# Arquitectura aplicaciones móviles nativas

**Breve descripción:**

En este componente formativo se abordan los conceptos básicos de las aplicaciones móviles nativas, los cuales servirán para identificar los tipos de metodologías usadas en el desarrollo móvil, en plataformas nativas, en entornos de desarrollo y su configuración.

**Junio 2024**

**Grupo de Ejecución de la Formación Virtual**

## Tabla de contenido

## Introducción

Sin duda, los dispositivos móviles son parte de nuestro día a día y cada vez son más avanzados. La creciente demanda de software específico para estos dispositivos ha generado nuevos desafíos para los desarrolladores, ya que este tipo de aplicaciones posee características propias, restricciones y necesidades únicas.

Es importante definir cuáles serán las tecnologías más adecuadas para la programación de móviles, y una de las arquitecturas más implantadas es la proporcionada por el sistema Android. En los dispositivos móviles actuales pueden utilizarse tres tipos de aplicaciones diferentes: nativas, híbridas y web. En este componente se tratarán las aplicaciones móviles nativas.

Las aplicaciones nativas son aquellas que enfocan su desarrollo en una plataforma específica como Android, iOS y Windows, haciendo uso de su correspondiente lenguaje de programación, el cual permite tener compatibilidad con sus características. En el caso de Android, se utilizan lenguajes como Java y Kotlin; para la plataforma iOS, Objective-C o Swift; y para la plataforma Windows, los lenguajes C++ o C#.

Las aplicaciones nativas son las más eficientes y pueden salir a producción en las tiendas más populares como App Store, Google Play y AppGallery, entre otras. Una de sus características principales es que permiten enviar notificaciones para realizar promociones comerciales de negocios, productos, etc. En cuanto al tiempo de desarrollo, este es mucho más largo y, por tanto, son más costosas debido a la especialización que deben tener los programadores en cada lenguaje. Su uso se

encuentra en aplicaciones complejas como editores de video, aplicaciones de medicina, aplicaciones de ingeniería, entre otros.

Desde el punto de vista de los programadores, uno de los mayores inconvenientes es que para que una misma aplicación pueda utilizarse en dos o más plataformas distintas, hay que desarrollar la misma aplicación en los diferentes lenguajes, iniciando desde cero y necesitando más tiempo de desarrollo y mayor costo.

# 1. Enfoque metodológico para el desarrollo de aplicaciones móviles nativas

Actualmente, las metodologías ágiles tienen una gran notoriedad, ya que ofrecen una buena solución para desarrollar proyectos en tiempos más cortos, especialmente aquellos proyectos que constantemente experimentan cambios en sus requisitos. Las aplicaciones para dispositivos móviles son un buen ejemplo de esto, ya que requieren una variedad de características como portabilidad, movilidad y adaptación, entre otras. Aunque hay muchas aplicaciones para dispositivos móviles que funcionan en diferentes plataformas, no siempre cumplen totalmente las expectativas de los usuarios debido a su baja calidad en el desarrollo. Esto se debe a que el uso de metodologías no es una prioridad en este tipo de aplicaciones, ignorando los métodos de ingeniería que son los que garantizan el mantenimiento y calidad de las aplicaciones.

Según, Letelier y Canos (2003)

Las metodologías ágiles son procesos para desarrollar software de manera rápida y con gran facilidad de adopción por los equipos de trabajo. Se han seleccionado las metodologías ágiles más referenciadas para emplearlas en el desarrollo de aplicaciones móviles, lo cual resulta muy conveniente teniendo en cuenta el crecimiento de los proyectos para estas tecnologías.

A continuación, se describen los conceptos principales para la introducción al uso de metodologías de desarrollo:

- **La metodología según Avison y Fitzgerald (2006):**
  Contribuye en la implementación de nuevos sistemas de información utilizando un conjunto de métodos, herramientas y documentos auxiliares.

Una metodología se divide en fases y subfases para planificar, gestionar y controlar el proyecto. También permite seleccionar las técnicas más indicadas en cada proceso de un proyecto.

- **La metodología ágil según Amaya Balaguera (2015):**

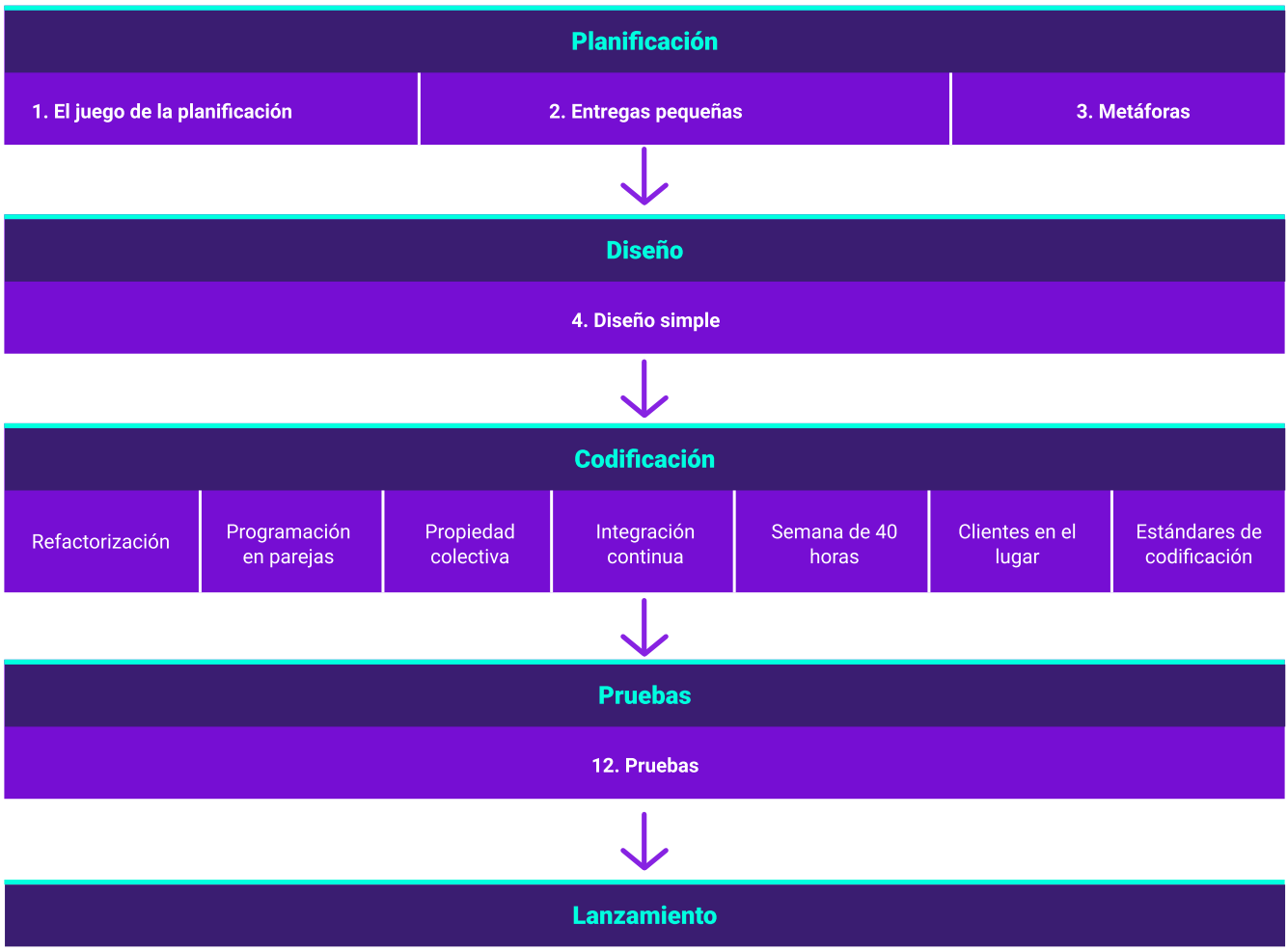
  Estas metodologías surgen como una alternativa a las metodologías tradicionales. Su enfoque se basa en los principios del manifiesto ágil, que propone un desarrollo iterativo. Su objetivo principal se centra en realizar una mejor captura de los requisitos cambiantes, la gestión de riesgos y la reducción del tiempo de desarrollo.

Entre las principales metodologías ágiles aplicadas en el desarrollo de aplicaciones móviles, se destacan las siguientes:

## Extreme Programming (XP)

Es una metodología liviana, aplicable en grupos de desarrollo pequeños y medianos, que se centra en requerimientos no exactos y en continuo cambio. Se caracteriza por tener un ciclo de vida dinámico, a través de ciclos de desarrollo cortos denominados iteraciones. Sus fases son análisis, diseño, desarrollo y pruebas, en las cuales se aplican 12 prácticas.

**Figura 1.** Prácticas de XP




Así pues, XP no es solo un conjunto de técnicas, sino un conjunto de principios probados y fiables con características de las metodologías convencionales, pero llevadas a un nivel extremo.

## SCRUM

Scrum, según Takeuchi & Nonaka (1986), presenta un proceso adaptativo, rápido y autoorganizado para el desarrollo de productos, centrado en la gestión de proyectos en situaciones que son difíciles de planificar a futuro. El término "Scrum" proviene del

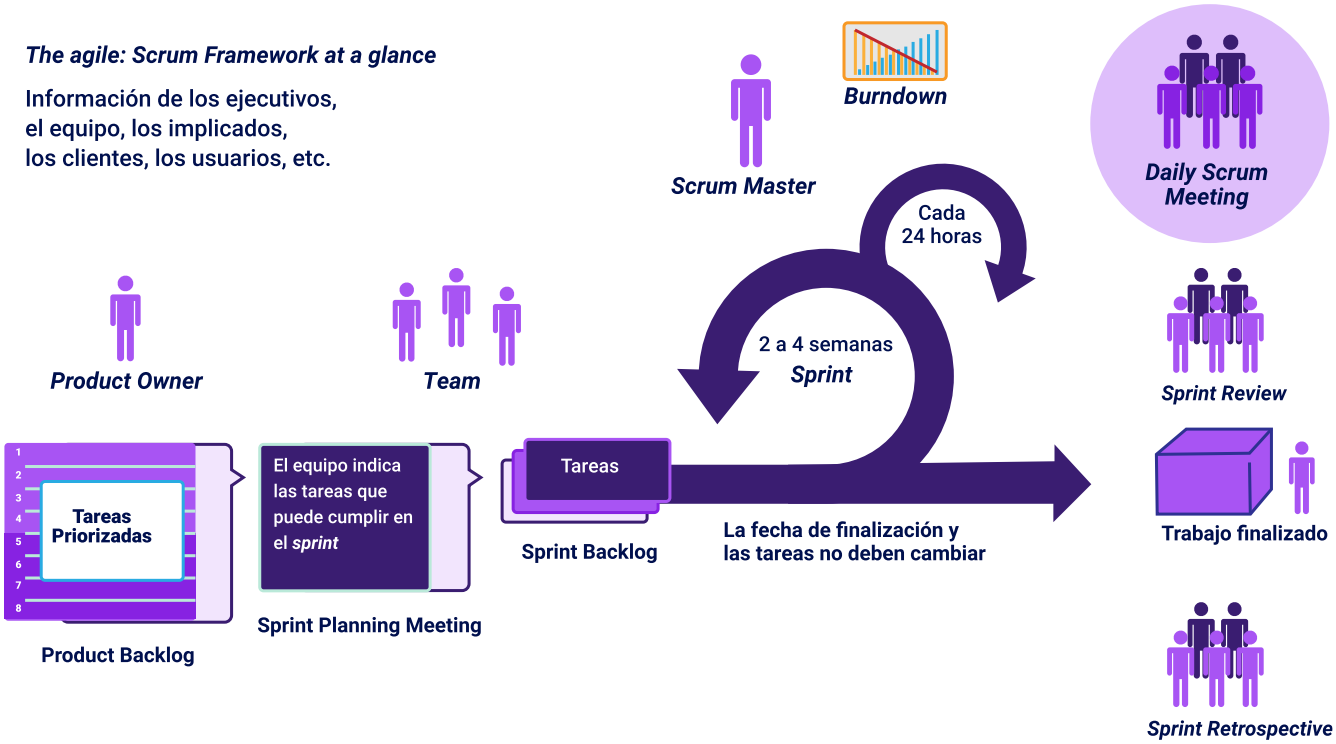
famoso juego de rugby y hace referencia a la forma en que se reintroduce un balón que ha salido del campo al terreno de juego de manera colectiva.

Este modelo se basa en el desarrollo incremental, es decir, conforme avanzan las fases y la iteración, mayor es el tamaño del proyecto que se está desarrollando. Los procesos que utiliza son:

- Product Backlog
- Sprint Backlog
- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Consiste en realizar un análisis de los requerimientos del sistema (Product Backlog), señalar cuáles serán los objetivos a corto o mediano plazo dentro de un sprint, es decir, en la fase de desarrollo. Posteriormente, los desarrolladores realizan su trabajo, se efectúan algunas pruebas y se retroalimenta de acuerdo con lo conseguido al terminar la última fase.

**Figura 2.** Modelo ágil Scrum




A continuación, se explora el enfoque iterativo e incremental que Scrum ofrece para la gestión de proyectos de software:

**Video 1.** Scrum y la especificación de requisitos


**Enlace de reproducción del video**

| **Síntesis del video: Scrum y la especificación de requisitos** |
| --- |
| El marco de trabajo Scrum está soportado en un proceso de construcción iterativo e incremental evolutivo, en el que se identifican tres roles principales: el equipo de trabajo o Team, conformado por los desarrolladores, diseñadores, personal de calidad y de infraestructura requerido para la construcción del producto de software; el Scrum Master, que realiza funciones parecidas a las de un director de proyecto pero, más enfocadas en garantizar que el equipo de trabajo tenga todas las herramientas y recursos necesarios para el desarrollo de su trabajo; y finalmente, el Dueño del Producto o Product Owner, que se convierte en un representante del |

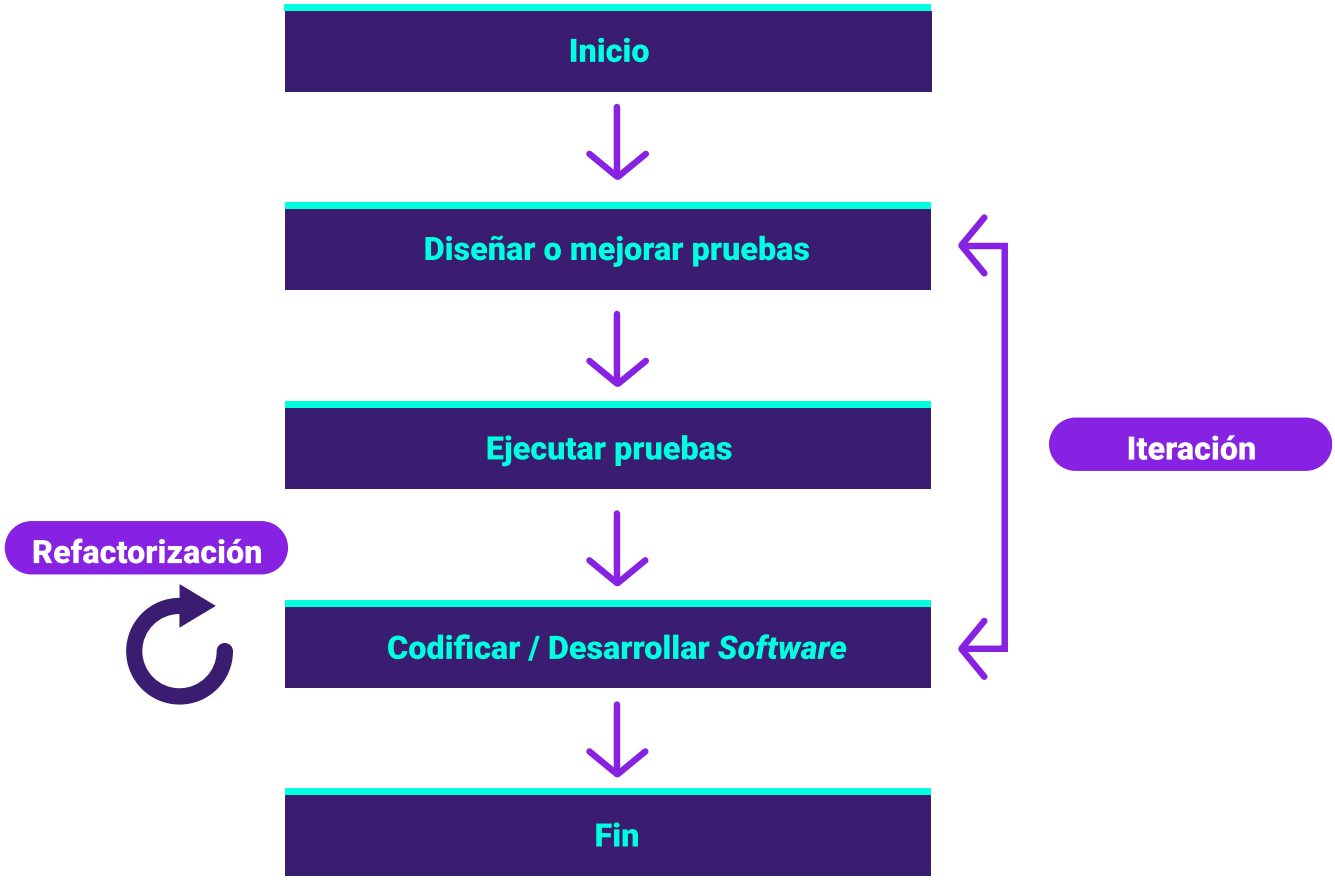
cliente y quien es el único encargado de la gestión de requisitos del proyecto. Scrum establece el concepto de Sprint para referirse a una iteración que contempla tiempos fijos entre dos y 4 semanas, dependiendo del equipo de trabajo. Durante este tiempo, se incluye la planeación del Sprint, donde se definen los requerimientos a desarrollar en ese periodo de tiempo, una fase de construcción del producto y finalmente, un proceso de despliegue para poder hacer la respectiva demostración de lo construido al final de la iteración en reuniones de revisión. En este marco de trabajo se redefine el concepto de requerimiento hecho, y normalmente va mucho más allá de construir el código. Por lo general, se incluyen procesos de validación con pruebas unitarias y pruebas de integración. El artefacto mediante el cual se condensan todos los requerimientos del sistema se denomina Pila de Producto o Product Backlog, la cual es una lista ordenada por prioridad de todos los requerimientos del sistema, generalmente descritos en la forma de historias de usuario.

## Test Driven Development (TDD)

Según Astel (2003), el Test Driven Development es un enfoque de desarrollo que mantiene un exhaustivo conjunto de pruebas realizadas por el programador. Su objetivo principal es que ninguna parte del código avance a producción sin antes realizar las pruebas asociadas y obtener la aprobación correspondiente. Se trata de un desarrollo orientado a las pruebas, que cambia la mentalidad del equipo de desarrollo, agilizando los resultados y aumentando la calidad del producto. La tendencia actual es integrar TDD en cualquier metodología, ya sea ágil (Scrum Alliance - TDD and Scrum, 2011) o tradicional (Letelier et al., s. f.), para aprovechar los beneficios de practicar una

metodología que permite corregir errores constantemente, asegurar la calidad del producto y protegerse de errores, tanto malintencionados como humanos.

**Figura 3.** Test Driven (TDD)

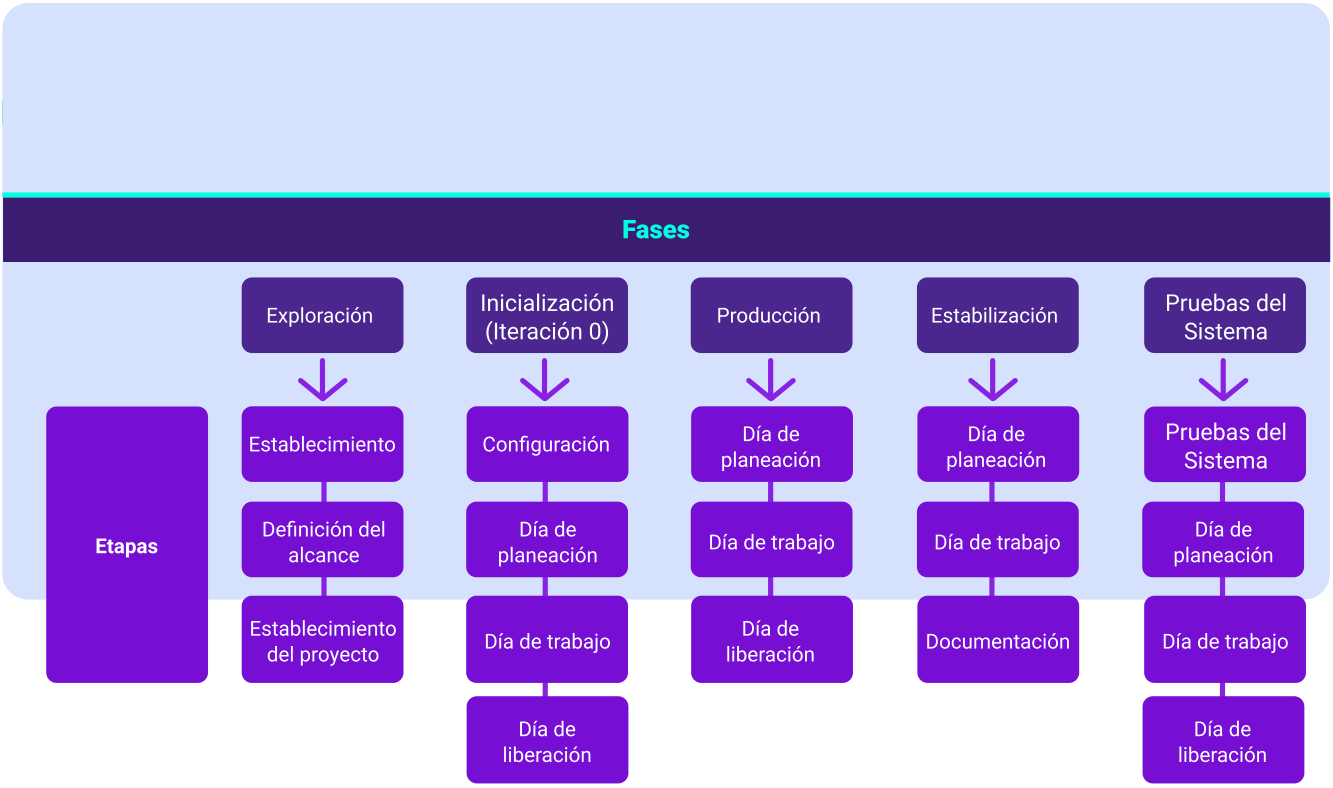



## Mobile-D

Según Abrahamsson (2004) es una metodología de desarrollo diseñada específicamente para el desarrollo de aplicaciones móviles, que integra prácticas ágiles como Extreme Programming y Crystal. Presenta características tales como la orientación a pruebas, la programación en parejas, la integración continua y la refactorización, además de las tareas de mejora del proceso de software. La

metodología incluye las fases de exploración, inicialización, producción, estabilización y pruebas. Cada fase cuenta con un día de planificación y un día de entrega.

**Figura 4.** Ciclo de desarrollo Mobile-D




A continuación, se explica cada fase:

- **Exploración**

  En esta fase se recolecta la información necesaria para la adecuada orientación del proyecto, sus requisitos, su alcance y las partes interesadas en el producto (Stakeholders).

- **Inicialización**

  Esta fase involucra la preparación de recursos físicos y tecnológicos, como la formación del equipo de trabajo y la disposición de los equipos tecnológicos de uso.

- **Producción**

  Implementación de las funcionalidades que requiere el producto mediante un desarrollo iterativo, dividido en tres etapas:

  a) Día de planificación, con ejecución de tests de aceptación.
  b) Día de trabajo, para el desarrollo de funcionalidades.
  c) Día de entrega de funcionalidad.

- **Estabilización**

  Fase de integración y verificación del correcto funcionamiento del sistema y la calidad del desarrollo. Incluye también la documentación del proyecto.

- **Pruebas del sistema**

  Verificación del funcionamiento de todos los requerimientos establecidos por el cliente y solución de errores encontrados.

## 2. Plataformas de desarrollo móvil nativas

Las plataformas de desarrollo móvil nativas, son aquellas que tienen un desarrollo específico para cada uno de sus sistemas operativos; IOS, Android, o Windows Phone, lo que quiere decir que para cada uno de estos Sistemas Operativos se adapta un lenguaje de desarrollo así:

- Sistema Operativo Móvil: iOS, Lenguajes: Objetive-C, Swift
- Sistema Operativo Móvil: Android, Lenguajes: Java, Kotlin
- Sistema Operativo Móvil: Windows Phone, Lenguajes: C++, C#

Una de las principales ventajas de las aplicaciones nativas es que pueden aprovechar todas las funcionalidades avanzadas de los dispositivos de cada plataforma, incluidos los procesadores gráficos. Los mayores beneficios de este tipo de aplicaciones siempre serán la flexibilidad y la usabilidad, que son superiores a las de las aplicaciones híbridas. Uno de los principales inconvenientes del desarrollo de las aplicaciones nativas es que tanto el desarrollo como las actualizaciones de estas apps tienen un alto costo.

### Rendimiento

El rendimiento de las aplicaciones nativas es otra de sus características principales, ya que ofrecen mayor velocidad, animaciones avanzadas, transiciones complejas, aprovechamiento total del hardware (GPS, cámara, sensores, etc.) y un menor consumo de memoria.

Ventajas de las aplicaciones nativas:

- Permiten el desarrollo para diferentes tipos de dispositivos como wearables, TV o carros.
- Ofrecen un nivel de seguridad superior al de las aplicaciones híbridas.
- Tienen un mayor grado de optimización.
- Acceso a funcionalidades de accesibilidad nativas.
- Proporcionan un entorno que permite el uso de herramientas de arrastrar y soltar para el diseño de interfaces de usuario.
- Disponen de SDK para la optimización de librerías.

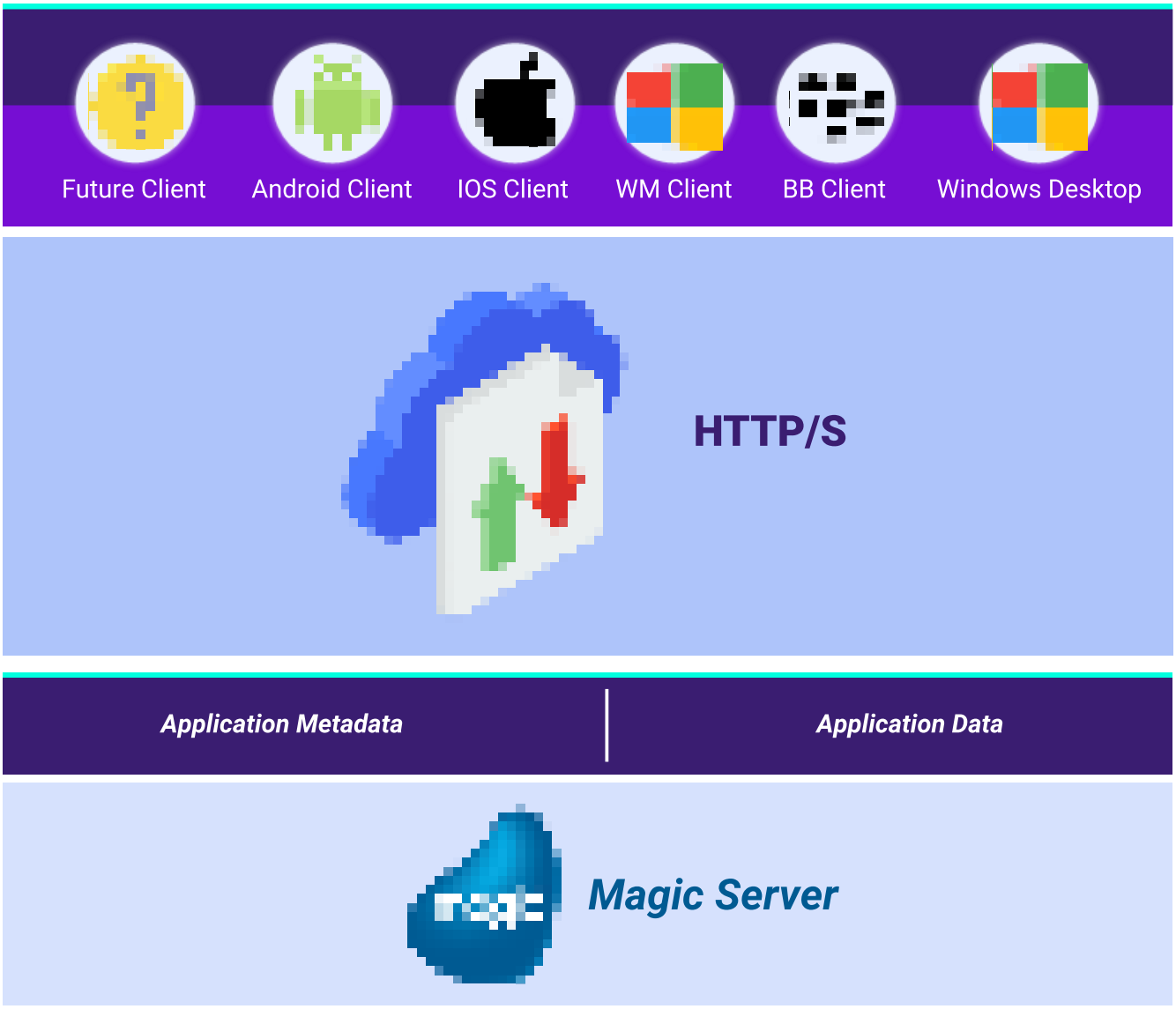
## Arquitectura de las aplicaciones móviles

Básicamente, el desarrollo de las apps móviles se compone de dos partes, las cuales se diferencian por su código: el front-end y la parte de web services.

El front-end, también llamado parte cliente, es el segmento de lógica de visualización e interacción con el usuario, que se ejecuta en los dispositivos a través de su sistema operativo, como Android o iOS.

La parte de web services, o parte servidora, es donde se encuentra la lógica del negocio de las aplicaciones, la persistencia de datos y la interacción con otras plataformas.

**Figura 5.** Arquitectura de una app



## 3. Entornos de desarrollo

Un entorno de desarrollo es un software que facilita un conjunto de herramientas que permiten crear una aplicación. Estos entornos contienen soporte para uno o varios lenguajes de programación y son los encargados de depurar, compilar, realizar pruebas y convertir nuestro código en una aplicación ejecutable.

Los entornos más utilizados para el desarrollo de aplicaciones nativas son: Xcode para la plataforma iOS y Android Studio para la plataforma Android.

- **Xcode**

  Entorno de desarrollo integrado (IDE) para iOS, macOS, watchOS y tvOS, creado por Apple. Incluye herramientas para construir, probar y empaquetar aplicaciones para su envío al App Store. Lenguajes soportados: C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit, Swift.

- **Android Studio**

  IDE oficial para el desarrollo de aplicaciones Android, ofrece compilación flexible y un entorno unificado para todos los dispositivos Android. Incluye integración con GitHub y herramientas de diagnóstico. Lenguajes soportados: Java, Kotlin.

- **Visual Studio**

  Creado por Microsoft en 1997, es un IDE compatible con Windows, Linux y macOS. Soporta múltiples lenguajes de programación y entornos de desarrollo web, con versiones gratuitas y de pago. Lenguajes soportados: C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP, ASP.NET.

El desarrollo de aplicaciones móviles se ha convertido en una necesidad debido a la gran utilización y evolución de estas plataformas. Por esta razón, es importante definir cuáles serán las tecnologías más adecuadas para la programación móvil. Una de las arquitecturas más implantadas es la proporcionada por el sistema Android.

## 3.1. Plataforma Android

Es un sistema operativo móvil desarrollado por Google, de código abierto y basado en Linux. Ofrece una optimización de la máquina virtual denominada Dalvik VM y soporta la mayoría de las API de Java SE para el desarrollo de aplicaciones. Está enfocado en ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos.

Sus características son:

- Contiene un framework de aplicaciones para la reutilización de componentes.
- Navegador integrado
- Cuenta con una base de datos que se integra directamente en las aplicaciones, SQLite.
- Ofrece diferentes formas de mensajería.
- Adaptable a muchas pantallas y resoluciones.

**Versiones**

Android ha lanzado muchas versiones desde su inicio, con numerosas actualizaciones. Cada nueva versión corrige fallos detectados en las anteriores e incluye nuevas funcionalidades con soporte para las nuevas tecnologías. Curiosamente, las

versiones de Android reciben el nombre de postres o dulces, cuyos nombres van en orden alfabético, una tradición que se mantuvo hasta la versión 9.

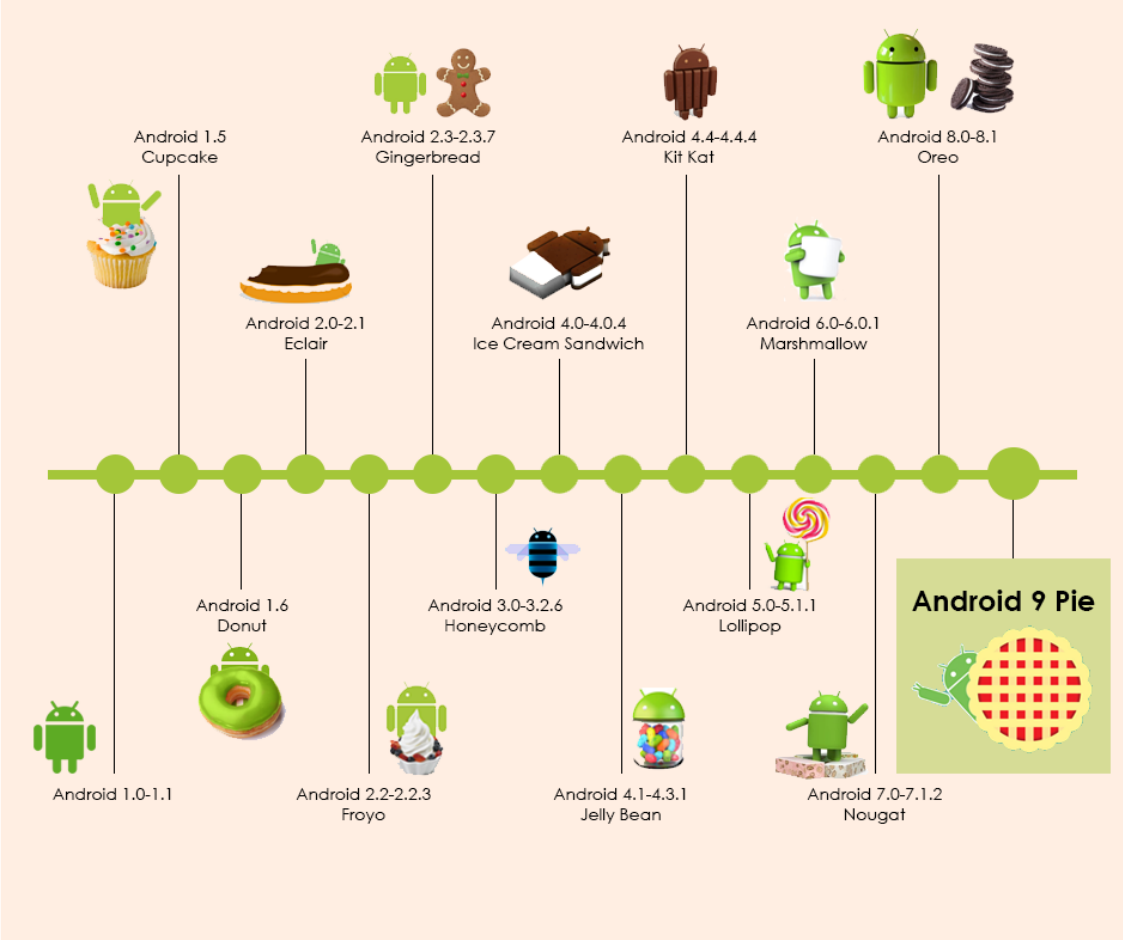
**Tabla 1.** Nombres versiones y lanzamiento de sistema operativo Android

| Nombre | Versión | Lanzamiento |
|---|---|---|
| Android Apple Pie | Versión 1.0 | 23 de septiembre de 2008 |
| Android Banana Bread | Versión 1.1 | 9 de febrero de 2009 |
| Android Cupcake | Versión 1.5 | 25 de abril de 2009 |
| Android Donut | Versión 1.6 | 5 de septiembre de 2009 |
| Android Éclair | Versión 2.0 | 26 de octubre de 2009 |
| Android Froyo | Versión 2.2 | 20 de mayo de 2010 |
| Android Gingerbread | Versión 2.3 | 6 de diciembre de 2010 |
| Android Honeycomb | Versión 3.0 | 22 de febrero de 2011 |
| Android Ice Cream Sandwich | Versión 4.0 | 18 de octubre de 2011 |
| Android Jelly Bean | Versión 4.1 | 9 de julio de 2012 |
| Android KitKat | Versión 4.4 | 31 de octubre de 2012 |
| Android Lollipop | Versión 5.0 | 12 de noviembre de 2014 |
| Android Marshmallow | Versión 6.0 | 5 de octubre de 2015 |
| Android Nougat | Versión 7.0 | 15 de junio de 2016 |
| Android Oreo | Versión 8.0 | 21 de agosto de 2017 |
| Android Pie | Versión 9.0 | 6 de agosto de 2018 |
| Android 10 | Versión 10 | 3 de septiembre de 2019 |

| Nombre | Versión | Lanzamiento |
|---|---|---|
| Android 11 | Versión 11 | 8 de septiembre de 2020 |
| Android 12 | Versión 12 | 4 de octubre de 2021 |
| Android 12 Beta 2 | Versión 12 | 9 de junio de 2021 |
| Android 13 | Versión 13 | 15 de agosto de 2022 |

A continuación, se presenta un esquema que resume lo anterior:

**Figura 6.** Versiones de Android

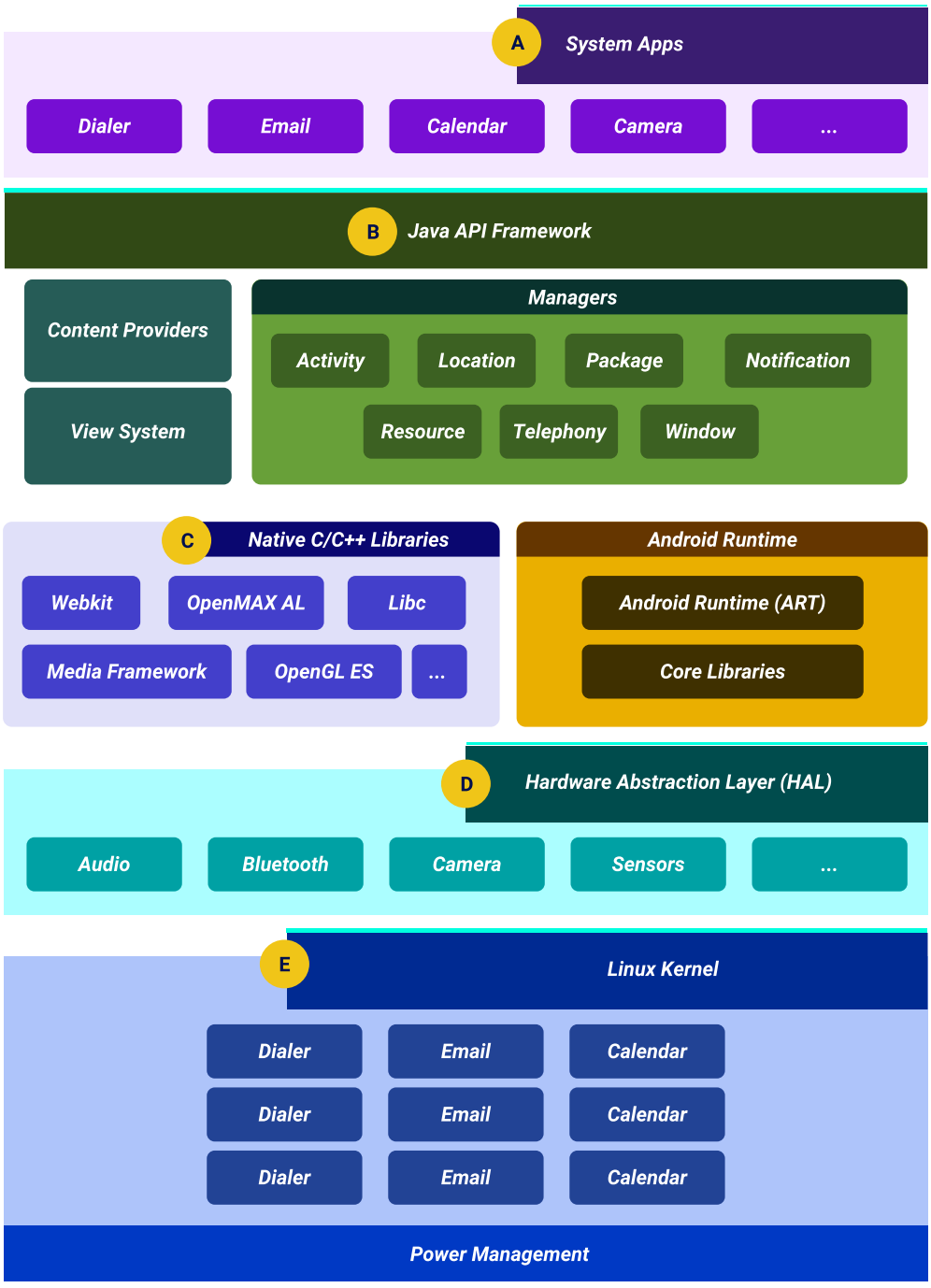



Nota. Tomado de El origen de Android (Características y Costos, por F. Olmos, 2020)

## Arquitectura

Los componentes principales de la plataforma Android están diseñados para una variedad de dispositivos. Android es de código abierto y está basado en Linux.

**Figura 7.** Componentes principales de la plataforma Android



**A) System Apps**

Aplicaciones del sistema que permiten a los desarrolladores acceder desde sus propias aplicaciones simplemente invocándolas, sin necesidad de compilar funcionalidades específicas.

**B) Java API Framework**

Conjunto de API escritas en Java que permiten acceder a las funciones del sistema operativo Android. Estas API facilitan la creación de aplicaciones Android mediante la reutilización de componentes del sistema y servicios como sistemas de vista (compilar IU), administrador de recursos (acceso a strings, gráficos, etc.), administrador de notificaciones, y proveedores de contenido.

**C) Nativa C/C++ Libraries**

Bibliotecas basadas en código nativo que proporcionan diversas funcionalidades esenciales al sistema operativo.

**D) Hardware Abstraction Layer HAL**

Capa de abstracción de hardware que ofrece interfaces para exponer las capacidades del hardware del dispositivo. Incluye módulos de bibliotecas que implementan una interfaz para cada tipo específico de hardware, como la cámara del dispositivo.

**E) Linux Kernel**

Base de la plataforma Android, necesaria para funcionalidades subyacentes como la administración de memoria de bajo nivel y la generación de subprocesos.

A continuación, se presenta una descripción completa del IDE Android Studio, que como se mencionó anteriormente, es el entorno oficial para el desarrollo de aplicaciones móviles nativas en Android.

## 3.2. Android Studio

Android Studio está basado en IntelliJ IDEA de la compañía JetBrains, que proporciona varias mejoras con respecto al plugin ADT (Android Developer Tools) para Eclipse. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma. Fue presentado por Google el 16 de mayo de 2013 en el congreso de desarrolladores Google I/O, con el objetivo de crear un entorno dedicado exclusivamente a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción. Actualmente, es el IDE recomendado por Google.

Las características son:

- Realiza renderizados de layouts en tiempo real, ejecutando las compilaciones de forma rápida.
- Permite ejecutar la aplicación en tiempo real en el dispositivo móvil.
- Incluye un potente emulador.
- Permite simular diferentes dispositivos y tabletas.
- Trabaja con un editor de elementos gráficos para la creación de la interfaz sin necesidad de código, solo arrastrando y soltando.

La realización de aplicaciones en Android Studio se enfoca en 4 fases de desarrollo:

- **Fase 1. Configuración de entorno.**

  En esta fase se realiza la instalación y configuración del entorno de desarrollo, se conectan dispositivos y se crean emuladores virtuales.

- **Fase 2. Configuración del proyecto y desarrollo.**

  Esta fase involucra configurar los módulos necesarios que contienen recursos de la aplicación y los archivos de código fuente.

- **Fase 3. Pruebas, depuración y construcción de la aplicación.**

  Se crea el ejecutable o APK que puede ejecutarse e instalarse en el emulador o dispositivo Android.

- **Fase 4. Publicación de la aplicación.**

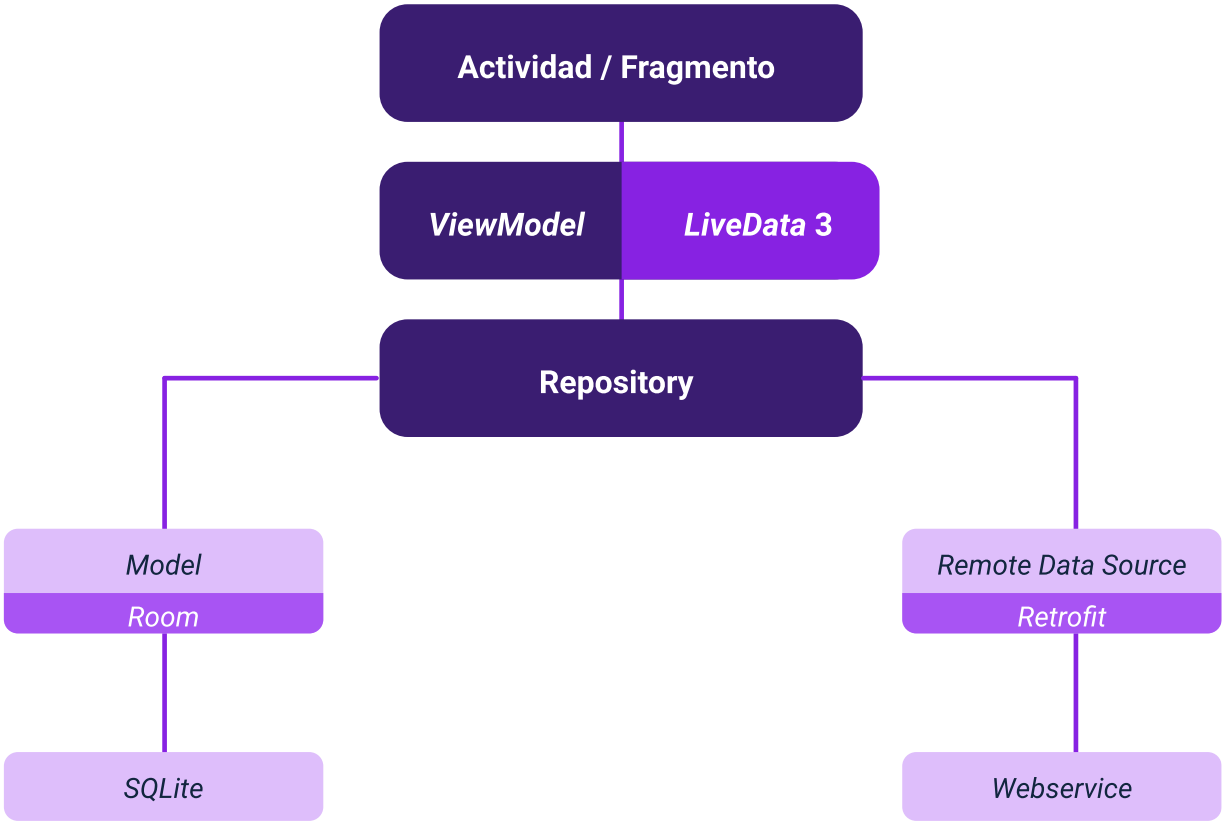
  En esta fase se realiza la solicitud en la tienda PlayStore para la distribución libre de la aplicación, permitiendo que los usuarios puedan descargarla e instalarla en sus dispositivos Android.

## 3.3. Arquitectura de una aplicación en Android

Cada componente depende solo del componente que está un nivel más abajo. Por ejemplo, las actividades y los fragmentos solo dependen de un modelo de vista. El repositorio es la única clase que depende de otras clases. En este ejemplo, el repositorio depende de un modelo de datos persistente y de una fuente de datos de back-end remota.

Este diseño crea una experiencia de usuario consistente y agradable. Independientemente de que el usuario vuelva a la app varios minutos después de cerrarla por última vez o varios días más tarde, verá instantáneamente la información del usuario que la app persiste a nivel local. Si estos datos están inactivos, el módulo de repositorio comienza a actualizar los datos en segundo plano.

**Figura 8.** Arquitectura de una aplicación en Android




## Versiones

Así como el sistema operativo Android tiene una gran variedad de actualizaciones, el ritmo de actualizaciones de Android Studio es bastante alto. A continuación, se listan las últimas versiones, desde la más actual hasta la más antigua.

**Tabla 2.** Nombres, versiones y lanzamiento de sistema operativo Android

| Versiones Android Studio | Fecha de lanzamiento |
|---|---|
| Android Studio Bumblebee (2021.1.1) | 27 de junio de 2021 |

| | |
|---|---|
| Android Studio Arctic Fox (2020.3.1) Beta 3 | 27 de mayo de 2021 |
| Android Studio 4.2.1 | 13 de mayo de 2021 |
| Android Studio 4.2.0 | 4 de mayo de 2021 |
| Android Studio 4.1.3 | 18 de marzo de 2021 |
| Android Studio 4.1 | 12 de octubre de 2020 |
| Android Studio 4.0 | 28 de mayo de 2020 |
| Android Studio 3.6.3 | 17 de abril de 2020 |
| Android Studio 3.0 | 25 de octubre de 2017 |
| Android Studio 2.3.2 | 11 de mayo de 2017 |
| Android Studio Chipmunk (2021.2.1) | 9 de mayo de 2022 |
| Android Studio Dolphin (2021.3.1) | 8 de agosto de 2022 |
| Android Studio Electric Eel (2022.1.1) | 27 de marzo de 2023 |

# 4. Instalación y configuración entorno Android Studio

Para la instalación, lo primero que se debe tener en cuenta son los requisitos mínimos que debe cumplir el equipo. Para un mejor rendimiento y para asegurar que el emulador funcione correctamente, se necesita un equipo de trabajo adecuado que cuente con las propiedades recomendadas para cada sistema operativo.

Requerimientos de sistema para instalación de Android Studio:

**a) Windows**

- 64-bit Microsoft® Windows® 8/10.
- 8 GB de RAM o más.
- 8 GB de espacio mínimo disponible en disco (IDE + SDK de Android + Emulador de Android).
- Resolución de pantalla mínima de 1280 x 800.

**b) macOS**

- MacOS® 10.14 (Mojave) o superior.
- 8 GB de RAM o más.
- 8 GB de espacio mínimo disponible en disco (IDE + SDK de Android + Emulador de Android).
- Resolución de pantalla mínima de 1280 x 800.

**c) Linux**

- Cualquier distribución de Linux de 64 bits que admita Gnome, KDE o Unity DE; Biblioteca GNU C (glibc) 2.31 o posterior.
- 8 GB de RAM o más.
- 8 GB de espacio mínimo disponible en disco (IDE + SDK de Android + Emulador de Android).

- Resolución de pantalla mínima de 1280 x 800.

Además de estos requisitos del sistema, para una instalación exitosa y un funcionamiento correcto del software, se deben considerar otros elementos. Dependiendo del sistema operativo que se utilice, ya sea Ubuntu, Linux, Windows 10 o Mac OS, es necesario seguir unas instrucciones específicas para cada uno de ellos. Se puede encontrar información al respecto en el enlace proporcionado.

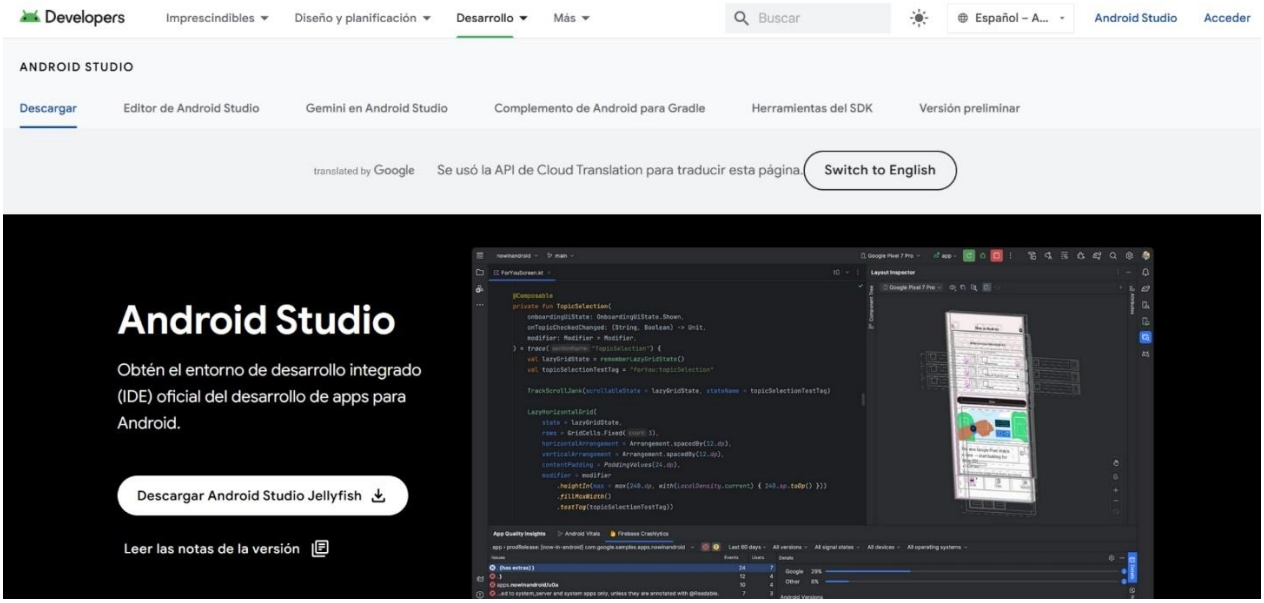
**Instalación**

La instalación de Android en este componente se realizará para Windows, para más información de cómo instalar en Mac, Linux, o Chrome ingresa al enlace:

https://developer.android.com/studio/install?hl=es-419

a) **Descargar e instalar Android Studio.** Ingresa a la página oficial Android developers y selecciona la opción Android Studio o ingresa al siguiente enlace: https://developer.android.com/studio?hl=es-419

**Figura 9.** Descarga de Android Studio




b) La página detecta el sistema operativo y descarga el instalador correspondiente, seleccionar la opción de: Leí y acepto los términos y condiciones anteriores; y hacer clic en descargar Android Studio.

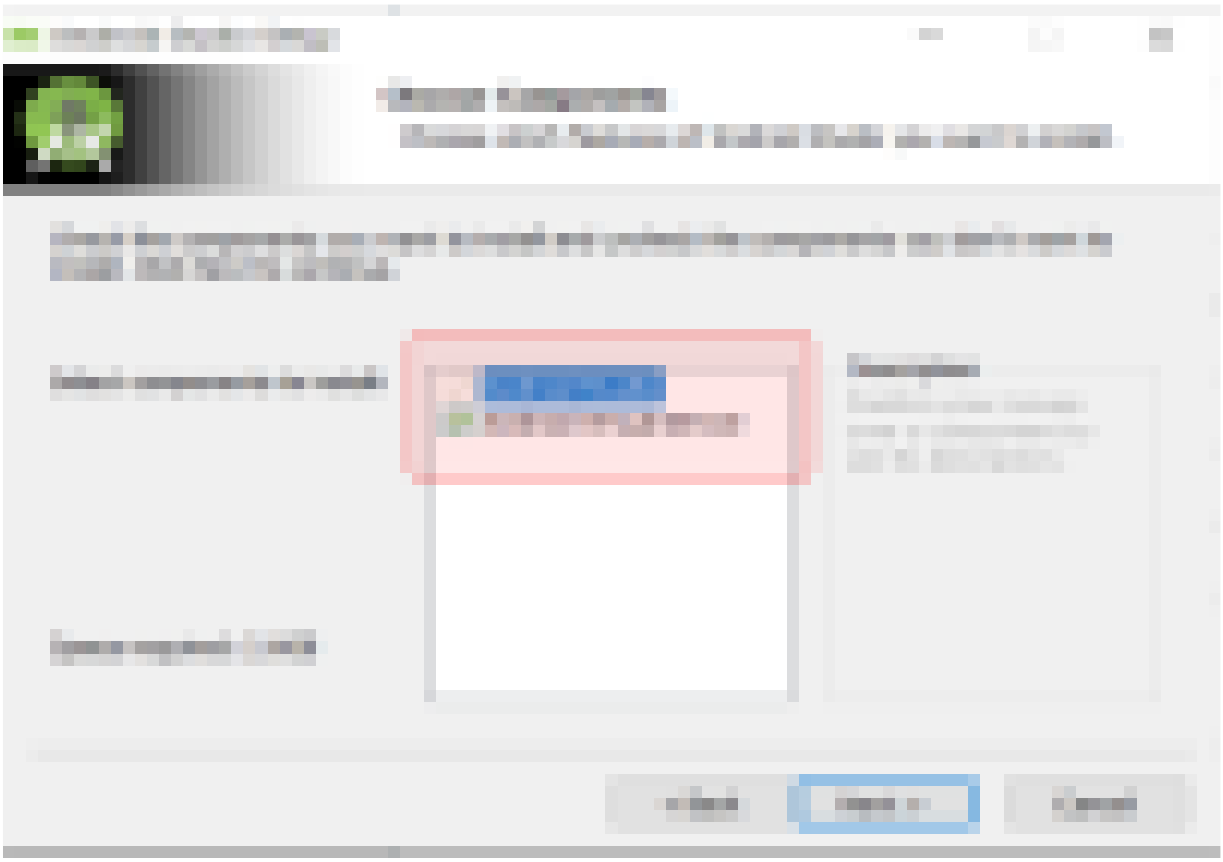
**Figura 10.** Términos y condiciones de la instalación

c) Para instalar la aplicación, se debe descargar y ejecutar el instalador. Al iniciar, se sigue el asistente de instalación.

**Figura 11.** Asistente de Instalación Android Studio

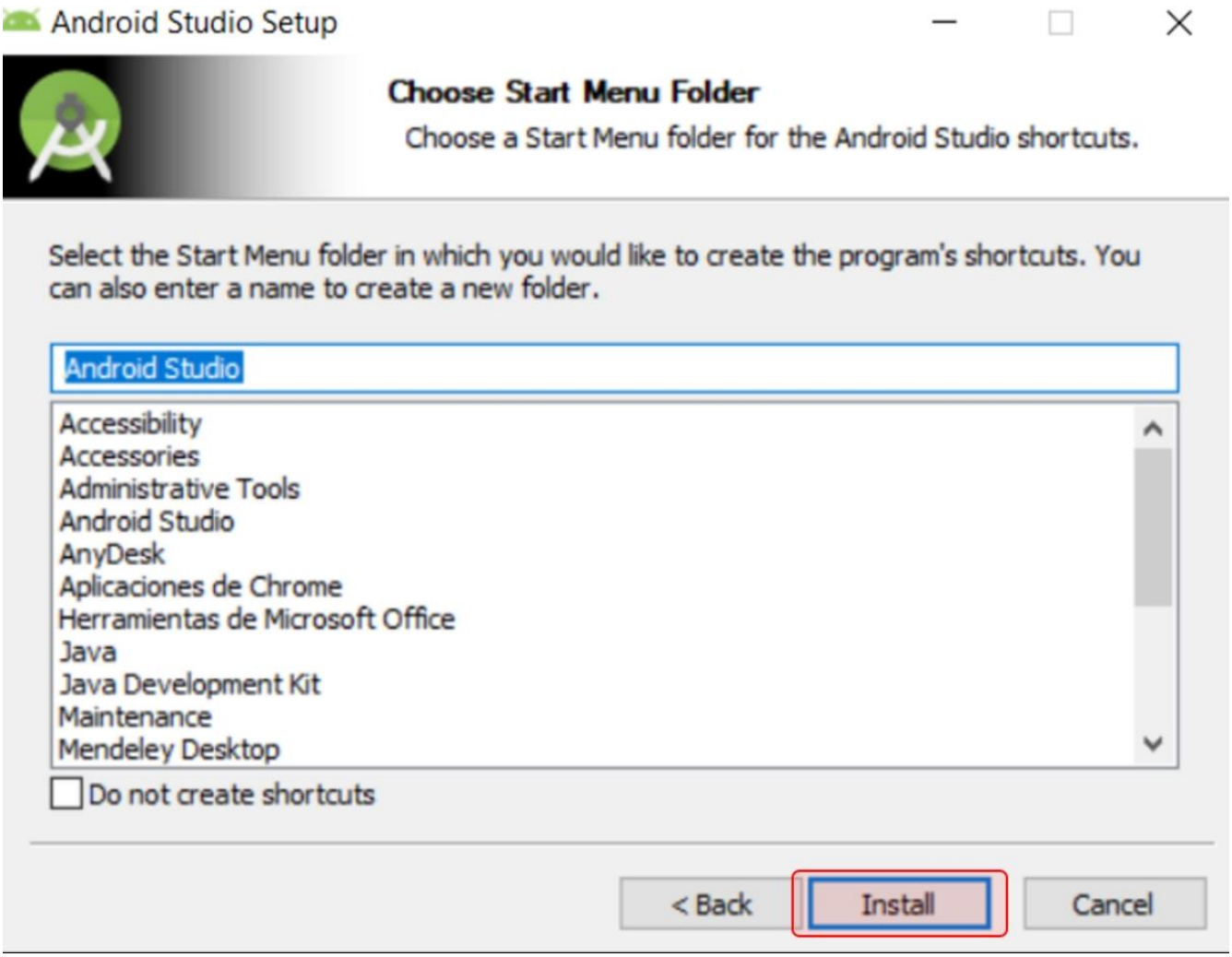

d) En este paso se muestra la ruta donde quedará instalado Android Studio, hacer clic en Next.

**Figura 12.** Asistente Instalación Selección componentes



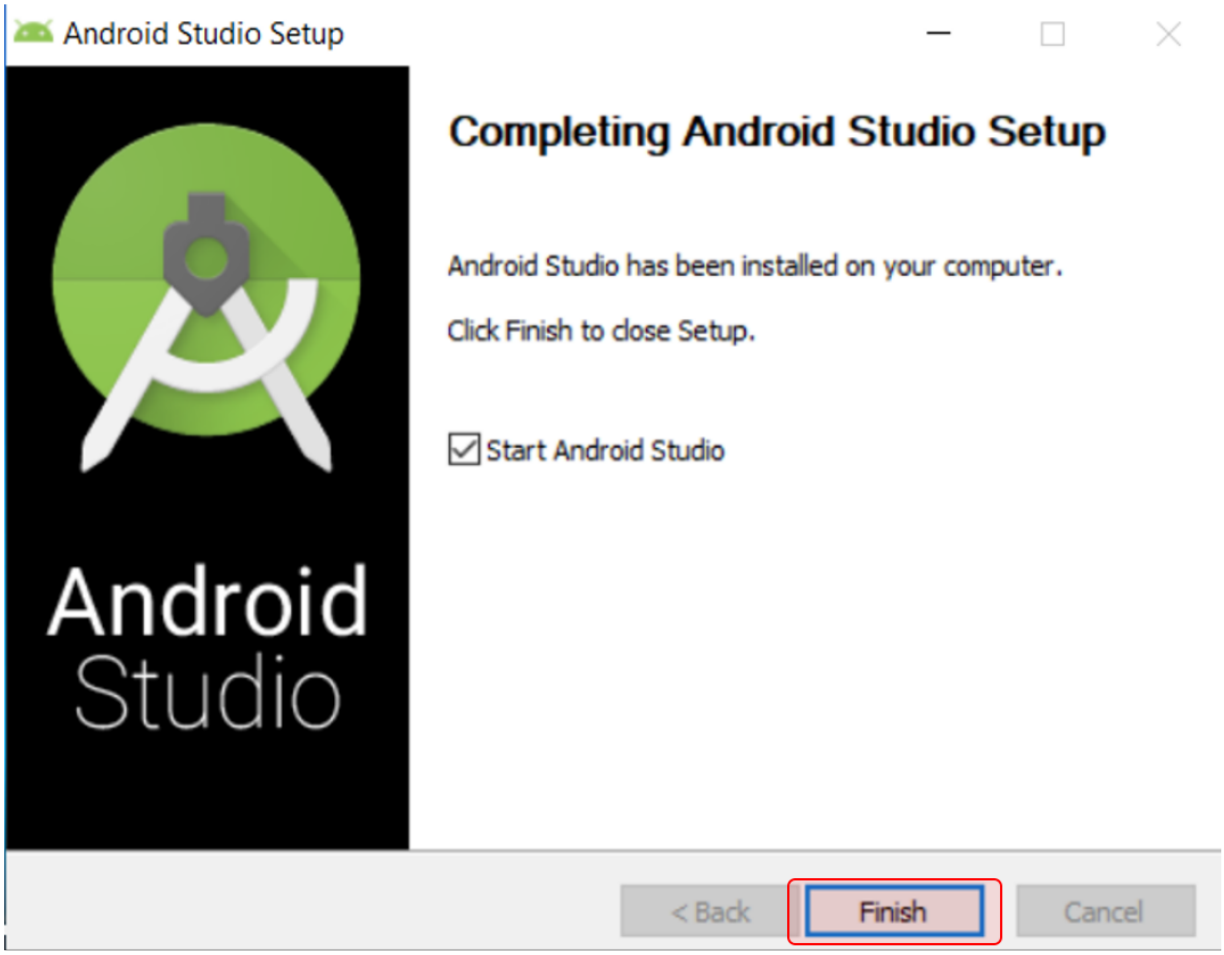
e) En este paso del asistente, se deben seleccionar los componentes a instalar. Se recomienda dejar marcados, en este caso, dos: el propio IDE Android Studio y un Virtual Device.

**Figura 13.** Asistente de Instalación



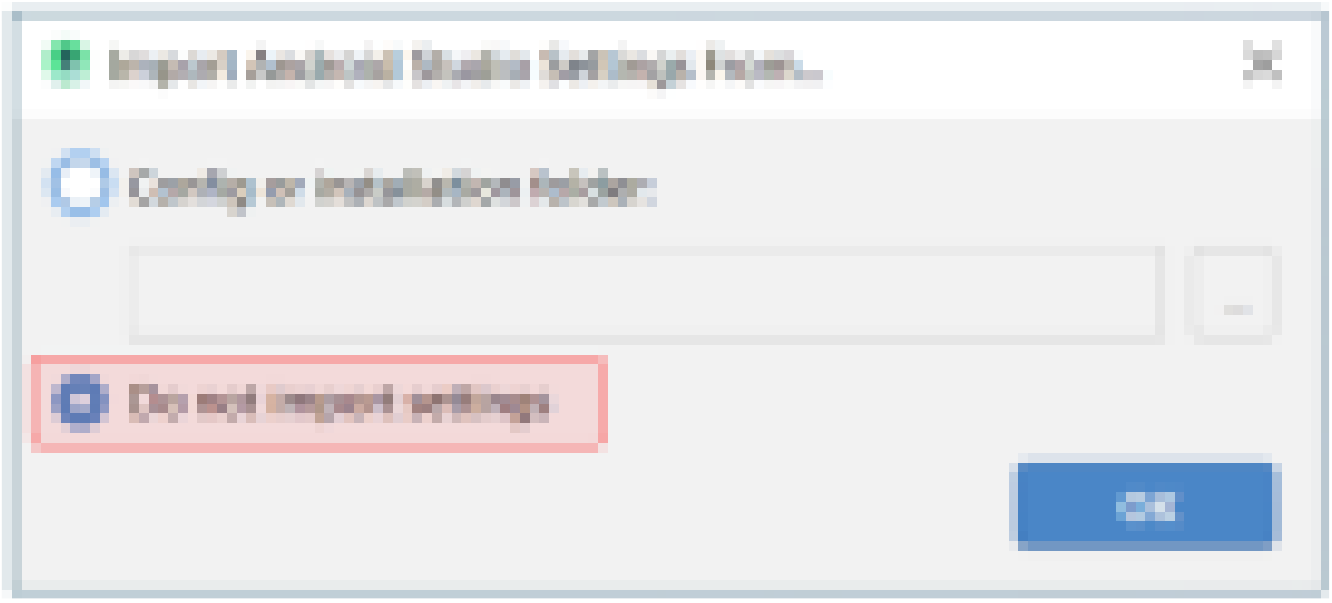

Para el resto de pasos de este asistente inicial, se deben aceptar sin modificar ninguna opción por defecto, hasta llegar al último paso donde se selecciona la opción "Start Android Studio" y se pulsa el botón "Finish". De esta manera, la aplicación se iniciará automáticamente.

**Figura 14.** Asistente de Instalación Iniciar Android Studio




f) Seleccionar: no importar configuraciones

**Figura 15.** Asistente de Instalación Importar

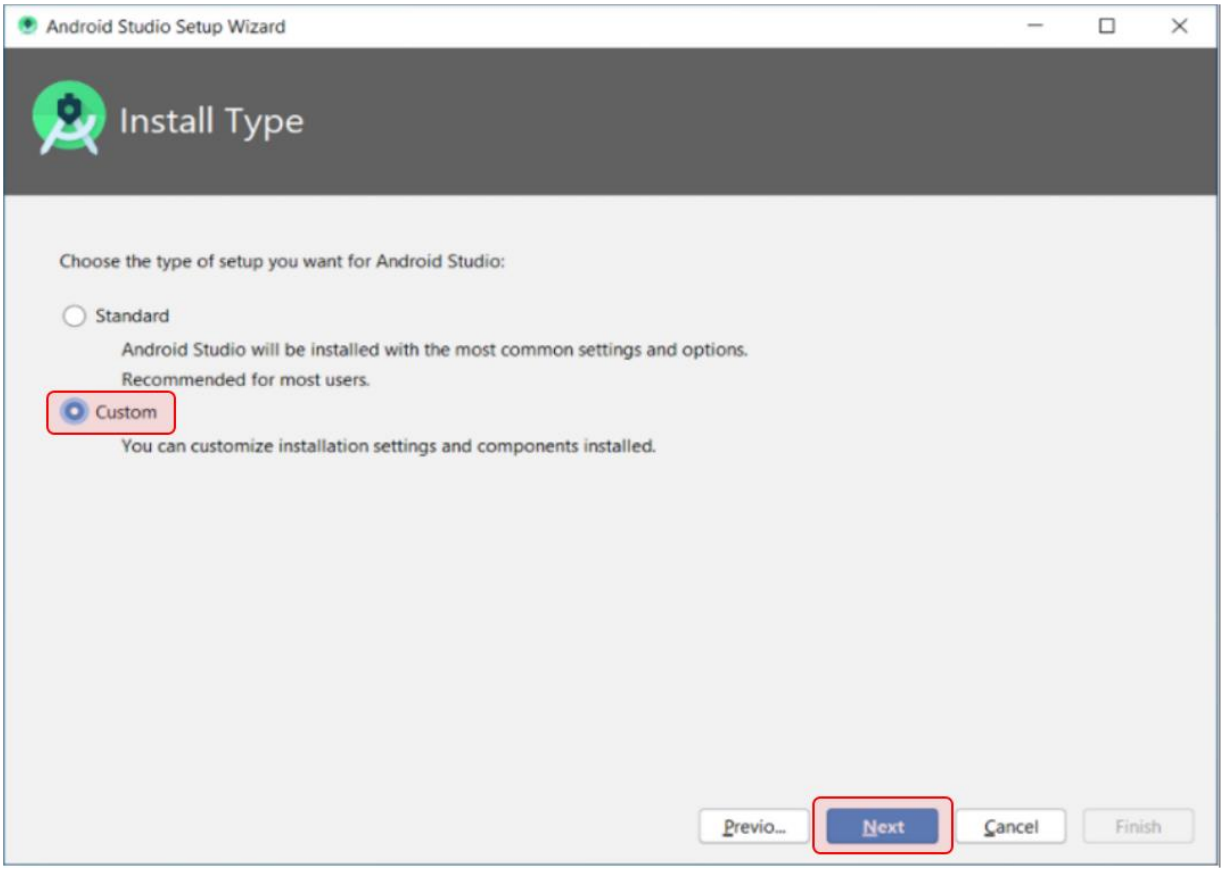


g) Instalación de Wizard Android Studio

**Figura 16.** Asistente de Instalación "Wizard Android Studio"

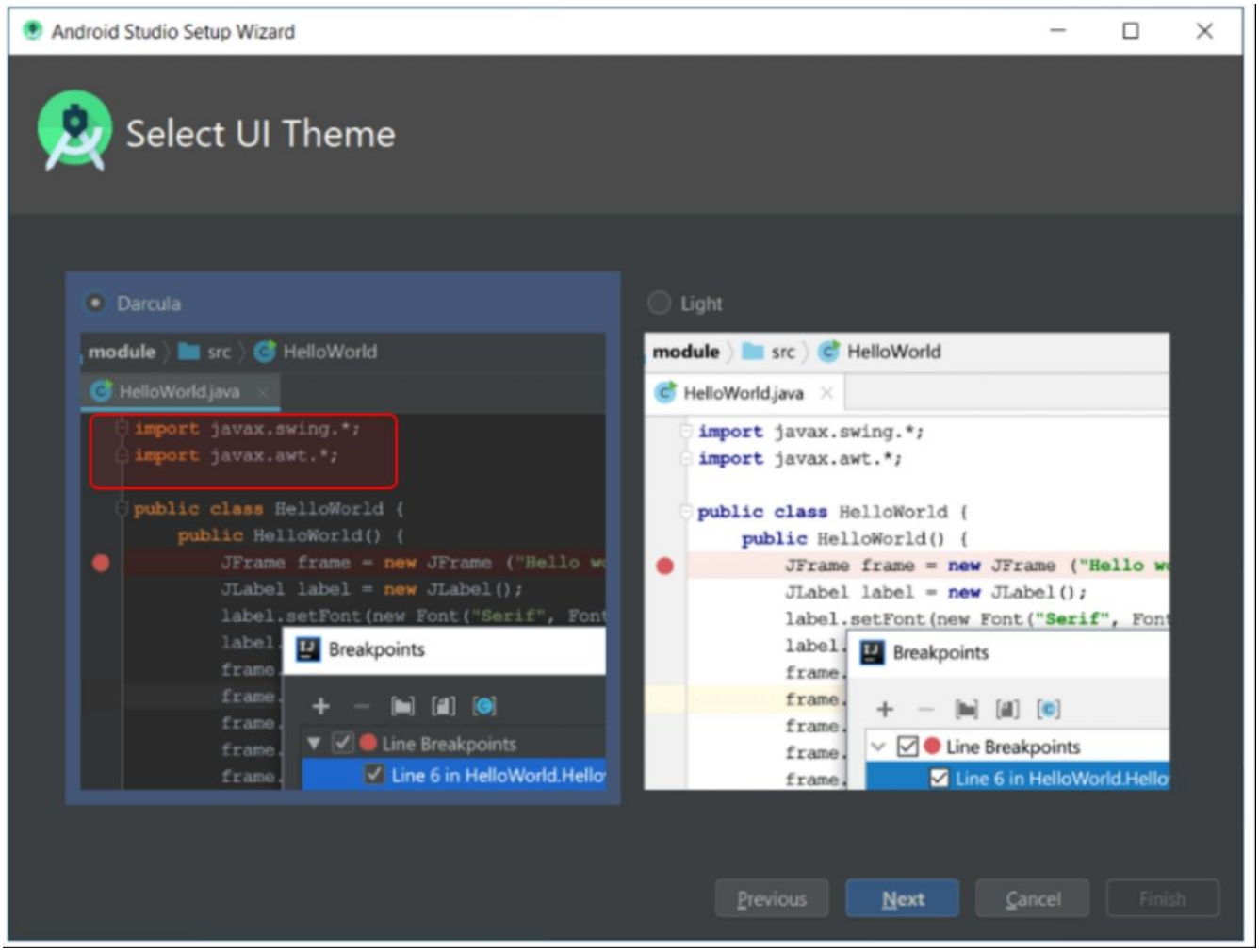
h) Clic en Next

**Figura 17.** Selección tipo configuración



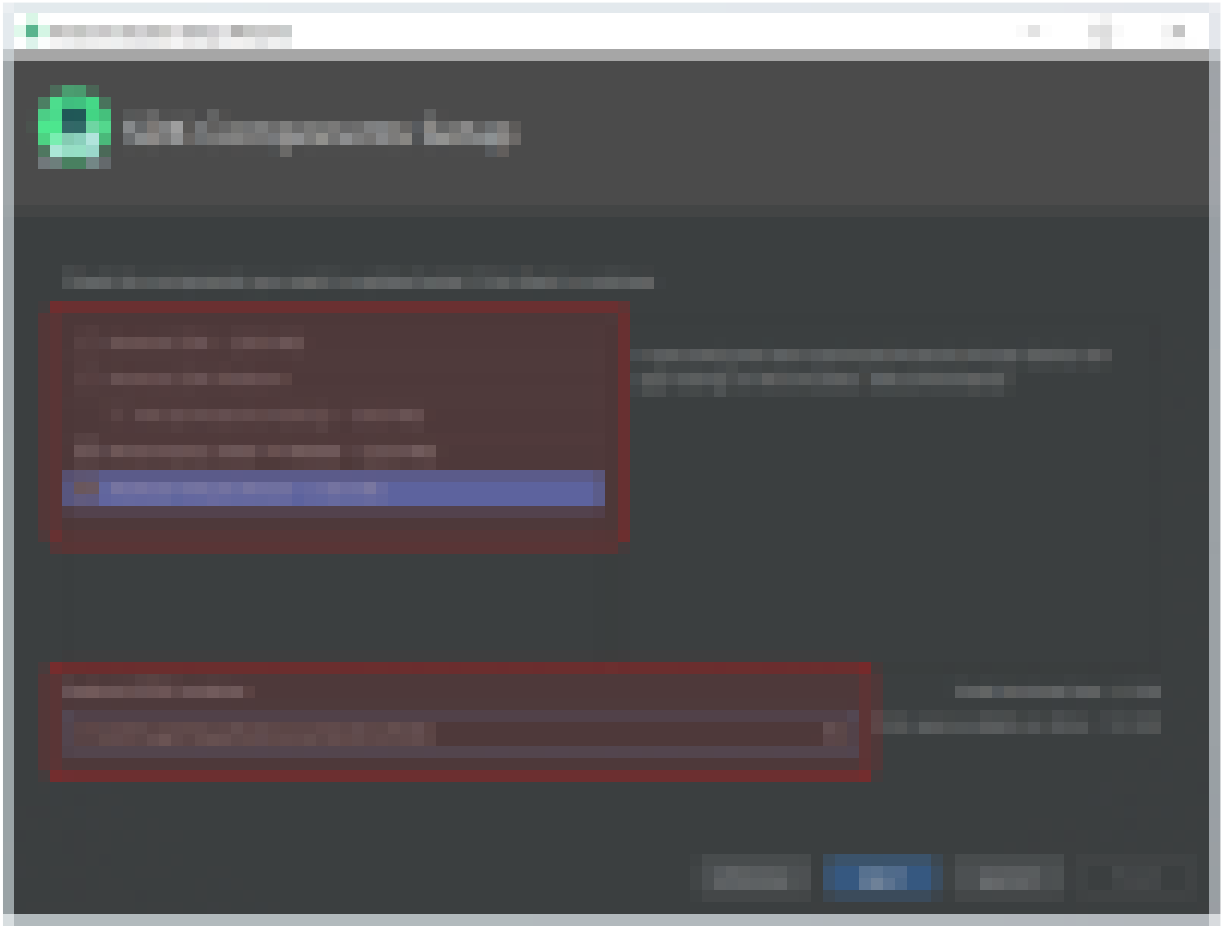

i) Para el tipo de configuración marcar la opción Custom (Personalizado) y clic en Next

**Figura 18.** Selección tema visual



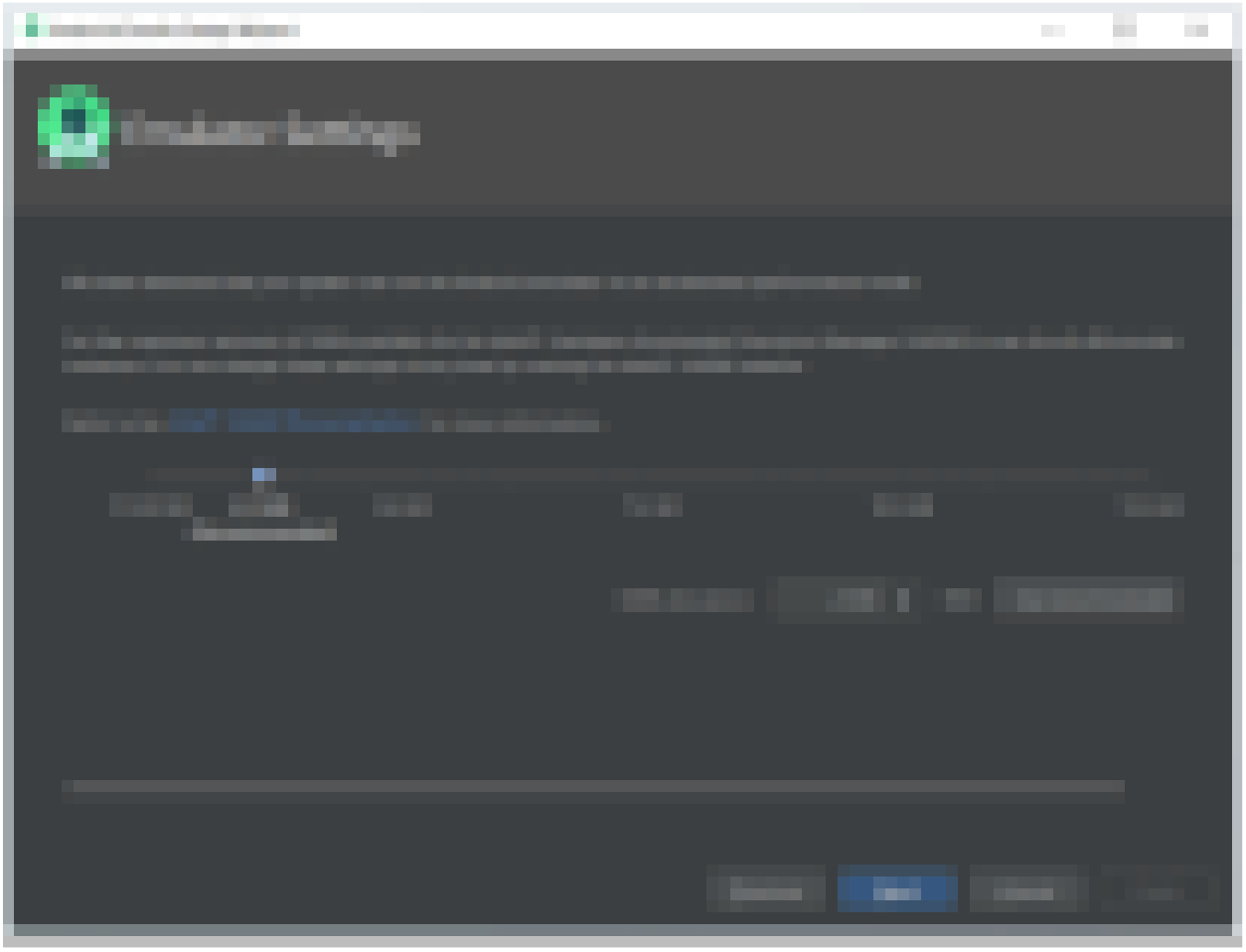

j) En la siguiente pantalla del asistente marcar los componentes adicionales a instalar. En el campo Android SDK Location se indica la ruta donde se desea instalar el SDK de Android.

**Figura 19.** Configuración de componentes



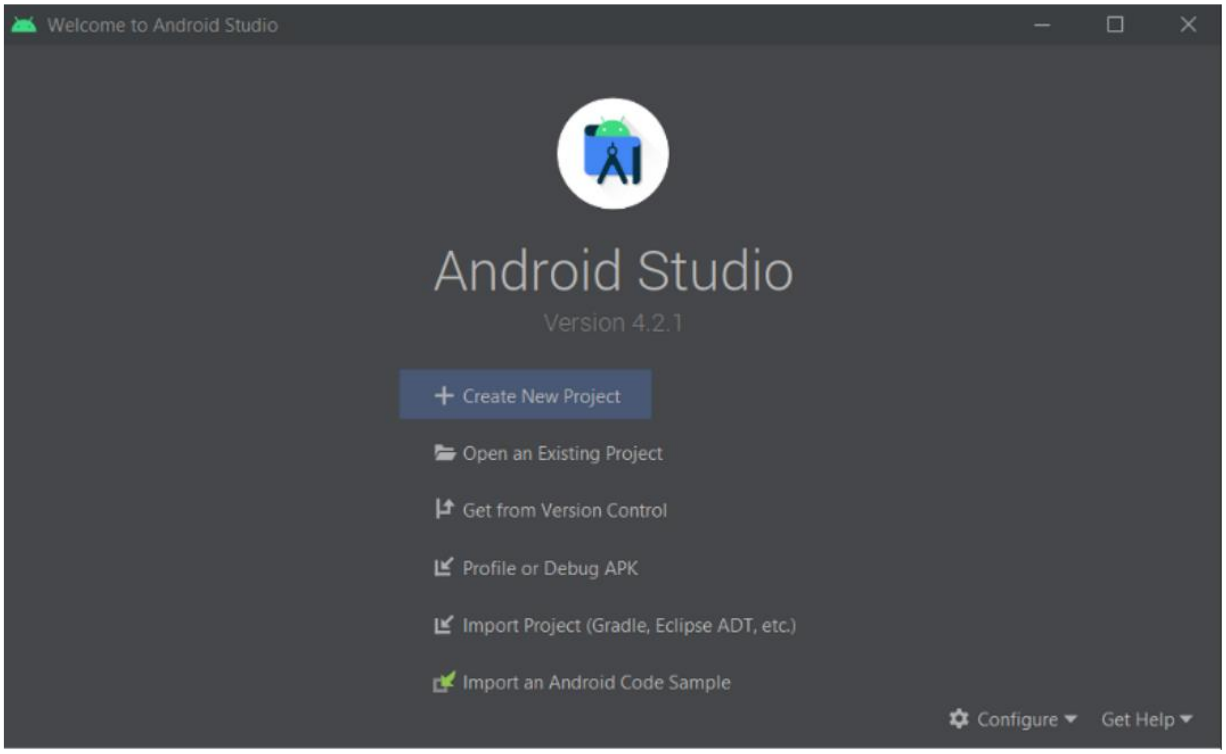
k) Indicar la cantidad de memoria que se utilizará para el emulador y clic en "Next".

**Figura 20.** Configuración del Emulador



l) Tras finalizar el asistente de inicio aparecerá la pantalla de bienvenida de Android Studio para este ejemplo en su versión 4.2.1

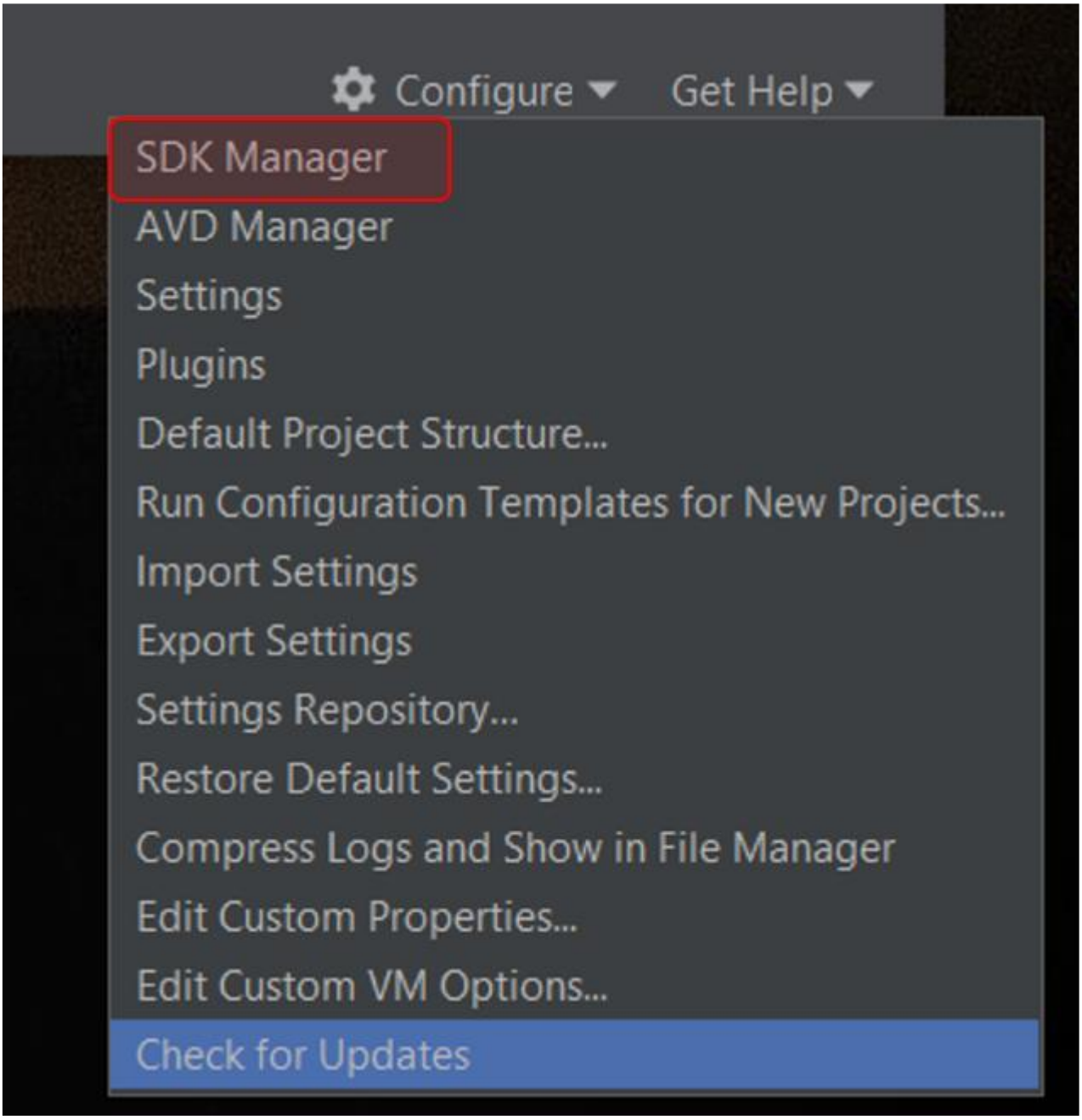
**Figura 21.** Bienvenida Android Studio




## SDK Manager

El SDK Manager permite gestionar las versiones de Android que están instaladas, así como otras herramientas necesarias para el desarrollo de aplicaciones móviles. El siguiente paso será revisar los componentes que se han instalado del SDK de Android Studio e instalar o actualizar componentes adicionales si fuera necesario para el desarrollo de las aplicaciones. En la pantalla de inicio de Android Studio, seleccione la opción "Configure" y haga clic en SDK Manager.

**Figura 22.** SDK Manager



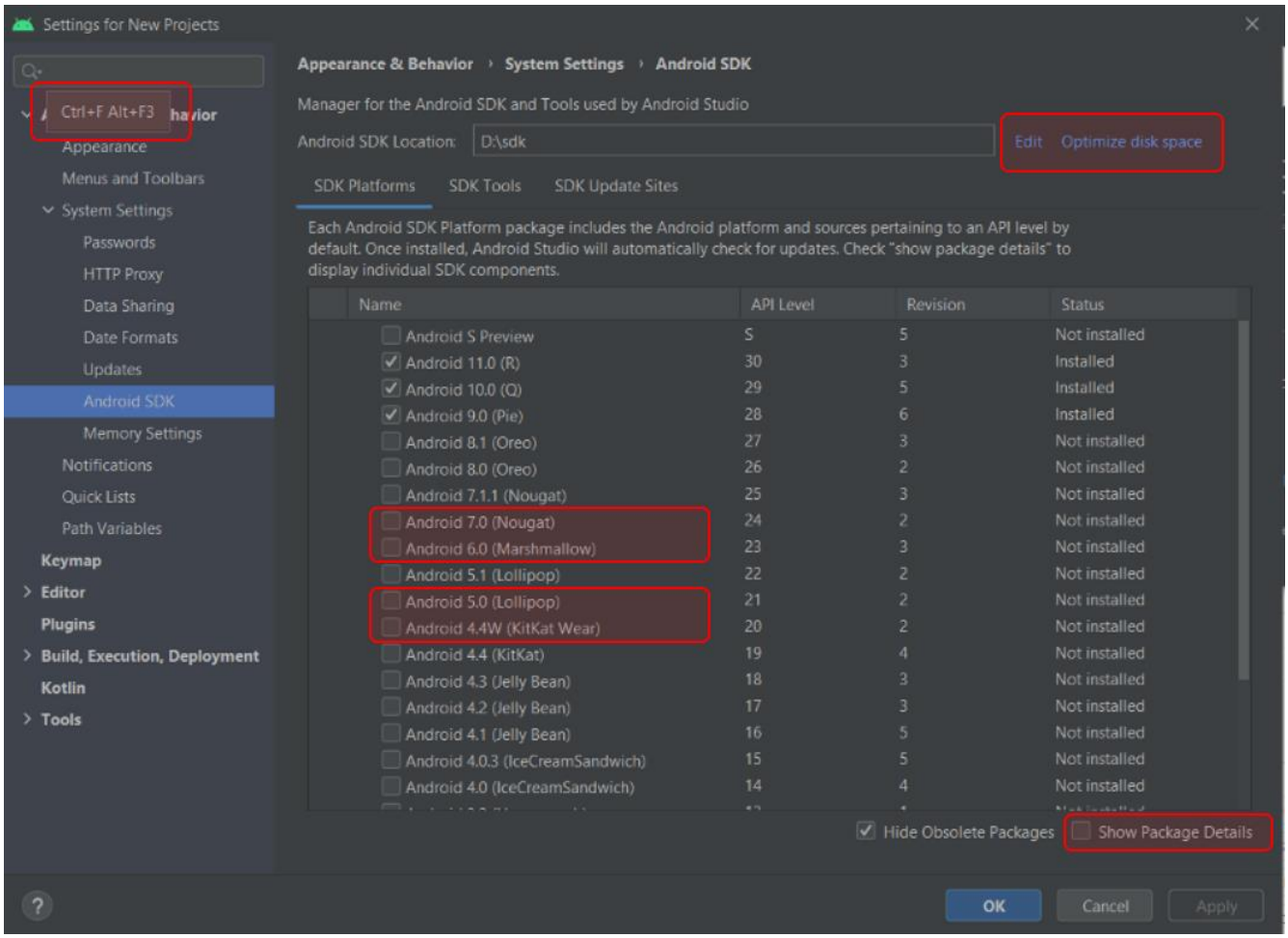

## SDK Platforms

SDK Platforms permite seleccionar los componentes y librerías necesarios para desarrollar sobre cada una de las versiones de Android. Por ejemplo, si se desea probar

la aplicación en dispositivos con Android 8 y Android 10, se deben descargar las plataformas correspondientes a las versiones 26 y 29, respectivamente. Para ver los subcomponentes de cada plataforma, seleccione la opción "Show Package Details" situada en la parte inferior de la ventana. Para cada versión de Android instalada, se deben tener al menos los siguientes dos elementos:

- Android SDK Platform.
- Google APIs.
- Intel x86 Atom System Image.

**Figura 23.** SDK Platforms



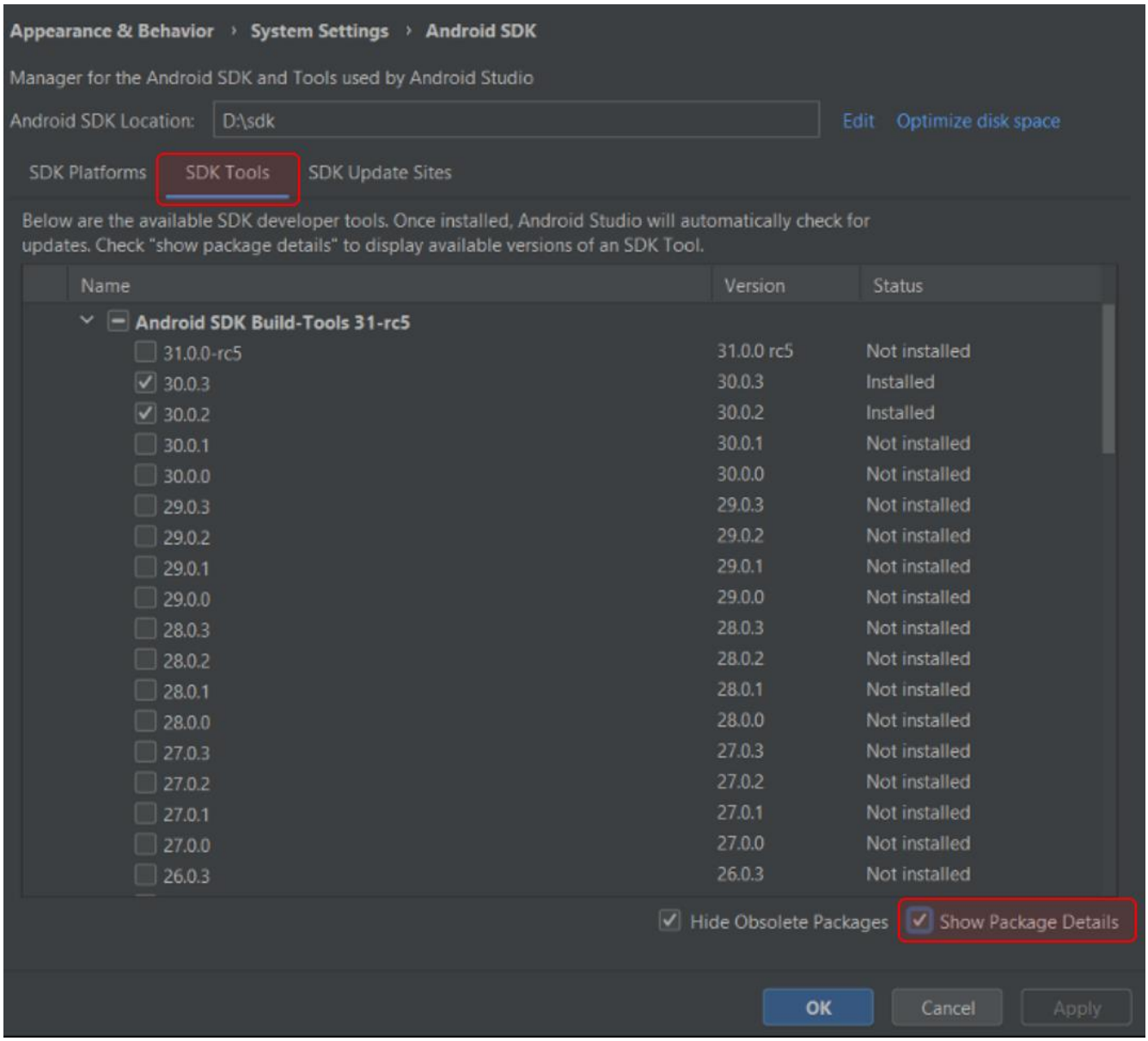
## SDK Tools

SDK Tools son componentes para el SDK que incluyen herramientas para el desarrollo y la depuración. Los indispensables por el momento, que ya deberían aparecer instalados por defecto, son los siguientes:

- Android SDK Build-Tools
- Android SDK Platform-Tools
- Android Emulator

Con estos pasos, ya se tendría configurado Android Studio para dar inicio al desarrollo de las aplicaciones.

**Figura 24.** SDK Tools



## 5. Componentes de una aplicación Android

Los componentes de la aplicación son mecanismos de creación básicos de una aplicación para Android. Cada componente es un punto de entrada por el que el sistema o un usuario ingresan a una aplicación. Algunos componentes dependen de otros.

- **Actividades**

  Representan el componente principal de la interfaz gráfica de una aplicación Android, correspondiendo cada una a una pantalla individual con una interfaz de usuario. Ejemplo: Una aplicación de fotos puede tener una actividad para mostrar opciones de edición y otra para realizar la edición.

- **Servicios**

  Componentes sin interfaz gráfica que se ejecutan en segundo plano. Son similares a los servicios de otros sistemas operativos y pueden realizar diversas acciones, como actualizar datos o lanzar notificaciones. En ciertos casos, pueden mostrar elementos visuales para interactuar con el usuario.

- **Proveedores de contenido**

  Mecanismos en Android para compartir datos entre aplicaciones. Permiten compartir datos sin revelar detalles sobre su almacenamiento interno o estructura. También facilitan el acceso a los datos de una aplicación a través de los proveedores de contenido definidos por otras aplicaciones.

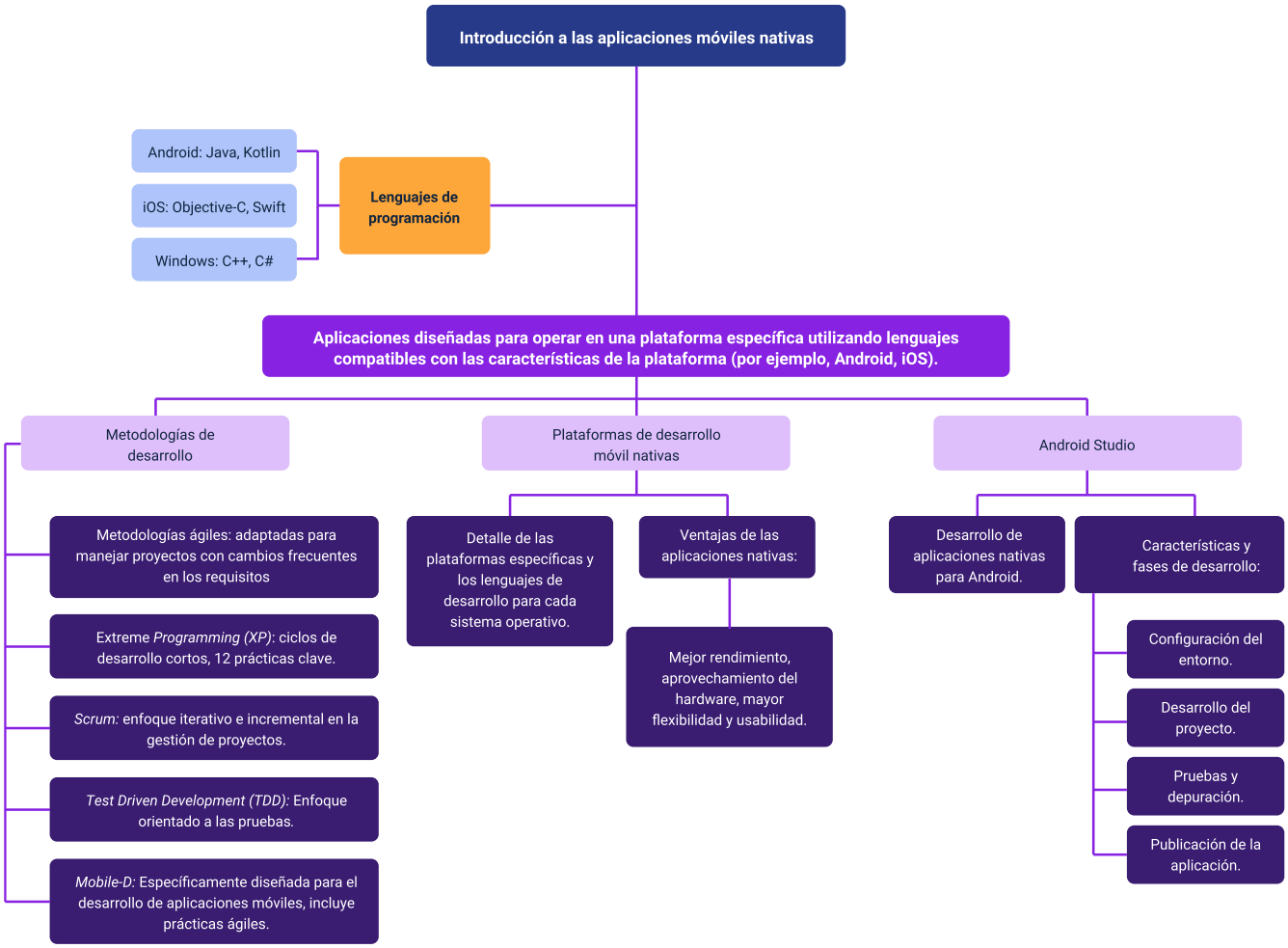
- **Receptores de emisiones**

  Componentes que detectan y reaccionan a mensajes o eventos globales generados por el sistema o por otras aplicaciones. Ejemplos incluyen la

detección de batería baja, recepción de SMS, o inserción de una tarjeta SD. Estos mensajes son broadcast, es decir, no dirigidos a una aplicación específica.

# Síntesis

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo.



## Material complementario

| Tema | Referencia | Tipo de material | Enlace del recurso |
|---|---|---|---|
| Metodología XP | Perú, M. H. (2019, 15 de diciembre). ¿Qué es la programación extrema? Metodología Ágil XP (Ciclo de Vida XP, Prácticas Básicas de XP) [Video]. YouTube. | Video YouTube | https://www.youtube.com/watch?v=tCl33R9jHBk |
| Metodología Scrum | Henao, C. (2018, 27 de junio). #3. SCRUM en 6 minutos Metodologías Ágiles [Video]. YouTube. | Video YouTube | https://www.youtube.com/watch?v=HhC75IonpOU |
| Mobile-D | Montoya, J. L. (2021, 15 de abril). Exposición metodologías de desarrollo de software. | Mobile-D | https://www.youtube.com/watch?v=I3yMHHgQCEk |
| Entornos de Desarrollo | Peñalba, I. (2021). Hacer aplicaciones para Android: ¿Qué es un entorno de desarrollo y cuál es el mejor? El Español. | Página | https://www.elespanol.com/elandroidelibre/20200518/hacer-aplicaciones-android-entorno-desarrollo-mejor/490952339_0.html |
| Visual Studio | Microsoft Build. (2023). Información general sobre Visual Studio. | Página | https://docs.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2019 |
| Android | Developers. (2020, 7 de mayo). Arquitectura de la plataforma. | Página Oficial | https://developer.android.com/guide/platform?hl=es-419 |
| Versiones Android | ADSLZone. (2021, 3 de marzo). Qué es Android. | Página | https://www.adslzone.net/reportajes/software/que-es-android |

| Tema | Referencia | Tipo de material | Enlace del recurso |
|---|---|---|---|
| Versiones Android Studio | Developer. (2021, 14 de junio). Archivos de descarga de Android Studio. | Página | https://developer.android.com/studio/archive?hl=es-419 |
| Instalación Android | Developer. (2020, 21 de diciembre). Cómo instalar Android Studio. | Página Oficial | https://developer.android.com/studio/install?hl=es-419 |

## Glosario

**Dalvik**: máquina virtual de Android.

**Manifiesto ágil**: es un documento redactado con principios para mejorar la forma de desarrollar.

**Wereable**: dispositivo conectado que se puede llevar puesto y que se conecta al teléfono móvil.

## Referencias bibliográficas


Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., & Salo, O. (2004). Mobile-D: An agile approach for mobile application development. En Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, 174-175. ACM.

Astel, D. (2003). Test-driven development: A practical guide. Prentice Hall PTR.

Avison, D. E., & Fitzgerald, G. (2006). Information system development. McGraw-Hill Education.

Balafuera, Y. D. A. (2015). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles: Estado actual. Revista de tecnología, 12(2). https://doi.org/10.18270/rt.v12i2.1291

Letelier, P., Canós, J. H., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Presentado en VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD), Alicante, España, 1-8. Addison Wesley.

Olmos, F. (2020, 19 de octubre). El origen de Android: Características y costos. Fredy Olmos. https://fredyolmos.com/ciencia-y-tecnologia/el-origen-de-android-costos-caracteristicas/

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Harvard Business Review.

# Créditos

| Nombre | Cargo | Centro de Formación y Regional |
|---|---|---|
| Milady Tatiana Villamil Castellanos | Responsable del Ecosistema | Dirección General |
| Olga Constanza Bermúdez Jaimes | Responsable de Línea de Producción | Centro de Servicios de Salud - Regional Antioquia |
| Zulema Yidney León Escobar | Experta Temática | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Paola Alexandra Moya Peralta | Evaluadora Instruccional | Centro de Servicios de Salud - Regional Antioquia |
| Andrés Felipe Herrera Roldán | Diseñador de Contenidos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Edwin Sneider Velandia Suárez | Desarrollador Fullstack | Centro de Servicios de Salud - Regional Antioquia |
| Edgar Mauricio Cortés García | Actividad Didáctica | Centro de Servicios de Salud - Regional Antioquia |
| Jaime Hernán Tejada Llano | Validador de Recursos Educativos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Margarita Marcela Medrano Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |
| Daniel Ricardo Mutis Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |