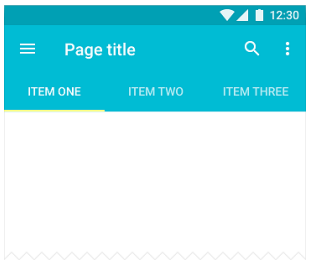
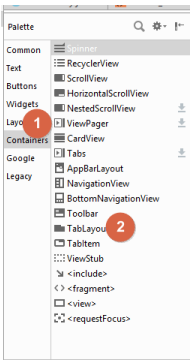
Pestañas

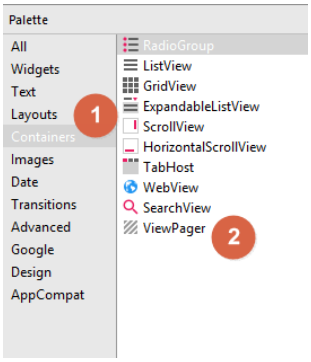


Contenidos

Pestañas en AppBar

Las pestañas o *tabs* suelen ir colocadas justo debajo de la AppBar y normalmente con el mismo color y elevación. Además, debemos poder alternar entre ellas tanto pulsando sobre la pestaña elegida (vista TabLayout) como desplazándonos entre ellas con el gesto de desplazar el dedo a izquierda o derecha sobre el contenido (vista ViewPager).





Crea un proyecto llamado P_45_Tabs desde la plantilla Basic Activity. En nuestra interfaz gráfica necesitamos:

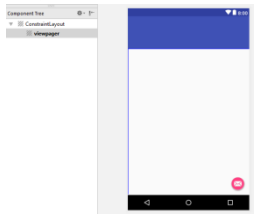
- la vista TabLayout de la librería de diseño para construir el bloque de las pestañas y que incluiremos en el layout **activity_main**:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
   ...

   <android.support.design.widget.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:theme="@style/AppTheme.AppBarOverlay">

      <android.support.v7.widget.Toolbar
         android:id="@+id/toolbar"
         android:layout_width="match_parent"
         android:layout_height="?attr/actionBarSize"
         android:background="?attr/colorPrimary"
         app:popupTheme="@style/AppTheme.PopupOverlay" />

      <android.support.design.widget.TabLayout
         android:id="@+id/tabs"
         android:layout_width="match_parent"
         android:layout_height="match_parent">

      </android.support.design.widget.TabLayout>

   </android.support.design.widget.AppBarLayout>
...

</android.support.design.widget.CoordinatorLayout>
```

Puedes eliminar los 3 ítem que crea el asistente ya que los crearemos por código. Como solo son 3 pestañas podríamos utilizarlo, pero imagina que queremos 25 pestañas, mejor trabajar desde código con un string-array guardado en recursos. Pero RECUERDA identificar la vista TabLayout.

- la vista ViewPager es el "contenedor del contenido" de cada pestaña y permite alternar entre ellas deslizando en horizontal y será el contenido del layout **content_main**:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
   …

   <android.support.v4.view.ViewPager
      android:id="@+id/viewpager"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
</android.support.constraint.ConstraintLayout>
```



Recursos

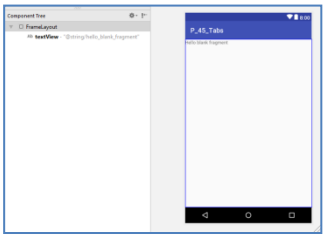
Añade al fichero string.xml las siguientes cadenas:

```
<string-array name="opciones">
   <item>Zaragoza</item>
   <item>Huesca</item>
   <item>Teruel</item>
</string-array>
<string name="texto">Pestaña seleccionada: %1$s</string>
```

Observa que el contenido de la última permite dar formato al primer parámetro (%1) para que sea una cadena ($s). Más información.

Fragmento

El contenido de cada pestaña lo incluiremos en fragmentos independientes, por tanto debemos construirlos con el mecanismo ya conocido de usar el asistente. Para no complicarnos en nuestro ejemplo, el contenido va a ser un TextView (identificado como textView) que tenga el texto de la pestaña escogida:



```
public class BlankFragment extends Fragment {

   private static final String ARG_NUM_TAB = "numTab";
   private String mNumTab;

   public BlankFragment() {
   }

   public static BlankFragment newInstance(String param1) {
      BlankFragment fragment = new BlankFragment();
      Bundle args = new Bundle();
      args.putString(ARG_NUM_TAB, param1);
      fragment.setArguments(args);
      return fragment;
   }

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      if (getArguments() != null) {
         mNumTab = getArguments().getString(ARG_NUM_TAB);
      }
   }

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      View view = inflater.inflate(R.layout.fragment_blank, container, false);
      TextView textView = view.findViewById(R.id.textView);
      textView.setText(getString(R.string.texto, mNumTab));
      return view;
   }

}
```

Repasa Java: El método **newInstance()** de la clase **Class** se usa para crear una nueva instancia de la clase de forma dinámica.

ViewPager

Un ViewPager es un ViewGroup que permite desplazarnos por distintos layouts o "páginas" dentro de una misma Activity/Fragmento simplemente deslizando las páginas a la derecha o izquierda. Se encuentra disponible en la librería de compatibilidad lo que permite su utilización en cualquier versión de Android.

Un ViewPager se añade a cualquier layout como una vista más (en nuestro caso lo hemos añadido al layout content_main).

Este componente va a basar su funcionamiento en un adaptador, de forma similar a los controles tipo lista, aunque en esta ocasión será algo más sencillo. Hay adaptadores estándar implementados que cubren los casos de uso más comunes

Cuando se usa en fragmentos (caso muy normal ya que es una forma conveniente de suministrar y gestionar el ciclo de vida de cada página) se usan **FragmentPagerAdapter** (para un nº de páginas pequeño y fijo) o **FragmentStatePagerAdapter** (nº indeterminado que destruye los fragmentos cuando el usuario se desplaza a otras páginas, minimizando el uso de memoria). Para las actividades es **PageAdapter**.

En nuestro ejemplo, ViewPager se encargará de permitirnos alternar entre los fragmentos que tienen el contenido. Para esto, deberemos construir un adaptador que extienda de FragmentPagerAdapter. En este adaptador tan sólo tendremos que implementar su **constructor** y sobrescribir los métodos **getCount()**, **getItem(pos)** y **getPageTitle()**. Los nombres son bastante ilustrativos, el primero se encargará de devolver el número total de pestañas, el segundo devolverá el fragmento correspondiente a la posición que reciba como parámetro y el último devolverá el título de cada pestaña :

```
public class MiFragmentPagerAdapter extends FragmentPagerAdapter {
   private final String[] textosTab;

   public MiFragmentPagerAdapter(FragmentManager fm, Context context) {
      super(fm);
      textosTab = context.getResources().getStringArray(R.array.opciones);
   }
   @Override
   public int getCount() {
      return 3;
   }
   @Override
   public Fragment getItem(int position) {
      Fragment fragment = null;
      switch (position) {
         case 0:
            fragment = BlankFragment.newInstance(textosTab[position]);
            break;
         case 1:
            fragment = BlankFragment.newInstance(textosTab[position]);
            break;
         case 2:
            fragment = BlankFragment.newInstance(textosTab[position]);
            break;
      }
      return fragment;
   }

   @Override
   public CharSequence getPageTitle(int position) {
      return textosTab[position];
  }

}
```

El parámetro Context context que aparece en el método MiFragmentPagerAdapter() no es obligatorio siempre, aquí está incluído porque es necesario acceder a los recursos de la aplicación.

El código del método getItem() debería optimizarse ya que el mismo en cualquiera de las opciones del switch. Esto es debido a la "simpleza" del ejemplo, la mayoría de las aplicaciones necesitarán fragmentos distintos para cada opción.

Para asociar este adaptador al componente viewpager llamaremos a su método setAdapter() desde el onCreate() de nuestra actividad.

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   …
   FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
   fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
               .setAction("Action", null).show();
      }
   });

   ViewPager viewPager = findViewById(R.id.viewPager);
   viewPager.setAdapter(new MiFragmentPagerAdapter(getSupportFragmentManager(),getApplicationContext()));
   …
```

TabLayout

La clase TabLayout proporciona una disposición horizontal para visualizar las pestañas.

Tras obtener la referencia al control, tendremos que indicar el tipo de pestañas que queremos utilizar, escogiendo entre fijas (TabLayout.MODE_FIXED) o deslizantes (TabLayout.MODE_SCROLLABLE), mediante su método setTabMode().

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   …

   ViewPager viewPager = findViewById(R.id.viewpager);
   viewPager.setAdapter(new MiFragmentPagerAdapter(getSupportFragmentManager(),getApplicationContext()));
   TabLayout tabLayout = findViewById(R.id.tabs);
   tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
   …
```

La modalidad de pestañas fijas (pestañas no deslizantes, todas del mismo tamaño) no debería utilizarse cuando el número de pestañas excede de 3, ya que probablemente la falta de espacio haga que los títulos aparezcan incompletos, no siendo nada buena la experiencia de usuario.

Por último, sólo nos quedará enlazar nuestro ViewPager con el TabLayout, lo que conseguiremos llamando al método setupWithViewPager().

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   …
   tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
   tabLayout.setupWithViewPager(viewPager);
}
```

Este último método hace bastante trabajo por nosotros, ya que se encargará de:

- Crear las pestañas necesarias en el TabLayout, con sus títulos correspondientes, a partir de la información proporcionada por el adaptador del ViewPager.
- Asegurar el cambio de la pestaña seleccionada en el TabLayout cuando el usuario se desplaza entre los fragments del ViewPager. Para ello define automáticamente el listener ViewPager.OnPageChangeListener.
- Y asegurar también la propagación de eventos en "sentido contrario", es decir, asegurará que cuando se selecciona directamente una pestaña del TabLayout, el ViewPager muestra el fragment correspondiente. Para ello definirá el listener TabLayout.OnTabSelectedListener.
- Es decir, nos hemos ahorrado tener que escribir:

```
viewPager.addOnPageChangeListener(new TabLayout.TabLayoutOnPageChangeListener(tabLayout));
tabLayout.addOnTabSelectedListener(new TabLayout.ViewPagerOnTabSelectedListener(viewPager));
```

Es importante tener todo esto en cuenta ya que si necesitamos personalizar de alguna forma la respuesta a estos eventos, o no estamos utilizando un ViewPager, quizá tengamos que hacer este trabajo. Si se diera este caso, existen los métodos newTab() y addTab() del TabLayout con el que poder crear y añadir manualmente las pestañas al control.

Ejecuta la aplicación para comprobar su funcionamiento. Las pestañas comparten estilo con la ActionBar debido a que las vistas ToolBar y TabLayout están definidas en el layout activity_main dentro del contenedor de la librería de diseño AppBarLayout.

Nota: Algo bastante semejante a todo lo anterior nos proporciona la plantilla TabbedActivity con la opción Action Bar Tabs (View Pager) seleccionada como Navigation Style.

Y si en vez de texto en las pestañas, se desea un icono?

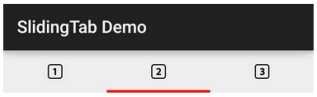
Inside your FragmentPagerAdapter, you can delete the `getPageTitle()` line or simply return null:

```
// ...

@Override
public CharSequence getPageTitle(int position) {
    return null;
}
```

After you setup your TabLayout, you can use the `getTabAt()` function to set the icon:

```
// setup TabLayout first

// configure icons
private int[] imageResId = {
        R.drawable.ic_one,
        R.drawable.ic_two,
        R.drawable.ic_three };

for (int i = 0; i < imageResId.length; i++) {
      tabLayout.getTabAt(i).setIcon(imageResId[i]);
}
```

Sliding tabs with images:

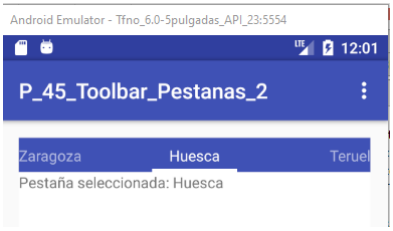


PagerTabStrip

Puede obviarse el uso de TabLayout si en el layout que contiene la vista ViewPager y dentro de dicha vista hay una vista PagerTabStrip.

Aunque definir el estilo es más complejo (ya que las pestañas no están en la AppBar):

```
<android.support.v4.view.ViewPager
   android:id="@+id/viewpager"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <android.support.v4.view.PagerTabStrip
      android:id="@+id/pager_header"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="top"
      android:background="@color/colorPrimary"
      android:textColor="@android:color/white"
      android:paddingBottom="4dp"
      android:paddingTop="4dp"/>

</android.support.v4.view.ViewPager>
```



Observaciones

Encontrarás otras muchas soluciones para trabajar con pestañas, algunas obsoletas, otras más elaboradas, pero la solución propuesta (TabLayout y ViewPager) es la que como desarrolladores nos evita más trabajo.

Práctica propuesta

Realiza el ejercicio planteado en Ejercicios_10_Pestañas