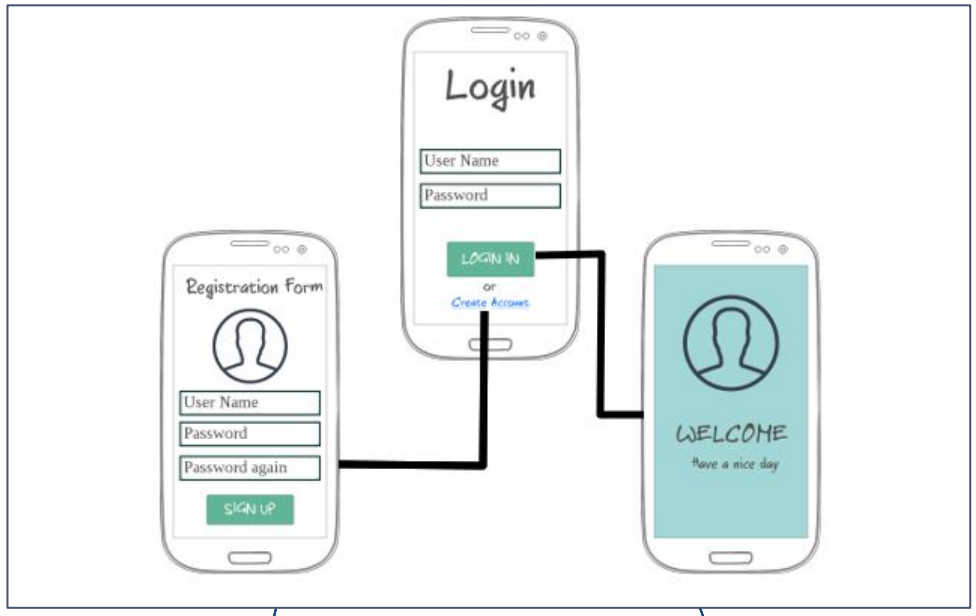
Desarrollo de Aplicaciones para Dispositivos Móviles

Ingeniero Walter Medina
walter.medina@correounivalle.edu.co

Introducción a Jetpack Compose



Para realizar las vistas de una aplicación android se puede hacer con **XML** o con **Jetpack Compose**.

XML: Es el método tradicional y más común para definir la interfaz de usuario.

Jetpack Compose: Es el framework moderno de Google para crear interfaces en Android.

JetPack Compose usa el mismo lenguaje kotlin para realizar las vistas.

Ventajas y Desventajas al usar XML

Ventajas:

1. **Amplia compatibilidad**: XML ha sido la forma estándar de definir interfaces en Android por años, lo que significa que muchos desarrolladores están familiarizados con su uso.

2. **Estabilidad**: Al ser más antigua, XML es muy estable y ampliamente soportada, lo que la hace adecuada para proyectos grandes y maduros.

3. **Más recursos y documentación**: Dado que XML ha sido el estándar durante mucho tiempo, hay una gran cantidad de recursos y ejemplos disponibles en línea.

4. **Separación clara de lógica y UI**: La UI y la lógica de negocio están separadas, lo que hace más fácil mantener el código limpio y organizado.

Desventajas:

1. **Verboso y repetitivo**: Definir la UI en XML puede ser tedioso, especialmente cuando se requieren muchas vistas anidadas o elementos complejos.

2. **Manejo de estado complicado**: El manejo de cambios en el estado de la UI requiere de mucho código adicional, y puede ser más difícil de gestionar cuando se tiene que actualizar la interfaz de manera dinámica.

3. **Menos flexible**: XML se basa en un conjunto de componentes predefinidos (widgets), por lo que hacer interfaces dinámicas o altamente personalizadas puede ser más desafiante.

Ventajas y Desventajas al usar JetPack Compose

Ventajas:

1. **Menos código**: Compose permite escribir menos código comparado con XML.
.
2. **Actualización automática del UI**: El UI se actualiza de manera automática cuando el estado cambia, lo que hace que la gestión del estado sea mucho más sencilla.

3. **Integración con Kotlin**: Compose se integra de manera natural con Kotlin, aprovechando sus características como funciones de extensión, lambdas y flujo de datos reactivo.

4. **Mayor flexibilidad**: Compose permite un alto grado de personalización, lo que facilita la creación de interfaces complejas sin necesidad de recurrir a soluciones complicadas.

5. **Soporte para previsualización en tiempo real**: Con el **Live Preview** de Android Studio, puedes ver el diseño de la interfaz en tiempo real mientras escribes el código, lo que acelera el proceso de desarrollo.

Desventajas:

1. **Curva de aprendizaje**: A pesar de su simplicidad, Compose introduce una nueva forma de pensar sobre las interfaces de usuario, lo que puede ser un desafío para los desarrolladores acostumbrados a XML.

2. **Compatibilidad limitada en proyectos antiguos**: En aplicaciones más antiguas, integrar Compose en un proyecto que ya usa XML puede ser difícil.

3. **Falta de herramientas visuales completas**: Aunque el Live Preview es útil, todavía no tiene la misma capacidad que el editor visual de XML, lo que puede hacer que sea más difícil para algunos desarrolladores visualizar y ajustar la interfaz.

4. **Evolución constante**: Jetpack Compose sigue evolucionando, lo que puede causar que algunos patrones de diseño cambien rápidamente o que algunos bugs sean comunes en nuevas versiones.

Ruta de aprendizaje Desarrollador Android

Esta lista que les presento a continuación, son temas que bajo mi experiencia considero que debe conocer un estudiante que quiera laborar como desarrollador android a nivel profesional

1. **Profundizar en el lenguaje Kotlin**
2. **Profundizar en XML y aprender jetpack Compose**
3. **Profundizar en Corrutinas**
4. **Profundizar en base de datos locales con room y servicios de firebase**
5. **Profundizar en consumo de servicios**
6. **Dominar la arquitectura MVVM**
7. **Inyección de dependencias**
8. **Ambientes de Desarrollo (Development, Debug, Laboratorio, Release, Production)**
9. **Pruebas Unitarias**
10. **Ofuscado de código con proguard o R8**
11. **Subir aplicaciones a google play**
12. **Uso de git flow**
13. **Aprender a usar Jenkins**
14. **Uso de Sonar**
15. **Kotlin multiplataforma (El futuro del desarrollo android !!)**

Aprendamos un poco sobre JepackCompose …

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.bodyLarge,
                )
            }
        }
    }
}
```

Bibliografía:

- https://developer.android.com/compose
- https://developer.android.com/codelabs/jetpack-compose-basics?hl=es-419#0