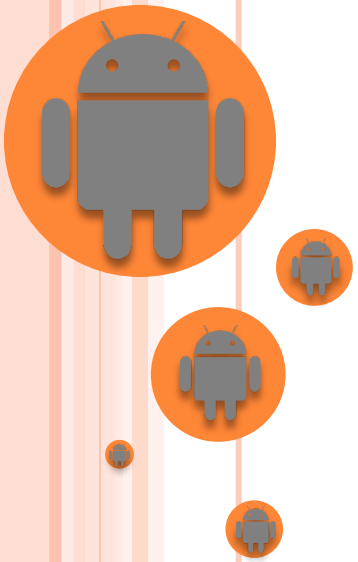


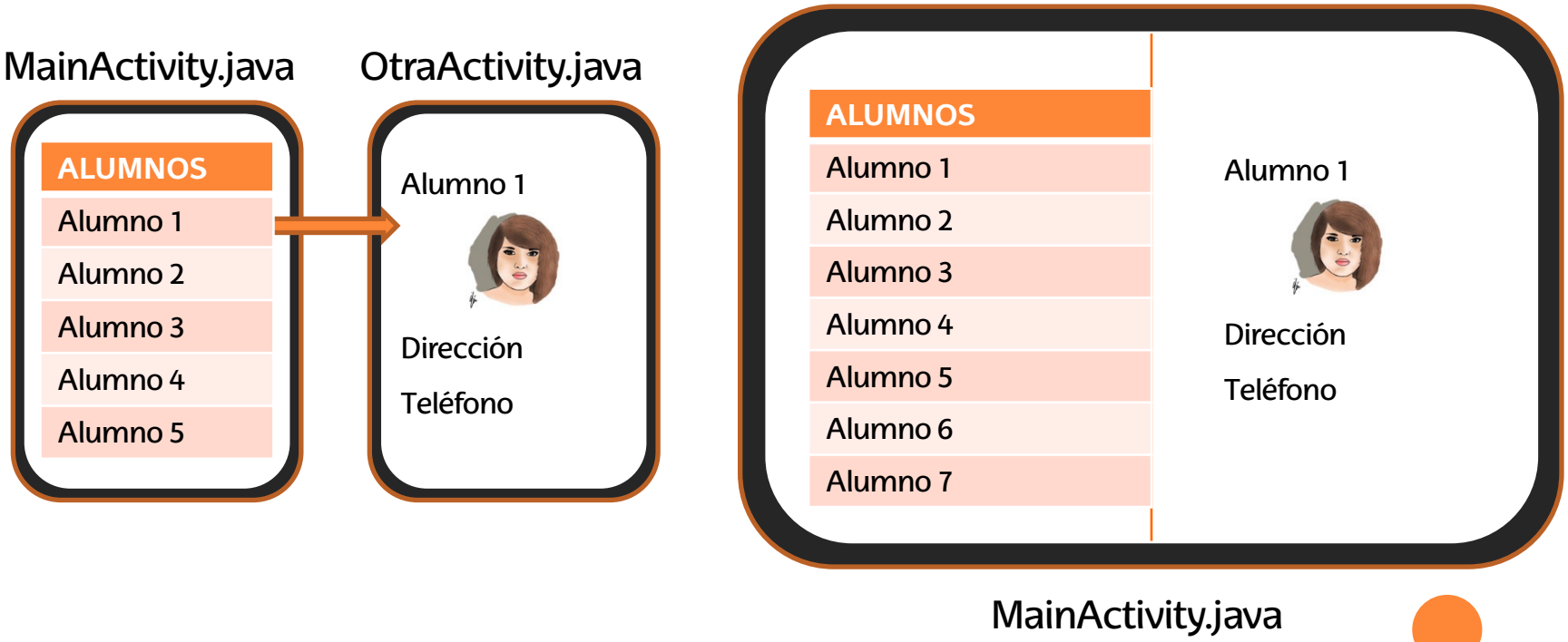
Fragments

Fragments

- Fragments
  - "Módulos" de la aplicación que tienen su propia:
    - Funcionalidad
    - Interfaz gráfica (a través de un fichero XML)
  - Siempre dependen de alguna actividad

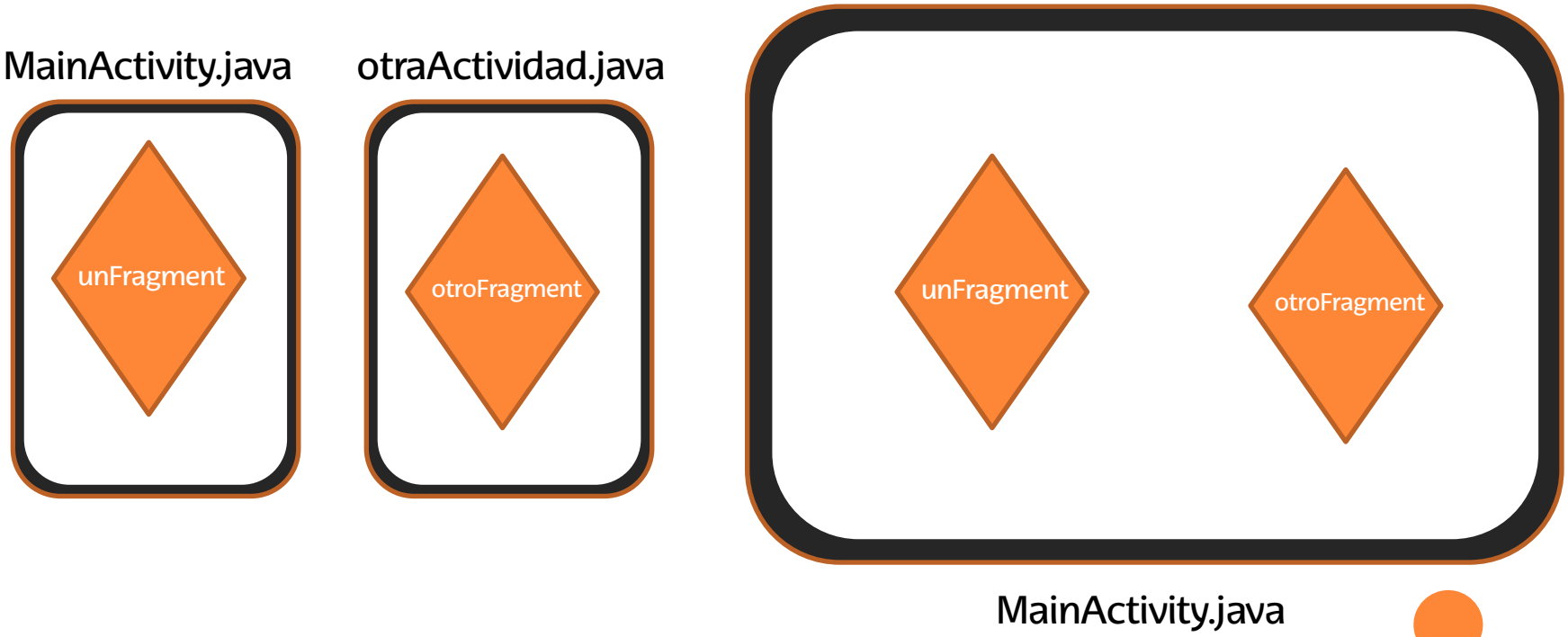
FRAGMENTS

- Fragments
  - El uso de *fragments* permite adaptar la interfaz gráfica en función del dispositivo



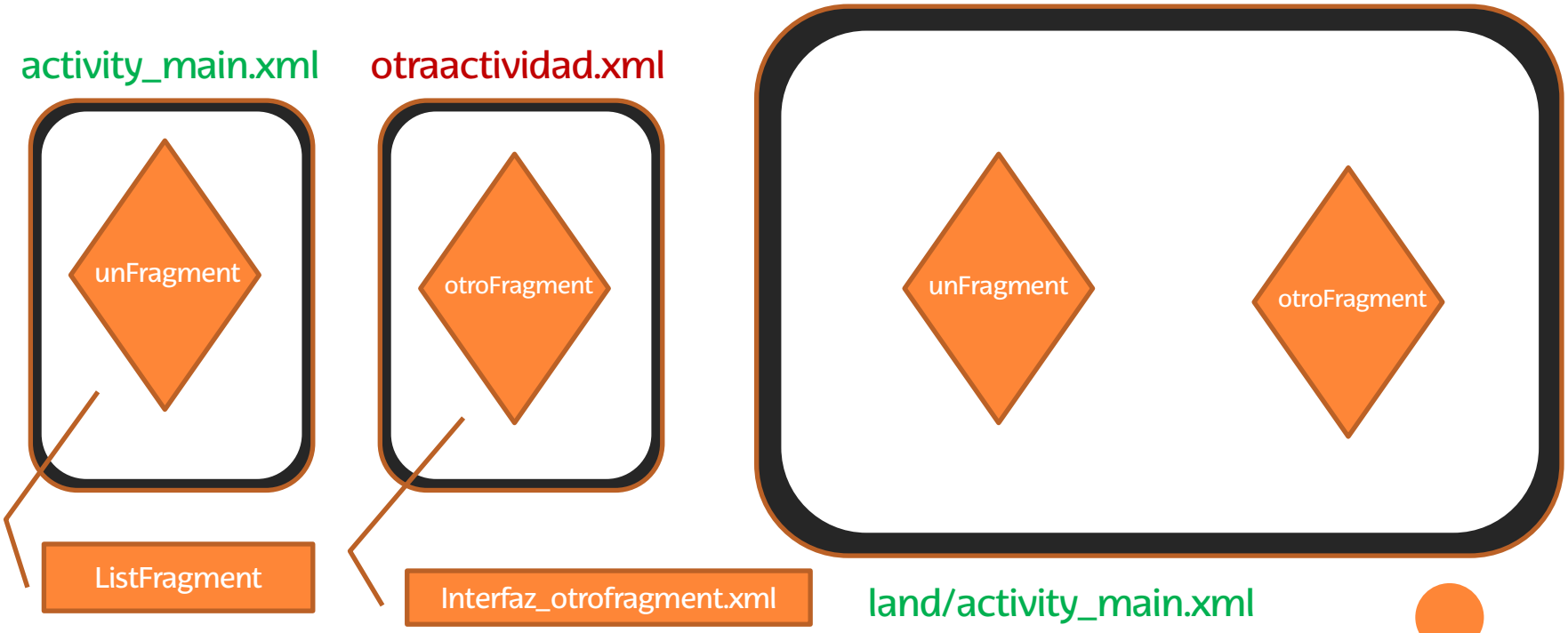
Fragments

- Fragments
  - El uso de *fragments* permite adaptar la interfaz gráfica en función del dispositivo



Fragments

- Fragments
  - El uso de *fragments* permite adaptar la interfaz gráfica en función del dispositivo



Fragments

- Ciclo de vida
  - Similar al de una Activity
- Métodos relevantes:
  - onAttach:
    - Enlace con la actividad.
    - Anterior a onCreate()
  - onCreateView:
    - Creación de la IU
  - onViewCreated:
    - Se llama tras onCreateView de la actividad
  - onViewStateRestored:
    - Para inicializar estados guardados

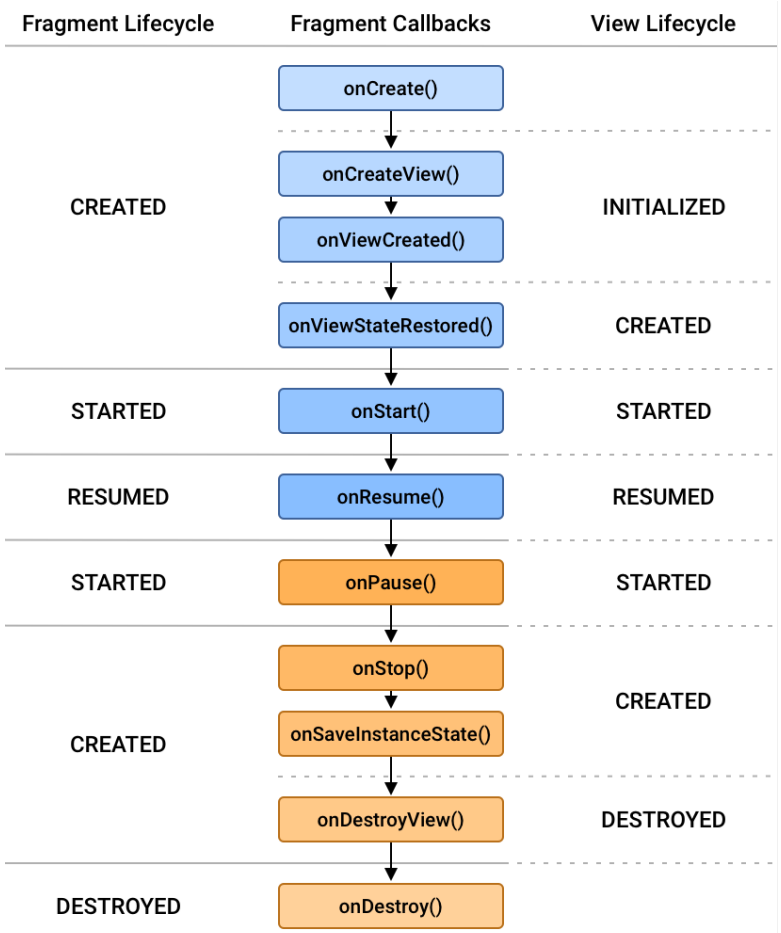


Imagen de: https://developer.android.com/guide/components/fragments?hl=es-419

FRAGMENTS

- Su funcionalidad se implementa en una clase que extiende la clase *Fragment*

```
public class unFragment extends Fragment {
…
```

Del paquete androidx.fragment.app

- Su interfaz se define en un fichero XML como un layout normal

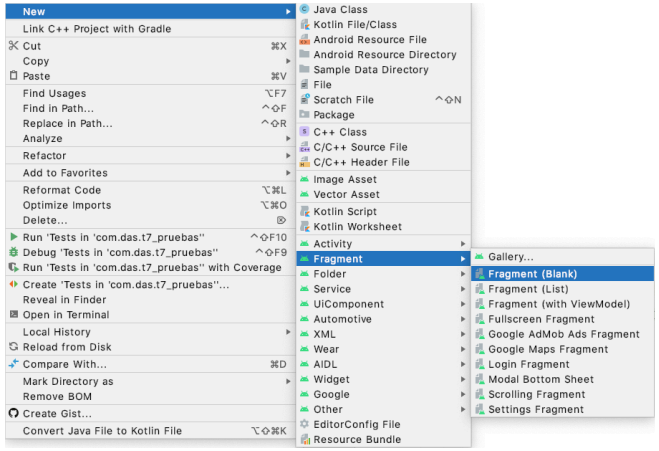
- El *layout* de la actividad es el que organiza los distintos *Fragments*

```
<fragment
    android:id="@+id/fragment"
    android:name="com.example.tema7.unFragment"
    …
```

Nombre de la clase java

FRAGMENTS

- Se puede añadir un nuevo Fragment a través del menú contextual de Android Studio:
  - Elegir *Fragment (Blank)* para obtener un Fragment vacío.



  - **Ventaja**: Genera un fichero de código plantilla y un fichero XML para el Layout.
  - **Desventaja**: Crea demasiado código a eliminar en el fichero plantilla.

FRAGMENTS

- Fragments
  - Hay que sobreescribir el método *OnCreateView(..)* para enlazar la clase java del *fragment* con su correspondiente XML

```
public View onCreateView(@NonNull LayoutInflater inflater,
                         @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    View v= inflater.inflate(R.layout.interfaz_otrofragment,container,false);
    return v;
}
```

Fichero xml con la interfaz para la funcionalidad del fragment

  - A tener en cuenta:
    - Un *fragment* no es una actividad, aunque siempre dependerá de una.
    - Cuando necesitemos el contexto, se obtendrá a través del método *getContext()*. El método *getActivity()* devuelve el FragmentActivity.

FRAGMENTS

- Fragments
  - El método *OnViewCreated(..)* se ejecuta cuando se ha creado la actividad relacionada con ese *fragment*
    - Será ahí donde podremos acceder al contexto y a los elementos de la interfaz gráfica
    - Para acceder a elementos de la interfaz gráfica, utilizar *view.findViewById(..)*

```
public void onViewCreated(@NonNull View view,
@Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    TextView laetiqueta= view.findViewById(R.id.etiqueta);
    …
}
```

FRAGMENTS

- Fragments
  - Existen subclases de *Fragment* con un propósito determinado
    - ListFragment: contiene un *ListView*
    - DialogFragment: para crear *Dialogs*
    - WebViewFragment: contiene una *WebView*
    - PreferenceFragmentCompat: almacena las preferencias del usuario
  - Para usarlos, el *fragment* debe extender de ellos

```
public class unFragment extends ListFragment {
   …
}
```

Del paquete androidx.fragment.app

  - Se comportan distinto que un *fragment* "normal"

FRAGMENTS

- Fragments
  - Un Fragment de tipo lista
    - Puede contener un *ListView* por defecto, lista personalizada o RecyclerView
    - Sobreescribir métodos para trabajar directamente con el *ListView*
    - Hay que crear el adaptador de lista y enlazarlo a la vista

Hay que crear un layout de la lista

```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_item_list,
container, false);
    view.setAdapter(new AdaptadorListView(view.getContext(),params))
return view;
}
```

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
…
}
```

FRAGMENTS

- Fragments
  - La comunicación entre *Fragments* no debería hacerse directamente entre los *Fragments*
    - No siempre existen todos los *Fragments*
    - Para poder reutilizarlo en otras ocasiones
  - La comunicación se hace mediante *listeners* definidos en los *fragments* e implementados en la *Actividad*

```
public class unFragment extends ListFragment {

    public interface listenerDelFragment{
        void seleccionarElemento(String elemento);
    }
    private listenerDelFragment elListener;
  …
```

Se define en el *fragment* con todos los métodos que necesitemos

FRAGMENTS

- Fragments
  - Hay que unir el listener con los métodos implementados en la actividad
    - Método *onAttach(..)*
      - Momento que la actividad "se enlaza" con el fragment (con el contexto)

```
public void onAttach(Context context) {
    super.onAttach(context);
    try{
        elListener=(listenerDelFragment) context;
    }
    catch (ClassCastException e){
        throw new ClassCastException("La clase " +context.toString()
                + "debe implementar listenerDelFragment");
    }
}
```

Aseguramos que toda actividad que se enlaza con el fragment, implementa el Listener

Fragments

- Fragments
  - En el fragment, llamamos a los métodos del listener cuando nos interese

```
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    elListener.seleccionarElemento(datos[position]);
}
```

Cuando ocurre algo en el fragment, llamamos al método del listener

FRAGMENTS

- Ejemplo:
  - Si el teléfono está en apaisado: actualizar 2º Fragment
  - Si el teléfono está en vertical: lanzar nueva actividad

La actividad implementa el *listener*

```
public class MainActivity extends AppCompatActivity implements unFragment.listenerDelFragment {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }

    public void seleccionarElemento(String elemento){
        int orientation = getResources().getConfiguration().orientation;
        if (orientation == Configuration.ORIENTATION_LANDSCAPE){
            //EL OTRO FRAGMENT EXISTE
            otroFragment elotro=(otroFragment) getSupportFragmentManager().
                                                findFragmentById(R.id.otrofragment);
            elotro.hacerAlgo(elemento);
        }

        else{
            //EL OTRO FRAGMENT NO EXISTE, HAY QUE LANZAR LA ACTIVIDAD QUE LO CONTIENE
            Intent i= new Intent(this,otraActividad.class);
            i.putExtra("contenido",elemento);
            startActivity(i);
        }
    }
```

Comprobar la orientación del telefono

Si está en apaisado, el 2º *fragment* existe. Se hace el *cast* a su clase y se llama al método que se desee

Si está en vertical, el 2º *fragment* no existe. Se hace un *intent* a la actividad que lo contiene con la información en el intent

FRAGMENTS

- Ejemplo:
  - Si el teléfono está en apaisado: actualizar 2º Fragment
  - Si el teléfono está en vertical: lanzar nueva actividad

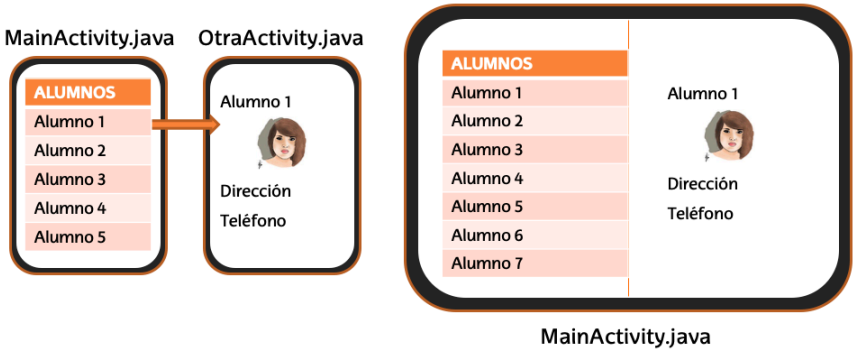
En la actividad que contiene el otro *fragment*, recogemos los datos del *intent*, enlazamos el *fragment* y ejecutamos su método de actualización

```
public class otraActividad extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_otra_actividad);
        otroFragment elotro = (otroFragment) getSupportFragmentManager().
                                                findFragmentById(R.id.otrofragment);
        String informacion=getIntent().getStringExtra("contenido");
        elotro.hacerAlgo(informacion);
    }
}
```

FRAGMENTS

- Ejercicio 1: Cread una aplicación que siga el comportamiento descrito en el ejemplo a lo largo de este tema.
  - Si el dispositivo está en vertical, al seleccionar un elemento del listado, se abre otra actividad con los detalles.
  - Si el dispositivo está en horizontal, los detalles del elemento seleccionado se muestran en la parte derecha de la interfaz.



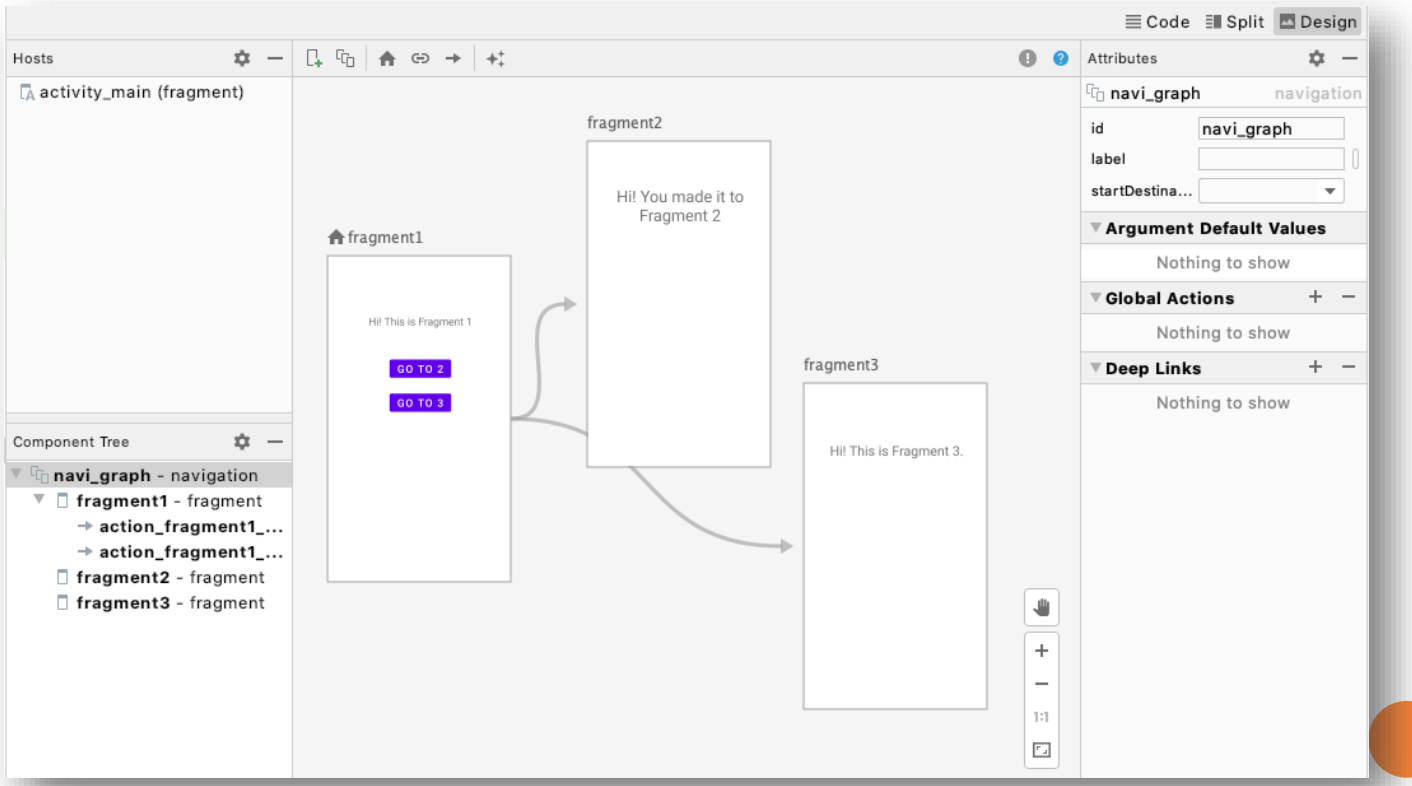
NAVIGATION

- Componente para gestionar el uso de Fragments de manera más visual
  - Introducido en 2018
  - Compuesto por 3 partes clave
- Para empezar a utilizarlo hay que incluir las siguientes dependencias en el fichero *build.gradle* dentro de la carpeta app (existe otro fichero igual en el proyecto):

```
def nav_version = "2.4.1"
// Java language implementation
implementation "androidx.navigation:navigation-fragment:$nav_version"
implementation "androidx.navigation:navigation-ui:$nav_version"
// Feature module Support
implementation "androidx.navigation:navigation-dynamic-features-fragment:$nav_version"
// Testing Navigation
androidTestImplementation "androidx.navigation:navigation-testing:$nav_version"
```

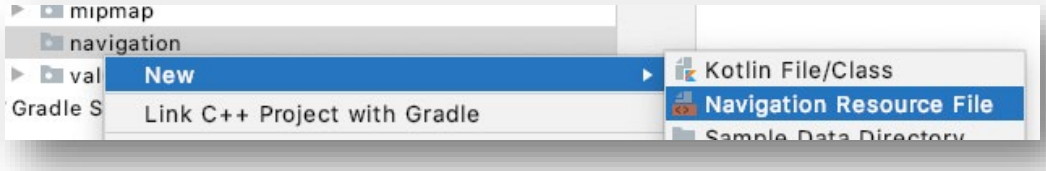
NAVIGATION

- Parte 1/3: Navigation Graph
  - Grafo que define los Fragments y las relaciones entre ellos



NAVIGATION

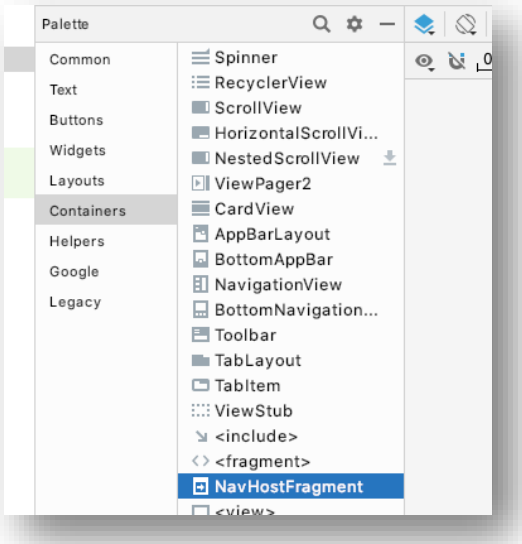
- Parte 1/3: Navigation Graph
  - Definido como un fichero XML
  - Para crearlo:
    - Crear una carpeta "Android Resource Directory" en la carpeta res
    - En la nueva carpeta, crear un nuevo "Navigation Resource File"



    - Definir nombre en Filename, p.e., "nav_graph"
  - Elementos:
    - *Destino*: Diferentes Fragments
    - *Acción*: Transición o ruta entre "Destinos"

NAVIGATION

- Parte 2: NavHostFragment
  - Fragment que se añade al Layout de una Actividad
    - Categoria “Containers” del editor.
  - Hace de contenedor para ir visualizando los destinos




```
<fragment
   android:id="@+id/nav_host_fragment"
   android:name="androidx.navigation.fragment.NavHostFragment"
   android:layout_width="0dp"
   android:layout_height="0dp"
   app:layout_constraintLeft_toLeftOf="parent"
   app:layout_constraintRight_toRightOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:layout_constraintBottom_toBottomOf="parent"

   app:defaultNavHost="true"
   app:navGraph="@navigation/nav_graph" />
```

Referencia al Navigation Graph

NAVIGATION

- Parte 3: NavController
  - Objeto asociado a un NavHostFragment
  - Sirve para ejecutar las acciones definidas en el grafo de navegación
    - Utilizar p.e. en un Listener onClick:

```
public class Fragment1 extends Fragment {
    …

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        Button b = view.findViewById(R.id.fra1_button);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Navigation.findNavController(v).navigate(R.id.action_fragment1_to_fragment2);
            }
        });
    }
```

Navigation

- Conceptos relacionados
  - Vínculos directos (Deep Links):
    - Salto entre Destinos sin usar Acciones
    - Pueden ser explícitos o implícitos
  - Safe Args:
    - Complemento a incluir en Gradle
    - Genera una serie de clases Java llamadas "Direcciones" para gestionar el paso de parámetros en cada Acción

- Mas información sobre Navigation:
  - https://developer.android.com/guide/navigation
  - https://developer.android.com/guide/navigation/navigation-migrate

Fragments

- Ejercicio 2:
  - Cread una aplicación con:
    - 1 Activity que contenga un NavHostFragment
    - 3 Fragments
      - 1 de ellos tendrá 2 botones para abrir cada uno de los otros 2
    - 1 grafo de navegación
      - Con las acciones definidas en el grafo de la Diapositiva 20.
  - Diseñar las acciones entre Fragments con Navigation Graph e implementarlas.