

Bottom Navigation / RecyclerView / Retrofit

PRIMERA PARTE BOTTOM NAVIGATION:

1. Crear una aplicación con un **Activity** en Blanco, colocan el nombre que deseen al proyecto.
2. Nos vamos al módulo **build.gradle** y agregamos o implementamos los siguientes paquetes

```
dependencies {  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.8.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    // For control over item selection of both touch and mouse driven selection  
    implementation 'androidx.recyclerview:recyclerview:1.1.0'  
    implementation 'androidx.recyclerview:recyclerview-selection:1.1.0'  
    implementation 'androidx.cardview:cardview:1.0.0'  
    //Consumir Apis  
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
    implementation 'com.squareup.okhttp3:logging-interceptor:3.5.0'  
    implementation 'com.google.code.gson:gson:2.8.6'  
    //Barra  
    implementation 'com.facebook.shimmer:shimmer:0.5.0'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
}
```

<https://square.github.io/retrofit/>

<https://facebook.github.io/shimmer-android/>

Coloco las dependencias:

```
// For control over item selection of both touch and mouse driven  
selection  
implementation 'androidx.recyclerview:recyclerview:1.1.0'  
implementation 'androidx.recyclerview:recyclerview-selection:1.1.0'  
implementation 'androidx.cardview:cardview:1.0.0'  
//Consumir Apis  
implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
implementation 'com.squareup.okhttp3:logging-interceptor:3.5.0'  
implementation 'com.google.code.gson:gson:2.8.6'  
//Barra  
implementation 'com.facebook.shimmer:shimmer:0.5.0'
```

3. El layout del Main_Activity lo cambiamos a un **RelativeLayout**

```
<RelativeLayout xmlns:android="http://schemas  
    xmlns:app="http://schemas.android.com/apk  
    xmlns:tools="http://schemas.android.com/t  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">
```

4. Agregamos un `FrameLayout`, que es donde se renderizará el contenido, saldrá error en el `layout_above` porque el objeto no se ha creado.

```
<FrameLayout
    android:id="@+id/frame_container"
    android:layout_width="match_parent"
    android:layout_height="715dp"
    android:layout_above="@+id/bottom_navigation"
    android:layout_alignParentTop="true"
    android:layout_marginBottom="4dp" />
```

5. Ahora agregamos el `bottom navigation`, debajo del `FrameLayout`, fijarse muy bien el los `@id` que se colocan, el `style` dará error al igual menú porque no se han creado.






```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:elevation="18dp"
    app:labelVisibilityMode="selected"
    style="@style/BottomNavigation"
    app:menu="@menu/bottom_navigation" />
```

Estilo del Bottom

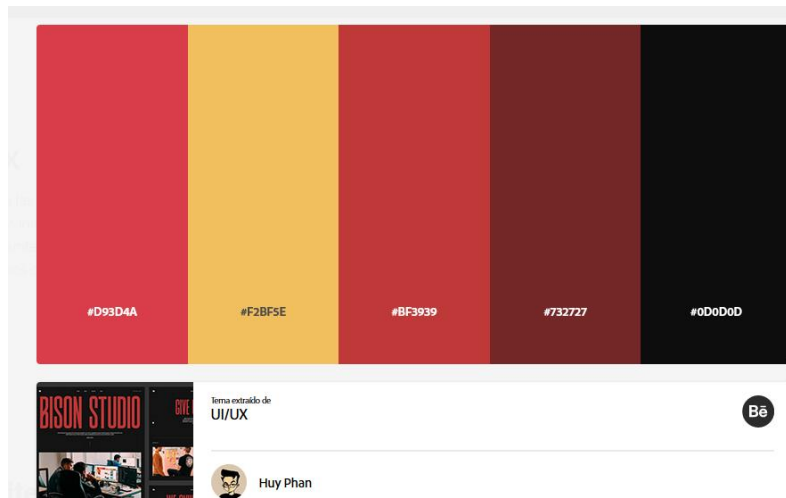
Nombre del directorio

Nombre del recurso

6. Nos vamos a la carpeta `color` y definimos una paleta de colores que utilizaremos en el estilo:

	<code><color name="pink01d">#D93D4A</color></code>
	<code><color name="yellow01d">#F2BF5E</color></code>
	<code><color name="red01d_">#BF3939</color></code>
	<code><color name="redWine">#732727</color></code>
	<code><color name="blackWine">#0D0D0D</color></code>

Tomada de Adobe: <https://color.adobe.com/es/trends>



7. Ahora nos vamos themes(carpetas res->values), están resaltados los cambios que vamos a hacer, el nuevo style tiene el mismo nombre que declaramos en el punto 5 "BottomNavigation"

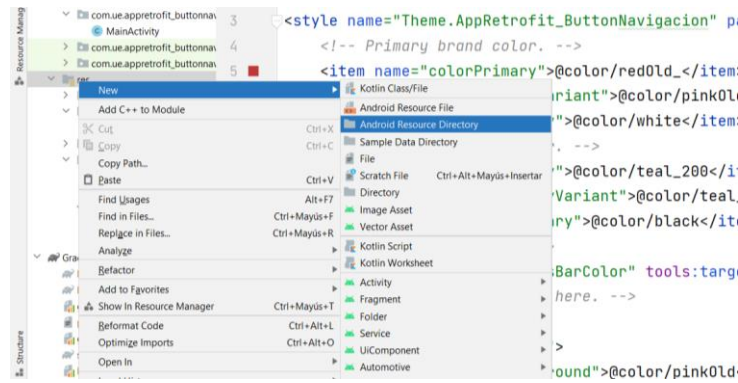
```
<style name="Theme.AppRetrofit_ButtonNavegacion" parent="Theme.MaterialComponents.DayNight.NoActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/red01d</item>
    <item name="colorPrimaryVariant">@color/pink01d</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    <!-- Status bar color. -->
    <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
</style>

<style name="BottomNavigation">
    <item name="android:background">@color/pink01d</item>
    <item name="itemIconTint">@color/yellow01d</item>
    <item name="itemTextColor">@color/blackWine</item>
</style>
```

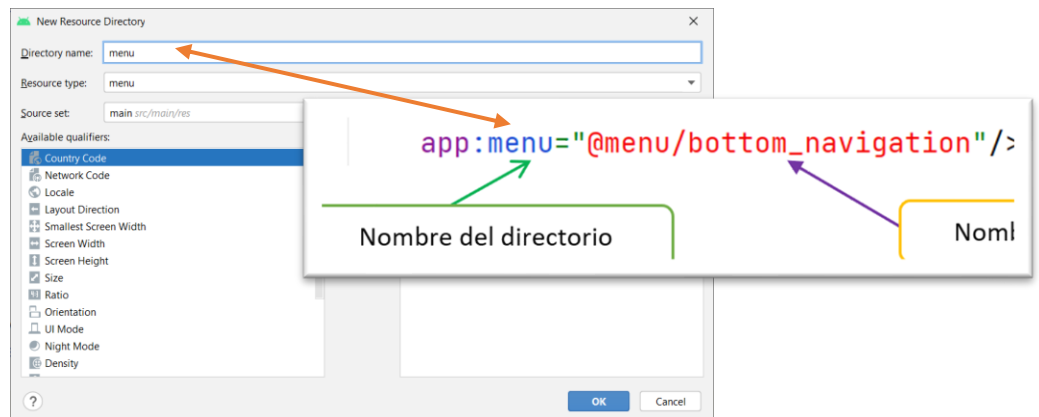
style="@style/BottomNavigation"
app:menu="@menu/bottom_navigation"/>

Estilo del Bottom

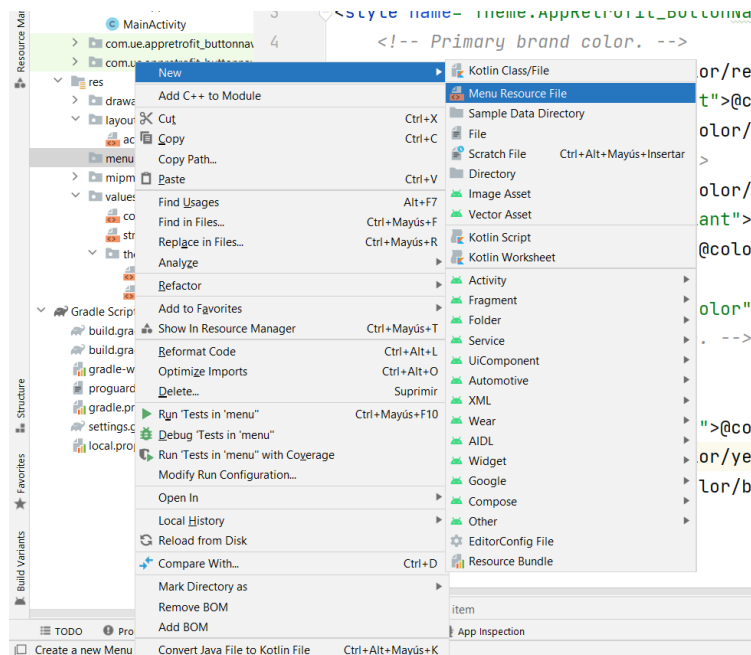
8. Ahora creamos un nuevo directorio (Android Resource Directory), en la carpeta "res"



9. Le decimos que es de tipo menú y colocamos un nombre: “**menu**”



10. Una vez creada la carpeta, nos paramos sobre ella, presionamos botón derecho y creamos un **Menu Resource File**



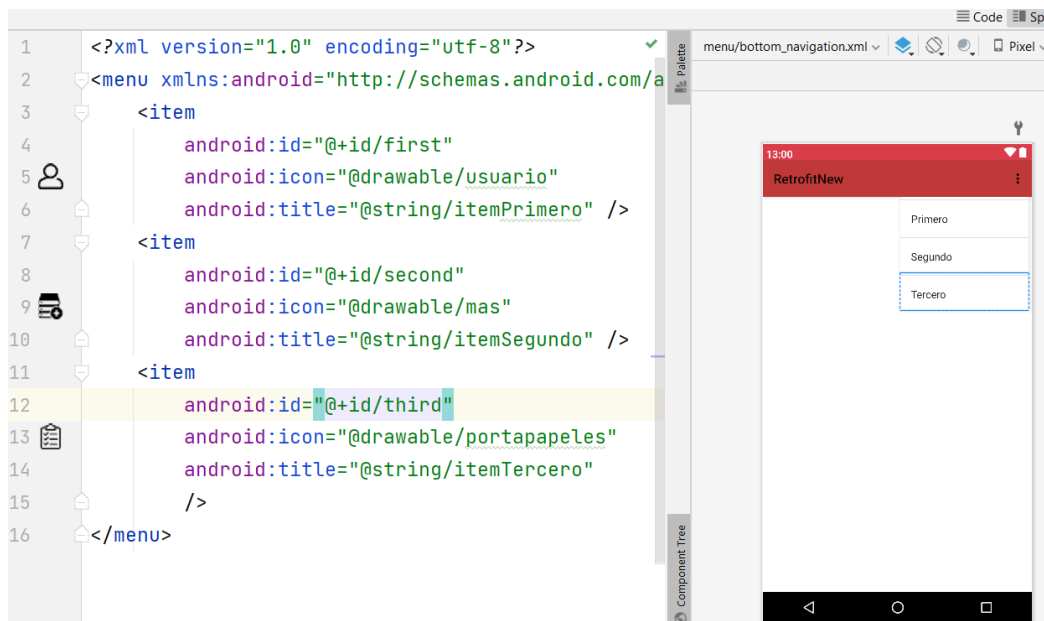
11. Colocamos en el apartado de nombre el nombre que colocamos en el punto 5:



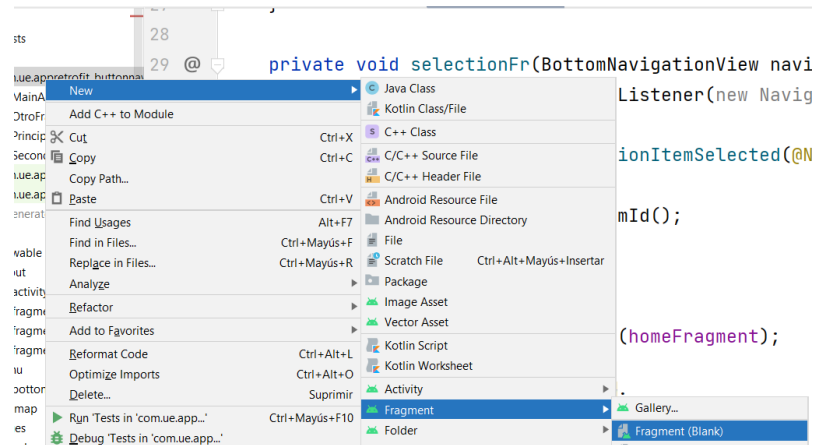
12. Nos vamos al archivo de string.xml y agregamos las etiquetas para los elementos del menú

```
<string name="itemPrimero">Primero</string>  
<string name="itemSegundo">Segundo</string>  
<string name="itemTercero">Tercero</string>
```

13. Al crear el archivo, configuramos el Bottom Navigation, fijarse en el id de cada item, que se utilizará para la acción que queremos asociar. También crear cada texto en el archivo string.xml y los íconos los pueden descargar de flaticon y agregar en la carpeta drawable



14. Vamos a crear por el momento 3 Fragments, hacemos click derecho sobre el paquete principal: colocar los nombres que deseen, yo los llamé: PrincipalFragment, SecondFragment y OtroFragment:



15. Nos vamos al MainActivity.java, vamos a crear una instancia por cada fragment creado:

```
public class MainActivity extends AppCompatActivity {  
  
    PrincipalFragment homeFragment = new PrincipalFragment();  
    SecondFragment secondFragment = new SecondFragment();  
    OtroFragment otroFragment = new OtroFragment();  

```

16. Ahora se crea un método para la carga del fragment que se pase por parámetro:

```
private void loadFragment(Fragment fr){  
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
    transaction.replace(R.id.frameContainer, fr);  
    transaction.commit();  
}
```

17. Creamos el método, según la opción seleccionada cargamos el fragment:

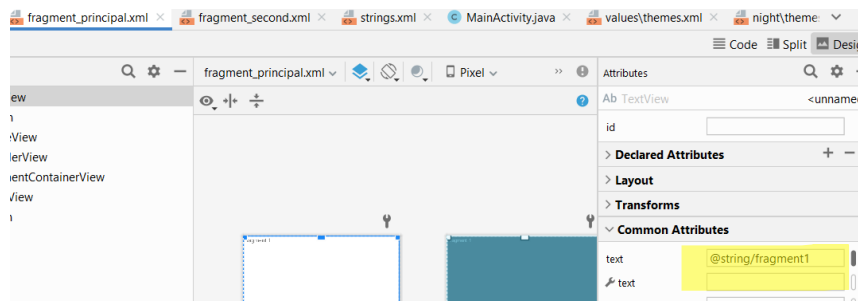
```
private void selectionFr(BottomNavigationView navigation){  
    navigation.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {  
        @Override  
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
            int id = item.getItemId();  
            switch(id){  
                //check id  
                case R.id.first:  
                    loadFragment(homeFragment);  
                    return true;  
                case R.id.second:  
                    loadFragment(secondFragment);  
                    return true;  
                case R.id.third:  
                    loadFragment(otroFragment);  
                    return true;  
            }  
            return true;  
        }  
    });  
}
```

18. Nos situamos en el método onCreate(MainActivity), mapeamos el Bottom Navigation, lo mandamos al método selectionFr y cargamos el fragment Principal

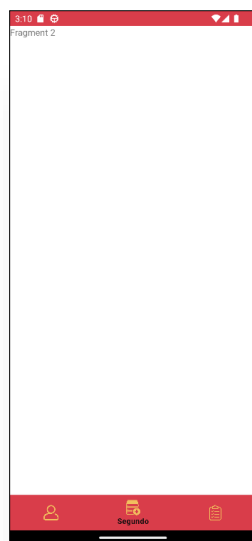
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    BottomNavigationView navigationView = findViewById(R.id.bottomNaviMenu);
    selectionFr(navigationView);
    loadFragment(homeFragment);
}
```

19. Para diferenciar cada fragment y ver el contenido, creamos 3 etiquetas en el string.xml y asociar en cada TextView de los fragments

```
<string name="fragment1">Fragment 1</string>
<string name="fragment2">Fragment 2</string>
<string name="fragment3">Fragment 3</string>
```



Resultado:

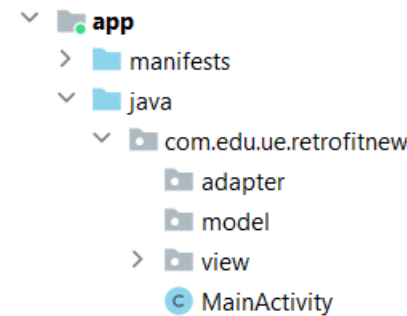


SEGUNDA PARTE Card View y Adaptador:

1. Colocar el permiso para salida de Internet

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ue.appretrofit_buttonnavigacion">
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

2. Creamos la siguiente estructura para trabajar con el patrón Maestro-Detalle



3. Vamos a utilizar el API de Pokémon <https://pokeapi.co/api/v2/>. Dentro de **model** creamos la clase genérica llamada Pokemon:

```
public class Pokemon {
    private String name;
    private String url;

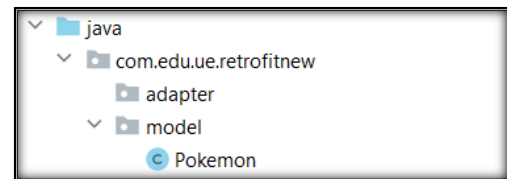
    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getUrl() { return url; }

    public void setUrl(String url) { this.url = url; }

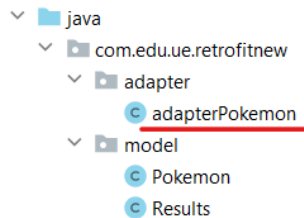
    @Override
    public String toString() {
        return "Pokemon{" +
            "name='" + name + '\'' +
            ", url='" + url + '\'' +
            '}';
    }
}
```



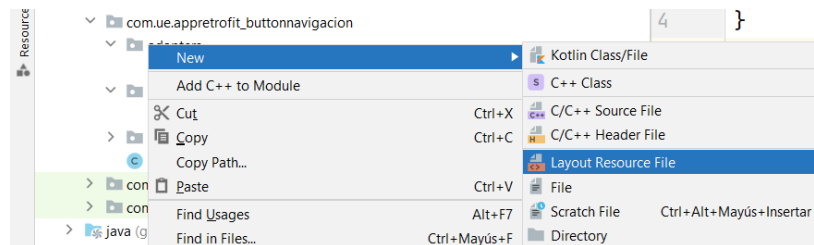
4. Como solo queremos el arreglo de objetos Results, creamos una clase Results para formar un ArrayList:

```
public class Results {  
    private ArrayList<Pokemon> results;  
  
    public ArrayList<Pokemon> getResults() { return results; }  
  
    public void setResults(ArrayList<Pokemon> results) { this.results = results; }  
}
```

5. Creamos otra clase dentro el paquete adapter, llamada AdapterPokemon



6. Vamos a crear y diseñar un xml para cada uno de los ítems que voy a mostrar en los Card. Botón derecho del ratón, sobre el paquete **layout**-> **New** -> **Layout Resource File**. Colocamos el nombre de **list_pokemons**



7. Dentro de este archivo, vamos a diseñar nuestro card

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:animateLayoutChanges="true"
        android:padding="4dp"
        app:cardCornerRadius="10dp"
        app:cardElevation="3dp"
        app:cardUseCompatPadding="true"
        app:contentPadding="1dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="2dp"
            android:orientation="vertical">
            <TextView
                android:id="@+id/tvName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/tvName" />
            <TextView
                android:id="@+id/tvUrl"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/tvUrl" />
            </LinearLayout>
        </androidx.cardview.widget.CardView>
    </RelativeLayout>

```

8. Ahora vamos al adaptador, vamos a extendernos de RecyclerView.Adapter<AdapterPokemon.vh>, esto nos va a generar un error, porque debemos implementar los métodos, damos click en el bombillito rojo, quien nos dirá que debemos implementar los métodos:

```

import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class AdapterPokemon extends RecyclerView.Adapter<AdapterPokemon.vh> {

}

```

```

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class AdapterPokemon extends RecyclerView.Adapter<AdapterPokemon.vh> {
    Click or press Alt+Intro

```

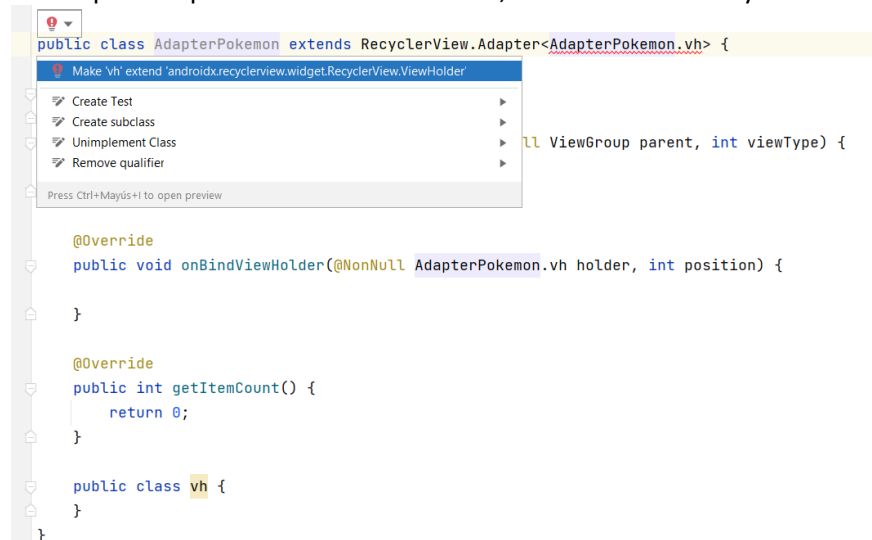
Select Methods to Implement

- androidx.recyclerview.widget.RecyclerView.Adapter
- onCreateViewHolder(parent:ViewGroup, viewType:int):VH
- onBindViewHolder(holder:VH, position:int):void
- getItemCount():int

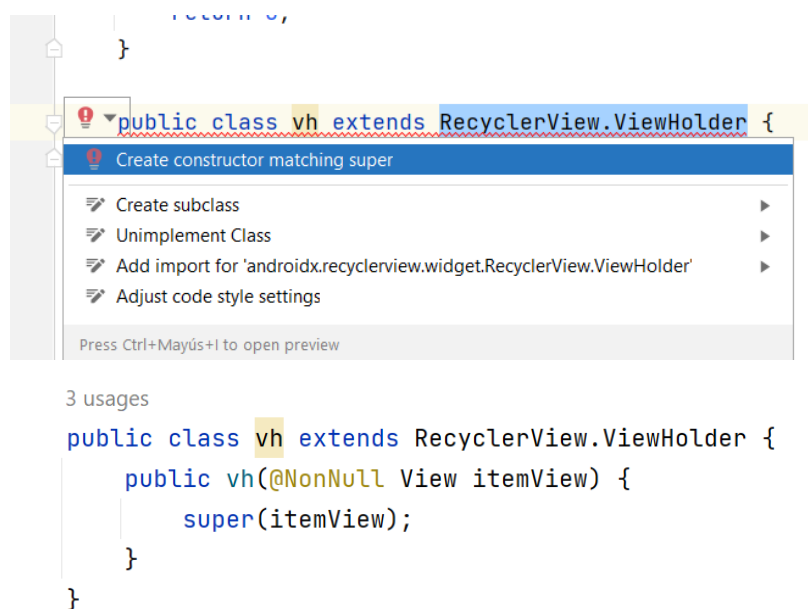
9. Todavía nos sigue apareciendo error, seleccionamos crear la class 'vh'



10. Ahora nos aparece que debemos extendernos, en esa clase de RecyclerView.ViewHolder



11. Y para terminar de implementar lo requerido para el adapter, debemos crear el constructor para invocar al super:



12. Dentro de AdapterPokemon, vamos a crear un list para los pokemones y un constructor:

```
List<Pokemon> pokemonList;

public AdapterPokemon(List<Pokemon> pokemonList){this.pokemonList = pokemonList;}
```

13. Sobre-escribimos el método onCreateViewHolder, que permitirá inflar el layout que será el list_pokemons.xml

```
@NonNull
@Override
public AdapterPokemon.vh onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_pokemons, root: null, attachToRoot: false);
    return new vh(view);
}
```

14. Ahora vamos a la clase vh que está al final y creamos 2 TextView para el nombre del pokemon y la url

```
4 usages
public class vh extends RecyclerView.ViewHolder {
    2 usages
    TextView txtName;
    2 usages
    TextView txtUrl;
    1 usage
    public vh(@NonNull View itemView) {
        super(itemView);
        txtName = itemView.findViewById(R.id.tvName);
        txtUrl = itemView.findViewById(R.id.tvUrl);
    }
}
```

15. Sobre-escribimos el getItemCount, que retornara la cantidad de elementos de la lista

```
@Override
public int getItemCount() {
    return pokemonList.size();
}
```

16. Para terminar con el adapter, sobre-escribimos el onBindViewHolder:

```
@Override
public void onBindViewHolder(@NonNull AdapterPokemon.vh holder, int position) {
    holder.txtName.setText(pokemonList.get(position).getName());
    holder.txtUrl.setText(pokemonList.get(position).getUrl());
}
```

17. Así debería quedar la clase AdapterPokemon

```
public class AdapterPokemon extends RecyclerView.Adapter<AdapterPokemon.vh> {  
    4 usages  
    List<Pokemon> pokemonList;  
  
    public AdapterPokemon(List<Pokemon> pokemonList){this.pokemonList = pokemonList;}  
  
    @NonNull  
    @Override  
    public AdapterPokemon.vh onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_pokemons, root: null, attachToRoot: false);  
  
        return new vh(view);  
    }  
  
    @Override  
    public void onBindViewHolder(@NonNull AdapterPokemon.vh holder, int position) {  
        holder.txtName.setText(pokemonList.get(position).getName());  
        holder.txtUrl.setText(pokemonList.get(position).getUrl());  
    }  
  
    @Override  
    public int getItemCount() {  
        return pokemonList.size();  
    }  
  
    4 usages  
    public class vh extends RecyclerView.ViewHolder {  
        2 usages  
        TextView txtName;  
        2 usages  
        TextView txtUrl;  
        1 usage  
        public vh(@NonNull View itemView) {  
            super(itemView);  
            txtName = itemView.findViewById(R.id.tvName);  
            txtUrl = itemView.findViewById(R.id.tvUrl);  
        }  
    }  
}
```