliega el menú desplegable lateral, pulsa en la opción “Mi Cuenta” o también puede pulsar en la imagen de perfil que se muestra en el mismo menú.
    2. La aplicación recoge el evento y redirecciona a la página de cuenta personal
    3. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

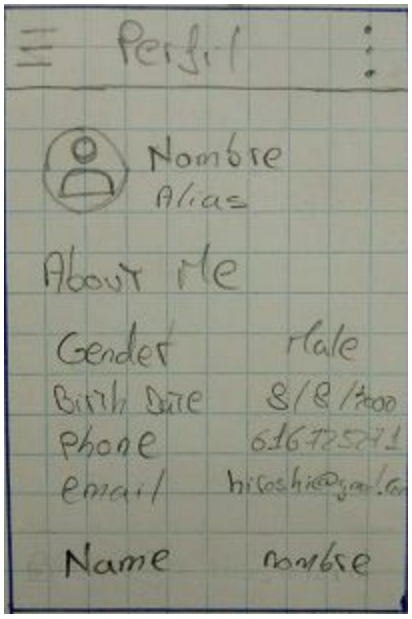


Figura 5.2.13: Prototipo de la ventana “Ver perfil”

5.2.13.1. Actualizar perfil

En este caso de uso un usuario con sesión iniciada puede modificar cualquier campo de su perfil.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la ventana “ver perfil”.
- **Escenario principal:**

    1. El usuario puede decidir modificar cualquiera de los siguientes campos simplemente clickeando en cada opción que se muestre:

        a. Imagen
        b. Nombre de usuario
        c. Email
        d. Nombre y Apellidos
        e. Género
        f. Cumpleaños
        g. Lugar
        h. Estilo de música

    2. La aplicación detecta qué opción desea editar el usuario y muestra una ventana con un formulario para su edición.

    3. El usuario modifica el dato a actualizar y clickea en aceptar

4. La aplicación actualiza la información del dato únicamente si se ha modificado algo, para los campos del nombre de usuario y email además no se pueden dejar vacíos, si se actualiza algún campo con éxito, el usuario puede ver mismamente en la ventana de cuenta personal el campo modificado.
5. El usuario puede volver atrás

- **Postcondición:** (Ninguna)

En la siguiente imagen se muestra como seria la ventana de actualizar el nombre completo en la ventana de actualizar perfil, como todos los campos a modificar tendrán este mismo tipo de ventana se muestra únicamente este.

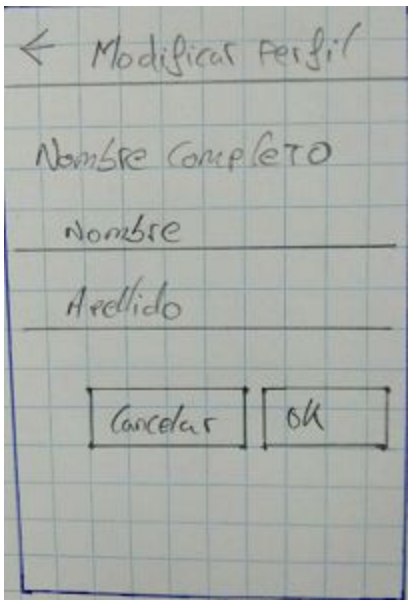


Figura 5.2.13.1: Prototipo de la ventana "Actualizar perfil, nombre completo"

5.2.13.2. Ver opciones

En este caso de uso, el usuario con sesión iniciada puede ver las opciones de cuenta que tienen estas, para poder acceder a la ventana de opciones de cuenta es necesario desplegar el menú derecho que se encuentra en la ventana "ver perfil".

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la página de cuenta personal.
- **Escenario principal:**

.

1. El usuario despliega el menú derecho horizontal y selecciona la opción "Opciones".

2. La aplicación recoge el evento y redirecciona a la página de de opciones.
3. El usuario puede volver atrás

- **Postcondición:** (Ninguna)

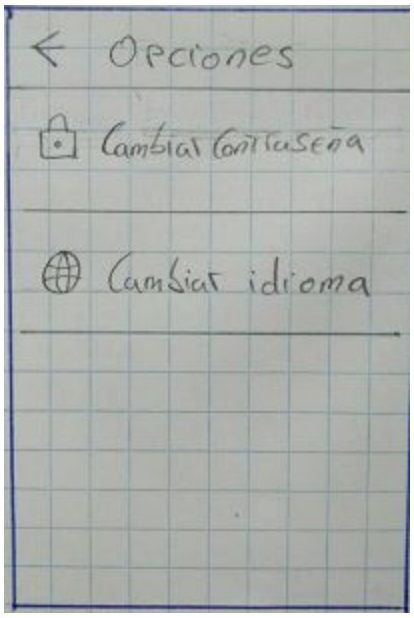


Figura 5.2.13.2: Prototipo de la ventana "Ver opciones"

5.2.13.2.1. Cambiar contraseña

Al igual que un usuario puede actualizar los datos de su cuenta, también puede cambiar la contraseña de esta, para poder acceder a esta funcionalidad debe de clickear en el menú derecho de la página de cuenta personal y seleccionar la opción "Opciones".

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la página de opciones
- **Escenario principal:**
    1. El usuario clickea en el botón "Cambiar contraseña"
    2. La aplicación abre una ventana con un formulario donde el usuario debe de introducir la contraseña actual y la nueva contraseña dos veces para poder cambiarla.
    3. El usuario introduce la contraseña y la nueva contraseña dos veces, después clickea en el botón "Cambiar".
    4. La aplicación actualiza la contraseña y cierra la ventana del formulario.

5. El usuario puede volver atrás

- **Postcondición:** (Ninguna)

Figura 5.2.13.2.1: Prototipo del caso de uso “Cambiar contraseña”

5.2.13.2.2. Cambiar idioma

Un usuario con sesión iniciada puede cambiar el idioma de la aplicación, para ello debe de acceder la página de opciones que se encuentra clickeando en la opción “Opciones” del menú derecho horizontal de la ventana cuenta personal.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la ventana de opciones
- **Escenario principal:**
    1. El usuario clickea en la opción “Cambiar idioma”
    2. La aplicación muestra una ventana con una serie de opciones para elegir, castellano, inglés y euskera.
    3. El usuario clickea en alguna de las opciones (castellano, inglés o euskera)
    4. La aplicación cambia el idioma a la opción seleccionada por el usuario.
    5. El usuario puede volver atrás
- **Postcondición:** (Ninguna)

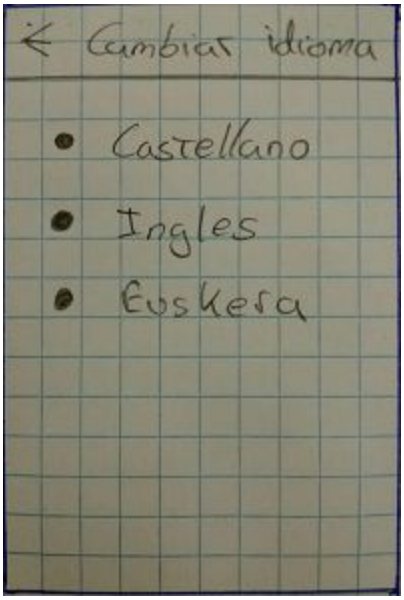


Figura 5.2.13.2.2: Prototipo del caso de uso “Cambiar idioma”

5.2.13.3. Cerrar sesión

En este caso de uso, un usuario con sesión iniciada puede cerrar sesión, esta opción se encuentra en la ventana de cuenta personal del usuario, en el menú derecho horizontal.

- **Actor:** Usuario registrado
- **Precondición:** El usuario se encuentra en la página de cuenta personal
- **Escenario principal:**
    1. El usuario despliega el menú derecho horizontal y clickea en la opción “Cerrar sesión”
    2. La aplicación muestra una ventana de aviso para asegurar si el usuario desea realmente cerrar sesión.
    3. El usuario clickea en el botón “Cerrar sesión”.
    4. La aplicación cierra sesión y redirige al usuario a la página principal.
- **Postcondición:** (Ninguna)

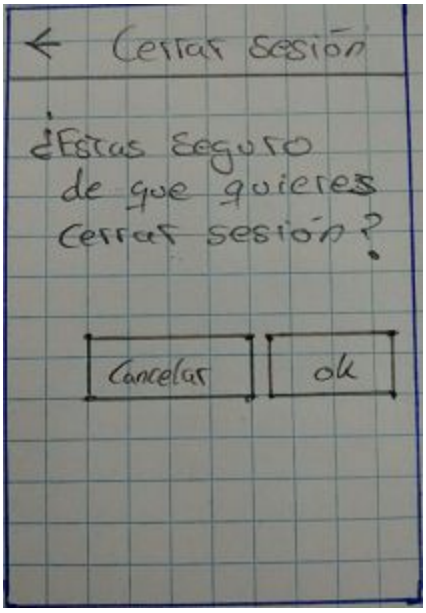


Figura 5.2.13.3: Prototipo del caso de uso “Cerrar sesión”

CAPÍTULO 6.

6. Diseño

En este capítulo se detalla el diseño y la estructura de la aplicación, se describe el diseño de los casos de uso mediante diagramas de secuencia, los diseño de las actividades y el modelo de datos utilizado para el desarrollo de SocialMusFest App.

6.1. Diseño de los casos de uso

En este apartado, se podrá visualizar por cada caso de uso su diagrama de secuencia correspondiente, en los diagramas de secuencia se explica el orden, las respuestas o acciones de la aplicación, los cambios realizados y los casos de error. Cabe mencionar que se ha intentado en lo máximo posible, buscar la simplicidad de las funciones para su mejor entendimiento.

6.1.1. Caso de uso “Introducir o Actualizar localización”

En este diagrama de secuencia se han separado mediante un recuadro azul, las dos formas que tiene un usuario en SocialMusFest App de introducir o actualizar la localización. Se ha añadido la capa de GUI (*Graphical User Interface*) para poder visualizar qué es lo que un usuario necesita clickear, cómo le responde el sistema en caso de introducir o actualizar satisfactoriamente la localización o caso de error, y cómo se realiza la solicitud de activación del GPS.

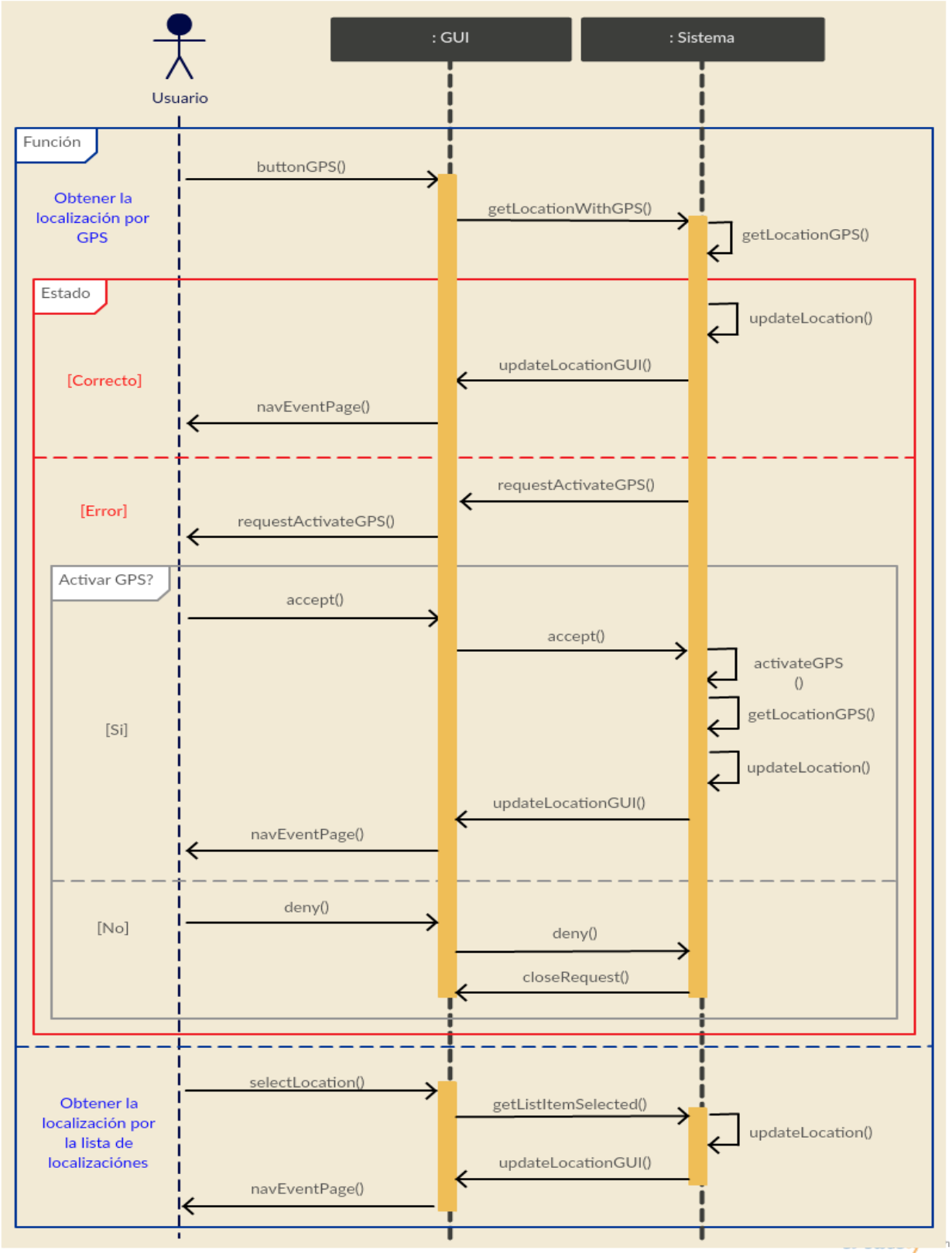


Figura 6.1.1: Diagrama de secuencia de "Introducir o Actualizar localización"

6.1.2. Caso de uso “Registrarse”

En este diagrama de secuencia podemos visualizar cómo se desarrolla en la aplicación, el sistema de creación de cuentas de usuario, en qué bases de datos realiza los *insert* y cómo responde la App tanto en caso éxito a la hora de crear la cuenta como en caso de error.

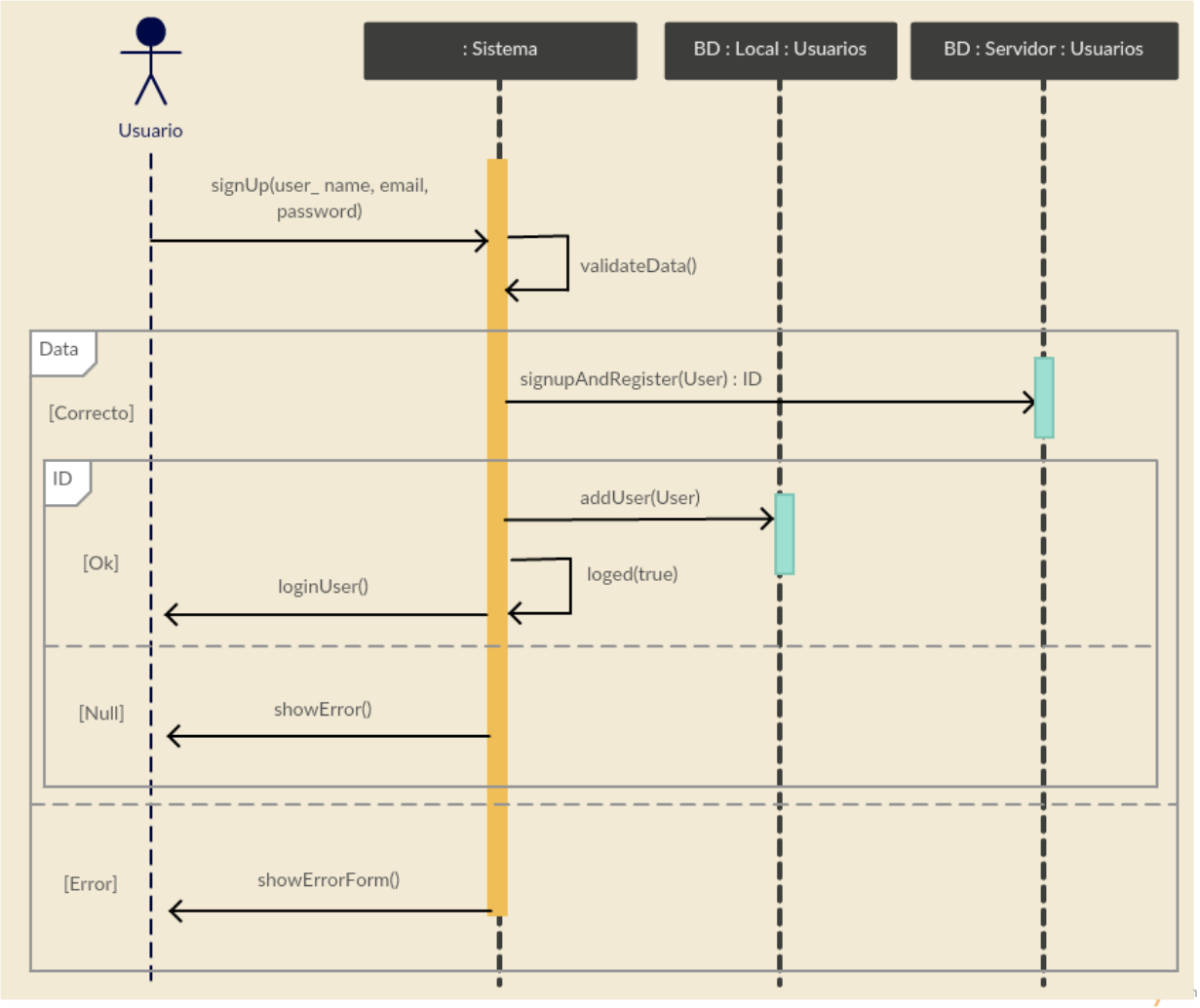


Figura 6.1.2: Diagrama de secuencia de “Registrarse”

6.1.3. Caso de uso “Iniciar sesión”

El siguiente diagrama de secuencia recoge las tres formas que tiene un usuario de iniciar sesión, se han separado con un recuadro azul la forma de logueo mediante formulario, login con Google y login con Facebook, en cada caso se obtiene un objeto *User*, y dependiendo de si este es válido (se ha recuperado los datos del usuario) o este vacío, se da un tipo de respuesta u otra por parte del sistema.

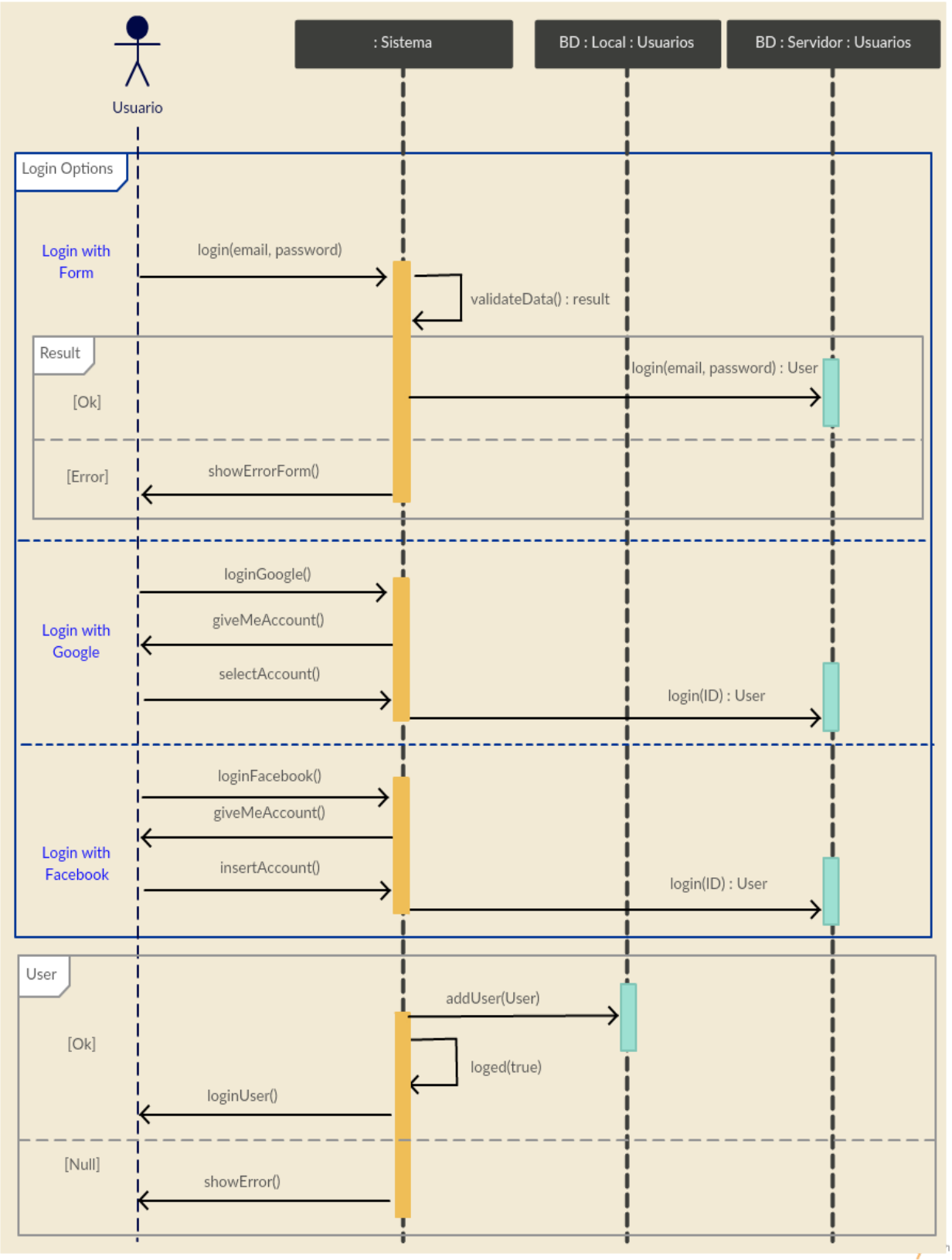


Figura 6.1.3: Diagrama de secuencia de "Iniciar sesión"

6.1.4. Caso de uso “Ver información de la App”

El diagrama de secuencia de la figura 6.1.4 representa el funcionamiento de SocialMusFest App cuando un usuario desea visualizar en la ventana de “Información” las diversas opciones que lo conforman, una vez más se han separado las opciones con un recuadro azul. En estos recuadros está contenido la secuencia de funciones que se desarrollan para el funcionamiento de cada opción.

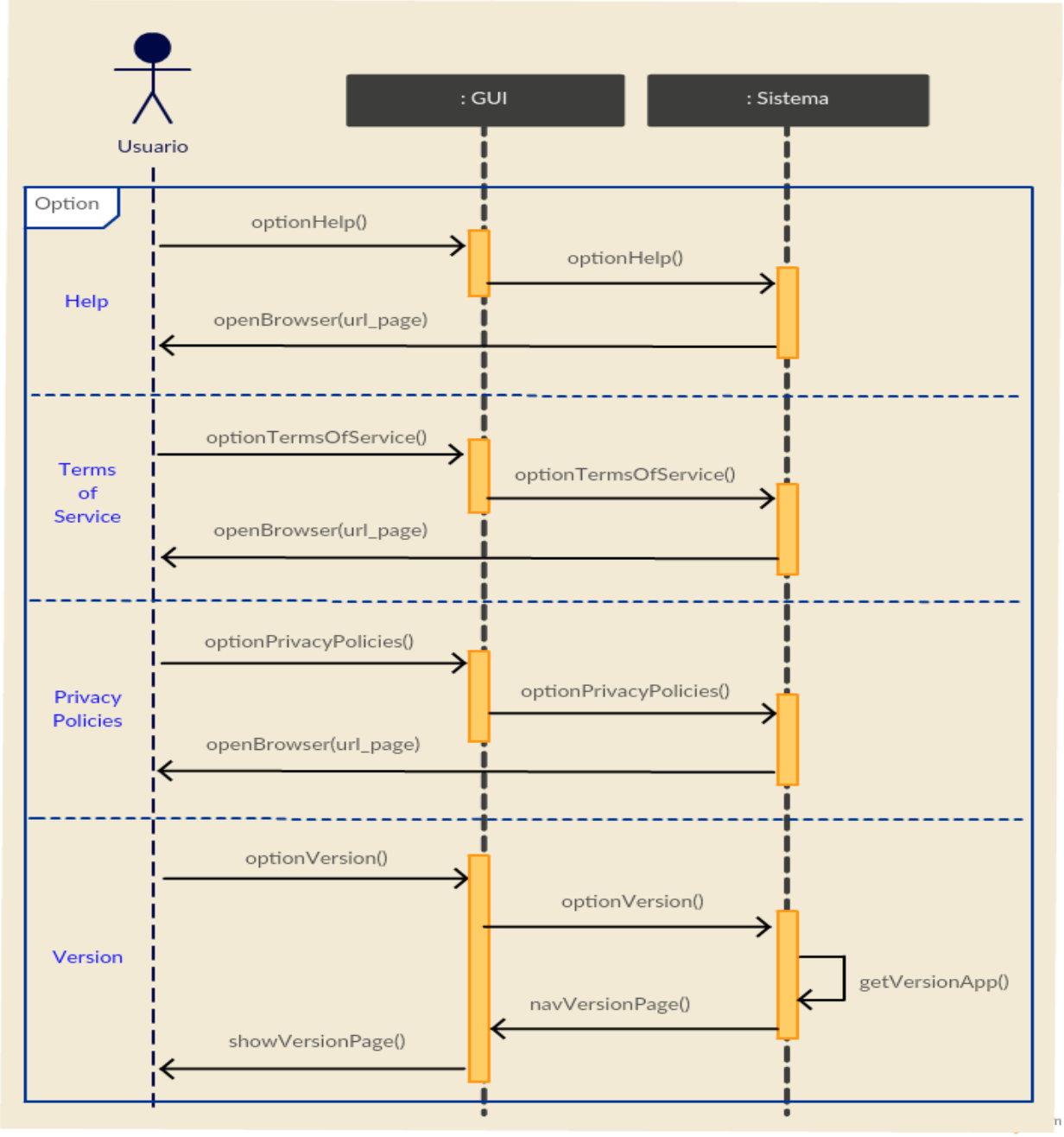


Figura 6.1.4: Diagrama de secuencia de “Ver información de la App”

6.1.5. Caso de uso “Buscar eventos”

El siguiente diagrama de secuencia representa la funcionalidad en SocialMusFest App a la hora de buscar eventos por parte de un usuario.

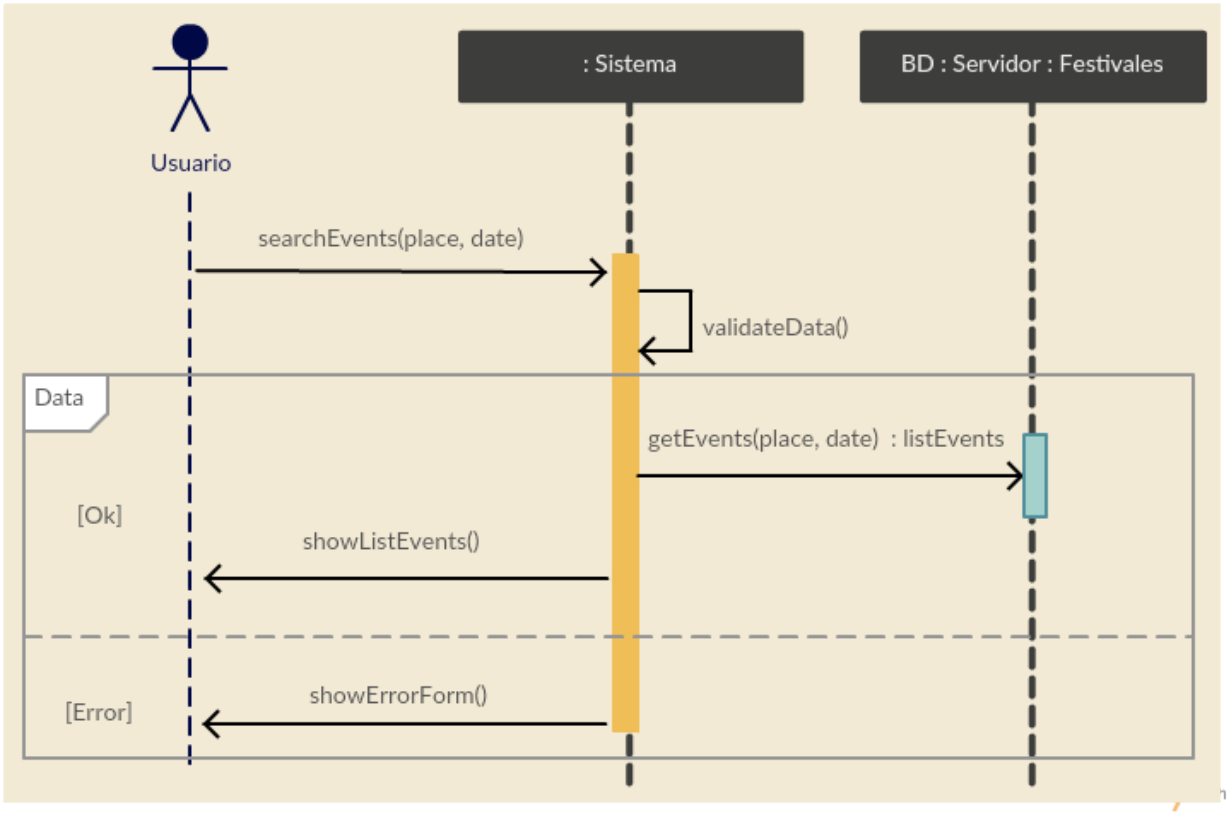


Figura 6.1.5: Diagrama de secuencia de “Buscar eventos”

6.1.6. Caso de uso “Ver evento”

Para ver las características de un evento, un usuario debe clickear el evento deseado a visualizar en la lista de eventos. En el momento en el que se abre la ventana “ver evento”, la aplicación no consulta nuevamente la base de datos servidor, en su lugar como este ya está recogido en la lista de eventos, se recupera de esta. En el siguiente diagrama de secuencia se representa esta misma situación.

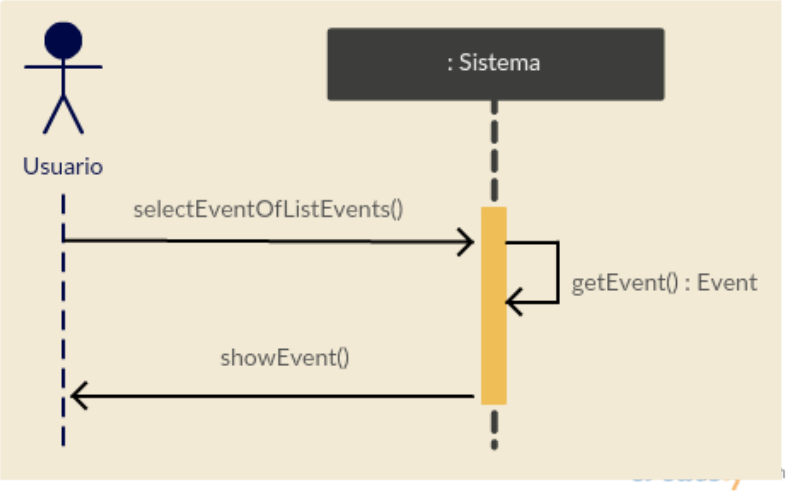


Figura 6.1.6: Diagrama de secuencia de "Ver evento"

6.1.7. Caso de uso "Ver eventos por localización"

Cada vez que se abre SocialMusFest App se cargan las listas de ver eventos por localización, ver eventos publicados y ver eventos registrados de las sub-ventanas "Eventos", "Publicados" y "Registrados", esto dependiendo del estado de usuario (localización conocida o registrado), además de ello, estas ventanas se actualizan cada vez que el usuario navega entre ellas.

En el siguiente diagrama de secuencia se representa la funcionalidad de ver eventos por localización, donde primero se consulta la base de datos local (Usuarios) para obtener la localización, y después utilizamos este dato para obtener todos los eventos de la base de datos servidor (Festivales) que estén localizados en esta misma ubicación.

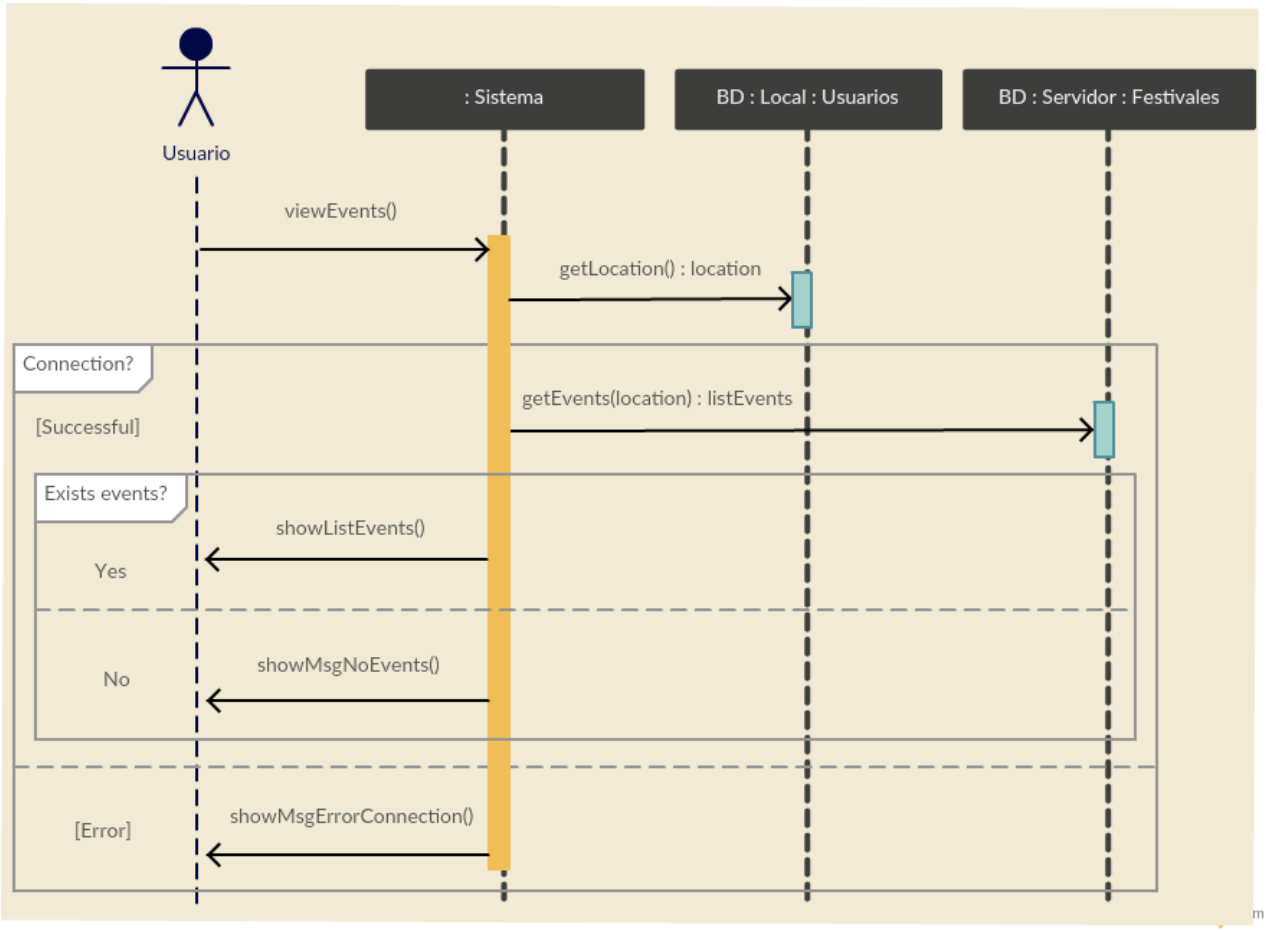


Figura 6.1.7: Diagrama de secuencia de "Ver eventos por localización"

6.1.8. Caso de uso "Ver eventos publicados"

Como se había comentado en el caso de uso "publicar evento", un usuario puede publicar eventos únicamente a través de la página web, aún así, eso no quiere decir que desde la App Android un usuario no pueda visualizar los eventos que haya publicado. Por ello se representa en el siguiente diagrama de secuencia la funcionalidad de SocialMusFest App a la hora de obtener dicha lista de eventos publicados.

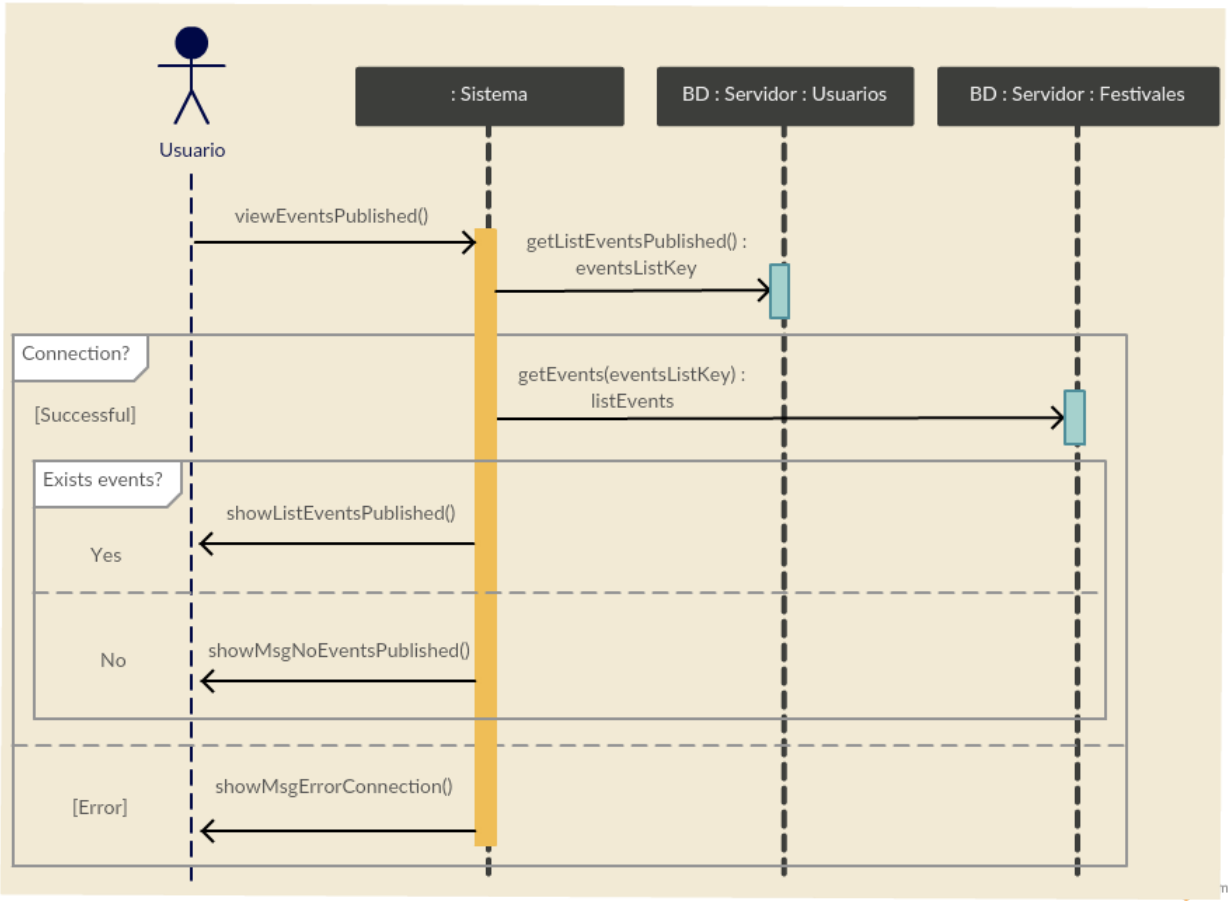


Figura 6.1.8: Diagrama de secuencia de "Ver eventos publicados"

6.1.9. Caso de uso "Ver eventos registrados"

El siguiente diagrama de secuencia es exactamente igual en funcionalidad al anterior, únicamente cambia en la lista de id's (eventos) que obtiene de la base de datos servidor (Usuarios) y en donde visualiza dicha lista. En el anterior ejemplo era la lista de eventos publicados lo que se obtenía, en este caso es la lista de eventos registrados.

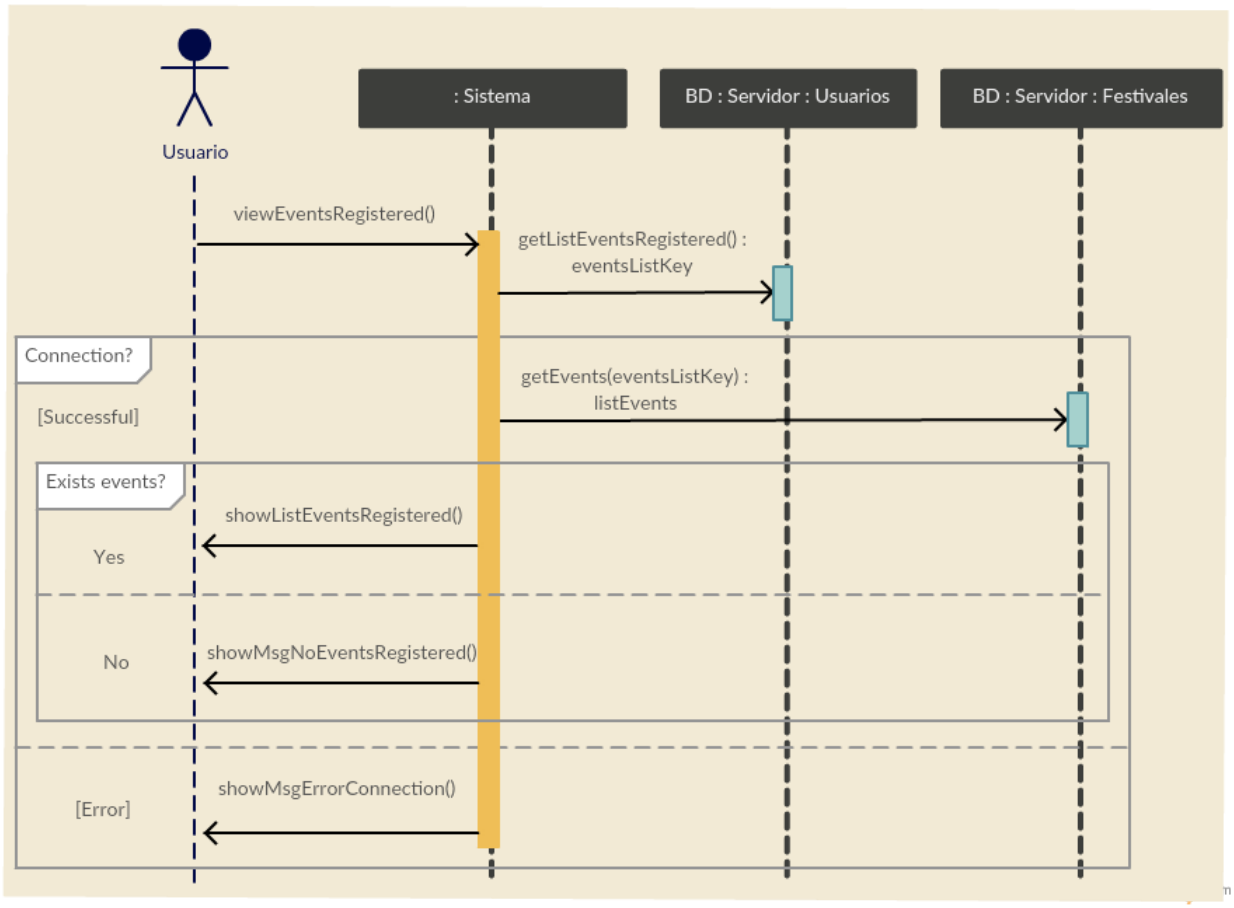


Figura 6.1.9: Diagrama de secuencia de "Ver eventos registrados"

6.1.10. Caso de uso "Apuntarse o Desapuntarse"

En la siguiente figura 6.1.10 podemos visualizar un diagrama de secuencia en el que se han separado con un recuadro azul los diferentes casos de uso "Apuntarse" y "Desapuntarse".

En el primero, antes de registrar al usuario se comprueba que el número de asistentes al evento no es igual a su capacidad, si así fuera se emite un mensaje de error, en caso contrario se incrementa en uno el número de asistentes, se le registra al usuario como evento registrado y se actualiza el estado del botón "Apuntarse" a "Desapuntarse".

En este segundo caso "Desapuntarse" únicamente se decrementa en uno el número de asistentes al evento, se elimina el registro de evento al usuario y se actualiza el estado del botón "Desapuntarse" a "Apuntarse".

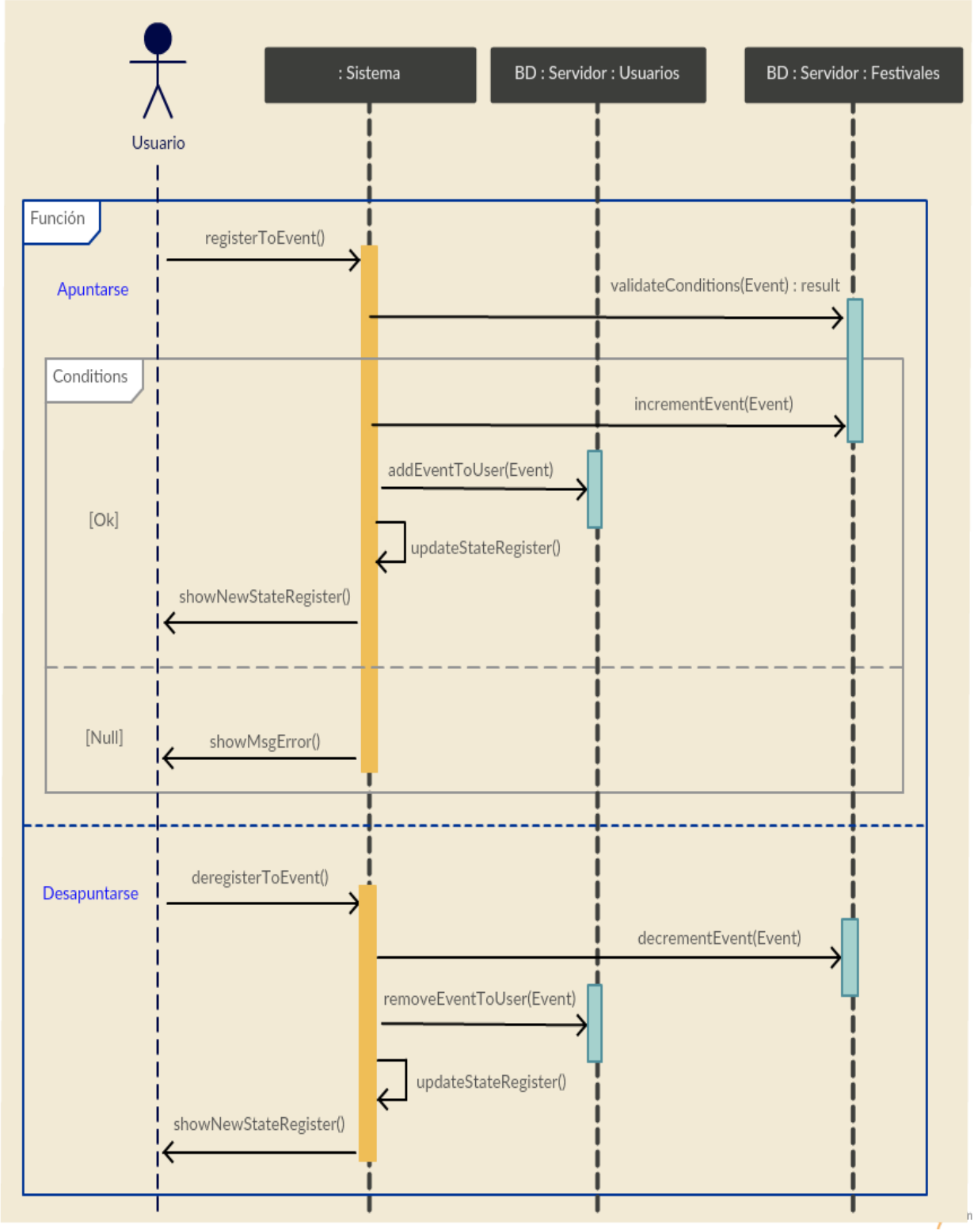


Figura 6.1.10: Diagrama de secuencia de "Apuntarse o Desapuntarse"

6.1.11. Caso de uso “Publicar evento”

En este diagrama de secuencia se ha querido añadir la capa de GUI (*Graphical User Interface*) para visualizar mejor el ejemplo de que SocialMusFest App abre el navegador web por defecto del dispositivo móvil del usuario para redireccionarlo a la página web de este proyecto.

Recordemos que para publicar un evento, un usuario debe realizarlo a través de la página web.

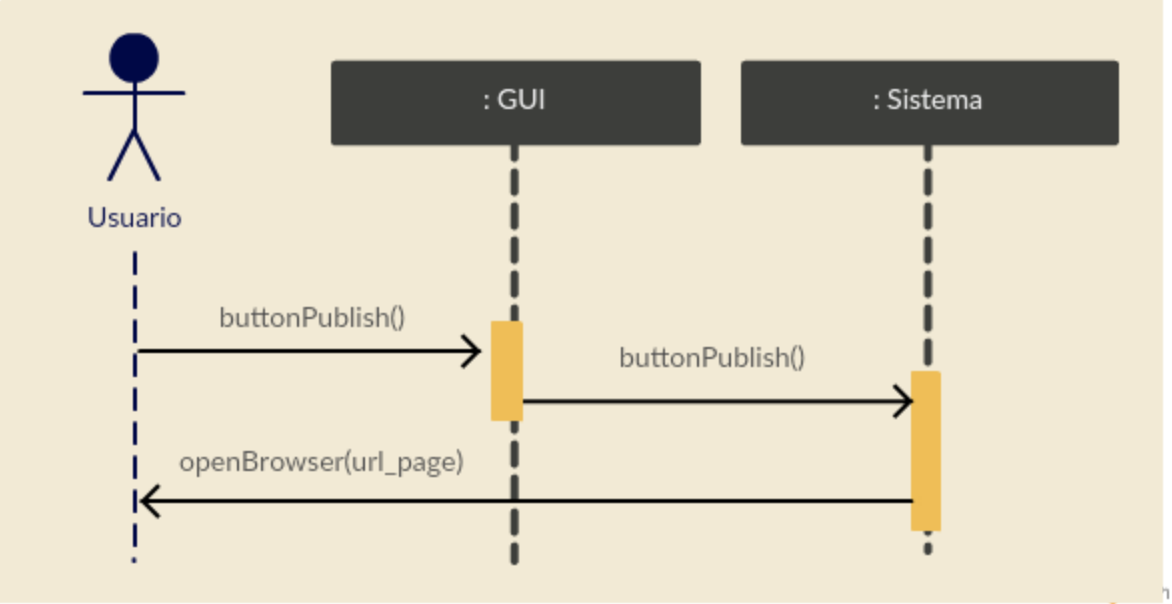


Figura 6.1.11: Diagrama de secuencia de “Publicar evento”

6.1.12. Caso de uso “Ver perfil”

Para ver el perfil de usuario o los datos de cuenta personales, el usuario debe de acceder a la ventana “perfil” a través del menú desplegable lateral, este menú se puede pulsar en la imagen de perfil que se carga dinámicamente cuando este inicia sesión o directamente en el botón “Mi Cuenta”.

Los datos al estar cargados en el sistema a través de la clase MyState simplemente necesita cargar los en la ventana “perfil”.

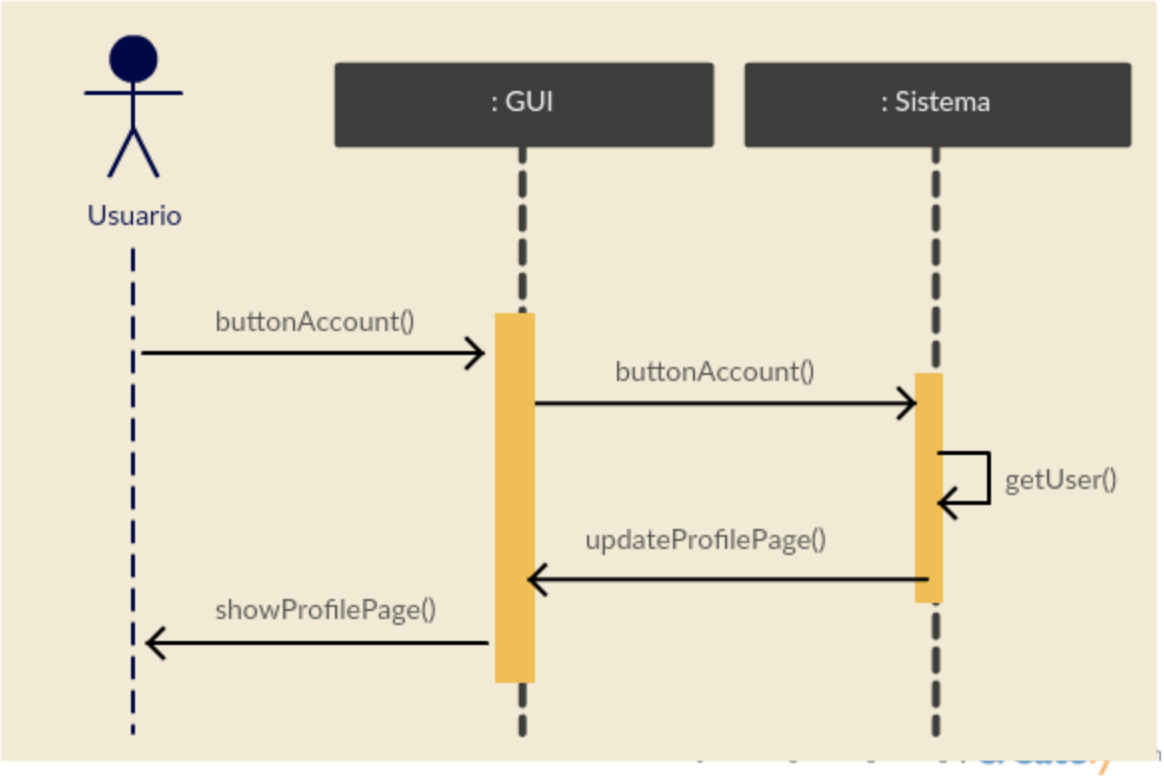


Figura 6.1.12: Diagrama de secuencia de "Ver perfil"

6.1.12.1. Caso de uso "Actualizar perfil"

En la ventana "ver perfil" además de visualizar los datos de cuenta y los datos personales, podemos modificar o actualizar directamente estos datos en esta ventana. Al clickear en alguno de esos datos nos aparece una ventana de diálogo que se visualizará según las características del dato a modificar. Un dato importante a la hora de modificar el perfil, es que si no tenemos conexión con el servidor no se podrá realizar modificación alguna, mostrándose un mensaje de error explicativo del motivo al usuario.

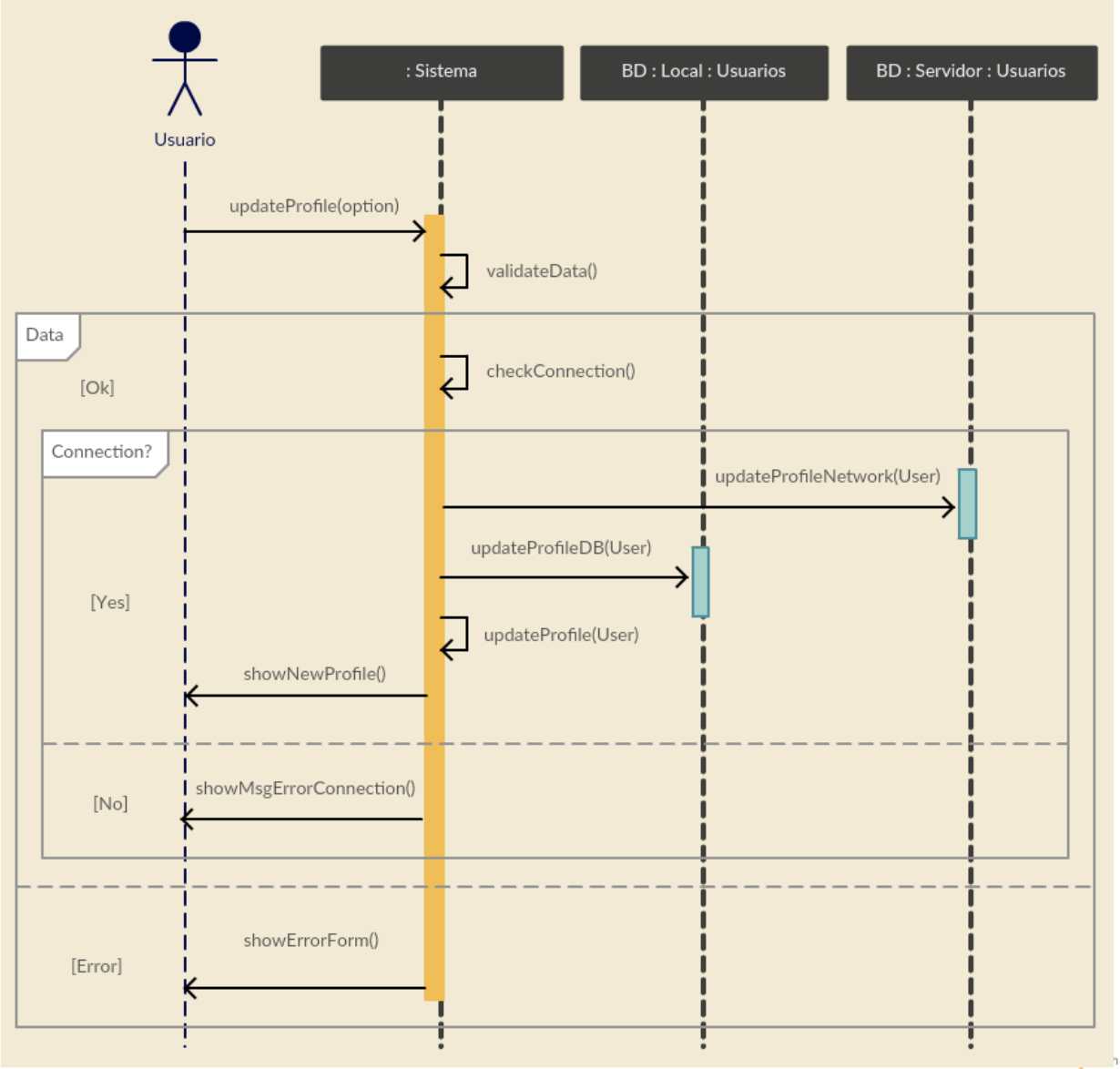


Figura 6.1.12.1: Diagrama de secuencia de "Actualizar perfil"

6.1.12.2. Caso de uso "Cambiar contraseña"

Para cambiar la contraseña de cuenta es necesario la introducción de la contraseña actual y la nueva contraseña por motivos de seguridad, ambas no pueden estar vacías y tienen que ser mayores a cuatro caracteres. Si la validación de estos datos es correcta y tenemos conexión, se modifica la contraseña en el servidor y se informa al usuario del cambio. En caso contrario se muestra un mensaje de error según el motivo de este.

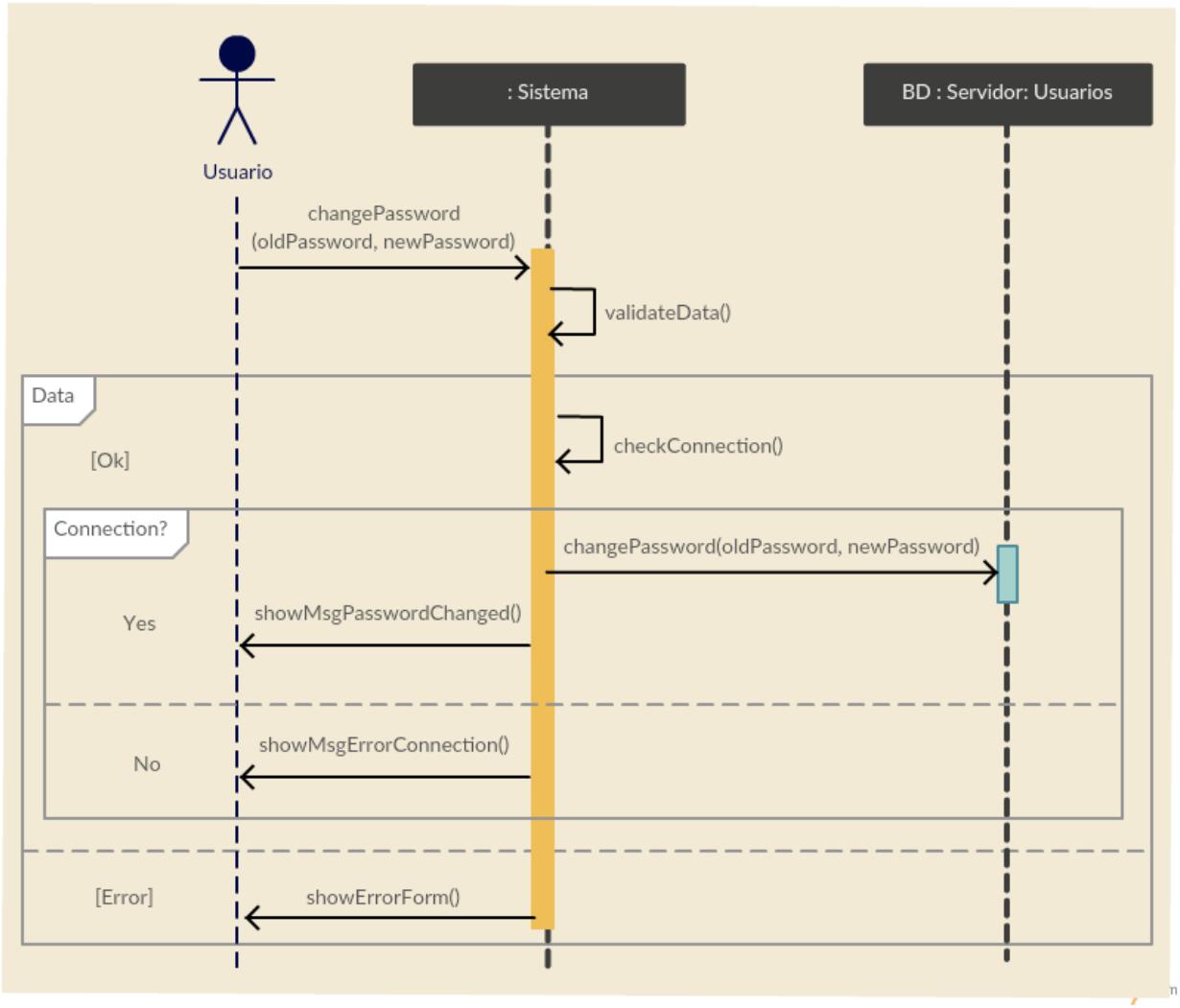


Figura 6.1.12.2: Diagrama de secuencia de "Cambiar contraseña"

6.1.12.3. Caso de uso "Cambiar idioma"

SocialMusFest App esta traducida a tres idiomas: Castellano, Inglés y Euskera.

Para cambiar el idioma de la aplicación es necesario que el usuario tenga sesión iniciada, desde la ventana "opciones" existe el botón "cambiar idioma" que abre una ventana de diálogo para seleccionar el idioma deseado, el sistema cargará el archivo .xml correspondiente donde se encuentren todos los textos de las frases y palabras que se muestran en la aplicación, y devolverá al usuario el poder de acción para comprobar que efectivamente el cambio fue realizado con éxito.

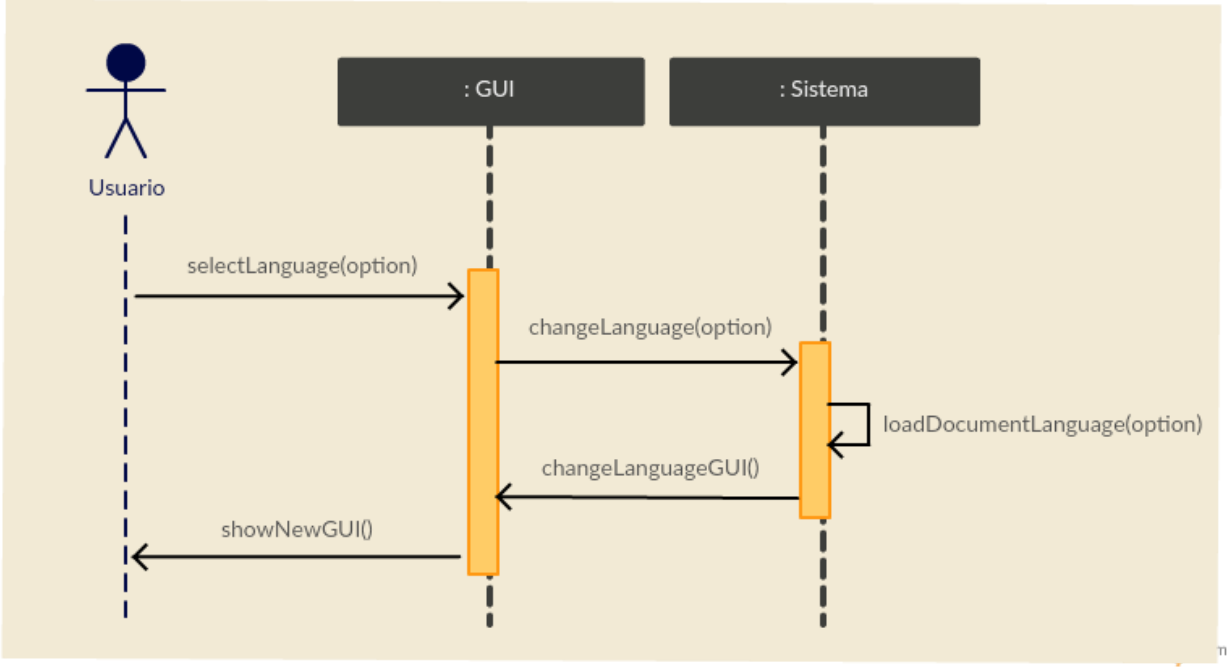


Figura 6.1.12.3: Diagrama de secuencia de "Cambiar idioma"

6.1.12.4. Caso de uso "Cerrar sesión"

En el diagrama de secuencia que se puede observar en la figura 6.1.12.4, representa el funcionamiento de la App SocialMusFest al cerrar sesión un usuario. Como podemos ver, el sistema pregunta al usuario si este esta seguro de cerrar sesión a través de una ventana de diálogo, si el usuario acepta, se elimina la sesión en el sistema, se elimina su cuenta de la base de datos local y se borra el objeto *User* del sistema para posteriormente redirigirlo a la página principal, en caso contrario, únicamente se cierra la ventana de duda.

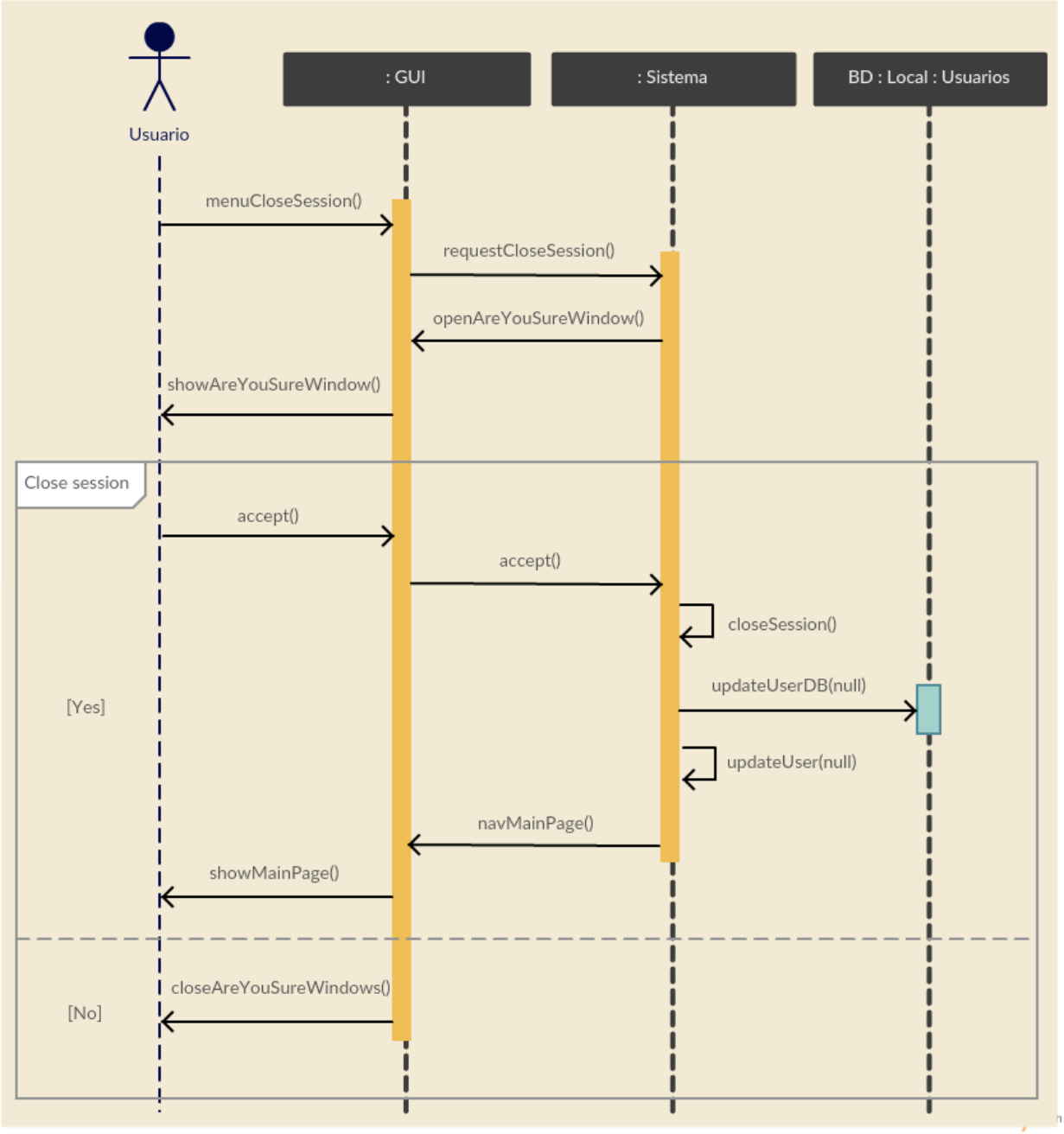
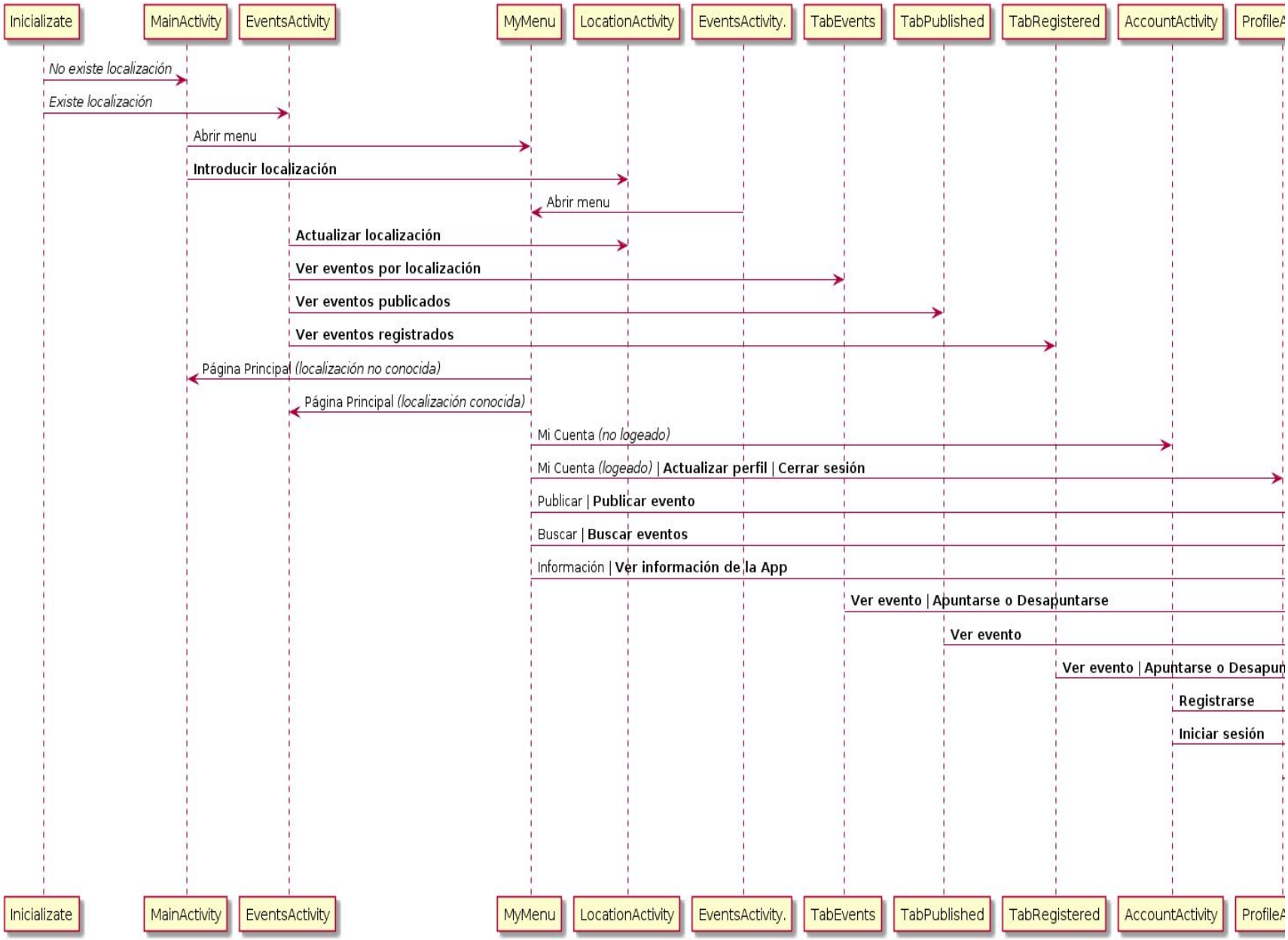


Figura 6.1.12.4: Diagrama de secuencia de "Cerrar sesión"

6.2. Diseño de las actividades

En este punto se visualizan las clases (ventana) realizadas para el desarrollo de la aplicación y la relación entre ellas. En la siguiente figura 6.2 podemos visualizar como queda el diseño de las actividades planteadas en SocialMusFest App, como podemos ver, los casos de uso han sido resaltados en negrita, las condiciones según sea el caso para la ejecución de una ventana u otra

queda resaltada en cursiva, y las opciones que puede realizar el usuario se encuentran descritas en texto normal.


Inicializate
MainActivity
EventsActivity
MyMenu
LocationActivity
EventsActivity.
TabEvents
TabPublished
TabRegistered
AccountActivity
ProfileA
No existe localización
Existe localización
Abrir menu
Introducir localización
Abrir menu
Actualizar localización
Ver eventos por localización
Ver eventos publicados
Ver eventos registrados
Página Principal (localización no conocida)
Página Principal (localización conocida)
Mi Cuenta (no logeado)
Mi Cuenta (logeado) | Actualizar perfil | Cerrar sesión
Publicar | Publicar evento
Buscar | Buscar eventos
Información | Ver información de la App
Ver evento | Apuntarse o Desapuntarse
Ver evento
Ver evento | Apuntarse o Desapun
Registrarse
Iniciar sesión

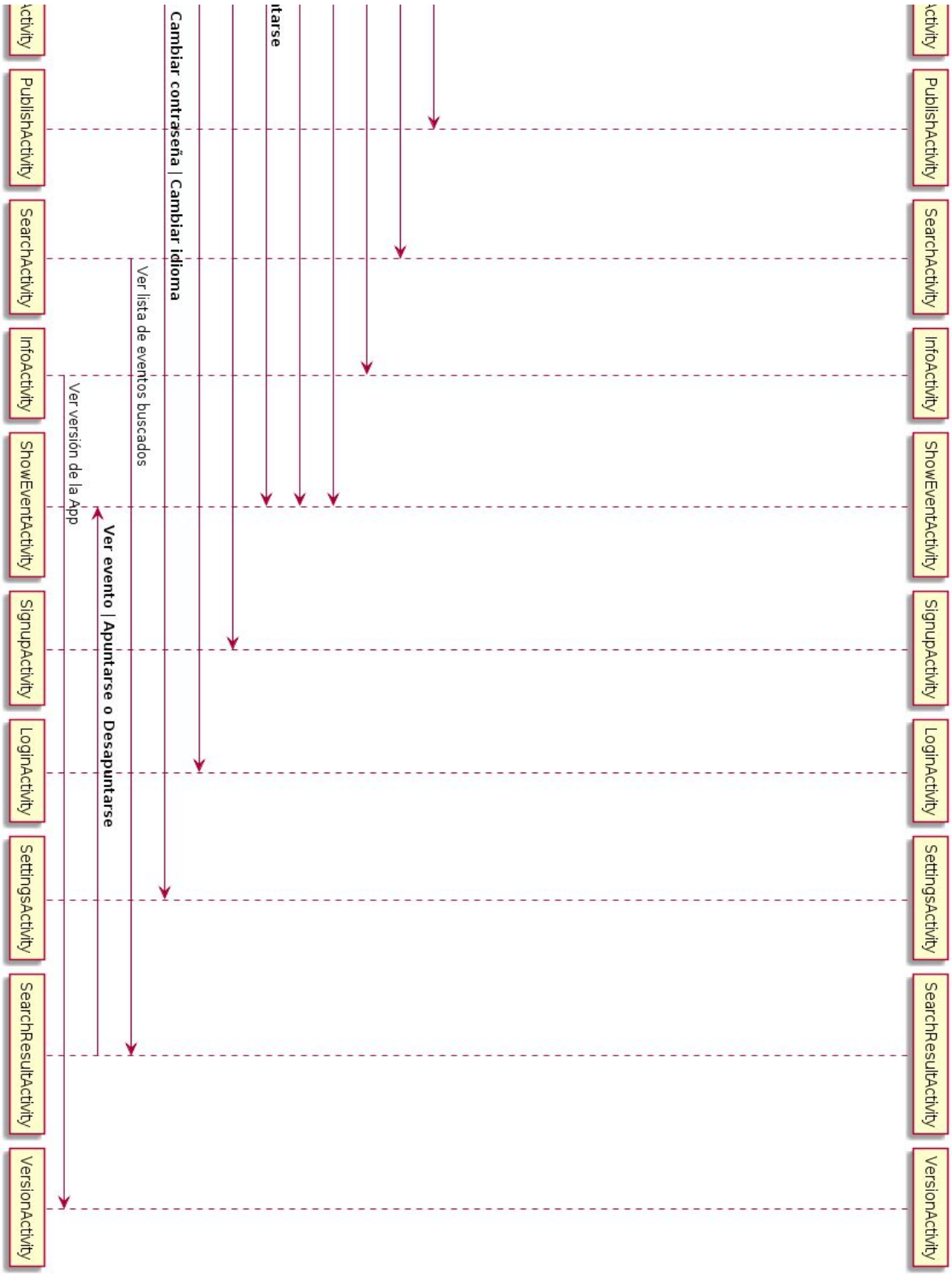


Figura 6.2: Diseño de las Actividades

6.3. Diseño del modelo de datos

En este apartado, podemos ver la tabla “Usuarios” de la base de datos local implementada con SQLite, y las colecciones (en MongoDB a las tablas se les llaman colecciones) “Usuarios” y “Festivales” de la base de datos servidor implementada en MongoDB.

Recordemos, esta aplicación esta desarrollada en Android Studio y la página web en Meteor. Para poder realizar la comunicación desde esta App Android con la base de datos servidor, necesitamos comunicarnos con Meteor a través de su API para que nos permita el acceso a sus recursos (*DB*).

En SocialMusFest App, se guardan los datos de un usuario tanto en una base de datos local como en una base de datos servidor, en cambio los datos de los festivales o conciertos solo se encuentran almacenados en la base de datos de lado servidor.

La base de datos local es relacional, siendo utilizada la versión SQLite de Android, en cambio, del lado servidor se utiliza una base de datos no relacional, siendo MongoDB la tecnología elegida para el desarrollo de la versión web de este proyecto.

De los usuarios guardamos todos los datos a excepción de la contraseña, la cual solo existe en la base de datos servidor bajo cifrado propio de las funcionalidades de Meteor. Se decidió guardar los datos del usuario también en local para evitar sobrecargas en el servidor ya que tendríamos que recuperar los datos del usuario cada vez que este cierre o abra la aplicación, además conseguimos ganar mayor velocidad de carga, y asimismo, permitimos al usuario poder seguir viendo su perfil aún sin estar conectado a la red, obviamente sin posibilidad de actualizar ningún dato, a menos que decida conectarse a una red.

A la hora de decidir qué campos contendrían los usuarios y los festivales además de que tipo de dato sería el que lo representará, fue la sincronización entre ambos autores (Rachid Boudhar Tannaoui y Jon Junguitu Iturrospe) para que ambas aplicaciones (versión Web y versión Android) pudieran compaginar, siendo la mayor parte de estos en formato String o Text para evitar problemas de compatibilidad, por ello las bases de datos se visualizan de la siguiente manera:

Tabla **Usuarios** (**local**): En esta base de datos guardamos la id del usuario y los datos personales de este, como los son el email, nombre de usuario, nombre, apellido, genero, año de nacimiento, lugar, estilo de música, imagen de perfil, id de Google y el id de Facebook, además, también guardamos en la misma tabla el campo localización que representa la localización del usuario o del dispositivo móvil. Como hemos comentado anteriormente la contraseña queda excluida de esta base de datos.

Colección **Usuarios** (**servidor**): Esta base de datos es más completa ya que además de contar con los datos que se guardan en local a excepción de la localización, tiene añadidos para que las funcionalidades de las aplicaciones (Web y Android) funcionen correctamente. Estos datos añadidos son la lista de eventos publicados por el usuario y la lista de eventos en los que se ha registrado un usuario.

Colección **Festivales:** Aquí guardamos los eventos que representan festivales o conciertos de música. Están compuestos por el id de evento, nombre, descripción, lugar, día inicio, día fin, capacidad, número de asistentes, foto, página web donde podemos obtener las entradas, página web oficial del encargado del evento, número de contacto de este, y para finalizar el nombre del usuario que ha creado dicho evento.

Podemos visualizar en la figura 6.3. el diseño de las tablas y colecciones, recordemos que aunque aquí podemos ver una serie de relaciones, estas han sido implementadas del lado servidor en una base de datos no relacional.

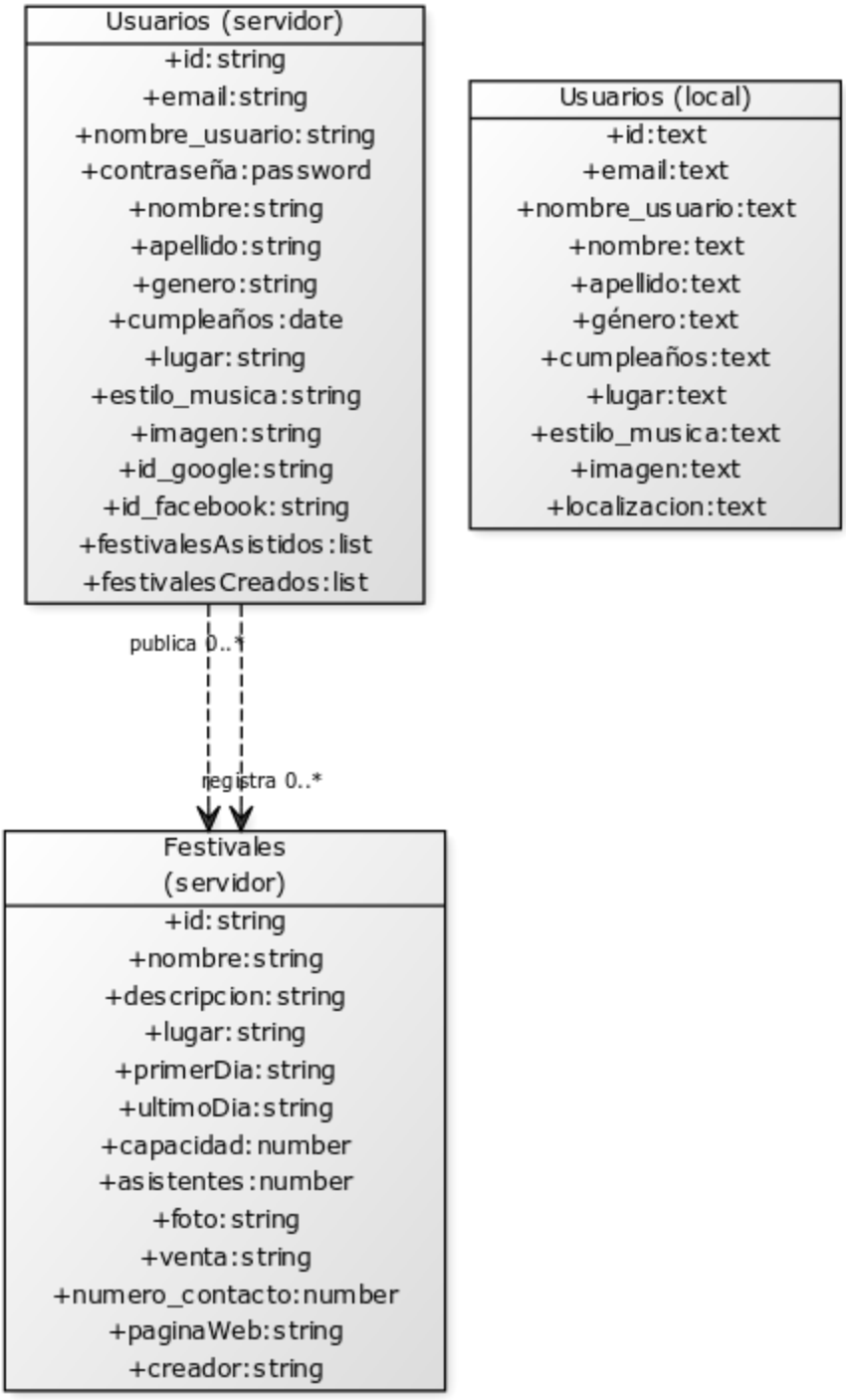


Figura 6.3: Diseño del modelo de datos

CAPÍTULO 7.

7. Implementación

En este capítulo de la memoria, se da una breve introducción a las características de Android en cuanto al desarrollo se refiere, este desglose de explicaciones se realiza para que después se pueda conocer cómo han sido realizadas las implementaciones y de qué forma se decidió desarrollar cada funcionalidad.

A continuación, se explican los diferentes diseños de interfaces de usuario (según el estado de este), la estructura de nuestra aplicación y las sección que la componen. Se detalla el uso de APIs, el archivo Manifest, los archivos .java desarrollados entre ellos las actividades y las clases, los archivos .xml que representan las ventanas de interfaz, los ficheros de recursos y finalmente el uso de librerías externas.

7.1. Entendiendo las características de Android

En este punto se explica que es una actividad, su funcionamiento y su ciclo de vida, cómo funcionan las ventanas, como está gestionado y estructurado el sistema de ficheros para que se pueda entender como se ha realizado la implementación de la App en los siguientes puntos.

7.1.1. Actividad

Llamamos actividad o en inglés *Activity*, al conjunto de dos archivos (java y xml) que llegan a definir la funcionalidad o la actividad que se realiza en una ventana de la aplicación, así mismo si tenemos múltiples actividades podemos definir un flujo de eventos entre ellas añadiendo funcionalidad a la aplicación, cada actividad tiene su propio ciclo de vida independientemente de otras, y gracias la distinción de fases que las caracteriza, podemos definir que hace una actividad según la fase en la que se encuentre, más tarde definiremos el ciclo de vida de una actividad y se desglosarán sus fases.

Como se ha comentado anteriormente, las actividades están compuestas por dos partes: la parte lógica (java) y la parte gráfica (xml).

- **Parte gráfica:** Este es un archivo .xml donde se definen los elementos interactivos hacia el usuario, botones, textos, listas, desplegables… etc, cada una de las pantallas o vistas que forman nuestra aplicación, está realizada en este formato y de esta forma.

- **Parte lógica:** Es una archivo .java en el cual se representa la clase que se crea para poder manipular, interactuar y colocar el código que define las funciones que tendrán los elementos interactivos diseñados en la parte gráfica.

7.1.2. Ciclo de vida de una actividad

En la figura 7.1.3 podemos visualizar que es el ciclo de vida de una actividad, como están las partes del ciclo de vida separadas y a que hace referencia cada estado en el que se encuentra.

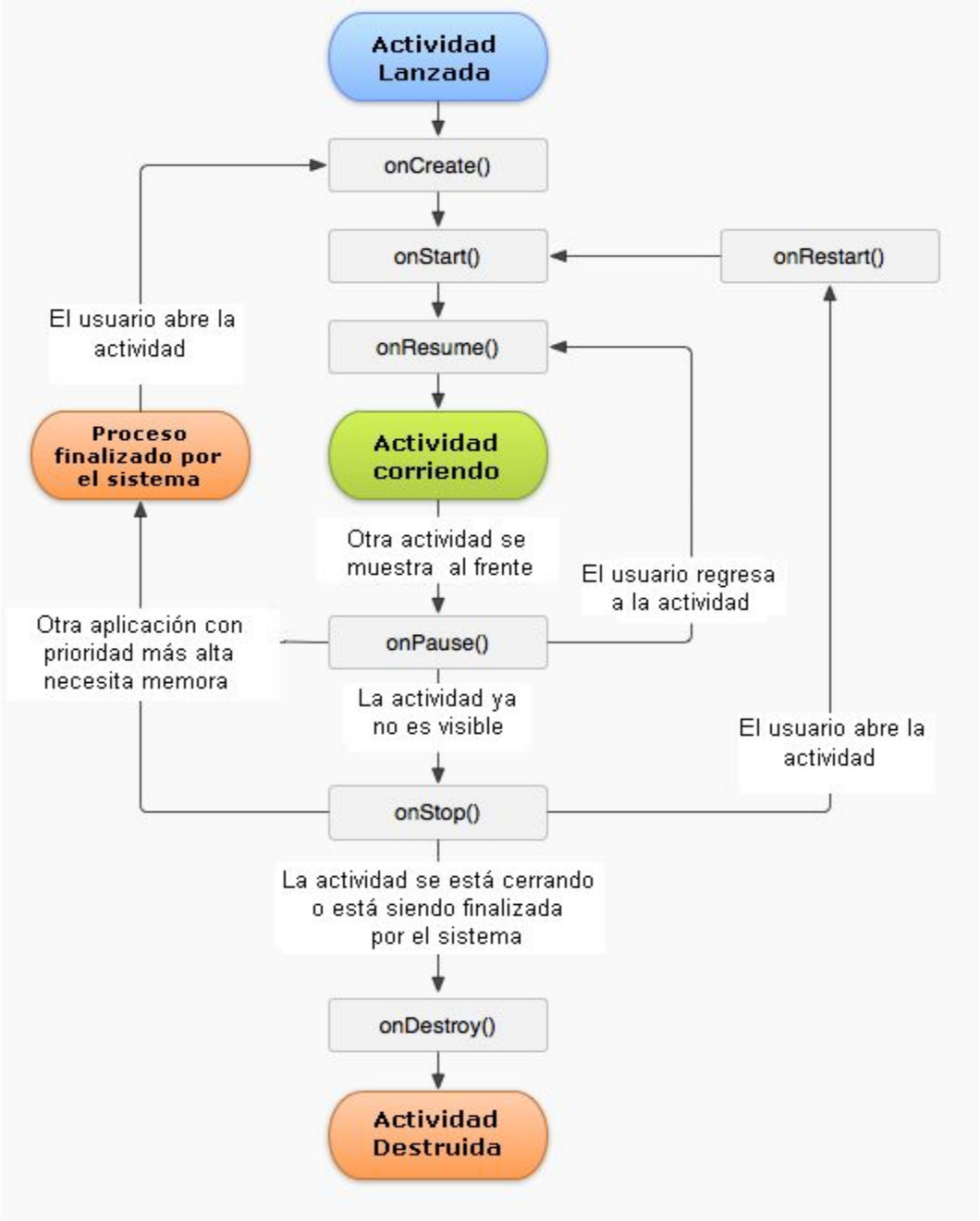


Figura 7.1.2: Ciclo de vida de una Actividad

A continuación, se explican cada uno de los estados en los que se puede encontrar una actividad:

- **onCreate():** Se dispara cuando la *Activity* es llamada por primera vez. Aquí es donde debemos crear la inicialización normal de la aplicación, crear vistas, hacer los *bind* de los datos, etc. Este método te da acceso al estado de la aplicación cuando se cerró. Después de esta llamada siempre se llama al estado onStart().
- **onRestart():** Se ejecuta cuando tu *Activity* ha sido parada, y quieres volver a utilizarla. Si ves el diagrama podrás ver que después de un onStop() se ejecuta el onRestart() e inmediatamente llama a un onStart().
- **onStart():** Se ejecuta cuando la *Activity* se está mostrando apenas en la pantalla del dispositivo del usuario.
- **onResume():** Se ejecuta una vez que la Activity ha terminado de cargarse en el dispositivo y el usuario empieza a interactuar con la aplicación. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause().
- **onPause():** Se ejecuta cuando el sistema arranca una nueva *Activity* que necesitará los recursos del sistema centrados en ella. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva *Activity*. Después de esta llamada puede venir un onResume() si la *Activity* que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario.
- **onStop():** Se ejecuta cuando la *Activity* ya no es visible para el usuario porque otra *Activity* ha pasado a primer plano. Si vemos el diagrama, después de que se ha ejecutado este método nos quedan tres opciones: ejecutar el onRestart() para que la *Activity* vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieren memoria o ejecutar el onDestroy() para apagar la aplicación.
- **onDestroy():** Esta es la llamada final de la *Activity*, después de ésta, es totalmente destruida. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si quisiéramos volver a ejecutar la *Activity* se arrancaría un nuevo ciclo de vida.

7.2. Interfaces de usuario

A continuación, se pueden visualizar los diferentes diseños de interfaces según el rol del usuario (anónimo, localización conocida y registrado).

En la figura 7.2.2 observamos que el menú tiene una cabecera frente al estilo clásico de menú de la figura 7.2.1, en esta podemos visualizar la localización y el botón para actualizar. Una vez que

el usuario introduce la localización también podemos visualizar la lista de eventos próximos que están en los alrededores.

En la figura 7.2.1 podemos ver los diseños de interfaces básicos en los cuales el usuario no está registrado ni se tiene conocida su localización, en la ventana principal, el usuario puede introducir la localización por primera vez y el menú no tiene ninguna cabecera.

Finalmente, en la figura 7.2.3 podemos visualizar el diseño de interfaces de un usuario registrado, se añaden nuevas ventanas donde el usuario además de poder ver la lista de eventos que haya publicado o en los que se haya registrado, también puede acceder a su perfil, modificar sus datos, ver las opciones para cambiar la contraseña o el idioma de la aplicación y cerrar sesión si lo requiere. Como se puede observar el menú es mucho más completo e informativo.

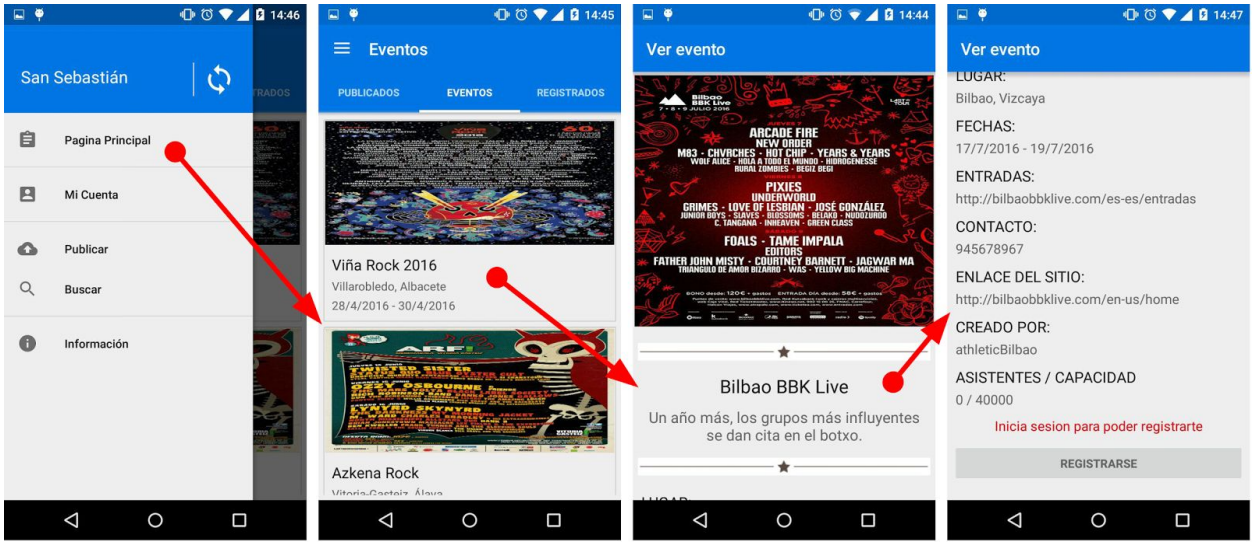


Figura 7.2.2: Diseño de interfaces (usuario con localización conocida)

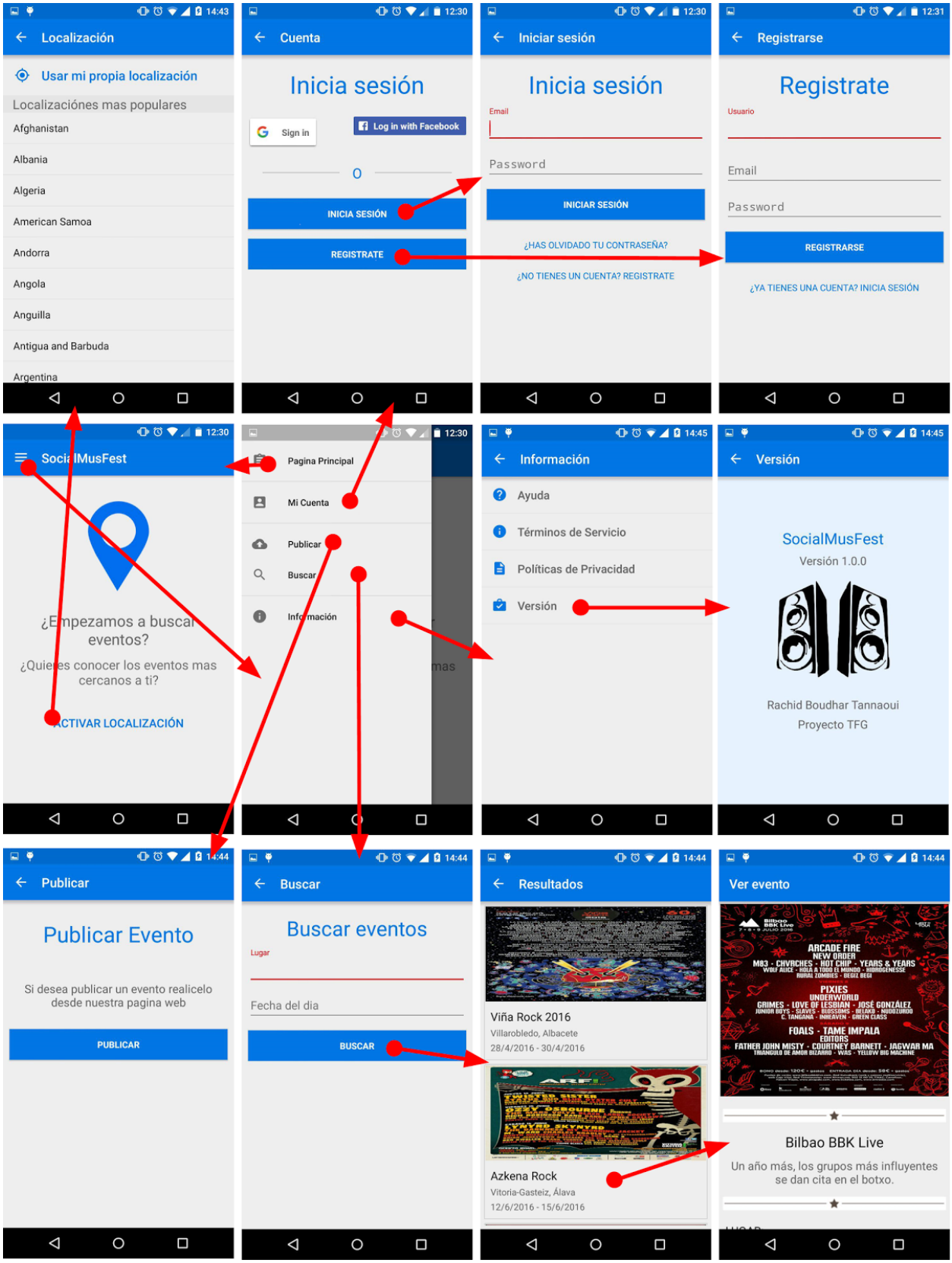


Figura 7.2.1: Diseño de interfaces (usuario anónimo)

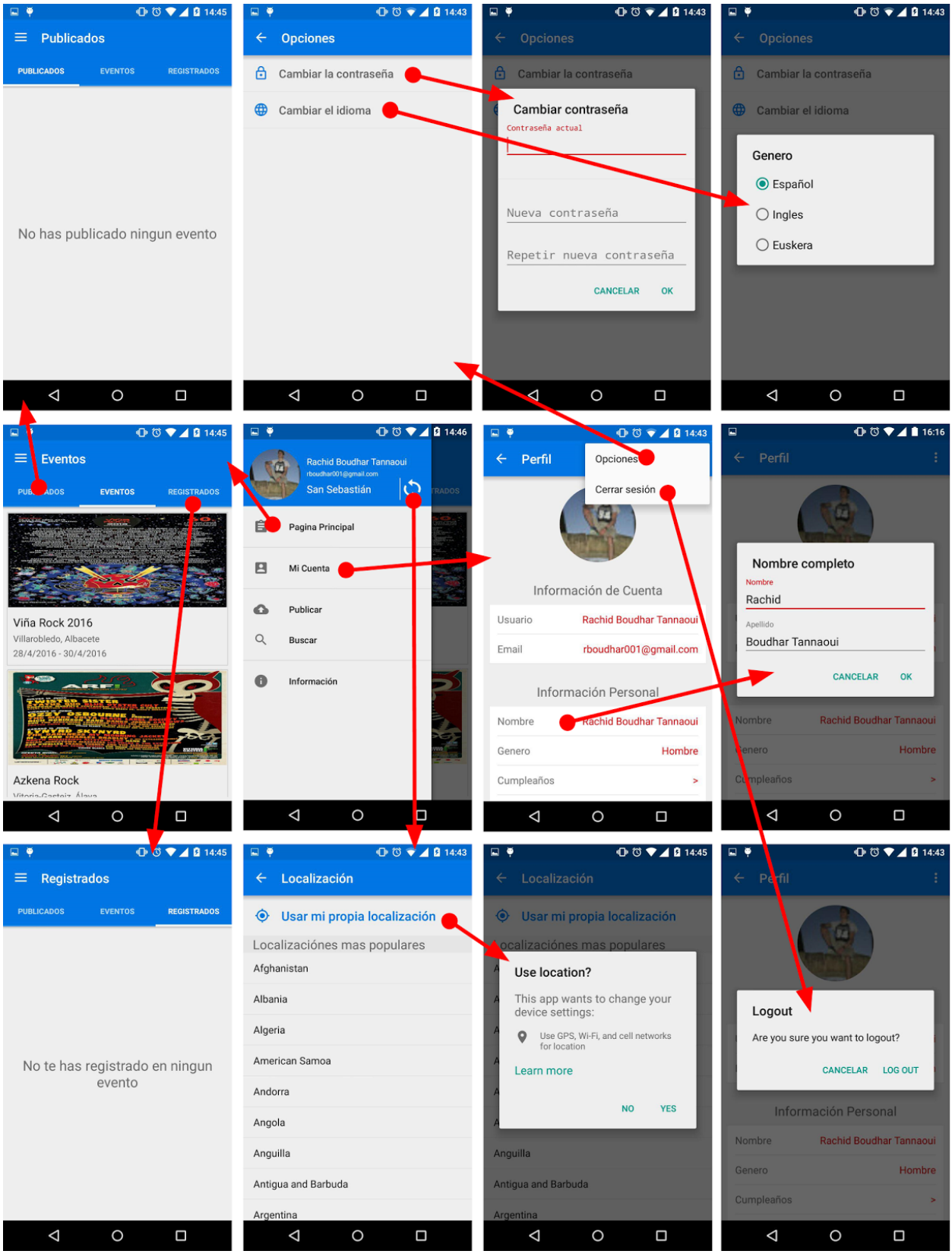


Figura 7.2.3: Diseño de interfaces (usuario registrado)

7.3. APIs

Una de las características de esta aplicación, es el uso de diferentes tipos de APIs, SocialMusFest App utiliza tres APIs de Google, el SDK de Facebook y la API de Meteor.

Las APIs de Google y el SDK de Facebook fueron implementadas por decisión propia ya que se deseaba el conocimiento para llevar a cabo tal tarea.

En cambio, la API de Meteor como habíamos comentado anteriormente al estar la versión web desarrollada en Meteor por decisión de Jon Junguitu Iturrospe a la hora de decidir con qué plataforma desarrollaría su proyecto, se encontró como respuesta a esa decisión con la obligación y necesidad de estudiar y utilizar esta API para la realización de la comunicación entre esta aplicación Android y la página web ya que en ella están alojados los recursos a utilizar por SocialMusFest App.

La comunicación del dispositivo móvil con el servidor se realizó vía *Sockets* gracias a la librería Android-DDR de la cual hablaremos más adelante, y aunque se utilizó esta librería que nos proporcionaba la capa de conexión, también contaba con algunas funciones ya implementadas para la comunicación con la API de Meteor, por desgracia se tuvo que completar esta, ya que solo contaba con dos de los seis métodos que SocialMusFest App requería. Los dos métodos básicos utilizados y localizados en Android-DDR fueron, iniciar sesión y crear cuenta (con email y contraseña). Los otros cuatro métodos añadidos a la librería gracias al estudio de la API de Meteor son, inicio de sesión con los servicios de Google, inicio de sesión con los servicios de Facebook, cambio de contraseña y actualizar cualquier campo de la base de datos de la colección Usuarios y Festivales.

7.3.1. APIs de Google

Las APIs de Google que se han utilizado como se había comentado anteriormente son las siguientes:

- **Auth API:** Se utiliza esta API para añadir la funcionalidad de iniciar sesión con una cuenta de Google, para la autentificación del usuario con los servicios de Google se ha utilizado la API *Auth.GOOGLE_SIGN_IN_API*.
- **Plus API:** Una vez que consigue el usuario iniciar sesión, hemos utilizado también la API *Plus.API*, para la obtención de los datos de cuenta de Google con tal de personalizar la cuenta que se crea para la aplicación, los datos que se recogen con “Plus.API” son el id de usuario de Google, el email, nombre de usuario, nombre, apellido, genero y año de nacimiento.

- **LocationServices API:** Esta API la hemos utilizado además de para solicitar al usuario el permiso de activación del GPS del dispositivo móvil en caso de que este estuviera apagado, también para realizar la funcionalidad de obtener la localización del dispositivo móvil mediante GPS.

7.3.2. SDK de Facebook

El SDK de Facebook fue algo más complicado de implementar, ya que al igual que en Google, se debía de cumplir el requisito de registrar la aplicación en sus servicios y obtener una id de desarrollador. Con Facebook además de ello, en el código era necesario realizar una solicitud inicial desde la aplicación hacia Facebook, para la obtención del visto bueno y los permisos necesarios con tal de permitir que un usuario pudiera iniciar sesión y obtener sus datos de cuenta.

En Facebook no hemos utilizado dos APIs como nos ocurría en el caso de Google sino que al ser un SDK caracterizados estos por ser más completos o mismamente un conjunto de APIs, solamente hemos tenido que utilizar el objeto *FacebookSDK* para realizar el *registerCallback*, este método nos devuelve un objeto JSON con el que se debía trabajar y donde se encontraba la información solicitada a Facebook (información de cuenta del usuario que intenta inicia sesión) y aceptado este requisito por el usuario para la obtención de sus datos personales en el momento en que realiza el intento de login.

7.3.3. Meteor API

Está sin duda ha sido la API más complicada a desarrollar, ya que han sido requeridas bastantes más horas de trabajo de las planificadas, eso fue debido a la necesidad de realizar amplias consultas a Jon Junguitu Iturrospe autor del proyecto SocialMusFest en su versión Web.

Se tuvo que realizar un amplio estudio de Meteor, resolución de bastantes dudas sobre los métodos existentes, la coherencia de datos, la compatibilidad de estos datos con Android, tipo de colecciones utilizadas en la base de datos, y finalmente el entendimiento por cuenta propia de las funciones que aporta Meteor API para la obtención de sus recursos de forma externa.

Para poder estructurar mejor el código se creo la clase MyMeteor.java en la cual se recogen todas las funciones y métodos que se utilizan para la conexión con el servidor.

7.4. Android Manifest

Es un archivo de configuración que está situado en la raíz de cada aplicación, en el podemos aplicar las configuraciones básicas de nuestra App, aunque también su configuración puede realizarse a través de una interfaz gráfica, se ha preferido trabajar con él a nivel texto ya que permite de un rápido vistazo ver la estructura y la configuración del mismo. La versión de la

App, el nombre y el icono de la aplicación se definen en este documento, la versión se debe ir modificando cada vez que sea publicada una nueva versión de la aplicación.

Todas y cada una de las actividades (*Activitys*) de la App deben ser declaradas en esta sección, sino no, más de una vez nos puede llegar a dar errores de compilación al no encontrarse la ventana solicitada, además de ello la actividad inicial debe ser declarada como tipo "Launcher", que en nuestro caso es, Inicializate.java. Aquí también se definen los permisos que se le solicitarán al usuario en el intento de instalar la aplicación en su dispositivo móvil.

En este caso, también se han definido en este documento las id's de desarrollador de Google y Facebook para la verificación de la App en sus servicios con el objetivo de dar uso a sus APIs.

7.5. Archivos Java

En este apartado se describirán todos los archivos .java implementados en la App, tanto las actividades vistas en el gráfico de diseño de actividades del punto 7.1.1, como el resto de clases desarrolladas para una función concreta, por ejemplo, tenemos las clases User.java y Events.java en las que se representa los datos de los usuarios y eventos, la configuración de la base de datos local en Database.java, los adapter utilizados para el relleno de las listas de eventos y la visualización de los Tabs de navegación de ventana EventsAdapter.java y ViewPagerAdapter.java, en la clase MyError.java donde se recogen los errores de conexión, MyLocation.java que obtiene la localización del dispositivo movil, MyDatabase.java que cuenta con las funciones encargadas de insertar o actualizar información en la base de datos local, MyMenu.java encargado de cargar dinámicamente un menú según el rol del usuario, MyMeteor.java que utiliza la librería Android-DDP para implementar las funciones de Meteor API, MyNetwork.java que es simplemente una capa mas por si se deseaba realizar la conexión a otro tipo de plataformas, y finalmente, MyState.java que controla el estado en el que se encuentra un usuario.

7.5.1. Actividades

Los documentos java que implementan las funcionalidades de una actividad o el propósito por el que están programadas estas, son los siguientes:

- **Inicializate.java:** Esta actividad es nuestra pagina principal de carga, o bien llamada *Laucher*. No carga una ventana xml en la aplicación, sino que esta decide cada vez que se carga la App si es que la hemos cerrado anteriormente o en eventos concretos como insertar localización, que ventana como pagina principal cargar hacia el usuario, bien MainActivity.java o EventsActivity.java.
- **MainActivity.java:** Esta es una de las ventanas principales de la aplicación, decimos que tenemos mas de una, porque esta ventana solo aparece cuando la aplicación se ha

instalado por primera vez y se cumple el requisitos de no tener o no haber obtenido la localización del dispositivo móvil.

- **EventsActivity.java:** Esta es la otra ventana principal, normalmente sera la que mas aparezca ya que la localización del usuario una vez introducida no es eliminada, puede ser actualizada por el usuario, pero se decidió no eliminarla por motivos de comodidad, ya que el tener que estar introduciendo una y otra vez la localización, no se vio como optimo ni lógico (recomendado esto por los usuarios *testers* en la fase de pruebas).

  Esta ventana cuenta a su vez con tres sub-ventanas que llamamos navegación de ventanas, en ella hemos implementado un menú deslizante que permite navegar entre las tres sub-ventanas, TabEvents.java, TabPublished.java y TabRegistered.java, la clase encargada de cargar la sub-ventana según en la que el usuario se encuentre es gracias a ViewPagerAdapter.java:

  - **TabEvents.java:** Esta sub-ventana es la encargada de visualizar la lista de eventos que se encuentran en la misma localización del dispositivo móvil o la que se haya introducido por parte del usuario.
  - **TabPublished.java:** Esta sub-ventana muestra la lista de eventos que el usuario haya publicado.
  - **TabRegistered.java:** Finalmente en esta ultima sub-ventana se muestran la lista de eventos en los cuales el usuario se haya registrado o apuntado.

- **LocationActivity.java:** En esta actividad se recoge la localización, bien mediante la funcionalidad GPS que se encuentra implementada en la clase MyLocation.java o bien mediante una lista de localizaciones.
- **AccountActivity.java:** Esta actividad permite iniciar sesión a través de Google y Facebook utilizando en ella las APIs correspondientes a Google o el SDK de Facebook, también se recogen los eventos de iniciar sesión con formulario o registrarse con formulario, los cuales han sido implementados en las actividades LoginActivity.java y SignupActivity.java
- **LoginActivity.java:** En esta actividad se realiza el tramite de iniciar sesión con formulario.
- **SignupActivity.java:** Esta actividad se encarga de crear una nueva cuenta en SocialMusFest mediante formulario.
- **ProfileActivity.java:** Esta actividad es bastante completa ya que se encarga de visualizar los datos del usuario, recoge los eventos *Dialog* para actualizar los datos de usuario mediante ventanas de dialogo y implementado un menú de opciones.

- **SettingsActivity.java:** Se carga a través del menú de opciones que se encuentra en la ventana ProfileActivity.java, muestra dos botones (Cambiar contraseña y Cambiar idioma) e implementa sus funcionalidades.
- **PublishActivity.java:** Podríamos decir que esta es la mas simple de las actividades, ya que simplemente cargar la ventana de visualización xml correspondiente (como hacen el resto de actividades), y tiene unicamente como añadido el recoger el evento al ser pulsado el botón de publicar un evento a través de la pagina web.
- **SearchActivity.java:** Permite al usuario realizar la búsquedas de eventos mediante formulario con los campos lugar y fecha, después muestra los resultados obtenidos de la búsqueda en la ventana SearchResultActivity.java
- **SearchResultActivity.java:** Carga la lista de eventos encontrados de la búsqueda realizada en SearchActivity.java
- **ShowEventActivity.java:** A esta actividad se le pasa un objeto *Event* y es la encargada de visualizar en pantalla sus características, también realizamos en ella el caso de uso "Apuntarse" o "Desapuntarse" según sea el estado del usuario en este evento.
- **InfoActivity.java:** Esta actividad muestra una lista de botones (Ayuda, Términos de Uso y Políticas de privacidad) que enlazan a la pagina web de este proyecto en su versión web, y un ultimo botón (Versión) que visualiza diversa información correspondiente a la aplicación en la actividad VersionActivity.java.
- **VersionActivity.java:** Una actividad simple que obtiene las características de la aplicación y las muestra en pantalla, tales como nombre, versión de esta, icono...etc

7.5.2. Clases

Los siguientes archivos .java no representan ninguna actividad, simplemente fueron implementadas para separar y estructurar de una forma más lógica el código de la aplicación. De entre todas ellas tenemos clases que representan desde un simple objeto, hasta clases convertidas en clases abstractas que realizan alguna funcionalidad muy concreta, esto se realizó de esta forma para evitar la duplicación de código ya que se vio que múltiples actividades daban uso de estas funcionalidades.

- **User.java:** Clase que representa un objeto "Usuario", esta constituida por los campos id, nombre de usuario, nombre, … etc vistos anteriormente, cuenta con múltiples constructores y los métodos Gets y Sets para obtener y modificar los campos citados anteriormente.

- **Events.java:** Al igual que la clase User.java, esta clase representa un objeto “Evento”, cuenta con los campos que caracterizan un evento vistos en el modelo de datos, diversos constructores y los métodos Gets y Sets.
- **Database.java:** En esta clase se implementa el diseño y configuración de la base de datos local SQLite gracias a que esta extiende de SQliteOpenHelper.
- **EventsAdapter.java:** Esta clase es bastante compleja, ya que cuenta con múltiples funciones y métodos. Estas funciones y métodos se encargan de cargar en una lista los eventos pasados por parámetro (ArrayList), obtener el evento que selecciona un usuario en la lista de eventos y devolver el evento a la actividad que lo haya llamado, además de la visualización de los eventos en la lista de forma personalizada, utilizando el estilo desarrollado en el archivo listview_event.xml para darle ese toque de lista de eventos que visualizamos en el diseño de interfaces.
- **ViewPagerAdapter.java:** Este adaptador es bastante mas simple que el anterior, ya que este, simplemente detecta en que pagina de la navegación de ventanas se encuentra el usuario y carga la correspondiente sub-ventana modificando también el titulo y los futuros iconos si es que los hubiera.
- **MyError.java:** En esta clase se ha querido englobar todos los casos de error de conexión que se han ido encontrado, como se realizaba una constante reutilización del código y quedaba mas estructurado que casos de error se podían dar, se ha quedado satisfecho con el resultado y decisión de crear esta clase.
- **MyLocation.java:** Esta clase es la encargada de habilitar la API de Google *LocationServices.API* para solicitar activar el GPS del móvil si este estuviera apagado y obtener la localización GPS del dispositivo si así lo requiso el usuario.
- **MyDatabase.java:** En esta clase se han implementado todas las funciones que se encargan de interactuar con la base de datos local.
- **MyMenu.java:** Esta clase es la encargada de cargar la cabecera de menú correspondiente en el menú desplegable lateral según el rol en el que se encuentre el usuario (con o sin localización conocida, y con o sin sesión iniciada)
- **MyMeteor.java:** Esta clase implementa las llamadas a los métodos que nos proporciona Meteor API para la interacción con los recursos del servidor.
- **MyNetwork.java:** Se encarga de utilizar el método de conexión, en este caso MyMeteor.java para la realización de la conexión con el servidor, se capturan los errores en caso de fallos y se envían a MyError.java

- **MyState.java:** Finalmente esta ultima clase controla el estado en el que se encuentra el usuario en la aplicación.

7.6. Documentos XML (Layouts)

Para el desarrollo de la aplicación se han necesitado una serie de ventanas, los diseños quedan implementados en archivos .xml que son los encargados de dar el formato de la ventana y los diseños o estilos a los elementos de interacción con el usuario.

Para el desarrollo de esta aplicación han sido necesarias las siguientes layouts que se pueden observar en los siguientes puntos, separados así para representar las ventanas de las actividades. Los objetos *Dialog* son ventanas de diálogo utilizadas para una interacción rápida con el usuario con tal de evitar salir de una actividad, los *Headers* representan las las cabeceras del menú y los *Tabs* las sub-ventanas de la navegación de ventanas de la página EventsActivity.java.

7.6.1. Actividades

En este punto se define de qué están compuestas las ventanas de las actividades, en estos archivos se define el diseño de una ventana, los elementos que interactúan con el usuario, botones, textos, listas… etc y su estilo. Las ventanas de las actividades desarrolladas para SocialMusFest App son las siguientes:

- **Activity_main.xml:** En esta ventana se define una menú desplegable lateral para añadir posteriormente un *include* hacia otro documento .xml, este documento es content_main.xml
- **Content_main.xml:** Contiene la interfaz de usuario de la ventana activity_main.xml, esta está compuesta por un objeto *Toolbar* (para la creación de una barra de de navegación personalizada ya que hemos añadido un menú desplegable lateral), una imagen, dos textos y un botón (para introducir la localización).
- **Activity_events.xml:** Al igual que la ventana de la actividad activity_main.xml, estas dos ventanas son las únicas que cuentan con el menú desplegable lateral, en un principio todas las ventanas halladas en las opciones del menú tenían esta característica, pero por opinión de los usuarios *testers* en la fase de pruebas se recomendó el diseño actual.
- **Content_events.xml:** Este archivo implementa el contenido de la ventana activity_events.xml, contiene el objeto *Toolbar*, un objeto *SlidingTabLayout* que representa la navegación de sub-ventanas y dentro de este un objeto *ViewPager* para cargar en el, la sub-ventana que se desea. Esto dependerá de la selección de sub-ventana por parte del usuario: tab_events.xml, tab_published.xml o tab_registered.xml.

- **Activity_account.xml:** En esta ventana se separan mediante *Layouts* las dos formas de iniciar sesión (con Google o Facebook o a través de formulario), así mismo Google y Facebook tienen sus propios tipos de botones *SignInButton* y *LoginButton*. Para implementar los botones de de iniciar sesión o registrarse a través de formulario simplemente se ha utilizan dos objetos *Button*.

- **Activity_forgotten_password.xml:** Esta ventana cuenta con un objeto *AutoCompleteTextView* (para introducir el email de recuperación de cuenta) y un objeto *Button*.

- **Activity_info.xml:** Aquí se han definido cuatro objetos *Button* para representar los botones de ver ayuda, ver los términos del servicio, las políticas de privacidad y la versión de la App.

- **Activity_location.xml:** Esta ventana cuenta con los siguientes elementos: *Button*, *TextView*, *ListView*, el botón se utiliza para obtener la localización por GPS, el texto es informativo y el listview para mostrar una amplia lista de localizaciones.

- **Activity_login.xml:** Esta ventana contiene un formulario, estando caracterizado por dos objetos *AutoCompleteTextView* (para introducir el email y contraseña) y un *Button* para poder iniciar sesión.

- **Activity_signup.xml:** En esta ventana se halla el formulario para poder crear una cuenta en la aplicación, este formulario está compuesto por dos *AutoCompleteTextView* (nombre de usuario, email), un *EditTextPassword* para la contraseña y un objeto *Button*.

- **Activity_profile.xml:** Ha sido la ventana más extensa y complicada de diseñar, ya que cuenta con una serie de *Layouts* que buscan separar y dar estilo a la ventana, dentro del primer *Layout* tenemos un objeto *CircleImageView* de una librería externa utilizada para mostrar la imagen de perfil del usuario en formato circular, en el segundo tipo de *Layout* (representa la información de cuenta) se incluyen dos *RelativeLayout* que ingeniosamente actúan como si fueran botones, esto se realizó de esta manera porque se buscaba tener la descripción y el resultado de esta en el mismo elemento, es más, se consiguió añadir incluso una animación al "botón", el primer "botón" muestra el nombre de usuario y el segundo el email, al pulsarlos, abre una ventana de diálogo para poder editar sus campos. Finalmente el tercer tipo de *Layout* (representa la información personal) cuenta con estos 5 tipo de "botones" que representan el nombre completo, el género, el cumpleaños, el lugar y el estilo de música. Para finalizar, también se añadió en la parte superior derecha un menú de opciones para visualizar las opciones o cerrar sesión.

- **Activity_publish.xml:** Esta ha sido la ventana más simple, ya que cuenta únicamente con dos textos *TextView* que buscan informar al usuario de como publicar un evento, un

objeto *Button* para abrir el navegador web del dispositivo móvil y redireccionar a la sección "publicar evento" en la página web.

- **Activity_search.xml:** Contiene dos elementos *AutoCompleteTextView* para introducir la localización y la fecha que se utilizan posteriormente como filtro a la hora de realizar la búsqueda de eventos, obviamente también cuenta con un objeto *Button* para poder realizar la búsqueda.
- **Activity_search_results.xml:** Esta ventana contiene un *TextView* y un objeto *ListView*. El texto se utiliza para informar al usuario en caso de que no se hayan encontrado eventos en la búsqueda, y el listview para mostrar la lista de eventos en caso de que al menos se haya encontrado uno, para poder mostrar uno u otra se juega con el parámetro *android:visibility*, que nos permite hacer desaparecer un elemento u otro.
- **Activity_settings.xml:** Muestra las opciones que puede realizar un usuario con sesión iniciada (cambiar contraseña y cambiar idioma), estos se representan con dos objetos *Button.*
- **Activity_show_event.xml:** Al ver que las dimensiones en altura de esta ventana eran demasiado grandes, se añadió un elemento *ScrollView* que permite al usuario moverse hacia abajo o hacia arriba. Dentro de este se encuentran los elementos con los que representamos las características de un evento, tales como un *ImageView* para ver la imagen del evento, dos *TextView* para mostrar el título y la descripción, y a partir de aquí se separa por *Layouts* cada sección que representa el texto informativo de una característica con el valor de este mismo. Se ha realizado lo anteriormente dicho para los atributos lugar, fechas de inicio, fecha final, URL de la página para la venta de entradas, teléfono de contacto, enlace a la página web oficial, nombre de usuario creador del evento y el número de asistentes junto a su capacidad. Para finalizar, abajo del todo también se añade un *Button* para que el usuario pueda "Apuntarse" o "Desapuntarse" a este evento.
- **Activity_version.xml:** Es otra de las ventanas más simples desarrolladas, ya que esta cuenta únicamente con una serie de objetos *TextView* con el objetivo de visualizar los datos de la App y un objeto *ImageView* para visualizar el icono de la aplicación.

7.6.2. Dialogs

Las ventanas de diálogo al igual que las actividades también requieren de un diseño, lo que las diferencia de estas, es que no cuentan con un archivo java exclusivo. Sino que son las actividades las que llaman a través de un objeto *Dialog* a un diálogo (ventana de diálogo genérica), esté a través de otro objeto *LayoutIflater,* "infla" o añade el diseño hallado en el archivo .xml, con el

método *inflate()* dentro de sí mismo, dándose el diseño o estilo que hemos definido en los archivos a la ventana de diálogo.

Los archivos .xml que se han desarrollado para crear los diferentes tipos de diálogo, que a su vez tienen como función actualizar algunos datos del usuario, son los siguientes:

- **Dialog_profile_email.xml:** Este diseño tiene únicamente un objeto *AutoCompleteTextView* para la inserción del nuevo email a actualizar. Los botones para aceptar o cancelar un diálogo son propios de la clase Dialog y no es necesario añadirlos
- **Dialog_profile_username.xml:** Tiene un objeto *AutoCompleteTextView* para introducir el nuevo nombre de usuario.
- **Dialog_profile_name.xml:** Cuenta con dos objetos *AutoCompleteTextView* para introducir el nombre y apellido, estando estos dos separados.
- **Dialog_profile_place.xml:** Tiene un objeto *AutoCompleteTextView* para introducir el nuevo lugar que desee el usuario, no nos equivoquemos a la hora de comparar el "lugar" con la "localización", el lugar representa de donde es el usuario, la localización representa la ubicación del dispositivo móvil o del usuario.
- **Dialog_profile_musicstyle.xml:** Tiene un objeto *AutoCompleteTextView* para que el usuario pueda añadir su estilo de música favorito.
- **Dialog_settings_change_password.xml:** Tiene tres objetos *EditTextPassword* para introducir la contraseña actual, y dos veces la nueva contraseña.

7.6.3. Headers

Los *Headers* que se han desarrollado en esta aplicación, representan las cabeceras que se le pueden añadir dinámicamente al menú desplegable lateral en las ventanas de actividades MainActivity.java y EventsActivity.java. Además también se creó el archivo donde se diseña y se da estilo a los eventos que se visualizan en las listas de eventos.

- **Nav_header_location.xml:** Esta cabecera está compuesta por un objeto *Layout*, un objeto *TextView*, un View (utilizado como separador) y un objeto *Button* que se utiliza para actualizar la localización.
- **Nav_header_login.xml:** En esta cabecera se jugó mucho con los *Layouts* para obtener el resultado deseado, cuenta con un objeto CircleImageView (para visualizar la imagen de perfil de usuario), tres objetos TextView donde se visualizan el nombre de usuario, el email y la localización, esta última solo si se diera el caso de que el usuario hubiera dado una localización sino aparecería como no introducida. Por último, también tenemos un

Button que redirecciona a la ventana LocationActivity.java donde el usuario puede introducir o actualizar su localización.

- **Listview_event.xml:** Al igual que con el archivo nav_header_login.xml este también tuvo sus quebraderos de cabeza para conseguir el estilo y diseño deseado, está compuesto por un objeto *ImageView* y cinco *TextView*, estos muestran el nombre, lugar, fecha inicio, fecha fin y un separador entre fechas, además de los múltiples *Layouts* configurados ingeniosamente para conseguir el resultado que se visualiza en el diseño de interfaces, concretamente cómo se visualizan los eventos en la lista de eventos..

7.6.4. Tabs

Los archivos .xml que definen el diseño de las sub-ventanas (Eventos, Publicados y Registrados), los llamamos Tabs, se han definido tres sub-ventanas, aunque sus diseños son prácticamente iguales. A continuación vemos dichos diseños:

- **tab_events.xml, tab_published.xml y tab_registered.xml:** Estas tres sub-ventanas incluyen un objeto *ProgressBar* (Para mostrar el icono de cargando mientras la aplicación se encuentra realizando alguna acción), tienen un objeto *TextView* y otro *ListView*, el texview lo utilizamos para mostrar mensajes informativos según sea el caso; “no existen eventos en la aplicación”, “el usuario no ha publicado ningún evento” o que “el usuario no se ha registrado en ningún evento”, al contrario, el listview lo utilizamos para visualizar los eventos encontrados por localización, los eventos que un usuario haya publicado o los eventos en los que un usuario se haya registrado. Cada mensaje informativo y cada lista se visualizan según la ventana en la que corresponda.

7.7. Ficheros de recursos

A continuación, se describen los ficheros de recursos que se han utilizado para personalizar la aplicación, estos ficheros se encuentran en la carpeta **values** de Android Studio y son propios de todas las aplicaciones Android. Los ficheros de recursos, se han utilizado tanto para modificar las características y el diseño de la App como también para implementar ciertas funcionalidades, como por ejemplo cambiar idioma.

Podríamos decir que en esta sección se guardan los datos constantes de la aplicación, siendo posible la referencia a estos desde el código, esta característica a mi parecer es un gran punto a favor en este sistema de desarrollo, ya que este sistema, nos ofrece toda la potencia y capacidad de adaptar la aplicación a las características del dispositivo móvil en el que se ejecute o a las solicitadas por el usuario sin mucha complicación.

Por ejemplo, si deseamos ajustar un elemento de la pantalla según el tamaño de esta (por motivos de dispositivo móvil), podemos predefinir en el archivo dimens.xml estos valores, o para cambiar

el idioma de la aplicación, es tan simple, como cambiar el documento al que hacemos referencia para obtener los textos que se visualizan en la App, por otro.

Los cuatro archivos utilizados en el desarrollo de SocialMusFest App son los siguientes:

- **Color.xml:** En este recurso se han añadido bastantes más colores de los que ofrecía la aplicación cuando se creó el proyecto en Android Studio, de entre ellos una amplia gama de color azul y grises característicos de esta aplicación.
- **Dimens.xml:** Aquí se han añadido múltiples dimensiones, las cuales son referenciadas por los elementos que se encuentran en los diferentes diseños de ventanas, estas dimensiones pueden representar la altura o anchura de un elemento o el tamaño de fuente de un texto.
- **Strings.xml:** En este archivo se han añadido todos los textos que se muestran en la aplicación, y por ello son los elementos de diseño (en archivos .xml) y los objetos de clases desde código (en archivos .java), los que las referencian. Se cuenta con dos archivos mas de strings.xml donde se tienen traducidos todos los textos de la App al Inglés y Euskera.
- **Styles.xml:** Cuenta con una serie de estilos que se utilizan tanto para definir el estilo de navegación que deseamos que tenga nuestra aplicación, como para aplicarlo a los elementos de diseño que se encuentran en los archivos .xml.

7.8. Librerías

7.8.1. Android-DDP

Esta es una librería de código abierto que implementa el Protocolo de Datos Distribuidos (DDP) hacia objetos *Meteor* con la finalidad de conectar una aplicación cliente basada en Android a los servicios que ofrece Meteor API, consiguiendo así, construir y trabajar con los recursos y datos en tiempo real hallados en el servidor.

Aunque solo cuente con dos métodos (iniciar sesión y crear cuenta a través del email y contraseña) de los seis métodos (leer más abajo) que se han tenido que utilizar para el desarrollo de esta aplicación, se ha conseguido evitar la implementación de toda la infraestructura para la conexión vía *sockets* que se requería con tal de lograr conectar SocialMusFest App con el servidor.

Los cuatro métodos añadidos a la librería y ofrecidos por la API de Meteor son los siguientes: inicio de sesión con los servicios de Google, inicio de sesión con los servicios de Facebook, cambiar la contraseña y actualizar cualquier campo de la base de datos de la colección Usuarios y Festivales.

Como requisito mínimo para su utilización era disponer de una versión de Android superior a la 2.3, aunque nuestra aplicación ya tenia ese pre-requisito desde la fase de planificación.

En un principio se ideó el realizar esta librería por cuenta propia, pero viendo el nivel de complicación que tiene realizar una librería así, y encontrando esta bastante completa, se decidió optar por usar esta librería y completar las funciones requeridas.

7.8.2.SlidingTabLayout y SlidingTabStrip

SlidingTabLayout es una librería de código abierto que nos permite añadir como widget la funcionalidad de sub-ventanas en las aplicación Android, se adapta a las nuevas aplicaciones previas a Android Lollipop y a los nuevos Material Design desarrollados por Google para poder tener compatibilidad con versiones anteriores de Android. Esta librería son únicamente dos clases que hemos añadido directamente al cuerpo de documentos java que componen nuestra aplicación. Gracias a estas clases se ha podido implementar el sistema de navegación de sub-ventanas de la pagina principal EventsActivity.java

Existían muchas más opciones a tener en cuenta además de esta, lo único en lo que se diferencia este widget a las otras formas de implementar el sistema de sub-ventanas, era el rápido sistema de actualizaciones que tenían y la recomendación expresa de Google al no estar regulado ni añadido oficialmente la navegación de sub-ventanas en Android Studio, nos recomiendan su uso para evitar en un futuro próximo, problemas de incompatibilidad de versiones con Android.

7.8.3. CircleImageView

Esta librería es de código abierto, nos ha permitido simplemente el uso del objeto *CircleImageView* en el diseño de ventanas, este objeto es simplemente un campo de imagen circular al estilo característico de Google.

7.8.4. Picasso

Esta librería al igual que CircleImageView y Android-DDP, es de código abierto, gracias a ella nos hemos evitado grandes dolores de cabeza a la hora de visualizar una imagen en un elemento de visualización de imágenes, bien sea en un objeto *CircleImageView* o un *ImageView* normal.

Nos ha permitido cargar una imagen en alguno de los elementos citados anteriormente bien dada su ruta a los recursos de la aplicación, o mismamente una URL externa para cargarla a través de Internet. Además de ello, esta librería nos ha permitido adaptar cualquier imagen a las características del elemento de visualización a través de las propiedades de la función cuando se la llama.

CAPÍTULO 8.

8. Pruebas

Este capítulo describe las pruebas realizadas a lo largo de todo el ciclo de vida de desarrollo. El *testing* y el *debugging* realizado, no tienen otra función, que comprobar que las cosas funcionan como tienen que funcionar, el objetivo por ello es buscar errores de todas las formas posibles, realizar cualquier acción permitida y no permitida con tal de "romper" la aplicación.

La filosofía del *testing*, nos ha ensañado con el paso del tiempo, que es mejor programar poco y bien, para después probar lo y depurar, que realizar directamente la implementación completa de la aplicación, y esperar después a que compile el compilador a la hora de construir la aplicación.

Para la realización de la fase de *testing* y *debugging* se siguió una metodología de depuración que permite un constante *feedback* con los usuarios *testers* utilizados en esta fase, mejorando la eficiencia a la hora de depurar la App, en los siguientes puntos se define dicha metodología y en que dispositivos móviles se han llevado a cabo.

8.1. Pruebas de desarrollo

Las pruebas o *tests* de la aplicación se llevaron a cabo en un ciclo de vida de desarrollo muy temprano, esto fue debido a la decisión de realizar la implementación por etapas, se buscaba completar la implementación de un caso de uso, y realizar los correspondientes testeos y pruebas por parte de los usuarios *testers* para decidir los cambios a realizar, o errores a corregir.

Las pruebas, errores, opiniones, tareas a realizar… etc, se han llevado a cabo con el sistema *Bugtracker*, este sistema recoge los errores de la aplicación, opiniones de usuarios y tareas a realizar y las engloba todas ellas en una lista, esta lista queda ordenada por orden de prioridad, al principio la lista queda encabezada por tareas a realizar y funciones a implementar, pero según se va avanzando en el desarrollo y se van comenzando las pruebas y primeros testeos por usuarios reales, se definen una serie de opiniones respecto a los estilos de interfaces, cambios a realizar para mejorar o añadir funcionalidades en la aplicación y *bugs* en la aplicación, si el *bug* o error encontrado es bastante grabe, su prioridad será mayor subiendo rápidamente en la lista, quedando normalmente en los puntos más bajos, los cambios de interfaz o estilos. Una vez realizado o subsanado un error se subrayaba para señalar que ya sea ha atendido.

De esta manera se ha conseguido estar en un estado constante de depuración que ha permitido una rápida evolución del producto.

A continuación, se definen las pruebas realizadas por casos de uso y funcionalidad de la App:

8.1.1. Pruebas de navegación entre ventanas

Al inicio del desarrollo, lo primero que se realizó fue crear el sistema de navegación que se iba a tener en SocialMusFest App. Se crearon las ventanas principales, se añadió un menú y posteriormente las ventanas compuestas en el menú.

En esta fase se llevó un corto periodo de pruebas para comprobar su correcto funcionamiento en dos dispositivos móviles diferentes. No se obtuvieron serios problemas.

8.1.2. Pruebas para crear una cuenta e iniciar sesión

Lo siguiente que se realizó fue la creación de cuentas y el caso de uso iniciar sesión de forma local, tanto con los servicios de Google como con los de Facebook , y también se añadió mediante formulario.

En esta fase de pruebas, se testeó el comportamiento de las APIs de Google, el SDK de Facebook, la validación de los datos en los formularios y la creación de la base de datos local y su correcto funcionamiento.

Por parte de los *testers* se solicitó un cambio de estilo a la ventana para hacerla más atractiva, ya que se tenía en la misma ventana los botones de inicio de sesión con Google y Facebook junto a los botones de iniciar sesión o registrarse mediante formulario juntos, por lo que no se sabía para qué era que.

Además del cambio de estilo hubo numerosos, pero demasiados errores:

- Al iniciar sesión con Google y Facebook por problemas del id del desarrollador al no ser público (mayoritariamente con Facebook) daba errores internos con las APIs y el SDK, no se pudo depurar en estos momentos.
- Cada actividad (ventana) no tomaba el inicio de sesión como debía, esto fue debido al conjuntar el sistema de navegación de ventanas con el sistema de inicio de sesión.
- No se realizaba correctamente la inserción de datos en la *DB* local o se perdía la información reiniciando se esta por ningún motivo.

8.1.3. Pruebas de obtención de la localización

La página principal después de ser diseñada e implementada en la primera versión de la planificación, contaba con un botón que obtenía mediante GPS la ubicación del dispositivo móvil y se tenía pensado otro para eliminarla.

Gracias a esta fase de pruebas y a un usuario de testeo muy rigurosos con su privacidad, me hizo ver que debería de existir otro método para poder dar un localización por parte del usuario. Así nació la necesidad de añadir una lista de localizaciones para aquellos usuarios que nunca utilizarían el servicio GPS.

Se separó la funcionalidad GPS a la clase MyLocation.java y el botón de la pagina principal MainActivity.java redireccionaria ahora a una nueva ventana LocationActivity.java donde en esta se daría opción al usuario el método a elegir para introducir o actualizar su localización.

8.1.4. Pruebas de menú

Una vez desarrollado la navegación del menú, el sistema de creación de cuentas y la obtención de la localización, se implementaron las cabeceras para visualizar la información del usuario y su localización.

Esta fase de pruebas fue un sinfín de errores ya que en un principio no se contaba con la clase MyMenu.java, por lo que cada actividad (ventana) implementaba la inclusión o no de la cabecera de inicio de sesión o de localización obtenida, se depuro bastante el estilo de las cabeceras con el objetivo de hacerlo atractivo para los *testers* y se corrigieron los errores añadiendo la clase MyMenu.java. Por petición de los usuarios de testeo se eliminaron los menús de todas las ventanas que no fueran las páginas principales.

8.1.5. Pruebas de actualizar de datos de cuenta

Esta fase de pruebas fue una de las más productivas y exigentes en cuanto a diseño se refiere.

Una vez realizado la posible navegación desde la imagen de perfil que se visualiza en el menú desplegable lateral de las páginas principales cuando un usuario tiene sesión iniciada. Se aconsejó la inclusión de otro botón ya que no era tan obvio, que para acceder a la ventana de “ver perfil” se tuviera que clickear en dicha imagen.

Así nació la opción “Mi Cuenta” en el menú desplegable lateral de las páginas principales, uniéndose las opciones “Iniciar Sesión” y “Crear Cuenta” que se encontraban antes. Además, este botón se implementó para que su comportamiento fuera de forma dinámica, si un usuario no registrado clickeara el botón, este sería redirigido a la ventana AccountActivity.java (donde se encuentran los caso de uso iniciar sesión con Google, Facebook, mediante formulario y registrarse), en cambio si este estuviera registrado seria redireccionado a la ventana ProfileActiviy.java.

Volviendo al tema inicial, se comentaba que esta ventana fue un desafío de diseño, ya que se tuvo que implementar el sistema de “botones” con *Layouts* para visualizar el texto informativo que hace referencia a dicho botón, además del texto que se carga dinámicamente desde la clase mostrando el dato de cuenta. Los *testers* quedaron muy satisfechos por el cambio realizado.

8.1.6. Pruebas de cerrar sesión y cerrar la aplicación

Esta fase nace de las quejas de los *testers*, porque cerraban la aplicación a la hora de volver atrás por desconocer su sistema de navegación de ventanas.

Al encontrarse que varias ventanas tienen la opción de volver atrás, se vio que se suele pulsar repetidas veces en este botón para agilizar la vuelta a la página principal, haciendo que si la aplicación estuviera en esta, se cierra directamente, por ello se solicitó la implementación de cierre de la App mediante dos pasos.

Así mismo, a la hora de cerrar sesión también se solicitó que se mostrará un aviso de si estas seguro de ello.

8.1.7. Pruebas de visualizar las listas de eventos

Esta fase fue la más longeva de todas, la implementación y depuración total de este caso de uso duró más de dos semanas, por lo que se desglosa en etapas:

- En la **primera etapa** era necesario que la versión web de SocialMusFest estuviera en un servidor. Al ver que no se podía subir el proyecto al hosting propio de Meteor (se convirtió en un servicio de pago), y la tan limitada lista de hostings donde poder alojar dicho proyecto, ya que se subieron y realizaron pruebas de conexión a todos ellos, ninguna llegó a ser viable por necesidades de pago. Finalmente se encontró un hosting (Scalingo) con limitadas opciones de tiempo (proyecto temporal durante 20 días) donde se pudo alojar el proyecto.
- Como **segunda etapa** se fue realizando poco a poco la conexión con el servidor gracias a la librería Android-DDR y verificando que fuera correcta. Después se intentaron obtener los datos del servidor, por ello se aprendió a base de pruebas como solicitar o responder para generar una comunicación (ida y vuelta) entre Android-Meteor.
- Una vez realizada la conexión, entendida el funcionamiento de la librería Android-DDR y obtenidos los datos, se tuvieron que adaptar en SocialMusFest App. Se llevó a cabo una **tercera etapa** de pruebas consultando a Jon Junguitu Iturrospe los tipos de datos de cada campo (Meteor guarda en la base de datos MongoDB los datos en el formato que le conviene ya que el desarrollador no visualiza esta capa), además de sincronizar el sistema de publicaciones característico de Meteor para obtener los datos de la *DB*.
- En la **cuarta etapa** una vez ya teniendo los datos en SocialMusFest App, se desarrollaron las sub-ventanas de la página principal EventsActivity.java, y se implementaron los adaptadores necesarios (adaptador para la lista y adaptador para el diseño de la lista) con tal de añadir en la lista de eventos a visualizar los eventos recogidos del servidor, los

eventos contenidos en un objeto ArrayList logrado generar de los datos obtenidos al realizar la consulta al servidor.

- Finalmente la **quinta etapa** era la prueba por parte de los *testers* para comprobar su correcto funcionamiento, de sus pruebas se decidió modificar el diseño de la ventana para hacerlo más atrayente, se modificó el objeto *ProgressDialog* por un *ProgressBar* y se corrigieron numerosos errores de conexión y se atraparon los tipos de estos (se creo la clase MyError.java) para mostrar el mensaje de error correspondiente.

8.1.8. Pruebas de buscar eventos y publicar evento

Ya que se tenía implementado en la página principal la visualización de la lista de eventos por localización, se implementó por consiguiente estos dos casos de uso, no teniendo tan graves problemas a excepción del modelo de tipos (text y date) que se mostraban en el formulario.

Se recomendó y solicitó separar la búsqueda que se realiza en el formulario con el campo lugar en dos, un campo para países y otro de lugares siendo este una calle, una ciudad, una comunidad autónoma … etc, por incompatibilidad en la base de datos servidor no se pudo llevar a cabo, en cambio, el campo lugar pasó a realizar las búsquedas mediante comparaciones de contenido, si una dirección contiene la palabra introducida en el campo lugar se mostraba ese evento.

8.1.9. Pruebas de iniciar sesión, registrarse, y actualizar perfil en servidor

Para llevar a cabo estas pruebas se tuvo que volver a consultar a Jon Junguitu Iturrospe sobre cómo había realizado la configuración de la base de datos en Meteor. Ya que la gestión de usuarios en la base de datos se realiza de otro método (con un objeto Account en lugar de trabajar directamente con la colección de MongoDB)

Fue necesario la creación de un canal de publicaciones, modificar las funciones de la clase MyMeteor.java para acceder la base de datos después de subscribirse al canal citado anteriormente, ya que si no se realizaba la subscripcion, los datos ni se emitían ni se podían visualizar.

Por parte de los *testers* también se añadieron los mensajes informativos hacia un usuario no registrado en las sub-ventanas “Registrados” y “Publicados” de que no tenían acceso a esta funcionalidad y en la ventana de “ver perfil” se modificó el diálogo para modificar la fecha del año de nacimiento o cumpleaños para realizarlo mediante un calendario.

8.1.10. Pruebas de funcionalidad de enlaces

Esta fase simplemente trataba de recoger si SocialMusFest App podía abrir correctamente el navegador por defecto del dispositivo móvil en el que se ejecutará, sobre todo en versiones antiguas de android que tienen el navegador web clásico.

8.1.11. Pruebas de visualización de evento y registro

El diseño de esta ventana y la interacción del botón “Apuntarse” o “Desapuntarse”, fue gracias a un *tester* que vio la necesidad de que si en la página web, la visualización de un festival o concierto y la animación del botón para apuntarse o desapuntarse se realizaba de una forma, sería lógico que la versión Android también tuviera un estilo similar.

Los errores encontrados entre ellos fueron, la no actualización de los datos, tiempo de espera infinito si la conexión no se llegaba a establecer, mensaje de aviso de correcto registro… etc.

8.2. Dispositivos móviles utilizados en las pruebas

Las pruebas realizadas por parte de los *testers* se llevaron a cabo en los siguientes dispositivos móviles con las siguientes características:

- Motorola Moto G1, Android 5.0
- Sony Xperia Z3, Android 5.1
- LG G3 con Cyanogenmod, Android 6.0
- BQ Aquaris 4.5, Android 5.0
- BQ Aquaris E5, Android 4.4
- LG E-460, Android 4.1.2.
- Sony Xperia Z3 Mini, Android 4.1.

CAPÍTULO 9.

9. Gestión del proyecto

En este capítulo, se describen todos los aspectos relacionados con la gestión del proyecto.

Al inicio del proyecto en la planificación inicial, se tenía previsto el realizar una aplicación Android que pudiera comunicar a las personas el conocimiento de los conciertos y festivales de música que los rodean, además de querer concentrar la información de estos en un único punto. Ahora, una vez finalizado la etapa de desarrollo se puede afirmar que los objetivos del proyecto se han cumplido.

Los requisitos mínimos iniciales de la aplicación al verse en numerosas ocasiones extendido por el *feedback* obtenido a través de los usuarios *testers*, se convirtió en un punto mucho más importante de lo que se creía en la fase de desarrollo y pruebas, ya que prácticamente se mejoró bastante la presentación de la aplicación y el funcionamiento real de sus casos de uso.

Este proyecto, al encontrarse al principio con incompatibilidades en el desarrollo entre Android-Meteor, la inclusión de nuevas tecnologías o herramientas que no se tenía previsto utilizar, y la puesta en común entre ambos autores con el modelo de datos y las amplias consultas que después se llevaron a cabo, hizo que las horas de desarrollo se dispararan para poder llevar a cabo la aplicación.

Cabe remarcar que por motivos de imposibilidad externas al proyecto, en el estudio de hostings donde alojar la versión web, cuando se vio que en el transcurso del ciclo de vida del proyecto, los hostings que se encontraban dando servicios gratuitos como lo son mismamente la propia página de Meteor (primer elección y planificación realizada), Heroku, 9cloud… etc, pasaron a tener únicamente servicios de pago, se registró el evento, como un verdadero caso de riesgo en el desarrollo del proyecto al no encontrarse un alojamiento posible.

Respecto al desarrollo de la memoria frente al desarrollo del producto, los factores de riesgo y el número de horas a trabajar no llegaron a tener nada que ver una cosa con la otra. Mientras que el desarrollo de la memoria fue lineal, controlado y perfectamente planificado, la etapa de desarrollo fue todo lo contrario al encontrarse con los problemas de hosting y el estudio de implementación de las nuevas tecnologías no previstas.

Para finalizar podemos ver en la siguiente figura 9.1, la tabla de dedicaciones donde podemos visualizar el tiempo de dedicación realizado junto a una suma total de las horas:

	Tareas	Total Tareas	Dedicación en horas
Planificación	Base	11 h.	
	Extensión	9 h.	20 h.
Desarrollo	Captura de requisitos	13 h.	
	Formación	78 h.	
	Implementación	311.5 h.	
	Pruebas	37 h.	439.5
Documentación	Memoria	61 h.	
	Presentación	---	61 h.*
Gestión	Seguimiento y Control	4.5 h.	4.5h.
TOTAL			525 h.

Figura 9.1: Tabla de dedicaciones

La figura 9.1 se desglosa en cuatro apartados siendo cada uno de ellos al mismo tiempo desfragmentado por etapas.

En la planificación se encuentra una planificación inicial y después la extensión de esta.

La fase de desarrollo se desfragmenta en una captura de requisitos inicial que fue siendo modificada en el transcurso de pruebas. La formación representa el número de horas utilizadas para la obtención del conocimiento necesario para que después en la fase de implementación se pudiera desarrollar la aplicación, y finalmente se cuenta con una fase de pruebas que se llevó a cabo nada más empezar esta.

Para terminar se distinguen también las fases de documentación y gestión, esta última llevándose a cabo siempre antes del final del día para capturar y registrar lo realizado en ese día. La documentación queda desglosada en la etapa de memoria realizada al final del ciclo de vida del proyecto y una vez terminado este, la presentación o defensa del proyecto.

CAPÍTULO 10.

10. Conclusiones y líneas futuras

En este capítulo se comentan las conclusiones del proyecto, el resultado del producto final, las tecnologías usadas y aprendidas junto a la experiencia personal del autor, también se describen las propuestas de extensión de la aplicación que se realizarán en un futuro, ya que se tiene pensado seguir con la fase de producción tras finalizar este proyecto.

10.1. Conclusión del proyecto

SocialMusFest App es una aplicación para el sistema operativo Android, concretamente es una aplicación que engloba todo el sistema de conciertos y festivales de música existentes en un único punto, cuenta con una gestión de usuarios, los cuales tiene la posibilidad de registrar su actividad en los eventos (conciertos o festivales) que aparezcan, pueden consultar sus características y buscar los eventos más próximos a ellos.

Al inicio del proyecto, se planteó esta aplicación como un generador de calendarios sobre conciertos y festivales de música, pero rápidamente se modificó la idea a lo que es ahora mismo esta aplicación. Su desarrollo se llevó a cabo en Android Studio siendo esta la herramienta oficial de desarrollo para aplicaciones Android. Se usaron diferentes librerías, APIs, SDKs, herramientas y tecnologías. El proyecto se desarrolló de forma muy lenta al inicio ya que no se contaba con el conocimiento necesario para tener una maniobrabilidad deseada en Android Studio, Git y sus herramientas. Al principio la dedicación residía en documentarse y probar todo un poco para ir familiarizándose con el entorno, se crearon múltiples proyectos con los casos de uso ya predefinidos por Android Studio, y se utilizó la herramienta Navigation Tool la cual no gusto mucho para desarrollar el sistema de navegación entre ventanas. Una vez conocida la arquitectura y el sistema de archivos, al mismo tiempo que se obtuvo la maniobrabilidad deseada se puso en marcha la fase de implementación.

Se creó como inicio de desarrollo, el sistema de navegación entre ventanas y desde ahí se fueron implementado los casos de uso y sus fases de pruebas correspondientes. En algunos casos se debía parar la implementación para documentarse una vez más, sobre como realizar o llevar a cabo un caso de uso para desarrollar o generarlo en Android Studio. Gracias a la metodología *BugTracking* se ha conseguido un desarrollo por fases; tomar un caso de uso, implementarlo en la aplicación y depurarlo con usuarios reales, recibir su *feedback*, aplicar los errores, consejos u

opiniones nuevamente al caso de uso, realizar nuevamente el *feedback* y hasta que no se tuviera la satisfacción de los usuarios *testers*, no se podía pasar a desarrollar el siguiente caso de uso.

Se siguió esta metodología hasta terminar satisfactoriamente la aplicación, ahora, una vez encontrados en este punto, y terminada la fase de desarrollo, se puede concluir que la obtención de los resultados corresponde con el planteamiento inicial desarrollado.

Por otra parte, la realización de este proyecto ha supuesto al autor a una adquisición de amplios conocimientos que no se habían adquirido en el transcurso de la carrera, tales como por ejemplo el desarrollo de una aplicación completa sin necesidad de salir de Android Studio. Se adquirieron los conocimientos sobre VSC (*Version Control Systems*), Git, AVD Manager, ADB y el SDK de Android. Se ha aprendido a utilizar Git y la plataforma de versiones GitHub fuera de Android Studio, tanto en la versión escritorio como por comandos. Se aprendió a colaborar en un proyecto abierto y completar una librería, como generar una SSH Key para poder realizar conexiones autenticadas y como recuperar o gestionar los ficheros según la versión. Se ha trabajado con plataformas que se desconocían, a leerse una API de arriba a abajo (Meteor API) e implementar algunas de sus funciones, lo cual no estaba planificado en el proyecto y fue un alto costo en la fase de implementación. Se ha hecho uso de librerías externas y se conoce como se realiza un sistema de conexión DDP. Se ha aprendido a utilizar las herramientas de desarrollador de Google y Facebook encontradas en sus páginas web para poder utilizar sus APIs o SDK, tengo que agradecer a los tutoriales y guías de estas empresas que ponen a disposición de los desarrolladores para poder implementar cualquiera de los servicios que estos ofrecen, ya que gracias a esto no solo se a aprendido a utilizar una API como se deseaba, sino que incluso se conoce como generar una, por lo que en un futuro serán un punto a favor cuando se vuelva a trabajar con este tipo de recursos.

Como habíamos comentado anteriormente, el modelo de depuración *BuggTracker* ha sido esencial en el desarrollo de esta aplicación y no hubiese sido lo mismo sin esta metodología de trabajo, tanto, que se seguirá utilizando y se utilizará para el desarrollo por parte del autor, ya que permite renovar y reciclar constantemente las ideas que te van bombardeando en la fase de pruebas, lo que amplía y mejora muchísimo la aplicación.

Aunque no se pudieron realizar las funcionalidades de extensión planificadas, se tienen añadidas en las propuestas de extensión cuando se vuelva a la fase de producción tras la finalización de este proyecto

10.2. Propuestas de extensión y mejoras futuras

Una vez finalizada la implementación de la App, se considera que la aplicación con los requisitos mínimos planificados han sido cumplidos, en cambio, esto no quiere decir que no se deje una gran lista de opciones y mejoras a aplicar, añadiendo también la serie de funcionalidades que se

tiene pensado en un futuro implementar con tal de llevar SocialMusFest a una fase de explotación.

La principal mejora a desarrollar sería implementar un sistema de red social, donde los usuarios al tener ya un perfil de usuario, puedan interactuar entre ellos, puedan ver sus perfiles, seguirse mutuamente y de ello la inclusión de otras tantas funcionalidades. La ventana de búsqueda de eventos se tiene pensado modificar una vez introducida la anterior característica para que un usuario pueda buscar eventos también por la asistencia de su círculo de amigos a los que sigue.

Otra característica muy importante que se ha observado es la necesidad de incluir dos nuevos role, el usuario que es un organizador o empresa, y el usuario cliente que busca los servicios que ofrece dicho organizador o empresa. Al mismo tiempo se desea también incluir una búsqueda de organizadores en la aplicación, el motivo de ello es que un usuario si estuviera interesado por ejemplo en seguir los eventos festivos de una empresa concreta porque le gustan todos los eventos que dicha empresa realiza, poder también seguir a ese "organizador". No sería lo mismo seguir a un usuario cliente que a un usuario "organizador", el primero únicamente sería para compartir aficiones y poder asistir en conjunto a conciertos o festivales y el segundo, para ser notificado en tiempo real cuando un "organizador" llegue a crear un nuevo evento para tener lo más rápidamente conocimiento de ello.

En la ventana de "ver usuario" que se tiene pensado incluir en la aplicación, al mismo tiempo se desea mostrar (si el usuario lo permite habilitando la opción), los eventos que haya publicado pudiendo convertirse en un "organizador", los eventos a los que ha asistido y los eventos en los que se ha registrado, además de poder verse otras características como cuantos seguidores tiene o a cuántos usuarios sigue.

Repasemos hasta el momento, con el añadido de las características citadas, se requerirá la modificación de la página principal mostrándose tres sub-ventanas que serán "Amigos", "Eventos" y "Organizadores". Cada sub-ventana al mismo tiempo tendrá la posibilidad de ver los usuarios a los que se sigue, ver los eventos por localización y las organizadoras que se sigue. Debajo de cada sub-ventana los correspondientes botones "Buscar más amigos", "Buscar más eventos", "Buscar más publicadores", siendo la búsqueda de amigos por ejemplo a través de la lista de contactos de Google, Facebook y otras páginas-servicios que se estudia implementar.

Ya se tienen realizados los prototipos a papel de las ventanas previamente citadas, los usuarios *testers* encargados de probar SocialMusFest App están encantados de seguir participando en el proyecto aunque se reconoce que este tema debe ser ampliado y llevado a cabo a través de entrevistas rápidas y programas de reporte de *buggs* en la aplicación de forma interna.

Para finalizar, se pretende en un futuro no muy lejano, subir la aplicación a Google Play Store para su fase de explotación, ya que por desgracia por falta de tiempo no se ha podido realizar en este proyecto.

Bibliografía

[1] Sashen Govender, Comenzando con Android Studio. URL: http://code.tutsplus.com/es/tutorials/getting-started-with-android-studio--mobile-22958

[2] Salvador Gómez Oliver, Cursos de programación Android. URL: http://www.sgoliver.net/blog/

[3] Stack Exchange, Inc. Stackoverflow. URL: http://stackoverflow.com/

[4] Google Inc.; Activity. URL: https://developer.android.com/reference/android/app/Activity.html

[5] James Revelo, Uso De Recursos En Android. URL: http://www.hermosaprogramacion.com/2015/08/uso-de-recursos-en-android/

[6] Carlos Toxtli, Todas las APIs de Google. URL: http://es.slideshare.net/carlostoxtli/all-the-google-ap-is

[7] Delight, Android-DDP. URL: https://github.com/delight-im/Android-DDP

[8] MongoDB, Inc. Documentación: https://docs.mongodb.com/

[9] Google Inc. Tabs. URL: https://material.google.com/components/tabs.html

[10] Ravi Tamada, Trabajar con Material Designs. URL: http://www.androidhive.info/2015/04/android-getting-started-with-material-design/

[11] James Revelo, Creación De Action Bar En Material Design. URL: http://www.hermosaprogramacion.com/2015/06/toolbar-en-android-creacion-de-action-bar-en-material-design/

[12] Google Inc. Google API Client. URL: https://developers.google.com/api-client-library/java/google-api-java-client/oauth2

[13] Google Inc. Integrating Google Sign-In into Your Android App. URL: https://developers.google.com/identity/sign-in/android/start-integrating

[14] Google Inc. Google APIs Console. URL: https://console.developers.google.com/apis

[15] Numetric Technologies Pvt Ltd, Android Google Plus Integration and Login Tutorial. URL: https://www.numetriclabz.com/android-google-plus-integration-and-login-tutorial/

[16] Salvador Gómez Oliver, Primeros pasos en SQLite. URL: http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/

[17] Belal Khan, Google Login Android Tutorial – Integrate GPlus Login. URL: https://www.simplifiedcoding.net/google-login-android-tutorial-integrate-gplus-login/

[18] Facebook, Inc. Facebook for developers. URL: https://developers.facebook.com/apps

[19] Facebook, Inc. Facebook Login for Android. URL: https://developers.facebook.com/docs/facebook-login/android

[20] Ashraff Hathibelagal, Login con Facebook en Android: http://code.tutsplus.com/es/tutorials/quick-tip-add-facebook-login-to-your-android-app--cms-23837

[21] GitHub, Inc. Getting Started with GitHub Desktop. URL: https://help.github.com/desktop/guides/getting-started/

[22] GitHub, Inc. Work with SSH Key. URL: https://help.github.com/categories/ssh/

[23] Leonardy Kristianto, Meteor on Heroku. URL: https://medium.com/@leonardykris/how-to-run-a-meteor-js-application-on-heroku-in-10-steps-7aceb12de234#.45a2uua87

[24] Meteor Development Group, Meteor API. URL: http://docs.meteor.com/index.html

Anexos

Documentos usados para la gestión

ACTA DE REUNIÓN 02/02/2016

Reunidos en la Facultad de Informática de San Sebastián, el día 02 de Febrero de 2016 a las 12:00 horas, con la asistencia de los participantes enumerados a continuación, la reunión sobre el proyecto SocialMusFest trató y acordó las siguientes cuestiones:

ASISTENTES:

- Jon Junguitu
- Rachid Boudhar
- José Ángel Vadillo

ORDEN DEL DÍA:

1. Propuesta de PFG.
2. División de trabajo.

ACUERDOS:

1. Se presenta por parte de Jon Junguitu y Rachid Boudhar una propuesta de PFG conjunto. La idea es la de realizar una aplicación, tanto en web como en Android, sobre festivales de música. La motivación de realizar la aplicación está presente en un documento que se le hace llegar a José Ángel Vadillo, el cual lo ha leído y acepta la idea. Se discuten algunos detalles de la aplicación. Finalmente, se acepta por parte de todos los participantes el documento existente.

2. De cara a realizar un proyecto en conjunto, Rachid Boudhar indica que el hace la aplicación para dispositivos Android y, por consiguiente, Jon Junguitu lo realiza para web. Este último propone la idea de MEAN, la cual es aceptada por José Ángel Vadillo. Tratados todos los temas, finaliza la reunión a las 12:45 horas del día citado, estando todos los presentes de acuerdo con las decisiones tomadas.

ACTA DE REUNIÓN 15/03/2016

Reunidos en la Facultad de Informática de San Sebastián, el día 15 de Marzo de 2016 a las 17:30 horas, con la asistencia de los participantes enumerados a continuación, la reunión sobre el proyecto SocialMusFest App trató y acordó las siguientes cuestiones:

ASISTENTES:

- Rachid Boudhar Tannaoui
- José Ángel Vadillo

ORDEN DEL DÍA:

1. Seguimiento del PFG.
2. Siguientes pasos del PFG.

ACUERDOS:

1. José Ángel Vadillo le indica a Rachid Boudhar Tannaoui que el trabajo que se esté realizando, se vaya entregando en un futuro. Por cada fase que se vaya desarrollando ir entregando partes para la corrección gradual.

2. Rachid Boudhar Tannaoui, indica que algunos apartados ya los esta realizando directamente en la memoria, se tiene la estructura de la misma, y se ha desarrollado hasta el momento el estudio de las tecnológicas, estando ya por terminar la fase de captura de requisitos y señalando que en próximas semanas se pasará a la fase de aprendizaje de las tecnologías a utilizar.

Tratados todos los temas, finaliza la reunión a las 17:00 horas del día citado, estando todos los presentes de acuerdo con las decisiones tomadas.

ACTA DE REUNIÓN 19/04/2016

Reunidos en la Facultad de Informática de San Sebastián, el día 19 de Abril de 2016 a las 10:00 horas, con la asistencia de los participantes enumerados a continuación, la reunión sobre el proyecto SocialMusFest trató y acordó las siguientes cuestiones:

ASISTENTES:
Jon Junguitu
Rachid Boudhar

ORDEN DEL DÍA:

1. Acordar detalles conjuntos de las aplicaciones.
2. Acordar una base de datos conjunta.
3. Acordar lugar para alojar la aplicación web.

ACUERDOS:

1. Se discuten los detalles de la aplicación.
2. En cuanto a la base de datos, se comparan los atributos introducidos en ambos proyectos. Primero se escogen los atributos más importantes. Una vez escogidas las comunes, se discuten el resto. Tras debatir, se crea una estructura común de base de datos que se crea en la aplicación web, y a la que tendrá acceso la aplicación Android mediante el acceso al servidor donde esté alojada la aplicación web.
3. En cuanto al alojamiento, se escogen una serie de opciones para ir comprobando su posible uso. Durante el transcurso de los siguientes días, entre los dos asistentes a la reunión se van a intentar las opciones planteadas.

- Meteor Server
- Hostinger
- Cloud9
- Heroku
- Scalingo

Tratados todos los temas, finaliza la reunión a las 12:30 horas del día citado, estando todos los presentes de acuerdo con las decisiones tomadas.

ACTA DE REUNIÓN 16/06/2016

Reunidos en la Facultad de Informática de San Sebastián, el día 16 de Junio de 2016 a las 11:00 horas, con la asistencia de los participantes enumerados a continuación, la reunión sobre el proyecto SocialMusFest App trató y acordó las siguientes cuestiones:

ASISTENTES:

- Rachid Boudhar Tannaoui
- José Ángel Vadillo

ORDEN DEL DÍA:

1. Seguimiento del PFG.
2. Cambios en la memoria.

ACUERDOS:

1. José Ángel Vadillo le indica a Rachid Boudhar Tannaoui las correcciones a realizar en la memoria, entre ellos la modificación de los casos de uso según el estado del usuario ya que no se entienden, el uso de enlaces *extends* e *includes* mal utilizados y la modificación de algunos casos de uso, moverlos algunos directamente al actor sin realizar tantas derivaciones. En el capítulo de diseño, modificar el objeto "aplicación" por "sistema" y la eliminación de las funciones que interactúan con el sistema, siendo necesaria la capa *GUI* de la aplicación.

2. Rachid Boudhar Tannaoui queda pendiente de realizar los cambios previamente citados para el lunes 20 de junio del 2016 y entrega a las 9:00 de la mañana para una siguiente revisión.

3. Rachid Boudhar Tannaoui también le presenta la aplicación final realizada a José Ángel Vadillo. El alumno le muestra la aplicación y ambos interactúan con ella. Quedando por implementar algunas características añadidas no comprendidas en el mínimo exigible para completar la aplicación definitivamente. José Ángel Vadillo señala que por el momento se realice la corrección de la memoria como prioridad ya que después se tendrá el tiempo hasta la fecha de defensa de implementar esas pequeñas características.

Tratados todos los temas, finaliza la reunión a las 17:00 horas del día citado, estando todos los presentes de acuerdo con las decisiones tomadas.