Integrantes: **Alejandra Chávez**

Grupo: **IC-VA-MAT-LabA**

Carrera: **Ingeniería en Computación**

Asignatura: **Programacion Movil**

Prof.: Ing. Luis Guido

Informe de Investigación: Fragments en Android

1. Introducción

Este informe tiene como objetivo proporcionar una visión detallada sobre los Fragments en el desarrollo de aplicaciones Android. Los Fragments permiten modularizar la interfaz de usuario y la lógica de una aplicación, facilitando la creación de interfaces más flexibles y reutilizables.

2. About Fragments

Los Fragments son componentes modulares de una interfaz de usuario en Android. Cada Fragment representa una parte de la interfaz de usuario y tiene su propio ciclo de vida, que está estrechamente ligado al ciclo de vida de la Activity que lo contiene. Las ventajas de utilizar Fragments incluyen la capacidad de crear interfaces de usuario más dinámicas y adaptables, así como la reutilización de componentes en diferentes partes de una aplicación.

3. Create a Fragment

Para crear un Fragment, se debe extender la clase Fragment y sobrescribir los métodos necesarios, como onCreateView. A continuación se muestra un ejemplo básico de creación de un Fragment:

```
public class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_example, container, false);
    }
}
```

4. Fragment Manager

El FragmentManager es responsable de la gestión de Fragments dentro de una Activity. Proporciona métodos para agregar, eliminar y reemplazar Fragments, así como para manejar la pila de retroceso (back stack). Los métodos principales incluyen beginTransaction, add, replace, remove y commit.

5. Fragment Transactions

Las transacciones de Fragments permiten realizar cambios en la jerarquía de Fragments. Los métodos comunes utilizados en las transacciones incluyen add, replace y remove. A continuación se muestra un ejemplo de cómo realizar una transacción de Fragment:

```
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.replace(R.id.fragment_container, new ExampleFragment());
transaction.commit();
```

6. Animate Transitions Between Fragments

Las transiciones animadas entre Fragments mejoran la experiencia del usuario al proporcionar una interfaz más suave y atractiva. Android ofrece varias formas de animar las transiciones, como mediante el uso de animaciones predefinidas o personalizadas. A continuación se muestra un ejemplo de cómo animar una transición entre Fragments:

```
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.setCustomAnimations(R.anim.enter, R.anim.exit);
transaction.replace(R.id.fragment_container, new ExampleFragment());
transaction.commit();
```

7. Fragment Lifecycle

El ciclo de vida de un Fragment incluye varios estados, que van desde su creación hasta su destrucción. Estos estados son gestionados por el sistema Android y permiten al desarrollador ejecutar código en momentos específicos del ciclo de vida del Fragment. Los métodos principales del ciclo de vida incluyen onAttach, onCreate, onCreateView, onActivityCreated, onStart, onResume, onPause, onStop, onDestroyView, onDestroy y onDetach.

A continuación se muestra un gráfico del ciclo de vida de un Fragment:

8. Saving State with Fragments

Guardar el estado de un Fragment es crucial para mantener la coherencia de la interfaz de usuario durante cambios de configuración, como la rotación de la pantalla. Android proporciona varios métodos para guardar y restaurar el estado, como onSaveInstanceState y onViewStateRestored. A continuación se muestra un ejemplo de cómo guardar y restaurar el estado de un Fragment:

```
@Override
public void onSaveInstanceState(Bundle outState) {
   super.onSaveInstanceState(outState);
   outState.putString('key', value);
}

@Override
public void onViewStateRestored(Bundle savedInstanceState) {
   super.onViewStateRestored(savedInstanceState);
   if (savedInstanceState != null) {
      value = savedInstanceState.getString('key');
   }
}
```

9. Communicate with Fragments

La comunicación entre Fragments y su Activity, o entre diferentes Fragments, es esencial para la interacción en una aplicación. Un enfoque común para esta comunicación es el uso de interfaces, que permiten a un Fragment enviar eventos a su Activity contenedora. A continuación se muestra un ejemplo de cómo comunicar un Fragment con su Activity:

```
public class ExampleFragment extends Fragment {
   private OnFragmentInteractionListener mListener;

   @Override
   public void onAttach(Context context) {
      super.onAttach(context);
      if (context instanceof OnFragmentInteractionListener) {
         mListener = (OnFragmentInteractionListener) context;
      } else {
         throw new RuntimeException(context.toString()
            + ' must implement OnFragmentInteractionListener');
      }
   }

   public interface OnFragmentInteractionListener {
      void onFragmentInteraction(Uri uri);
   }
}
```

10. Conclusión

En conclusión, los Fragments son una herramienta poderosa en el desarrollo de aplicaciones Android, ofreciendo flexibilidad y modularidad. Comprender su ciclo de vida, cómo manejarlos y cómo comunicarse entre ellos es crucial para crear aplicaciones robustas y mantenibles.