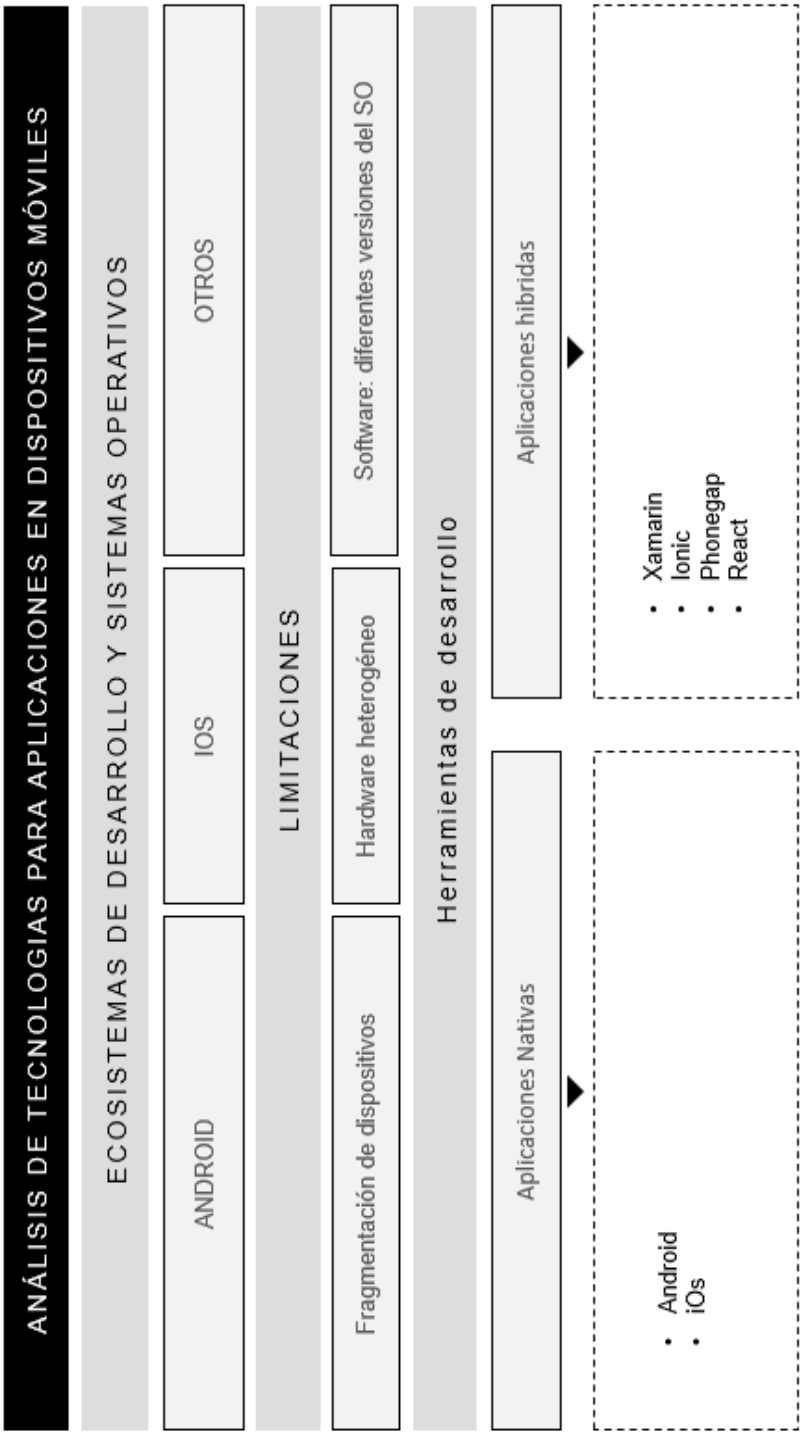
Programación multimedia y dispositivos móviles

Análisis de tecnologías para aplicaciones en dispositivos móviles

Índice

Esquema


ANÁLISIS DE TECNOLOGIAS PARA APLICACIONES EN DISPOSITIVOS MÓVILES
ECOSISTEMAS DE DESARROLLO Y SISTEMAS OPERATIVOS
ANDROID
IOS
OTROS
LIMITACIONES
Fragmentación de dispositivos
Hardware heterogéneo
Software: diferentes versiones del SO
Herramientas de desarrollo
Aplicaciones Nativas
Aplicaciones hibridas
Android
iOs
Xamarin
Ionic
Phonegap
React

Material de estudio

En el transcurso de este tema se realizará una introducción al mundo del desarrollo de aplicaciones móviles, analizando sus características y componentes principales. En particular nos centraremos en el ecosistema Android, analizando los entornos de desarrollo y emuladores como el desarrollo y ciclo de vida de las aplicaciones Android.

Durante el desarrollo de este tema se deben conseguir los siguientes objetivos:

- Comprender los conceptos básicos relacionados con las aplicaciones en dispositivos móviles.
- Aprender las limitaciones de estas aplicaciones.
- Conocer qué posibilidades tecnológicas tenemos a nuestra disposición.
- Conocer los diferentes IDES para el desarrollo de apps en dispositivos móviles.
- Conocer los emuladores que nos van a servir para veri car nuestra aplicación.
- Aprender a instalar y configurar Android Studio como entorno de desarrollo para aplicaciones Android

1.1. Aplicaciones en dispositivos móviles: conceptos, limitaciones y tecnologías disponibles

Actualmente hay a nuestra disposición multitud de dispositivos móviles, desde teléfonos hasta tabletas, pasando por reproductores multimedia, etc.

Estas son las características comunes a estos dispositivos:

- Tamaño pequeño: Son aparatos que se puede transportar fácilmente y utilizar tanto con una, como con las dos manos.
- Movilidad: Los dispositivos pueden utilizarse sin necesidad de estar conectados físicamente a una fuente de energía. También poseen movilidad en sus datos, pudiendo modificarse mientras se encuentran en distintos lugares.
- Conexión: Los dispositivos tienen posibilidad de conectarse a Internet o distintas redes, de manera continua o intermitente. También pueden conectarse entre si y a otros dispositivos auxiliares con distintos protocolos, como, por ejemplo, Bluetooth.
- Capacidades: Tienen capacidad de procesamiento mediante CPU multinúcleo, capacidad de memoria (RAM, tarjetas MicroSD, flash, etc.) Además, pueden tener cámaras, GPS, acelerómetros y otros sensores o capacidades.
- Interacción: Poseen una alta capacidad de interacción, habitualmente a través de una pantalla táctil o de algún otro método como, por ejemplo, el teclado.
- Uso: Lo habitual es que sean de uso individual y, en muchos casos, de una única persona.

Limitaciones

Todos estos nuevos dispositivos y tecnologías también tienen sus limitaciones, que vamos a enumerar.

1. Dispositivos: Uno de los problemas que nos encontramos a la hora de desarrollar una aplicación para un dispositivo móvil es la fragmentación de dispositivos, es decir, que existen infinidad de tipos de dispositivos, marcas y, dentro de las marcas, también modelos variados. Cada uno de ellos tendrá distintas características, siendo el tamaño y la resolución de la pantalla las características que más quebraderos de cabeza suelen dar.

2. Hardware: Derivada de esta fragmentación de dispositivos que mencionamos, nos encontramos con las características técnicas, tales como el procesador, o las posibilidades de conexión y sobre todo memoria. Todas estas características harán que una aplicación pueda ir muy lenta en el dispositivo, o que, directamente, no sea capaz de ejecutarse.

3. Software: La versión del sistema operativo que tenga instalado el dispositivo móvil influirá en las posibilidades que podrá tener la aplicación y, en el caso más extremo, incluso en el hecho de que sea posible o no la instalación.

Tecnologías disponibles

A lo largo de estos años hemos podido conocer distintos sistemas operativos para distintos dispositivos móviles. Algunos de ellos han desaparecido prácticamente y otros, o están a punto de hacerlo, o tienen un uso muy residual.

Veamos algunos de los Sistemas Operativos más importantes:

1. Android

Android es un sistema operativo diseñado por Google, basado en el kernel (núcleo) de Linux. Fue implementado principalmente para dispositivos móviles, tabletas y actualmente también para relojes, televisores y automóviles.

Si deseamos crear una aplicación para nuestro smartphone o para la tienda Play Store, el SDK de Android nos ofrece un conjunto de herramientas de desarrollo. Tiene un depurador de Tecnología disponible código, biblioteca, un simulador (emulador) de teléfono basado en QEMU, documentación, y muchos ejemplos de código.

La web oficial Android Developer nos ofrece toda la información para poder empezar a desarrollar apps para dispositivos Android. Pagando una única cuota podemos registrarnos para obtener una cuenta de desarrollador de Play Store.

2. iOS

iOS es un sistema operativo móvil desarrollado por Apple Inc. para sus dispositivos, incluyendo el iPhone, iPad y iPod Touch. Fue lanzado por primera vez en 2007 junto

con el primer iPhone, y desde entonces ha sido uno de los sistemas operativos más populares del mundo.

iOS permite a los usuarios descargar y comprar aplicaciones a través de la App Store, que es administrada por Apple. Esta tienda ofrece millones de aplicaciones, desde juegos hasta herramientas de alta productividad.

iOS está diseñado para integrarse perfectamente con otros productos de Apple, como el Apple Watch, Apple TV y Mac, permitiendo una experiencia de usuario fluida y sincronizada entre dispositivos. Apple controla estrictamente el desarrollo de aplicaciones para iOS, utilizando su entorno de desarrollo Xcode y el lenguaje de programación Swift, lo que garantiza un alto nivel de calidad y rendimiento en las aplicaciones. La información completa para convertirnos en desarrolladores Apple la podemos encontrar en developer.apple.com, donde conseguir el SDK es gratis. Para poder publicar nuestras aplicaciones en la App Store necesitamos una cuenta de desarrollador, hay dos opciones: la estándar, por 99 dólares (al año) y la de empresas, por 299.

3. Otros

Windows Phone: Este es el sistema operativo que Microsoft utiliza en los smartphones y en otros dispositivos móviles. En 2010, Microsoft inauguró la nueva plataforma para smartphones, conocida como Windows Phone 7. La última actualización es el sistema operativo Windows 10 Mobile. Aunque actualmente se mantiene en algunos dispositivos y con actualizaciones de seguridad, muchas aplicaciones están desapareciendo de su ecosistema de apps.

Amazon Fire OS: El S.O. Amazon Fire está basado en el sistema operativo de Android. Amazon es su productor y está específicamente diseñado para el Fire Phone de Amazon y el rango de tabletas de Kindle Fire. Este sistema operativo se enfoca, principalmente, en el consumo de contenidos. Viene con una interfaz de usuario fortificada y está hecho a medida para hacer disponible el contenido de las tiendas y servicios de Amazon.

También podemos nombrar otros que, o bien o han desaparecido, han pasado a ser

mantenidos por comunidades de usuarios, o son intentos de fabricantes por crear sus propios SO y desvincularse de Google:

- Firefox OS.
- Ubuntu Touch.
- Symbian OS.
- Tizen.
- HarmonyOS.

1.2. Introducción al desarrollo de aplicaciones móviles

El desarrollo de aplicaciones móviles es el proceso en el que un software para dispositivos móviles es desarrollado para realizar una determinada tarea.
Estas aplicaciones pueden venir preinstaladas en los dispositivos, o bien ser descargadas por los usuarios desde las distintas plataformas móviles (Android, iOS, Windows Phone).

¿Qué es una aplicación móvil?
Una aplicación móvil o app, es un tipo de software diseñado específicamente para ejecutarse en dispositivos móviles, como teléfonos inteligentes o tabletas. Aunque generalmente son programas pequeños con funciones específicas, logran ofrecer a los usuarios servicios y experiencias satisfactorias de alta calidad.

En contraposición de las aplicaciones de escritorio, las apps móviles no forman parte de sistemas de software integrados. En su lugar, cada app móvil ofrece una funcionalidad específica. Por ejemplo, podría ser un juego, un block de notas o un navegador web adaptado para dispositivos móviles.

En los primeros dispositivos móviles, las apps evitaban la multifuncionalidad debido a las limitaciones de hardware. Sin embargo, a pesar de que los dispositivos actuales son mucho más avanzados y potentes, las aplicaciones móviles continúan siendo especializadas en sus funciones. Esto permite a los usuarios elegir exactamente qué funciones desean en sus dispositivos.

Aplicaciones nativas

El desarrollo de aplicaciones nativas supone utilizar el lenguaje nativo de la propia plataforma para la construcción:

- Android: Kotlin (hasta 2019 era Java, se puede seguir usando, pero Google recomienda el uso de Kotlin para proyectos nuevos).
- iOS: Switf

Ventajas de las aplicaciones nativas:

- Estamos trabajando directamente con las funciones y SDK oficial del sistema operativo, por lo que no tendremos ningún límite en esa plataforma.
- La experiencia de usuario y el entorno visual ofrecen un rendimiento del 100%, ya que se hace uso directamente de las funciones visuales y los elementos gráficos que ofrece cada sistema.
- Podemos acceder a todo tipo de sensores y hardware del dispositivo, así como conectarnos a hardware externo.
- Podemos acceder a todo tipo de funcionalidades del dispositivo.

Aplicaciones híbridas

El desarrollo de una aplicación sencilla o que use elementos comunes en las distintas plataformas, puede ser realizado a través de un framework de manera híbrida. De esta manera, se realiza un solo desarrollo, que será exportado a distintas plataformas abaratando su creación, debido a la necesidad de menos personas y menos tiempo.

Existe una gran variedad de librerías (frameworks) que nos permiten realizar este desarrollo híbrido. Algunas de las más utilizadas son:

1. Xamarin. Es una herramienta que utilizan los programadores para crear aplicaciones móviles (Apps). Permite crear código en el lenguaje C# de Microsoft, el mismo que permite traducir el código del lenguaje para diferentes dispositivos, tanto Android, como IOS y Windows Phone.

2. Ionic. Es el framework con el que puedes utilizar CSS para crear increíbles diseños que parezcan nativos. Pero el verdadero potencial de Ionic es lo sincronizado que está con Angular JS. Otra gran ventaja es la interfaz de línea de comandos, que está llena de interesantes características, como emuladores integrados y una app packager basada en Cordova.

3. Phonegap. Es la herramienta por excelencia de Adobe para desarrollar aplicaciones móviles. Es una distribución de código abierto de Cordova framework, que puede reutilizar las habilidades de desarrollo web existentes para crear rápidamente aplicaciones híbridas creadas con HTML, CSS y JavaScript para múltiples plataformas con una única base de código, para que pueda llegar a su audiencia sin importar su dispositivo.

4. React Native. Nació en 2015, creado por Facebook. Basado en React, utiliza JSX para desarrollar y renderizar las aplicaciones. React Native no permite utilizar el mismo código para todas las plataformas, sino que ofrece componentes para cada una de ellas y deja a los desarrolladores la responsabilidad de ver qué componentes se asemejan más al comportamiento nativo que se esperaría en cada caso.

Las aplicaciones híbridas son esencialmente páginas web incrustadas en una aplicación móvil a través de un WebView. Pueden ejecutar el mismo código, independientemente de cuál sea la plataforma objetivo, aunque también cabe la posibilidad de usar código específico para cada plataforma objetivo.

Pueden acceder a alguna capa nativa de los dispositivos (cámara, GPS, etc.) usando plugins. Aunque en algunos casos no es posible, o requiere desarrollos independientes para cada sistema.

1.3. Entornos de desarrollo y emuladores en Android

Android Studio es el IDE oficial de Google en el desarrollo de aplicaciones para Android, actualmente soporta los lenguajes de Java y Kotlin, siendo este último el estándar actual y el recomendado por Google para desarrollar los nuevos proyectos.

Para poder ejecutar Android Studio en nuestro ordenador necesitamos cumplir con unos requisitos mínimos. La cantidad de memoria RAM recomendado es de 8 GB, nuestro procesador debe tener capacidades para virtualizar (emulador).

Una de las funciones que nos ofrece Android Studio es la posibilidad de montar una máquina virtual de Android con la que poder hacer uso del sistema operativo de Google, y de sus aplicaciones, desde nuestro ordenador.

Esta versión ha supuesto para Google un gran desafío, ya que la compañía ha trabajado duro en implementar una serie de novedades, mejoras y cambios que los usuarios llevaban tiempo pidiendo y que, por motivos técnicos, no se habían podido implementar antes.

Android Emulator ejecuta una pila completa de sistema de Android, hasta el nivel del kernel, que incluye un conjunto de apps preinstaladas (como el teléfono) a las que puedes acceder desde tus apps. Puedes elegir la versión del sistema Android que deseas ejecutar en el emulador al crear AVD.

Las imágenes de sistema de Android disponibles a través del Administrador de AVD contienen código para el kernel de Linux de Android, las bibliotecas nativas, la VM y

los diferentes paquetes de Android (por ejemplo, las apps preinstaladas y el framework de Android).

Hay que tener en cuenta que desde el emulador no son accesibles todas las características del dispositivo, y que no incluye hardware virtual para lo siguiente:

- Wifi.
- Bluetooth.
- NFC.
- Inserción/expulsión de tarjetas SD.
- Auriculares conectados a dispositivos.
- USB.

Además de Android Studio, podemos utilizar otros IDEs para crear aplicaciones en Android. El más famoso y valorado es IntelliJ IDEA de la compañía JetBrains, la misma compañía que desarrolla actualmente Android Stuido.

IntelliJ IDEA a diferencia de Android Studio, es un IDE de propósito general que soporta los lenguajes de Java y Kotlin, al que podemos añadir plugins o extensiones para administrar y desarrollar proyectos Android.

1.4. Desarrollo de aplicaciones móviles en Android

Un elemento imprescindible para el desarrollo de Android es el uso del SDK (el acrónimo de Software Development Kit - Kit de desarrollo de software). Se trata de una serie de API's que facilita el desarrollo de las aplicaciones móviles. Algunas de las **características** que posee son:

- Licencias, distribución y desarrollo gratuitos; tampoco hay procesos de aprobación del software.
- Acceso al hardware de WiFi, GPS, Bluetooth y telefonía, permitiendo realizar y recibir llamadas y SMS.
- Control completo de multimedia, incluyendo la cámara y el micrófono.

- APIs para los sensores: acelerómetros y brújula.
- Mensajes entre procesos (IPC).
- Almacenes de datos compartidos, SQLite, acceso a SD Card.
- Aplicaciones y procesos en segundo plano.
- Widgets para la pantalla de inicio (escritorio).
- Integración de los resultados de búsqueda de la aplicación con los del sistema.
- Uso de mapas y sus controles desde las aplicaciones.
- Aceleración gráfica por hardware, incluyendo OpenGL ES 2.0 para los 3D.
- Controles de Google Maps en nuestras aplicaciones.
- Procesos y servicios en segundo plano.
- Proveedores de contenidos compartidos y comunicación entre procesos.
- No diferencia entre aplicaciones nativas y de terceros; todas se crean igual, con el mismo aspecto y con las mismas posibilidades de usar el hardware y las APIs.
- Widgets de escritorio.

Android Studio dispone de una herramienta de gestión, SDK Manager que nos permitirá instalar y administrar diferentes versiones del SDK dependiendo la versión de Android que queramos utilizar.

Debemos de tener en cuenta que las versiones más modernas poseen características que quizás no encontremos en versiones anteriores, pero, a su vez, habrá menos dispositivos donde podamos instalar nuestra aplicación. Este punto es muy importante cuando generamos un proyecto, debemos de tener un balance de no coger un SDK mínimo muy antiguo que nos limítela funcionalidad ni uno muy moderno que nos limite el numero de dispositivos compatibles con la aplicación.

Capas en el sistema operativo

Los sistemas operativos implementan multitud de servicios y funciones, como la gestión de entrada y salida, control de los programas, la gestión de la memoria, entre otros.

Estos procesos se encuentran en capas, construyendo una jerarquía de niveles de

abstracción, de modo que cada nivel proporciona un conjunto específico de funciones primitivas que podrán usar las funciones de la capa superior.

1. Aplicaciones: está compuesta por las aplicaciones nativas y por las de terceros, así como las de desarrollo.
2. Framework de aplicaciones: está compuesto por las clases que se utilizan para crear aplicaciones Android: actividades, servicios, views, proveedores de contenidos, etc.
3. Android runtime: Es lo que hace a Android diferente de una distribución de Linux embebido. Está compuesto por las librerías "core" (núcleo) y por Dalvik, la máquina virtual de Java, basada en registros, y que cuenta con el núcleo de Linux para la gestión de hilos y para el manejo de memoria a bajo nivel.
4. Librerías: compuesta por las librerías generales.
5. Núcleo de Linux: es la capa encargada de los controladores (drivers) del hardware, los procesos, la memoria, seguridad, red y gestión de energía. Es la capa que abstrae el resto de las capas del hardware.

Las clases más importantes para el desarrollo de aplicaciones en Android son:

- ActivityManager: controla el ciclo de vida de las actividades.
- View: se usa para construir interfaces en las actividades.
- NotificationManager: mecanismo para mostrar avisos al usuario.
- ContentProvider: permite intercambiar datos de una manera estandarizada.
- Resource Manager: permite usar en la aplicación recursos que no forman parte del código, como XML, recursos gráficos, audio, vídeo, etc.

Android Manifest

El fichero AndroidManifest.xml es un fichero que se encuentra en todos los proyectos de Android, su función es declarar una serie de metadatos de la aplicación que el dispositivo debe conocer antes de instalarla, describe información esencial sobre tu app para las herramientas de compilación de Android, el sistema operativo Android y Google Play.

En este fichero se indican aspectos como los siguientes:

- El nombre del paquete Java para la aplicación. El nombre del paquete sirve como un identificador único.
- Los componentes de la aplicación, como las actividades, los servicios, los receptores de mensajes y los proveedores de contenido que la integran. También nombra las clases que implementa cada uno de los componentes y publica sus capacidades, como los mensajes Intent con los que pueden funcionar. Estas declaraciones notifican al sistema Android los componentes y las condiciones para el lanzamiento.
- Los procesos que alojan los componentes de la aplicación.
- Los permisos que debe tener la aplicación para acceder a las partes protegidas de una API e interactuar con otras aplicaciones. Este archivo también declara los permisos que otros deben tener para interactuar con los componentes de la aplicación.
- Las clases Instrumentation que proporcionan un perfil y otra información mientras la aplicación se ejecuta. Estas declaraciones, enumeradas en el archivo, están en el manifiesto solo mientras la aplicación se desarrolla y se eliminan antes de su publicación.
- El nivel mínimo de Android API que requiere la aplicación.
- Las bibliotecas con las que debe estar vinculada la aplicación.

1.5. Ciclo de vida de una aplicación en Android

Cuando se empieza a programar en un lenguaje como Java, lo primero que se enseña es el método main(), que es el punto al que llamará el sistema operativo cuando vayamos a arrancar nuestra aplicación.

En Android no existe un método main() como tal, pero sí existen varios métodos de nuestra actividad que serán llamados por el sistema operativo cuando ocurran eventos importantes.

La clase Activity es un componente crucial de una aplicación Android, y la forma en que se inician y se organizan las actividades es una parte fundamental del modelo de aplicación de la plataforma.

El ciclo de vida de una aplicación Android se refiere a los estados y transiciones que atraviesa una aplicación desde que se inicia hasta que se cierra. Comprender este ciclo es crucial para desarrollar aplicaciones eficientes y evitar problemas de rendimiento o pérdida de datos. A continuación, podemos ver los diferentes estados y métodos clave que forman parte del ciclo de vida de una Activity en una aplicación Android:

Estados del ciclo de vida de una Activity

Una Activity es una sola pantalla con una interfaz de usuario en una aplicación Android. A lo largo de su ciclo de vida, una Activity puede estar en varios estados:

- Running (En ejecución): La Activity está en primer plano y es interactiva.
- Paused (En pausa): La Activity está parcialmente oculta pero aún visible (por ejemplo, por una ventana emergente o un diálogo) y no es interactiva.
- Stopped (Detenida): La Activity está completamente oculta y no es visible, pero sigue en memoria.
- Destroyed (Destruida): La Activity ha sido eliminada de la memoria.

Métodos clave del ciclo de vida

Para gestionar estos estados, Android proporciona varios métodos en la clase Activity que se llaman automáticamente al cambiar el estado de la actividad:

- onCreate(): Este método se llama cuando la actividad se crea por primera vez. Aquí es donde inicializas los componentes básicos de la actividad, como la interfaz de usuario y los recursos necesarios. Este es el primer método que se ejecuta y es esencial para configurar la actividad.

- onStart(): Este método se llama cuando la Activity se vuelve visible para el usuario, pero aún no es interactiva. Es el lugar adecuado para realizar las configuraciones finales antes de que la Activity sea visible.

- onResume(): Este método se llama cuando la Activity comienza a interactuar con el usuario. La Activity está en primer plano y es el estado en el que permanece la mayor parte del tiempo mientras el usuario la utiliza. Aquí es donde se reanuda cualquier tarea que estaba pausada, como la reproducción de un video.

- onPause(): Este método se llama cuando la Activity pierde el enfoque, pero aún es visible. Aquí es donde debes pausar las operaciones que no deberían continuar cuando la Activity no está en primer plano, como detener animaciones o liberar recursos que no son necesarios hasta que la Activity vuelva a estar en primer plano.

- onStop(): Este método se llama cuando la Activity ya no es visible para el usuario. Puedes usar este método para realizar operaciones más pesadas de guardado de datos o liberar recursos que no se necesitan mientras la Activity está detenida.

- onDestroy(): Este método se llama antes de que la Activity sea destruida por completo, ya sea porque el sistema está recuperando memoria o porque la Activity se está cerrando intencionalmente. Aquí puedes hacer la limpieza final de recursos.

- onRestart(): Este método se llama cuando la Activity se está reiniciando después de haber sido detenida. Es seguido por onStart(), y es útil para realizar cualquier inicialización que se necesite cuando la Activity vuelve a ser visible.

Ejemplo del ciclo de vida en acción

Imagina que el usuario lanza una aplicación, la navega y luego la minimiza:

- Lanzar la aplicación: Se llama a onCreate(), seguido de onStart() y onResume(). La aplicación ahora está en el estado "Running".
- Minimizar la aplicación: Se llama a onPause() y luego a onStop(). La Activity ahora está en el estado "Stopped".
- Volver a la aplicación: Se llama a onRestart(), seguido de onStart() y onResume(). La Activity vuelve al estado "Running".
- Cerrar la aplicación: Se llama a onPause(), onStop() y finalmente onDestroy().

Gestión de la configuración y persistencia de datos

onSaveInstanceState(Bundle outState) y onRestoreInstanceState(Bundle savedInstanceState) son métodos cruciales para manejar la persistencia del estado de la actividad durante cambios en la configuración, como la rotación de la pantalla. onSaveInstanceState se utiliza para guardar los datos importantes antes de que la actividad sea destruida, mientras que onRestoreInstanceState restaura estos datos cuando la actividad se recrea.

Consideraciones de diseño

Al diseñar y desarrollar una aplicación Android se debe de tener presente la gestión de recursos del dispositivo y la optimización de memoria del mismo:

- Recursos limitados: Como las actividades pueden ser destruidas y recreadas en cualquier momento por el sistema operativo para liberar recursos, es importante diseñar la aplicación para manejar correctamente estas transiciones.
- Optimización de la memoria: Debes ser cuidadoso con la gestión de recursos, especialmente en los métodos onPause(), onStop() y onDestroy(), para evitar fugas de memoria y otros problemas relacionados con la gestión ineficiente de los recursos.

Comprender y manejar correctamente el ciclo de vida de las actividades es fundamental para desarrollar aplicaciones Android robustas y eficientes. Permite asegurar que la aplicación se comporte correctamente en diferentes situaciones, como cambios en la configuración o cuando el sistema recupera recursos.

1.6. Referencias bibliográficas

Google. (11 de 07 de 2024). *Android Developers*. Obtenido de https://developer.android.com/

Leiva, A. (2016). *Kotlin for Android Developers: Learn Kotlin the easy way while developing and Android App.* CreateSpace Independent Publishing Platform.

Trivedi, H. (2020). *Android application development with Kotlin: Build Your First Android App In No Time.* BPB Publications.

A fondo

Android API Levels

https://apilevels.com/ Documentación en línea. (2024)

Este es un resumen de todas las versiones de Android y sus correspondientes identificadores para desarrolladores de Android, así como, del nivel de uso acumulativo.

Documentación oficial de Gradle

https://docs.gradle.org/current/userguide/quick_start.html Documentación en línea. (2024)

Gradle es una herramienta de automatización de la construcción de nuestro código que bebe de las aportaciones que han realizado herramientas como ant y maven pero intenta llevarlo todo un paso más allá. En esta documentación encontrarás toda la información necesaria. Los proyectos de Android Studio trabajan con Gradle.

Android Manifest

https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419 Documentación en línea. (2024)

Todos los proyectos de apps deben tener un archivo AndroidManifest.xml, con ese nombre preciso, en la raíz del conjunto de orígenes del proyecto. Este archivo describe información esencial sobre tu app para las herramientas de compilación de Android, el sistema operativo Android y Google Play. En esta documentación encontraras toda la información, elementos y configuración de este importante fichero.

Android Developers

https://developer.android.com/studio?hl=es-419 Documentación en línea. (2024)

Web oficial para desarrolladores Android, podemos encontrar toda la información para comenzar a programar, así como descargar nuestro IDE.

Jet Brains

https://www.jetbrains.com/es-es/ Documentación en línea. (2024)

Portal web de Jet Brains, donde se pueden encontrar todos sus productos y proyectos.

Entrenamientos

Entrenamiento 1

- Desarrollar un cronograma con los sistemas operativos móviles más relevantes a lo lago de la historia, junto con sus fechas de lanzamiento y algunas de sus características clave. Debe de abarcar desde los primeros sistemas operativos móviles (años 90) hasta los más recientes
- Desarrollo paso a paso: Realiza una búsqueda por internet y plasma los resultados encontrados.
- Solución: https://github.com/Anuar-UNIR/PMDM_2024-2025/tree/main/Tema%201

Entrenamiento 2

- Desarrollar un cronograma de las diferentes versiones de Android desde su lanzamiento hasta la actualizad.
- Desarrollo paso a paso: Realiza una búsqueda por internet y plasma los resultados encontrados.
- Solución: https://github.com/Anuar-UNIR/PMDM_2024-2025/tree/main/Tema%201

Entrenamiento 3

- Instalar Android Studio y su SDK.
- Desarrollo paso a paso:
    - Descarga la aplicación desde https://developer.android.com/studio?hl=es-419 y sigue los pasos de instalación

- Solución: Ver video de la asignatura.

Entrenamiento 4

- Configurar dos dispositivos en el emulador de Android Studio.
- Desarrollo paso a paso:
    - Abrir el configurador de dispositivos de Android
    - Configurar un dispositivo.
    - Repetir el proceso para un segundo dispositivo.
- Solución: Ver video de la asignatura.

Entrenamiento 5

- Generar un primer proyecto en Android Studio en java y mostrar el mensaje de Hola mundo por la Activity principal.
- Desarrollo paso a paso
- Solución: https://github.com/Anuar-UNIR/PMDM_2024-2025/tree/main/Tema%201/Entrenamiento%205