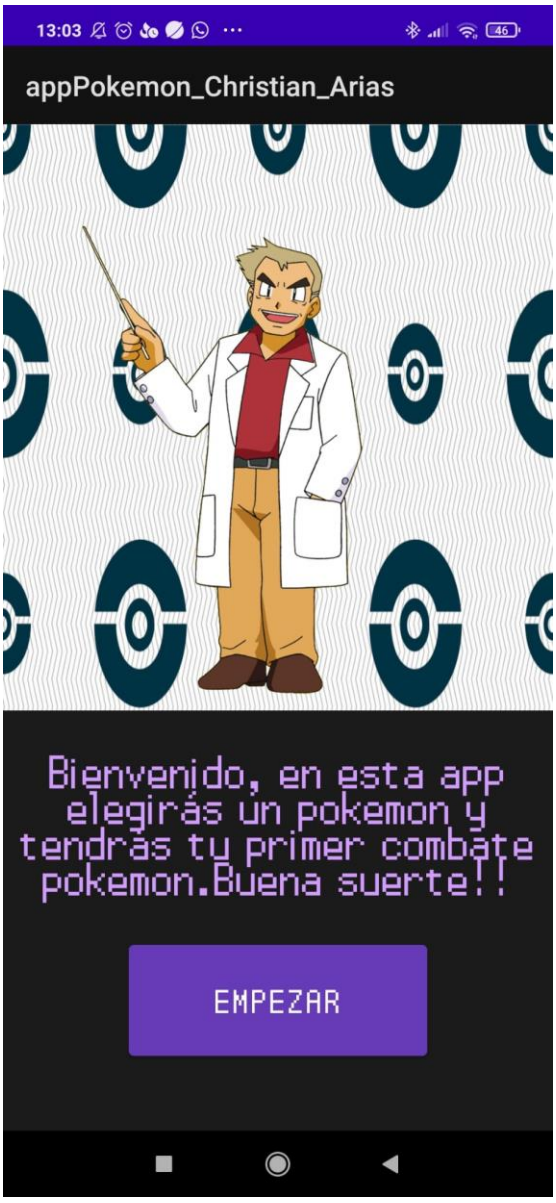
Documentación AUT2-5. **Christian Daniel Arias Chocano**

App Pokémon

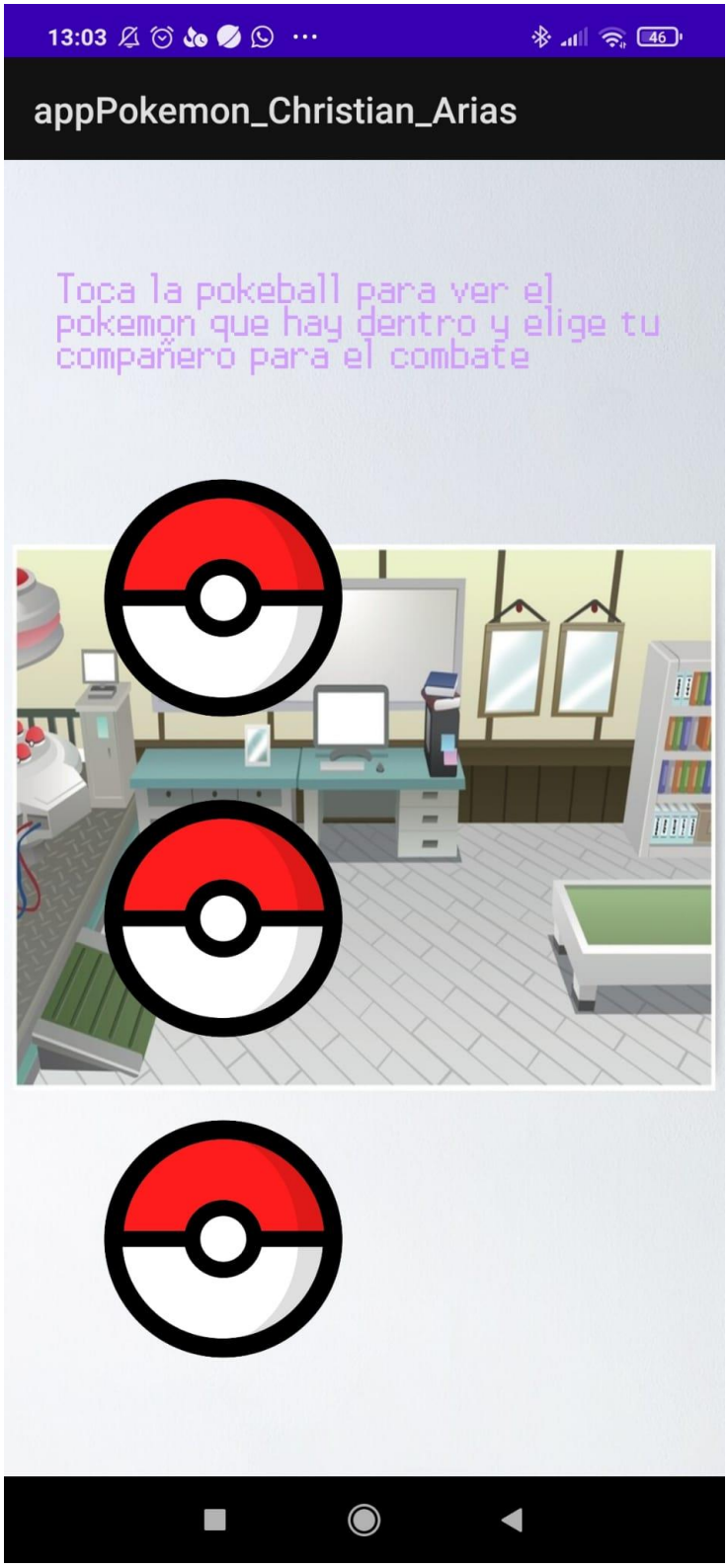
La documentación se dividirá en dos partes, en la primera se verá el funcionamiento de la app y en la segunda se explicarán las clases de la aplicación. Se recomienda ver el video adjuntado antes de leer la documentación.

Funcionamiento

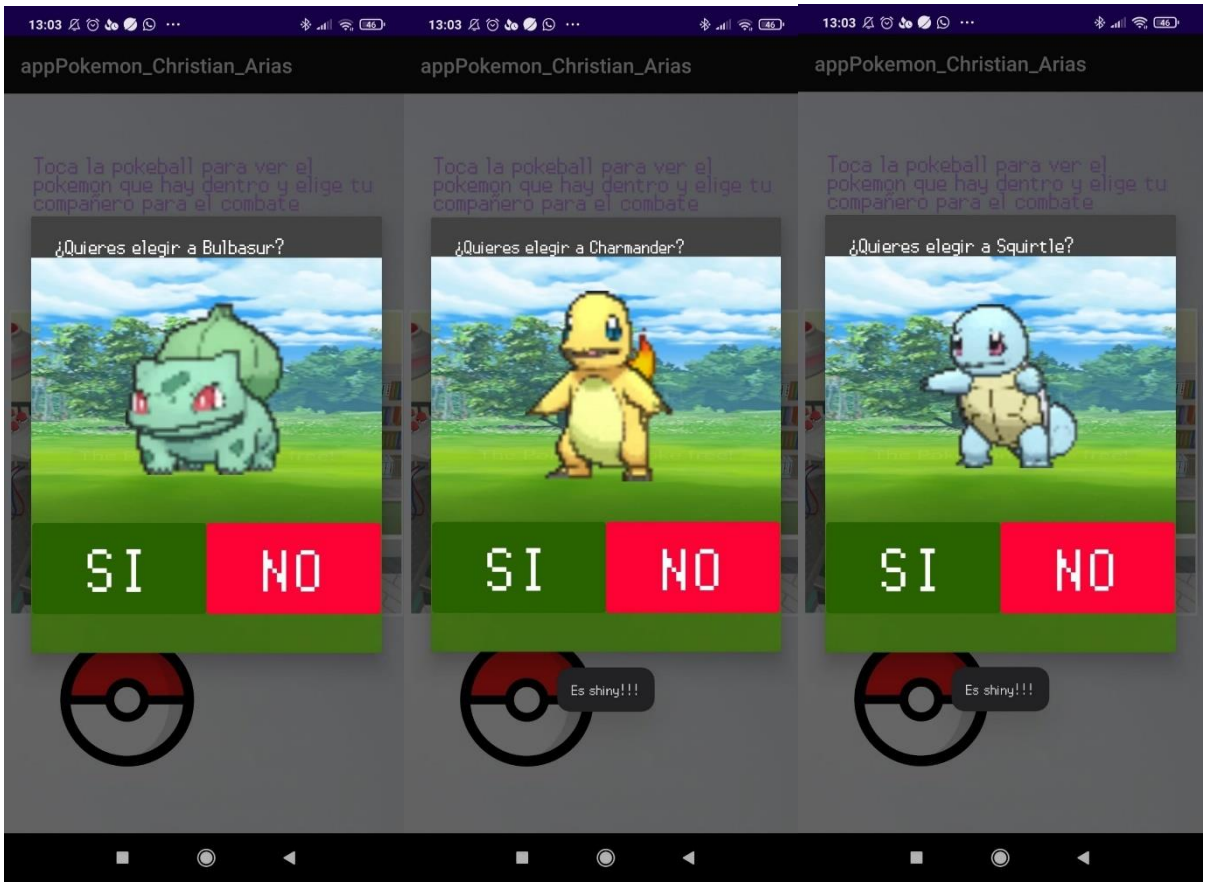
Nada más entrar a la app se nos explicará la función de la app y tendremos un botón “Empezar” en el cual si hacemos click pasaremos a la siguiente pantalla



En la siguiente pantalla tendremos una pequeña explicación y 3 botones con forma de pokeball, cada uno tendrá un gif de un pokemon y nos lo mostrará en un dialogo al pulsar el botón



Podremos elegir a Bulbasur, Squirtle o Charmander, además tendrán una probabilidad de salir shinys (de otro color).

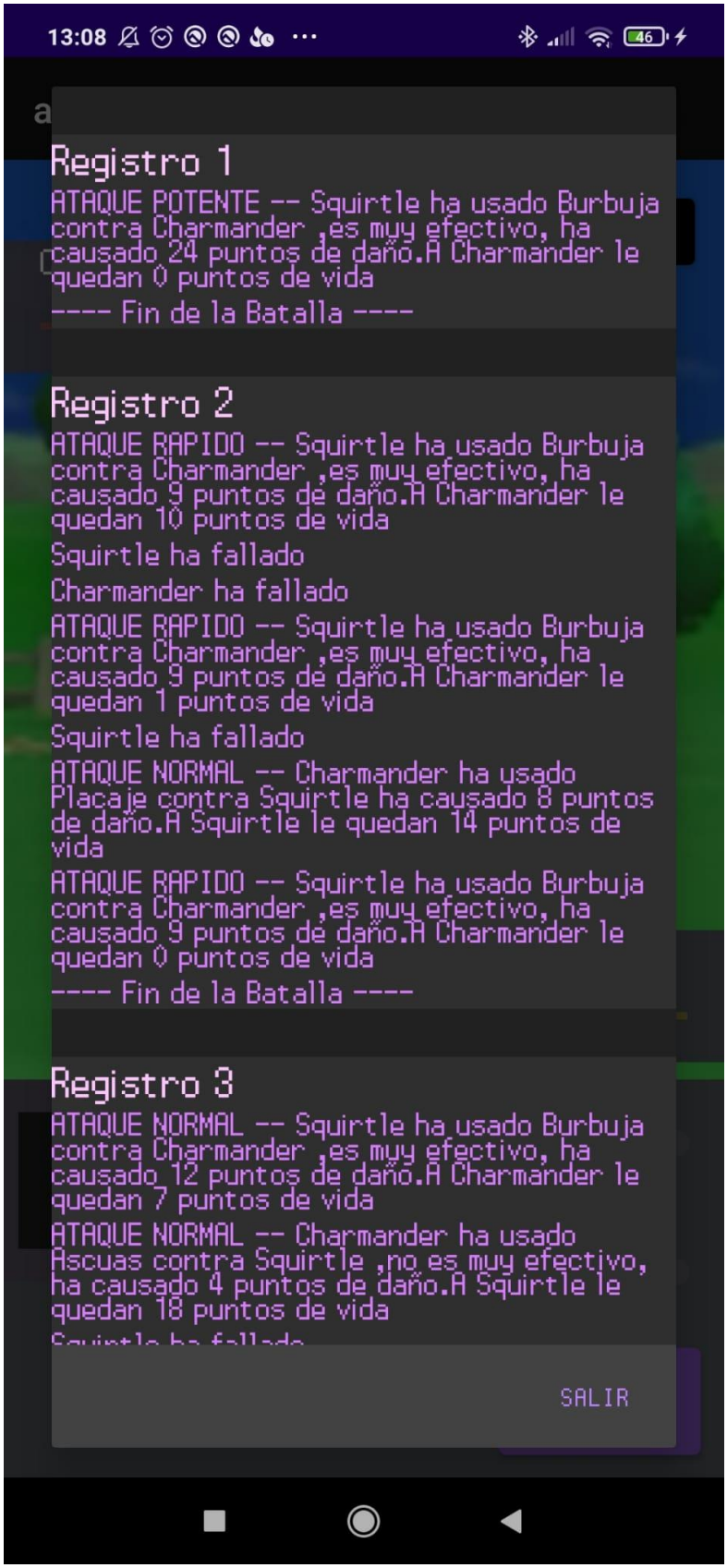


Además, Al elegir al pokemon luego te preguntará en otro dialogo si le quieres poner un mote para que se muestre después en el combate

La interfaz del combate es estará compuesta por:

- Tendremos entre 1 y 4 botones con los ataques del pokemon, siendo estos del color del tipo correspondiente
- Un switch para hacer ataques rápidos, los cuales tendrán una probabilidad de atacar dos veces pero bajando el daño del ataque normal.
- Un switch para hacer ataques potentes, los cuales pegarán más pero le darán al enemigo la oportunidad de atacarnos dos veces.
- Dos barras de vida con el nombre de nuestro pokemon y la vida que le falta.
- Un botón de reset para empezar un nuevo combate.
- Un botón de registro para ver lo que ha sucedido en combates anteriores.

El registro se verá de la siguiente manera:



Explicación del código.

La app estará formada 4 paquetes e importará dos librerías (androidx.dynamicanimation:dynamicanimation:1.0.0 y com.github.bumptech.glide:glide:4.14.2). Los paquetes son los siguientes:

PokemonPackage

Este será el encargado de contener todas las clases relacionadas a como guardamos la información de un pokemon y los metodos que nos permiten trabajar con ellos. Estará formado por 3 clases:

1. TipoPokemon

Como su nombre indica, en esta clase guardaremos toda la información relacionada al tipo de los pokemon. Implementará la interfaz Serializable y está formada por:

Una enumeración llamada NombreTipoPokemon que será privada y contendrá todos los nombres posibles de los pokemon

```
private enum NombreTipoPokemon implements Serializable {
    AGUA, FUEGO, PLANTA, NORMAL
}
```

Los atributos de la clase serán los siguientes:

```
private final int nombre;
private final NombreTipoPokemon tipoPokemon;
private final NombreTipoPokemon[] debilidades;
private final NombreTipoPokemon[] fortalezas;
private final int color;
```

En este caso cabe destacar que tanto nombre como color serán referencias a R.string y a R.color respectivamente.

En cuanto a los métodos tendremos getters para todos los atributos menos debilidades y fortalezas , y un metodo contains que nos servirá para ver si el nombre de un objeto tipoPokemon pertenece a un array de nombres, esto lo haremos para ver si un tipo es fuerte o débil en otros dos metodos, el codigo sería el siguiente:

```
public boolean esDebil(TipoPokemon otroTipo) {
    return contains(this.debilidades, otroTipo.getTipoPokemon());
}

public boolean esFuerte(TipoPokemon otroTipo) {
    return contains(this.fortalezas, otroTipo.getTipoPokemon());
}

private boolean contains(NombreTipoPokemon[] arrayNombres, NombreTipoPokemon nombreTipo) {
    for (NombreTipoPokemon miTipoPokemon : arrayNombres)
        if (nombreTipo == miTipoPokemon)
            return true;
    return false;
}
```

Y para el final, hay que destacar que nuestro constructor de TipoPokemon será privado y solo podremos acceder a unos atributos estáticos de TipoPokemon creados por defecto, esto lo hago para evitar que se puedan crear otros tipos de pokemon y tener unos ya creados por defecto. El código será el siguiente:

```
public static final TipoPokemon TIPO_FUEGO =
        new TipoPokemon(
                R.string.fuego,
                NombreTipoPokemon.FUEGO,
                new NombreTipoPokemon[]{NombreTipoPokemon.AGUA},
                new NombreTipoPokemon[]{NombreTipoPokemon.PLANTA},
                R.color.tipoFuego);

public static final TipoPokemon TIPO_AGUA =
        new TipoPokemon(
                R.string.agua,
                NombreTipoPokemon.AGUA,
                new NombreTipoPokemon[]{NombreTipoPokemon.PLANTA},
                new NombreTipoPokemon[]{NombreTipoPokemon.FUEGO},
                R.color.tipoAgua);

public static final TipoPokemon TIPO_PLANTA =
        new TipoPokemon(
                R.string.planta,
                NombreTipoPokemon.PLANTA,
                new NombreTipoPokemon[]{NombreTipoPokemon.FUEGO},
                new NombreTipoPokemon[]{NombreTipoPokemon.AGUA},
                R.color.tipoPlanta);

public static final TipoPokemon TIPO_NORMAL =
        new TipoPokemon(
                R.string.normal,
                NombreTipoPokemon.NORMAL,
                new NombreTipoPokemon[]{},
                new NombreTipoPokemon[]{},
                R.color.tipoNormal);
```

2. AtaquePokemon

En esta clase guardamos la información referente a los ataque de los pokemon, implementará la interfaz Serializable y sus atributos serán los siguientes:

```
private final int nombre;
private final int danio;
private final TipoPokemon tipoAtaque;
private final double porcentajeAcierto; //entre 0 y 1
private boolean isSpeeded = false;
private boolean isPowered = false;
```

En cuanto a metodos tendremos getters de todos los atributos menos danio y porcentajeAcierto y tendremos setters de isSpeeded y de isPowered. A parte de estos métodos tendremos el método de atacar en el que calcularemos el daño que le haremos a un TipoPokemon:

```
public int atacar(TipoPokemon tipoPokemonEnemigo) {
    int damage;

    if (Math.random() > this.porcentajeAcierto) return 0;
    else if (tipoPokemonEnemigo.esDebil(this.tipoAtaque)) damage = (int) Math.round(danio * 1.5);
    else if (tipoPokemonEnemigo.esFuerte(this.tipoAtaque)) damage = (int) Math.round(danio * 0.5);
    else damage=this.danio;

    if(this.isSpeeded) damage*=0.75d;
    else if (this.isPowered) damage*=2;

    return damage;
}
```

A parte de esto al igual que con la clase anterior tendremos el constructor en privado y utilizaremos unos métodos estáticos para crear los ataques:

```
public static AtaquePokemon PLACAJE() {
    return new AtaquePokemon(R.string.placaje, potencia: 8, TipoPokemon.TIPO_NORMAL, porcentajeAcierto: 1);
}

public static AtaquePokemon ASCUAS() {
    return new AtaquePokemon(R.string.ascuas, potencia: 8, TipoPokemon.TIPO_FUEGO, porcentajeAcierto: 0.8d);
}

public static AtaquePokemon BURBUJA() {
    return new AtaquePokemon("Burbuja", potencia: 8, TipoPokemon.TIPO_AGUA, porcentajeAcierto: 0.8d);
}

public static AtaquePokemon LATIGOCEPA() {
    return new AtaquePokemon("Latigo cepa", potencia: 8, TipoPokemon.TIPO_PLANTA, porcentajeAcierto: 0.8d);
}

public static AtaquePokemon ARANIAZO() {
    return new AtaquePokemon("Arañazo", potencia: 12, TipoPokemon.TIPO_NORMAL, porcentajeAcierto: 0.6d);
}
```

3. Pokemon

Esta será la clase donde se guardará toda la información relacionada al pokemon, implementará Serializable y tendrá los siguientes atributos:

```
private final int id;
private final int nombre;
private String mote;

private final boolean isShiny;
private final int imagenNormal;
private final int imagenShiny;
private final int imagenNormalEspalda;
private final int imagenShinyEspalda;

private final TipoPokemon tipoPokemon;
private final AtaquePokemon[] ataquesPokemon;
private int vidaRestantePokemon;
private final int vidaTotalPokemon;
```

Después tendremos varios getters y además setters de los atributos que no son constantes. Además tendremos los siguientes métodos:

- **buildPokemonbyId**. Nos da un pokemon con la id que le hayamos pasado.

```
public static Pokemon buildPokemonbyId(int id) {
    if (id == 1) return Pokemon.BULBASUR();
    else if (id == 2) return Pokemon.CHARMANDER();
    else if (id == 3) return Pokemon.SQUIRTLE();
    else return Pokemon.CHARMANDER();
}
```

- **getImage**. Nos devuelve la imagen frontal del pokemon, ya sea la shiny o la normal.

```
public int getImagen() {
    if (this.isShiny) return this.imagenShiny;
    else return this.imagenNormal;
}
```

- **getImageEspalda**. Igual que getImage pero con la imagen de la espalda.

```
public int getImagenEspalda() {
    if (this.isShiny) return this.imagenShinyEspalda;
    else return this.imagenNormalEspalda;
}
```

- **getPorcentajeVida**. Nos devuelve el porcentaje de la vida que le queda al pokemon.

```
public int getPorcentajeVida() {
    return (this.vidaRestantePokemon * 100) / this.vidaTotalPokemon;
}
```

- **estaVivo**. Nos dice si está vivo el pokemon.

```
public boolean estaVivo() { return this.vidaRestantePokemon > 0; }
```

- **recibeDanio**. Metodo para bajarle la vida al pokemon.

```
public void recibeDanio(int vidaPerdida) {
    this.vidaRestantePokemon = this.vidaRestantePokemon - vidaPerdida;
    if (this.vidaRestantePokemon < 0) this.vidaRestantePokemon = 0;
}
```

- **getRandomAttack**. Nos devuelve un ataque random del pokemon.

```
public AtaquePokemon getRandomAttack() {
    int numeroAtaque;
    AtaquePokemon ataqueAleatorio;
    do {
        numeroAtaque = new Random().nextInt( bound: 4);
        ataqueAleatorio = this.getAtaquesPokemon()[numeroAtaque];
    } while (ataqueAleatorio == null);

    return ataqueAleatorio;
}
```

- **curarVida**. Cura la vida del pokemon.

```
public void curarVida() { this.vidaRestantePokemon = this.vidaTotalPokemon; }
```

Además como en las clases anteriores tendremos el constructor privado y un metodo estático para construir el objeto pokemon.

```
public static Pokemon BULBASUR() {

    AtaquePokemon[] ataquesPokemon = new AtaquePokemon[4];
    ataquesPokemon[0] = AtaquePokemon.PLACAJE();
    ataquesPokemon[1] = AtaquePokemon.LATIGOCEPA();
    if (Math.random() > 0.3d) ataquesPokemon[2] = AtaquePokemon.ARANIAZO();
    else ataquesPokemon[2] = null;
    ataquesPokemon[3] = null;

    return new Pokemon( id: 1,
            "Bulbasur",
             mote: null,
            R.drawable.bulbasaur,
            R.drawable.bulbasur_shiny,
            R.drawable.bulbasur_back,
            R.drawable.bulbasur_shiny_back,
            TipoPokemon.TIPO_PLANTA,
            ataquesPokemon,
             porcentajeShiny: 0.25d,
             vidaRestantePokemon: 24,
             vidaTotalPokemon: 24
    );
}
```

```
public static Pokemon CHARMANDER() {

    AtaquePokemon[] ataquesPokemon = new AtaquePokemon[4];
    ataquesPokemon[0] = AtaquePokemon.PLACAJE();
    ataquesPokemon[1] = AtaquePokemon.ASCUAS();
    if (Math.random() > 0.7d) ataquesPokemon[2] = AtaquePokemon.ARANIAZO();
    else ataquesPokemon[2] = null;
    ataquesPokemon[3] = null;

    return new Pokemon( id: 2, "Charmander",
             mote: null, R.drawable.charmander,
            R.drawable.charmander_shiny,
            R.drawable.charmander_back,
            R.drawable.charmander_shiny_back,
            TipoPokemon.TIPO_FUEGO,
            ataquesPokemon,
             porcentajeShiny: 0.25d,
             vidaRestantePokemon: 19,
             vidaTotalPokemon: 19
    );
}
```

```
public static Pokemon SQUIRTLE() {

    AtaquePokemon[] ataquesPokemon = new AtaquePokemon[4];
    ataquesPokemon[0] = AtaquePokemon.PLACAJE();
    ataquesPokemon[1] = AtaquePokemon.BURBUJA();
    if (Math.random() > 0.5d) ataquesPokemon[2] = AtaquePokemon.ARANIAZO();
    else ataquesPokemon[2] = null;
    ataquesPokemon[3] = null;

    return new Pokemon( id: 3,
            "Squirtle",
             mote: null,
            R.drawable.squirtle,
            R.drawable.squirtle_shiny,
            R.drawable.squirtle_back,
            R.drawable.squirtle_shiny_back,
            TipoPokemon.TIPO_AGUA,
            ataquesPokemon,
             porcentajeShiny: 25d,
             vidaRestantePokemon: 22,
             vidaTotalPokemon: 22);
}
```

UI_Elements

En este paquete tendremos las clases encargadas de ciertas operaciones de la interfaz de usuario. Tendremos 3 clases:

1. RelativeLayoutButton.

Este botón está sacado del siguiente tutorial, este tipo de botón lo uso en la actividad elección pokemon en las pokeball, para saber más entrar en el siguiente enlace:

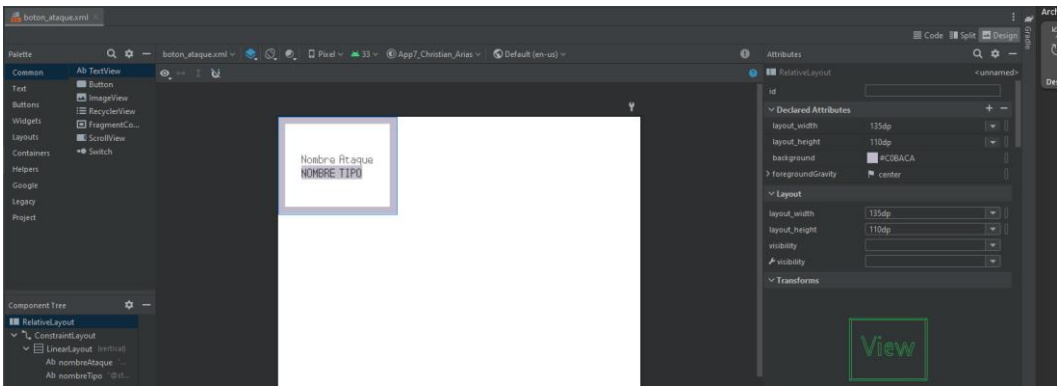
http://izvornikod.com/Blog/tabid/82/EntryId/8/Creating-Android-button-with-image-and-text-using-relative-layout.aspx

2. RelativeAtaquePokemonLayoutButton.

Esta clase hereda de RelativeAtaquePokemonLayoutButton, y nos crea los botones de la actividad de combate pokemon pasándole un AtaquePokemon, el codigo de la clase queda así:

```
public class RelativeAtaquePokemonLayoutButton extends RelativeLayoutButton {
    private AtaquePokemon ataquePokemon;

    public RelativeAtaquePokemonLayoutButton(Context context, int id, AtaquePokemon ataquePokemon) {
        super(context, id);
        this.ataquePokemon = ataquePokemon;
        RelativeLayout relativeLayout = findViewById(id);
        if (ataquePokemon != null) {
            TipoPokemon tipoAtaquePokemon = ataquePokemon.getTipoAtaque();
            TextView textViewTipoAtaque = relativeLayout.findViewById(R.id.nombreTipo);

            relativeLayout.setBackgroundColor(getResources().getColor(tipoAtaquePokemon.getColor()));
            setText(R.id.nombreAtaque,getResources().getString(ataquePokemon.getNombre()));
            setText(textViewTipoAtaque.getId(),getResources().getString(tipoAtaquePokemon.getNombre()));
            textViewTipoAtaque.setBackgroundColor(getResources().getColor(tipoAtaquePokemon.getColor()));

        }else relativeLayout.setVisibility(INVISIBLE);
    }

    public AtaquePokemon getAtaquePokemon() { return ataquePokemon; }
}
```

El botón deberá usar el siguiente layout como modelo:



El codigo XML será el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="135dp"
    android:layout_height="110dp"
    android:background="#C0BACA"
    android:foregroundGravity="center">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:background="#FFFFFF">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:orientation="vertical"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <TextView
                android:id="@+id/nombreAtaque"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
                android:text="@string/nombre_ataque" />

            <TextView
                android:id="@+id/nombreTipo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:background="#F0C0BACA"
                android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
                android:text="@string/nombre_tipo"
                android:textAllCaps="true"
                android:textColor="#424242" />
        </LinearLayout>
    </androidx.constraintlayout.widget.ConstraintLayout>
</RelativeLayout>
```

3. TypeFaceStringBuilder.

Esta clase la usaremos para crear objetos de la clase SpannableString con la fuente que nosotros queramos. Tendremos un metodo para crearlo introduciendo una secuencia de caracteres y otro introduciendo un id de R.string. El codigo es el siguiente:

```
public class TypeFaceStringMaker {
    private static int font;

    public TypeFaceStringMaker(int font) { this.font = font; }

    public SpannableString build(Context context, CharSequence chars) {
        Typeface typeface = ResourcesCompat.getFont(context, font);
        if (chars == null) {
            return null;
        }
        SpannableString s = new SpannableString(chars);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
            s.setSpan(new TypefaceSpan(typeface),  start: 0, s.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
        }
        return s;
    }

    public SpannableString build(Context context, int id) {
        Typeface typeface = ResourcesCompat.getFont(context, font);
        CharSequence chars = context.getString(id);
        if (chars == null) {
            return null;
        }
        SpannableString s = new SpannableString(chars);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
            s.setSpan(new TypefaceSpan(typeface),  start: 0, s.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
        }
        return s;
    }

}
```

BattleProcesation

Esta clase será la encargada de procesar los turnos de ataque y sus acciones en la UI. Este paquete estará formado por las siguientes clases:

1. Animación

Es la clase que se ocupa de las animaciones de las dos imágenes, tendrá dos atributos y un constructor público

```
private View imagenMiPokemon;
private View imagenPokemonRival;

public Animacion(View imagenMiPokemon, View imagenPokemonRival) {
    this.imagenMiPokemon = imagenMiPokemon;
    this.imagenPokemonRival = imagenPokemonRival;
}
```

Cada metodo de esta clase será una animación, Además para los desplazamientos de objetos tenemos un metodo especial desplazamiento encargado de mover en diagonal nuestra View:

```
private void desplazamientoLateral(float desplazamientoX, float desplazamientoY, View view) {
    // Setting up a spring animation to animate the view1 and view2 translationX and translationY properties
    final SpringAnimation anim1X = new SpringAnimation(view,
            DynamicAnimation.TRANSLATION_X, desplazamientoX);
    final SpringAnimation anim1Y = new SpringAnimation(view,
            DynamicAnimation.TRANSLATION_Y, desplazamientoY);
    anim1X.start();
    anim1Y.start();
}
```

Los dos métodos que hacen uso de desplazamiento son:

- animacionAtaqueMiPokemon: Es el método encargado de la animación de cuando atacamos con nuestro pokemon en la actividad de batalla, recibirá por parámetros el tiempo en milisegundos que tardará en volver a su posición inicial. El código es el siguiente:

```
public void animacionAtaqueMiPokemon(int delay) {
    float desplazamientoXIda = imagenPokemonRival.getRight() - imagenMiPokemon.getRight();
    float desplazamientoYIda = imagenMiPokemon.getTop() - imagenPokemonRival.getTop();
    float desplazamientoEmpuje = 15;

    desplazamientoLateral(desplazamientoXIda, desplazamientoYIda, imagenMiPokemon);
    desplazamientoLateral(desplazamientoEmpuje, -desplazamientoEmpuje, imagenPokemonRival);

    Handler handler = new Handler();
    handler.postDelayed(() -> {
        float desplazamientoXVuelta = ((imagenMiPokemon.getRight() - imagenMiPokemon.getLeft()) * 0.75f) - desplazamientoXIda;
        desplazamientoLateral(desplazamientoXVuelta, -desplazamientoYIda, imagenMiPokemon);
        desplazamientoLateral(-desplazamientoEmpuje, desplazamientoEmpuje, imagenPokemonRival);
    }, delay);
}
```

- animacionDefensaMiPokemon. Es el método encargado de la animación de cuando nuestro pokemon es atacado en la actividad de batalla, recibirá por parámetros el tiempo en milisegundos que tardará en volver a su posición inicial. El código es el siguiente:

```
public void animacionDefensaMiPokemon(int delay) {
    float desplazamientoXIda = imagenMiPokemon.getRight() - imagenPokemonRival.getRight();
    float desplazamientoYIda = imagenMiPokemon.getTop() - imagenPokemonRival.getTop() + 40;
    float desplazamientoEmpuje = -15;

    desplazamientoLateral(desplazamientoXIda, desplazamientoYIda, imagenPokemonRival);
    desplazamientoLateral(desplazamientoEmpuje, -desplazamientoEmpuje, imagenMiPokemon);

    Handler handler = new Handler();
    handler.postDelayed(() -> {
        float desplazamientoXVuelta = (imagenMiPokemon.getRight() - imagenPokemonRival.getLeft()) * 0.3f;
        desplazamientoLateral(-desplazamientoXVuelta, -desplazamientoYIda, imagenPokemonRival);
        desplazamientoLateral(-desplazamientoEmpuje, desplazamientoEmpuje, imagenMiPokemon);
    }, delay);
}
```

Después tenemos el método animacionMuerte que recibirá un booleano con true si es mi pokemon el debilitado y con false si es el pokemon enemigo el debilitado y el contexto de la imagen.

Usaremos una animación de R.anim llamada desaparecer cuyo xml será el siguiente:

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<alpha
    android:fromAlpha="1.0"
    android:toAlpha="0.0"
    android:duration="2000"
    />
</set>
```

El codigo del método será el siguiente:

```java
public void animacionMuerte(boolean isMiPokemon,Context context) {
    View imagen;
    if (isMiPokemon) imagen = this.imagenMiPokemon;
    else imagen = this.imagenPokemonRival;

    Animation animacion = AnimationUtils.loadAnimation(context, R.anim.desaparecer);
    animacion.setFillAfter(true);
    imagen.startAnimation(animacion);
    new Handler().postDelayed(() -> imagen.setAlpha(0), animacion.getDuration());
}
```

Y por último tenemos el método animaciónAparicion, al cual le pasaremos el contexto de nuestras imágenes y se encargará de hacer visibles a los dos pokemon después del combate.

Usará la animación aparecer de R.anim, su código es el siguiente:

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    />
</set>
```

El codigo del método es el siguiente:

```java
public void animacionAparicion(Context context){
    Animation animacion = AnimationUtils.loadAnimation(context, R.anim.aparecer);
    animacion.setFillAfter(true);
    this.imagenPokemonRival.startAnimation(animacion);
    this.imagenMiPokemon.startAnimation(animacion);
    this.imagenPokemonRival.setAlpha(1);
    this.imagenMiPokemon.setAlpha(1);
}
```

2. Pokemon_HPBar

Esta es la clase encargada de la gestión de la barra de vida, está formada por un pokemon y una barra de progreso, el constructor y los atributos son los siguientes:

```
public class Pokemon_HPBar {
    private final Pokemon pokemon;
    ProgressBar barraVida;


    public Pokemon_HPBar(ProgressBar progressBar, Pokemon pokemon) {
        this.barraVida = progressBar;
        this.pokemon = pokemon;
        DrawableCompat.setTint(this.barraVida.getProgressDrawable(), ContextCompat.getColor(this.barraVida.getContext(), R.color.green));
    }
```

*La última línea del constructor es para poner la barra de color verde

Tendremos los siguientes métodos:

- **getPokemon**. Es un getter del pokemon que tenemos.
- **getString**. Método que introduciendo un id de R.string nos devuelve el String que hemos pedido.

```
private String getString(int id) {
    return barraVida.getContext().getString(id);
}
```

- **getNombrePokemon.** Devuelve el nombre del pokemon en String o el mote en caso de tenerlo.

```
private String getNombrePokemon() {
    String mote = this.pokemon.getMote();
    if (mote!= null) return mote;
    else return getString(this.pokemon.getNombre());
}
```

- **isPokemonDebilitado.** Llama al metodo estaVivo del pokemon de la clase.

```
public boolean isPokemonDebilitado() {return !(this.pokemon.estaVivo());}
```

- **actualizarHPBar.** Como su nombre indica actualiza la barra de vida con el porcentaje de vida del pokemon y cambia de color la barra en caso de haber bajado de un porcentaje de vida, siendo la barra verde por encima del 50%, amarilla cuando está entre el 25% y el 50% y roja cuando baja del 25%

```
public void actualizarHPBar() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N)
        this.barraVida.setProgress(this.pokemon.getPorcentajeVida(), animate: true);

    if (this.barraVida.getProgress() < 25)
        DrawableCompat.setTint(this.barraVida.getProgressDrawable(), ContextCompat.getColor(this.barraVida.getContext(), R.color.red));
    else if (this.barraVida.getProgress() < 50)
        DrawableCompat.setTint(this.barraVida.getProgressDrawable(), ContextCompat.getColor(this.barraVida.getContext(), R.color.yellow));
    else
        DrawableCompat.setTint(this.barraVida.getProgressDrawable(), ContextCompat.getColor(this.barraVida.getContext(), R.color.green));
}
```

- **buildMensaje**. Construye el mensaje que se va a meter en el registro y en caso de que el ataque falle lo muestra en un Toast.

```
private SpannableString buildMensaje(int damage, String nombrePokemonAtacante, AtaquePokemon ataquePokemon) {
    TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gela);
    String nombrePokemon = getNombrePokemon();
    String nombreAtaque = getString(ataquePokemon.getNombre());
    String mensaje = getString(R.string.ataqueNormal);
    if (ataquePokemon.isPowered()) mensaje = getString(R.string.ataquePotente);
    else if (ataquePokemon.isSpeeded()) mensaje = getString(R.string.ataqueRapido);
    mensaje = mensaje + " ";
    if (damage == 0) {
        mensaje = nombrePokemonAtacante + " " + getString(R.string.mensajeFallo);
        Toast.makeText(this.barraVida.getContext(), typeFaceStringMaker.build(this.barraVida.getContext(), mensaje), Toast.LENGTH_SHORT).show();
    } else if (ataquePokemon.getTipoAtaque().esDebil(this.pokemon.getTipoPokemon()))
        mensaje = mensaje + (nombrePokemonAtacante + " ha usado " + nombreAtaque
                + " contra " + nombrePokemon + " ,no es muy efectivo, ha causado "
                + damage + " puntos de daño.A " + nombrePokemon + " le quedan "
                + this.pokemon.getVidaRestante() + " puntos de vida");
    else if (ataquePokemon.getTipoAtaque().esFuerte(this.pokemon.getTipoPokemon()))
        mensaje = mensaje + (nombrePokemonAtacante + " ha usado " + nombreAtaque
                + " contra " + nombrePokemon + " ,es muy efectivo, ha causado "
                + damage + " puntos de daño.A " + nombrePokemon + " le quedan "
                + this.pokemon.getVidaRestante() + " puntos de vida");
    else
        mensaje = mensaje + (nombrePokemonAtacante + " ha usado " + nombreAtaque
                + " contra " + nombrePokemon + " ha causado "
                + damage + " puntos de daño.A " + nombrePokemon + " le quedan "
                + this.pokemon.getVidaRestante() + " puntos de vida");
    return typeFaceStringMaker.build(this.barraVida.getContext(), mensaje);
}
```

- **recibeAtaque.** Es el método más importante de la clase, es el que llamamos cuando un pokemon recibe un ataque de otro, devuelve el mensaje creado en buildMensaje.

```
public SpannableString recibeAtaque(String nombrePokemonaAtacante, AtaquePokemon ataquePokemon) {
    int damage = ataquePokemon.atacar(this.pokemon.getTipoPokemon());
    this.pokemon.recibeDanio(damage);
    actualizarHPBar();
    return buildMensaje(damage, nombrePokemonaAtacante, ataquePokemon);
}
```

3. Turno

Esta clase unirá las dos clases anteriores del paquete y gestionará el procesamiento de un turno (un ataque).

Los atributos y constructor son los siguientes:

```
public class Turno {
    private final Pokemon_HPBar progressBarHP;
    private final AtaquePokemon ataquePokemon;
    private final Animacion animacion;
    private final String nombrePokemonAtacante;
    private final boolean isAttacking;

    public Turno(Pokemon_HPBar progressBarHP, AtaquePokemon ataquePokemon, Animacion animacion, String nombrePokemonAtacante, boolean isAttacking) {
        this.progressBarHP = progressBarHP;
        this.ataquePokemon = ataquePokemon;
        this.animacion = animacion;
        this.nombrePokemonAtacante = nombrePokemonAtacante;
        this.isAttacking = isAttacking;
    }
```

Tendrá el metodo procesarTurno, el cual devuelve el mensaje generado en el objeto Pokemon_HPBar. Dependiendo de si el ataque está potenciado o es rápido, la rapidez del ataque cambiará en la animación.

```
public SpannableString procesarTurno() {
    int delayAttack = 500;
    if (this.isAttacking) {
        if (this.ataquePokemon.isPowered()) delayAttack = 1000;
        else if (this.ataquePokemon.isSpeeded()) delayAttack = 250;
        animacion.animacionAtaqueMiPokemon(delayAttack);
    } else animacion.animacionDefensaMiPokemon(delayAttack);
    return progressBarHP.recibeAtaque(this.nombrePokemonAtacante, this.ataquePokemon);
}
```

Actividades y Diálogos.

Por último, tendremos el paquete con todas las actividades y diálogos de la app, a continuación se irá explicando cada actividad y dialogo según su orden de aparición.

1. Actividad Bienvenida.

Esta actividad es la primera que aparece, y se muestra un mensaje explicando el propósito de la app y un botón que nos llevará a la siguiente actividad

```
package com.example.app7_christian_arias;

import ...

public class Bienvenida extends AppCompatActivity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bienvenida);
        Button botonComenzar = findViewById(R.id.buttonComenzar);
        botonComenzar.setOnClickListener(this);
    }

    public void onClick(View view) {
        Intent intent = new Intent( packageContext: this, EleccionPokemon.class);
        startActivity(intent);
    }

}
```


Bienvenido, en esta app elegirás un pokemon y tendrás tu primer combate pokemon.Buena suerte!!
EMPEZAR

Código:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:background="@drawable/background_bienvenida"
    tools:context=".Bienvenida">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="40dp"
        android:layout_marginBottom="120dp"
        android:src="@drawable/professor_oak"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:contentDescription="@string/imagen_profesor_oak" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="0dp"
        android:layout_height="300dp"
        android:background="#FFFDFD"
        android:backgroundTint="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
            android:text="@string/bienvenido"
            android:textAlignment="center"
            android:textColor="#673AB7"
            android:textSize="34sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/buttonComenzar"
            android:layout_width="213dp"
            android:layout_height="91dp"
            android:layout_marginTop="24dp"
            android:backgroundTint="#673AB7"
            android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
            android:text="@string/boton_Comenzar"
            android:textSize="24sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView" />

    </androidx.constraintlayout.widget.ConstraintLayout>

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

2. Actividad EleccionPokemon.

En esta actividad tendremos un pequeño texto de explicación y 3 RelativeLayoutsButtons que nos mostrarán un dialogo con los pokemon y otro dialogo para ponerles motes cuando escojamos nuestro Pokemon.

A continuación veremos las interfaces que implementa el dialogo, sus atributos y el metodo onCreate(). Solo hay que destacar que los pokemon que vamos a tener a elegir se crearan junto con la actividad y no cuando pulsemos el botón para evitar que se cree un nuevo objeto Pokemon cada vez que pulsemos un botón.

```
public class EleccionPokemon extends AppCompatActivity
        implements DialogoPonerMote.RespuestaDialogoMote,
        DialogoEleccionPokemon.RespuestaEleccionPokemon,
        View.OnClickListener {

    DialogoEleccionPokemon dialogoEleccionPokemon;
    DialogoPonerMote dialogoPonerMote;
    Pokemon[] pokemonDisponibles = {Pokemon.BULBASUR(), Pokemon.CHARMANDER(), Pokemon.SQUIRTLE()};
    Pokemon pokemonElegido;

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

        RelativeLayoutButton buttonBulbasur = new RelativeLayoutButton( context: this, R.id.pokeball_bulbasur);
        RelativeLayoutButton buttonCharmander = new RelativeLayoutButton( context: this, R.id.pokeball_charmander);
        RelativeLayoutButton buttonSquirtle = new RelativeLayoutButton( context: this, R.id.pokeball_squirtle);

        buttonBulbasur.setOnClickListener(this);
        buttonCharmander.setOnClickListener(this);
        buttonSquirtle.setOnClickListener(this);
    }
```

El método onClick() de la interfaz View.OnClickListener crea un DialogoEleccionPokemon con el pokemon correspondiente al botón pulsado.

```
@Override
public void onClick(View view) {
    Pokemon pokemonElegido = Pokemon.CHARMANDER();
    if (view.getId() == R.id.pokeball_bulbasur) pokemonElegido = pokemonDisponibles[0];
    else if (view.getId() == R.id.pokeball_charmander) pokemonElegido = pokemonDisponibles[1];
    else if (view.getId() == R.id.pokeball_squirtle) pokemonElegido = pokemonDisponibles[2];
    dialogoEleccionPokemon = new DialogoEleccionPokemon(pokemonElegido);
    dialogoEleccionPokemon.show(getSupportFragmentManager(),  tag: "Elecion");
}
```

La actividad recibirá la respuesta en el método onRespuesta del cual destacar que se llamará al otro dialogo para poner el mote del Pokemon

```
@Override
public void onRespuesta(Pokemon pokemonElegido) {
    dialogoEleccionPokemon.dismiss();
    if (pokemonElegido != null) {
        this.pokemonElegido = pokemonElegido;
        dialogoPonerMote = new DialogoPonerMote();
        dialogoPonerMote.show(getSupportFragmentManager(), tag: "Poner Mote");
    }
}
```

Y por último recibirá la respuesta del segundo dialogo en el metodo onRespuestaMote y pasará a la siguiente actividad BatallaPokemon

```
@Override
public void onRespuestaMote(String respuesta) {
    if (respuesta != null)
        this.pokemonElegido.setMote(respuesta);
    Intent intent = new Intent( packageContext: this, BatallaPokemon.class);
    intent.putExtra( name: "pokemonElegido", pokemonElegido);
    startActivity(intent);

}
```

El layout es el siguiente:

Codigo:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:background="@drawable/background_eleccion"
    tools:context=".EleccionPokemon">
    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="14dp"
        android:layout_marginBottom="32dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/instrucciones"
        app:layout_constraintVertical_bias="0.52">
        <include
            android:id="@+id/pokeball_bulbasur"
            layout="@layout/boton_pokeball"
            android:layout_width="175dp"
            android:layout_height="175dp" />
        <include
            android:id="@+id/pokeball_charmander"
            layout="@layout/boton_pokeball"
            android:layout_width="175dp"
            android:layout_height="175dp" />
        <include
            android:id="@+id/pokeball_squirtle"
            layout="@layout/boton_pokeball"
            android:layout_width="175dp"
            android:layout_height="175dp" />
    </LinearLayout>
```

```
    <TextView
        android:id="@+id/instrucciones"
        android:layout_width="350dp"
        android:layout_height="70dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="56dp"
        android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
        android:text="@string/instrucciones"
        android:textColor="#673AB7"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.491"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Dialogo EleccionPokemon.

Este es el primer dialogo de la actividad anterior, de este dialogo destacar que tendrá su propio layout ,que mostrará un Toast en caso de que nuestro pokemon salga shiny y que usará la librería Glide para que nuestro gif tenga movimiento.

```
public class DialogoEleccionPokemon extends DialogFragment implements View.OnClickListener {
    RespuestaEleccionPokemon respuestaDialogoJava;
    Pokemon pokemonElegido;

    public DialogoEleccionPokemon(Pokemon pokemonElegido) {
        super();
        this.pokemonElegido = pokemonElegido;
    }

    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        String nombrePokemon = getResources().getString(this.pokemonElegido.getNombre());

        TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gela);
        builder.setTitle(typeFaceStringMaker.build(getContext(),  chars: getString(R.string.preguntaEleccion) +" "+ nombrePokemon + "?"));

        View constraintLayout = getLayoutInflater().inflate(R.layout.dialog_confirmar_pokemon,  root: null);
        ImageView gifPokemon = constraintLayout.findViewById(R.id.imageConfirmacion);
        int imagenPokemon = this.pokemonElegido.getImagen();
        Glide.with( fragment: this).asGif().load(imagenPokemon).into(gifPokemon);

        constraintLayout.findViewById(R.id.buttonSI).setOnClickListener(this);
        constraintLayout.findViewById(R.id.buttonNO).setOnClickListener(this);

        if (this.pokemonElegido.isShiny())
            Toast.makeText(getContext(),typeFaceStringMaker.build(getContext(), getString(R.string.mensajeShiny)), Toast.LENGTH_SHORT).show();

        builder.setView(constraintLayout);
        AlertDialog alertDialog = builder.create();
        alertDialog.show();
        return alertDialog;
    }
```

Por otra parte, este dialogo contendrá la interfaz llamada RespuestaEleccionPokemon implemntada en la actividad e implementará la interfaz View.OnClickListener.

```java
@Override
public void onClick(View view) {
    if (view.getId() == R.id.buttonNO) respuestaDialogoJava.onRespuesta( pokemonElegido: null);
    if (view.getId() == R.id.buttonSI) respuestaDialogoJava.onRespuesta(this.pokemonElegido);
}

public interface RespuestaEleccionPokemon {
    void onRespuesta(Pokemon pokemonElegido);
}

public void onAttach(Context activity) {
    super.onAttach(activity);
    respuestaDialogoJava = (RespuestaEleccionPokemon) activity;
}
```

El layout es el siguiente:

Codigo:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondopokemon">

    <ImageView
        android:id="@+id/imageConfirmacion"
        android:layout_width="156dp"
        android:layout_height="252dp"
        app:layout_constraintBottom_toTopOf="@+id/linearLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.942"
        app:srcCompat="@drawable/charmander"
        android:contentDescription="@string/mi_pokemon" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:layout_marginStart="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="32dp"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <Button
            android:id="@+id/buttonSI"
            style="?android:attr/buttonBarButtonStyle"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:backgroundTint="#8BC34A"
            android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
            android:text="@string/Si"
            android:textColor="#FFFFFF"
            android:textSize="60sp"
            app:iconTint="#FFFFFF"
            app:rippleColor="#FFFFFF" />

        <Button
            android:id="@+id/buttonNO"
            style="?android:attr/buttonBarButtonStyle"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:backgroundTint="#FD0335"
            android:fontFamily="@font/pkmndp_peter_o_and_mr_gela"
            android:text="@string/No"
            android:textColor="#FFFFFF"
            android:textSize="58sp" />
    </LinearLayout>

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

4. Dialogo PonerMote.

Este será el segundo Dialogo de la actividad, llamaremos a un layout con un editText y si el usuario pulsa el botón sí se enviará lo que haya escrito y si pulsa no se devuelve null

```java
public class DialogoPonerMote extends DialogFragment {
    RespuestaDialogoMote respuestaDialogoJava;


    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gela);
        builder.setTitle(typeFaceStringMaker.build(getContext(), getString(R.string.ponerMote)));

        LinearLayout linearLayout = (LinearLayout) getLayoutInflater().inflate(R.layout.dialog_poner_mote,  root: null);
        EditText editTextMote = linearLayout.findViewById(R.id.editTextMote);

        builder.setView(linearLayout);

        builder.setPositiveButton(typeFaceStringMaker.build(getContext(),R.string.Si), new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {
                if (editTextMote.getText().equals("")) respuestaDialogoJava.onRespuestaMote(null);
                else respuestaDialogoJava.onRespuestaMote(String.valueOf(editTextMote.getText()));
            }
        });
        builder.setNegativeButton(typeFaceStringMaker.build(getContext(),R.string.No), new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {
                respuestaDialogoJava.onRespuestaMote(null);
            }
        });
        return builder.create();
    }

    public void onAttach(Context activity) {
        super.onAttach(activity);
        respuestaDialogoJava = (RespuestaDialogoMote) activity;
    }

    public interface RespuestaDialogoMote {
        void onRespuestaMote(String respuesta);

    }

}
```

5. Actividad BatallaPokemon.

Esta es la actividad principal de la app, en esta se desarrollará el combate pokemon. Tendrá varios atributos que nos permitirán usar ciertos objetos en varios metodos.

```
public class BatallaPokemon extends AppCompatActivity implements View.OnClickListener, CompoundButton.OnCheckedChangeListener {
    private ArrayList<SpannableString> listaMensajes = new ArrayList<>();
    private ArrayList<ArrayList<SpannableString>> listaRegistros = new ArrayList<>();
    private Pokemon_HPBar progressBarMiVida;
    private Pokemon_HPBar progressBarVidaEnemigo;
    private int delay;
    private MediaPlayer mediaPlayer;
    private RelativeAtaquePokemonLayoutButton buttonAtaque1;
    private RelativeAtaquePokemonLayoutButton buttonAtaque2;
    private RelativeAtaquePokemonLayoutButton buttonAtaque3;
    private RelativeAtaquePokemonLayoutButton buttonAtaque4;

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

        mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.musica_batalla);
        mediaPlayer.start();
        mediaPlayer.setLooping(true);

        listaRegistros = new ArrayList<>();

        Switch switchSpeed = findViewById(R.id.switchSpeed);
        Switch switchPower = findViewById(R.id.switchPower);
        switchPower.setOnCheckedChangeListener(this);
        switchSpeed.setOnCheckedChangeListener(this);

        Button buttonReset = findViewById(R.id.buttonReset);
        buttonReset.setOnClickListener(this);

        Button buttonRegistro = findViewById(R.id.buttonRegistro);
        buttonRegistro.setOnClickListener(this);

        updateMiPokemon();
        updatePokemonEnemigo();
    }
```

Como se puede apreciar al final llamamos a 2 metodos, updateMiPokemon y updatePokemonEnemigo, estos metodos son los encargados de cargar nuestro pokemon y el enemigo respectivamente, de estos metodos destacar que el pokemonEnemigo se genera aleatoriamente, que en updateMiPokemon generamos los botones de los ataques que tengamos y solo les ponemos listener si son distintos de null.

En las líneas de Glide (la librería encargada de los Gifs) usamos la opción diskCacheStrategy(DiskChacheStraregy.NONE) para evitar un bug en las animaciones de nuestros pokemon.

```
private void updatePokemonEnemigo() {
    int idPokemonRival = new Random().nextInt( bound: 4) + 1;
    Pokemon pokemonRival = Pokemon.buildPokemonbyId(idPokemonRival);
    pokemonRival.curarVida();
    ImageView gifPokemonRival = findViewById(R.id.pokemonRival);
    int imagenPokemonRival = pokemonRival.getImagen();
    Glide.with( activity: this).asGif().load(imagenPokemonRival).diskCacheStrategy(DiskCacheStrategy.NONE).into(gifPokemonRival);

    TextView textViewNombreEnemigo = findViewById(R.id.nombreEnemigo);
    setPokemonName(textViewNombreEnemigo, pokemonRival);

    this.progressBarVidaEnemigo = new Pokemon_HPBar(findViewById(R.id.vidaEnemigo), pokemonRival);
    this.progressBarVidaEnemigo.actualizarHPBar();
}
```

```
private void updateMiPokemon() {
    Pokemon miPokemon = (Pokemon) getIntent().getSerializableExtra( name: "pokemonElegido");
    miPokemon.curarVida();
    ImageView gifMiPokemon = findViewById(R.id.miPokemon);
    int imagenPokemon = miPokemon.getImagenEspalda();
    Glide.with( activity: this).asGif().load(imagenPokemon).diskCacheStrategy(DiskCacheStrategy.NONE).into(gifMiPokemon);

    TextView textViewMiNombre = findViewById(R.id.nombreMiPokemon);
    setPokemonName(textViewMiNombre, miPokemon);

    this.progressBarMiVida = new Pokemon_HPBar(findViewById(R.id.vidaMiPokemon), miPokemon);
    this.progressBarMiVida.actualizarHPBar();

    this.buttonAtaque1 = new RelativeAtaquePokemonLayoutButton( context: this, R.id.buttonAtaque1, miPokemon.getAtaquesPokemon()[0]);
    this.buttonAtaque2 = new RelativeAtaquePokemonLayoutButton( context: this, R.id.buttonAtaque2, miPokemon.getAtaquesPokemon()[1]);
    this.buttonAtaque3 = new RelativeAtaquePokemonLayoutButton( context: this, R.id.buttonAtaque3, miPokemon.getAtaquesPokemon()[2]);
    this.buttonAtaque4 = new RelativeAtaquePokemonLayoutButton( context: this, R.id.buttonAtaque4, miPokemon.getAtaquesPokemon()[3]);

    if (buttonAtaque1.getAtaquePokemon() != null) buttonAtaque1.setOnClickListener(this);
    if (buttonAtaque2.getAtaquePokemon() != null) buttonAtaque2.setOnClickListener(this);
    if (buttonAtaque3.getAtaquePokemon() != null) buttonAtaque3.setOnClickListener(this);
    if (buttonAtaque4.getAtaquePokemon() != null) buttonAtaque4.setOnClickListener(this);
}
```

Aparte del onCreate también hacemos uso del onPause para parar la música

```
@Override
protected void onPause() {
    super.onPause();
    mediaPlayer.stop();
    mediaPlayer.release();
}
```

Como se ha podido apreciar tenemos dos interfaces View.OnClickListener, CompoundButton.OnCheckedChangeListener por lo cual tendremos que implementar los metodos onClick y onCheckedChange respectivamente.

El método onCheckedChange solo se hará cargo de evitar que los dos switches estén pulsados a la vez

```
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    Switch switchSpeed = findViewById(R.id.switchSpeed);
    Switch switchPower = findViewById(R.id.switchPower);
    if (buttonView.getId() == switchSpeed.getId()) {
        if (isChecked) switchPower.setChecked(false);
    } else if (buttonView.getId() == switchPower.getId()) {
        if (isChecked) switchSpeed.setChecked(false);
    }
}
```

El método onClick será el método principal de nuestra app e iremos explicándolo poco a poco:

Al principio comprobaremos que si se ha pulsado el botón de registros y en caso afirmativo llamaremos al metodo verRegistro y saldremos del programa.

```
public void onClick(View v) {
    //
    if (v.getId() == R.id.buttonRegistro) {
        verRegistro();
        return;
    }
```

Después comprobaremos si no se ha pulsado el botón reset y en caso afirmativo llamaremos al metodo encargado de resetear la batalla

```
if (v.getId() == R.id.buttonReset) {
    resetBatalla(imagenMiPokemon, imagenPokemonRival);
    return;
}
```

Y como última comprobación nos aseguraremos de que los dos pokemon no estén debilitados para continuar con la ejecución del combate:

```
if (progressBarMiVida.isPokemonDebilitado() || progressBarVidaEnemigo.isPokemonDebilitado()
    return;
```

Ahora que ya nos hemos asegurado de que puede empezar/continuar el combate desactivaremos los listener de todos los botones para que el usuario no pueda hacer nada hasta que la ejecución haya terminado.

```
Button buttonReset = findViewById(R.id.buttonReset);
Button buttonRegistro = findViewById(R.id.buttonRegistro);
this.buttonAtaque1.setOnClickListener(null);
this.buttonAtaque2.setOnClickListener(null);
this.buttonAtaque3.setOnClickListener(null);
this.buttonAtaque4.setOnClickListener(null);
buttonReset.setOnClickListener(null);
buttonRegistro.setOnClickListener(null);
```

Después cogeremos el ataque del botón que se ha marcado y el pokemon rival usará un movimiento al azar

```
AtaquePokemon miAtaque;
if (v.getId() == R.id.buttonAtaque1) miAtaque = this.buttonAtaque1.getAtaquePokemon();
else if (v.getId() == R.id.buttonAtaque2) miAtaque = this.buttonAtaque2.getAtaquePokemon()
else if (v.getId() == R.id.buttonAtaque3) miAtaque = this.buttonAtaque3.getAtaquePokemon()
else if (v.getId() == R.id.buttonAtaque4) miAtaque = this.buttonAtaque4.getAtaquePokemon()
else return;

AtaquePokemon ataqueRival = this.progressBarVidaEnemigo.getPokemon().getRandomAttack();
```

Ahora haremos unos preparativos en los que crearemos los objetos Animación, guardaremos los nombres en variables y pondremos nuestro ataque como powered o speeded si así se ha marcado

```
Animacion animacionMiPokemon = new Animacion(imagenMiPokemon, imagenPokemonRival);
Animacion animacionRival = new Animacion(imagenMiPokemon, imagenPokemonRival);

TextView textViewMiNombre = findViewById(R.id.nombreMiPokemon);
TextView textViewNombreEnemigo = findViewById(R.id.nombreEnemigo);
String nombreMiPokemon = (String) textViewMiNombre.getText();
String nombrePokemonEnemigo = (String) textViewNombreEnemigo.getText();

Switch switchSpeed = findViewById(R.id.switchSpeed);
Switch switchPower = findViewById(R.id.switchPower);
boolean isSpeeded = switchSpeed.isChecked();
boolean isPowered = switchPower.isChecked();
miAtaque.setPowered(isPowered);
miAtaque.setSpeeded(isSpeeded);
```

Para que nuestro combate siga un orden y las animaciones se vean bien usaremos objetos de la clase Handler, clase que nos permite ejecutar un codigo pasado un tiempo (delay). En este caso tendremos una variable delay con la que iremos aumentando el tiempo según los ataques que sucedan. En caso de que el pokemon se debilite llamaremos al metodo finBatalla y pararemos la ejecución. Además en caso de que se termine el turno sin debilitar, activaremos de nuevo todos los listeners.

```
this.delay = 0;
int aumentoDelay = 750;

//Podemos fallar en un 20% de ocasiones el ataque rapido
if (isSpeeded && Math.random() < 0.8d) {
    Turno turnoJugadorSpeed = new Turno(progressBarVidaEnemigo, miAtaque, animacionMiPokemon, nombreMiPokemon,  isAttacking: true);
    listaMensajes.add(turnoJugadorSpeed.procesarTurno());
    this.delay += aumentoDelay;
    if (progressBarVidaEnemigo.isPokemonDebilitado()) {
        finBatalla( isMiPokemon: false, imagenMiPokemon, imagenPokemonRival);
        return;
    }
}

new Handler().postDelayed(() -> {
    Turno turnoJugador = new Turno(progressBarVidaEnemigo, miAtaque, animacionMiPokemon, nombreMiPokemon,  isAttacking: true);
    listaMensajes.add(turnoJugador.procesarTurno());
    if (this.progressBarVidaEnemigo.isPokemonDebilitado()) {
        finBatalla( isMiPokemon: false, imagenMiPokemon, imagenPokemonRival);
        return;
    }
    if (isPowered) delay += 500;
    this.delay += aumentoDelay;
    new Handler().postDelayed(() -> {
        Turno turnoEnemigo = new Turno(progressBarMiVida, ataqueRival, animacionRival, nombrePokemonEnemigo,  isAttacking: false);
        listaMensajes.add(turnoEnemigo.procesarTurno());
        if (this.progressBarMiVida.isPokemonDebilitado()) {
            finBatalla( isMiPokemon: true, imagenMiPokemon, imagenPokemonRival);
            return;
        }

        //Hay un 60% de que nos ataquen por segunda vez
        if (isPowered && Math.random() < 0.6d) {
            new Handler().postDelayed(() -> {
                Turno turnoEnemigo2 = new Turno(progressBarMiVida, ataqueRival, animacionRival, nombrePokemonEnemigo,  isAttacking: false);
                listaMensajes.add(turnoEnemigo2.procesarTurno());
                if (this.progressBarMiVida.isPokemonDebilitado()) {
                    finBatalla( isMiPokemon: true, imagenMiPokemon, imagenPokemonRival);
                    return;
                }

                //Hay un 60% de que nos ataquen por segunda vez
                if (isPowered && Math.random() < 0.6d) {
                    new Handler().postDelayed(() -> {
                        Turno turnoEnemigo2 = new Turno(progressBarMiVida, ataqueRival, animacionRival, nombrePokemonEnemigo,  isAttacking: false);
                        listaMensajes.add(turnoEnemigo2.procesarTurno());
                        if (this.progressBarMiVida.isPokemonDebilitado()) {
                            finBatalla( isMiPokemon: true, imagenMiPokemon, imagenPokemonRival);
                            return;
                        }
                    }, this.delay);
                }

                this.delay += 200;
                new Handler().postDelayed(() -> {
                    if (this.buttonAtaque1.getAtaquePokemon() != null)
                        this.buttonAtaque1.setOnClickListener(this);
                    if (this.buttonAtaque2.getAtaquePokemon() != null)
                        this.buttonAtaque2.setOnClickListener(this);
                    if (this.buttonAtaque3.getAtaquePokemon() != null)
                        this.buttonAtaque3.setOnClickListener(this);
                    if (this.buttonAtaque4.getAtaquePokemon() != null)
                        this.buttonAtaque4.setOnClickListener(this);
                    buttonReset.setOnClickListener(this);
                    buttonRegistro.setOnClickListener(this);
                }, this.delay);
            }, this.delay);
        }, this.delay);
    }, this.delay);
}
```

Ahora se explicarán los métodos nombrados anteriormente:

- **verRegistro**. Nos muestra el dialogo VerRegistro donde podremos ver todos los combates que hemos librado.

```
private void verRegistro() {
    DialogoVerRegistro dialogoVerRegistro = new DialogoVerRegistro(this.listaRegistros);
    dialogoVerRegistro.show(getSupportFragmentManager(), tag: "Poner Mote");
}
```

- **resetBatalla**. Método llamado al pulsar el botón reset y nos comenzará un nuevo combate.

```
private void resetBatalla(View imagenMiPokemon, View imagenPokemonRival) {
    updateMiPokemon();
    updatePokemonEnemigo();

    TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gelo);
    this.listaMensajes.add(typeFaceStringMaker.build(getApplicationContext(), getString(R.string.madeReset)));

    if (listaMensajes.size() < 1) this.listaRegistros.add(this.listaMensajes);
    this.listaMensajes = new ArrayList<>();

    Animacion animacionAparecerPokemon = new Animacion(imagenMiPokemon, imagenPokemonRival);
    animacionAparecerPokemon.animacionAparicion(getApplicationContext());
}
```

- **finBatalla**. Método llamado cuando un pokemon se debilita.

```
private void finBatalla(boolean isMiPokemon, View imagenMiPokemon, View imagenPokemonRival) {

    this.delay = 2000;
    new Handler().postDelayed(() -> {
        Button buttonReset = findViewById(R.id.buttonReset);
        buttonReset.setOnClickListener(this);
        Button buttonRegistro = findViewById(R.id.buttonRegistro);
        buttonRegistro.setOnClickListener(this);
    }, this.delay);

    TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gelo);
    this.listaMensajes.add(typeFaceStringMaker.build(getApplicationContext(), getString(R.string.finBatalla)));
    this.listaRegistros.add(this.listaMensajes);
    this.listaMensajes = new ArrayList<>();

    new Animacion(imagenMiPokemon, imagenPokemonRival).animacionMuerte(isMiPokemon, getApplicationContext());
}
```

6. Dialogo VerRegistro.

Nos permite crear un layout en un dialogo para que podamos ver que ha sucedido en todos nuestros combates terminados.

El registro se pasará por el constructor y será una lista de listas de objetos de la clase SpannableString. Ya que cada lista de objetos de SpannableString son los mensajes generados en un combate en la actividad anterior, teniendo así los mensajes separados en una lista por combate.

```
public class DialogoVerRegistro extends DialogFragment {
    ArrayList<ArrayList<SpannableString>> listaRegistros;

    public DialogoVerRegistro(ArrayList<ArrayList<SpannableString>> listaRegistros) {
        super();
        this.listaRegistros = listaRegistros;
    }
```

Nos apoyaremos de dos metodos para crear este registro:

- buildTypefaceTextView. Nos permite crear un TextView con un SpannableString.

```
private View buildTypefaceTextView(SpannableString mensaje, int color, int textSize) {
    TextView textView = new TextView(getContext());
    textView.setText(mensaje);
    textView.setTextSize(textSize);
    textView.setTextColor(color);
    return textView;
}
```

- buildLayoutSeparacion. Nos permite crear una pequeña separación visual entre los registros de los combates.

```
private LinearLayout buildLayoutSeparacion(TypeFaceStringMaker typeFaceStringMaker) {
    LinearLayout linearLayoutSeparacion = new LinearLayout(getContext());
    linearLayoutSeparacion.setOrientation(LinearLayout.VERTICAL);
    linearLayoutSeparacion.setBackgroundColor(getResources().getColor(R.color.darkBackground));
    linearLayoutSeparacion.addView(buildTypefaceTextView(typeFaceStringMaker.build(getContext(), chars: " "),
            getResources().getColor(R.color.textoRegistro),
            textSize: 24));
    return linearLayoutSeparacion;
}
```

Todos los TextView Generados los meteremos en distintos layouts y estos a su vez irán en una ScrollView que nos permitirá ver todos los mensajes aún cuando se salgan de la pantalla:

```
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    ScrollView scrollView = new ScrollView(getContext());
    LinearLayout linearLayoutGeneral = new LinearLayout(getContext());
    linearLayoutGeneral.setOrientation(LinearLayout.VERTICAL);
    TypeFaceStringMaker typeFaceStringMaker = new TypeFaceStringMaker(R.font.pkmndp_peter_o_and_mr_gela);

    linearLayoutGeneral.addView(buildLayoutSeparacion(typeFaceStringMaker));
    int numeroCombate = 1;
    for (ArrayList<SpannableString> registro : listaRegistros) {
        LinearLayout linearLayoutRegistro = new LinearLayout(getContext());
        linearLayoutRegistro.setOrientation(LinearLayout.VERTICAL);
        linearLayoutRegistro.setBackgroundColor(getResources().getColor(R.color.lightBackground));
        linearLayoutRegistro.addView(buildTypefaceTextView(typeFaceStringMaker.build(getContext(),  chars: getText(R.string.registro) + " " + numeroCombate),
                getResources().getColor(R.color.textoTitulo),
                 textSize: 24));
        for (SpannableString mensaje : registro) {
            linearLayoutRegistro.addView(buildTypefaceTextView(mensaje,
                    getResources().getColor(R.color.textoRegistro),
                     textSize: 18));
        }
        linearLayoutGeneral.addView(linearLayoutRegistro);
        linearLayoutGeneral.addView(buildLayoutSeparacion(typeFaceStringMaker));
        numeroCombate++;
    }

    scrollView.addView(linearLayoutGeneral);

    builder.setView(scrollView);
    builder.setPositiveButton(typeFaceStringMaker.build(getContext(), "Salir"), new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialogInterface, int i) { dismiss(); }
    });

    return builder.create();
}
```