Nombre: Nayelis B. Velásquez Cruz

Carnet: 2011298

Curso: Aplicación Movil

Carrera: Ingeniería en Computación

Informe de investigation About fragments, Create a fragment, Fragment manager, Fragment transactions, Animate transitions between fragments, Fragment lifecycle, Saving state with fragments y Communicate with fragments

- **Introducción a los fragmentos**:
  - **Descripción y Uso**: Los fragmentos son componentes modulares y reutilizables de la interfaz de usuario que se integran dentro de una actividad. Cada fragmento tiene su propio ciclo de vida, recibe sus propios eventos de entrada y puede añadirse o eliminarse dinámicamente mientras la actividad que los contiene está en ejecución (Android Developers).
  - **Beneficios**: Permiten una mejor organización del código y facilitan la creación de interfaces de usuario flexibles y adaptables, especialmente en dispositivos con diferentes tamaños de pantalla, como tabletas y teléfonos (Android Developers).

- **Crear un fragmento**:
  - **Configuración**: Para crear un fragmento, necesitas añadir una dependencia de la biblioteca AndroidX Fragment a tu proyecto. Esto se hace agregando las dependencias correspondientes en el archivo `build.gradle` de tu aplicación (Android Developers).
  - **Implementación**: Extiende la clase `Fragment` y sobreescribe sus métodos para incluir la lógica de tu aplicación. Puedes definir el diseño del fragmento proporcionando un recurso de diseño al constructor base. También puedes utilizar clases base especializadas como `DialogFragment` para mostrar diálogos flotantes, o `PreferenceFragmentCompat` para mostrar una lista de preferencias (Android Developers).

- **Gestión de fragmentos**:
  - **Transacciones de Fragmentos**: En tiempo de ejecución, el `FragmentManager` puede realizar operaciones como agregar, eliminar o reemplazar fragmentos en respuesta a la interacción del usuario. Cada conjunto de cambios que se realiza se llama una transacción, y puedes agrupar múltiples acciones en una sola transacción. Estas transacciones pueden guardarse en una pila de retroceso para permitir la navegación hacia atrás a través de los cambios en los fragmentos (Android Developers).

    - **Ejemplo de Código**: Puedes obtener una instancia de `FragmentTransaction` desde el `FragmentManager` llamando a `beginTransaction()`, y luego usar métodos como `add()`, `replace()`, y `remove()` antes de llamar a `commit()` para aplicar la transacción (Android Developers).

- **Animaciones en fragmentos**:
    - **Animaciones y Transiciones**: La API de fragmentos proporciona dos formas principales de añadir efectos visuales durante la navegación entre fragmentos: el Marco de Animación y el Marco de Transición. Las animaciones controlan cómo los fragmentos entran y salen de la pantalla, mientras que las transiciones de elementos compartidos determinan cómo un elemento de vista compartida se mueve entre fragmentos (Android Developers).
    - **Configuración de Animaciones**: Puedes definir animaciones personalizadas para los efectos de entrada y salida creando recursos de animación en el directorio `res/anim` y aplicándolos durante las transacciones de fragmentos (Android Developers).

- **Ciclo de vida de los fragmentos**:
    - **Métodos Clave**: Los fragmentos tienen un ciclo de vida que incluye métodos como `onCreate()`, `onCreateView()`, `onActivityCreated()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onDestroyView()`, y `onDestroy()`. Es crucial manejar correctamente estos métodos para asegurar que los fragmentos se comporten como se espera durante su creación, visualización, pausa y destrucción (Android Developers).
    - **Estado del Fragmento**: Es importante gestionar el estado del fragmento a lo largo de su ciclo de vida para asegurar que se mantenga la consistencia y se eviten fugas de memoria (Android Developers).

- **Guardar estado en fragmentos**:
    - **Persistencia de Estado**: Para asegurar que el estado del fragmento se mantenga a través de cambios en la configuración o reinicios del proceso, debes utilizar métodos como `onSaveInstanceState()` para guardar el estado relevante y `onCreate()` para restaurarlo. Además, el framework de Android

maneja automáticamente la restauración de la pila de retroceso y el estado de los fragmentos (Android Developers).
    - **Ejemplo de Guardado de Estado**: Variables locales en el fragmento, estado de la vista, estado guardado y datos de configuración no se deben perder durante cambios de configuración, muerte del proceso o cuando el fragmento se añade a la pila de retroceso (Android Developers).

- **Comunicación con fragmentos**:
    - **Opciones de Comunicación**: Para mantener los fragmentos autocontenidos y reutilizables, deben comunicarse con su actividad anfitriona o con otros fragmentos utilizando un `ViewModel` compartido o la API de Resultados de Fragmentos. Un `ViewModel` es ideal para compartir datos persistentes, mientras que la API de Resultados es adecuada para resultados únicos que se pueden pasar en un `Bundle` (Android Developers).
    - **Ejemplo de Comunicación**: Puedes definir un `ViewModel` que almacene datos observables y utilizarlos en múltiples fragmentos o actividades para compartir información de manera eficiente (Android Developers).