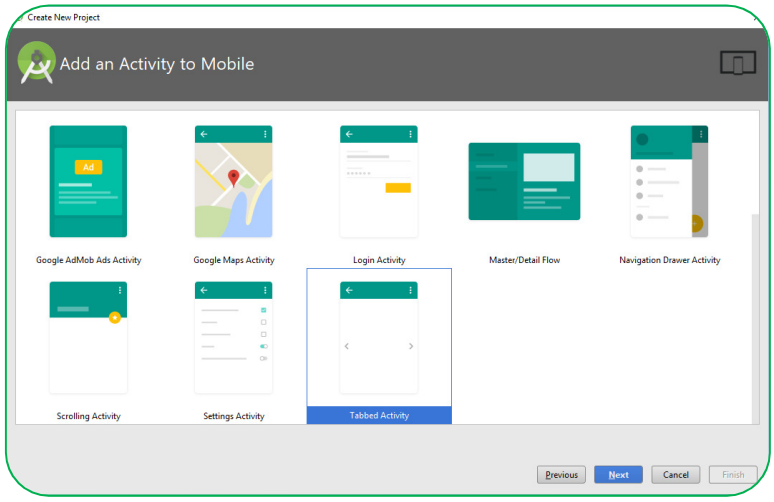
UNIDAD 3: Múltiples Layout

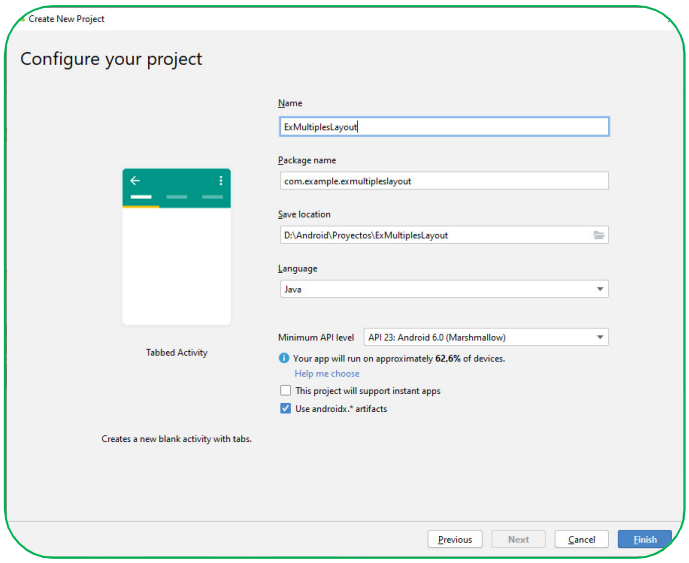
Vamos a crear una aplicación con múltiples pantallas para poder probar sobre ella diferentes tipos de *layout* en lugar de tener una aplicación para cada uno.

Empezaremos por crear un nuevo proyecto al que llamaremos: "ExMultiplesLayout". Selecciona alguna de las APIs que tienes instaladas, pero asegúrate de que sea superior a la 14, ya que utilizaremos funciones que no estaban disponibles anteriormente.

Como tipo de actividad selecciona *Tabbed Activity*:



En la siguiente pantalla establece el nombre del proyecto, el lenguaje de programación (Java) y el nivel mínimo de API (cualquiera de los que tengas instalados por encima de 14). Marca también el uso de androidx.* artifacts.



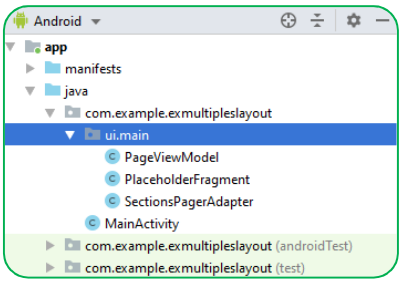
Al utilizar un tipo de actividad diferente al de los proyectos anteriores, el código generado también va a ser bastante diferente, pero por el momento no vamos a analizarlo en profundidad, simplemente lo adaptaremos a nuestras necesidades.

En el archivo "*MainActivity.java*" nos encontramos el siguiente método:

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

    SectionsPagerAdapter sectionsPagerAdapter
      = new SectionsPagerAdapter(this, getSupportFragmentManager());
    ViewPager viewPager = findViewById(R.id.view_pager);
    viewPager.setAdapter(sectionsPagerAdapter);
    TabLayout tabs = findViewById(R.id.tabs);
    tabs.setupWithViewPager(viewPager);

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
      Snackbar.make(view, "Replace…", Snackbar.LENGTH_LONG)
      .setAction("Action", null).show();
        }
    });
}
```

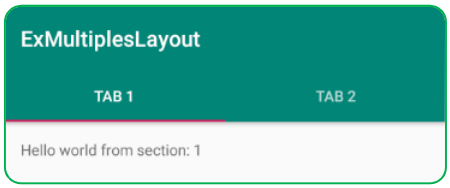
Las dos primeras líneas del método son las habituales, primero se llama al constructor de la superclase y a continuación se establece como vista el *layout* definido en el fichero "*activity_main.xml*", que podemos encontrar en la carpeta *res/layout*.

El siguiente bloque es el encargado de gestionar las secciones por medio de pestañas, para lo que se utilizan las clases: *SectionsPagerAdapter*, *PlaceholderFragment*, y *PageViewModel*, que podemos ver dentro de la carpeta "ui.main" de nuestro proyecto:



El último bloque se encarga de gestionar el botón flotante que aparece en la parte inferior derecha y como no lo necesitamos para nada podemos eliminar este bloque de código y el elemento correspondiente (*FloatingActionButton*) que aparece al final del *layout* "*activity_main.xml*".

Si ejecutamos la aplicación tendremos dos pestañas y podremos pasar de una a otra haciendo *click* sobre ella o deslizando el dedo horizontalmente por la pantalla, en cada una de ellas se muestra el texto "*Hello world form section:*" seguido del número: 1 o 2.



Vamos a ver cómo adaptar la clase "*SectionPagerAdapter*" a nuestras necesidades:

En primer lugar, podemos establecer el número de pestañas que deseamos modificando el valor devuelto por el método "*getCount*".

```
public int getCount() {
    // Show 3 total pages.
    return 3;
}
```

A continuación, podemos modificar el título para las pestañas en el método "*getPageTitle*".

Vemos que actualmente utiliza un array con valores estáticos, cuyo contenido está fijado en el archivo de recursos "*strings*", de manera que al modificar el número de pestañas tendremos que editar tanto el archivo "*strings*" como el array.

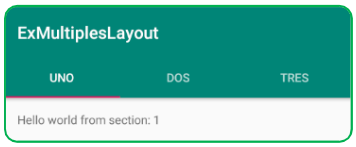
```
private static final int[] TAB_TITLES
    = new int[]{
        R.string.tab_text_1,
        R.string.tab_text_2,
        R.string.tab_text_3
    };
…
public CharSequence getPageTitle(int position) {
    return mContext.getResources().getString(TAB_TITLES[position]);
}
```

```
"strings.xml"
<resources>
    <string name="app_name">ExMultiplesLayout</string>
    <string name="tab_text_1">UNO</string>
    <string name="tab_text_2">DOS</string>
    <string name="tab_text_3">TRES</string>
</resources>
```

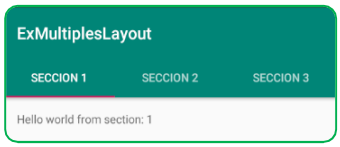
Si el nombre de las pestañas fuera genérico "SECCION 1", "SECCION 2", etc. Podríamos optar por generarlo dinámicamente:

```
public CharSequence getPageTitle(int position) {
    return "SECCION " + String.valueOf(position+1);
}
```

Título personalizado



Título genérico



El siguiente paso será adecuar la aplicación para que permita insertar nuevos *layouts*, y para ello hemos de modificar el método *OnCreateView* de la clase "*PlaceholderFragment*".

Tal y como está el método lo que hace es utilizar el mismo *layout* para todas las pestañas (*fragment_main.xml*) y a continuación modificar el TextView de dicho layout para que muestre el texto " *Hello world form section:*" acompañado del número de sección.

```
public View onCreateView(@NonNull LayoutInflater inflater,
  ViewGroup container, Bundle savedInstanceState) {
    View root = inflater.inflate(R.layout.fragment_main, container, false);
    final TextView textView = root.findViewById(R.id.section_label);
    pageViewModel.getText().observe(this, new Observer<String>() {
        @Override
        public void onChanged(@Nullable String s) {
            textView.setText(s);
        }
    });
    return root;
}
```

Vamos a modificar este método para que al seleccionar una pestaña se muestre un *layout* independiente para cada una. Para ello creamos una vista vacía y le iremos asignando el *layout* que le corresponda a cada pestaña mediante un *switch*.

Podemos mantener el código existente en el "default" para que si una pestaña no tiene establecido un *layout* propio utilice el genérico:

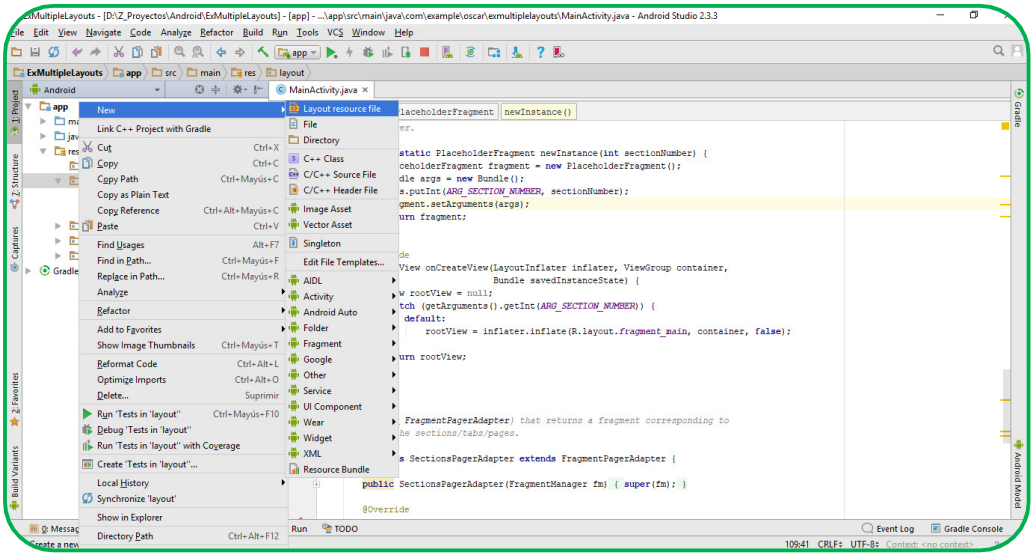
```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
savedInstanceState) {
  View root = null;
  switch (getArguments().getInt(ARG_SECTION_NUMBER)) {
    default:
      root=inflater.inflate(R.layout.fragment_main, container, false);
      final TextView textView = root.findViewById(R.id.section_label);
      pageViewModel.getText().observe(this, new Observer<String>() {
        @Override
        public void onChanged(@Nullable String s) {
            textView.setText(s);
        }
      });
    }
    return root;
}
```

A partir de aquí tendríamos que ir creando cada uno de los *layout* y asignarlo a la pestaña que le corresponda en el *switch*.

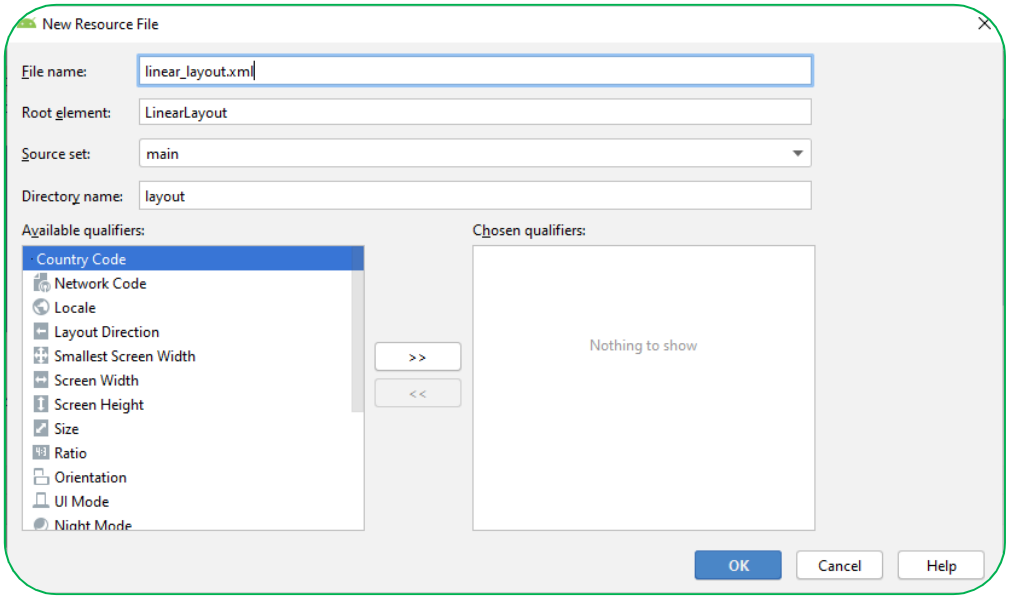
AÑADIR UN LAYOUT A LA APLICACIÓN

Vamos a ver cómo podemos añadir un *layout* a la aplicación que acabamos de crear utilizando como ejemplo un *LinearLayout*, y a partir de ahí el proceso será idéntico para todos los *layout* que queramos incorporar a la aplicación.

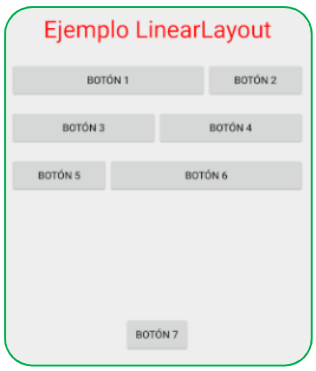
Para agregar un nuevo *layout* vamos al panel de proyecto y sobre la carpeta "*layout*" pulsamos el botón derecho del ratón y seleccionamos "*New*", "*Layout resource file*":



En el diálogo que se nos muestra hemos de establecer un nombre para el fichero de nuestro nuevo *layout* (siempre en minúsculas y con extensión xml) y el elemento raíz del *layout* (tipo de *layout* que queramos utilizar).



A continuación, abriremos el archivo del nuevo *layout* "*linear_layout.xml*" y lo modificaremos introduciendo los elementos que queramos mostrar en pantalla.
Como ejemplo crea un *layout* con esta apariencia:

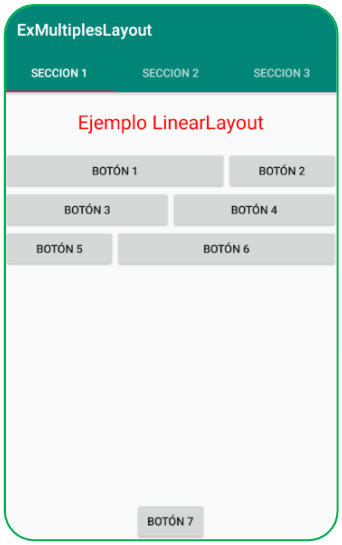


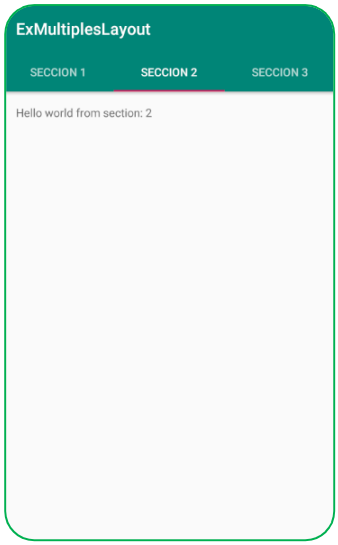
Una vez creada la vista hemos de editar el método "*OnCreateView*" de la clase "*PlaceholderFragment*" para añadir esta nueva vista a la primera pestaña:

```
public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
  View root = null;
  switch (getArguments().getInt(ARG_SECTION_NUMBER)) {

    case 1:
      root=inflater.inflate(R.layout.linear_layout, container, false);
      break;

    default:
      root=inflater.inflate(R.layout.fragment_main, container, false);
      final TextView textView = root.findViewById(R.id.section_label);
      pageViewModel.getText().observe(this, new Observer<String>() {
      @Override
      public void onChanged(@Nullable String s) {
        textView.setText(s);
      }
    });
  }
  return root;
}
```

Lo que hemos hecho es agregar al *switch()* el caso '1' lo que significa que estamos en la primera sección de nuestra aplicación, y lo que hacemos es establecer como vista el archivo "*linear_layout.xml*".

Si ahora ejecutamos nuestra aplicación veremos que como pantalla inicial (sección 1) se muestra nuestra nueva vista, mientras que en el resto de secciones permanece el *layout* por defecto:





Para agregar más pantallas a nuestra aplicación habría que repetir el proceso con cada una de ellas.