DOCUMENTACIÓN TFG

APP DE ESTADÍSTICAS DE BALONCESTO

Miguel Pavón Limones

9 DE JUNIO 2025

1.Requisitos y hardware utilizado.

Yo en mi caso he utilizado Windows 10, el IDE utilizado sería **Android Studio,** la base de datos utilizada es Firebase y los componentes que lleva mi ordenador son:

- 8GB de ram ddr4
- Disco duro 128GB ssd
- Disco duro 1000GB hdd
- Procesador Intel core i5-8256U

1.1 Requisitos mínimos para utilizar Android Studio.

Toda la información que se encuentra en la tabla está sacada de esta página de internet:
https://developer.android.com/studio/install?hl=es-419

REQUISITOS PARA WINDOWS:

REQUISITOS	MÍNIMO	RECOMENDADO
SO	Microsoft Windows 10 de 64 bits	La versión más reciente de Windows de 64 bits
RAM	Studio: 8GB Studio y emulador: 16GB	32GB
CPU:	Se recomienda un procesador Intel i5 de 8ª generación o AMD Ryzen 1ª gen en adelante, con soporte para virtualización (VT-x o AMD-V) habilitada en la BIOS.	Se recomienda una CPU moderna con virtualización habilitada (Intel VT-x o AMD-V), preferiblemente Intel Core i5/i7/i9 con sufijos H/HK/HX (laptops) o S/F/K (escritorio), o AMD Ryzen 5/6/7/9; se desaconsejan las series Intel N y U por bajo rendimiento.
ESPACIO EN DISCO DURO	Estudio: 8 GB de espacio libre Studio y emulador: 16 GB de espacio libre	Unidad de estado sólido con 32 GB o más
Resolución de pantalla	1280X800	1920X1080

GPU	Studio: Ninguno Studio y emulador: GPU con 4 GB de VRAM, como Nvidia GeForce 10 o versiones posteriores, o AMD Radeon RX 5000 o versiones posteriores con los controladores más recientes	GPU con 8 GB de VRAM, como Nvidia Geforce 20 series o posterior, o AMD Radeon RX6600 o posterior con los controladores más recientes

REQUISITOS PARA MAC:

REQUISITO	MÍNIMO	RECOMENDADO
SO	macOS12	La versión más reciente de macOS de 64 bits
RAM	Studio: 8 GB Studio y emulador: 16 GB	32 GB
CPU	Chip Apple M1 o Intel Core de 6ª generación o posterior Por ejemplo, MacBook Pro 2016 con procesador i7-4770HQ o superior. La compatibilidad con Mac con chips Intel dejará de estar disponible	Apple Silicon más reciente
ESPACIO EN EL DISCO	Studio: 8 GB de espacio libre Studio y Emulator: 16 GB de espacio libre	Unidad de estado sólido con 32 GB o más de espacio libre
RESOLUCIÓN DE PANTALLA	1280x800	1920x1080
GPU	Integrado	Integrado

REQUISITOS PARA LINUX:

REQUISITOS	MÍNIMO	RECOMENDADO
SO	Cualquier distribución de Linux de 64 bits que sea compatible con Gnome, KDE o Unity DE; GNU C Library (glibc) 2.31 o versiones posteriores	Versión más reciente de Linux de 64 bits
RAM	Studio: 8 GB Studio y emulador: 16 GB	32 GB de RAM o más
CPU	Se requiere compatibilidad con la virtualización (Intel VT-x o AMD-V, habilitada en el BIOS). Microarquitectura de CPU después de 2017 Intel Core de 8ª gen. i5 / AMD Zen Ryzen (p. ej., Intel i5-8xxx, Ryzen 1xxx).	Se requiere compatibilidad con la virtualización (Intel VT-x o AMD-V, habilitado en el BIOS). Microarquitectura de CPU más reciente. Busca CPUs de las series Intel Core i5, i7 o i9, o los sufijos H/HK/HX para laptops o los sufijos S/F/K para computadoras de escritorio, o las series AMD Ryzen 5, 6, 7 o 9. Ten en cuenta que no se recomiendan los procesadores Intel Core serie N y serie U debido a que su rendimiento es insuficiente.
Espacio en el disco	Studio: 8 GB de espacio libre Studio y Emulator: 16 GB de espacio libre	Unidad de estado sólido con 32 GB o más
Resolución de pantalla	1280X800	1920X1080
GPU	Studio: Ninguno Studio y emulador: GPU con 4 GB de VRAM, como Nvidia GeForce serie 10 o posterior, o AMD Radeon RX 5000 o posterior con los controladores más recientes	GPU con 8 GB de VRAM, como Nvidia GeForce 20 Series o una posterior, o AMD Radeon RX 6600 o una posterior con los controladores más recientes

2. Instalación del entorno de desarrollo.

A continuación vamos a explicar como instalar Android Studio en los diferentes Sistemas Operativos.

2.1 Instalación en Windows.

INSTALACIÓN Y CONFIGURACIÓN DE ANDROID STUDIO:

- Descargamos el instalador desde el siguiente enlace: developer.android.com/studio
- Ejecutamos el archivo .exe y seguimos los pasos del asistente(Next, aceptamos componentes, hacemos click en Install y luego Finish al finalizar).
- Seleccionamos **“Do not import settings”**.
- Esperamos a que se descargue el SDK y las herramientas necesarias.

2.2 Instalación en Mac.

INSTALACIÓN y CONFIGURACIÓN DE ANDROID STUDIO:

- Descargamos el instalador desde el siguiente enlace: developer.android.com/studio
- Abrimos el archivo .dmg y arrastramos Android Studio a la carpeta Aplicaciones.
- Abrimos Android Studio desde Aplicaciones(si es necesario autoriza Seguridad y privacidad).
- Elegimos **“Do not import settings”**, luego elegimos instalación Standard, elegimos tema, y dejamos que se descargue el SDK.

2.3 Instalación en Linux.

INSTALACIÓN Y CONFIGURACIÓN DE ANDROID STUDIO:

- Descargamos el archivo desde developer.android.com/studio
- Extraemos la carpeta: **tar -xvzf android-studio-*.tar.gz**
- Y la movemos con el comando: **sudo mv android-studio /opt/la-ruta-que-quieras**
- Ejecutamos el instalador que se encuentra en la carpeta que acabamos de mover: **/opt/ruta que tu quieras/android-studio/bin/studio.sh**
- (Esto es opcional) Creamos acceso desde **Tools> Create Desktop Entry**.

3. Explicación del funcionamiento de mi aplicación.

3.1 Herramientas utilizadas.

- **Firebase:** Esta es la base de datos que he utilizado, es una base de datos no relacional y la verdad que para la idea de aplicación que tenía es la que mejor se adecuaba a mis necesidades, además es muy sencilla de usar. Solo necesitas crear la base de datos a través de su página en internet, descargarte el .json e introducirlo en tu proyecto, y por último hacer la implementación.
- **Firebase Authentication:** Es un servicio que ofrece la propia base de datos, la cual se encarga de hacer la autenticación del usuario, ya sea para el registro o para iniciar sesión de manera sencilla, rápida y segura, con esto no hace falta que el programador tenga que construir todo el sistema de autenticación desde el principio. Además ya trae algunas restricciones como por ejemplo que el usuario no puede introducir un email sin el "@" puesto.

3.1.1.Características de Firebase Authentication.

Permite autenticar usuarios mediante:

- Correo electrónico y contraseña.
- Proveedores externos.
- Inicio de sesión anónimos.
- Inicio de sesión con número de teléfono.

3.2 Explicación Funcionamiento De La Aplicación.

Mi aplicación funciona de la siguiente forma:
- Lo primero es que el usuario se tiene que dar de alta en la aplicación para poder utilizarla, si no, no podría utilizarla. Si tiene un usuario ya dado de alta, podrá iniciar sesión dándole a la frase que tiene debajo del botón de registrar, que le redirigirá a una pantalla para iniciar sesión.

-Una vez que se da de alta, le saldrá una pantalla la cual tiene que registrar a su equipo y una vez que haya registrado al equipo, podrá crear un partido. En caso de no registrar un equipo, no podrá crear partidos. Si el usuario ha iniciado sesión, le saldrá tanto el equipo como los partidos que haya creado anteriormente.

-Una vez que ha registrado al equipo, ya podrá crear un partido. Para crear el partido tendrá que meter los siguientes datos: nombre del equipo rival, fecha, tipo de partido(liga, torneo, amistoso, etc), en caso de que sea liga deberá rellenar que jornada es, y por último si juega de local o visitante.

-Una vez que haya rellenado los datos ya tendrá el partido creado. Cuando entra al partido, le saldrá un botón que pondrá empezar partido, cuando le de se le desbloquearan todos los botones que se verán en la pantalla. En esta aplicación podrás coger todo tipo de estadísticas de baloncesto, como puede ser tiros libres anotados/intentados, tiro de 2 punto anotados/intentados, triples anotados/intentados y tiros de campo metidos/intentados en total, todas estas estadísticas con su respectivo porcentaje, y luego todas las demás estadísticas de equipo como son rebotes(en general), rebotes defensivos, rebotes ofensivos, asistencias, faltas de equipo, pérdidas, tapones, robos.

Una vez que le das a iniciar partido, te saldrá la opción de acabar partido, y justamente arriba te pondrá **Exportar PDF**, **Ver Gráfica.** Durante cualquier momento, tanto del partido como una vez acabado, podrás dar a cualquiera de esos 2 botones, los cuales siempre funcionaran para cuando quieran ver las estadísticas.

Si le das a **Exportar PDF,** se te descargara un archivo pdf en la carpeta Descargas del dispositivo en el que esté la aplicación.

Si le damos al botón de **Ver Gráfica**, se verán 2 gráficas, una para los tiros realizados tanto de triple, tiro de 2 puntos y tiros libres, y debajo había otra gráfica con las demás estadísticas.

Una vez acabado el partido, los botones para guardar estadísticas estarán desactivados y no se podrá guardar más datos de ese partido.

4. Explicación de Código.

La explicación será por pantalla, primero se explicara que hace cada pantalla Poner todo el código de cada pantalla es difícil ya que hay pantallas que tiene muchas líneas de código, por eso se explicara que hace cada pantalla, y cuando se termine, se explicara el layout utilizado en cada pantalla sin su código.

4.1 Parte lógica de la aplicación.

4.1.1 Pantalla de registro.

En esta pantalla lo que se va a hacer es que el usuario se pueda registrar en la aplicación, deberá introducir su nombre de usuario, un correo electrónico y una contraseña de mínimo 6 caracteres.Una vez que le dé al botón de registrar, el usuario se registrará en la BD y automáticamente le cambiará de pantalla.

En el código nos encontramos con la siguiente función, que es la parte más importante de esta pantalla:

```
btnRegistrar.setOnClickListener {

   val nombre = etNombre.text.toString().trim()

   val email = etEmail.text.toString().trim()

   val password = etContraseña.text.toString().trim()



   if (nombre.isEmpty() || email.isEmpty() || password.isEmpty()) {

       mostrarAlerta("Campos incompletos", "Por favor, completa todos los campos.")

       return@setOnClickListener

   }



   if (password.length < 6) {

       mostrarAlerta("Contraseña inválida", "La contraseña debe tener al menos 6 caracteres.")

       return@setOnClickListener

   }
```

```
   btnRegistrar.isEnabled = false



   auth.createUserWithEmailAndPassword(email, password)

       .addOnCompleteListener { task ->

          if (task.isSuccessful) {

               val userId = auth.currentUser?.uid

              val datosUsuario = mapOf(

                   "nombre" to nombre,

                   "email" to email

              )



              val db =
FirebaseDatabase.getInstance("https://miguelpavonlimones-tfg-default-rtdb.europe-west1.firebas
edatabase.app/")

              db.getReference("usuarios").child(userId!!)

                   .setValue(datosUsuario)

                   .addOnSuccessListener {

                      mostrarAlertaConAccion(

                           "Registro exitoso",

                           "El usuario se ha registrado correctamente."

                       ) {

                          val intent = Intent(this, pantalla2::class.java)

                           startActivity(intent)

                           finish()

                       }

                  }

                  .addOnFailureListener {

                      btnRegistrar.isEnabled = true

                       mostrarAlerta("Error en Firebase", "No se pudieron guardar los datos:
${it.message}")
```

```
                    }

                } else {

                    btnRegistrar.isEnabled = true

                    mostrarAlerta("Error de autenticación", task.exception?.message ?: "Error
desconocido.")

                }

            }

}
```

El código empieza recogiendo los datos que el usuario escribe en un formulario: nombre, correo, contraseña. Antes de hacer nada, se comprueba que no haya ningún campo vacío y a continuación se comprueba que la contraseña tenga al menos 6 caracteres, ya que Firebase lo exige. Si todo está correcto, el botón se desactiva para evitar que haya registros múltiples.

Luego se usa la función **auth.creatUserWithEmailAndPassword**, que crea un nuevo usuario en Firebase usando el email y la contraseña introducidos. Aunque la contraseña se usa para crear el usuario, **no se guarda ni se muestra** en la base de datos por motivos de seguridad. Solo se guardan el nombre y el email.

Si todo sale bien, se mostrará un mensaje de éxito y redirigirá al usuario a la siguiente pantalla. En caso de hubiese un error (por ejemplo, email ya está en uso) se muestra una alerta indicando que ha pasado.

4.1.2 Pantalla iniciar sesión.

El código empieza recogiendo los datos que el usuario mete en el formulario, en este caso serían email y contraseña. Una vez recogidos se comprueba que los campos no estén vacíos.

Una vez hecha esta comprobación, se usa la función **auth.signInWithEmailAndPassword(email, password)** que su función es iniciar sesión con las cuentas que hay registradas en Firebase, si los datos están bien, se redirigirá a la pantalla principal de la aplicación, en caso de que haya algún fallo, saltaría un mensaje con el error en concreto.

A esta pantalla se puede llegar de 2 maneras diferentes:

- la primera es desde la pantalla de registro, si el usuario tiene ya cuenta.
- desde la pantalla principal de la aplicación, si el usuario ha cerrado sesión o ha eliminado su cuenta, se redirigirá a la pantalla de registro y de ahí podrá entrar a la pantalla de iniciar sesión.

4.1.3 Pantalla2 (pantalla principal).

Este código representa la pantalla principal de la aplicación, donde se muestran los partidos creados por el usuario y se gestionan las opciones como cerrar sesión, eliminar usuario, borrar equipo o crear un nuevo partido. Como he utilizado un recyclerview más abajo se explicará que se hace en cada archivo.

Nada más abrir la pantalla se comprueba de que haya un usuario logueado, en caso de que no lo haya, se le redirige solo a la pantalla de registro.

A continuación, se intenta recuperar el nombre del usuario desde la base de datos y se muestra en el botón superior derecho. También se busca que el usuario tenga un equipo registrado, en caso de que lo tenga se muestra su nombre en el botón superior izquierdo y se desbloquea el botón de crear partido. Si el usuario no tiene un equipo registrado, se oculta el botón y tiene que registrar un equipo.

En la parte central hay una lista con los partidos del equipo. Si clicamos en un partido, se abrirá otra pantalla la cual el código se explicará más adelante. También tenemos la opción de borrar el partido, con el icono de basura que hay a la derecha.

En el menú del usuario encontramos 2 opciones:

- Borrar usuario: se elimina el usuario de la base de datos y se le redirige automáticamente a la pantalla de registro.
- Cerrar sesión: se cierra la sesión del usuario y se le redirige a la pantalla de registro, una vez que el usuario inicie sesión, se cargará el equipo y los partidos que tenga ese usuario creados.

En el menú de equipo encontramos una sola opción:

- Eliminar equipo: se borrar el equipo del usuario de la base de datos, al hacer esto, el botón de crear partido se oculta y debe registrar un equipo nuevo antes de crear un partido.

4.1.3.1 SplashActivity.

```
class SplashActivity : AppCompatActivity() {


   private lateinit var auth: FirebaseAuth


   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)


       auth = FirebaseAuth.getInstance()
       val currentUser = auth.currentUser


       if (currentUser != null) {
           startActivity(Intent(this, pantalla2::class.java))
       } else {
           startActivity(Intent(this, MainActivity::class.java))
       }


       finish()
   }
}
```

Este archivo lo que hace es comprobar si hay un usuario con la sesión abierta cuando accedemos a la pantalla principal, en caso de que haya un usuario se queda en pantalla2 (pantalla principal) y si no hay un usuario con la sesión abierta se le redirige a la pantalla de registro.

4.1.3.2 Clase Partido

```
data class Partido(

   var fecha: String = "",

   var rival: String = "",

   var tipo: String = "",

   var nombreEquipo: String? = null,

   var usuarioId: String = "",

   var local: Boolean = true,

   var jornada: String? = null,

   @Transient var id:String? = null

)
```

Aquí lo que hago es crear las variables las cuales después se verán en la tarjeta del partido dentro del recyclerview. @Transient sirve para que un campo no pueda guardarse en la base de datos ni en archivos de almacenamiento. En mi código significa que el campo “id” no se guardará en Firebase cuando subas el objeto Partido. Solo se usa dentro de la app para tener una referencia temporal.

4.1.3.3 PartidoAdapter

Aquí lo que hago es crear un adapter para mostrar los partidos en un recyclerview. Se define como se van a ver las tarjetas de cada partido usando un layout(item_partido), y se rellenan los datos como el nombre del equipo, el rival, tipo de partido, fecha, si es local o visitante y la jornada(si es un partido de liga). Además también se gestionan 2 acciones como son:

- cuando le das a la imagen de la papelera se borra el partido.
- el cambio de pantalla al seleccionar el partido.

4.1.4 Registrar equipo.

Esta pantalla permite al usuario registrar un equipo, el cual quedará asociado a su cuenta.

Lo primero es definir y enlazar los elementos del formulario: un campo de texto donde se ingresa el nombre del equipo, un menú desplegable con las categorías disponibles y un botón para guardar.

Cuando el usuario pulse el botón de guardar pasara lo siguiente:

- Se comprobará que el nombre del equipo no esté vacío.
- Se comprobará que el usuario esté logueado.
- Se creará un objeto en la base de datos con el nombre del equipo, categoría y el id del usuario.
- Y se guardará en la parte de equipos en la base de datos.

Si se guarda correctamente, se mostrará un mensaje indicando el éxito del registro y se redirigirá al usuario a la pantalla principal.

En caso de que haya un fallo, saltará un mensaje indicando el fallo que hay.

4.1.5 Registrar partido.

Se van a cargar todos los componentes del formulario: fecha, rival, tipo de partido, jornada(si se aplica), local o visitante y el botón de guardar.

Cuando el usuario en tipo de partido elige la opción liga, se desbloquea un campo de texto para ingresar el número de la jornada. Si el usuario elige otro campo, la opción de ingresar la jornada queda oculta.

Para seleccionar la fecha, se selecciona con un DatePicker, lo que va a facilitar al usuario elegir una fecha sin tener que escribirla a mano.

Cuando pulsemos el botón de guardar, pasara lo siguiente:

- Se comprobará que la fecha y el nombre del rival estén completos.
- Se recogen todos los datos del formulario, incluido si el equipo juega como local o visitante.
- Si todo está bien, se creará un objeto en la base de datos llamado (partidos) con los datos.

Una vez que se le da al botón de guardar, se mostrará un mensaje de confirmación y se redirige a la pantalla principal. En caso de que haya algún fallo, saltará un mensaje con el fallo que haya.

4.1.6 Pantalla de estadísticas.

Esta pantalla sirve para llevar el registro de las estadísticas de un partido. Se llega a esta pantalla a través de la pantalla principal, creando el partido y luego clicando en él.

Al iniciar la pantalla se recogen los datos básicos del partido (nombre del equipo, rival, jornada y fecha) a través de un intent. Después se conecta a firebase para ver si el partido fue finalizado y también para cargar las estadísticas actuales del equipo si existen.

Para registrar la estadísticas hay varios botones:

- Botón de tiro metido: te da la opción de elegir el tipo de tiro que es como: triples, tiros de 2, tiros libres.
- Botón de tiro fallado: te da la opción de elegir el tipo de tiro que es como: triples, tiros de 2, tiros libres.
- Botón de rebotes: te da la opción de rebote defensivo u ofensivo.
- Y hay 1 botón para cada estadística restante.

Mientras que el partido esté iniciado, al pulsar un botón se irán actualizando las estadísticas en Firebase, para que todo quede guardado a tiempo real.

Los botones al principio estarán bloqueados hasta que se le den a “Empezar partido” y cuando se le de a “Finalizar partido” no se podrán coger más estadísticas de ese partido.

Hay 2 botones que se le podrá dar siempre que son “Exportar PDF” y “Ver Gráfica”.

Si le damos al botón “Exportar PDF” se nos descargara un archivo en pdf con todas las estadísticas del partido. En esas estadísticas encontraremos:

- Todos los tiros y sus porcentajes.
- Total de puntos metidos.
- Rebotes y otras estadísticas.

Si le damos al botón “Ver Gráficas”, nos mandara a una pantalla la cual habrán 2 gráficas, una para los tiros y otra para las demás estadísticas.

4.1.7 Gráficas.

El objetivo de esta pantalla es conseguir que el usuario vaya viendo de forma fácil y rápida las estadísticas durante el partido.

Al iniciarse la pantalla, se recogen el ID del partido y el nombre del equipo desde un intent. Luego se conecta a Firebase para leer los datos de estadísticas guardados. Se usarán 2 gráficas para visualizar todas las estadísticas.

Primera Gráfica.

Gráfica de tiro.

En la primera gráfica se van a representar tantos los tiros metidos/intentados como los porcentajes de:

- tiros libres
- tiros de 2
- triples

Segunda Gráfica.

Gráfica de estadística.

Esta gráfica va a mostrar estadísticas generales como:

- Rebotes totales, rebotes defensivos/ofensivos.
- Asistencia.
- Robos.
- Pérdidas.
- Tapones.
- Faltas.

Todos los datos de ambas gráficas se irán actualizando a medida de que se van metiendo datos.

4.1.7.1 BarChartCanvasView.

Este componente sirve para dibujar gráficos de barras personalizados dentro de la app. Se utiliza tanto para mostrar estadísticas de tiro como otras estadísticas del partido en la pantalla de gráficas.

Este componente hereda de View, lo que significa que se dibuja directamente sobre el lienzo.Se adapta al tamaño del dispositivo y pinta las barras una a una según los datos que se le pasan.

El método setDatos(...) se usa para enviarle los datos al gráfico. Este método recibe:

- Un mapa con los nombres de cada estadística y su valor (Map<String, Int>)
- Un booleano que indica si los valores son porcentajes (true) o cantidades (false)
- Y opcionalmente un mapa de detalles para mostrar (por ejemplo 5/12)

Cuando recibe los datos, se redibuja automáticamente gracias al método postInvalidate().

entro de onDraw(...), se calcula:

- Cuántas barras debe dibujar
- El ancho y altura de cada barra
- Su posición en la pantalla

Luego, para cada estadística:

- Se dibuja una línea horizontal como guía cada 10 unidades
- Se dibuja la barra vertical, con un color específico según el tipo de dato
- Se escribe el nombre de la estadística debajo
- Los detalles (como 5/12), se muestran encima de la barra

Cada tipo de dato tiene un color diferente, que se obtiene usando ContextCompat.getColor(...) y hace referencia a un color definido en res/values/colors.xml como por ejemplo:

@color/colorTiro2, @color/colorRobos, etc.

4.2 Layouts.

Aquí se va a explicar el layout utilizado en cada pantalla.

4.2.1. Pantalla de registro.

Aquí nos encontramos con una foto de background, utilizaremos un cardview y dentro del cardview un linearlayout para que este todo centrado, dentro del layout nos encontraremos el formulario de registro. Justamente arriba y abajo del cardview utilizaremos 2 textview, uno pondrá "**Pantalla de registro**" y el otro pondrá "**¿Ya tienes una cuenta? Inicia sesión**" y servirá para redirigir a la pantalla de inicio sesión.

4.2.2. Pantalla de inicio de sesión.

La pantalla de iniciar sesión es igual que la pantalla de registro, la única diferencia es que solamente hay 1 textview, aparte el cardview, el textview y el botón de iniciar sesión van dentro del linearlayout.

4.2.3. Pantalla principal.

En esta pantalla tendremos 2 card views iguales, con un linearlayout cada uno, en uno irán los Materials Button con el nombre del usuario y el nombre del equipo del usuario, además si clickamos en cualquiera de los 2 botones se desplegará un menú hecho con opciones para cada botón. En el otro cardview irán 2 botones normales, 1 botón para registrar equipo y otro para registrar partidos.

Habrá también un recyclerview para mostrar todos los partidos que se vayan creando.

El color de fondo de esta pantalla (background) será un color gris-azulado.

4.2.3.1 Pantalla de item_estadisticas(recyclerview).

Se crea un Material cardview para el partido,en la columna izquierda habrá varios textview para mostrar datos del partido y a la derecha en la otra columna habrá un icono de una basura.

4.2.4. Pantalla registrar equipo.

El color de fondo de la pantalla es un azul claro. Creamos un textview que irá arriba del todo en el que pondrá “Registrar equipo”. A continuación irá un scrollview, el cual está puesto por si hay pantallas que sean pequeñas o si el teclado está abierto se puedan meter los datos bien. Dentro de ese scroll view, tendremos un linearlayout y dentro del linearlayout tendremos un editText para meter el nombre del equipo rival, un spinner con un menú con las categorías disponibles y por último un botón para guardar los datos.

4.2.5. Pantalla registrar partido.

El color de fondo de la pantalla es un azul claro. En este caso la pantalla será un scrollview por el mismo motivo que en la pantalla anterior, dentro del scrollview habrá un linearlayout y dentro de ese linearlayout tendremos 1 textview con la frase “Registrar partido”, tendremos 3 editText para recoger los datos de fecha del partido, nombre del equipo rival, y otro que sería el número de la jornada(este saldría en caso de que hiciese falta), además tendremos un spinner para seleccionar el tipo de partido y un radiobutton para elegir si el partido se juega de local o visitante. Por último tendríamos el botón de guardar.

4.2.6. Pantalla de estadísticas.

El color de fondo es un azul claro. En la parte de superior hay un linearlayout el cual va a pintar los datos del partido, que estarán divididos en izquierda y derecha:

- nombre del equipo
- nombre del rival
- puntos totales del equipo
- fecha
- tipo de partido
- jornada

Debajo de ese linear layout se ve una imagen de una cancha de baloncesto. Y debajo de esa imagen hay un linear layout con 2 botones que son para “Exportar PDF” y para “Ver grafica”.

Después aparecerá un botón el cual pondrá “Empezar partido”, una vez que se le dé a ese botón, ese botón desaparecerá y se activaran todos los botones de abajo, además aparecerá otro botón que ponga “Acabar partido”. Debajo justo hay un scrollview para que se pueda ver bien en pantalla pequeñas, el cual va a tener 4 linear layouts distintos:

- Primero: Botones de tiros metidos y tiros fallados.
- Segundo: Botones de rebotes y asistencias.

- Tercero: robos y tapones.
- Cuarto: faltas personales totales y pérdidas.

Una vez que se le dé al botón de “Acabar partido”, esos 4 linearlayouts más el botón de “Acabar partido” se desactivaran y no se podrán coger más estadísticas, los únicos que funcionan en todo momento son los botones de “Exportar PDF” y “Ver Gráfica”.

4.2.7 Pantalla de gráficas.

Esta es la última pantalla de mi aplicación, creamos un scrollview ya que vamos a meter 2 graficas y no habría espacio suficiente si no se mete scrollview. Dentro del scrollview vamos a meter un linearlayout el cual tendrá un primer textview que pondrá “Gráfica de Tiro” y a continuación haremos una llamada a mi clase BarChartCanvasViews para que pinte la gráfica de tiro. Después de esa gráfica, metemos un View para dejar espacio entre una gráfica y el textView de la siguiente gráfica, el textView de la siguiente gráfica pondrá “Otras Estadísticas” y a continuación se volverá a hacer una llamada a la clase BarChartCanvasViews para que pinte la gráfica de las estadísticas.