UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
DIVISIÓN SISTEMA UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA

PROGRAMACION DE DISPOSITIV.MOVILES

UNIDAD 4

ACTIVIDAD TEORICA

NOMBRE DEL ALUMNO:

PAULINA RODRIGUEZ SAMPEDRO

NOMBRE DEL ASESOR:

CRISTIAN CARDOSO ARELLANO

SEXTO SEMESTRE

GRUPO: 9696

ACTIVITIES

Objetivo

Comprender los fundamentos teóricos de la arquitectura de la interfaz de usuario en Android studio mediante el uso de Java y XML, dominando el ciclo de vida de los componentes, la gestión de recursos visuales y la vinculación lógica para el desarrollo de aplicaciones móviles funcionales.

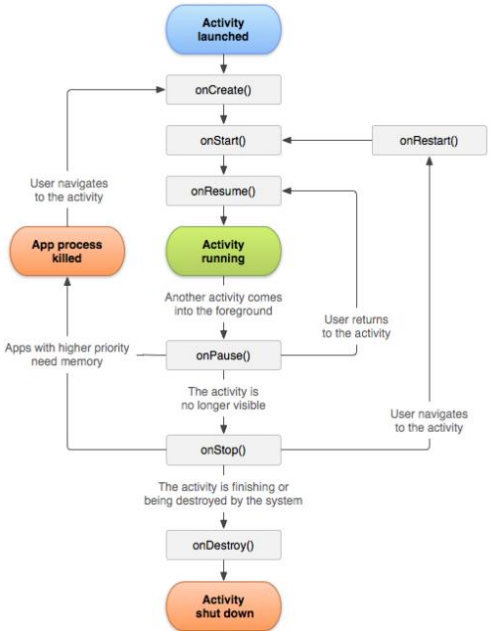
Ciclo de vida de una actividad en Android Studio.

Es el conjunto de estados por los que pasa una actividad desde que se crea hasta que se destruye. El sistema operativo gestiona esto mediante "callbacks" (métodos) que permiten a la app reaccionar correctamente a cambios (como recibir una llamada o girar la pantalla).

Para navegar por las transiciones entre las etapas del ciclo de vida de la actividad, la La clase Activity proporciona un conjunto básico de seis devoluciones de llamada: onCreate() se llama cuando la actividad se crea por primera vez, (aquí se inicializa la UI), onStart() la actividad se vuelve visible para el usuario, onResume() la actividad está en primer plano y el usuario puede interactuar con ella, onPause() a actividad pierde el enfoque, onStop() la actividad ya no es visible para el usuario y onDestroy() la actividad se elimina de la memoria.

El sistema invoca cada una de estas devoluciones de llamada cuando la actividad entra en un nuevo estado.

Ilustración 1. Ciclo de vida de una actividad



Fuente: Información obtenida de Android Developers

Definición de vistas utilizando XML.

En Android, la interfaz de usuario (UI) se define de forma declarativa mediante archivos XML ubicados en la carpeta res/layout/. Este enfoque permite separar la estructura visual de la lógica de programación en Java. Los elementos fundamentales de esta interfaz son las Vistas (View), que actúan como los componentes básicos de construcción. Una View es un objeto que dibuja algo en la pantalla con lo que el usuario puede interactuar; por ejemplo, el TextView (visualización de texto), el Button (acciones de clic) y el ImageView (despliegue de gráficos).

Unión entre código de programación JAVA y XML con Android Studio.

Para que el diseño visual que se creó en XML se visualice, es necesario establecer un "puente" con la lógica en Java. Este proceso se basa en la vinculación de los componentes declarados en el archivo de diseño con objetos manipulables en el código fuente.

El proceso de unión se realiza principalmente en la clase Activity y consta de dos pasos fundamentales regulados por el SDK de Android:

1. Inflación del Layout (setContentView): En el método onCreate de la actividad, se utiliza la instrucción setContentView(R.layout.nombre_archivo). Este comando le indica a Android que tome el archivo XML de la carpeta res/layout/ y lo transforme ("infle") en objetos de memoria.
2. Vinculación de Componentes (findViewById): Una vez inflado el diseño, se utiliza el método findViewById(R.id.id_componente) para obtener una referencia específica de una View del XML. Para que esto funcione, el componente en el XML debe tener un atributo android:id.

Layouts simples.

Los Layouts son contenedores que definen cómo se posicionan sus elementos hijos en la pantalla. Los más comunes para comenzar son los LinearLayout que organizan los elementos en una sola dirección (fila o columna) mediante el atributo android:orientation, el relativeLayout que posiciona los elementos en relación con el contenedor padre o con otros elementos y el frameLayout diseñado para mostrar un solo elemento o elementos encimados (como una imagen con un texto encima).

Conclusión

Conocer los conceptos relacionados con Android Studio, es un pilar fundamental ya que representan la base técnica sobre la cual se construye cualquier aplicación profesional. Comprender el Ciclo de Vida de la Actividad es el primer paso para garantizar la estabilidad de una app, pues permite gestionar correctamente la memoria y los datos cuando el usuario recibe una llamada o rota la pantalla, evitando cierres inesperados. Al combinar esto con la definición de vistas en XML, el estudiante en informática entenderá que separar la estética de la lógica, es una práctica esencial llamada desacoplamiento que facilita el mantenimiento del código a largo plazo.

Por otro lado, dominar la unión entre Java y XML es lo que permite transformar un diseño estático en una herramienta funcional y reactiva, convirtiendo simples etiquetas visuales en objetos con los que el usuario puede interactuar. Finalmente, el uso de layouts simples proporciona la destreza necesaria para organizar interfaces de forma jerárquica y adaptable a múltiples dispositivos. En conjunto, estos conocimientos no solo permiten escribir código, sino entender la arquitectura interna de Android, lo cual es indispensable para avanzar con éxito hacia temas más complejos como el consumo de APIs, bases de datos o arquitecturas de software modernas.

Referencias

Android Developers.(s.f.). *The activity lifecycle*. https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419

Android Developers. (s.f.). *Declaring layout*. https://developer.android.com/develop/ui/views/layout/declaring-layout?hl=es-419 (developer.android.com

Android Developers (s.f.). *Relative layout*. https://developer.android.com/develop/ui/views/layout/relative?hl=es-419