87. Layout en Jetpack Compose

Para comprender como funciona el Jetpack Compose, debemos tener en cuenta estos puntos :

1. Componentes en un Árbol: en Jetpack Compose, los componentes (o composables) se organizan en un árbol de UI. Este concepto es fundamental en Compose y se asemeja a otros frameworks de UI declarativos. Cada composable puede contener otros composables, creando una estructura jerárquica.
2. Contenedores y Elementos Finales:
3. Contenedores: Estos son composables que pueden contener otros composables dentro de ellos. Ejemplos comunes son **Row**, **Column**, **Box**, y **ConstraintLayout**. Actúan como estructuras para organizar otros elementos en la pantalla, gestionando su disposición y alineación.
4. Elementos Finales (Leaf Composables): Estos son composables que generalmente no contienen otros composables (pero pueden contenerlos o formar parte del composable) y suelen ser los bloques de construcción de la UI. Ejemplos incluyen Text, Button, Image, etc. Estos elementos son los que finalmente se renderizan en la pantalla.
5. Importancia del Estado y la Recomposición: En Compose, el estado juega un papel crucial. Los cambios en el estado pueden provocar la recomposición de parte del árbol de UI. Esto significa que los composables se vuelven a ejecutar para reflejar el estado actualizado, permitiendo interfaces dinámicas y reactivas.
6. Modifiers: No mencionaste los modifiers, pero son un aspecto esencial en Compose. Permiten modificar o decorar los composables de muchas maneras, como ajustar el tamaño, la disposición, la apariencia, añadir comportamientos como el click o el scroll, etc.
7. Declarativo y Reactivo: Compose es un framework declarativo y reactivo. Esto significa que describes cómo debería verse tu UI en función del estado actual, y Compose se encarga de hacer los cambios necesarios en la pantalla cuando este estado cambia.

Los composables contenedores organizan los composables hijos teniendo en cuenta las preferencias de los hijos. Se pueden resumir en los sigientes puntos:

- **Organización de Elementos Hijos**: Los contenedores como `Row`, `Column`, y `Box` en Jetpack Compose están diseñados para organizar y posicionar elementos hijos en la interfaz de usuario. Estos contenedores controlan cómo se dispone cada elemento hijo en relación con los demás.
- **Preferencias de Tamaño**: Cada composable hijo expresa sus propias preferencias de tamaño, que pueden incluir un tamaño mínimo, máximo, o específico. Los contenedores toman en cuenta estas preferencias para decidir cómo disponer y qué tamaño asignar a cada hijo
- **Modifiers**: Los modifiers se pueden utilizar para influir en estas preferencias de tamaño. Por ejemplo, puedes usar `Modifier.weight()` en un hijo dentro de una Row o Column para asignarle una proporción del espacio disponible.
- **Arrangement y Alignment**: Además del tamaño, los contenedores consideran otros factores como el alineamiento (Alignment) y la disposición (Arrangement). Estos determinan la posición exacta de los elementos hijos en el eje horizontal o vertical del contenedor.

- **Constraints Propagadas**: En Compose, los contenedores también propagan restricciones a sus hijos. Estas restricciones son reglas sobre el tamaño mínimo y máximo que los hijos pueden tener, basadas en el espacio disponible y las necesidades de los otros elementos en la interfaz.

87.1 Layout básicos

- Column, Row, Box
- Ajustar espacios entre componentes: Padding size, spacing

87.2 Alineación y distribución

- Alineación (Alignment): Controlar cómo se alinean los elementos dentro de un layout.
- Distribución (Arrangement): Manejar la distribución de espacio en Column y Row.

87.3 Layouts Flexibles y Adaptativos

- ConstraintLayout: Para diseños más complejos y flexibles.
- Weight: Distribuir espacio en Row y Column.
- Manejo de diferentes tamaños de pantalla y orientaciones.

87.4 Scrollables y Listas

- LazyColumn y LazyRow: Para listas y filas scrollables.
- ScrollableColumn y ScrollableRow: Alternativas (menos eficientes) para contenido scrollable.

87.5 Animaciones y Transiciones

Animaciones en los layouts. Transiciones entre estados.

87.6 Componentes para layout

87.6.1 Scaffold

Es un componente muy útil que proporciona una estructura básica para las aplicaciones Android con una UI común. `Scaffold` se encarga de organizar la AppBar, el contenido principal, un FloatingActionButton, un Drawer, y otros elementos comunes en una sola estructura coherente.

Algunos puntos claves sobre Scaffold:

- Estructura de la Aplicación: Scaffold ayuda a estructurar la interfaz de usuario de manera lógica y coherente. Te permite definir un layout consistente en toda la aplicación.
- Flexibilidad: Aunque ofrece una estructura predefinida, Scaffold es bastante flexible y te permite personalizar sus diferentes secciones como el cuerpo, la barra de aplicaciones, el botón flotante, etc.

- Facilidad de Uso: Con Scaffold, puedes crear una UI compleja de manera más sencilla. Gestiona automáticamente aspectos como la coordinación entre el FloatingActionButton y el Snackbar.
- Integración con Material Design: Scaffold está diseñado para trabajar perfectamente con los principios de Material Design, lo que te permite crear interfaces que se sienten naturales en Android.

87.7 Apendice

Versión 0.5 (20~~-12-~~23)