UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"DESARROLLO DE UNA APLICACIÓN ANDROID PARA COMPARTIR EVENTOS DEPORTIVOS"

TRABAJO FIN DE GRADO

Enero -2020

AUTOR: Carlos Villalba Baeza

DIRECTOR/ES: César Fernández Peris

INDICE

AGRADECIMIENTOS

Para comenzar me gustaría agradecer a mi familia y a mi novia por todo el apoyo recibido durante los años que he realizado este grado. Por aguantarme en los momentos de agobios y siempre intentar que me sintiese mejor y que confiera en mí. Darme motivación para continuar esforzándome e invertir en mi futuro como lo es realizar una ingeniería que no siempre es fácil.

También mencionar a mis amigos desde pequeños que siempre que necesitaba algo estaban ahí para apoyarme. Y para la gente que he conocido a lo largo del grado que han hecho de las clases y el estudio más fácil. Sobre todo, el grupo más cercano a mí con el que he compartido muchas horas de estudio y nos hemos ayudado mucho entre nosotros.

Agradecer a César Fernández Peris, profesor de la UMH y mi tutor de TFG. Gracias por guiarme a lo largo de la realización del presente proyecto y por ayudarme cuando estaba estancado con la aplicación.

Finalmente, agradecer a la Universidad Miguel Hernández de Elche y a todo su profesorado y demás personas que trabajan en ella que nos hacen el día un poco más fácil a los estudiantes.

RESUMEN

En el presente proyecto, "Desarrollo de una aplicación Android para compartir eventos deportivos", se explicará cómo ha sido la programación y desarrollo de la aplicación que tiene como fin que los usuarios puedan publicar eventos deportivos y otros usuarios puedan apuntarse a dichos eventos.

La aplicación tiene el nombre de “Pachanga”. Desde esta aplicación tendrás tres opciones principales:

- Publicar partido: te permitirá publicar un evento deportivo que estas organizando.

- Mis anuncios: te permitirá acceder a los eventos que hayas publicado y poder modificarlos o eliminarlos.

- Buscar partido: podrás buscar eventos que se pueden ver en un mapa y apuntarse a dichos eventos.

El presente trabajo ha sido posible gracias a un anuncio de César Fernández Peris, profesor de la Universidad Miguel Hernández de Elche en el área de Ingeniería de Sistemas y Automática. El cual me ha permitido desarrollar mi propia propuesta, “Pachanga”.

Con la presente memoria se realiza con el fin de explicar cómo ha sido el proceso de programación y diseño de las interfaces de la aplicación.

1.- INTRODUCCIÓN

El presente documento tiene como objetivo describir el Trabajo de Fin de Grado (TFG) realizado por Carlos Villalba Baeza alumno del Grado en Ingeniería Electrónica y Automática Industrial en la Universidad Miguel Hernández de Elche.

"Pachanga" es una aplicación para dispositivos Android, desarrollada en la plataforma de desarrollo oficial de dicho sistema operativo, Android Studio. Dicha aplicación es capaz de publicar partidos y poder apuntarse a partidos publicados por otros usuarios.

Además, se podrán modificar o eliminar los anuncios que el usuario haya publicado. Y una vez que todas las vacantes del partido estén completas, este será eliminado. Los partidos se mostrarán en un mapa de Google Maps donde también aparecerá la ubicación actual del usuario pudiendo ver donde tiene eventos cercanos a su posición. Y a los que se podrá apuntar gracias a una lista donde muestra los eventos y los datos de cada evento que se le muestra en el mapa.

1.1.- MOTIVACIÓN Y OBJETIVOS DEL PROYECTO

En la actualidad, nos encontramos en una sociedad cada vez más vinculada a la tecnología, la cual va evolucionando continuamente y hace que las personas confíen cada más en ella para delegar actividades de su vida cotidiana, y a su vez, así facilitarles la vida.

Hoy en día todo el mundo tiene Smartphones, ya que es actualmente es la forma de comunicarse más fácil, cómoda y rápida. Además, podemos encontrar infinitud de aplicaciones para ocio, comunicación y diversión.

Al ser estudiante del Grado en Ingeniería Electrónica y Automática Industrial, he aprendido diversos lenguajes de programación impartidos a lo largo del grado. Pero nunca había programado en Java. Cuando consulte las propuestas de Trabajos de Fin de Grado opté por la oferta de César, ya que en la primera reunión que tuve con el me transmitió seguridad y sentí que me ayudaría. Además, era una oportunidad de aprender un nuevo lenguaje de programación y aprender como es el funcionamiento de las aplicaciones y como es el proceso de diseño.

En un principio no se me ocurrió ninguna idea para desarrollar la aplicación y un día con mis amigos nos faltaba gente para poder jugar un partido con lo que me vino a la mente que sería una buena idea crear una aplicación que permitiera conectar a personas que buscaban un partido con otras que necesitaban esas personas.

Estuve indagando, encontré "Timpik" una aplicación, con un funcionamiento parecido al que me vino a la mente. Con lo que me sirvió de inspiración y a raíz de esa aplicación poder crear los cimientos de "Pachanga".

1.2.- CONTEXTO DE DESARROLLO

Como ya he mencionado anteriormente, la aplicación ha sido desarrollada para el sistema operativo Android.

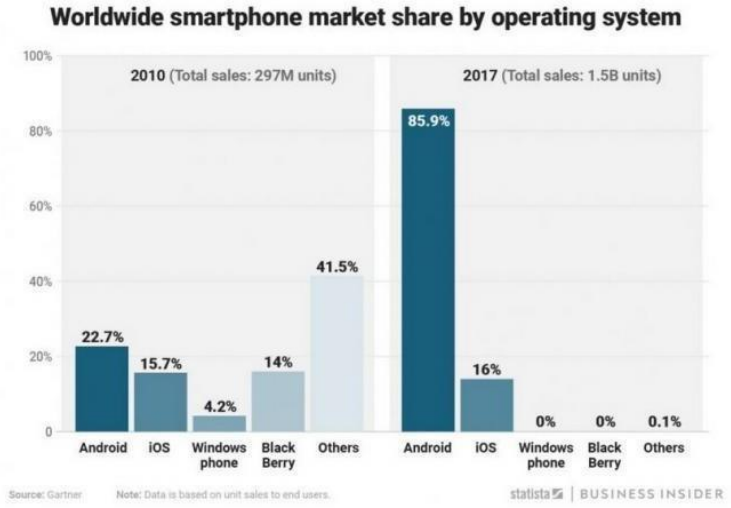
Android es un sistema operativo móvil desarrollado por Google, basado en Kernel de Linux y otros softwares de código abierto. Fue diseñado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas, relojes inteligentes, automóviles y televisores.

Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y que adquirió en el año 2005.

Android fue presentado en 2007 junto con la fundación del Open Handset Alliance un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio, para avanzar en los estándares abiertos de los dispositivos móviles. El mismo día se anuncia la primera versión del sistema operativo: Android 1.0 Apple Pie. Los terminales con Android no estarían disponibles hasta el año 2008, siendo el HTC Dream el primer dispositivo vendido con sistema operativo Android en octubre de 2008.

La implantación de Android ha sido progresiva desde unos inicios marcados por una gran fragmentación en la elección de cada fabricante en el sistema operativo escogido para sus dispositivos.

Actualmente, Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado superior al 85%, muy por encima de IOS, el sistema operativo para sistemas móviles de la compañía Apple y segundo sistema operativo más extendido.



El historial de versiones de sistema operativo Android se inició con el lanzamiento de Android beta en noviembre de 2007. La primera versión comercial , Android 1.0, fue lanzada en septiembre de 2008. La cual ha tenido actualizaciones a su sistema operativo base que normalmente tenían la finalidad de corregir fallos del programa y agregan nuevas funcionalidades.

Desde abril de 2009 hasta septiembre de 2019, las versiones de Android han sido desarrolladas bajo un nombre en clave y sus nombres siguen un orden alfabético: Apple Pie, Banana Bread, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo y Pie.

El 3 de septiembre de 2019, durante la ceremonia en Mountain View, Google rompe con la tradición de nombrar a su sistema operativo como un postre y anuncia Android 10 como nombre oficial de la nueva versión Android. De esta manera, Google numerará consecutivamente las próximas versiones de su sistema operativo para smarthphone y tablet.

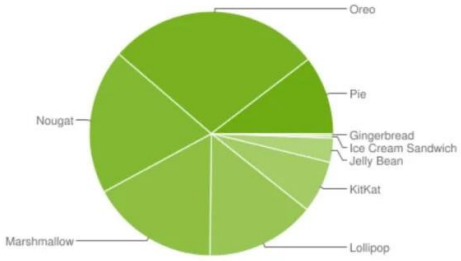
Nombre código	Número de versión	Fecha de lanzamiento	Nivel de API
Apple Pie[1]	1.0	23 de septiembre de 2008	1
Banana Bread[1]	1.1	9 de febrero de 2009	2
Cupcake	1.5	25 de abril de 2009	3
Donut	1.6	15 de septiembre de 2009	4
Eclair	2.0 – 2.1	26 de octubre de 2009	5 – 7
Froyo	2.2 – 2.2.3	20 de mayo de 2010	8
Gingerbread	2.3 – 2.3.7	6 de diciembre de 2010	9 – 10
Honeycomb[2]	3.0 – 3.2.6	22 de febrero de 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.5	18 de octubre de 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	9 de julio de 2012	16 – 18
KitKat	4.4 – 4.4.4	31 de octubre de 2013	19 – 20
Lollipop	5.0 – 5.1.1	12 de noviembre de 2014	21 – 22
Marshmallow	6.0 – 6.0.1	5 de octubre de 2015	23
Nougat	7.0 – 7.1.2	15 de junio de 2016	24 – 25
Oreo	8.0 – 8.1	21 de agosto de 2017	26 – 27
Pie	9.0	6 de agosto de 2018	28
Android 10 [3]	10.0	3 de septiembre de 2019	29

El problema con las versiones de Android es que no todos los dispositivos se actualizan a la última versión, y esto, unido al poco espacio temporal entre unas versiones y otras, hacen que haya muchos dispositivos operativos con distintas versiones de Android.

Este factor se ha de tener en cuenta a la hora de desarrollar en Android, cada versión de Android tiene un nivel de API o Application Programming Interface (interfaz de programación de aplicaciones), esto es importante porque utilizar un nivel u otro de API puede hacer que una aplicación no pueda ser utilizada en versiones más antiguas de Android, pero por otro lado cuanto mayor sea el nivel de API mayor número de ventajas y recursos para el desarrollador.

Por este motivo hay que alcanzar un compromiso entre las ventajas que puedan ofrecer las nuevas versiones de Android y el número de dispositivos en el mercado con un sistema operativo capaz de ejecutar tu aplicación.

Version	October 2018 distribution (%)	Current distribution (%)	Change (pp)
2.3	0.2	0.3	+0.1
4.0	0.3	0.3	0
4.1	1.1	1.2	+0.1
4.2	1.5	1.5	0
4.3	0.4	0.5	+0.1
4.4	7.6	6.9	-0.7
5.0	3.5	3	-0.5
5.1	14.4	11.5	-2.9
6.0	21.3	16.9	-4.4
7.0	18.1	11.4	-6.7
7.1	10.1	7.8	-2.3
8.0	14	12.9	-1.1
8.1	7.5	15.4	+7.9
9	N/A	10.4	+10.4



2.- DESCRIPCIÓN DE LA APP

La aplicación se llama “Pachanga” y ha sido creada para compartir eventos deportivos y también para poder apuntarnos a dichos eventos. Estos eventos son mostrados en un mapa.

La aplicación desarrollada en Android Studio cuenta con una serie de actividades que le dan la funcionalidad a la aplicación. Cada actividad cuenta con un archivo .java que da funcionalidad a los elementos que son definidos en el archivo del layout.

Comenzando por la actividad “Principal”, que es la actividad que nos encontramos al iniciar la aplicación:

En esta actividad hay cuatro elementos. Un TextView con el nombre de la aplicación y tres botones que cada uno no lleva a diferentes actividades.

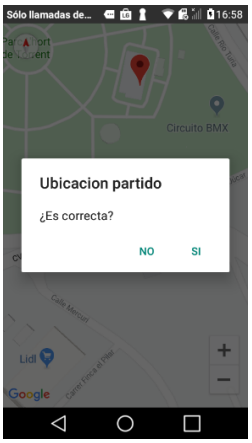
Al pulsar “Publicar anuncio” iremos a la actividad “SeleccionarUbicacionPubAnu”, al pulsar “Mis anuncios” nos conducirá a la actividad “MisAnuncios” y cuando se pulsa “Buscar partido” iremos a la actividad “BuscarPartido”.

En la actividad “SeleccionarUbicacionPubAnu” nos encontraremos con la siguiente interfaz:

Esta actividad es previa a la actividad “PublicarAnuncio”, y nos permite situar el evento en el mapa para luego enviar la ubicación a la actividad “PublicarAnuncio”.

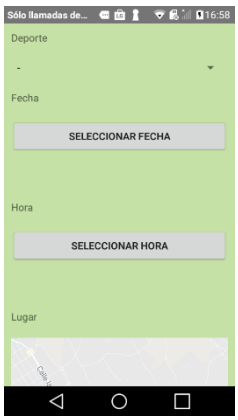
La actividad muestra un mapa, centrado en Alicante. Donde podemos desplazarnos deslizando el dedo, hacer zoom con gesto o usando los botones situados en la parte inferior derecha de la pantalla.

Una vez situado el punto donde queremos pulsamos en el mapa sobre la localización del evento, de tal forma que aparecerá un marcador y un letrero como se muestra:



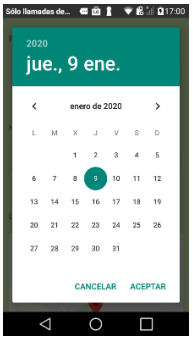
Como se muestra aparece un dialogo que no pregunta si es correcta la ubicación del evento. Si nuestra respuesta es no, borrar el marcador que se muestra en la imagen y nos dejara marcar de nuevo en el mapa. Por el contrario, si la respuesta es sí, nos enviara a la actividad “PublicarAnuncio” y mandara la ubicación a esta actividad.

Una vez en “PublicarAnuncio” deberemos de rellenar todos los campos que nos piden para publicar el anuncio:



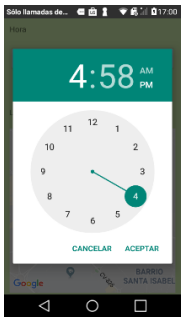
Para rellenar todos los campos, lo primero que nos encontramos es un menú desplegable que nos muestra los deportes que podemos escoger:

A la hora de seleccionar la fecha, pulsamos sobre el botón “Seleccionar fecha” y aparecerá la siguiente ventana:



Donde podemos escoger la fecha cambiando el mes, año y pulsando sobre el día. Una vez seleccionada aparecerá en un TextView la fecha para que podamos observar que es la fecha correcta.

Para seleccionar la hora, pulsamos sobre el botón “Seleccionar hora” y aparecerá la siguiente ventana:



En la cual escogemos si es por la mañana AM y si es por la tarde PM. Luego seleccionamos la hora y una vez seleccionada la hora seleccionamos el minuto. Aceptamos y la hora aparecerá en un TextView de igual forma que la fecha.

Podremos ver en un Fragment de Google Maps la posición del evento y en dos TextView justo debajo la latitud y la longitud.

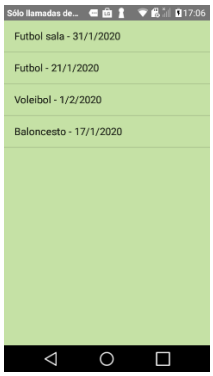
Introducimos el número de vacantes en el EditText a través de teclado del móvil y de igual modo la descripción.

Una vez rellenos todos los campos pulsamos sobre el botón “Publicar”. Nos enviara a la actividad “Principal” y nos mostrara un texto diciendo que el partido ha sido publicado.

En cambio, si no todos los campos se mostrara un texto como en la siguiente imagen:

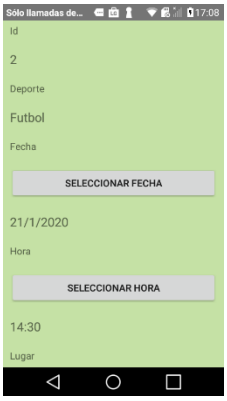
Para poder publicar el anuncio, todos los campos deben estar completados. Por otro lado, si pulsamos el botón “Cancelar” volveremos a la actividad “Principal” pero no se publicará el evento.

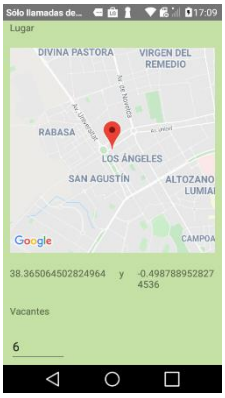
Desde la actividad “Principal” si pulsamos el botón “Mis anuncios” accederemos a la actividad “MisAnuncios”:

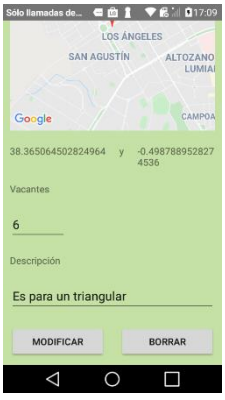


Podemos ver que se trata de un ListView que muestra todos los eventos que hemos publicado. Al pulsar sobre alguno de los eventos accederemos a la actividad “DetallesMisAnuncios”:

Donde podemos observar todos los detalles del evento que hemos seleccionado. Si pulsamos sobre el botón “Modificar/Eliminar” accederemos a la actividad “ModificarEliminar”:







Pulsando “Seleccionar fecha” o “Seleccionar hora” podremos cambiar la fecha y hora respectivamente de igual modo que la introducíamos en la actividad “PublicarAnuncio”.

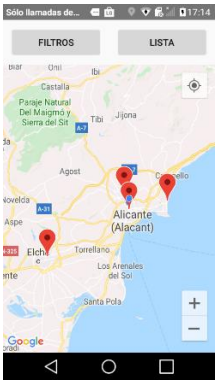
También podremos cambiar el numero de vacantes y la descripción. Lo único que no se puede modificar es el id, el deporte y la ubicación.

Para modificar el evento pulsamos sobre el botón “Modificar” que realizara los cambios que hayamos realizado y nos enviara a la actividad “Principal” y nos mostrara un texto que nos comunicara que el partido ha sido modificado. Si, por el contrario, pulsamos el botón “Borrar”, se eliminará el evento y nos enviara a la actividad “Principal” y nos mostrara un texto diciendo que el partido ha sido eliminado.

Por último, desde la actividad “Principal” si pulsamos el botón “Buscar partido” accedemos a la actividad “BuscarPartido”.

Lo primero que aparecerá cuando abramos la aplicación por primera vez y busquemos partido es:

Donde tenemos que permitir el uso de nuestra ubicación. Una vez el usuario permita el uso de su ubicación podremos observar:

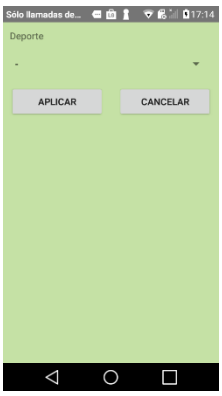


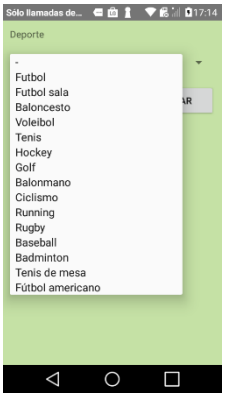
Se muestra un mapa con diferentes eventos que se sitúan con marcas. Si pulsamos sobre alguna marca podemos observar que aparece el deporte y la fecha del evento.

En la parte superior de la interfaz se localizan dos botones:

- “Filtros”: al pulsar accedemos a la actividad “FiltrosBuscarPartido”:

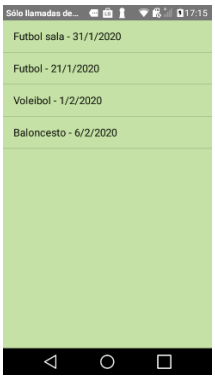
Esta actividad no tiene utilidad puesto que los filtros están desactivados. Pero si desplegamos el spinner podemos ver los diferentes deportes de los eventos:



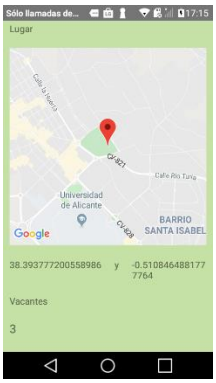


Pulsando “Cancelar” podemos volver a la actividad “BuscarPartido”.

- “Lista”: al pulsar accedemos a la actividad “ListaBuscarPartido”:



Desde esta actividad podemos ver una lista con los eventos que hay publicados. Si pulsamos sobre algún evento podremos acceder a la actividad “DetallesBuscarPartido”:



En esta actividad podemos observar todos los detalles del evento que hemos seleccionado. Si pulsamos sobre el botón “Apuntarse”, no apuntaremos al evento:

Como se muestra en las imágenes, volvemos a la actividad “Principal”, pero hay dos escenarios posibles. Que el una vez inscrito el usuario el número de vacantes no sea cero con que al volver a la actividad “Principal” se mostrara un mensaje diciendo que se ha apuntado al evento y el numero de personas que faltan. Si, por el contrario, el número de vacantes es cero una vez inscrito, al volver a la actividad “Principal” se mostrará el mensaje que has sido apuntado y además el evento ha sido completado.

3.- HERRAMIENTAS DE DESARROLLO EN EL PROYECTO

En este aparatado se definirá el programa de desarrollo dentro de la programación en Android, que ha servido para la creación del TFG.

3.1.- ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android, comercializado por Google. Reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software IntelliJ IDEA de JetBrains y ha sido diseñado específicamente para el desarrollo de Android. Utiliza una licencia de software libre Apache 2.0, está programada en Java y es multiplataforma. Tiene como principales características:

- Soporte para programar aplicaciones para Android Wear (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Herramientas Lint (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en Android.

- Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de Android Studio (Gradle) utiliza ANT.
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar Mercurial, Git, GitHub o Subversión.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml

Este entorno de desarrollo integra emuladores para probar las aplicaciones. Además, también es posible conectar dispositivos móviles reales al ordenador y, tras la instalación de los respectivos controladores, permite probar las aplicaciones en nuestros dispositivos, previamente habilitando las opciones de desarrollador de los dispositivos además de la depuración de USB.

En cuanto a los requisitos del sistema para su instalación y uso:

	Windows	Mac	Linux
Sistema Operativo	Microsoft Windows 7/8/10 (32 o 64 bits) *El emulador de Android solo es compatible con Windows de 64 bits	Mac OS X 10.10 (Yosemite) o superior, hasta 10.14 (masOS Mojave)	GNOME o KDE desktop
RAM	4GB de RAM mínimo, se recomienda 8GB de RAM		
Espacio de disco	2 GB de espacio disponible en disco mínimo, 4 GB recomendados (500 MB para IDE + 1.5 GB para Android SDK e imagen del sistema emulador)		
Resolución de pantalla	Resolución de pantalla mínima de 1280 x 800		

4.- MATERIALES Y MÉTODOS

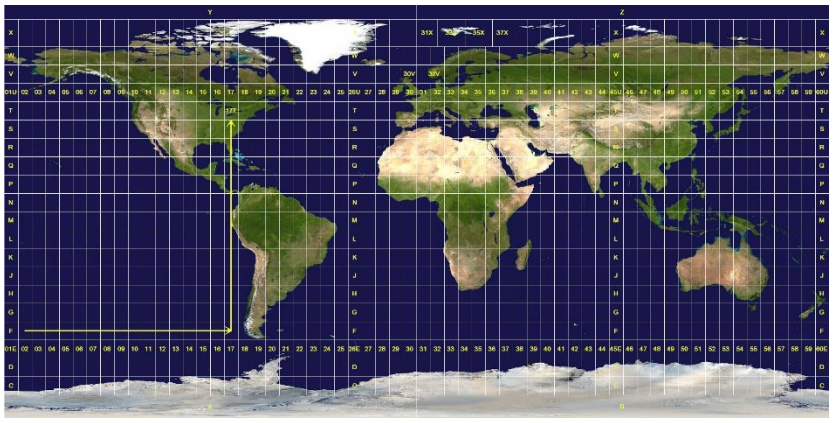
4.1.- GOOGLE MAPS

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. Que ofrece imágenes de mapas desplazables, fotografías por satélite de la Tierra, rutas entre diferentes ubicaciones, imágenes a pie de calle con Google Street View, condiciones de tráfico en tiempo real, un calculador de rutas a pie, bicicleta, coche y transporte público y un navegador GPS.

Ofrece la capacidad de acercar y alejar el mapa para mostrarlo con diferente zoom. Se puede manejar desde un Smartphone o desde un ordenador. Permite ingresar direcciones, intersecciones o áreas en general para buscar en el mapa. También establece las rutas más breves y convenientes desde un punto a otro que escojamos. Permite visualizar los caminos en 3D cuando es posible.

A lo que al sistema de coordenadas se refiere, Google Maps utiliza WGS84 (World Geodetic System 84). Sistema de coordenadas geográficas mundial que permite localizar cualquier lugar del mundo sin necesitar otro sitio de referencia. Se basa en el sistema de posicionamiento global GPS y esta expresado en grados, minutos y segundos. Para el calculo de superficies y distancias, se transforman estas coordenadas geodésicas en cartesianas, utilizando el sistema UTM (Universal Transverse Mercator). Basado en la proyección cartográfica transversa de Gerardus Mercator.

Divide la tierra en 60 husos de 6º de longitud, numerados del 1 al 60 en orden ascendente hacia el este. Y en cuanto a la latitud, se divide en 20 bandas de 8º denominadas con letras desde la C hasta la X (excluyendo I y O). Situando las menores de N en el hemisferios sur y las superiores en el hemisferio norte.



La imagen muestra los husos y zonas UTM de todo el mundo. Se trata de coordenadas latitud y longitud siendo positivas para Norte y Este, y negativas para Sur y Este.

4.2.- LENGUAJE DE PROGRAMACIÓN

Android Studio, software con el que hemos realizado el TFG trabaja con dos tipos de archivo xml y java.

4.2.1.- ARCHIVOS XML

Son los encargados de crear y diseñar los layouts del proyecto que el usuario visualiza e interactúa con ellos. Aunque es posible modificarlos desde las clases java, principalmente la parte estética es configurada en estos archivos.

4.2.2.- ARCHIVOS JAVA

Archivos en los cuales se programa el funcionamiento de los elementos que forman parte de la aplicación. Estos elementos deben estar declarados en los archivos xml, los cuales deben estar identificados por un “id”, para poder declararlos en las clases con la función findViewById.

4.3.- DISEÑO Y PROGRAMACIÓN

4.3.1.- ESTRUCTURA DEL PROYECTO

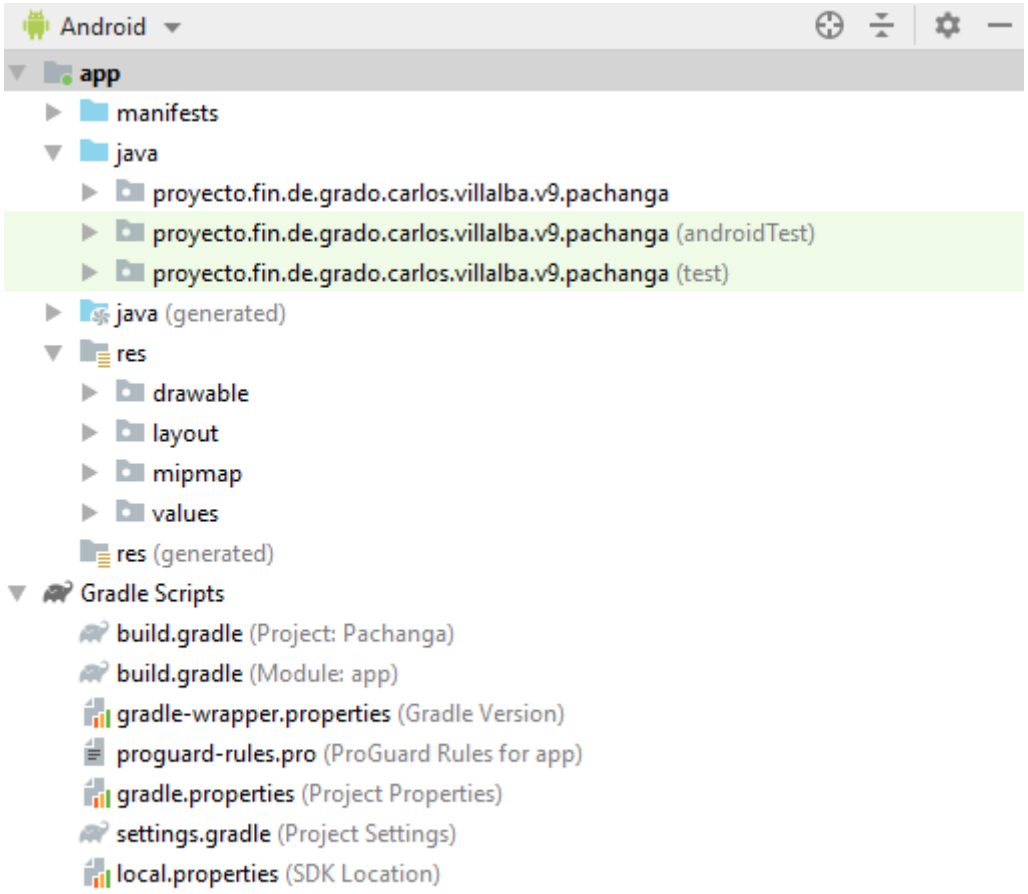
Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android.
- Módulos de bibliotecas.
- Módulos de Google App Engine.

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android, como se muestra en la ilustración siguiente. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de Secuencias de comando de Gradle y cada módulo de la aplicación contiene las siguientes carpetas:

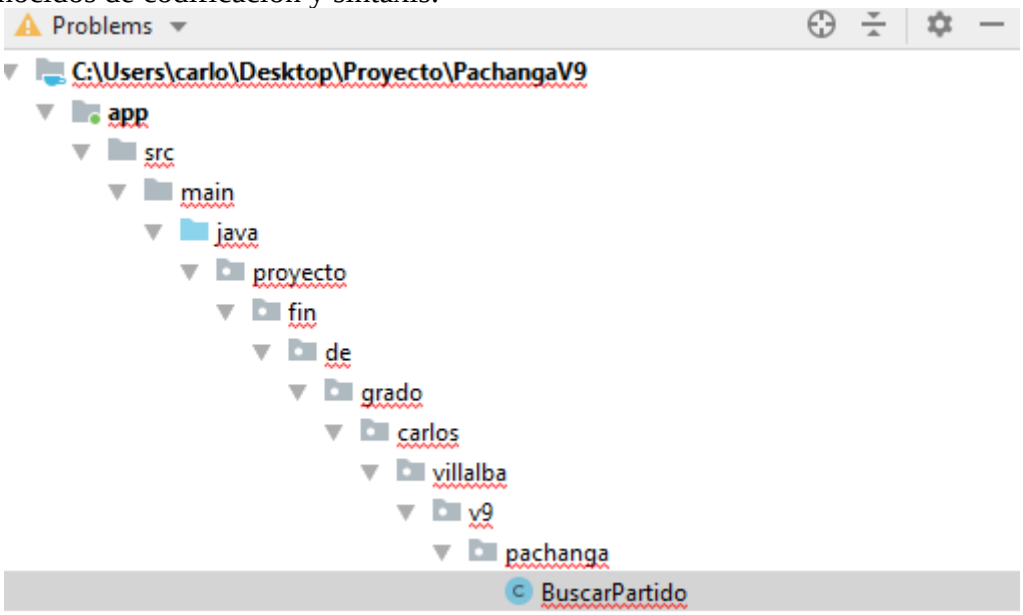
- Manifest: Contiene el archivo AndroidManifest.xml.
- Java: Contiene los archivos de código fuente de java, incluido el código de prueba.
- Res: Contiene todos los recursos, como los diseños XML, cadenas de IU e imágenes de mapa de bits.



La estructura del proyecto para Android en el disco difiere de esta representación plana. Para ver lo que sería el esquema de archivos real del proyecto, debemos seleccionar

Project en la lista desplegable arriba del todo, en la ventana la cual se muestra como Android en la ilustración anterior.

También puedes personalizar la vista de los archivos del proyecto para enfocarte en aspectos específicos del desarrollo de tu app. Por ejemplo, al seleccionar la vista Problems de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis.



4.3.2.- APLICACIÓN "PACHANGA"

En este apartado comentaré el código de la aplicación conforme lo vemos estructurado en el propio proyecto. Comenzando por el AndroidManifest, describiendo su función y cuáles son los elementos más importantes a tener en cuenta. A continuación, los archivos java, que se integran dentro de la app mostrando y aclarando partes del código del que se componen los diferentes archivos. Continuando con la carpeta resources, donde se presentarán los layouts de la app y se presentara elementos de las interfaces. Y como último, veremos tres archivos de la carpeta values.

4.3.2.1.- ANDROID MANIFEST

Toda aplicación debe de contener un archivo AndroidManifest en el directorio raíz. Proporciona información esencial sobre la aplicación al sistema Android, la cual el sistema necesita para poder ejecutar el código de la app.

El archivo Manifest entre otras cosas hace:

- Nombra el paquete de Java para la aplicación. El nombre del paquete sirve como un identificador único para la aplicación.
- Describe los componentes de la aplicación, como las actividades, los servicios, los receptores de mensajes y los proveedores de contenido que la integran. También nombra las clases que implementa cada uno de los componentes y publica sus capacidades, como los mensajes Intent con los que pueden funcionar. Estas declaraciones notifican al sistema Android los componentes y las condiciones para el lanzamiento.
- Determina los procesos que alojan a los componentes de la aplicación.

- Declara los permisos que debe tener la aplicación para acceder a las partes protegidas de una API e interactuar con las aplicaciones. También declara los permisos que otros deben tener para interactuar con los componentes de la aplicación.
- Enumera las clases Instrumentation que proporcionan un perfil y otra información mientras la aplicación se ejecuta. Estas declaraciones están en el manifiesto solo mientras la aplicación se desarrolla y se quitan antes de la publicación de esta.
- Declara el nivel mínimo de Android API que requiere la aplicación.
- Enumera las bibliotecas con las que debe estar vinculada la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="proyecto.fin.de.grado.carlos.villalba.v9.pachanga">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Pachanga"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        <activity android:name=".FiltrosBuscarPartido"></activity>
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="                                    " />
        <activity
            android:name=".DetallesBuscarPartido"
            android:label="Map"
            android:parentActivityName=".ListaBuscarPartido"/>
        <activity android:name=".ListaBuscarPartido" />
        <activity
            android:name=".BuscarPartido"
            android:label="Map"
            android:parentActivityName=".Principal"/>
        <activity
            android:name=".ModificarEliminar"
            android:label="Map"
            android:parentActivityName=".DetallesMisAnuncios"/>
        <activity
            android:name=".DetallesMisAnuncios"
            android:label="Map"
            android:parentActivityName=".MisAnuncios"/>
        <activity android:name=".MisAnuncios"
            android:parentActivityName=".Principal"/>
        <activity
            android:name=".SeleccionarUbicacionPubAnu"
            android:label="Map"
            android:parentActivityName=".Principal"/>
        <activity
            android:name=".PublicarAnuncio"
            android:label="Map"
            android:parentActivityName=".SeleccionarUbicacionPubAnu"/>
        <activity android:name=".Principal">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

De estas líneas de código cabe destacar:

1: `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`

Permite que la API determine una ubicación lo más precisa posible de los proveedores de ubicación disponibles, incluido el Sistema de Posicionamiento Global (GPS), así como los datos de Wifi y celdas móviles.

2:
```
android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Estas líneas conectan la aplicación con nuestro proyecto en el servidor de Google.

3: `android:parentActivityName=".Principal"/>`

Esta línea marca cual es la actividad padre de la actividad donde se encuentra esta línea. Para que, a la hora de volver, la actividad actual vuelta a su actividad padre.

4.3.2.2.- JAVA

En la carpeta Java se encuentran todos los archivos que contienen el código fuente de la aplicación. Aquí se localizan las clases que tiene asignadas un layout y que se encargan de darle funcionalidad a los elementos de las interfaces como pueden ser botones, TextViews, EditText… A su vez, también se encuentras dos carpetas entidades y utilidades las cuales se describirá su función a continuación.

4.3.2.2.1.- Principal

Este archivo .java es complementa al archivo .xml activity_principal

```
package proyecto.fin.de.grado.carlos.villalba.v9.pachanga;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Principal extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);

        ConexionSQLiteHelper conn = new ConexionSQLiteHelper( context: this, name: "bd_partidos", factory: null, version: 1);
    }

    public void onClick(View view) {
        Intent miIntent=null;
        switch (view.getId()){
            case R.id.btnPubAnuP:
                miIntent=new Intent( packageContext: Principal.this,SeleccionarUbicacionPubAnu.class);
                break;
            case R.id.btnMisAnuP:
                miIntent=new Intent( packageContext: Principal.this,MisAnuncios.class);
                break;
            case R.id.btnBusParP:
                miIntent=new Intent( packageContext: Principal.this,BuscarPartido.class);
                break;
        }
        if (miIntent!=null){
            startActivity(miIntent);
        }
    }
}
```

Como se puede observar es un código bastante corto. Esta actividad cuenta con tres botones que nos enviaran a otras actividades. Para realizar esta función, opte por crear una función onClick donde en un switch, con la variable del id de cada botón, elige el caso que cumple. Los nombres de los botones son definidos en el activity que más adelante comentare.

De este modo se puede observar que según el botón que pulsemos en la pantalla, se llevara a cabo un intent partiendo de la clase en la que nos encontramos a la clase donde queremos ir. Aclarando, que la clase SeleccionarUbicacionPubAnu es una actividad previa a la actividad PublicarAnuncio que explicare a continuación.

4.3.2.2.2- SeleccionarUbicacionPubAnu

Este archivo .java es complementa al archivo .xml activity_seleccionar_ubicacion_pub_anu.

```java
package proyecto.fin.de.grado.carlos.villalba.v9.pachanga;

import ...

public class SeleccionarUbicacionPubAnu extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    private Marker marcador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seleccionar_ubicacion_pub_anu);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync( onMapReadyCallback: this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        LatLng ali = new LatLng( v: 38.35329918545196,  v1: -0.49092373762243824);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ali,  v: 11));

        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.getUiSettings().setZoomGesturesEnabled(true);
        mMap.getUiSettings().setScrollGesturesEnabled(true);

        mMap.setOnMapClickListener((latLng) -> {
                marcador = mMap.addMarker(new MarkerOptions().position(latLng));

                new AlertDialog.Builder(SeleccionarUbicacionPubAnu.this)
                        .setTitle("Ubicacion partido")
                        .setMessage("¿Es correcta?")
                        .setPositiveButton( text: "SI", (dialog, id) -> {
                                Intent intent = new Intent( packageContext: SeleccionarUbicacionPubAnu.this, PublicarAnuncio.class);

                                intent.putExtra( name: "latitud", String.valueOf(latLng.latitude));
                                intent.putExtra( name: "longitud",String.valueOf(latLng.longitude));

                                startActivity(intent);
                        }).setNegativeButton( text: "NO", (dialog, id) -> {
                        marcador.remove();
                })
                        .show();
        });
    }
```

Esta actividad es la primera en la que utilizamos Google Maps. En esta actividad voy a explicar diferentes partes del código que me parece bueno comentar.

Lo primero que voy a comentar es la forma de centrar el mapa en la ciudad de Alicante:

```
LatLng ali = new LatLng(38.35329918545196, -0.49092373762243824);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ali, 11));
```

Como se puede observar, guardamos en variable ali la latitud y la longitud de la ciudad de Alicante, y en la siguiente línea con la opción moveCamera centramos la vista del mapa en la ubicación de Alicante pasándole como parámetro ali y dándole un zoom de 11.

A continuación, habilitamos funciones del mapa para poder movernos por el mapa y realizar zoom:

```
mMap.getUiSettings().setZoomControlsEnabled(true);

mMap.getUiSettings().setZoomGesturesEnabled(true);

mMap.getUiSettings().setScrollGesturesEnabled(true);
```

SetZoomControlsEnabled, habilita dos botones en la parte inferior derecha para acercar y alejar.

SetZoomGesturesEnabled, modifica la disponibilidad de los gestos de zoom.

SetScrollGesturesEnabled, realiza exactamente lo mismo que el método anterior, pero para los gestos de desplazamiento.

Para finalizar con esta actividad, la función setOnMapClickListener, donde al nos permite pulsar el mapa y con un AlertDialog se pregunta al usuario si la ubicación es correcta. Si la ubicación es correcta se realiza un intent para pasar a la actividad siguiente, y además con PutExtras pasaremos los parámetros de latitud y longitud para utilizarlos en la actividad PublicarAnuncio.

En caso de no ser la ubicación correcta se borrará la marca y permitirá al usuario pulsar otra ubicación.

4.3.2.2.3.- PublicarAnuncio

Este archivo .java es complementa al archivo .xml activity_publicar_anuncio. Esta clase tiene muchas cosas que comentar, por lo que directamente comentaré las partes del código interesantes.

Para comenzar, declaramos a nivel de clase:

```
private GoogleMap mMap;

Button btnSelFechaPA, btnSelHoraPA;

EditText etVacantesPA, etDescripcionPA;

TextView tvFechaPA, tvHoraPA, tvLatitudPA, tvLongitudPA;

Spinner spDeportePA;

private int diaIni, mesIni, anioIni, diaOut, mesOut, anioOut, horaIni, minutosIni, horaOut, minutosOut;
private final int DATE_ID=0;

Calendar c = Calendar.getInstance();
```

Esto se ira utilizando a lo largo del código.

Para comenzar, lo primero que realizo es asignar un array que contiene los deportes al spinner del activity:

```
spDeportePA = (Spinner) findViewById(R.id.spDeportePA);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource( context: this, R.array.deportes,android.R.layout.simple_spinner_item);
spDeportePA.setAdapter(adapter);
```

Como se observa, se necesita un array adapter para el spinner, al cual le proporcionamos la ubicación del array que queremos utilizar. También en el onCreate asignamos cada botón, EditText y TextView del activity con la función findViewById:

```
btnSelFechaPA = (Button) findViewById(R.id.btnSelFechaPA);
btnSelHoraPA = (Button) findViewById(R.id.btnSelHoraPA);

etVacantesPA = (EditText) findViewById(R.id.etVacantesPA);
etDescripcionPA = (EditText) findViewById(R.id.etDescripcionPA);

tvFechaPA = (TextView) findViewById(R.id.tvFechaPA);
tvHoraPA = (TextView) findViewById(R.id.tvHoraPA);
tvLatitudPA = (TextView) findViewById(R.id.tvLatitudPA);
tvLongitudPA = (TextView) findViewById(R.id.tvLongitudPA);
```

A continuación, inicializamos las variables fecha y hora para que a la hora de utilizar las funciones comience por el día y hora actual:

```
diaIni = c.get(Calendar.DAY_OF_MONTH);
mesIni = c.get(Calendar.MONTH);
anioIni = c.get(Calendar.YEAR);

horaIni = c.get(Calendar.HOUR_OF_DAY);
minutosIni = c.get(Calendar.MINUTE);
```

Además, recibimos los parámetros de longitud y latitud que hemos enviado desde la actividad anterior y los enviamos a los TextViews de latitud y longitud:

```
Bundle bundle = this.getIntent().getExtras();

if(bundle!= null){

    String lat = bundle.getString( key: "latitud");
    String lng = bundle.getString( key: "longitud");

    tvLatitudPA.setText(lat);
    tvLongitudPA.setText(lng);

}
```

Con la función OnClick le damos la funcionalidad a los botones:

```
public void onClick(View view) {
    Intent miIntent=null;
    switch (view.getId()){
        case R.id.btnSelFechaPA:

            showDialog(DATE_ID);
            break;

        case R.id.btnSelHoraPA:

            TimePickerDialog timePickerDialog = new TimePickerDialog( context: this, (timePicker, hourOfDay, minute) → {
                    horaOut=hourOfDay;
                    minutosOut = minute;
                tvHoraPA.setText(horaOut+":"+minutosOut);
        },horaIni,minutosIni, is24HourView: false);
        timePickerDialog.show();
            break;

        case R.id.btnPubPA:
            registrarPartido();
            break;

        case R.id.btnCanPA:
            miIntent=new Intent( packageContext: PublicarAnuncio.this,Principal.class);
            break;
    }
    if (miIntent!=null){
        startActivity(miIntent);
    }
}
```

Al pulsar Seleccionar Fecha se ejecutará el siguiente código que nos permite seleccionar y guardar la fecha:

```
private void colocar_fecha() { tvFechaPA.setText(diaOut + "/" + (mesOut) + "/" + anioOut+" "); }

private DatePickerDialog.OnDateSetListener mDateSetListener =
        (view, year, monthOfYear, dayOfMonth) → {
                anioOut = year;
                mesOut = monthOfYear+1;
                diaOut = dayOfMonth;
                colocar_fecha();
        };


@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case DATE_ID:
            return new DatePickerDialog( context: this, mDateSetListener, anioIni, mesIni, diaIni);
    }
    return null;
}
```

Guardamos cada parámetro una variable previamente declarada y, además, guardamos en un TextView la fecha para que el usuario pueda ver la fecha mientras rellena el resto de los campos.

Al pulsar Seleccionar Fecha, de modo parecido a la fecha se ejecuta una función que nos permite obtener hora que el usuario quiere poner. De igual modo se guarda la hora en un TextView para que el usuario pueda ver la hora mientras rellena los demás campos:

```
    TimePickerDialog timePickerDialog = new TimePickerDialog( context: this, (timePicker, hourOfDay, minute) → {
            horaOut=hourOfDay;
            minutosOut = minute;
        tvHoraPA.setText(horaOut+":"+minutosOut);
},horaIni,minutosIni, is24HourView: false);
timePickerDialog.show();
```

Al pulsar Registrar, se ejecutará la función registrarPartido( ):

```
private void registrarPartido() {

    ConexionSQLiteHelper conn = new ConexionSQLiteHelper( context: this, name: "bd_partidos", factory: null, version: 1);

    SQLiteDatabase db = conn.getWritableDatabase();

    String deporte = spDeportePA.getSelectedItem().toString();
    String dia = String.valueOf(diaOut);
    String mes = String.valueOf(mesOut);
    String anio = String.valueOf(anioOut);
    String hora = String.valueOf(horaOut);
    String minuto = String.valueOf(minutosOut);
    String latitud = tvLatitudPA.getText().toString();
    String longitud = tvLongitudPA.getText().toString();
    String vacantes = etVacantesPA.getText().toString();
    String descripcion = etDescripcionPA.getText().toString();

    if(!deporte.isEmpty() && !dia.isEmpty() && !mes.isEmpty() && !anio.isEmpty()
            && !hora.isEmpty() && !minuto.isEmpty() && !latitud.isEmpty() && !longitud.isEmpty() &&
            !vacantes.isEmpty() && !descripcion.isEmpty()){

        ContentValues registro = new ContentValues();
        registro.put(Utilidades.CAMPO_DEPORTE, deporte);
        registro.put(Utilidades.CAMPO_DIA, dia);
        registro.put(Utilidades.CAMPO_MES, mes);
        registro.put(Utilidades.CAMPO_ANIO, anio);
        registro.put(Utilidades.CAMPO_HORA, hora);
        registro.put(Utilidades.CAMPO_MINUTO, minuto);
        registro.put(Utilidades.CAMPO_LATITUD, latitud);
        registro.put(Utilidades.CAMPO_LONGITUD, longitud);
        registro.put(Utilidades.CAMPO_VACANTES, vacantes);
        registro.put(Utilidades.CAMPO_DESCRIPCION, descripcion);

        db.insert(Utilidades.TABLA_PARTIDO, nullColumnHack: null,registro);

        db.close();

        Toast.makeText( context: this, text: "Partido publicado",Toast.LENGTH_SHORT).show();

        Intent intent = new Intent( packageContext: PublicarAnuncio.this, Principal.class);
        startActivity(intent);
    }else{
        Toast.makeText( context: this, text: "Debes llenar todos los campos",Toast.LENGTH_SHORT).show();
    }
}
```

Lo primero que se realizar en el código es abrir la base de datos y decirle que vamos a escribir sobre ella. A continuación, declaramos y le pasamos los valores a los Strings de cada uno de los parámetros que queremos guardar en la base de datos. Este paso no sería necesario, pero lo realice así para poder comprobar que todos los campos del partido estén completados y así no se guarden partidos sin tener todos los campos completados y para eso sirve el if. Cuando cumple realiza la función y cuando no, con un toast de comunicamos al usuario que debe llenar todos los campos

Creamos un contentValues llamado “registro” donde asignamos cada variable al campo definido en la clase Utilidades e insertamos “registro” en la base de datos. Cerramos la base de datos y con un toast comunicamos al usuario que el partido ha sido registrado. Y con un intent nos trasladamos a la actividad Principal.

Y si por el contrario seleccionamos el botón Cancelar, con un intent nos trasladamos a la actividad Principal.

Por otro lado, utilizamos funciones de Google Maps para crear una marca en la posición que el usuario ha seleccionado y le permita ver que es esta la ubicación mientras rellena los demás campos:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    LatLng ubi = new LatLng(Double.parseDouble(tvLatitudPA.getText().toString()),
            Double.parseDouble(tvLongitudPA.getText().toString()));
    mMap.addMarker(new MarkerOptions().position(ubi).title("Ubicacion partido"));
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ubi, v: 14));
}
```

Donde a "ubi" le pasamos los parámetros de latitud y longitud que hemos guardado en los TextView, pero que necesitan ser convertidos a Double. Al crear la marca le asignamos un título "ubicación partido" y, además, centramos el mapa en "ubi" con un zoom de 14.

4.3.2.2.4.- MisAnuncios

Este archivo .java es complementa al archivo .xml activity_mis_anuncios. Esta clase nos permite hacer una consulta de la base de datos y nos muestra los partidos en un ListView. Lo primero que realizamos es declarar unos elementos que vamos a utilizar más adelante:

```
ListView lvListaMisAnunciosMA;

ArrayList<String> listaInformacion;
ArrayList <Partido> listaPartidos;

ConexionSQLiteHelper conn;
```

Continuando con el código, dentro de onCreate declaramos el ListView, asignamos la base de datos y llamamos a la función consultarListaPartidos( ):

```
lvListaMisAnunciosMA = (ListView) findViewById(R.id.lvListaMisAnunciosMA);

conn = new ConexionSQLiteHelper( context: this, name: "bd_partidos", factory: null, version: 1);

consultarListaPartidos();
```

Ya en la función consularListaPartidos( ), lo primero que realizamos es abrir la base de datos y decirle que vamos a leerla. A continuación, utilizamos el parámetro "listaPartidos" y le decimos que va a ser un ArrayList.

Creamos un cursor y con la sentencia While recorremos la base de datos, y obtenemos los datos y lo añadimos a "listaPartidos". Llamamos a la función obtenerLista( ), que es la encargada de darle la apariencia a cada elemento ListView. Y guardamos todos los elementos para el ListView en el parámetro "listaInformacion":

```
private void consultarListaPartidos() {
    SQLiteDatabase db = conn.getReadableDatabase();

    Partido partido = null;
    listaPartidos = new ArrayList<Partido>();

    Cursor cursor=db.rawQuery( sql: "SELECT * FROM "+ Utilidades.TABLA_PARTIDO, selectionArgs: null);

    while (cursor.moveToNext()){
        partido=new Partido();
        partido.setId(cursor.getInt( i: 0));
        partido.setDeporte(cursor.getString( i: 1));
        partido.setDia(cursor.getString( i: 2));
        partido.setMes(cursor.getString( i: 3));
        partido.setAnio(cursor.getString( i: 4));
        partido.setHora(cursor.getString( i: 5));
        partido.setMinuto(cursor.getString( i: 6));
        partido.setLatitud(cursor.getString( i: 7));
        partido.setLongitud(cursor.getString( i: 8));
        partido.setVacantes(cursor.getString( i: 9));
        partido.setDescripcion(cursor.getString( i: 10));

        listaPartidos.add(partido);
    }
    obtenerLista();
}

private void obtenerLista() {
    listaInformacion = new ArrayList<String>();

    for(int i=0;i<listaPartidos.size(); i++){
        listaInformacion.add(listaPartidos.get(i).getDeporte()+" - " +listaPartidos.get(i).getDia()
                +"/"+listaPartidos.get(i).getMes()+"/"+listaPartidos.get(i).getAnio());
    }
}
```

Y por último, en el OnCreate, asignamos “listaInformacion” al ListView a través de un adaptador. También hacemos un setOnItemClickListener para que cuando el usuario pulse sobre un anuncio este haga un intent y vaya a la actividad DetallesMisAnuncios, donde se muestran los detalles del anuncio que son pasado por un bundle a dicha actividad:

```
ArrayAdapter adaptador = new ArrayAdapter( context: this, android.R.layout.simple_list_item_1,listaInformacion);
lvListaMisAnunciosMA.setAdapter(adaptador);

lvListaMisAnunciosMA.setOnItemClickListener((adapterView, view, pos, l) → {

        Partido partido = listaPartidos.get(pos);

        Intent intent = new Intent( packageContext: MisAnuncios.this,DetallesMisAnuncios.class);

        Bundle bundle = new Bundle();
        bundle.putSerializable("partido",partido);

        intent.putExtras(bundle);
        startActivity(intent);
});
```

4.3.2.2.5.- DetallesMisAnucios

Este archivo .java es complementa al archivo .xml activity_detalles_mis_anuncios. Nos permite ver en detalle los anuncios que hemos publicado.

De esta clase, me gustaría comentar solo dos partes del código puesto que lo demás es parecido a otras clases.

Lo primero es la recepción del parámetro bundle que recibimos de la actividad anterior:

```
Bundle objetoEnviado = getIntent().getExtras();
Partido partido = null;

if(objetoEnviado!=null){
    partido = (Partido) objetoEnviado.getSerializable( key: "partido");

    tvIdDMA.setText(partido.getId().toString());
    tvDeporteDMA.setText(partido.getDeporte());
    tvDiaDMA.setText(partido.getDia());
    tvMesDMA.setText(partido.getMes());
    tvAnioDMA.setText(partido.getAnio());
    tvHoraDMA.setText(partido.getHora());
    tvMinutoDMA.setText(partido.getMinuto());
    tvLatitudDMA.setText(partido.getLatitud());
    tvLongitudDMA.setText(partido.getLongitud());
    tvVacantesDMA.setText(partido.getVacantes());
    tvDescripcionDMA.setText(partido.getDescripcion());
}
```

Podemos observar que con el if registramos si hay un cambio en la variable bundle y cuando detecta el cambio guardamos los parametros en los TextView definidos en el activity.

Por otro lado, al pulsar el botón Modificar/Eliminar se realizada un intent para ir a la actividad ModificarEliminar, que además en un putExtra le pasamos el valor de la id para utilizarlo en la consulta en dicha clase:

```
public void onClick(View view) {
    Intent miIntent=null;
    switch (view.getId()){
        case R.id.btnModEliDMA:
            Intent intent = new Intent( packageContext: DetallesMisAnuncios.this,ModificarEliminar.class);

            intent.putExtra( name: "id",tvIdDMA.getText());

            startActivity(intent);
            break;
    }
    if (miIntent!=null){
        startActivity(miIntent);
    }
}
```

4.3.2.2.6.- ModificarEliminar

Este archivo .java es complementa al archivo .xml activity_modificar_eliminar. En esta clase, podremos modificar y eliminar partidos que hemos publicado. En esta clase nos centramos las funciones modificar( ) y borrar( ) que son las encargadas de modificar y eliminar respectivamente. Ya que el resto de código es parecido a la actividad PublicarAnuncio.

Al pulsar sobre el botón Modificar, se llamará a la función modificar( ):

```
case R.id.btnModificarME:

    modificar();
    miIntent=new Intent( packageContext: ModificarEliminar.this,Principal.class);
    break;
```

Cuyo código:

```
private void modificar() {

    SQLiteDatabase db = conn.getWritableDatabase();
    String[] parametros = {tvIdME.getText().toString()};

    ContentValues values = new ContentValues();

    values.put(Utilidades.CAMPO_DIA,String.valueOf(diaOut));
    values.put(Utilidades.CAMPO_MES,String.valueOf(mesOut));
    values.put(Utilidades.CAMPO_ANIO,String.valueOf(anioOut));
    values.put(Utilidades.CAMPO_HORA,String.valueOf(horaOut));
    values.put(Utilidades.CAMPO_MINUTO,String.valueOf(minutosOut));
    values.put(Utilidades.CAMPO_VACANTES,etVacantesME.getText().toString());
    values.put(Utilidades.CAMPO_DESCRIPCION,etDescripcionME.getText().toString());

    db.update(Utilidades.TABLA_PARTIDO,values, whereClause: Utilidades.CAMPO_ID+"=?",parametros);
    Toast.makeText(getApplicationContext(), text: "Partido modificado",Toast.LENGTH_LONG).show();
    db.close();

}
```

Como podemos observar, lo primero es abrir la base de datos y decirle que vamos a escribir en ella. De forma similar al registro de partido realizamos la escritura en la base de datos y una vez realizada con un toast le comunicamos al usuario que el partido ha sido modificado y cerramos la base de datos. Con el intent volvemos a la pantalla principal.

Por otro lado, si pulsamos el botón Eliminar, llamamos a la función borrar( ):

```
case R.id.btnBorrarMe:

    borrar();
    miIntent=new Intent( packageContext: ModificarEliminar.this,Principal.class);
    break;
```

Cuyo código:

```
private void borrar() {

    SQLiteDatabase db = conn.getWritableDatabase();
    String[] parametros = {tvIdME.getText().toString()};

    db.delete(Utilidades.TABLA_PARTIDO, whereClause: Utilidades.CAMPO_ID+"=?",parametros);
    Toast.makeText(getApplicationContext(), text: "Partido eliminado",Toast.LENGTH_LONG).show();

    tvIdME.setText("");
    limpiar();

    db.close();
}
```

Abrimos la base de datos y le decimos que vamos a escribir. Pasamos al "parametros" donde se encuentra el id del partido que queremos eliminar y con db.delete lo eliminamos. Con un toast se comunica al usuario que se ha eliminado el partido y se ponemos en blanco todos los TextView con la función limpiar( ). Luego cerramos la base de datos y con un intent volvemos a la actividad Principal.

4.3.2.2.7.- BuscarPartido

Este archivo .java es complementa al archivo .xml activity_buscar_partido. En esta actividad podemos visualizar con marcas donde tenemos partidos cerca de nuestra ubicación. También nos permite centrar el mapa en nuestra ubicación por si nos vamos moviendo por él, poder volver a nuestra ubicación pulsando un botón. A continuación, explicare las partes del código más relevantes.

Los primero que realizamos es una consulta de igual modo que la realizada para el ListView:

```
private void consultarPartido() {
    SQLiteDatabase db = conn.getReadableDatabase();

    Partido partido = null;
    listaPartidos = new ArrayList<Partido>();

    Cursor cursor=db.rawQuery( sql: "SELECT * FROM "+ Utilidades.TABLA_PARTIDO, selectionArgs: null);

    while (cursor.moveToNext()){
        partido=new Partido();
        partido.setId(cursor.getInt( i: 0));
        partido.setDeporte(cursor.getString( i: 1));
        partido.setDia(cursor.getString( i: 2));
        partido.setMes(cursor.getString( i: 3));
        partido.setAnio(cursor.getString( i: 4));
        partido.setHora(cursor.getString( i: 5));
        partido.setMinuto(cursor.getString( i: 6));
        partido.setLatitud(cursor.getString( i: 7));
        partido.setLongitud(cursor.getString( i: 8));
        partido.setVacantes(cursor.getString( i: 9));
        partido.setDescripcion(cursor.getString( i: 10));

        listaPartidos.add(partido);
    }
    dibujarMarker();
}
```

Al llamar a la función dibujarMarker( ), se crearán las marcas de los partidos que se han publicado:

```
private void dibujarMarker() {

    for(int i=0;i<listaPartidos.size(); i++){
        LatLng ubi = new LatLng(Double.parseDouble(listaPartidos.get(i).getLatitud()),
                Double.parseDouble(listaPartidos.get(i).getLongitud()));
        mMap.addMarker(new MarkerOptions()
                .position(ubi)
                .title(listaPartidos.get(i).getDeporte())
                .snippet((listaPartidos.get(i).getDia()+"/"+listaPartidos.get(i).getMes()+"/"
                        +listaPartidos.get(i).getAnio())));
    }
}
```

Donde se recorre la “listaPartidos” y se van creando las marcas, con la ubicación obtenida de la base de datos y asignándole como título el deporte y añadiendo la fecha del evento.

Otra parte importarte del código los permisos para poder obtener la ubicación del usuario. Dentro de onMapReady:

```
if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
    mMap.setMyLocationEnabled(true);
} else {
    if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
            Manifest.permission.ACCESS_FINE_LOCATION)) {
        // Mostrar diálogo explicativo
    } else {
        // Solicitar permiso
        ActivityCompat.requestPermissions(
                activity: this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                LOCATION_REQUEST_CODE);
    }
}
```

Y fuera:

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == LOCATION_REQUEST_CODE) {
        // ¿Permisos asignados?
        if (permissions.length > 0 &&
                permissions[0].equals(Manifest.permission.ACCESS_FINE_LOCATION) &&
                grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mMap.setMyLocationEnabled(true);
        } else {
            Toast.makeText( context: this,  text: "Error de permisos", Toast.LENGTH_LONG).show();
        }

    }
}
```

De este modo, cuando el usuario acepte las condiciones, se habilitará la ubicación y así poder centrar el mapa en el usuario.

También con la función onClick definimos los intent de los botones Filtros y Lista:

```
public void onClick(View view) {
    Intent miIntent=null;
    switch (view.getId()){
        case R.id.btnFiltrosBP:
            miIntent=new Intent( packageContext: BuscarPartido.this,FiltrosBuscarPartido.class);
            break;
        case R.id.btnListaBP:
            miIntent=new Intent( packageContext: BuscarPartido.this,ListaBuscarPartido.class);
            break;
    }
    if (miIntent!=null){
        startActivity(miIntent);
    }
}
```

4.3.2.2.8.- FiltrosBuscarPartido

Este archivo .java es complementa al archivo .xml activity_filtros_buscar_partido. Esta actividad estaba destinada a seleccionar una serie de filtros para que, a la hora de buscar un evento, el usuario pudiera filtrar la búsqueda según diferentes parámetros. En esta actividad no hay parte nueva de código que explicar, puesto que lo único que contiene el archivo .java es el spinner. El cual se rellana de igual modo que en la clase PublicarAnuncio. Además de la configuración de los botones con una función onClick:

```
package proyecto.fin.de.grado.carlos.villalba.v9.pachanga;

import ...

public class FiltrosBuscarPartido extends AppCompatActivity {

    Spinner spDeporteFBP;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_filtros_buscar_partido);

        spDeporteFBP = (Spinner) findViewById(R.id.spDeporteFBP);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource( context: this, R.array.deportes,android.R.layout.simple_spinner_item);
        spDeporteFBP.setAdapter(adapter);
    }

    public void onClick(View view) {
        Intent miIntent=null;
        switch (view.getId()){
            case R.id.btnAplicarFBP:
                miIntent=new Intent( packageContext: FiltrosBuscarPartido.this,BuscarPartido.class);
                break;
            case R.id.btnCanFBP:
                miIntent=new Intent( packageContext: FiltrosBuscarPartido.this,BuscarPartido.class);
                break;
        }
        if (miIntent!=null){
            startActivity(miIntent);
        }
    }
}
```

4.3.2.2.9.- ListaBuscarPartido

Este archivo .java es complementa al archivo .xml activity_lista_buscar_partido. Esta actividad es exactamente igual a la clase MisAnuncios. Nos muestra en un ListView todos los partidos que hay publicados, por ello no voy a recalcar ninguna parte del código puesto que ya se ha comentado en el apartado de la clase ya mencionada.

Solo como observación, cuando se pulsa un elemento del ListView, gracias a un intent y con un bundle pasamos a la actividad DetallesBuscarPartido. Además de pasar la información del evento a dicha clase.

4.3.2.2.10.- DetallesBuscarPartido

Este archivo .java es complementa al archivo .xml activity_detalles_buscar_partido. Esta actividad es similar a la clase DetallesMisAnuncios, ya que nos muestra en detalle el partido que hemos seleccionado en la clase ListaBuscarPartido.

A diferencia que la actividad DetallesMisAnuncios, esta actividad cuenta con un botón para apuntarse a la actividad:

```
public void onClick(View view) {
    Intent miIntent=null;
    switch (view.getId()){
        case R.id.btnApuntarseDBP:

            apuntarse();
            consultar();
            if(Integer.valueOf(tvVacantesDBP.getText().toString())<=0){
                borrar();
                Toast.makeText(getApplicationContext(), text: "Te has apuntado, partido completo",Toast.LENGTH_LONG).show();
                miIntent=new Intent( packageContext: DetallesBuscarPartido.this,Principal.class);
            }else{
                Toast.makeText(getApplicationContext(), text: "Te has apuntado, faltan "
                        + tvVacantesDBP.getText().toString()+" para completar el partido",Toast.LENGTH_LONG).show();
                miIntent=new Intent( packageContext: DetallesBuscarPartido.this,Principal.class);
            }
            break;
    }
    if (miIntent!=null){
        startActivity(miIntent);
    }
}
```

Lo primero la actividad realizara al pulsar el botón Apuntarse es llamar a la función apuntarse( ):

```
private void apuntarse() {

    SQLiteDatabase db = conn.getWritableDatabase();
    String[] parametros = {tvIdDBP.getText().toString()};

    ContentValues values = new ContentValues();

    values.put(Utilidades.CAMPO_VACANTES,String.valueOf(Integer.valueOf(tvVacantesDBP.getText().toString())-1));

    db.update(Utilidades.TABLA_PARTIDO,values, whereClause: Utilidades.CAMPO_ID+"=?",parametros);
    db.close();
}
```

Lo primero que hace esta función es abrir la base de datos y decirle que vamos a escribir sobre ella. Y lo único que hacemos es variar el CAMPO_VACANTE restando una unidad. Y luego lo guardamos en la base de datos y cerramos la base.

A continuación, llamamos a la función consultar( ), que es exactamente igual que la función consultar( ) en la actividad ModificarEliminar. Esta consulta, la realizamos para cambiar el valor del número de vacantes que se muestra en el TextView. Y así, además, mediante la función if comparar el número de vacantes y realizar una acción u otra:

```
if(Integer.valueOf(tvVacantesDBP.getText().toString())<=0){
    borrar();
    Toast.makeText(getApplicationContext(), text: "Te has apuntado, partido completo",Toast.LENGTH_LONG).show();
    miIntent=new Intent( packageContext: DetallesBuscarPartido.this,Principal.class);
}else{
    Toast.makeText(getApplicationContext(), text: "Te has apuntado, faltan "
            + tvVacantesDBP.getText().toString()+" para completar el partido",Toast.LENGTH_LONG).show();
    miIntent=new Intent( packageContext: DetallesBuscarPartido.this,Principal.class);
}
```

En el caso que cumplir la condición, se llamara a la función borrar( ), que es la misma que la función borrar( ) en la actividad ModificarEliminar. Esta función eliminara el evento cuando el número de vacantes sea igual a cero, es decir, cuando el evento tenga a todos los participantes que solicitaba en el anuncio. Además, mostrara un toast con el texto “Te has apuntado, partido completo” y con un intent nos enviara a la actividad Principal.

Por otro lado, si no cumple la condición de la función if. Se mostrará con un toast que se el usuario se ha inscrito al evento y también mostrara el número de usuarios que faltan para completar el evento. De igual modo, con un intent enviara al usuario a la actividad Principal.

4.3.2.2.11.- BASE DE DATOS SQLITE

4.3.2.2.11.1.- PAQUETE ENTIDADES

Este paquete reúne las clases que necesitamos para crear la base de datos necesaria para nuestra app. Encontramos una clase llamada Partido.

4.3.2.2.11.1.1- Partido

Clase que contiene los atributos que necesitamos para guardar el partido:
- Id, deporte, día, mes, año, hora, minuto, latitud, longitud, vacantes y descripción.

```
public Partido() {
    this.id = id;
    this.deporte = deporte;
    this.dia = dia;
    this.mes = mes;
    this.anio = anio;
    this.hora = hora;
    this.minuto = minuto;
    this.latitud = latitud;
    this.longitud = longitud;
    this.vacantes = vacantes;
    this.descripcion = descripcion;
}
```

Además, también contiene los setters and getters de los atributos para poder escribir o leer los elementos de nuestra base de datos.

```
public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }
```

4.3.2.2.11.2.- PAQUETE UTILIDADES

Este paquete es creado por comodidad, para poder modificarlo más fácil si quisiésemos modificar la base de datos en algún momento. En el paquete se encuentra una clase llamada Utilidades.

4.3.2.2.11.2.1.- Utilidades

Esta clase es la encargada de definir el nombre de la tabla y de las columnas.

```
1  package proyecto.fin.de.grado.carlos.villalba.v9.pachanga.utilidades;
2
3  public class Utilidades {
4
5      public static final String TABLA_PARTIDO = "partido";
6      public static final String CAMPO_ID = "id";
7      public static final String CAMPO_DEPORTE = "deporte";
8      public static final String CAMPO_DIA = "dia";
9      public static final String CAMPO_MES = "mes";
10     public static final String CAMPO_ANIO = "año";
11     public static final String CAMPO_HORA = "hora";
12     public static final String CAMPO_MINUTO = "minuto";
13     public static final String CAMPO_LATITUD = "latitud";
14     public static final String CAMPO_LONGITUD = "longitud";
15     public static final String CAMPO_VACANTES = "vacantes";
16     public static final String CAMPO_DESCRIPCION = "descripcion";
17
18     public static final String CREAR_TABLA_PARTIDO="CREATE TABLE " +
19             ""+TABLA_PARTIDO+" ("+CAMPO_ID+" INTEGER PRIMARY KEY AUTOINCREMENT, "
20             +CAMPO_DEPORTE+" TEXT,"+CAMPO_DIA+" TEXT,"+CAMPO_MES+" TEXT,"
21             +CAMPO_ANIO+" TEXT," +CAMPO_HORA+" TEXT,"+CAMPO_MINUTO+" TEXT,"
22             +CAMPO_LATITUD+" TEXT,"+CAMPO_LONGITUD+" TEXT,"+CAMPO_VACANTES +" TEXT,"
23             +CAMPO_DESCRIPCION+" TEXT)";
24 }
25
```

En la imagen del código se puede observar cómo se crear cada campo (columna) de la base de datos y luego en un String generamos la sentencia para crear la tabla que utilizamos en la clase ConexionSQLiteHelper para crear la base de datos.

4.3.2.2.11.3.- ConexionSQLiteHelper

Clase encargada de crear y actualizar la base de datos.

```
package proyecto.fin.de.grado.carlos.villalba.v9.pachanga;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import proyecto.fin.de.grado.carlos.villalba.v9.pachanga.utilidades.Utilidades;

public class ConexionSQLiteHelper extends SQLiteOpenHelper {
    public ConexionSQLiteHelper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        db.execSQL(Utilidades.CREAR_TABLA_PARTIDO);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int il) {

        db.execSQL("DROP TABLE IF EXISTS "+Utilidades.TABLA_PARTIDO);
        onCreate(db);

    }
}
```

Esta clase extiende SQLiteOpenHelper como podemos ver en el código. Es la clase encargada de crear la base de datos:

```
@Override
public void onCreate(SQLiteDatabase db) {

    db.execSQL(Utilidades.CREAR_TABLA_PARTIDO);

}
```

Y de igual modo, nos da la opción de actualizar la base de datos como se muestra a continuación:

```
@Override
public void onUpgrade(SQLiteDatabase db, int i, int i1) {

    db.execSQL("DROP TABLE IF EXISTS "+Utilidades.TABLA_PARTIDO);
    onCreate(db);

}
```

Donde se puede observar, que introducimos una secuencia para comprobar si existe la TABLA_PARTIDO.

4.3.2.3.- LAYOUT

Es la parte de nuestro programa donde se encuentran todas las interfaces de la aplicación. Es lo que el usuario va a ver a la hora de utilizar la aplicación por lo que debe de ser lo más intuitiva posible a la vez que funcional.

En la realización de los diferentes archivos que corresponden al layout para la realización de nuestra app he utilizado una serie de objetos los cuales voy a describir a continuación, para poder entender la programación de estos archivos:

- RelativeLayout: es un grupo de vistas que muestra vistas secundarias en posiciones relativas. La posición de cada vista puede especificarse como relativa a elementos

equivalentes (como a la izquierda o por debajo de otra vista) o en posiciones relativas al área RelativeLayout superior (como alineada a la parte inferior, izquierda o central).

- TextView: elemento de interfaz de usuario que muestra texto al usuario.

- Button: elemento de interfaz de usuario que el usuario puede tocar o hacer clic para realizar una acción.

- Fragment: representa un comportamiento o una parte de la interfaz de usuario en un FragmentActivity. Puedes combinar varios fragmentos en una sola actividad para crear un IU multipanel y volver a usar un fragmento en diferentes actividades. Se puede ver como una sección modular de una actividad que tiene un ciclo de vida propio, que recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando.

- ScrollView: permite desplazar los elementos de la vista colocada dentro de él.

- LinearLayout: diseño que organiza otras vistas, ya sea horizontalmente en una sola columna o verticalmente en una sola fila.

- Spinner: desplegable que muestra diferentes opciones que el usuario debe escoger. Estas opciones provienen de un adapter que este asociado al spinner.

- EditText: tiene dos funciones. Mostrar texto al usuario y a diferencia del TextView, el usuario puede acceder al EditText y escribir en él.

- ListView: muestra una lista de elementos que pueden ser introducidos por el usuario o predefinidos en la programación del archivo .java.

También les explicare una serie de características que tienen estos objetos y que he utilizado en la aplicación para dimensionar cada elemento.

- Width: define la anchura que ocupara el elemento en la pantalla. Esta característica es obligatoria definirla.

- Height: define la altura que ocupara el elemento en la pantalla. Esta característica es obligatoria definirla.

- Weigth: nos permite darles pesos a los elementos. Por ejemplo, si queremos que un elemento ocupe una cuarta parte y otro tres cuartas partes en el LinearLayout o RelativeLayout le otorgamos un peso de cuatro y a los elementos respectivamente pesos de 1 y 3.

- Orientation: define la orientación hacia donde se crean los elementos uno detrás de otros. Es aconsejable definirlo en los LinearLayout puesto que tenemos que decir si es una columna o una fila.

- Background: nos permite cambiar el fondo de la pantalla o cambiar los colores de los botones.

- Text: esta función nos permite introducir texto de forma predeterminada.

- Hint: sirve para los EditText. Nos permite introducir texto de forma predeterminada, pero podemos escribir encima de él. Sirve para dar pista de que tenemos que escribir en ese EditText.

- TextSize: se utiliza para dar el tamaño a la letra de los textos.

- Id: función necesaria si queremos utilizar el elemento en los archivos .java. Dan nombre al elemento para más tarde poder buscarlo.

- OnClick: es un atributo para los botones. Sirve para que al pulsarlo el botón reaccione y haga la función para la cual está programado.

- InputType: cambia el tipo de teclado según el texto que vayamos a introducir en los EditText. Por ejemplo: numero, entero, correo, normal,…

- Margin: nos permite definir márgenes entre elementos.

Para la organización de los elementos en un RelativeLayout, para poder posicionar los elementos de la forma que yo he querido he utilizado:

- CenterHorizontal: centra el elemento en el centro horizontal de pantalla. Cuando es true es que se centra, por el contrario, si es false no se centra.

- marginTop: te deja marcar un margen del elemento a la parte superior de la pantalla.

- CenterInParent: centra de forma vertical y horizontal el elemento. Cuando es true es que se centra, por el contrario, si es false no se centra.

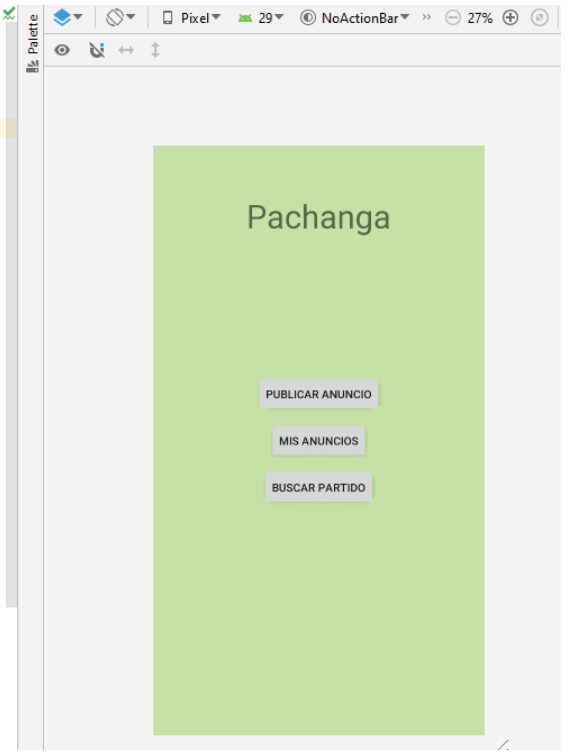
- Above: posiciona al elemento en la parte superior al elemento que le hemos dicho.

- Below: posiciona al elemento en la parte inferior al elemento que le hemos dicho.

Una vez ya definidos todos estos elementos procedo a mostrar cada archivo del layout de forma individualizada.

4.3.2.3.1.- activity_principal

```
<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#C5E1A5">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pachanga"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="60dp"
        android:textSize="40sp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Mis anuncios"
        android:layout_centerInParent="true"
        android:id="@+id/btnMisAnuP"
        android:layout_margin="10dp"
        android:onClick="onClick"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Publicar anuncio"
        android:layout_centerHorizontal="true"
        android:layout_above="@+id/btnMisAnuP"
        android:id="@+id/btnPubAnuP"
        android:onClick="onClick"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Buscar partido"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/btnMisAnuP"
        android:id="@+id/btnBusParP"
        android:onClick="onClick"/>

</RelativeLayout>
```



4.3.2.3.2.- activity_seleccionar_ubicación_pub_anu

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SeleccionarUbicacionPubAnu" />
```

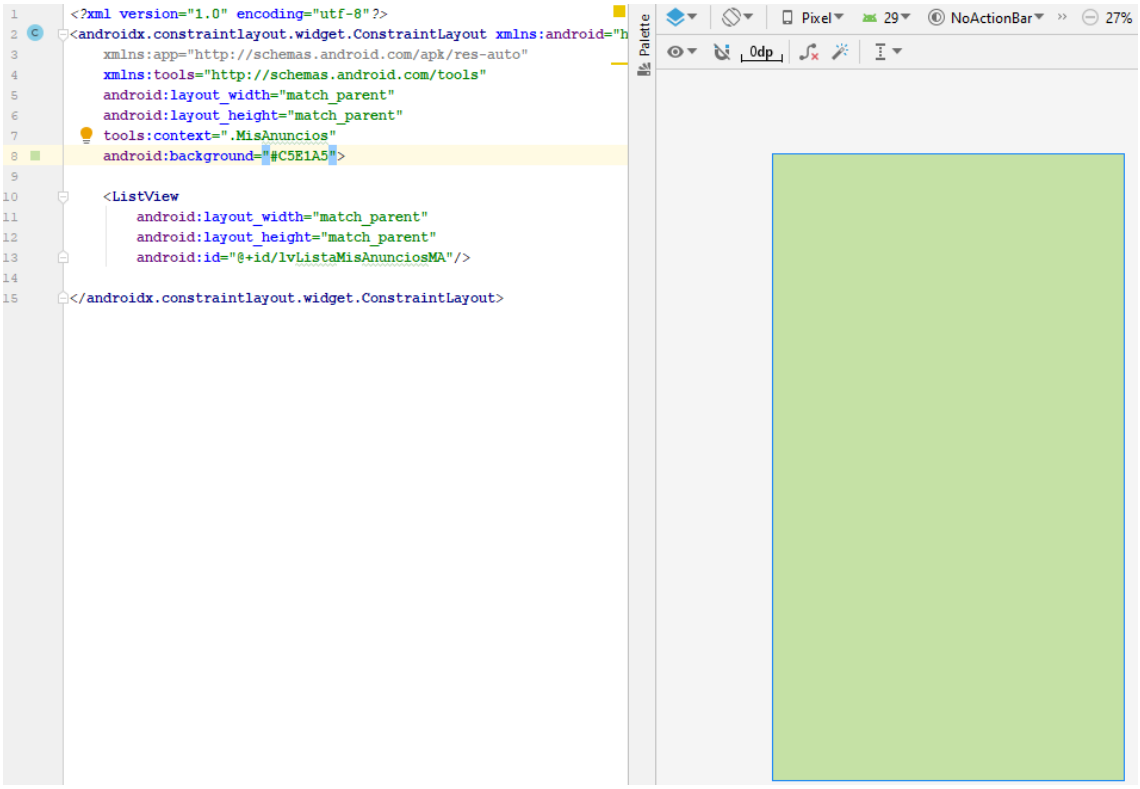
Este fragmento muestra el mapa de Google Maps.

4.3.2.3.3.- activity_publicar_anuncio

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#C5E1A5">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Deporte"
            android:layout_margin="10dp"/>

        <Spinner
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/spDeportePA"
            android:layout_margin="10dp">
        </Spinner>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Fecha"
            android:layout_margin="10dp"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="Seleccionar fecha"
            android:id="@+id/btnSelFechaPA"
            android:onClick="onClick"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text=""
            android:textSize="18sp"
            android:id="@+id/tvFechaPA"
            android:layout_margin="10dp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hora"
            android:layout_margin="10dp"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="Seleccionar hora"
            android:id="@+id/btnSelHoraPA"
            android:onClick="onClick"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text=""
            android:textSize="18sp"
            android:id="@+id/tvHoraPA"
            android:layout_margin="10dp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Lugar"
            android:layout_margin="10dp"/>
```

```
        <fragment
            xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:tools="http://schemas.android.com/tools"
            android:id="@+id/map"
            android:name="com.google.android.gms.maps.SupportMapFragment"
            android:layout_width="match_parent"
            android:layout_height="300dp"
            tools:context=".PublicarAnuncio"
            android:layout_margin="10dp"/>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text=""
                android:id="@+id/tvLatitudPA"
                android:layout_margin="10dp"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="y"
                android:layout_margin="10dp"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text=""
                android:id="@+id/tvLongitudPA"
                android:layout_margin="10dp"/>
        </LinearLayout>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Vacantes"
            android:layout_margin="10dp"/>

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Numero"
            android:id="@+id/etVacantesPA"
            android:layout_margin="10dp"
            android:inputType="number"
            android:maxLength="2"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Descripción"
            android:layout_margin="10dp"/>

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/etDescripcionPA"
            android:layout_margin="10dp"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_weight="2">
            <Button
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:layout_margin="10dp"
                android:text="Publicar"
                android:id="@+id/btnPubPA"
                android:onClick="onClick"/>

            <Button
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:layout_margin="10dp"
                android:text="Cancelar"
                android:id="@+id/btnCanPA"
                android:onClick="onClick"/>

        </LinearLayout>
    </LinearLayout>
</ScrollView>
```

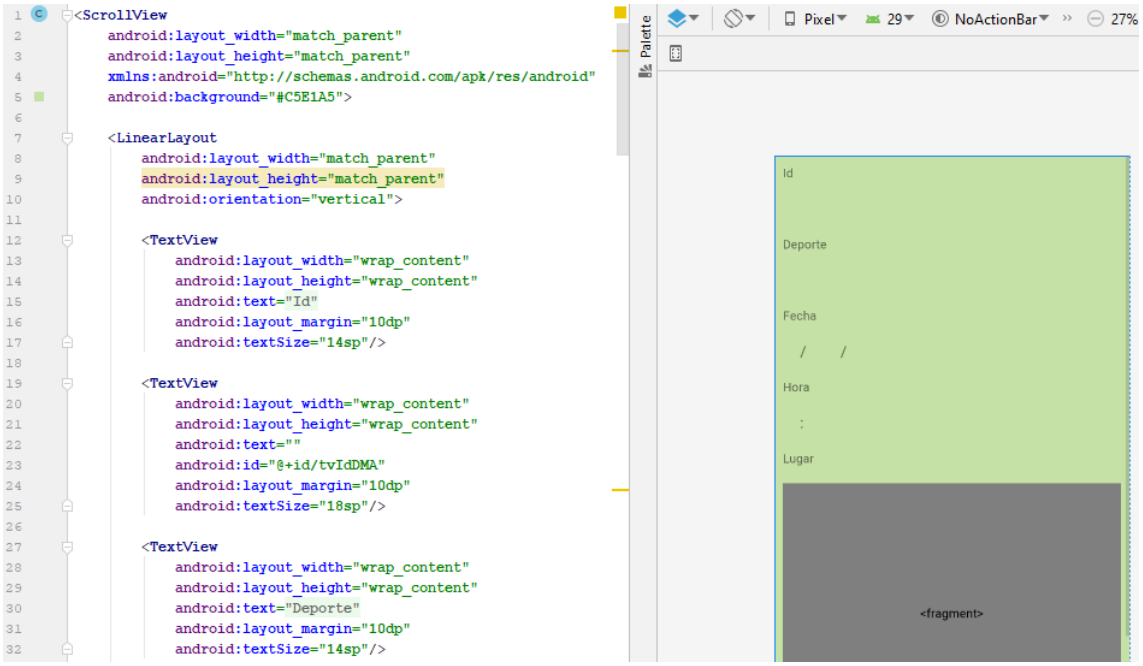
4.3.2.3.4.- activity_mis_anuncios



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="h
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MisAnuncios"
    android:background="#C5E1A5">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/lvListaMisAnunciosMA"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Aunque no se aprecia en la pantalla, puesto que no hay elementos, se crea un ListView donde irán apareciendo los eventos que vayamos publicando.

4.3.2.3.5.- activity_detalles_mis_anuncios



```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#C5E1A5">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Id"
            android:layout_margin="10dp"
            android:textSize="14sp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:id="@+id/tvIdDMA"
            android:layout_margin="10dp"
            android:textSize="18sp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Deporte"
            android:layout_margin="10dp"
            android:textSize="14sp"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:id="@+id/tvDeporteDMA"
    android:layout_margin="10dp"
    android:textSize="18sp"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Fecha"
    android:layout_margin="10dp"
    android:textSize="14sp"/>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvDiaDMA"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="/"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvMesDMA"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="/"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvAnioDMA"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hora"
    android:layout_margin="10dp"
    android:textSize="14sp"/>
```

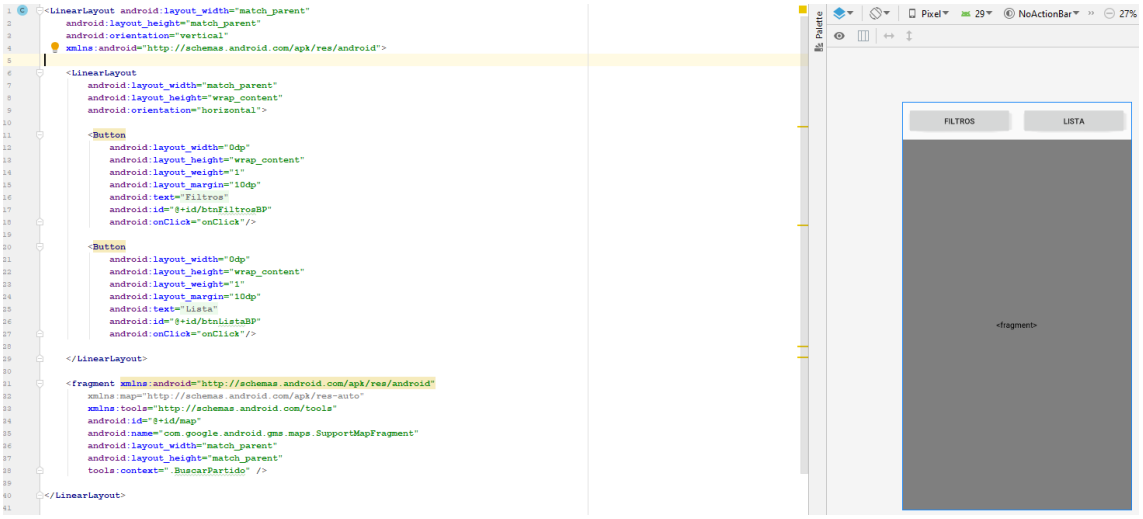
```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvHoraDMA"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=":"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvMinutoDMA"
        android:layout_margin="10dp"
        android:textSize="18sp"/>

</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Lugar"
    android:layout_margin="10dp"
    android:textSize="14sp"/>

<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    tools:context=".PublicarAnuncio"
    android:layout_margin="10dp"/>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvLatitudDMA"
        android:layout_margin="10dp"
        android:textSize="14sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="y"
        android:layout_margin="10dp"
        android:textSize="14sp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/tvLongitudDMA"
        android:layout_margin="10dp"
        android:textSize="14sp"/>
```

```
175
176         </LinearLayout>
177
178         <TextView
179             android:layout_width="wrap_content"
180             android:layout_height="wrap_content"
181             android:text="Vacantes"
182             android:layout_margin="10dp"
183             android:textSize="14sp"/>
184
185         <TextView
186             android:layout_width="wrap_content"
187             android:layout_height="wrap_content"
188             android:text=""
189             android:id="@+id/tvVacantesDMA"
190             android:layout_margin="10dp"
191             android:textSize="18sp"/>
192
193         <TextView
194             android:layout_width="wrap_content"
195             android:layout_height="wrap_content"
196             android:text="Descripción"
197             android:layout_margin="10dp"
198             android:textSize="14sp"/>
199
200         <TextView
201             android:layout_width="match_parent"
202             android:layout_height="wrap_content"
203             android:text=""
204             android:id="@+id/tvDescripcionDMA"
205             android:layout_margin="10dp"
206             android:textSize="18sp"/>
207
208         <LinearLayout
209             android:layout_width="match_parent"
210             android:layout_height="wrap_content"
211             android:orientation="horizontal"
212             android:layout_weight="2">
213
214             <Button
215                 android:layout_width="0dp"
216                 android:layout_height="wrap_content"
217                 android:layout_weight="1"
218                 android:layout_margin="10dp"
219                 android:text="Modificar/Eliminar"
220                 android:id="@+id/btnModEliDMA"
221                 android:onClick="onClick"/>
222
223         </LinearLayout>
224     </LinearLayout>
225 </ScrollView>
```

4.3.2.3.6.- activity_modificar_eliminar

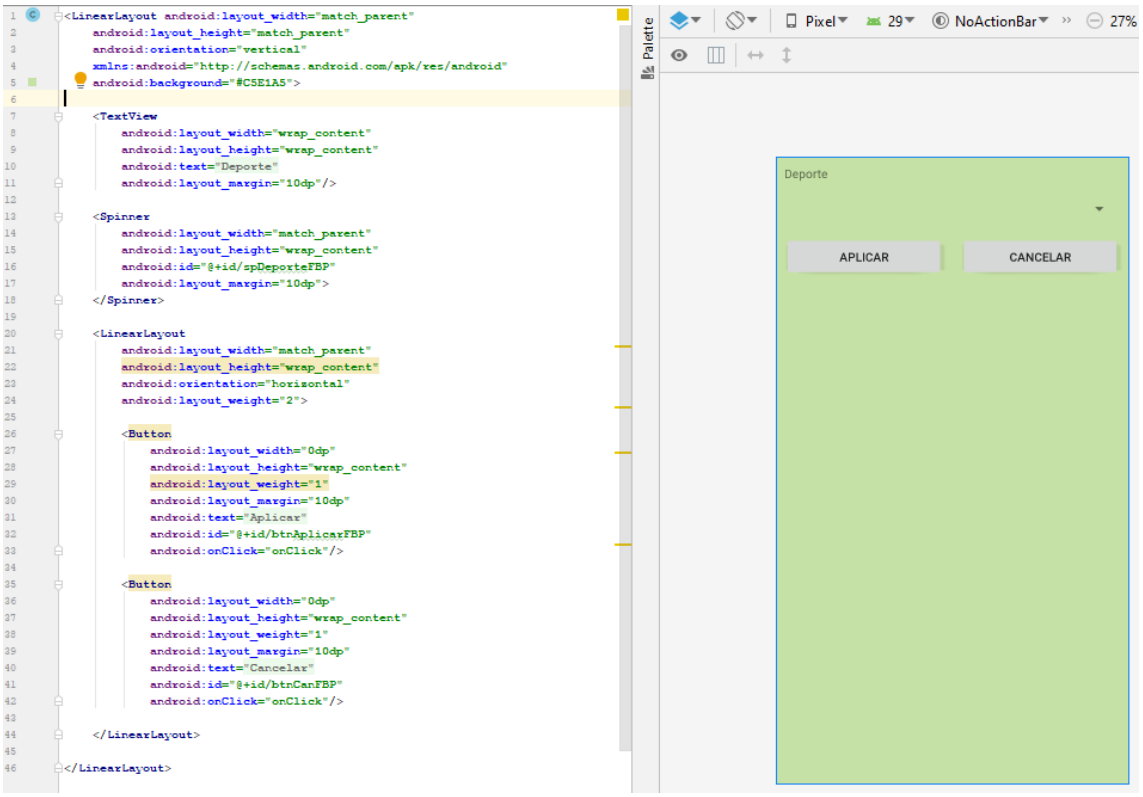
Esta activity tiene los mismos elementos que el activity_publicar_anuncio.

4.3.2.3.7.- activity_buscar_partido



El fragment es un mapa de Google Maps donde visualizamos las marcas de los eventos que hemos publicado.

4.3.2.3.8.- activity_filtros_buscar_partido



4.3.2.3.9.- activity_lista_buscar_partido

Este activity es un ListView por lo que es igual que el activity_mis_anuncios.

4.3.2.3.10.- activity_detalles_buscar_partido

Tiene la misma forma y los mismos elementos que el activity_detalles_mis_anuncios. Solo cambiando los nombres de los elementos y cambiando el botón que en vez de ser "Modificar/Eliminar" es "Apuntarse".

4.3.2.4.- VALUES

En este apartado, voy a mostrar tres archivos que hemos utilizado en la carpeta values:

- arraydeportes

Este archivo contiene los elementos que forman los spinner que de las actividades donde se muestran los diferentes deportes a escoger. El código de este archivo:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="deportes">

        <item>-</item>
        <item>Futbol</item>
        <item>Futbol sala</item>
        <item>Baloncesto</item>
        <item>Voleibol</item>
        <item>Tenis</item>
        <item>Hockey</item>
        <item>Golf</item>
        <item>Balonmano</item>
        <item>Ciclismo</item>
        <item>Running</item>
        <item>Rugby</item>
        <item>Baseball</item>
        <item>Badminton</item>
        <item>Tenis de mesa</item>
        <item>Fútbol americano</item>

    </string-array>
</resources>
```

Donde cada elemento que queremos mostrar debe estar en <ítem> y </ítem>. Comenzando por un guion para que la de forma inicial el spinner no muestre ninguna opción.

- google_maps_api

En este archivo es donde introducimos la clave de API de nuestro proyecto en Google para sincronizarlo con nuestra app y aparezca el mapa de Google Maps.

- strings

En este archivo es donde creamos todos los strings que luego utilizaremos en las diferentes interfaces y partes de código. Este documento no da mayor facilidad a hora de

si queremos cambiar algún nombre ya que no tenemos que ir cambiando en varios sitios si hemos utilizado el mismo string en diferentes sitios. El código es el siguiente:

```
<resources>
    <string name="app_name">Pachanga</string>
    <string name="MisAnu">Mis anuncios</string>
    <string name="PonerAnu">Publicar anuncio</string>
    <string name="BusPar">Buscar partido</string>
    <string name="Deporte">Deporte</string>
    <string name="Fecha">Fecha</string>
    <string name="SelecFecha">Seleccionar fecha</string>
    <string name="Hora">Hora</string>
    <string name="SelecHora">Seleccionar hora</string>
    <string name="Lugar">Lugar</string>
    <string name="Y">y</string>
    <string name="Vacantes">Vacantes</string>
    <string name="Numero">Numero</string>
    <string name="Descripcion">Descripción</string>
    <string name="Publicar">Publicar</string>
    <string name="Cancelar">Cancelar</string>
    <string name="Id">Id</string>
    <string name="barra">/</string>
    <string name="Dospuntos">:</string>
    <string name="ModEli">Modificar/Eliminar</string>
    <string name="CambUbi">Cambiar ubicacion</string>
    <string name="Modificar">Modificar</string>
    <string name="Borrar">Borrar</string>
    <string name="Cargar">Cargar</string>
    <string name="Filtros">Filtros</string>
    <string name="Lista">Lista</string>
    <string name="Apuntarse">Apuntarse</string>
    <string name="Aplicar">Aplicar</string>
    <string name="title_activity_publicar_anuncio">Map</string>
    <string name="title_activity_seleccionar_ubicacion_pub_anu">Map</string>
    <string name="title_activity_detalles_mis_anuncios">Map</string>
    <string name="title_activity_modificar_eliminar">Map</string>
    <string name="title_activity_seleccinar_ubicacion_mod_eli">Map</string>
    <string name="title_activity_buscar_partido">Map</string>
    <string name="title_activity_detalles_buscar_partido">Map</string>
</resources>
```

5.- RESULTADOS Y DISCUSIÓN

Comencé con este proyecto con la idea de aprender un nuevo lenguaje de programación y aprender como se realizan las aplicaciones móviles que todos tenemos instaladas en nuestros smartphones. Y he de decir que he aprendido mucho durante estos meses que he dedicado al estudio y desarrollo de aplicaciones Android.

Con la ayuda de mi tutor y varios portales de internet he conseguido hacer uso de diferentes herramientas que del lenguaje. En esta aplicación he hecho uso de una base de datos en SQLite, una base de datos de forma local. También he empleado diferentes objetos para la interfaz como pueden ser spinner, Scrollview, ListView,… Aunque en el proyecto final no lo he utilizado, también utilice ListView personalizados que son realmente útiles a para cualquier aplicación móvil.

También la utilización de las librerías TimePicker y DatePicker , que sirven para introducir la hora y la fecha respectivamente a través de un fragment que aparece en la pantalla ya programado. Ya que al principio este proceso lo hacia de forma manual y podía llevar a errores, pues los usuarios podrían introducir fecha u hora erróneas.

Este proyecto me ha dado la capacidad que plantearme problemas que quizás con otro tipo de lenguajes de programación no se dan, ya que programando en la aplicación Android Studio debes tener en cuenta una parte de interfaz y que debes de mostrar de forma clara para el usuario y que sea fácil de usar. Pero, además, debes de programar el funcionamiento de dicha interfaz de manera que su funcionamiento sea el deseado.

En cuanto a la aplicación “Pachanga”, no se ha realizado con fines comerciales. Lo que he presentado en el presente proyecto es un prototipo inspirada en otra aplicación similar y que tiene varios puntos que podrían mejorarse de los cuales hablare en las propuestas de mejora.

En la realización de la app además de la utilización de los elementos ya mencionado también he tenido que trabajar con la API del servidor de Google Maps. Para así, poder visualizar los mapas y poder marcar ubicaciones y mostrar donde hay eventos cerca de nosotros.

Puedo decir que este proyecto, me ha ayudado a la compresión de las aplicaciones móviles. He aprendido cosas básicas para realizar una aplicación y otras mas avanzadas que han creado una base para poder desarrollar mi potencial en esta materia y que realmente me ha parecido muy interesante y amplia. He aprendido además lenguaje Java, un lenguaje de programación que no había usado a lo largo del grado pero que quería aprender. Y realmente me ha parecido fácil de aprender ya que considero que el grado, aunque nos enseña lenguaje C, sirve como base para aprender otros tipos de lenguajes de programación.

6.- CONCLUSIONES

Como conclusión que gustaría mencionar la satisfacción que siento al haber realizado este proyecto. Cuando comencé la aplicación se me hizo un mundo, pero una vez vas avanzando y aprendiendo te va gustado cada vez más.

He aprendido muchas cosas de programación como ya he mencionado, pero también he aprendido a solucionarme problemas solo sin necesitar ayuda de los demás, apoyándome en foros, tutoriales… que quizás sin sentir la necesidad de solucionarlo hubiese esperado a solucionarlos con el profesor.

La aplicación parece sencilla, pero tiene su miga. Ya que cuenta una gran cantidad de actividades y de diversos elementos de programación. Pero he de decir que me siento muy satisfecho con el trabajo realizado en el presente proyecto.

7.- PROPUESTAS DE MEJORA

Como propuestas de mejora de la aplicación que gustaría mencionar cinco mejores que me vienen a la cabeza:

- Presentar una interfaz más bonita, poner fondos a las pantallas y quizás presentar algún tipo de introducción a la hora de abrir la aplicación.

- Realizar un servidor online en el que puedas registrar diferentes usuarios en diferentes dispositivos así poder utilizar la aplicación de forma online y entre diferentes dispositivos.

- Implementar filtros a la hora de buscar partidos para poder buscar solo los eventos que realmente nos interesen. Por ejemplo, filtrar por deporte, por distancia o fecha.

- Diferenciar los marcados en los mapas según el deporte, es decir, que cada marcador tenga una imagen que vaya ligada al deporte.

- A la hora de pulsar sobre un marcador, nos de la opción de acceder a la actividad de "DetallesBuscarPartido", para poder apuntarnos y no tener que ir a la lista para hacerlo.

Estas serían algunas de las mejoras que me gustaría implementar en el presente proyecto. Algunas son fáciles de implementar, pero otras habría que cambiar partes de la aplicación para poderlas llevar a cabo.

BIBLIOGRAFIA

Es este apartado expondré las referencias sobre las publicaciones utilizadas en el presente proyecto:

Academia Android:
https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

Wikipedia, Android Studio:
https://es.wikipedia.org/wiki/Android_Studio

Wikipedia, Cuota de mercado Android:
https://es.wikipedia.org/wiki/Android#Cuota_de_mercado_2

Wikipedia, Google Maps:
https://es.wikipedia.org/wiki/Google_Maps

Wikipedia, Sistema coordenadas:
https://es.wikipedia.org/wiki/Sistema_de_coordenadas_universal_transversal_de_Mercator

Wikipedia, Versiones Android:
https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

Blogthinkbig:
https://blogthinkbig.com/las-mejores-apps-para-hacer-deporte-en-grupo

Hermosa programación:
http://www.hermosaprogramacion.com/2016/05/google-maps-android-api-v2/#markers

Developer Android:
http://www.hermosaprogramacion.com/2016/05/google-maps-android-api-v2/#markers

API con Google Maps:
https://console.developers.google.com/apis/

Stackoverflow:
https://es.stackoverflow.com/