UT1: Tecnologías móviles

1) Explica la evolución de la red de comunicación móvil y las principales diferencias entre cada generación.

Generación	Tecnología	Servicios	Características
1G	Analógica	Solo voz	• Llamadas básicas con baja calidad de sonido • Baja seguridad (fácil de interceptar) • Cobertura limitada
2G	Digital (GSM, CDMA)	Voz +SMS (Mensajes de texto)	• Mejor calidad de llamada • Aparición de SMS y MMS • Mayor seguridad en la comunicación • Velocidad baja
3G	UMTS, HSPA	Voz + Datos (Internet Móvil)	• Internet móvil con velocidades de hasta varios Mbps • Videollamadas • Mejora en la transmisión multimedia (música, imágenes) • Inicia la verdadera "conectividad móvil"
4G	LTE (Long Term Evolution)	Datos de alta velocidad + VoIP	• Velocidades desde decenas hasta cientos de Mbps • Streaming de video en HD • Juegos online de tiempo real • Redes totalmente basadas en IP (voz sobre datos, VoLTE) • Nace la experiencia de "smartphone" como la conocemos hoy
5G	NR (New Radio)	Datos Ultrarrápidos, IoT, baja latencia	• Velocidades de hasta 10 Gbps • Latencia muy baja (<1 ms) clave para tiempo real • Soporte masivo para dispositivos IoT • Aplicaciones en realidad aumentada/virtual, vehículos autónomos, telemedicina, ciudades inteligentes

2) Android Vs iOS

a) Explica el modelo de capas de ambos Sistemas Operativos

Android (basado en Linux)

1. ***Linux Kernel***
    - *Base del sistema. Maneja hardware, memoria, procesos, drivers, seguridad.*
2. ***Capa de Bibliotecas Nativas (Libraries)***
    - *Incluye librerías en C/C++ como OpenGL, WebKit, SQLite, etc.*
    - *Se usa para gráficos, bases de datos, navegador, etc.*
3. ***Android Runtime (ART, antes Dalvik VM)***
    - *Ejecuta las aplicaciones Android (archivos .apk).*
    - *Proporciona la máquina virtual optimizada para dispositivos móviles.*
4. ***Application Framework***
    - *Conjunto de APIs que permiten a los desarrolladores acceder a funciones como cámara, GPS, contactos, notificaciones.*
5. ***Aplicaciones***
    - *Apps del sistema (teléfono, contactos, navegador) + apps de usuario descargadas de la Play Store.*

---

iOS (basado en Darwin/Unix de Apple)

1. ***Core OS (Núcleo)***
    - *Basado en XNU (híbrido de Mach + BSD). Maneja seguridad, memoria, procesos y comunicación con hardware.*
2. ***Core Services***
    - *Ofrece servicios básicos como Core Foundation, SQLite, redes, multitarea, servicios de ubicación.*
3. ***Media Layer***
    - *Maneja gráficos, audio y video (OpenGL ES, Metal, Core Audio, Core Animation).*

4. ***Cocoa Touch***
    - *Framework de más alto nivel, diseñado para la interacción con el usuario (UI, gestos multitáctiles, notificaciones).*
5. ***Aplicaciones***
    - *Apps nativas de iOS + apps de usuario descargadas desde la App Store.*

B) INDICA EL NOMBRE DEL PAQUETE DE INSTALACIÓN DE CADA SISTEMA OPERATIVO Y EL NOMBRE DE LA TIENDA OFICIAL EN EL QUE PUEDEN DESCARGARSE ESTE TIPO DE APLICACIONES.

ANDROID:

- **PAQUETE:** .APK (ANDROID PACKAGE KIT).
- **TIENDA OFICIAL: GOOGLE PLAY STORE**.

IOS:

- **PAQUETE:** . IPA (IOS APP STORE PACKAGE).
- **TIENDA OFICIAL: APPLE APP STORE**.

C) ¿QUÉ LENGUAJES DE PROGRAMACIÓN SE USAN PARA EL DESARROLLO DE APLICACIONES PARA CADA SO?

ANDROID:

- PRINCIPALMENTE **JAVA** Y **KOTLIN** (LENGUAJE OFICIAL RECOMENDADO POR GOOGLE).
- TAMBIÉN SE PUEDEN USAR **C/C++** CON NDK, Y OTROS FRAMEWORKS COMO FLUTTER (DART) O REACT NATIVE (JAVASCRIPT).

IOS:

- PRINCIPALMENTE **SWIFT** (LENGUAJE OFICIAL RECOMENDADO POR APPLE).
- TAMBIÉN SE PUEDE USAR **OBJECTIVE-C** (LENGUAJE MÁS ANTIGUO).
- ADEMÁS, FRAMEWORKS MULTIPLATAFORMA COMO FLUTTER (DART) O REACT NATIVE (JAVASCRIPT).

3) EN EL CASO DE ANDROID, INDICA QUÉ ES ART Y QUÉ ES DALVIK.

DALVIK

- FUE LA **MÁQUINA VIRTUAL ORIGINAL DE ANDROID** (HASTA ANDROID 4.4 KITKAT).
- SE ENCARGABA DE **EJECUTAR LAS APLICACIONES ANDROID**.

- Cada app se ejecutaba en su propia máquina virtual Dalvik, garantizando aislamiento y seguridad.
- Usaba el formato de bytecode **DEX (Dalvik Executable)**, más eficiente en memoria que el bytecode normal de Java.
- Funcionaba con **JIT (Just-In-Time compilation)** → el código se compilaba "justo cuando" se necesitaba.
    - Ventaja: menor uso de espacio en disco.
    - Inconveniente: más consumo de batería y menor rendimiento en tiempo de ejecución.

ART (Android Runtime)

- Sustituyó a Dalvik a partir de **Android 5.0 Lollipop (2014)**.
- También ejecuta las apps Android, pero con una diferencia clave:
- Usa **AOT (Ahead-Of-Time compilation)** → las apps se **compilan en código nativo al instalarse**.
    - Ventaja: mayor rendimiento, menor consumo de batería.
    - Inconveniente: ocupa más espacio en disco, porque el código ya está precompilado.
- En versiones recientes, ART combina **AOT + JIT + perfiles** para optimizar aún más la ejecución.

4) Explica las diferencias a la hora de desarrollar aplicaciones entre los siguientes acrónimos: UX, UI e IxD

1. UX (User Experience – Experiencia de Usuario)

- Se centra en **cómo se siente el usuario** al interactuar con la aplicación.
- Busca que la experiencia sea **fácil, intuitiva, eficiente y satisfactoria**.
- Incluye:
    - Flujo de navegación.
    - Arquitectura de la información.
    - Usabilidad y accesibilidad.
    - Investigación de usuarios (qué necesitan, cómo piensan).

👉 Ejemplo: que en una app de compras el proceso de pago sea rápido, claro y sin confusiones.

---

2. UI (User Interface – Interfaz de Usuario)

- Se refiere a la **parte visual y estética** de la aplicación.
- Involucra lo que el usuario **ve y toca**: botones, menús, tipografías, colores, iconos, imágenes.
- El objetivo es que la interfaz sea **atractiva y coherente**, además de funcional.

👉 Ejemplo: el diseño del botón "Comprar" (su color, tamaño y posición en pantalla).

---

3. IxD (Interaction Design – Diseño de Interacción)

- Se enfoca en **cómo interactúa el usuario con la aplicación** y cómo esta responde.
- Busca crear una **comunicación fluida** entre el usuario y el sistema.
- Incluye:
    - Animaciones y transiciones.
    - Respuestas visuales o sonoras al pulsar un botón.
    - Gestos táctiles (deslizar, pellizcar, mantener pulsado).
    - Microinteracciones (notificaciones, vibración, feedback instantáneo).

👉 Ejemplo: que al deslizar hacia abajo en una app de noticias se actualice la lista con una animación.

5) Realiza una comparativa, indicando ventajas e inconvenientes de los siguientes frameworks.

a) Xamarin b) Flutter c) React Native d) Ionic e) NativeScript

Framework	**Lenguaje**	**Rendimiento**	**Comunidad**	**Uso recomendado**
Xamarin	C#	Alto	Media	Apps empresariales en ecosistema Microsoft

Flutter	Dart	Muy alto	Alta	Apps con UI personalizable y alto rendimiento
React Native	JavaScript / TS	Alto	Muy alta	Startups, apps multiplataforma rápidas, ecosistema JS
Ionic	HTML, CSS, JS	Medio	Alta	PWAs, apps híbridas simples y rápidas de desarrollar
NativeScript	JS / TS / Angular	Alto	Media	Apps nativas con acceso directo a APIs en JS

De forma más breve la resolución de para que se usa cada uno son:

- **Mejor rendimiento** → Flutter
- **Más popular** → React Native
- **Más fácil para web developers** → Ionic
- **Más orientado a Microsoft** → Xamarin
- **Acceso directo a APIs nativas con JS** → NativeScript

6) ¿Qué tienen en común los frameworks anteriormente mencionados?

Las características que comparten dichos frameworks son:

- **Son frameworks multiplataforma** → permiten desarrollar una única base de código para desplegar en **Android e iOS** (algunos incluso en web y escritorio).
- **Ahorro de tiempo y costes** → evitan tener que desarrollar dos apps nativas independientes (una en Swift/Objective-C y otra en Java/Kotlin).
- **Acceso a funcionalidades nativas** → mediante plugins, librerías o bridges pueden usar cámara, GPS, notificaciones, sensores, etc.

- **Reutilización de código** → gran parte del código se comparte entre plataformas, aunque la UI puede necesitar ajustes.
- **Soporte para hot reload / live reload** → permiten ver cambios en la app de forma casi inmediata, agilizando el desarrollo.
- **Amplia comunidad y ecosistema** → todos cuentan con documentación, librerías y paquetes que aceleran el trabajo.
- **Enfoque en la experiencia de usuario** → aunque varían en nivel, todos buscan que las aplicaciones se comporten lo más parecido posible a las nativas.

7) Enumera, quitando a iOs y Android, al menos 3 sistemas operativos móviles, indicando el año en el que debutaron.

- **Symbian OS** → 1997 (muy popular en los primeros smartphones de Nokia).
- **Windows Phone** → 2010 (desarrollado por Microsoft, sucesor de Windows Mobile).
- **BlackBerry OS** → 1999 (usado en los dispositivos BlackBerry, centrados en el entorno empresarial).

He puesto otros tres más porque no se los que van a caer en el cuestionario con lo que pienso que cuantos más tenga más posibilidades de saber los que van a caer:

- **WebOS** (2009, creado por Palm).
- **Firefox OS** (2013, impulsado por Mozilla).
- **Tizen OS** (2012, desarrollado por Samsung e Intel).

8) Investiga la proporción de SO móviles según su uso en la actualidad.

En este caso existen dos posibilidades de porcentajes, a nivel mundial o a nivel nacional, con lo que me dio la curiosidad y busqué las dos:

En caso mundial, Android se usa en un porcentaje aproximado del 72% mientras que iOS lo usaría una población estimada del 28% del mundo. En cambio, en España no cambian mucho los datos ya que los españoles mientras unos tantos usan iOS lo que refleja el 24%, en el sector de Android representan un 76%

9) Explica qué es Kotlin y para qué se utiliza

Kotlin es un lenguaje de programación moderno, creado por JetBrains en 2011 y adoptado oficialmente por Google en 2017 como lenguaje preferido para el desarrollo en Android.

¿Para qué se utiliza?

- Principalmente para desarrollar aplicaciones móviles Android.
- También se usa en backend (con frameworks como Ktor o Spring), aplicaciones de escritorio y hasta en proyectos multiplataforma (Kotlin Multiplatform).

10) Indica la importancia del uso de IDEs a la hora de desarrollar aplicaciones móviles.

La importancia de los IDEs (Entornos de Desarrollo Integrado) en el desarrollo de aplicaciones móviles radica en que:

- Centralizan herramientas: permiten programar, compilar, depurar y probar en un mismo entorno.
- Aumentan la productividad gracias a funciones como autocompletado, resaltado de sintaxis y refactorización automática.
- Simulación y pruebas: integran emuladores de dispositivos móviles para verificar la app sin necesidad de un móvil físico.
- Integración con SDKs y frameworks: facilitan el trabajo con Android, iOS u otros entornos multiplataforma.
- Gestión de errores y control de versiones: permiten detectar fallos rápidamente y trabajar en equipo usando Git u otras herramientas.