Carlos Licha de la Encarnación |Manuel González Saavedra

Escuela Politécnica Curso 2018/2019

Descripción breve

Aplicación desarrollada para entornos Android para consultar festivales y conciertos en diferentes provincias de España

Entrega seguimiento Asee

FestivApp v1.0

Contenido

[Descripción de la idea 2](#_Toc529954137)

[Funcionalidad principal 2](#_Toc529954138)

[Público objetivo 2](#_Toc529954139)

[Análisis 2](#_Toc529954140)

[Requisitos funcionales 2](#_Toc529954141)

[Requisitos no funcionales 3](#_Toc529954142)

[Diseño 3](#_Toc529954143)

[Diseño de la IU 4](#_Toc529954144)

[Mapa de navegación 4](#_Toc529954145)

[Mockup/Wireframe de la aplicación 4](#_Toc529954146)

[Implementación 6](#_Toc529954147)

[Decisiones 6](#_Toc529954148)

[Vistas modulares de detalle de los componentes. 7](#_Toc529954149)

[Información básica de implementación. 7](#_Toc529954150)

# Descripción de la idea

El proyecto que estamos desarrollando para la asignatura de Arquitecturas Software para Entornos Empresariales es una aplicación para entornos móviles Android que incluya funcionalidad completa de acceso a base de datos, *networking*, tareas asíncronas y patrones de diseño singulares en Android. Además, también pretende que nos desenvolvamos en el desarrollo de interfaces gráficas (*Activities*) valiéndonos de principios como *Material Design 2.0*

## Funcionalidad principal

La funcionalidad principal de la aplicación consiste en la consulta de eventos, que pueden ser festivales y conciertos, a partir de una configuración (Nombre, provincia, fecha de inicio, etc.) Además, estos eventos pueden almacenarse localmente guardándose en categorías como *Favoritos*, *Eventos próximos* o *Asistidos*. Podrá verse toda la información de un evento, que será proporcionada por la API de *SongKick.*

## Público objetivo

Corresponde a personas, generalmente jóvenes o aficionados a la música, que suelan con frecuencia asistir a conciertos de grupos en su provincia o en otras provincias de España, de manera que con esta aplicación tengan más fácil y rápido el acceso a la información: dónde es, cuándo y quiénes participan.

# Análisis

## Requisitos funcionales

### Buscador:

* R1- Buscar eventos(Festival/Concierto) por nombre.
* R2- Buscar eventos(Festival/Concierto) por provincia.
* R3- Buscar eventos(Festival/Concierto) por fecha de inicio y/o finalización
* R4- Poder elegir si se quiere buscar, Festival, Concierto o ambos.

### Menú Buscador:

* R5- Acceder a la lista de eventos Favoritos
* R6- Acceder a la lista de eventos Pendientes
* R7- Acceder a la lista de eventos Asistidos

### Lista de eventos:

* R8- Acceder a un evento de la lista pulsandolo.
* R9- Alternar entre páginas de la lista en caso de que las haya.

### Detalles del evento:

* R10- Operaciones *CRUD* para gestionar lista de Favoritos.
* R11- Operaciones *CRUD* para gestionar lista de Pendientes.
* R12- Operaciones *CRUD* para gestionar lista de Asistidos.
* R13- Consultar información relevante al evento.
* R14- Acceder a la página de *SongKick* del evento en cuestión mediante un botón.
* R15- Poder cambiar el idioma de la aplicación.
* R16- Recibir notificaciones cuando un evento de interés se acerque.

## Requisitos no funcionales

* Aplicación accesible e intuitiva mediante el uso de *Material Design*.
* Uso de base de datos mediante la *API SQLite*.
* Obtención de los datos mediante llamadas *REST* a la API *SongKick*.
* Uso fluido de la aplicación, sin pantallas de carga ni retrasos.
* Programación de la lógica en *Java* y en *Kotlin*, haciendo uso de la interoperabilidad.
* Patrones de diseño para mejorar la calidad del código.

# Diseño

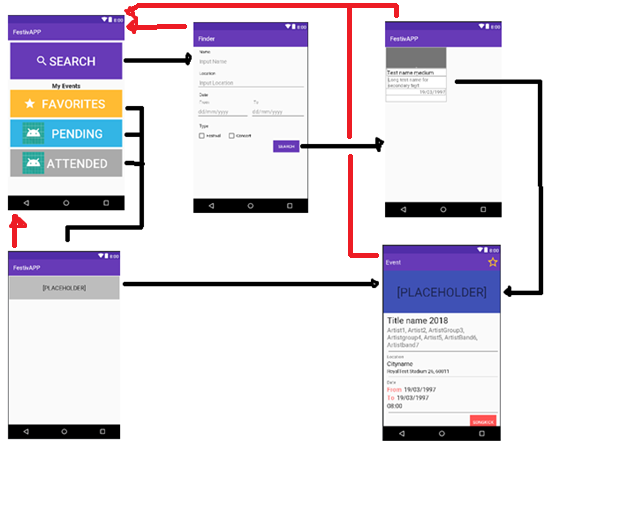
La arquitectura de la aplicación sigue el patrón MVC (Modelo-Vista-Controlador)

Analizamos la estructura:

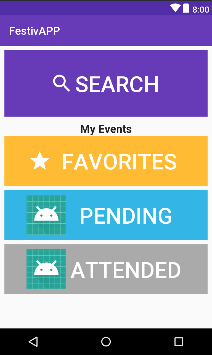
* Modelo: El modelo es la capa de datos. Los datos que van a guardarse es la información de un evento, ya sea concierto o festival, que consta de un nombre, ciudad, provincia, fecha de inicio y los artistas que participan, obtenidos mediante una API. Forma parte el paquete *database*.
* Vista: La vista es la capa relacionada con la IU. Forman parte las activities principal, activity de búsqueda de un evento, la activity que lista estos eventos y la activity de detalle de un evento.
* Controlador: El controlador contiene procedimientos que relacionan la vista y el modelo de datos. Principalmente son los adapters de la aplicación, junto con los métodos CRUD de cada una de las tablas de la base de datos.

## Diseño de la IU

### Mapa de navegación

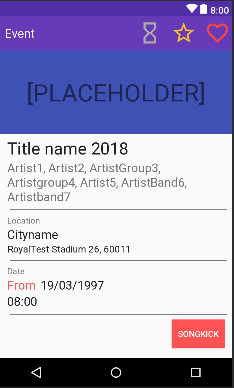


### Mockup/Wireframe de la aplicación

 MainActivity

Este es el Activity principal que se cargará cuando se inicie la aplicación. Permite navegar hacia los eventos marcados como Favoritos, eventos que aún no próximos (Pending) y eventos a los que ya se haya asistido (Attended).

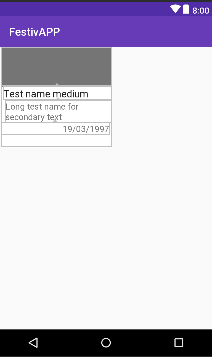
También se puede acceder a una búsqueda por diferentes cambios en Search

EventActivity

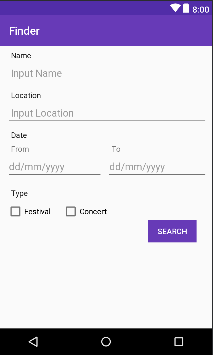
Esta Activity permite mostrar la información de un evento concreto: su nombre, los artistas que participan, la ciudad en la que se desarrolla, el lugar exacto en esa ciudad y la fecha y hora de comienzo.

En la AppBar se implementan las opciones de añadir el evento a Favoritos

También se implementa la posibilidad de acceder a su página en SongKick

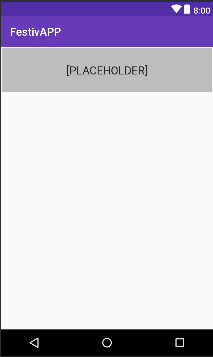
EventListActivity

Este Activity muestra todos los eventos según la búsqueda que se haya hecho en el FinderActivity. Si se presiona sobre uno, llevará a los detalles del evento.

FinderActivity

Este Activity permite buscar un evento accediendo a la API de SongKick. Permite hacer una búsqueda según el nombre del evento, localización (campo obligatorio), y fecha de comienzo o cierre.

También permite discriminar si el evento es un festival, concierto o ambos.



EventListActivity

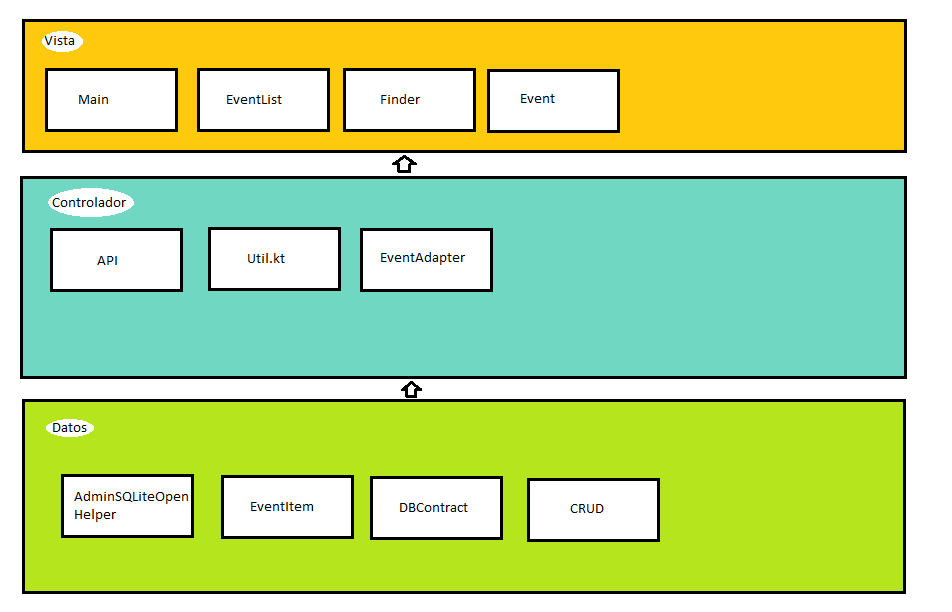
La lista de eventos es accesible desde el buscador o desde las opciones de Favoritos, Asistidos o Pendientes en ella se mostrarán las distintas previews de los eventos que coinciden con los criterios de búsqueda.

# Implementación

## Decisiones

* En esta primera versión, se ha decidido implementar la base de datos sin ningún patrón de diseño (Room, GreenDao) Corresponde a la siguiente entrega realizar el cambio a Room.
* El acceso a networking mediante API se realiza con llamada asíncrona, eligiendo la API de *SongKick*
* El lenguaje utilizado será Kotlin en algunos módulos y Java en otros.
* Se ha optado por construir principalmente Activitites en vez de Fragments.
* Se ha seguido el patrón MVC para el desarrollo de la aplicación
* Se ha decidido implementar los DAO mediante Singleton.

## Vistas modulares de detalle de los componentes.



## Información básica de implementación.

Básicamente, la aplicación consta de:

* Paquete *database*. Contiene el acceso a base de datos
  + AdminSQLiteOpenHelper: Contiene la creación y destrucción de las tablas
  + DBContract: Contiene los nombres que se usarán en la transformación Java-SQL
  + EventItem: Objeto que contiene la información de un evento, que guardará los campos del JSON que se obtenga en las llamadas a la API
  + FavoritesCRUD: Implementación DAO para los eventos de la tabla favoritos: devolver eventos de esa tabla según id, insertarlos, obtenerlos todos…
  + PendingCRUD: Implementación DAO para los eventos de la tabla pendientes: devolver eventos de esa tabla según id, insertarlos, obtenerlos todos…
  + PastCRUD: Implementación DAO para los eventos de la tabla de eventos pasados: devolver eventos de esa tabla según id, insertarlos, obtenerlos todos…
* Util.kt: Permite establecer el formato de fecha
* SongkickAPI: Implementación REST para obtener y procesar el fichero JSON de la página web de Songkick
* MainActivity: Pantalla principal de la aplicación con acceso al buscador y a las pestañas de Favoritos, Pendientes y Pasados.
* EventActivity: Pantalla con la información de un evento
* Finder: Realiza la llamada a la API una vez se han insertado los parámetros de búsqueda