INDICE

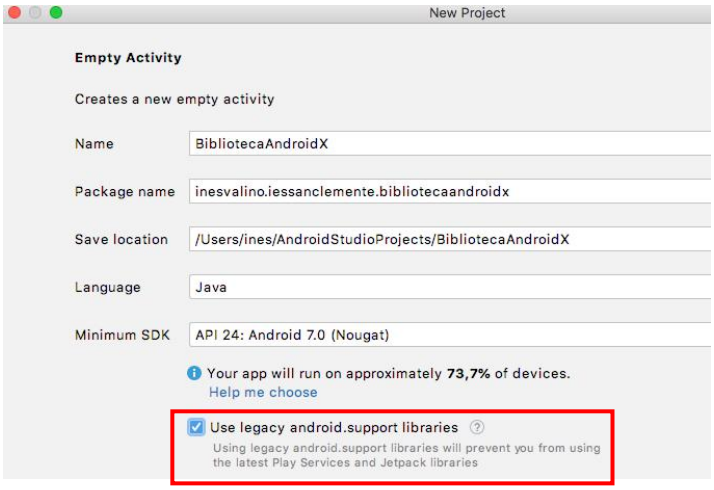
1. Introducción

- Las bibliotecas de compatibilidad son creadas por Android para admitir versiones anteriores del sistema operativo Android y que estas puedan ejecutar nuevas funciones que aparecieron en versiones posteriores.
- A partir de la versión 9.0 de Android (API 28), hay otra forma de implementar dichas bibliotecas, llamadas 'artefactos' y se encuentra dentro del **espacio de nombres de** androidx.

1.1.Versión actual

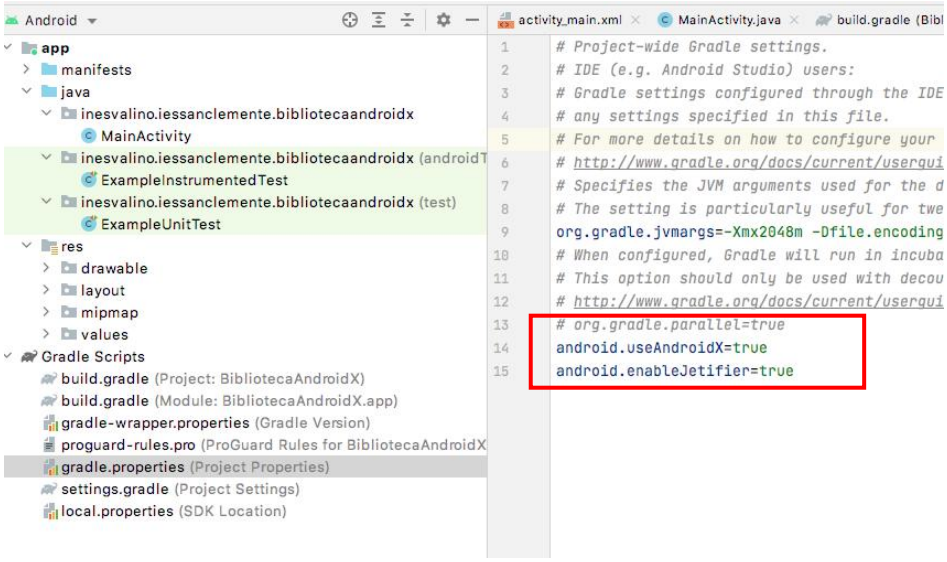
- Todas las bibliotecas de compatibilidad están alojadas en un paquete que comienza con el nombre **androidx**.
- A diferencia de las bibliotecas de compatibilidad utilizadas anteriormente, estas se mantienen y actualizan por separado y no es necesario actualizar todo el sistema operativo como era el caso antes con las bibliotecas de **compatibilidad android.compact.**

Cómo emplear el nuevo sistema de biblioteca de compatibilidad usando android



Cuando creamos un nuevo proyecto debemos marcar la opción use **android.support libraries**. Para que esto aparezca debemos tener al menos API 29 o posterior instalada.

Debemos configurar dos marcas de complementos de Gradle para Android con el valor TRUE en el archivo **gradle.properties**: *android.useAndroidX* y *android.enableJetifier.*



También se debe comprobar que existe un repositorio en el archivo **build.gradle** (**a nivel de Proyecto**) con el nombre *google ()* (esto lo tendríamos que hacer manualmente si no configuramos el proyecto desde el principio para hacer uso de estas bibliotecas).

* Podéis consultar en este enlace la nomenclatura de las versiones utilizadas y qué significa cada número.

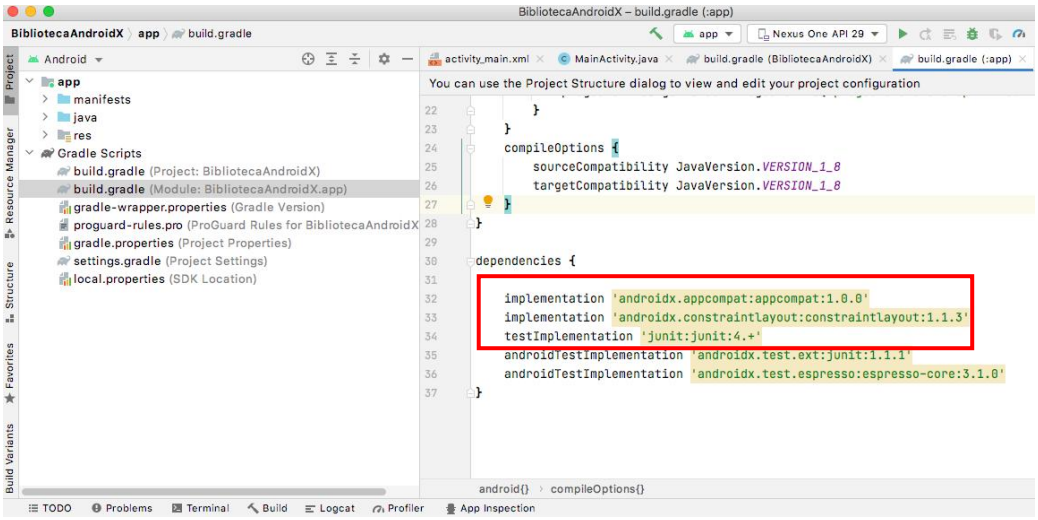
Para migrar un proyecto existente (con Android Studio 3.2 o versión posterior) para usar AndroidX basta con seleccionar **Refactor->Migrate to AndroidX** en la barra de menú. Si hacéis así quizá os dé avisos de problemas en la clase MainActivity.java, simplemente es porque ahora el import de appcompactivity es:

```
import androidx.appcompat.app.AppCompatActivity;
```

1.1.1. Ejemplo de uso

- Puede encontrar en este enlace la lista de bibliotecas con las versiones actuales que se encuentran dentro del espacio de nombres *androidx*.

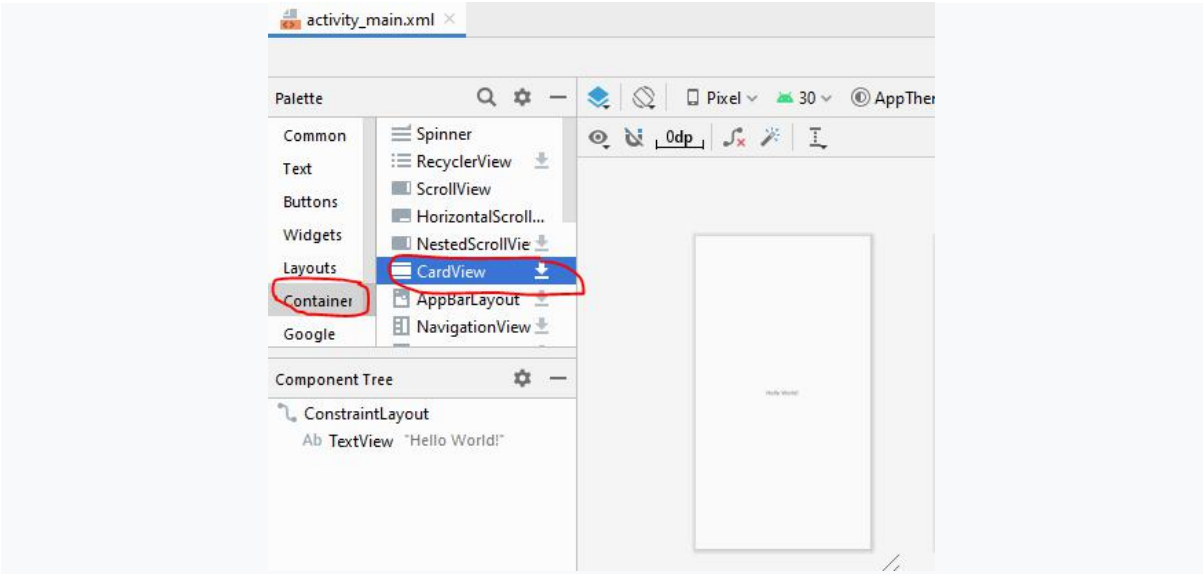
  Para agregar un nuevo recurso tenemos que editar el archivo **build.gradle** a nivel de Módulo y en la sección *'dependences'* debemos usar la orden **implementation**:



Como vemos, por defecto hace uso de dos librerías, una para colocar los elementos gráficos en las actividades (un tipo de layout llamado *ConstraintLayout*) y la librería appcompat, que entre otros elementos incorpora la *ToolBar*.

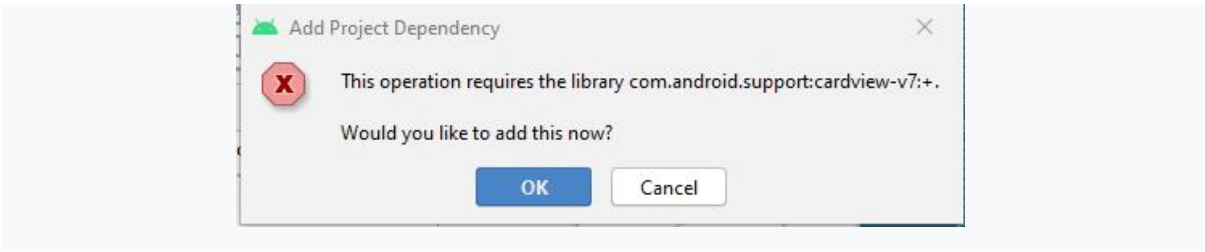
Ejemplo de uso empleando biblioteca androidx

- Imagina que en nuestro proyecto queremos usar el widget (control gráfico) CARDVIEW.



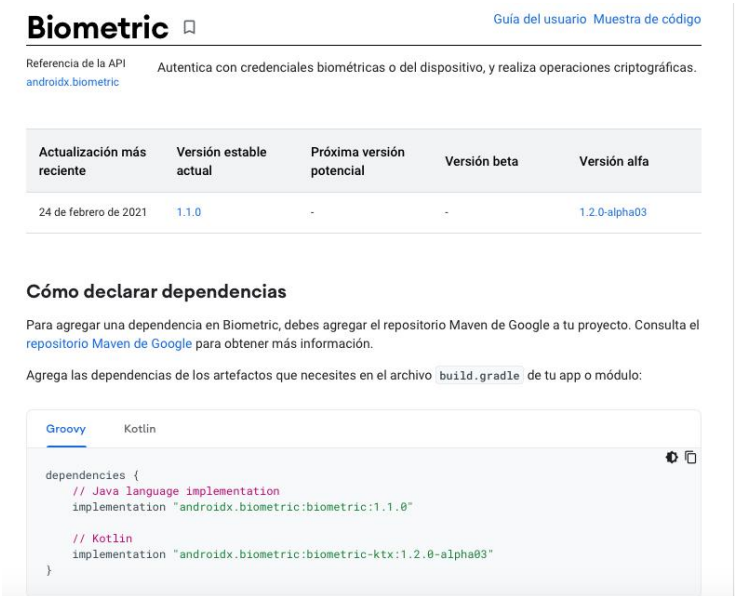
Vayamos al diseño gráfico de la Activity. Podemos ver como dicho control no está en el proyecto y es necesario descargarlo (puede tardar un poquito).

Lo arrastramos al diseño.

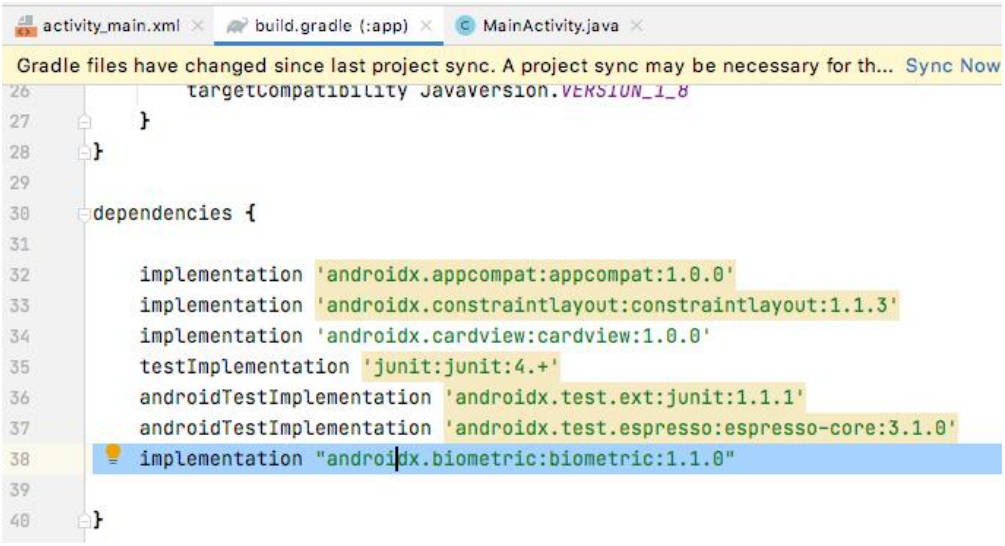


Al hacerlo, nos indicará que es necesario utilizar la biblioteca **androidx.cardview**. Presionamos el botón OK y comienza a hacer una sincronización de Gradle descargando la biblioteca. Esto se debe a que lo que realmente está haciendo es agregar una nueva línea al archivo **build.gradle**. Podemos comprobar cómo se agrega la línea de implementación con la nueva biblioteca.

Imaginamos que queremos agregar una biblioteca a nuestro proyecto que nos permita autenticar con credenciales biométricas. Explorarmos en la biblioteca de jetpack (enlace) y vemos que se llama *Biometric* y que la versión estable actual es 1.1.0.



Como se observa indica la dependencia que hay añadir (dependiendo del lenguaje de programación que se esté usando)



Escribimos la línea "**implementation**" con la biblioteca que queremos y hacemos clic en el enlace **Sync Now**.

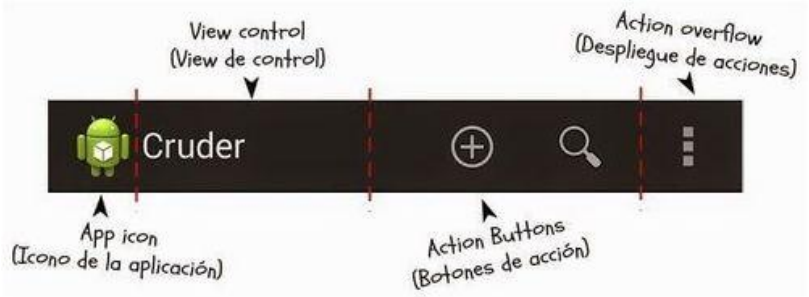
1.2.En versión antigua

- Las bibliotecas comienzan con el nombre: **com.android.support**

  Solo se actualizarán hasta la versión 28, a partir de entonces solo se dará soporte a las bibliotecas de espacio de nombres **androidx**.
- Se puede consultar las mejoras a nivel general que ofrecen dichas librerías de compatibilidad en este enlace
- Se puede consultar las funciones que incorporan cada una de las bibliotecas de compatibilidad en este enlace
- Se puede consultar como configurar Android Studio para descargar y usar estas bibliotecas en este enlace

1.2.1. Ejemplo de uso

- Un ejemplo de uso sería el uso de ActionBar:

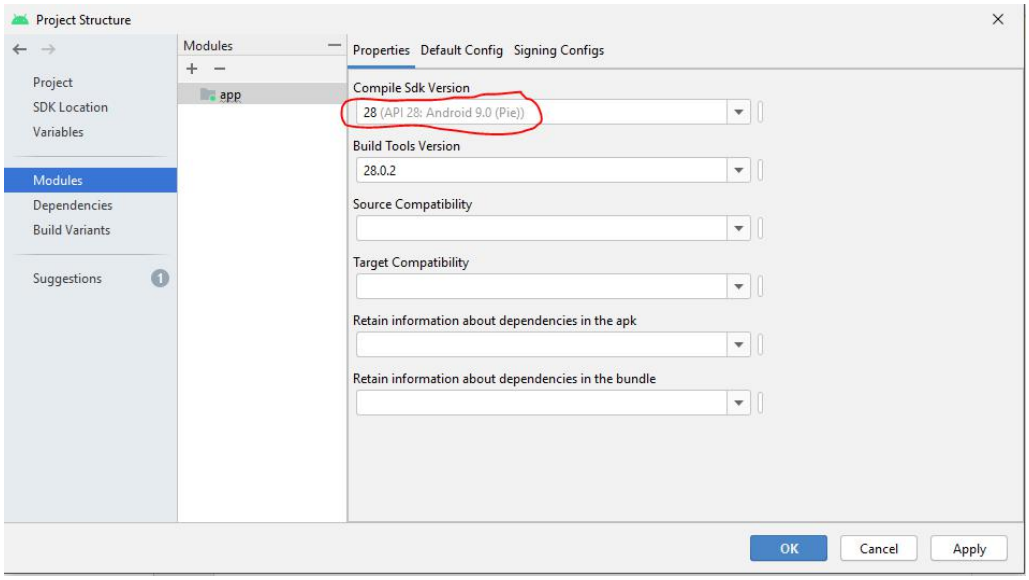


- Desde la versión 3.0 (API 11) todas las Activity's pueden llevar un ActionBar (de hecho, viene en el Theme por defecto). Pero a medida que han surgido nuevas versiones de la API, se han agregado nuevas funciones.

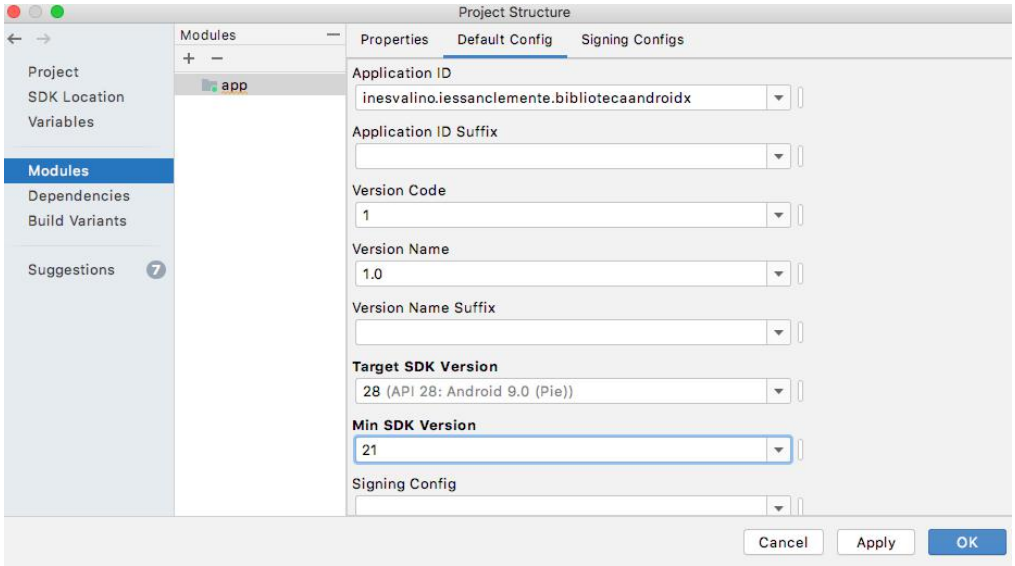
  Por ejemplo, a partir de API versión 21 (Android 5) se han añadido nuevas opciones para modificar el diseño y adaptarlo a la filosofía de Material Design. De hecho, la ActionBar pasó a llamarse ToolBar.

  Una de estas nuevas propiedades es la elevación, que se convierte en un efecto como si el componente se hubiera colocado "encima" de otro. En el caso de la barra de acción (Action Bar) de Android 5 se llama **ToolBar** (AppBar consta de Tool Bar, Status Bar y Tab Bar) este método es setElevation y la guía Material Design indica que debe estar en 4dp.
- Si creamos un nuevo proyecto con un targetSDK a partir de la API 21 y un minSDK también de 21, podremos hacer uso de este método (ToolBar):

Proyecto creado con una API 28



Proyecto compilado con API 28



MinSDK 21 y TargetSDK con API 28

```
asandroidxbis 〉 MainActivity   app   Nexus One API 29
activity_main.xml   MainActivity.java
1  package inesvalino.iessanclemente.bibliotecasandroidxbis;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.Toolbar;
6
7  public class MainActivity extends Activity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13
14         Toolbar myToalbar = (Toolbar) findViewById(R.id.toolbar2);
15
16         float scale = getResources().getDisplayMetrics().density;
17         int pix = (int)(4*scale+0.5f);
18         myToalbar.setElevation(pix);
19     }
20 }
```

Podemos comprobar cómo podemos hacer uso del método **setElevation**.

Resultado de la ejecución

Código del AndroidManifest.xml:

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BibliotecaAndroidX_bis"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

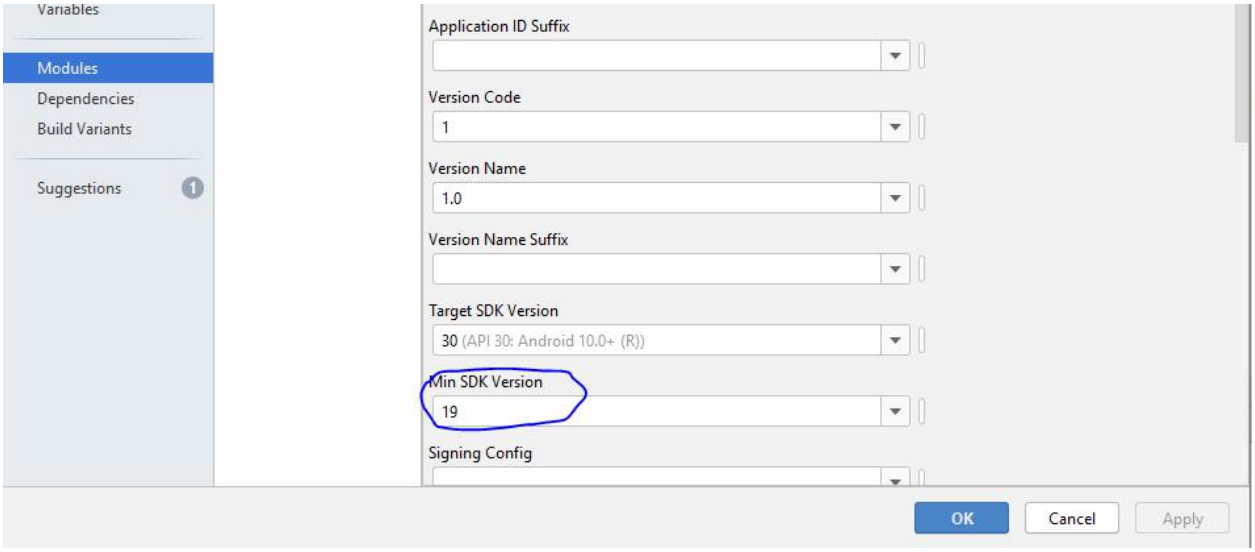
Código del Layout (activity_main.xml):

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9  <android.widget.Toolbar
10         android:id="@+id/toolbar2"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:layout_marginTop="4dp"
14         android:background="?attr/colorPrimary"
15         android:minHeight="?attr/actionBarSize"
16         android:theme="?attr/actionBarTheme"
17         android:title="ToolBar"
18         app:layout_constraintEnd_toEndOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20 </androidx.constraintlayout.widget.ConstraintLayout>
21
```

Línea 9: Observar cómo el control *Toolbar* no es el de la biblioteca de compatibilidad.

```java
import android.app.Activity;
import android.os.Bundle;
import android.widget.Toolbar;

public class MainActivity extends Activity {

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

        Toolbar myToolbar = (Toolbar) findViewById(R.id.toolbar2);

        float scale = getResources().getDisplayMetrics().density;
        int pix = (int) (4 * scale + 0.5f);

        myToolbar.setElevation(pix);
    }
}
```

- Ahora cambiamos el MinSDK a API 19.



Ponemos un minSDK d API 19

En la API 19 no existía el ToolBar. El equivalente se denominaba *ActionBar*, por lo tanto tendremos un error en el código de la Activity.

```
2
3   import ...
6
7   public class MainActivity extends AppCompatActivity {
8
9       @Override
10      protected void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.activity_main);
13
14          Toolbar myToolbar = (Toolbar) findViewById(R.id.toolbar2);
15
16          float scale = getResources(
17          int pix = (int) (4 * scale
18
19          myToolbar.setElevation(pix);
20      }
21  }
```

Class requires API level 21 (current min is 19): android.widget.Toolbar

Add @RequiresApi(LOLLIPOP) Annotation Alt+Mayús+Intro More actions... Alt+Intro

Comprobamos como la llamada al método provoca un error ya que dicho método fue incorporado en la versión API 21

Bibliotecas de compatibilidad v7

Estas bibliotecas proporcionan conjuntos de funciones específicas y se pueden incluir en tu aplicación de manera individual.

Biblioteca appcompat v7, que forma parte de Android Jetpack.

Nota: La biblioteca appcompat migró a la biblioteca de AndroidX, que es un componente de Android Jetpack. Mira cómo funciona en la app de demostración de Sunflower.

Esta biblioteca agrega compatibilidad con el patrón de diseño de la interfaz de usuario de la barra de acciones. Esta biblioteca incluye compatibilidad con implementaciones de interfaz de usuario de material design.

Nota: Esta biblioteca depende de la biblioteca de compatibilidad v4.

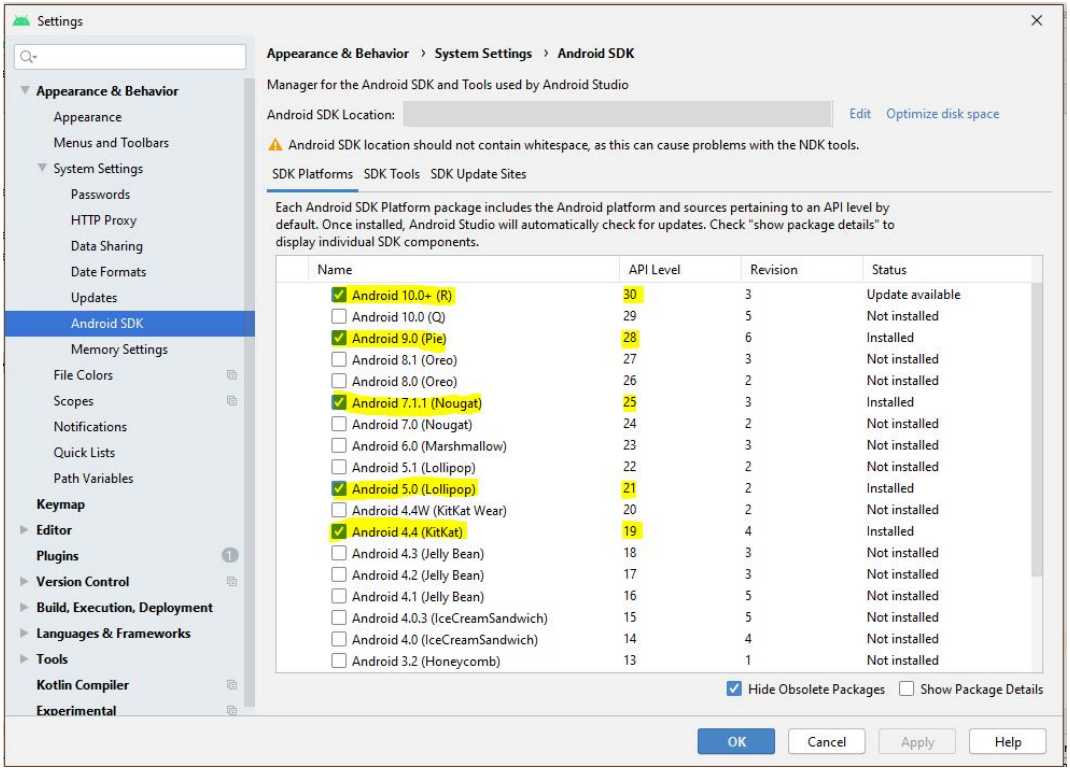
Estas son algunas de las clases clave incluidas en la biblioteca appcompat v7:

- `ActionBar`: Proporciona una implementación del patrón de interfaz de usuario de la barra de acciones. Para obtener más información sobre cómo usar la barra de acciones, consulta la guía para desarrolladores de la barra de acciones.
- `AppCompatActivity`: Agrega una clase de actividad de aplicación que se puede usar como clase base para actividades que usan la implementación de la barra de acciones de la biblioteca de compatibilidad.
- `AppCompatDialog`: Agrega una clase de diálogo que se puede usar como una clase base para los diálogos temáticos de AppCompat.
- `ShareActionProvider`: Agrega compatibilidad con una acción estandarizada de compartir contenido (como correo electrónico o publicación en aplicaciones sociales) que se puede incluir en una barra de acciones.

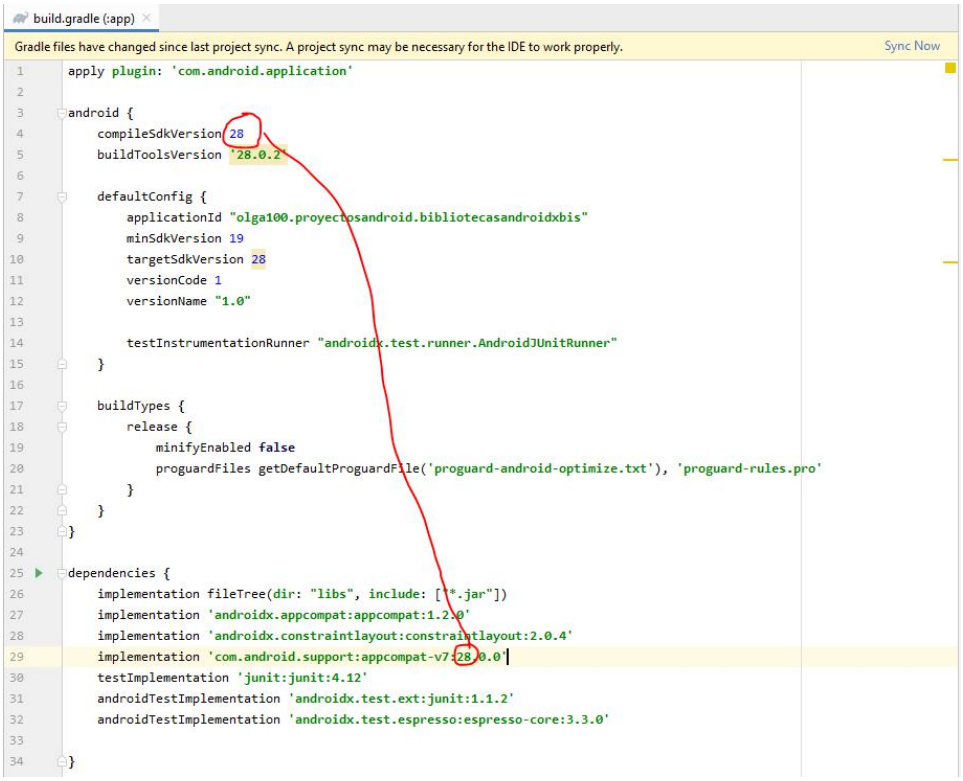
El identificador de dependencias de la secuencia de comandos de compilación de Gradle para esta biblioteca es el siguiente:

```
com.android.support:appcompat-v7:28.0.0
```

Como se puede comprobar la biblioteca de compatibilidad *appcompact-v7* permite hacer uso de las funcionalidades de ActionBar (ToolBar a partir de API 21). **Importante**: En la imagen aparece la versión 28.0.0. Este número tiene que corresponder con la versión de **Compile SDK**.



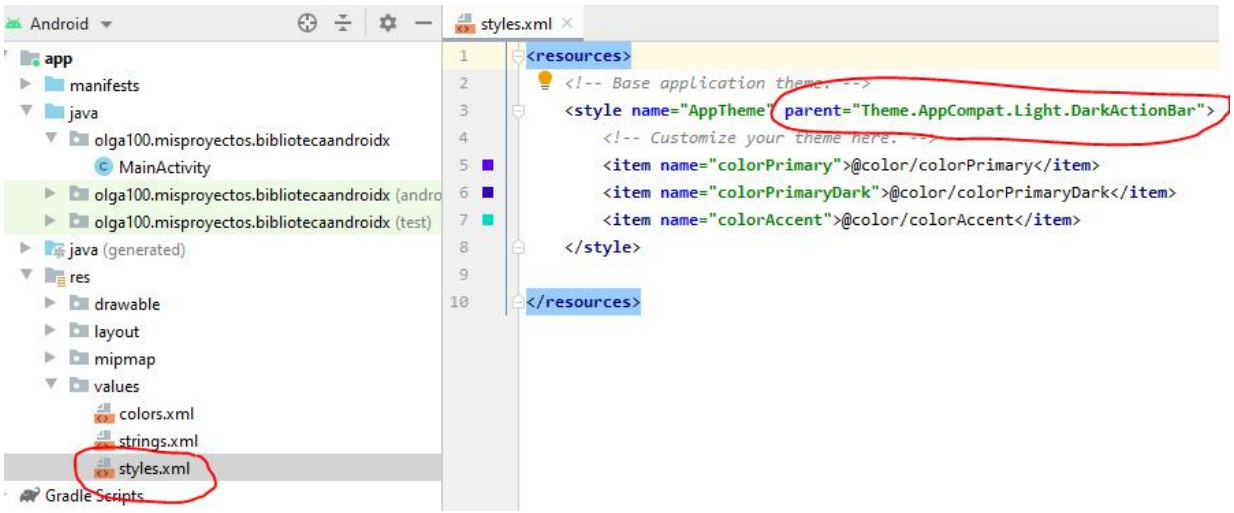
Las API's que aparecen en 'Compile SDK' son las que están descargadas desde el SDK Manager.



Para incorporar una biblioteca de compatibilidad solo tenemos que escribir **implementation 'nombre_biblioteca:version'** en el archivo **build.grandle** del módulo. En nuestro ejemplo es:

```
implementation com.android.support:appcompat-v7:28.0.0
```

No debemos emplear la API 28 ya que cambia el funcionamiento



Ahora hay varias formas de agregar ToolBar-ActionBar. Una de ellas consiste en poner un Theme que incorpore una barra de acción. En nuestro ejemplo: **Theme.AppCompat.Light.DarkActionBar** Por lo tanto, eliminamos del layout de la Activity el Toolbar que teníamos puesto.

MainActivity.java

```
package olga100.proyectosandroid.bibliotecaandroidxbis;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

        android.support.v7.app.ActionBar actionBar = getSupportActionBar();
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        actionBar.setIcon(R.drawable.ic_launcher_foreground);

        float scale = getResources().getDisplayMetrics().density;
        int pixels = (int) (4 * scale + 0.5f);
        actionBar.setElevation(pixels);
    }
}
```

Tenemos que modificar la Activity y ahora será una subclase de la clase AppCompactActivity. El ActionBar la obtenemos llamando al método 'getSupportActionBar ()'. Podemos ver cómo ya podemos hacer uso del método 'setElevation ()'. Indicar que esta forma sería la recomendada para hacer uso de ActionBar en cualquier versión de Android. Así nos aseguramos que todas van a funcionar igual. Si no lo hacemos así y usamos las ActionBar que fueron apareciendo en cada versión das Api's estaremos limitados a los métodos y propiedades del MinSDK indicado.

Resultado de la ejecución.