

MUNDIAL QATAR 2022

MI ÁLBUM

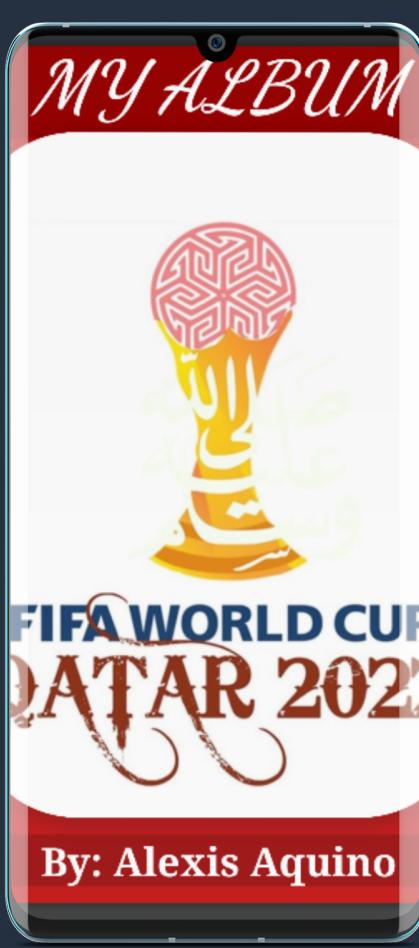
INTRODUCCIÓN

La creación de un Álbum Virtual tiene como resultado una aplicación móvil desde la cual, muchos fanáticos podrán acceder a ello desde el celular. De esta manera, se facilita a los clientes la opción de visualizar las figuras que tienen disponibles o consultar todos los jugadores disponibles junto a sus países y la parte fundamental realizar compra de tokens para conseguir jugadores de manera aleatoria y almacenarlo sin salir de casa.

FUNCIONES

La aplicación tiene tres funciones principales de las cuales la principal sería la obtención de ficha de jugadores de manera aleatoria, la visualización de todos los jugadores disponibles para su obtención y una lista en donde se encuentran almacenados todos los jugadores que hemos obtenido de manera aleatoria.

INICIO APP



PLANTEAMIENTO

Con la realización del mundial de fútbol de Qatar 2022 he optado en realizar un álbum referido a ello, con todos los jugadores más relevantes con sus países en dicho evento de fútbol en donde su principal funcionalidad es la obtención de figuras de jugadores de manera aleatoria para posteriormente colecciónarlos en nuestro álbum virtual.

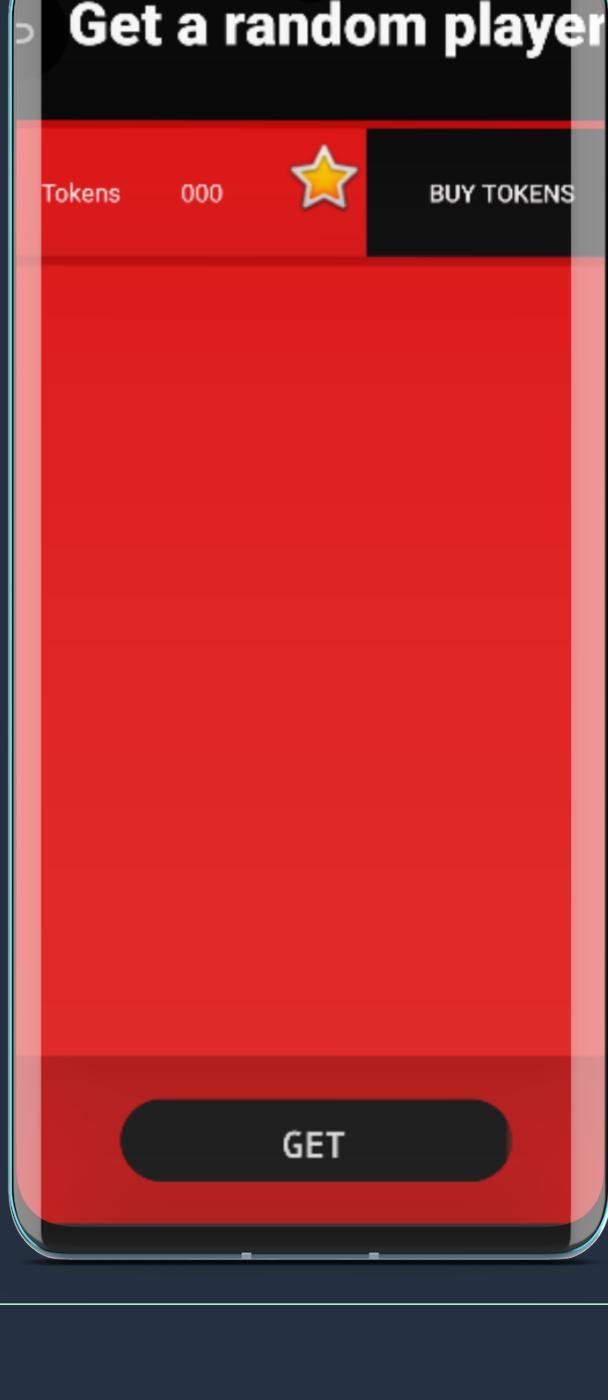
TECNOLOGÍA

La tecnología utilizada para la creación de esta aplicación es el lenguaje de programación java con el editor oficial de android que es el Android Studio junto a sus elementos de la paleta de diseños en los cuales resalta la utilización del RecyclerView para visualizar datos y la utilización del SQLite para el almacenamiento de datos de toda nuestra aplicación la cual ya viene integrada con el Android Studio

FUNCTION PRINCIPAL

GET PLAYER

INTERFAZ



CÓDIGO JAVA

```
Random r = new Random();
```

```
int value = r.nextInt(listaJugadores.size());
```

```
Jugador obtenido = listaJugadores.get(value);
```

```
if (listaDeMisJugadores.contains(obtenido)) {
```

```
Log.i(" -Log-", "Ya existe .");
```

```
}else{
```

```
listaDeMisJugadores.add(obtenido);
```

```
}
```

FUNCIONALIDAD

Para obtener fichas de jugadores de forma aleatoria usamos la clase Random de

generación de números aleatorios. Esta función

generará un número al azar que tiene asignado a cada

ficha de jugador.

podemos generar números aleatorios dentro de un rango

específico y de esta manera

obtener a cada ficha de

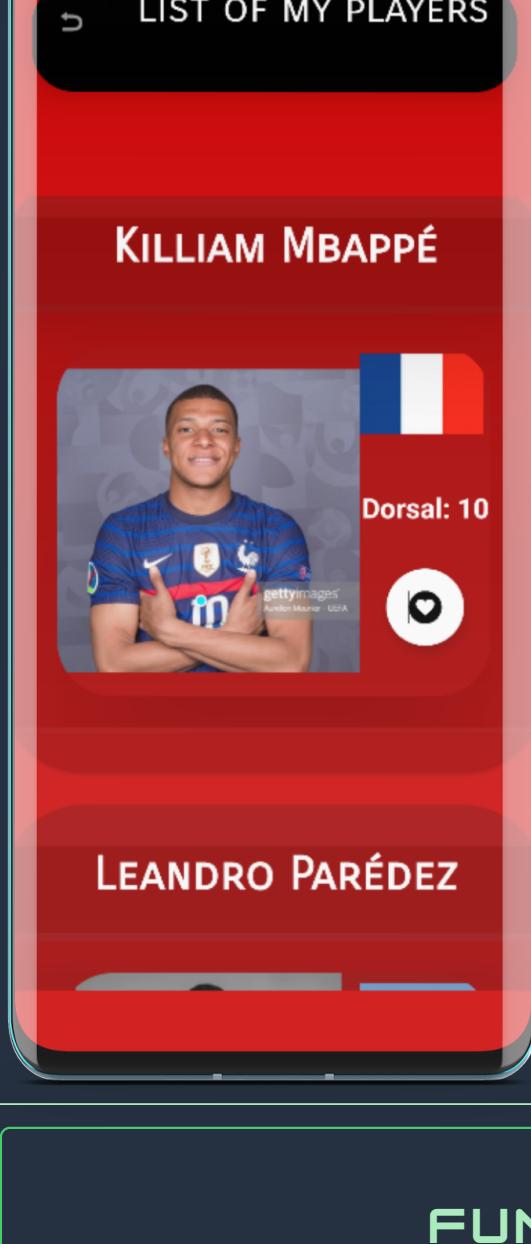
jugador de manera aleatoria.

CLASE RANDOM

La clase "Random" de Java es una herramienta que genera números aleatorios utilizando diferentes algoritmos de generación de números pseudoaleatorios. Los métodos de la clase permiten generar enteros, números en coma flotante y booleanos aleatorios, entre otros.

MY PLAYERS

INTERFAZ



CÓDIGO JAVA

```
recyclerView.setLayoutManager(  
    new LinearLayoutManager(this));  
//bbdd {contiene el list player}  
BBDD bd = new BBDD(this);  
//Adapter {recibe list player}  
MisJugadoresAdapter adapter =  
    new MisJugadoresAdapter(bd.getMyPlayers(), this);
```

```
recyclerView.setAdapter(adapter);
```

FUNCIONALIDAD

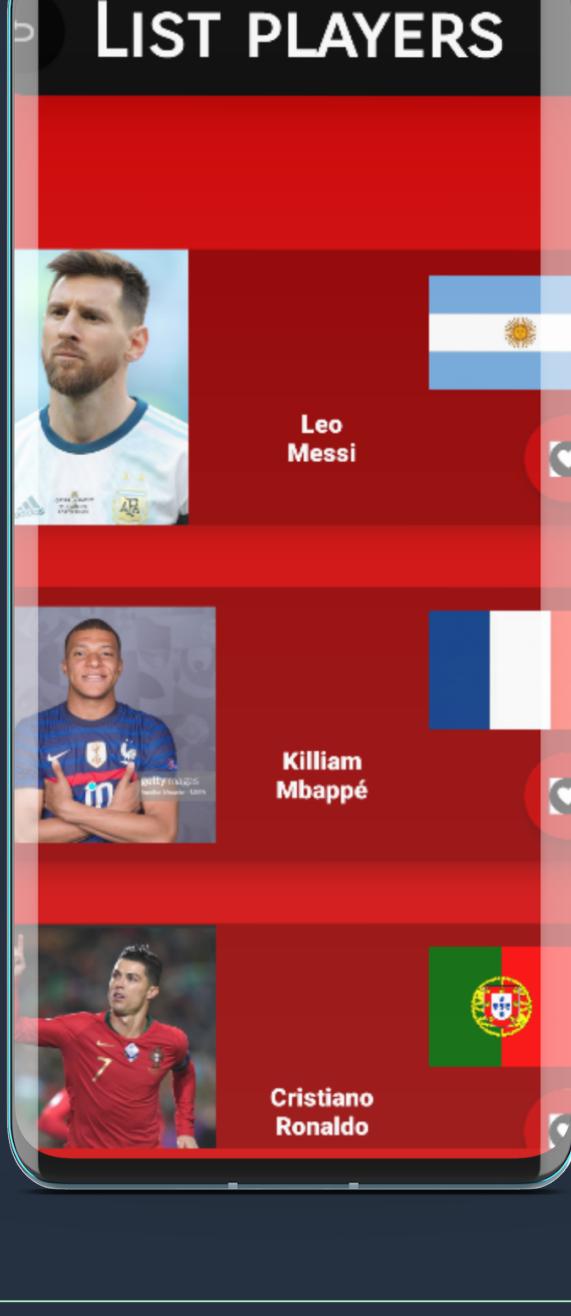
Visualización de fichas de los jugadores que se ha obtenido de manera aleatoria, gracias a los widgets que nos brinda el Android Studio Los datos se pasan a RecyclerView a través de un adaptador, que es responsable de crear vistas para cada elemento de datos y vincularlas a los datos correspondientes. Se visualiza la imagen, el nombre, apellido, dorsal y país del jugador junto a un botón like-dislike

```
holder.btnAdd.setOnClickListener(  
    new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            if (ale.isLike()) {  
                holder.btnAdd.setImageResource(R.drawable.dislike);  
                ale.setLike(false);  
            } else {  
                holder.btnAdd.setImageResource(R.drawable.like);  
                ale.setLike(true);  
            }  
        }  
    });
```

Un botón que permite poner "like" o "dislike" se utiliza en la aplicación donde los usuarios pueden expresar su opinión sobre el jugador tanto como si le gusta o no, en donde cada modo tiene un logo y es verificado con booleans, si es true significa 'me gusta' y false el otro caso.

LIST ALL PLAYERS

INTERFAZ



CÓDIGO JAVA

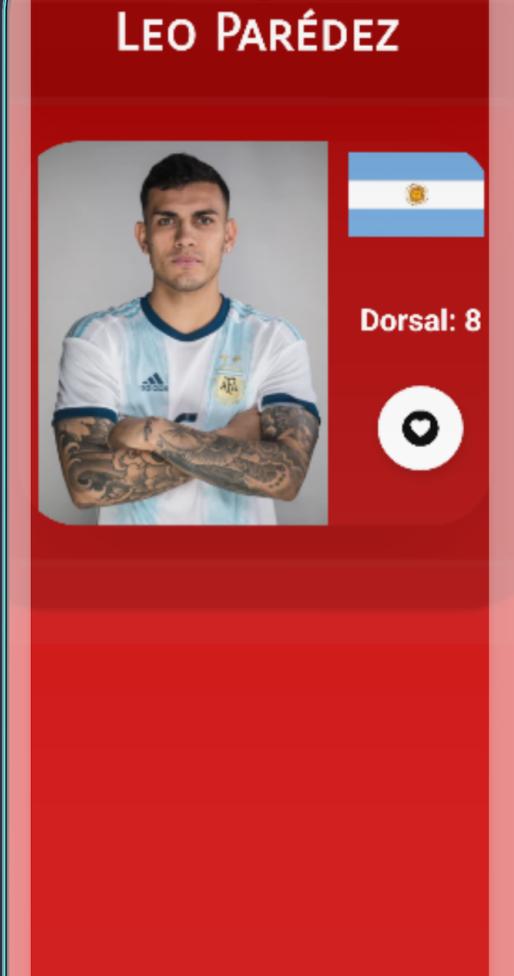
```
recyclerView = findViewById(R.id.recycler);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
BBDD bd = new BBDD(this);
JugadorAdapter adapter =
new JugadorAdapter(bd.getAllPlayers(), this);
recyclerView.setAdapter(adapter);
```

FUNCIONALIDAD

Visualización de datos de todos los jugadores más relevantes que pueden ser obtenidos de manera aleatoria, gracias a los widgets que nos brinda el Android Studio. Los datos se pasan a RecyclerView a través de un adaptador, que es responsable de crear vistas para cada elemento de datos y vincularlas a los datos correspondientes.

RECYCLER VIEW

INTERFAZ



FUNCIONALIDAD

RecyclerView en Android es un componente de IU que se utiliza para mostrar grandes conjuntos de datos de una manera eficiente. Con RecyclerView, los datos se representan mediante vistas de elementos de diseño personalizado, que se reutilizan y solo se muestran en la pantalla cuando son necesarios. Esto mejora el rendimiento de la aplicación y permite una navegación suave y rápida a través de grandes conjuntos de datos.



```
public class JugadorViewHolder extends RecyclerView.ViewHolder {  
    TextView viewNombre;  
    ImageView imagenJugador;  
    ImageView imagenPais;  
    FloatingActionButton btnLike;  
  
    public JugadorViewHolder(@NonNull View itemView) {  
        super(itemView);  
        viewNombre = itemView.findViewById(R.id.nombreDelJugador);  
        imagenJugador = itemView.findViewById(R.id.imagenDelJugador);  
        imagenPais = itemView.findViewById(R.id.imagenDelPais);  
        btnLike = itemView.findViewById(R.id.botonLike);  
    }  
}
```

Un ViewHolder en RecyclerView es una clase que mantiene las referencias a las vistas de cada elemento en la lista, permitiendo el acceso eficiente y rápido a los datos. Se utiliza para reducir la carga en la CPU y la memoria, ya que solo se inflan las vistas necesarias en lugar de todas las vistas de la lista.

onBindViewHolder() es un método de la clase RecyclerView.Adapter que se llama cuando una vista de

ViewHolder está lista para ser mostrada en la pantalla. Este método enlaza los datos del modelo con la vista del ViewHolder y actualiza la información que se muestra al usuario.

```
public void onBindViewHolder(@NonNull JugadorViewHolder holder,  
                           int position) {  
    holder.viewNombre.setText(listaJugadores.get(position).getNombre());  
  
    String uri = "@drawable/" + listaJugadores.get(position).getImagen();  
    Drawable imagen = ContextCompat  
        .getDrawable(this.contexto.getApplicationContext(), uri);  
    holder.imagenJugador.setImageDrawable(imagen);  
  
    //para mostrar la foto del pais.  
    String image = "@drawable/" + listaJugadores.get(position).getPais();  
  
    Drawable imagens = ContextCompat  
        .getDrawable(this.contexto.getApplicationContext(), image);  
    holder.imagenPais.setImageDrawable(imagens);  
}
```

PERSISTENCIA

SQLITE

SQLite es una base de datos relacional liviana y autónoma, implementada como una biblioteca de C que se integra en aplicaciones. En Android Studio, se puede utilizar SQLite para almacenar y recuperar datos de manera eficiente. Para integrar SQLite en una aplicación de Android, es necesario crear una clase de ayuda de base de datos que extienda SQLiteOpenHelper. Luego, se puede utilizar la clase SQLiteDatabase para realizar operaciones de base de datos, como crear tablas, insertar datos, actualizar y eliminar registros. Las consultas se realizan mediante el método query() y los resultados se pueden manipular utilizando un Cursor.

TABLA

En SQLite, las tablas se crean utilizando una sentencia CREATE TABLE. En una aplicación de Android, la sentencia CREATE TABLE se suele ejecutar dentro del método onCreate() de una clase que extiende SQLiteOpenHelper. En este método, se crea una instancia de SQLiteDatabase y se llama al método execSQL() para ejecutar la sentencia CREATE TABLE

```
@Override  
public void onCreate(SQLiteDatabase db) {  
    //creación de tabla de todos los jugadores  
    db.execSQL("create table " + nombreTable +  
              "( " + nombre TEXT NOT NULL," +  
              "apellido TEXT NOT NULL," +  
              "imagen TEXT NOT NULL," +  
              "pais TEXT NOT NULL," +  
              "MeGusta TEXT NOT NULL," +  
              "dorsal INTEGER NOT NULL");  
}
```

```
public class DBHelper extends SQLiteOpenHelper {  
  
    private static final int DATABASE_VERSION = 1;  
    private static final String DATABASE_NAME = "MiBaseDeDatos.db";  
  
    public DBHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        //crear tabla  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion,  
                         int newVersion) {  
        //eliminar tabla y crear nuevo en su lugar  
    }  
}
```

RESULTADO APP

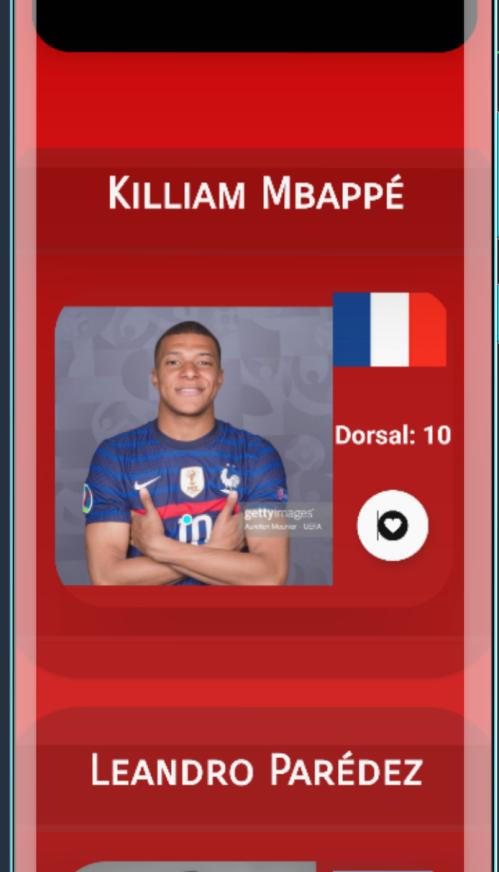
INICIO



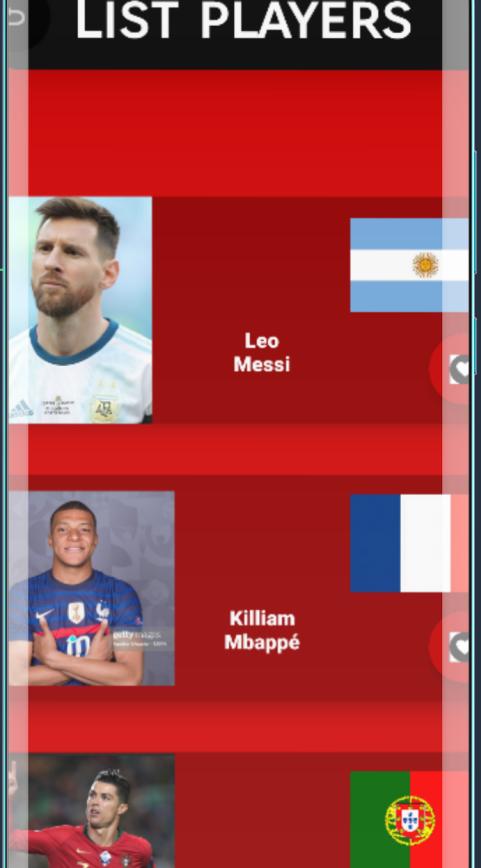
VIEW MENÚ



MY PLAYERS



LIST PLAYERS



GET PLAYER

