Fundamentos del Desarrollo Móvil

1. Sistemas Operativos Móviles y Arquitecturas

Android (Google): Basado en el kernel de Linux. Su arquitectura se divide en capas: Kernel, HAL (Capa de abstracción de hardware), Librerías nativas/Runtime, Framework de aplicaciones y Aplicaciones de sistema.
iOS (Apple): Basado en Unix (Darwin). Su arquitectura (Cocoa Touch) tiene capas: Core OS, Core Services, Media y Cocoa Touch.

2. Nativas vs Multiplataforma

- **Nativas:** Desarrolladas específicamente para un SO (Swift para iOS, Kotlin para Android). Mayor rendimiento y acceso total al hardware.
- **Multiplataforma:** Un solo código funciona en ambos (Flutter, React Native). Desarrollo más rápido pero a veces menor rendimiento.

3. Hardware de Dispositivos

Los componentes clave que limitan o potencian una app son: CPU (Arquitectura ARM), GPU, Memoria RAM (limitada en móviles), Sensores (GPS, Giroscopio), Batería (consumo crítico) y Pantalla táctil.

4. Lenguajes de Programación

- **Android Nativo:** Kotlin (oficial), Java.
- **iOS Nativo:** Swift (moderno), Objective-C (legado).
- **Multiplataforma:** Dart (Flutter), JavaScript (React Native/Ionic), C# (Xamarin).

5. Entornos de Desarrollo (IDE)

- **Android Studio:** El oficial para Android. Basado en IntelliJ IDEA.
- **Xcode:** El oficial para iOS (solo funciona en Mac).
- **VS Code:** Muy popular para tecnologías web y multiplataforma como Flutter o React Native.

6. Diseño de Interfaces (UI)

Se utilizan "Controles" o "Widgets" (botones, listas, inputs). El diseño debe ser responsivo para adaptarse a diferentes tamaños de pantalla y orientaciones.

7. Guías de Estilo

- **Material Design (Google):** Se basa en superficies, sombras y movimiento realista (papel y tinta).
- **Human Interface Guidelines (Apple):** Se enfoca en la claridad, la deferencia (el contenido es rey) y la profundidad.

8. Arquitecturas y Patrones de Diseño

Para mantener el código ordenado se usan patrones como:

- **MVC:** Modelo - Vista - Controlador.
- **MVVM:** Modelo - Vista - VistaModelo (Estándar moderno en Android/iOS).
- **Clean Architecture:** Separación estricta por capas para hacer la app testeable y mantenible.

9. Control de Versiones

Herramientas esenciales para trabajar en equipo:

- **Git:** El sistema de control de versiones.
- **GitHub / GitLab:** Plataformas para alojar el código en la nube.

10. Persistencia y Acceso a Datos

Formas de guardar datos en el móvil:

- **Shared Preferences / UserDefaults:** Para datos pequeños (configuraciones).
- **Bases de Datos Locales:** SQLite, Room (Android), CoreData (iOS), Realm.

- **Archivos:** Guardar imágenes o documentos directamente en el almacenamiento interno.

Descargar Tema en PDF